Efficient Bi-Directional Verification of ReLU Networks via Quadratic Programming

Anonymous authors
Paper under double-blind review

Abstract

Neural networks are known to be sensitive to adversarial perturbations. To investigate this undesired behavior we consider the problem of computing the distance to the decision boundary (DtDB) from a given sample for a deep NN classifier. In this work we present an iterative procedure where in each step we solve a convex quadratic programming (QP) task. Solving the single initial QP already results in a lower bound on the DtDB and can be used as a robustness certificate of the classifier around a given sample. In contrast to currently known approaches our method also provides upper bounds used as a measure of quality for the certificate. We show that our approach provides better or competitive results in comparison with a wide range of existing techniques.

1 Introdution

The high predictive power of neural network classifiers makes them the method of choice to tackle challenging classification problems in many areas. However, questions regarding the robustness of their performance under slight input perturbations still remain open, severely limiting the applicability of deep NN classifiers to sensitive tasks that require certain certification of the obtained results.

In recent years this issue gained a lot of attention, resulting in a large amount of methods covering topics ranging from adversarial attacks and defenses against these to robustness verification and robust training. In this work, we focus on robustness verification of deep classifiers. That is, computing the distance from a given anchor point x^0 in the input space to its closest adversarial, i.e. a point that is assigned a different class label by the network. This problem plays a fundamental role in understanding behavior of deep classifiers and provides the only reliable way to assess classifier robustness. Unfortunately, its complexity does not let us hope for it to be solvable by a polynomial time algorithm. For deep classifiers with ReLU activation the verification problem can equivalently be reformulated as a mixed integer programming (MIP) task and was shown to be NP-complete by Katz et al. (2017). Even worse, Weng et al. (2018) showed that an approximation of DtDB of a certain (high) quality cannot be found within polynomial time.

Related work There exists three streams of related work on robustness verification of deep ReLU classifiers. Grouping depends on whether they are solving the verification problem exactly or verifying a bound on the distance to the decision boundary (DtDB).

The first group are **exact computation** approaches. As mentioned above, the verification task can be modeled using MIP techniques. Katz et al. (2017) present a modification of the simplex algorithm that can be used to solve the verification task exactly for smaller ReLU networks based on satisfiable modulo theory (SMT). Other approaches (see Ehlers (2017)) rely on SMT solvers when solving the described task. Bunel et al. (2018) provides an overview and comparison of those. Other exact methods (Dutta et al. (2018); Lomuscio and Maganti (2017); Tjeng et al. (2017)) deploy MIP solvers together with presolving trying to find a tight formulation of the MIP problem.

The second popular class of methods to verify classifier robustness deals with **verification of an ϵ-neighborhood**: given an anchor point x^0 and an $\epsilon > 0$, the task is to verify whether an adversarial point exists within the ϵ neighborhood of x^0 which is defined w.r.t. to a certain norm in the input space. All these methods relax the initial problem and require bounds on activation inputs in each layer. These bounds should be as tight as possible to ensure good final results.

Raghunathan et al. (2018b;a); Dvijotham et al. (2018; 2019) consider semidefinite (SDP) and linear (LP) problems as relaxations of the ϵ-verification problem. Wong and Kolter (2018) replace ReLU constraints by linear constraints and consider the dual formulation of the obtained LP relaxation. Weng et al. (2018) present an approach that also uses linear functions (and Zhang et al. (2018) later quadratic) to deal with nonlinear activation functions and propagate the layer-wise output bounds until the final layer. Finally Hein and Andriushchenko (2017); Tsuzuku et al. (2018) use the Lipschitz constant of the transformations within classifier's architecture.

Our approach belongs to the third group of approaches dealing with **constructing lower bounds** on DtDB without restricting admissible adversarial points to a given neighborhood. The ϵ-verification task is closely related to the problem of constructing lower bounds on DtDB: each ϵ-neighborhood that can be certified as adversarial-free immediately provides a lower bound on the minimal adversarial perturbation magnitude. It is also a common strategy for the methods from the previous group to set e.g. a binary search or a Newton method on top of their algorithm to find the largest ϵ such that the ϵ-neighborhood around x^0 can still be verified as robust. Croce et al. (2019) deploy the piecewise affine nature of the logits in a ReLU classifier and compute lower bounds on DtDB by assuming that the classifier behaves globally as it does in the linear region of the given anchor point.

Robust training The question of how to actually train a robust classifier is closely related to robustness verification since the latter might allow us to construct some type of robust loss based on the insights from the verification procedure (see Wong and Kolter (2018); Croce et al. (2019); Hein and Andriushchenko (2017); Raghunathan et al. (2018a); Tsuzuku et al. (2018)). We leave this direction for future work.

