Provable Robustness of ReLU networks via Maximization of Linear Regions

Francesco Croce*,[1] **Maksym Andriushchenko*,[2]** **Matthias Hein[1]**
[1]University of Tübingen, Germany [2]Saarland University, Germany

Abstract

It has been shown that neural network classifiers are not robust. This raises concerns about their usage in safety-critical systems. We propose in this paper a regularization scheme for ReLU networks which provably improves the robustness of the classifier by maximizing the linear region of the classifier as well as the distance to the decision boundary. Our techniques allow even to find the minimal adversarial perturbation for a fraction of test points for large networks. In the experiments we show that our approach improves upon adversarial training both in terms of lower and upper bounds on the robustness and is comparable or better than the state of the art in terms of test error and robustness.

1 Introduction

In recent years it has been highlighted that state-of-the-art neural networks are highly non-robust: small changes to an input image, which are almost non-perceivable for humans, change the classifier decision and the wrong decision has even high confidence [26, 11]. This calls into question the use of neural networks in safety-critical systems e.g. medical diagnosis systems or self-driving cars and opens up possibilities to actively attack an ML system in an adversarial way [21, 17, 16, 17]. Moreover, this non-robustness has also implications on follow-up processes like interpretability. How should we be able to interpret classifier decisions if very small changes of the input lead to different decisions?

The finding of the non-robustness initiated a competition where on the one hand increasingly more sophisticated attack procedures were proposed [11, 14, 23, 6] and on the other hand research was focused to develop stronger defenses against these attacks [12, 11, 33, 21, 14, 3, 18]. In the end it turned out that for many proposed defenses there exists still a way to attack the classifier successfully [5, 2, 20], with the notable exception of [18] which was shown to be empirically robust wrt to the l_∞-norm. Thus considering the high importance of robustness in safety-critical machine learning applications, we need robustness guarantees, where one can provide for each test point the radius of a ball on which the classifier will not change the decision and thus no attack whatsoever will be able to create an adversarial example inside this ball.
Therefore recent research has focused on such provable guarantees of a classifier with respect to the l_1-norm [4], l_2-norm [13] and l_∞-norm [3, 15, 22, 27, 29, 28]. Some works try to solve the combinatorial problem of computing for each test instance the norm of the minimal perturbation necessary to change the decision [4, 15, 27]. Unfortunately, these approaches do not scale with normal training to large networks. Another line of research thus focuses on lower bounds on the norm of the minimal perturbation necessary to change the decision [13, 22, 29, 28].

Moreover, in recent years several new ways to regularize neural networks [24, 7] or new forms of losses [10] have been proposed with the idea of enforcing a large margin, that is a large distance between training instances and decision boundaries. However, these papers do not directly optimize a robustness guarantee. In spirit our paper is closest to [13, 22, 29, 19]. All of them are aiming at providing robustness guarantees and at the same time they propose a new way how one can optimize the robustness guarantee during training. Currently, up to our knowledge only [29] can optimize robustness wrt to multiple p-norms, whereas [13] is restricted to l_2 and [22, 19] to l_∞.

In this paper we propose a regularization scheme for the class of ReLU networks (feedforward networks with ReLU activation functions including convolutional and residual architectures with max- or sum-pooling layers)

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

*Equal contribution.

which provably increases the robustness of classifiers. We use the fact that these networks lead to continuous piecewise affine classifier functions and show how to get either the optimal minimal perturbation or a lower bound using the properties of the linear region in which the point lies. As a result of this analysis we propose a new regularization scheme which directly maximizes the lower bound on the robustness guarantee. This allows us to get classifiers with good test error and good robustness guarantees at the same time. While we focus on robustness with respect to l_2- and l_∞-distance, our approach applies to all l_p-norms. Finally, we show in experiments on four data sets that our approach improves lower bounds as well as upper bounds on the robust test error and can be integrated with adversarial training [11, 18]. Interestingly, for our models the combinatorial solver of [27] leads often to tight lower and upper bounds on the robust test error whereas it fails for normal or adversarial training.

2 Local properties of ReLU networks

Feedforward neural networks which use piecewise affine activation functions (e.g. ReLU, leaky ReLU) and are linear in the output layer can be rewritten as continuous piecewise affine functions [1, 8].

Definition 2.1. *A function $f : \mathbb{R}^d \to \mathbb{R}$ is called* piecewise affine *if there exists a finite set of polytopes $\{Q_r\}_{r=1}^M$ (referred to as* linear regions *of f) such that $\cup_{r=1}^M Q_r = \mathbb{R}^d$ and f is an affine function when restricted to every Q_r.*

In the following we introduce the notation required for the explicit description of the linear regions and decision boundaries of a multi-class ReLU classifier $f : \mathbb{R}^d \to \mathbb{R}^K$ (where d is the input space dimension and K the number of classes) which has no activation function in the output layer. The decision of f at a point x is given by $\underset{r=1,\ldots,K}{\arg\max}\, f_r(x)$. The discrimination between linear regions and decision boundaries allows us to share the linear regions across classifier components.

