Certified error rates for neural networks

MIGUEL DE BENITO DELGADO, appliedAI

January 14th, 2022

Adversarial training has been shown to improve robustness of neural networks to certain classes of data perturbations. Despite constant progress, counterattacks appear immediately after each new method is proposed. This is because of a lack of bounds on the error that an attack can induce. We review a series of papers working towards certified error rates for networks using either special certification training objectives or arbitrary ones, including those employed for adversarial training.

Contents

From Nov 8, 2021 to Nov 10, 2021, the Simons Institute conducted the workshop Average-Case Complexity: From Cryptography to Statistical Learning, in the context of their semester program Computational Complexity of Statistical Inference. This post summarizes and extends parts of [Rag21].

Statistical robustness is concerned with how estimators react to shifts in data distributions. One of the questions it tries to answer is the following: If one has a good estimator for a given parametric family, but it is fitted on data sampled from a distribution "slightly outside" the family, does the estimator still behave well? In practical applications, the model is almost never correct, so it is of great interest to design estimators which are **robust** in this sense. This is sometimes called *robustness to poisoning* (against modification of the training set, possibly malicious).

In the same spirit as above, "robustness" in supervised learning refers to the ability to conserve predictive power under different scenarios. In some critical applications, neural networks need to be robust against data specially crafted to fool them at test time, for instance in autonomous vehicles or intrusion detection. Here we focus on this case, namely the *evaluation* on a distribution slightly different from the training one, sometimes known as **adversarial robustness**.

Several methods exist to improve test-time robustness, with *adversarial training* forming a family of the most popular and successful ones.[1] However, despite their success and that of other heuristics, they typically lack theoretical guarantees. As a matter of fact, for every adversarial training technique published in the past years, a counterattack has been immediately developed which beats it (see the references in [Rag21]).

[1] Since the seminal works on adversarial attacks [SZS+13] in 2013 and [GSS15] on adversarial training in 2014 , thousands of papers have appeared in the field. For a recent review and taxonomy of the field, we refer to [BLZ+21].

1 Bounding the maximal error

A sensible path is therefore to avoid a never ending game of cat and mouse devising heuristic attacks and counterattacks, and instead try to bound or reduce the maximal expected loss when perturbations to the test data are allowed in a certain class.[2]

[2] This usually means allowing perturbations of test data up to a magnitude $\varepsilon > 0$ in some norm. For vision tasks, it is natural and convenient to take l_∞: an ε-ball around x contains all images whose pixels differ at most ε from those of x.

Bounds on this error can be added as a penalty term to the loss which directly reduces the maximal error that a class of attacks can achieve. As we will see below, this technique can also provide upper bounds on the error during training, cf. the end of Section 3. The additional term is however at the cost of final predictive power of the model (and, interestingly, it usually also increases sparsity in the weights).

It is therefore advantageous to compute instead bounds for arbitrary, powerful networks (known as "*verification-agnostic*" networks because they haven't been specifically trained to reduce this error). However, the methods to do so typically suffer from problems of scalability and tightness, particularly losing power when large modifications to the inputs are allowed.

There are three main lines of work pursuing these two goals.

Exact verification falls in the second category, and tries to bound the exact error of arbitrary (albeit small) networks. In order to do this, it performs an exhaustive search over the class of allowed perturbations for each input. This is usually done with Satisfiability Modulo Theory or Mixed Integer Programming formulations and despite there being several optimizations in the literature, their cost is typically prohibitive for modern neural networks.[3] For this reason, we won't cove any such techniques in this post.

[3] These are on par with some convex relaxations as SDPs, solved with interior point methods, see below.

Convex relaxations of the bound on the maximal error can be used in either of the two forms mentioned: for post-training bounding or during training. The idea is to express the maximum error that an attacker can cause for any given test datum (x, y) as a Quadratically Constrained Quadratic Program (QCQP), typically a maximization over a set of allowed perturbations of x with y fixed. Then one uses linear or semidefinite relaxations to convexify the objective. Linear Programming (LP) methods scale well but provide certifications which are typically too loose, and some research shows that there is a fundamental gap that cannot be covered with linear relaxations, see e.g. [RSL18b, Proposition 1] or [SYZ+20]. On the other hand Semi-Definite Programs (SDPs) provide tighter bounds but scale poorly when using off-the-shelf interior point methods.[4] Recent work leverages dual formulations of the SDPs achieving linear memory cost and running times, while still providing reasonably good bounds, see Section 5.