Contributions

1. We propose a novel **relaxation of the DtDB problem in form of a QP task** allowing efficient computation of high quality lower bounds on DtDB. For networks with up to two hidden layers we reach state-of-the-art performance with an improvement of over 50% when compared to the bounds obtained from the methods based on LP relaxations (**CROWN** by Zhang et al. (2018) and **ConvAdv** by Wong and Kolter (2018)). On the other hand, in comparison to the methods based on SDP relaxations (Raghunathan et al., 2018b) our model performs over 2000 times faster while still providing competitive bounds.
2. Unlike ϵ-verification techniques, we provide a lower bound on DtDB **without an initial guess and without relying on computing bounds for the neuron activation values** in each layer. Such bounds should be tight enough and play an important role in other relaxation techniques such as the SDP based approaches by Raghunathan et al. (2018b); Dvijotham et al. (2019) (that start from the same observation as we do, but proceed with different relaxations).
3. Additionally to computing the lower bounds after solving a single QP, our approach allows to make further iterations converging to an adversarial example. One of the main limitations of inexact approaches is the fact that the situation around the given anchor point outside of the certified radius remains unclear. Since these approaches rely on relaxations of the initial problem (and do not solve it exactly), failed certification does not indicate the presence of an adversarial example. We present **the first bi-directional robustness verification technique** (besides the exact computation of DtDB, which is intractable), that includes a built-in mechanism to construct upper bounds and assess the quality of the lower bounds without externally constructing adversarial examples.

2 Notation and idea

We consider a neural network consisting of L linear transformations and $L-1$ ReLU activations (no activation after the last hidden layer). The number of neurons in layer l is denoted as n_l for $l = 0, \dots, L$, meaning that the data has n_0 features and n_L classes. Furthermore we present our analysis for the l_2-norm as perturbation magnitude measure since only few available methods are applicable for this setting. To make our method comparable with Raghunathan et al. (2018b) a generalization to the l_∞-perturbations is addressed in Appendix A.

Given sample $x^0 \in \mathbb{R}^{n_0}$, weight matrices $W^l \in \mathbb{R}^{n_l \times n_{l-1}}$, and bias vectors $b^l \in \mathbb{R}^{n_l}$, we define the output of the i-th neuron in the l-th layer after the ReLU activation as

$$\begin{aligned} x_i^l &= \left[W_i^l x^{l-1} + b_i^l\right]_+ \text{ and} \\ f_i(x^0) = x_i^L &= W_i^L x^{L-1} + b_i^L \end{aligned} \tag{1}$$

where $[x]_+$ is the positive part of x and $f(x^0) = x^L$ denotes the output of the complete forward pass through the network. We start with an observation that the ReLU activation function acts as one of two linear functions depending on its input's sign. That allows us to reformulate it (see section 3.1) and obtain an optimization problem with so called linear complementarity constraints (also used by Raghunathan et al. (2018b); Dvijotham et al. (2019) for ϵ-verification). Note that for each pair of scalars a and b the following holds where (2) is known as complementarity condition for b and $b - a$.

$$b = [a]_+ \iff b \geq 0,\ b - a \geq 0,\ b(b-a) = 0 \tag{2}$$

3 Optimization problem

3.1 Formulation of DtDB

For a given sample $\tilde{x}^0$, pre-trained NN classifier f, predicted label $\tilde{y}$ and a target label y we aim to find the closest point to $\tilde{x}^0$ in $\mathbb{R}^{n_0}$ that has a larger or equal probability to be classified as y compared to the initial label. This can be done by solving the following optimization problem.

$$\min_{x^0 \in \mathbb{R}^{n_0}} \|x^0 - \tilde{x}^0\|^2, \text{ s.t. } (e_{\tilde{y}} - e_y)^T f(x^0) \leq 0, \tag{DtDB}$$

where e_i is the i-th unit vector in $\mathbb{R}^{n_L}$ and $\|x\|$ denotes the Euclidean norm of x. Note that we explore the boundary between classes y and $\tilde{y}$ around $\tilde{x}^0$. To compute the distance from $\tilde{x}^0$ to the (full) decision boundary, one needs t compute the solution of the problem for all target labels $y = 1, \ldots, n_L$ except $\tilde{y}$.

We can "unfold" the above optimization problem using (1), where x denotes a container with all variables $x^0, \ldots, x^L$.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \|x^0 - \tilde{x}^0\|^2, \text{ s.t. } (e_{\tilde{y}} - e_y)^T x^L \leq 0, \quad x^L = W^L x^{L-1} + b^L \\ x^l = ReLU(W^l x^{l-1} + b^l) \text{ for } l = 1, \ldots, L-1 \end{aligned}$$

We apply (2) to reformulate the problem and eliminate x^L, such that from now on $n = n_0 + \ldots + n_{L-1}$ and x contains only the remaining variables $x^0, \ldots, x^{L-1}$.

$$\min_{x \in \mathbb{R}^n} \|x^0 - \tilde{x}^0\|^2, \text{ s.t. } (e_{\tilde{y}} - e_y)^T \left(W^L x^{L-1} + b^L\right) \leq 0 \tag{DtDB}$$

$$\left(x^l\right)^T \left(x^l - \left(W^l x^{l-1} + b^l\right)\right) = 0 \text{ for } l = 1, \ldots, L-1 \tag{3}$$

$$x^l - \left(W^l x^{l-1} + b^l\right) \geq 0,\ x^l \geq 0 \text{ for } l = 1, \ldots, L-1 \tag{4}$$