We denote by $\sigma : \mathbb{R} \to \mathbb{R}$, $\sigma(t) = \max\{0, t\}$, the ReLU activation function, $L+1$ is the number of layers and $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $b^{(l)} \in \mathbb{R}^{n_l}$ respectively are the weights and offset vectors of layer l, for $l = 1, \ldots, L+1$ and $n_0 = d$. If $x \in \mathbb{R}^d$ and $g^{(0)}(x) = x$ we define recursively the pre- and post-activation output of every layer as

$$f^{(k)}(x) = W^{(k)} g^{(k-1)}(x) + b^{(k)}, \quad \text{and}$$
$$g^{(k)}(x) = \sigma(f^{(k)}(x)), \quad k = 1, \ldots, L,$$

so that the resulting classifier is obtained as $f^{(L+1)}(x) = W^{(L+1)} g^{(L)}(x) + b^{(L+1)}$.

We derive, following [8, 9], the description of the polytope $Q(x)$ in which x lies and the resulting affine function when f is restricted to $Q(x)$. We assume for this that x does not lie on the boundary between two polytopes (which is almost always true as the faces shared by two or more polytopes have dimension strictly smaller than d). Let $\Delta^{(l)}, \Sigma^{(l)} \in \mathbb{R}^{n_l \times n_l}$ for $l = 1, \ldots, L$ be diagonal matrices defined elementwise as

$$\Delta^{(l)}(x)_{ij} = \begin{cases} \operatorname{sign}(f_i^{(l)}(x)) & \text{if } i = j, \\ 0 & \text{else.} \end{cases},$$
$$\Sigma^{(l)}(x)_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } f_i^{(l)}(x) > 0, \\ 0 & \text{else.} \end{cases}.$$

This allows us to write $f^{(k)}(x)$ as composition of affine functions, that is

$$\begin{aligned} f^{(k)}(x) =& W^{(k)} \Sigma^{(k-1)}(x) \Big(W^{(k-1)} \Sigma^{(k-2)}(x) \\ & \times \Big(\ldots \Big(W^{(1)} x + b^{(1)} \Big) \ldots \Big) + b^{(k-1)} \Big) + b^{(k)}, \end{aligned}$$

We can further simplify the previous expression as $f^{(k)}(x) = V^{(k)} x + a^{(k)}$, with $V^{(k)} \in \mathbb{R}^{n_k \times d}$ and $a^{(k)} \in \mathbb{R}^{n_k}$ given by

$$V^{(k)} = W^{(k)} \Big(\prod_{l=1}^{k-1} \Sigma^{(k-l)}(x) W^{(k-l)} \Big) \quad \text{and}$$

$$a^{(k)} = b^{(k)} + \sum_{l=1}^{k-1} \Big(\prod_{m=1}^{k-l} W^{(k+1-m)} \Sigma^{(k-m)}(x) \Big) b^{(l)}.$$

Note that a forward pass through the network is sufficient to compute $V^{(k)}$ and $b^{(k)}$ for every k, which results in only a small overhead compared to the usual effort necessary to compute the output of f at x. We are then able to characterize the polytope $Q(x)$ as intersection of $N = \sum_{l=1}^{L} n_l$ half spaces given by

$$\Gamma_{l,i} = \big\{ z \in \mathbb{R}^d \,\big|\, \Delta^{(l)}(x) \big(V_i^{(l)} z + a_i^{(l)} \big) \geq 0 \big\},$$

for $l = 1, \ldots, L$, $i = 1, \ldots, n_l$, namely

$$Q(x) = \bigcap_{l=1,\ldots,L} \bigcap_{i=1,\ldots,n_l} \Gamma_{l,i}.$$

Note that N is also the number of hidden units of the network. Finally, we can write

$$f^{(L+1)}(z) \Big|_{Q(x)} = V^{(L+1)} z + a^{(L+1)},$$

which represents the affine restriction of f to $Q(x)$. One can further distinguish the subset $Q_c(x)$ of $Q(x)$ assigned to a specific class c, among the K available ones, which is given by

$$\begin{aligned} Q_c(x) = \bigcap_{\substack{s=1,\ldots,K \\ s \neq c}} & \big\{ z \in \mathbb{R}^d \,\big|\, \Big\langle V_c^{(L+1)} - V_s^{(L+1)}, z \Big\rangle \\ & + a_c^{(L+1)} - a_s^{(L+1)} \geq 0 \big\} \cap Q(x), \end{aligned}$$

where $V_r^{(L+1)}$ is the r-th row of $V^{(L+1)}$. The set $Q_c(x)$ is again a polytope as it is an intersection of polytopes and it holds $Q(x) = \bigcup_{c=1,\ldots,K} Q_c(x)$. We refer to [9] how to integrate other operations/layer types e.g. max-pooling, residual and dense nets or other piecewise linear activation functions like leaky ReLU.

3 Robustness guarantees for ReLU networks

In the following we first define the problem of the minimal adversarial perturbation and then derive robustness guarantees for ReLU networks. We call a decision of a classifier f robust at x if small changes of the input do not alter the decision. Formally, this can be described as optimization problem (1) [26]. If the classifier outputs class c for input x, assuming a unique decision, the *robustness* of f at x is given by

$$\min_{\delta \in \mathbb{R}^d} \left\|\delta\right\|_p, \quad \text{s.th.} \quad \max_{l \neq c} f_l(x+\delta) \geq f_c(x+\delta) \qquad (1)$$
$$x + \delta \in C,$$

where C is a constraint set which the generated point $x+\delta$ has to satisfy, e.g., an image has to be in $[0,1]^d$. The complexity of the optimization problem (1) depends on the classifier f, but it is typically non-convex, see [15] for a hardness result for neural networks.
For standard neural networks $\left\|\delta\right\|_p$ is very small for *almost any* input x of the data generating distribution, which questions the use of such classifiers in safety-critical systems. The solutions of (1), $x+\delta$, are called *adversarial samples.*