[4] Specifically, for n nodes, $\mathcal{O}(n^6)$ in runtime and $\mathcal{O}(n^4)$ in memory, see refs [41, 60] in [DDK+20].

Finally, **randomized smoothing** belongs in a different category, since it is based on ideas in differential privacy and randomizes the outputs to improve robustness. We will not cover any such approach here.

We focus on convex relaxations, and review a series of papers using SDPs both for certification of pre-trained networks and penalised training.

2 The setting

For simplicity, we discuss binary classification of images, but the case of multiple classes is handled similarly and other data domains are also possible.

We allow a class of data manipulations of the form "change each pixel of an image by at most 10/255ths". More precisely, an **ℓ_∞-admissible attack** is of the following form: given a data domain $\mathbb{R}^d$ and labels $\mathcal{Y} = \{0, 1\}$ we allow an attacker $A: \mathbb{R}^d \times \mathcal{Y} \to \mathbb{R}^d$ to modify any coordinate of $x \in \mathbb{R}^d$ by at most a fixed amount $\varepsilon > 0$, i.e. we allow any $\tilde{x} = A(x, y)$ with $\|x - \tilde{x}\|_\infty < \varepsilon$. The goal of the attacker is to make a classifier $h: \mathbb{R}^d \to [0, 1]^2$ produce an output $h(\tilde{x})$ which maximises the **margin**:[5]

$$m_y(\tilde{x}) := h_{2-y}(\tilde{x}) - h_{1+y}(\tilde{x}), \tag{1}$$

where the two outputs h_1, h_2 are the confidences for classes 0 and 1 respectively. A positive margin is a succesful attack. The attacker has access to all information about the network and data distribution (this full-information attack model is sometimes called **white-box attack**).

[5] This cumbersome choice of subindices subsumes the two possible situations for misclassification into one when subindices start at 1 and $\mathcal{Y} = \{0, 1\}$. While in the two-dimensional case the margin can always be reduced to an expression solely dependent on the confidence in the correct class, for higher-dimensional certification it is common to define the margin as a function of two selected classes. This notation then generalizes to such cases.

We work with 2-layer networks of the form

$$h(x) := V\sigma(Wx), \tag{2}$$

with $V \in \mathbb{R}^{2 \times d_1}$, $W \in \mathbb{R}^{d_1 \times d}$ and non-linearity σ.[6]

[6] The limitation to two layers is only for simplicity. All but the first of the papers reviewed generalize to an arbitrary number of them.

Now, given a sample (x, y) consider the **worst possible modification** of x within ε distance, leaving the label untouched:

$$\delta_y^\star(x) := \underset{\|x-\tilde{x}\|_\infty \leqslant \varepsilon}{\operatorname{argmax}}\ m_y(\tilde{x}). \tag{3}$$

Note that this maximization depends on h and is typically intractable. The **robust error** is the maximal expected loss over data within an ε-ball around "true" data:

$$E_{\text{robust}}(h) := \mathbb{E}_{X,Y}[l(h(\delta_Y^\star(X)), Y)],$$

with $l(\hat{y}, y)$ being the 0-1 loss $\mathbb{I}(\hat{y} \neq y)$ in our case. The corresponding sample quantity, which we also name "robust error", is the maximal proportion of misclassified test data, under ℓ_∞-attacks of magnitude $\varepsilon > 0$:

$$\hat{E}_{\text{robust}}(h) := \frac{1}{n} \sum_{i=1}^{n} l(h(\delta_{y_i}^\star(x_i)), y_i).$$

With 0-1 loss, the argmax in $\delta_{y_i}^\star$ is not necessary to compute this error, since only the sign of the margin matters: If the quantity

$$\operatorname{opt}(x, y) := \max_{\|x-\tilde{x}\|_\infty \leqslant \varepsilon} m_y(\tilde{x}) \tag{4}$$

is negative, the network's confidence in the correct class will be maximal in the whole ε-neighbourhood around x and $l(h(\delta_y^\star(x)), y) = 0$ for this sample. In other words, one can count the number of points in which opt is positive to compute the robust error:

$$\hat{E}_{\text{robust}}(h) = \frac{1}{n} |\{x_i : \operatorname{opt}(x_i, y_i) > 0\}|. \tag{5}$$

If $\text{opt}(x, y) \leqslant 0$ one says that (x, y) is **certified** (or rather ℓ_∞, ε-certified): no admissible attack A can deterministically fool the classifier for this sample. We call m a **certification function**.[7]

[7] In Section 5 we will see a method that generalizes to arbitrary quadratic certification functions.