3.2 QP Relaxation

Next we consider a Lagrangian relaxation of DtDB without constraints (3). We will refer to the resulting problem as QPRel(λ), when the explicit dependency on the multipliers is required.

$$\min_{x \in \mathbb{R}^n} \|x^0 - \tilde{x}^0\|^2 + \sum_{l=1}^{L-1} \lambda_l \left(x^l\right)^T \left(x^l - \left(W^l x^{l-1} + b^l\right)\right), \text{ s.t.} \tag{QPRel}$$

$$(e_{\tilde{y}} - e_y)^T \left(W^L x^{L-1} + b^L\right) \leq 0 \tag{5}$$

$$x^l - \left(W^l x^{l-1} + b^l\right) \geq 0,\ x^l \geq 0 \text{ for } l = 1, \ldots, L-1 \tag{6}$$

The obtained problem is indeed a QP with linear constraints. We need to clarify two questions. How does the problem QPRel help us solving DtDB and how can this problem be itself solved efficiently?

QPRel vs. DtDB First, we describe properties of the solution of QPRel providing information about DtDB. For that we define for arbitrary L vectors $x^0 \in \mathbb{R}^{n_0}, \dots, x^{L-1} \in \mathbb{R}^{n_{L-1}}$ and $\lambda \in \mathbb{R}_+^{L-1}$

$$c(x, \lambda) := \sum_{l=1}^{L-1} \lambda_l \left(x^l\right)^T \left(x^l - \left(W^l x^{l-1} + b^l\right)\right)$$

as the corresponding propagation gap. It follows directly from the definition that for arbitrary non-negative λ it holds that:

- if x is feasible for DtDB we have $c(x, \lambda) = 0$ meaning that x comes from x^0 propagated through the NN as defined in (1),
- if x is feasible for QPRel we have $c(x, \lambda) \geq 0$ meaning that there might be a slack between the true output of layer l when getting x^0 as an input and the value of x^l.

In general the following holds for the relation between the solution of QPRel and DtDB (see also Figure 1, every point in the hatched area is certified to be classified belonging to the class $\tilde{y}$).

Lemma 1. *Denote the solution of QPRel by $\hat{x}_{qp}$ and the square root of its optimal objective value by $\hat{d}_{qp}$, let $\hat{d}$ be the square root of the optimal objective value of DtDB. The following holds:*

1. *$\hat{d}_{qp} \leq \hat{d}$ and when $c(\hat{x}, \lambda) = 0$ we have $\hat{d}_{qp} = \hat{d}$ and $\hat{x}$ is optimal for DtDB.*
2. *For two non-negative λ^1, λ^2 with $\lambda^1 \leq \lambda^2$ elementwise it holds that $\hat{d}_{qp}(\lambda^1) \leq \hat{d}_{qp}(\lambda^2)$.*

The first result from Lemma 1 ensures that $\hat{d}_{qp}$ provides a radius of a certified neighborhood around the anchor point. Whereas the second part indicates that we should choose λ as large as possible to get our lower bound closer to DtDB. Unfortunately, as we show below, the problem QPRel becomes non-convex for large values of λ. While one could try to tackle a non-convex QP with proper optimization methods, we will address necessary conditions such that QPRel is guarantied to be convex and can be solved efficiently next.

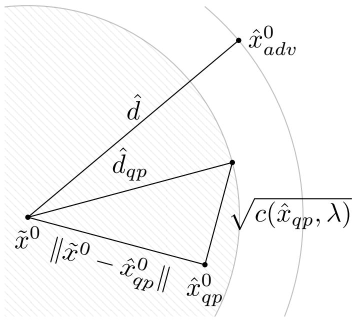


Figure 1: Setting of the optimal solutions for DtDB $\hat{x}_{adv}$ and QPRel $\hat{x}_{qp}$.

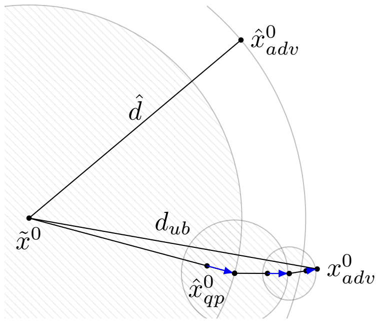


Figure 2: Graphical representation of Algorithm 1.

Convexity of QPRel To look into the problem QPRel in more detail we introduce the Hessian M^λ (which is a constant matrix) of its objective function.