For a linear classifier, $f(x) = Wx + b$, with $C = \mathbb{R}^d$ one can compute the solution of (1) in closed form [14]

$$\left\|\delta\right\|_p = \min_{s \neq c} \frac{\left|\left\langle w_c - w_s, x\right\rangle + b_c - b_s\right|}{\left\|w_c - w_s\right\|_q},$$

where $\left\|\cdot\right\|_q$ is the dual norm of $\left\|\cdot\right\|_p$, that is $\frac{1}{p}+\frac{1}{q}=1$. In [13] it has been shown that box constraints $C=[a,b]^d$ can be integrated for linear classifiers which results in simple convex optimization problems. In the following we use the intuition from linear classifiers and the particular structures derived in Section 2 to come up with robustness guarantees, that means lower bounds on the optimal solution of (1), for ReLU networks. Moreover, we show that it is possible to compute the minimal perturbation for some of the inputs x even though the general problem is NP-hard [15].

Let us start analyzing how we can solve efficiently problem (1) inside each linear region $Q(x)$. We first need the definition of two important quantities:

Lemma 3.1. *The l_p-distance $d_B(x) = \min_{z \in \partial Q(x)} \left\|z - x\right\|_p$ of x to the boundary of the polytope $Q(x)$ is given by*

$$d_B(x) = \min_{l=1,\ldots,L} \min_{j=1,\ldots,n_l} \frac{\left|\left\langle V_j^{(l)}, x\right\rangle + a_j^{(l)}\right|}{\left\|V_j^{(l)}\right\|_q},$$

where $V_j^{(l)}$ is the j-th row of $V^{(l)}$ and $\left\|\cdot\right\|_q$ is the dual norm of $\left\|\cdot\right\|_p$ ($\frac{1}{p}+\frac{1}{q}=1$).

For the decision boundaries in $Q(x)$, with $c = \operatorname*{arg\,max}_{r=1,\ldots,K} f_r^{(L+1)}(x)$, we define the quantity $d_D(x)$ as

$$\min_{\substack{s=1,\ldots,K \\ s \neq c}} \frac{\left|\left\langle V_c^{(L+1)} - V_s^{(L+1)}, x\right\rangle + a_c^{(L+1)} - a_s^{(L+1)}\right|}{\left\|V_c^{(L+1)} - V_s^{(L+1)}\right\|_q}.$$

Lemma 3.2. *If $d_D(x) \leq d_B(x)$, then $d_D(x)$ is the minimal l_p-distance of x to the decision boundary of the ReLU network $f^{(L+1)}$.*

Proof. First we note that $d_D(x)$ is the distance of x to the decision boundary for the linear multi-class classifier $g(x) = V^{(L+1)}x + a^{(L+1)}$ which is equal to $f^{(L+1)}$ on $Q(x)$. If $d_D(x) \leq d_B(x)$ then the point realizing the minimal distance to the decision boundary $d_D(x)$ is inside $Q(x)$ and as $d_D(x) < d_B(x)$ there cannot exist another point on the decision boundary of $f^{(L+1)}$ outside of $Q(x)$ having a smaller l_p-distance to x. Thus $d_D(x)$ is the minimal l_p-distance of x to the decision boundary of $f^{(L+1)}$. □

The next theorem combines both results to give lower bounds resp. the solution to the optimization problem of the minimal adversarial perturbation in (1).

Theorem 3.1. *We get the following robustness guarantees:*

1. *If $d_B(x) \leq d_D(x)$, then $d_B(x)$ is a lower bound on the minimal l_p-norm of the perturbation necessary to change the class (optimal solution of (1)).*
2. *If $d_D(x) \leq d_B(x)$, then $d_D(x)$ is equal to the minimal l_p-norm necessary to change the class (optimal solution of optimization problem (1)).*

Proof. If $d_B(x) \leq d_D(x)$, then the ReLU classifier does not change on $B_p(x, d_B(x))$ and thus $d_B(x)$ is a lower bound on the minimal l_p-norm perturbation necessary to change the class. The second statement follows directly by Lemma 3.2. □

In Figure 1 we illustrate the different cases for $p=2$. On the left hand side $d_B(x) < d_D(x)$ and thus we get that on the ball $B_2(x, d_B(x))$ the decision does not

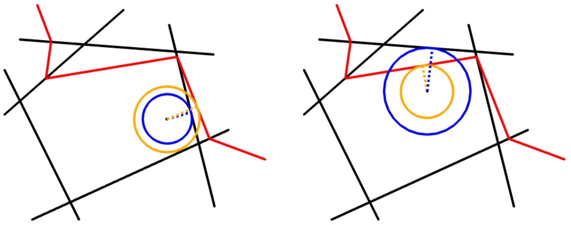

Figure 1: Left: the input x is closer to the boundary of the polytope $Q(x)$ (black) than to the decision boundary (red). In this case the smallest perturbation that leads to a change of the decision lies outside the linear region $Q(x)$. **Right**: the input x is closer to the decision boundary than to the boundary of $Q(x)$, so that the projection of the point onto the decision hyperplane provides the adversarial example with smallest norm.

change, whereas in the rightmost plot we have $d_D(x) < d_B(x)$ and thus we obtain the minimum distance to the decision boundary. Using Theorem 3.1 we can provide robustness guarantees for every point and for some even compute the optimal robustness guarantees. Finally, we describe in the Appendix how one can improve the lower bounds by checking neighboring regions of $Q(x)$. Compared to the bounds of [29, 28] ours are slightly worse (see Appendix). However, our bounds can be directly maximized and have a clear geometrical meaning and thus motivate directly a regularization scheme for ReLU networks which we propose in the next section.