The main difficulty for certification is the maximization (4), which in the works reviewed here is tackled with convex relaxations. One problem arising from this approach is evaluating the tightness of the bound because the exact maximization is typically intractable. As a proxy, a lower bound on $\text{opt}(x, y)$ is obtained, e.g. with an adversarial attack like Projected Gradient Descent (PGD, [MMS+18]).[8]

[8] The more powerful the attack, the tighter this lower bound will be.

A word on the allowed threat model

Working within the framework of ℓ_∞ or, more generally, ℓ_p attacks is both convenient from a mathematical point of view, and of interest since many of them try to modify inputs in ways imperceptible to the human eye, a characteristic well modeled by small ℓ_p balls around samples. But it is important to note that it ignores more natural and often very effective manipulations of the empirical data distribution which can be used to force misclassifications.

One could for instance sample new images from the same source as the training data, obtaining qualitatively similar data on which the classifier fails often due to the network having overfitted to particulars of the specific training set.[9]

[9] The field of "domain generalization" tries to devise methods that learn from multiple datasets pertaining to the same task but potentially obtained independently.

The class of ℓ_p-bounded attacks also does not cover realistic attack scenarios like adversarial patches, which preserve the semantics of an image with alterations which, despite being obvious to the eye, might not stand out as an attack to the untrained observer.[10]

[10] Of course, one can design countermeasures against these, but, as mentioned above, certification against broad classes of attacks is in general more desirable.

3 Certified robustness via gradient bounds

[RSL18a] studies the two different goals described above:

1. To compute bounds on the maximal error that an attacker can cause for fixed weights V, W of h (see (2)).

2. To learn new weights V, W for h which reduce this maximal error.

The authors circumvent the problem of computing (5) with a tractable upper bound

$$U(m, x, y) \geqslant l(h(\delta_y^\star(x)), y),$$

which avoids the maximization (3). They define the ***U*-certified error** to be[11]

[11] The corresponding sample quantity is then the fraction of points that can be potentially attacked.

$$E_{\text{cert}} := \mathbb{E}_{X,Y}[\mathcal{U}(m, X, Y) > 0].$$

By construction, no ℓ_∞-attack can attain an error higher than the certified error. Also, because it is a pointwise bound, wherever $U = 0$, the integrand in the robust error must be 0 as well.

The function U is chosen to be a convex relaxation of (3) called MaxGrad. The name comes from the following bound on the margin (1):

$$\begin{aligned} m_y(\tilde{x}) &= m_y(x) + \int_0^1 \nabla m_y(t\tilde{x} + (1-t)x)^\top (x - \tilde{x})\, \mathrm{d}t \\ &\leqslant m_y(x) + \varepsilon \max_{\|\tilde{x}-x\| < \varepsilon} \|\nabla m_y(\tilde{x})\|_1 \\ &\leqslant m_y(x) + \varepsilon \max_{\substack{s \in [-1,1]^{d_1} \\ t \in [-1,1]^d}} \frac{1}{2} t^\top W^\top (V_1 - V_2) \odot (\mathbf{1} + s), \end{aligned}$$

where $\odot$ denotes the Hadarmard product and the last inequality is valid in the case of two-layer networks with ReLU activations (see [RSL18a, Eqs. (5) to (7)] for the details).[12] The final right hand side is a non-convex quadratic maximization problem, which is relaxed to an SDP roughly as follows. First introduce new variables $a = (1, t, s)^\top$and M, a matrix which is a function of $V_1 - V_2$ and W. In the resulting problem, the new variable a appears as a rank-one matrix $P := a\,a^\top \in \mathbb{R}^{(m+d+1)\times(m+d+1)}$. This is relaxed as $P = A A^\top$ for some full-rank A, or equivalently, $P \succcurlyeq 0$ (see [RSL18a, Eqs (6) to (10)] for details). One obtains the SDP:

$$m_y(x) + \frac{\varepsilon}{4} \max_{\substack{P \succcurlyeq 0 \\ \mathrm{diag}(P) \leqslant 1}} \langle M, P \rangle, \tag{6}$$

which can be solved with off-the-shelf tools to obtain an upper bound denoted MaxGrad(m, x, y).