Definition 1. Let $E_l \in \mathbb{R}^{n_l \times n_l}$ be the identity matrix of the corresponding dimension. We define $M^\lambda \in \mathbb{R}^{n \times n}$ as the following symmetric block tridiagonal matrix

$$M^\lambda := \begin{pmatrix} D_0 & M_1^T & & \\ M_1 & D_1 & M_2^T & \\ & M_2 & \ddots & \ddots \\ & & \ddots & D_{L-1} \end{pmatrix} \text{ with } M_l := -\frac{1}{2}\lambda_l W^l, D_l := \lambda_l E_l \text{ and } \lambda_0 := 1.$$

Using this matrix we can rewrite the objective function from QPRel as (see Appendix C, Lemma 2):

$$\min_{x \in \mathbb{R}^n} x^T M^\lambda x + x^T \bar{b}(\lambda, \tilde{x}^0) + \bar{c}(\tilde{x}^0), \text{ s.t. (5), (6)}$$

where $\bar{b}$ and $\bar{c}$ influence only the linear and constant terms and are therefore not relevant in this section. From this reformulation we can clearly see that the matrix M^λ determines the (non-)convexity of the objective function. The following theorem provides sufficient and necessary conditions on λ depending on the weights W_l assuring that M^λ is positive semi-definite. Finally, it allows us to use off-the-shelf QP-solvers with excellent convergence properties.

Theorem 1. *Let $W^1, \ldots, W^{L-1}$ be the weights of an arbitrary pre-trained NN and $\|W\|$ the spectral norm of an arbitrary matrix. Then the following two conditions for λ provide correspondingly a sufficient and a necessary criterion for the matrix M^λ to be positive semi-definite.*

$$\textit{(suf. condition)} \qquad \lambda_1 \leq \frac{2\lambda_0}{\|W^1\|^2} \textit{ and } \lambda_l \leq \frac{\lambda_{l-1}}{\|W^l\|^2} \qquad \textit{for } l = 2, \ldots, L-1 \tag{7}$$

$$\textit{(nec. condition)} \qquad \lambda_l \leq \frac{4\lambda_{l-1}}{\|W^l\|^2} \qquad \textit{for } l = 1, \ldots, L-1 \tag{8}$$

Further, we define $\underline{\lambda}$ and $\bar{\lambda}$ that correspondingly satisfy conditions (7) *and* (8) *with equality:*

$$\underline{\lambda}_l = 2 \prod_{k=1}^{l} \frac{1}{\|W^k\|^2}, \quad \bar{\lambda}_l = 4^l \prod_{k=1}^{l} \frac{1}{\|W^k\|^2}.$$

In case with a single hidden layer M^λ with $\lambda = \bar{\lambda}$ from (8) *is guaranteed to be positive-semi definite.*

We use (7), (8) and our previous results as guidelines to the choice of λ. Since QPRel(λ) is monotonous in the sense of Lemma 1 we perform a binary search between $\underline{\lambda}$ and $\bar{\lambda}$ from Theorem 1 to find the point closest to $\bar{\lambda}$ such that the QP remains convex. This preprocessing step does not considerably affect the runtime since checking whether a matrix is positive semi-definite is done efficiently by Cholesky decomposition. However, it improves the final bounds by up to a factor two comparing to the bounds when using $\lambda = \underline{\lambda}$ without the binary search.

Note that this procedure has to be done *once* for a given classifier. The obtained λ can subsequently be used to solve QPRel for *all* anchor points and target labels. This gives a strong advantage regarding the computational complexity compared to the application of SDP based ϵ-verification procedures. For example, the method by Dvijotham et al. (2019) includes the dual multipliers as variables into the SDP problem that has to solved for each combination of the anchor point, target label and an epsilon to verify.

4 Upper bounds

Now we describe an iterative procedure generating a sequence of points in $\mathbb{R}^{n_0}$ converging to a point on the decision boundary. It provides an upper bound on the distance to it and allows in combination with the lower bound a bi-directional verification of a classifier. The idea is to iteratively explore the neighborhood of points that we obtain from the solution of QPRel starting at $\tilde{x}_0$. In each step after solving QPRel we choose an anchor point for the next iteration by projecting the obtained solution x^0 onto the boundary of the verified region along the ray from the previous anchor point. Algorithm 1 provides a pseudo-code for this procedure.

Figure 2 illustrates how the constructed sequence of anchor points and solutions of QPRel might look like for the setting of Figure 1. Here only the first three steps are shown assuming that we would reach a small value for the propagation gap in the last step. The blue arrow shows how we proceed from the solution of QPRel to the next anchor point in each iteration (compare to the update formula from the line 7 in Algorithm 1). Empirical findings show that the number of iterations that is necessary to achieve a reasonably small propagation gap (see next section) remains at about 3 steps. Furthermore the resulting point always lies on the decision boundary such that its distance to the anchor point provides an excellent measure of how far the true DtDB can be from the previously obtained lower bound. However, further analytic investigation of this algorithm including a convergence proof remains future work.