4 Large margin and region boundary regularization

Using the results of Section 3, a classifier with guaranteed robustness requires large distance to the boundaries of the linear region as well as to the decision boundary. Even the optimal guarantee (solution of (1)) can be obtained in some cases. Unfortunately, as illustrated in Figures 2a and 2c for simple one hidden layer networks, the linear regions $Q(x)$ are small for networks trained without regularization and thus no meaningful guarantees can be obtained. Thus we propose a new regularization scheme which simultaneously aims for each training point to achieve large distance to the boundary of the linear region it lies in, as well as to the decision boundary. Using Theorem 3.1 this directly leads to good robustness guarantees of the resulting classifier.

However, note that just maximizing the distance to the decision boundary might be misleading during training as this does not discriminate between points which are correctly (correct side of the decision hyperplane) or wrongly classified (wrong side of the decision hyperplane). Thus we introduce the signed version of $d_D(x)$, where y is the true label of the point x,

$$\overline{d_D}(x) = \min_{\substack{s=1,\dots,K \\ s\neq y}} \frac{f_y^{(L+1)}(x) - f_s^{(L+1)}(x)}{\left\| V_y^{(L+1)} - V_s^{(L+1)} \right\|_q}. \qquad (2)$$

Please note that if $\overline{d_D}(x) \geq 0$, then x is correctly classified, whereas $\overline{d_D}(x) < 0$ if x is wrongly classified. If $|\overline{d_D}(x)| \leq d_B(x)$, then it follows from Lemma 3.2 that $|\overline{d_D}(x)|$ is the distance to the decision hyperplane. If $|\overline{d_D}(x)| > d_B(x)$ this does not need to be any longer true, but $|\overline{d_D}(x)|$ is at least a good proxy as $\left\| V_y^{(L+1)} - V_s^{(L+1)} \right\|_q$ is an estimate of the local cross Lipschitz constant [13]. Finally, we propose to use the following regularization scheme:

Definition 4.1. *Let x be a point of the training set and define the Maximum Margin Regularizer (MMR) for x, for some $\gamma_B, \gamma_D \in \mathbb{R}_{++}$, as*

$$MMR(x) = \max\left(0, 1 - \frac{d_B(x)}{\gamma_B}\right) + \max\left(0, 1 - \frac{\overline{d_D}(x)}{\gamma_D}\right). \qquad (3)$$

The MMR penalizes distances to the boundary of the polytope if $d_B(x) \leq \gamma_B$ and positive distances (x is correctly classified) if $d_D(x) \leq \gamma_D$. Notice that wrongly classified points are always penalized. The part of the regularizer corresponding to $d_D(x)$ has been suggested in [10] in a similar form as a loss function for general neural networks without motivation from robustness guarantees. They have an additional loss penalizing difference in the outputs with respect to changes at the hidden layers which is completely different from our geometrically motivated regularizer penalizing the distance to the boundary of the linear region. The choice of γ_B, γ_D allows different trade-off between the terms. In particular $\gamma_D < \gamma_B$ (stronger maximization of $d_B(x)$) leads in practice to more points for which the optimal robustness guarantee (case $d_D(x) \leq d_B(x)$) can be proved.
For practical reasons, we also propose a variation of our MMR regularizer in (3):

$$\begin{aligned} kMMR(x) =& \frac{1}{k_B} \sum_{i=1}^{k_B} \max\left(0, 1 - \frac{d_B^i(x)}{\gamma_B}\right) \\ &+ \frac{1}{k_D} \sum_{i=1}^{k_D} \max\left(0, 1 - \frac{\overline{d_D^i}(x)}{\gamma_D}\right), \end{aligned} \qquad (4)$$

where $d_B^i(x)$ is the distance of x to the i-th closest hyperplane of the boundary of the polytope and $d_D^i(x)$ is the analogue for the decision boundaries. Basically, we are optimizing, instead of the closest decision hyperplane, the k_D-closest ones and analogously the k_B-closest hyperplanes defining the linear region $Q(x)$ of x.

This speeds up the training time as more hyperplanes are moved in each update. Moreover, when deriving lower bounds using more than one linear region, one needs to consider more than just the closest boundary hyperplane. Finally, many state-of-the-art schemes for proving lower bounds [29, 28] work well only if the activation status of most neurons is constant for small changes of the points. This basically amounts to ensure that the boundaries of all hyperplanes are sufficiently far away, which is exactly what our regularization scheme is aiming at. Thus our regularization scheme helps to improve other schemes to establish better lower bounds (see Section 5). This is also the reason why the term pushing the polytope boundaries away is essential. Just penalizing the distance to the decision boundary is not sufficient to prove good lower bounds as we will show in Section 5. Compared to the regularization scheme in [29] using a dual feasible point of the robust loss, our approach has a direct geometric interpretation and allows to derive the exact minimal perturbation for some fraction of the test points varying from dataset to dataset but it can be as high as 99%. In practice, we gradually decrease k_B and k_D in (4) after some epochs (note that for $k_B = k_D = 1$ formulations (3) and (4) are equivalent) so that only the closest hyperplanes of each training point influence the regularizer.