[12] The number of two layers (one hidden and a linear mapping for the outputs) is required for the argument in MaxGrad, but the activations can be swapped for any other with uniformly bounded derivative.

Limitations

The MaxGrad method has three major disadvantages:

1. It is limited to 2-layer networks, a problem solved later in [RSL18b], see Section 4.

2. It has a very high computational cost, manageable only for very small networks, see Note 4.

3. It can yield very loose bounds. As a matter of fact MaxGrad is effectively vacuous on adversarially trained networks. For a very small ε the quantity MaxGrad(m, x, y) is zero for less than 10% of the data.[13]

[13] In the literature this is often phrased in reverse as the *certified error* being greater than 90%, which, as defined above, is the fraction of the points where MaxGrad$(m, x, y) > 0$.

Certified robustness via a new objective

One way to mitigate the third problem is to add MaxGrad to the training objective, i.e. to set

$$(\hat{V}, \hat{W}) \in \operatorname*{argmin}_{V,W} \sum_{i=1}^{n} l(h(x_i), y) + \lambda \max_{\substack{P \succcurlyeq 0 \\ \mathrm{diag}(P) \leqslant 1}} \langle M, P \rangle,$$

with $\lambda > 0$ a regularization hyperparameter. Because computing the gradient of the above requires the expensive operation of solving the inner maximum, the authors use its dual formulation, which provides a cheaper upper bound for the primal, [RSL18a, Eq. (15)].

Importantly, because every solution of the dual bounds the optimum of the primal from above, at every iteration of the optimization one has an upper bound on the certified error.

By penalising the loss, what was a vacuous upper bound becomes non-vacuous in practice and a general strategy emerges: *in order to obtain certified defenses, it is possible to train networks by minimizing a loose but tractable upper bound for the worst-case loss.*

Subsequent work building upon this paper improves the certified error for some examples. However, the final real-world performance of these networks on unaltered test samples remains subpar, something that motivates an interest in certifiers which operate on certification-agnostic networks.

4 Certified robustness with SDP

The previous technique changes the objective and obtains new parameters which are certified to have at most some given error. But what about arbitrary, or "foreign", networks (i.e. not robustly trained)? We have seen that the gradient bound is mostly ineffective. Is it possible to improve it?

[RSL18b] does so with a direct computation of the maximum margin instead of a first-order approximation. The method also generalizes to any number of layers (ignoring computational cost), but we leave this out for simplicity.[14]

[14] One aspect of the generalization is the need to add bounds on the output of each layer analogous to those for the attack. The authors use very rough ones based on interval arithmetic, but suggest that they could be made tighter for added performance.

Fix some input (x, y) and let $\tilde{x} = A(x, y)$ be an attack. Recall that we want to bound the margin

$$\begin{aligned} m_y(\tilde{x}) &:= h_{2-y}(\tilde{x}) - h_{1+y}(\tilde{x}) \\ &= (V_{2-y} - V_{1+y})\, x^1, \end{aligned}$$

where $x^1 = \sigma(W\tilde{x})$, and V is the final linear layer of the network, i.e. we want, cf (4):

$$\begin{aligned} \operatorname{opt}(x, y) \;\; = \;\; & \max_{\tilde{x} \in \mathbb{R}^d} (V_{2-y} - V_{1+y})^\top x^1 \\ \text{s.t.} \;\; & x^1 = \sigma(W\tilde{x}), && \text{(ReLU)} \\ \text{and} \;\; & \|x - \tilde{x}\|_\infty \leqslant \varepsilon. && \text{(Attack)} \end{aligned}$$

Note how the non-linear activation function enters the problem as a constraint. The idea to make the maximization tractable is to write it as a quadratic problem, then relax the non-convex constraints into an SDP.

The key insight enabling this is that each ReLU activation $z_j = \max((Wx)_j, 0)$ in the network can be expressed as the quadratic constraint:[15]

$$z_j \geqslant 0, z_j \geqslant (Wx)_j, z_j(z_j - (Wx)_j) = 0, \quad \forall j.$$

[15] Later work has derived similar quadratic formulations for the sigmoid, tanh, leaky ReLU and other activations [FMP20].