Algorithm 1: Algorithm to construct an adversarial example by iteratively solving QPRel.

```
Data: $\tilde{x}_0$, trained NN and $\lambda$ such that $M^\lambda$ is positive semi-definite, $tol$.
Result: $d_{ub}, x^0_{adv}$
1  begin
2      $x^0 \longleftarrow \tilde{x}^0$; $c \longleftarrow +\infty$
3      while $c > tol$ do
4          $\hat{x}_{qp} \longleftarrow$ optimal solution of QPRel($\lambda$) solved from the anchor point $x^0$
5          $\hat{d}_{qp} \longleftarrow$ optimal objective value of QPRel($\lambda$) solved from the anchor point $x^0$
6          $c \longleftarrow c(\hat{x}_{qp}, \lambda)$
7          $x^0 \longleftarrow x^0 + \hat{d}_{qp} \frac{\hat{x}^0_{qp} - x^0}{\|\hat{x}^0_{qp} - x^0\|}$, chose the next anchor on the boundary of the certified region
8      end
9      $x^0_{adv} \longleftarrow \hat{x}^0_{qp}$; $d_{ub} \longleftarrow \|\tilde{x}^0 - \hat{x}^0_{qp}\|$
10 end
```

5 Experiments

In this section we report and discuss the results we obtained by testing our method of verifying deep NN classifiers from both directions on different datasets. The main focus lies on the quality of the obtained lower and upper bounds in comparison to the other approaches.

5.1 Setup

We have two hyper-parameters for tuning our method. λ is chosen for each classifier according to Theorem 1 and the discussion afterwards such that an accuracy level of 10^{-6} is achieved during the binary search. The second parameter tol represents the largest value of the propagation gap that we assume to be small enough such that Algorithm 1 terminates. We set for all tests $tol = 10^{-4}$. Other methods are tested with the default settings as provided in the corresponding repositories unless we mention a change. Runtime of the compared algorithms is shown in Appendix B. To solve the QP tasks we use Gurobi Optimization (2018).

Datasets and classifiers The experiments are performed using the MNIST and Fashion-MNIST datasets scaled such that the feature values lie in $[0, 1]$ interval. For each of the datasets we use the correctly classified samples from 1200 train points or 1250 test points (labels distribution preserved) to evaluate the verification approaches. The exact number of used samples is reported below for each experiment.

For classification we take ReLU networks consisting of dense linear layers. The architectures we used are named by the number of hidden layers and the overall number of hidden neurons: **L1N50R**, **L1N100**, **L1N300** and **L2N100** with the same number of neurons in each hidden layer. We use normally trained classifiers as well as a robustly trained one (with suffix **R**) using the method by Wong and Kolter (2018) with $\epsilon = 0.1$ in l_2-setting. The weights will be available in the project repository.

Competitors We compare our approach **QPRel** with the following ϵ-verification methods: **ConvAdv** by Wong and Kolter (2018) based on the LP relaxation of ReLU constraints (we use its implementation supporting the l_2-norm by Croce et al. (2019)), **CROWN** by Zhang et al. (2018) which is a layerwise bound propagation technique including performance boosting quadratic approximations and warm start (for each setting we report only the best result from these two competitors), and **SDPRel** by Raghunathan et al. (2018b) based on a SDP relaxation solved by MOSEK (only for MNIST test data, **L1N50R** net and l_∞-setting). Finally, we take **FGSM** attack by Goodfellow et al. (2015) as a competitor for **QPRel-UB**. We use the notation **QPRel-LB** or **QPRel-UB**, when we want to emphasize what bounds are meant.

5.2 Results

State-of-the-art bounds on compact networks First we qualitatively describe the results for the MNIST training dataset and **L1N100** classifier in comparison with **CROWN** equipped with their built-in algorithm to find the largest verifiable ϵ. Figure 3a shows the computed lower and upper bounds for the correctly classified samples. We sorted the samples in increasing order of the obtained **QPRel-LB** to see the relation to other values more clearly. The bounds computed by **QPRel** are better in comparison to the competitors in average and for most individual images. The relative difference (**QPRel-LB** minus **CROWN** and then divided by **CROWN**, positive if our bounds are better) between the computed bounds is shown as a histogram on Figure 3b including the 10%-, 50%- and 90%-quantile. For 50% of the considered MNIST images we have improved the lower bound from **CROWN** by over 82.4%. In Table 1 the median of the relative improvement is reported in column *"MedRelDiff to QPRel (%)"* (e.g. you will find the just mentioned result in the second row).

Tightness of the bi-directional verification The second measure of interest is the ratio of the verifiable samples (VerRatio) both as robust and non-robust. In case when only a lower bound on DtDB is available the user does not get any information when certification fails. That means the information obtained from the lower bounds only will decrease if we try to certify a model against stronger perturbations. To illustrate this behavior dependent on the maximal allowed perturbation norm ϵ we plot the ratio of verifiable samples against ϵ, that is for how many samples the obtained lower bound is larger than ϵ providing a robustness certificate (see Figure 3c). Since our lower bounds are tighter the decay in VerRatio is not as steep as for **CROWN**.