Thus, denoting the cross entropy loss $CE(f(x), y)$, the final objective of our models is

$$\frac{1}{n}\sum_{i=1}^{n} CE(f(x_i), y_i) + \lambda\, kMMR(x_i), \qquad (5)$$

where $(x_i, y_i)_{i=1}^{n}$ is the training data and $\lambda \in \mathbb{R}_+$ the regularization parameter. Figure 2 shows the effect of the regularizer. Compared to the unregularized case the size of the linear regions is significantly increased.

5 Experiments

5.1 Main experiments

We provide a variety of experiments aiming to show the state-of-the-art performance of MMR to achieve robust classifiers. We use four datasets: MNIST, German Traffic Signs (GTS) [25], Fashion MNIST [31] and CIFAR-10. We consider in the paper robustness wrt to both l_2 and l_∞ distance. We use two criteria. First, upper and lower bounds on the robust test error for a given threshold ϵ, that is the largest achievable test error if every test input can be modified with a perturbation δ such that $\|\delta\|_p \leq \epsilon$ and δ is chosen so that $x + \delta$ is classified wrongly. Second, we show in the appendix results for lower and upper bounds on the minimal adversarial perturbation $\|\delta\|_p$ from (1).

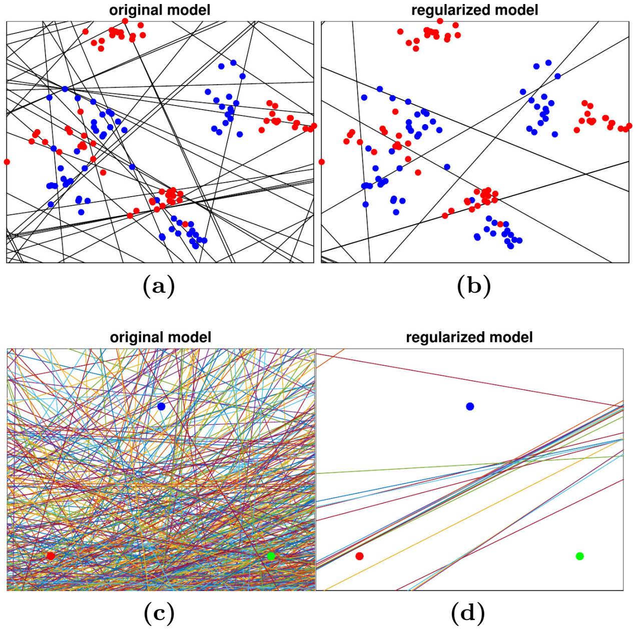

Figure 2: The effects of MMR. *Top row*: we train two networks with one hidden layer, 100 units, on 128 points belonging to two classes (red and blue). Figure 2a shows the points and how the input space is divided in regions on which the classifier is linear. Figure 2b is the analogue for our MMR regularized model. *Bottom row*: we show region boundaries (one hidden layer, 1024 units) on a 2D slice of $\mathbb{R}^{784}$ spanned by three random points from different classes of the MNIST training set. We observe a clear maximization of the linear regions for the MMR-regularized case (Figure 2d) versus the non-regularized case (Figure 2c).

Lower bounds on the robust test error for l_2 and l_∞ are computed using the attack via Projected Gradient Descent (PGD) [18]. Upper bounds on the robust test error are computed using the approach of [29]. Finally, [27] yields upper and lower bounds on the robust test error via mixed integer programming (MIP). We use the solver for the MIP [27] with a timeout of 120s per point, that is if the point has not been verified in this time the solver is stopped. This technique is currently effective for l_∞ but not for l_2, where basically in almost every case the timeout is reached and thus we discard for l_2 the MIP. Finally, we report the minimum of the upper bounds on the robust test error found by [29] and [27]. We compare our own upper bounds with the ones of [29] in the appendix. Moreover, we take the maximum of the lower bounds on the robust test error found by PGD, the MIP and the attack of [8, 9].

We compare five methods: plain training, adversarial training of [18] which has been shown to significantly increase robustness, the robust loss of [29] (which supports both l_2 and l_∞ norms), denoted as KW, our regularization scheme MMR and MMR together with adversarial training again as in [18]. All schemes are evaluated on a one hidden layer fully connected network with 1024 hidden units (FC1) and a convolutional

Table 1: Comparison of different methods regarding robustness. We here report the statistics of 5 different training schemes: *plain* (usual training), *at* (adversarial training [18]), MMR (ours), MMR+*at* (MMR plus adversarial training), and KW (the robust training introduced in [29]). We show, in percentage, test error (TE), lower (LB) and upper (UB) bounds on the robust test error at the threshold ϵ indicated for each dataset. The robustness statistics are computed on the first 1000 points of the respective test sets for l_∞-robustness, on the full test set for l_2-robustness. KW models marked with * are taken from [29, 30], while the other have been retrained according to the available code. However also for the pretrained models the bounds have been evaluated again with the discussed combination of multiple techniques. The model indicated by [1] has been obtained at epoch 50 instead of 100 due to optimization issues in this particular setting.