The same applies to ℓ_∞ bounds on the perturbed inputs: $l_j \leqslant x_j \leqslant u_j$ for all j and given $l_j, u_j \in \mathbb{R}$, is equivalent to $(x_j - l_j)(x_j - u_j) \leqslant 0$ for every j, or

$$x_j^2 \leqslant (l_j + u_j) x_j - l_j(u_j), \quad \forall j.$$

With this, one can rewrite the above into the following quadratic, non-convex problem:[16]

[16] Comparison of vectors is done coordinate-wise.

$$\begin{aligned} \operatorname{opt}(x, y) &= \max_{x, z \in \mathbb{R}^d} V^\top z && (7)\\ &\text{s.t.} \quad z \geqslant 0, z \geqslant Wx, z \odot z = z \odot Wx, && \text{(ReLU)}\\ &\text{and} \quad x \odot x \leqslant (l + u) \odot x - l \odot u. && \text{(Attack)} \end{aligned}$$

The trick to rewrite this as an SDP is to introduce new variables $a = (1, x, z)^\top$ and $P := a a^\top$. With "symbolic indexing" $P[\cdot]$ one can write

$$P = \begin{pmatrix} P[1] & P[x^\top] & P[z^\top]\\ P[x] & P[x x^\top] & P[x z^\top]\\ P[z] & P[z x^\top] & P[z z^\top] \end{pmatrix},$$

and relax the problem above as:[17]

[17] Note that, as before, the key modification is the convexification of the constraint of having rank one, $P = a a^\top$, via the constraint $P = A A^\top$ for full-rank A, or equivalently, $P \succcurlyeq 0$.

$$\begin{aligned} \widetilde{\operatorname{opt}}(x, y) &= \max_{P \succcurlyeq 0} V^\top P[z] && \text{(SDP)}\\ \text{s.t.} \quad & P[z] \geqslant 0,\\ & P[z] \geqslant W P[x],\\ & \operatorname{diag}(P[z z^\top]) = \operatorname{diag}(W P[x z^\top]),\\ \text{and} \quad & \operatorname{diag}(P[x x^\top]) \leqslant (l + u) \odot P[x] - l \odot u. \end{aligned}$$

Using off-the-shelf solvers, the resulting SDP certifier, SDP-Cert, yields non vacuous bounds for several networks trained with standard adversarial methods, in contrast to MaxGrad (albeit only for small networks for which the optimization is possible). The intuition behind this is that MaxGrad is a linear approximation bounding the gradient uniformly, whereas SDP-Cert is a relaxation of the true quadratic problem of computing the worst possible margin $\operatorname{opt}(x, y)$.

SDP-Cert is also shown to provide better bounds than methods using linear programs, both empirically and theoretically. The latter statement is the content of [RSL18b, Proposition 1], which proves a dimension-dependent gap between LP and SDP bounds of order $\sqrt{d}$ for random networks.[18]

[18] Roughly, for networks of m hidden nodes and dimension d, with high probability $\tilde{\delta}^\star_{y,\text{LP}} = \Theta(m\, d)$ and $\tilde{\delta}^\star_{y,\text{SDP}} = \Theta(m \sqrt{d} + d \sqrt{m})$.

Limitations

Like MaxGrad, SDP-Cert has several drawbacks. First, the bounds are loose: the strongest known attacks attain errors which are significantly lower. This is due to points where the attack fails but also does the certification. Upon close inspection it can be seen that the distribution of margins for these is close to 0, i.e. the attacks were “almost” succesful, see Figure 1.

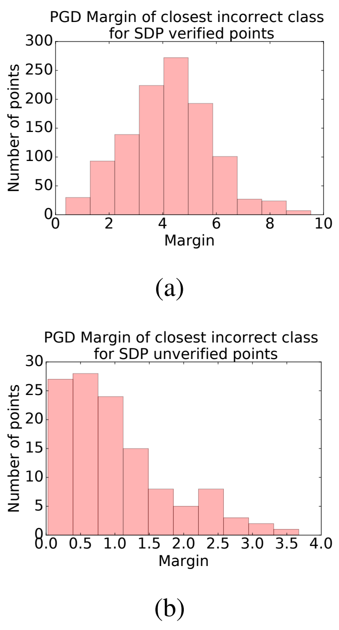

Figure 1. [RSL18b, Figure 3] Histogram of PGD margins for (a) points that are certified by the SDP and (b) points that are not certified by the SDP.