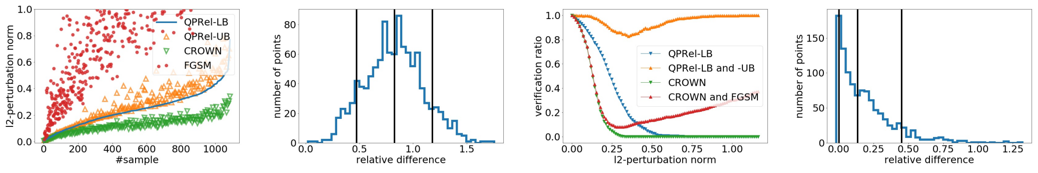
For the other settings we observe the same results, as summarized in Table 1. There we report the largest perturbation magnitude such that 50% of the data can still be verified as robust by the lower bounds (column *"ϵ to hit 50% VerRatio"*). That is, larger numbers are better.

We also compute the upper bounds on DtDB using **QPRel-UB** such that samples with an upper bound smaller than ϵ can be verified as non-robust. The competing approaches are equipped with the **FGSM** attack providing an alternative upper bound. We see that the verification ratio of **QPRel-LB** together with **QPRel-UB** never drops below 82.8% for the MNIST training dataset and **L1N100** classifier in comparison to the worst case of 8.1% for **CROWN+FGSM**. You can find the smallest bi-directional verification ratio in Table 1 in column *"VerRatio worst (%)"*. Finally, to assess the quality of the

Table 1: Better bounds and verification ratio for the compact networks (train data, l_2-perturbations)

Setting				MedRelDiff	ϵ to hit 50%	VerRatio	MedRelDiff
Data	L/N	NrPts	Method	to QPRel (%)	VerRatio	worst (%)	UB-LB (%)
MNIST	1/100	1086	QPRel-LB	–	**0.246**	**82.8**	**13.4**
	1/100	1086	CROWN	**+82.4**	0.129	8.1	1033.7
	2/100	965	QPRel-LB	–	**1.174**	**99.3**	**<0.01**
	2/100	965	ConvAdv	**+30.1**	0.891	23.3	277.9
	1/300	1142	QPRel-LB	–	**0.243**	**94.0**	**5.5**
	1/300	1142	CROWN	**+149.1**	0.101	10.9	627.8
	1/50	1180	QPRel-LB	–	**0.872**	**82.4**	**14.3**
	1/50	1180	CROWN	**+67.0**	0.506	8.8	419.4
F-MNIST	1/100	1083	QPRel-LB	–	**0.252**	**81.9**	**17.2**
	1/100	1083	CROWN	**+49.1**	0.159	15.9	678.8
	2/100	924	QPRel-LB	–	**0.332**	**83.0**	**20.3**
	2/100	924	ConvAdv	**+68.0**	0.205	22.2	375.1
	1/300	1094	QPRel-LB	–	**0.183**	**78.8**	**23.1**
	1/300	1094	CROWN	**+56.6**	0.122	12.8	902.2

lower bounds we look onto the relative gap between them and the computed upper bounds (discussed in Section 4). On Figure 3d we show the histogram and also report the median value of the relative gap between **QPRel-LB** and **QPRel-UB** (see the column *"MedRelDiff UB-LB (%)"* in Table 1). We see that for 50% of the considered data points the relative gap is below 13.4% (first row, last column in Table 1). In a few other settings the gap between **QPRel-LB** and **QPRel-UB** vanishes resulting in over 99% VerRation (see the third row in Table 1). That means the computed lower bound is almost the exact DtDB for the majority of the samples. Note that no exact method computing DtDB via MIP techniques is used to achieve that.



(a) Bounds (b) QPRel-LB vs. CROWN (c) Verification ratio (d) QPRel-UB vs. -LB

Figure 3: Comparison with **CROWN** for **L1N100** on MNIST training dataset, l_2-setting.

For other datasets we report the same measures: number of samples we used for tests, median of the relative difference between lower bounds of **QPRel-LB** and the corresponding approach, ϵ where we reach VerRatio of 50%, minimal VerRatio for bi-directional certification with upper bounds and median of the relative difference between the lower and the upper bounds. In the supplementary material you can find similar plots as on Figures 3a - 3d for other classifiers and datasets. Complete results are shown in Appendix D.

Comparison with SDP-relaxations in l_∞-setting In order to compare our method with Raghunathan et al. (2018b) we generalize **QPRel** to l_∞-setting as described in Appendix A. Note, that the resulting relaxation is looser then the initial QPRel for the l_2-setting since we introduce an additional penalty term to make the problem convex. That leads to worse results shown in Tables 9 and 10. To compute the largest ϵ such that the SDP verification succeeds we perform a binary search between the lower bounds ϵ_{min} computed by **QPRel-LB** and $\epsilon_{max} = 1.0$ which is the maximal perturbation for the l_∞-norm on images. Since this approach takes longer to run we test it only on the **L1N50R** net and MNIST test data. Further, we speed up this approach by modifying MOSEK parameters (see Table 6 for details) such that the optimization procedure terminates earlier (approximately after a half of the usual number of iterations). We can still rely on the obtained results since we are not interested in the exact value of the SDP objective, but only whether it is positive or negative which was observed to be determined far sooner during the solution process then when the solver would reach a true optimum.