	l_∞-norm robustness						**l_2-norm robustness**					
training	FC1			CNN			FC1			CNN		
scheme	TE	LB	UB	TE	LB	UB	TE	LB	UB	TE	LB	UB
MNIST	$\epsilon = 0.1$						$\epsilon = 0.3$					
plain	1.44	93.0	100	0.91	74.0	100	1.73	9.7	66.3	0.85	3.1	100
at	0.92	10.0	99.0	0.82	3.0	100	1.15	2.6	16.9	0.87	1.8	100
KW	3.19	10.9	10.9	1.26*	4.4	4.4	1.19	2.4	5.2	1.11	2.2	6.0
MMR	2.11	22.5	24.9	1.65	6.0	6.0	2.40	5.9	8.8	2.57	5.8	11.6
MMR+at	2.04	14.0	14.1	1.19	3.6	3.6	1.77	3.8	6.4	2.12[1]	4.6	9.7
F-MNIST	$\epsilon = 0.1$						$\epsilon = 0.3$					
plain	9.49	100	100	9.50	96.0	100	9.70	42.8	91.8	9.32	57.1	100
at	11.69	29.5	95.5	11.54	21.5	73.0	9.15	19.9	61.4	8.10	20.4	100
KW	21.31	32.8	32.8	21.73*	32.4	32.4	11.24	17.2	22.7	13.08	18.5	21.7
MMR	18.11	37.6	42.0	14.52	33.2	33.6	13.28	25.0	28.0	12.85	25.4	35.3
MMR+at	15.84	31.3	34.8	14.50	26.6	30.7	12.12	19.7	23.4	13.42	26.2	39.1
GTS	$\epsilon = 4/255$						$\epsilon = 0.2$					
plain	12.72	61.0	77.0	7.11	63.0	99.5	12.24	37.5	44.7	6.78	33.3	99.2
at	9.33	34.5	48.5	6.84	29.5	81.0	13.55	33.1	43.2	8.75	23.8	98.6
KW	13.99	33.0	33.0	15.56	36.1	36.6	16.84	16.9	31.2	14.33	28.7	34.7
MMR	14.29	39.8	39.8	13.31	49.5	49.6	14.55	33.2	34.7	14.22	36.2	36.9
MMR+at	13.10	33.1	35.4	14.88	38.3	38.4	13.94	29.7	32.1	15.35	32.1	33.2
CIFAR-10	$\epsilon = 2/255$						$\epsilon = 0.1$					
plain	-	-	-	24.63	91.0	100	-	-	-	23.31	47.2	100
at	-	-	-	27.04	52.5	88.5	-	-	-	25.82	35.8	100
KW	-	-	-	38.91*	46.6	48.0	-	-	-	40.24	43.9	49.0
MMR	-	-	-	34.61	57.5	61.0	-	-	-	40.92	50.6	57.1
MMR+at	-	-	-	35.38	47.9	54.2	-	-	-	37.75	43.9	53.3

network (CNN) with 2 convolutional and 2 dense layers as used in [29]. For more details see Appendix.
We make our code and models publicly available[1].

Improvement of robustness: The results can be found in Table 1. For CIFAR-10 we evaluate the different methods only on the CNNs as fully connected networks do not have good test performance. We report clean test error, lower and upper bounds on robust test error at the threshold indicated in Table 1, computed on 1000 points for l_∞, and on the full test sets for l_2 (as we do not do the MIP evaluation there).
Both KW and MMR-at achieve similar performance regarding lower and upper bounds on the robust test error. For l_∞ our MMR-at achieves overall better performance than KW, sometimes with significantly better clean test error like on F-MNIST. In some cases, e.g. on CIFAR-10, MMR-at provides slightly worse upper bounds, but instead preserves better test error.
For l_2 KW performs better than MMR-at with the exception of GTS. This is to be expected as we use for the evaluation of the upper bounds on the robust test error the approach of [29] which are directly optimized by their robust training procedure. Note that both KW and MMR/MMR-at outperform by large margin plain and adversarial training regarding provable robustness (upper bounds on the robust test error). With the exception of the CNN-l_2 models for MNIST and F-MNIST, the upper bound on robust test error of MMR-at is smaller than the lower bound of the plain model. Thus our models are provably better than the plain models regarding robust test error. Moreover,

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

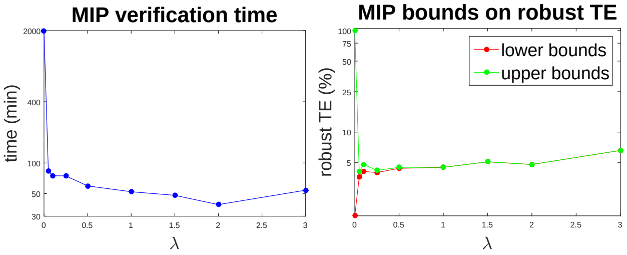

Figure 3: Verifiability of models. We show the runtime (left) in minutes that MIP [27] takes to verify 1000 points, setting a timeout of 120s (that is the mixed-integer optimization stops anyway after the time limit is reached), with models trained with different values of λ (see Equation (5)). Note that a logarithmic scale is used on the y-axis. Moreover, we report (right) lower (red) and upper (green) bounds on robust test error. The *plain* model, trained without MMR ($\lambda = 0$) needs 51 times more to be verified, with only 1% of the points certified. Conversely, even with a light MMR regularization lower and upper bounds are tight.

regarding robustness wrt to l_∞ the gaps between lower and upper bounds are often very small for KW and MMR/MMR-at showing that both techniques lead to models which can be easier checked via the MIP which we discuss next in more detail.