Second, the use of an SDP and off-the-shelf interior point methods limits applications to networks of at most a few thousand nodes. The next paper we study addresses this issue taking ideas from the SDP literature and achieves linear running times.

5 Efficient SDP certification

To overcome the problem of scalability, [DDK+20] work with the dual of the SDP formulation and express it as a maximum eigenvalue problem with linear constraints. In doing so, they generalize the work of Section 4 without compromising the tightness of the bound and can extend their method to any quadratically constrained quadratic program.

An important contribution is that the optimization can be done with first-order methods whose subgradient computations can be carried out as forward and backward passes on the network, so that the per-iteration complexity is that of a constant number of such passes and can be implemented using standard frameworks like TensorFlow, PyTorch or Jax. This allows scaling up the size of the networks one order of magnitude up to 20K nodes and 2M parameters, which is certainly far from the size of state of the art models in vision and language, but can still be easily encountered in applications.

As before we want to compute $\operatorname{opt}(x, y)$, but after deriving the convex relaxation (SDP), we move on to the dual problem, and express it as a maximum eigenvalue problem.

To this end, we first write the Lagrangian for problem (7).[19] Recall that the linear and quadratic constraints for a ReLU activation are $z \geqslant 0$, $z \geqslant Wx$ and $z \odot (z - Wx) \leqslant 0$ (where we write the last one as an inequality in view of the Lagrangian) and the quadratic constraints for an ℓ_∞ attack, $x \odot x \leqslant (l + u) \odot x - l \odot u$, for given bounds l, u. The Lagrangian of the constraints for this layer is then

[19] As in Section 4, the method is presented for an arbitrary number of layers, but we stay in a simpler setting for clarity.

$$\begin{aligned} \tilde{\mathscr{L}}(x, z, \lambda) \;:=\; & -z^\top \lambda_a \\ & +(Wx - z)^\top \lambda_b \\ & +z \odot (z - Wx)^\top \lambda_c \\ & +(x \odot x - (l + u) \odot x + l \odot u) \lambda_d . \end{aligned}$$

Rearranging terms, adding the objective $(V_{2-y} - V_{1+y})^\top x^1$ and renaming all network variables x, z as x, one obtains a full Lagrangian

$$\mathscr{L}(x, \lambda) := c(\lambda) + x^\top g(\lambda) + \tfrac{1}{2} x^\top H(\lambda)\, x,$$

where all the coefficients c, g, H are affine functions of λ, [DDK+20, Eqs. (2) to (3)]. Interestingly, because g and H are the gradient and Hessian of $\mathscr{L}$, and all they involve is forward passes and element-wise operations, they can be computed with automatic differentiation using all common ML frameworks.

A standard duality argument provides:

$$\operatorname{opt}(x, y) \leqslant \min_{\lambda \geqslant 0} \max_{l \leqslant x \leqslant u} \mathscr{L}(x, \lambda),$$

which in [DDK+20, Proposition 1, p. 5] is transformed into the minimization

$$\min_{\lambda \geqslant 0, \kappa \geqslant 0} f(\lambda, \kappa), \tag{SDP-FO}$$

with $\lambda \in \mathbb{R}$ and $\kappa \in \mathbb{R}^{1+N}$, N being the sum of the dimensions of all layers (input and hidden in our setting), and with the objective given by

$$f(\lambda, \kappa) \coloneqq c(\lambda) + \frac{1}{2} \mathbf{1}^{\top} [\operatorname{diag}(\kappa) - \lambda_{\min}^{-}(Z)\, \mathbf{1}]^{+},$$

where $\lambda_{\min}^{-}$ is the negative part of the smallest eigenvalue of a certain matrix $Z = Z(g, H)$.[20]

[20] See [DDK+20, Appendix A.4] for the proof that optimizing this new problem is equivalent to SDP-CERT.

The problem (SDP-FO) can be solved efficiently with a first-order projected subgradient method thanks to automatic differentiation and iterative computation of the eigendecomposition of Z. Good choices for initialization, regularization and learning rates improve convergence rates [DDK+20, Section 5.3].

The authors test SDP-FO in networks both with a verification penalty term in the loss and without it, but their main result is its effectiveness on the latter. For these verification-agnostic networks, the dual form provides a bound which is reasonably tight (wrt. a proxy lower bound obtained with PGD, as described in Section 2) in many cases while enabling verification of larger models.