While **QPRel-LB** in this setting still preforms better then **CROWN** and **ConvAdv** in average, our bounds are about 1.54 times larger then by **SDPRel** (see Table 10), but computed about 2000 times faster (see Appendix B). This shows that the QP relaxation is not well suited for obtaining tight bounds in l_∞-setting as already indicated by the arguments above.

6 Conclusion

In this work we present a novel approach to solve the problem of approximating the minimal adversarial perturbations for ReLU networks based in a convex QP relaxation of DtDB. We show that the lower bounds computed with **QPRel** improve the results of the available LP based methods and allow certification of larger neighborhoods. Since convexity of the underlying QP determines computational efficiency of our approach we derive the necessary and sufficient conditions on the Lagrangian multipliers for it. Additionally, a method that constructs a sequence of points converging to the point on the decision boundary provides us upper bounds on DtDB. Quality of the obtained bounds in the l_2-setting is shown to be good enough to verify consistently over 80% of the considered samples for MNIST and FashionMNIST data for all values of the perturbation magnitude, while competitors' verification ratio might drop below 10%.

With our contribution we make a step towards robustness verification of deep ReLU-based classifiers. To be able to apply the approach on a wider class of networks it should be generalized to popular architectures beyond dense linear layers. Good upper bounds obtained from Algorithm 1 indicate further intriguing research direction of studying its properties.

REFERENCES

Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems 31*, pages 4790–4799. Curran Associates, Inc., 2018.

Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. *AISTATS*, 2019.

Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods*, pages 121–138. Springer International Publishing, 2018.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A. Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2018. URL http://auai.org/uai2018/proceedings/papers/204.pdf.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Chongli Qin, Soham De, and Pushmeet Kohli. Efficient neural network verification with exactness characterization. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2019. URL http://auai.org/uai2019/proceedings/papers/164.pdf.

Rüdiger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *Automated Technology for Verification and Analysis*, pages 269–286. Springer International Publishing, 2017.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018. URL http://www.gurobi.com.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems 30*, pages 2266–2276. Curran Associates, Inc., 2017.

Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification*, pages 97–117. Springer International Publishing, 2017.

Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward ReLU neural networks. *arXiv e-prints*, art. arXiv:1706.07351, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018a.

Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems 31*, pages 10877–10887. Curran Associates, Inc., 2018b.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *CoRR*, abs/1711.07356, 2017. URL http://arxiv.org/abs/1711.07356.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems 31*, pages 6541–6550. Curran Associates, Inc., 2018.

Lily 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 *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5276–5285. PMLR, 2018.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5286–5295. PMLR, 2018.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems 31*, pages 4939–4948. Curran Associates, Inc., 2018.

A Generalization to l_∞

For comparison with Raghunathan et al. (2018b) we show how our method can be applied to compute bounds on the distance to the closest adversarial measured using the l_∞ norm. A straight forward way would be to modify the objective function accordingly. By introducing a new variable m representing $\|x^0 - \tilde{x}^0\|_\infty^2 = \max_i |x_i^0 - \tilde{x}_i^0|^2$ and n_0 new quadratic constraints we get the following versions of QPRel. Note that the quadratic constraints do not harm the complexity since they describe a convex cone and can be handled by the QP-solvers including *gurobi*.

$$\begin{aligned}\min_{x\in\mathbb{R}^n,\ m\in\mathbb{R}} & m + c(x,\lambda), \text{ s.t.}\\ & (x_i^0 - \tilde{x}_i^0)^2 \le m,\ i = 1,\dots,n_0\\ & (e_{\tilde{y}} - e_y)^T \left(W^L x^{L-1} + b^L\right) \le 0\\ & x^l - \left(W^l x^{l-1} + b^l\right) \ge 0,\ x^l \ge 0\end{aligned}$$

While this formulation is of a similar structure as the QPRel (quadratic objective and linear plus quadratic constraints), the Hessian of the objective function is not positive semi-definite for any value of λ. Since $c(x,\lambda)$ is the only source of quadratic terms now (squared distance to the anchor point is now replaced by m), the new M^λ is of the same form as in Lemma 2, but with $\lambda_0 = 0$. To see that we cannot affect the convexity of the objective function by the parameter λ anymore consider vector x with an arbitrary $x^0 \in \mathbb{R}^{n_0}$, $x^1 = \alpha W^1 x^0$ with $0 < \alpha < 1$ and $x^l = 0$ for $l > 1$. Then

$$x^T M^\lambda x = \lambda_1 \left(\|x^1\|^2 - \left(x^1\right)^T W^1 x^0\right) = \lambda_1(\alpha^2 - \alpha)\|W^1 x^0\|^2 < 0$$

meaning that M^λ cannot be positive semi-definite.