Enhancing verifiability: A key aspect of any robust training should be the ability of producing models both resistant to adversarial manipulation and being *verifiable*, in the sense of having guarantees on the minimal perturbation changing the decision for a test input. In fact, even if empirically model seems to be robust, only computing certificates allows to completely trust it. In our experiments, we could use successfully the method of [29] to get meaningful bounds, which so far, as noted in [22], could be achieved only on models provided by the specific training of [29]. Moreover, the MIP is too slow to run on standard models, so that ad hoc techniques have been developed to train classifiers verifiable by MIP [32]. Our MMR training produces also models which can be checked by MIP. This is due to the fact that the hyperplanes representing the boundary of the polytope are actually the boundary between the different behaviours of ReLU units, so that pushing the hyperplanes implies that many inputs of ReLU units have constant sign in a wide region around the test points (and ReLU units with unstable sign are the main problem for the MIP).

In Figure 3 we show the time MIP needs to run on 1000 points (with timeout of 120s per point) and lower and upper bounds on robust test error wrt l_∞-distance at $\epsilon = 0.1$. We use CNNs trained on MNIST with $\gamma_B = \gamma_D = 0.15$ and different values of λ representing the weight of our regularizer in the loss (5) (for $\lambda = 0$ one gets the plain model). It is clear that MIP performs poorly both in runtime (almost 2000 minutes) and performance (LB=1%, UB=100%) on the plain model. In contrast, the MMR models are verified quickly (between 35 and 79 minutes) and almost completely (the rate of certified points is between 99.3% and 100%), that is lower and upper bounds are close or even equal. Please note that both statistics improve with increasing λ up to 2, while runtime gets worse with a larger value (as at $\lambda = 3$ the classifier becomes less robust and then requires a higher computational effort to be certified).

5.2 Further experiments

We present a series of experiments for a detailed understanding of how MMR works. For this section we consider models trained to be l_2-robust and evaluate robustness as the average l_2-norm of the perturbations necessary to change the classification when applied to test inputs. Lower bounds on this perturbation are computed either with [29] or our method (Theorem 3.1), while upper bounds are provided by Carlini-Wagner l_2-attack (CW) [6]. We also introduce a second fully connected architecture, FC10, with 10 hidden layers (see Appendix for details).

Importance of linear regions maximization: In order to highlight the importance of both parts of the MMR regularization, i) penalization of the distance to decision boundary and ii) penalization of the distance to boundary of the polytope, we train, for each dataset/architecture, models penalizing only the distance to the decision boundary, that is the second term in the r.h.s. of (3) and (4). We call this partial regularizer MMR-d_D, in contrast to the full version MMR-*full*. Then we compare the lower and upper bounds on the solution of (1) for MMR-d_D and MMR-*full* models. For a fair comparison we consider models with similar test error. We clearly see in Table 2 that the lower bounds, computed by the method presented in [29], are always significantly better when MMR-*full* is used, while the behavior of the upper bounds does not clearly favor one of the two. This result shows that in order to get good lower bounds one has to increase the distance of the points to the boundaries of the polytope.

Guaranteed optimal solutions via MMR: Theorem 3.1 provides a simple and efficient way to obtain in certain cases the solution of (1). Although for normally trained networks the conditions are rarely satisfied, we show in Table 3 that for the MMR-models for fully connected networks for a significant fraction of the test set we obtain the globally optimal solution of (1), that is the true l_2-robustness. Moreover, we report how much better our globally optimal solutions are compared to the lower bounds of [29]. Interestingly, we can provably get the true robustness for around 10% of points for F-MNIST and for over 98% of the points on GTS for the case of fully connected networks. For these cases the optimal solutions have roughly 7% larger l_2-norm for F-MNIST and 0.5% larger for GTS

Table 2: Full version of MMR is necessary. We compare the statistics of models trained with the full version of MMR as in (3) and (4) (left) and with only the second part penalizing the distance to the decision boundary (right). While the the test error is for the full test set, are the lower resp. upper bounds on the l_2-norm of the optimal adversarial manipulation are compared on the first 1000 points of the test set. One can clearly see that the lower bounds improve significantly when one uses the full MMR regularization.

		MMR-*full*			MMR-d_D		
dataset	model	test error	lower bounds	upper bounds	test error	lower bounds	upper bounds
MNIST	FC1	1.51%	**0.69**	**1.69**	0.93%	0.35	**1.69**
	FC10	1.87%	**0.48**	1.48	1.21%	0.20	**1.62**
GTS	FC1	11.15%	**0.69**	**0.69**	12.09%	0.48	0.63
	FC10	12.82%	**0.64**	**0.67**	12.41%	0.12	0.48
F-MNIST	FC1	10.22%	**0.50**	0.85	9.83%	0.31	**1.30**
	FC10	11.73%	**0.68**	**1.18**	10.32%	0.13	1.15

Table 3: Occurrence of guaranteed optimal solutions. For each dataset and architecture we report the percentage of points of the test set for which we can compute the guaranteed optimal solution of (1) for models trained without (plain setting) and with MMR regularization. We show the test error of the models as well. In most of the fully connected cases, we achieve certified optimal solutions for a significant fraction of the points without degrading significantly, or sometimes improving, the test error. Moreover, where we have a meaningful number of points with provable minimal perturbation, it is interesting to check how much worse the lower bounds computed by [29] (LB) are. Thus the column *opt vs LB* indicates, in percentage, how much larger the l_2-norm of the optimal solution of (1) is compared to its lower bound, explicitly $\left(\|\delta_{opt}\|_2/LB - 1\right) \times 100\%$.