An additional positive aspect of the method is that it can handle quadratic certification functions. The margin (1) for adversarial robustness is linear, but the authors show an example of robustness of VAEs to perturbations in latent space where the certification function is given by the (quadratic) reconstruction error [DDK+20, Eq. (7)]. Such quadratic objectives cannot be directly plugged into LP or exact solvers without relaxing them first.

Limitations

Despite a linear running time, the bounds provided by SDP-FO lose tightness as the network size increases. This can be seen in Figure 2, which shows the accuracy under an attack using Projected Gradient Descent and the one from the SDP-FO upper bound. The broader the gap, the worse the performance of the certifier.

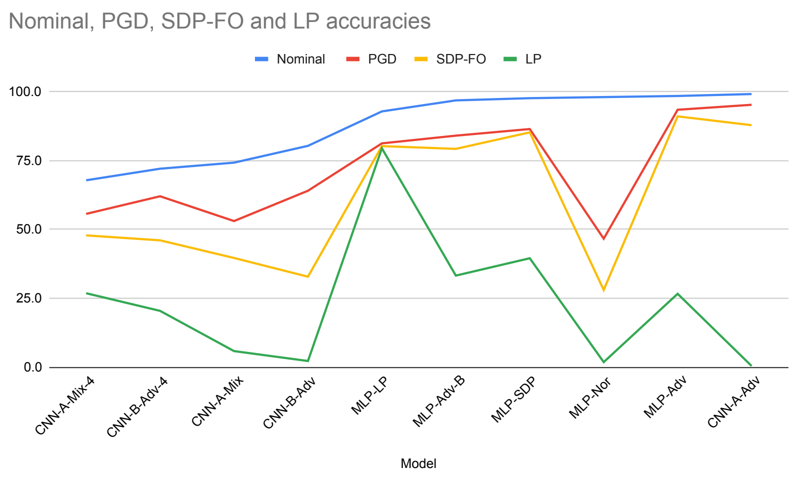

Figure 2. Data from [DDK+20, Table 1]. Accuracy for different models, using MNIST and CIFAR-10, evaluated on the same test set as given by: as-is (nominal), PGD attack, SDP-FO certifier, and the popular LP certifier [Ehl17].

6 Open questions in relaxed certification

Besides the obvious problem of scalability, several issues remain untackled in the literature on convex relaxations of the maximal expected error.

It is necessary to understand why networks trained with the certification penalty are in general less performant, as well as the role that architecture and initialization play in this matter. There is no characterization of the effective hypothesis class that one minimises over with certified training. Interestingly, these networks show higher sparsity, a phenomenon for which there is no explanation yet. We discuss the "robustness tradeoff" in a future post.

From a technical point of view, tighter relaxations are required at larger network sizes. A clear candidate for improvement is the computation of better bounds for the intermediate layers in the case of more than two layers.[21] Finally it would be interesting to see adoption of techniques from the SDP community beyond the 1st order dual method of Section 5.

[21] This refers to a set of constraints required for each additional layer.

BIBLIOGRAPHY

[BLZ+21] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent Advances in Adversarial Training for Adversarial Robustness. In *arXiv:2102.01356 [Cs]*. Montreal, Canada, apr 2021.

[DDK+20] Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy Liang, and Pushmeet Kohli. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In *Advances in Neural Information Processing Systems 33*. Nov 2020.

[Ehl17] Rüdiger Ehlers. Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis*, Lecture Notes in Computer Science, pages 269–286. Cham, 2017. Springer International Publishing.

[FMP20] Mahyar Fazlyab, Manfred Morari, and George J. Pappas. Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming. *IEEE Transactions on Automatic Control*, page 16, dec 2020.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. mar 2015.

[MMS+18] 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*. Feb 2018.

[Rag21] Aditi Raghunathan. Worst-Case Robustness in Machine Learning. nov 2021.

[RSL18a] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified Defenses against Adversarial Examples. In *6th International Conference on Learning Representations (ICLR 2018)*. Vancouver, Canada, feb 2018.

[RSL18b] Aditi Raghunathan, Jacob Steinhardt, and Percy S. Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *NeurIPS 2018*. Jan 2018.

[SYZ+20] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks. In *Proc. of the Thirty-Third Conference on Neural Information Processing Systems*. Vancouver, Canada, jan 2020.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *ArXiv:1312.6199 [cs]*, page 10, dec 2013.