		MMR training			plain training	
dataset	model	test error	optimal points	opt vs LB	test error	optimal points
MNIST	FC1	1.51%	0.20%	14.11%	1.59%	0.02%
	FC10	1.87%	0.06%	-	1.81%	0.02%
	CNN	1.17%	0.00%	-	0.97%	0.00%
GTS	FC1	11.15%	99.97%	0.59%	12.27%	1.12%
	FC10	12.82%	94.86%	0.66%	11.28%	0.14%
	CNN	7.40%	0.00%	-	6.73%	0.00%
F-MNIST	FC1	10.22%	11.22%	6.53%	9.61%	0.37%
	FC10	11.73%	9.90%	7.27%	10.53%	0.04%
	CNN	10.30%	0.09%	-	8.86%	0.00%

than the lower bounds. Note that currently the MIP [27] is not efficient for l_2 even though there is room for improvement. Globally optimal solutions for larger networks achieved via our method can serve as a test both for lower and upper bounds. This is an important issue as currently large parts of the community relies just on upper bounds of robustness using attack schemes like the CW-attack which we address in the the next paragraph.

Evaluation of CW-attack: The CW-attack [6] which we use for our upper bounds is considered state of the art. Thus it is interesting to see how close it is to the globally optimal solution. On GTS FC1 we find for the MMR model with best test error the globally optimal solution of (1) for 12596 out of 12630 test points. We compare on this subset the optimal norm $\|\delta_{opt}\|_2$ to $\|\delta_{CW}\|_2$ obtained by the CW-attack.While the CW-attack performs in general well, there are 2330 points (18.5% of the test set) where the CW-attack has at least 10% larger norms and 1145 points (around 9.1%) with at least 20% larger norms. The maximal relative difference is 235%. Thus at least on a pointwise basis evaluating robustness with respect to an attack scheme can significantly overestimate robustness, see also [8]. This shows the importance of techniques to prove lower bounds. Moreover, the time to compute the adversarial examples by the CW-attack is 16327s, while our technique provides both lower bounds and optimal solutions in 1701s.

6 Conclusion

We have introduced a geometrically motivated regularization scheme which leads to provably better robustness than plain training. Moreover, it performs as well as the state of the art [29] in terms of robust test error. Finally, our scheme allows to obtain the provably optimal solution in a significant fraction of cases for large fully connected networks which can be used to test lower and upper bounds.

Acknowledgements

We would like to thank Vincent Tjeng for helping to set up the MIP evaluation. Furthermore, we thank Eric Wong and Zico Kolter for adapting their code to the l_2-norm, as well as for many helpful discussions.

References

[1] R. Arora, A. Basuy, P. Mianjyz, and A. Mukherjee. Understanding deep neural networks with rectified linear unit. In *ICLR*, 2018.

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[3] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi. Measuring neural net robustness with constraints. In *NIPS*, 2016.

[4] N. Carlini, G. Katz, C. Barrett, and D. L. Dill. Provably minimally-distorted adversarial examples. preprint, arXiv:1709.10207v2, 2017.

[5] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017.

[6] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

[7] M. Cisse, P. Bojanowksi, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness to adversarial examples. In *ICML*, 2017.

[8] F. Croce and M. Hein. A randomized gradient-free attack on relu networks. In *GCPR*, 2018.

[9] F. Croce, J. Rauber, and M. Hein. Scaling up the randomized gradient-free attack reveals overestimation of robustness using established attacks. In preparation, 2019.

[10] G. F. Elsayed, D. Krishnan, H. Mobahi, K. Regan, and S. Bengio. Large margin deep networks for classification. preprint, arXiv:1803.05598v1, 2018.

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[12] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. In *ICLR Workshop*, 2015.

[13] M. Hein and M. Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *NIPS*, 2017.

[14] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvari. Learning with a strong adversary. In *ICLR*, 2016.

[15] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *CAV*, 2017.

[16] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In *ICLR Workshop*, 2017.

[17] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017.

[18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Valdu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[19] M. Mirman, T. Gehr, and M. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *ICML*, 2018.

[20] M. Mosbach, M. Andriushchenko, T. Trost, M. Hein, and D. Klakow. Logit pairing methods can fool gradient-based attacks. In *NeurIPS 2018 Workshop on Security in Machine Learning*, 2018.

[21] N. Papernot, P. McDonald, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep networks. In *IEEE Symposium on Security & Privacy*, 2016.

[22] A. Raghunathan, J. Steinhardt, and P. Liang. Certified defenses against adversarial examples. In *ICLR*, 2018.

[23] P. F. S.-M. Moosavi-Dezfooli, A. Fawzi. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, pages 2574–2582, 2016.

[24] J. Sokolic, R. Giryes, G. Sapiro, and M. R. D. Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65:4265 – 4280, 2017.

[25] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, pages 2503–2511, 2014.

[27] V. Tjeng, K. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. preprint, arXiv:1711.07356v3, 2019.

[28] T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. S. Boning, and I. S. Dhillon. Towards fast computation of certified robustness for relu networks. In *ICML*, 2018.

[29] E. Wong and J. Z. Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018.

[30] E. Wong, F. Schmidt, J. H. Metzen, and J. Z. Kolter. Scaling provable adversarial defenses. In *NeurIPS*, 2018.

[31] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. preprint, arXiv:1708.07747, 2017.

[32] K. Y. Xiao, V. Tjeng, N. M. Shafiullah, and A. Madry. Training for faster adversarial robustness verification via inducing relu stability. preprint, arXiv:1809.03008, 2018.

[33] S. Zheng, Y. Song, T. Leung, and I. J. Goodfellow. Improving the robustness of deep neural networks via stability training. In *CVPR*, 2016.