To overcome this issue we utilize the new quadratic constraints. We return back to a convex QP by considering the following problem with a positive μ.

$$\begin{aligned}\min_{x\in\mathbb{R}^n,\ m\in\mathbb{R}} & m + c(x,\lambda) + \mu \sum_{i=1}^{n_0} \left((x_i^0 - \tilde{x}_i^0)^2 - m\right), \text{ with}\\ & (x_i^0 - \tilde{x}_i^0)^2 \le m,\ i = 1,\dots,n_0\\ & (e_{\tilde{y}} - e_y)^T \left(W^L x^{L-1} + b^L\right) \le 0\\ & x^l - \left(W^l x^{l-1} + b^l\right) \ge 0,\ x^l \ge 0\end{aligned}$$

Clearly, for $0 < \mu \le n_0^{-1}$ the solution of this problem is a finite lower bound on DtDB with the l_∞-norm. On the other side we are back in the setting of Theorem 1 with $\lambda_0 = \mu$ allowing us to use the same framework as before. Results in section 5 for the l_∞-setting were obtained by solving this problem with $\mu = (2n_0)^{-1}$.

B Runtime and number of iterations

Tables 2, 3, 4 and 5 show the average runtime and its standard deviation for all considered settings as well as the number of iterations Algorithm 1 needs to terminate. During the binary search procedure we apply with **SDPRel** we always make 10 bisection steps. All tasks necessary for the computation of bounds on DtDB for one sample are run on a single CPU (including the solution of QPs and SDPs with Gurobi Optimization (2018) and MOSEK respectively). Here, we do not consider **ConvAdv** as **CROWN** performs always faster. From the comparison of the time for the computation of our lower bounds (column *Runtime-LB (s)*) and the overall runtime (column *Runtime (s)*) we conclude that obtaining the upper bounds takes $n-1$ times longer than solving the initial QP where n is approximately the overall number of the iterations in Algorithm 1. While in almost all considered settings n remains smaller than 4 it might become a limiting factor when applying **QPRel-UB** on larger networks. That means further investigation of the convergence properties of Algorithm 1 is necessary for generalizing the described method. From the comparison of **QPRel-LB** and **CROWN** we see the clear advantage of the latter since it doesn't involve any optimization task. However, this advantage comes in cost of the verification properties as discussed above. On the other hand, **SDPRel** with a binary search provides better bounds, but is about 2000 times slower then **QPRel-LB** (see the last line in the MNIST section in Table 5).

Table 2: Runtime comparison, train data, l_2-setting

Setting				Runtime-LB (s)		Runtime-Full (s)		Nr. iterations(-UB)	
Data	L/N	NrPts	Method	mean	std	mean	std	mean	std
MNIST	1/100	1086	QPRel	1.467	0.142	4.200	2.091	2.845	1.416
	1/100	1086	CROWN	0.021	0.002	–	–	–	–
	2/100	965	QPRel	0.280	0.017	0.472	0.140	1.679	0.484
	2/100	965	CROWN	0.034	0.013	–	–	–	–
	1/300	1142	QPRel	5.421	0.309	12.631	3.414	2.317	0.629
	1/300	1142	CROWN	0.064	0.016	–	–	–	–
	1/50	1180	QPRel	0.253	0.018	0.805	0.257	3.176	1.002
	1/50	1180	CROWN	0.017	0.006	–	–	–	–
F-MNIST	1/100	1083	QPRel	1.441	0.156	4.534	4.124	3.142	2.687
	1/100	1083	CROWN	0.025	0.009	–	–	–	–
	2/100	924	QPRel	0.289	0.019	0.992	0.469	3.431	1.600
	2/100	924	CROWN	0.031	0.012	–	–	–	–
	1/300	1094	QPRel	5.385	0.317	18.782	12.443	3.503	2.313
	1/300	1094	CROWN	0.063	0.018	–	–	–	–

Table 3: Runtime comparison, test data, l_2-setting

Setting				Runtime-LB (s)		Runtime-Full (s)		Nr. iterations(-UB)	
Data	L/N	NrPts	Method	mean	std	mean	std	mean	std
MNIST	1/100	1137	QPRel	1.451	0.142	4.225	1.994	2.898	1.342
	1/100	1137	CROWN	0.024	0.009	–	–	–	–
	2/100	1028	QPRel	0.280	0.018	0.473	0.141	1.683	0.486
	2/100	1028	CROWN	0.033	0.012	–	–	–	–
	1/300	1166	QPRel	5.353	0.328	12.954	3.386	2.398	0.619
	1/300	1166	CROWN	0.059	0.017	–	–	–	–
	1/50	1212	QPRel	0.257	0.018	0.817	0.256	3.183	0.993
	1/50	1212	CROWN	0.016	0.006	–	–	–	–
F-MNIST	1/100	1061	QPRel	1.456	0.148	4.635	4.186	3.161	2.676
	1/100	1061	CROWN	0.024	0.008	–	–	–	–
	2/100	949	QPRel	0.276	0.022	0.985	0.455	3.557	1.613
	2/100	949	CROWN	0.031	0.012	–	–	–	–
	1/300	1069	QPRel	5.406	0.301	19.585	12.080	3.638	2.251
	1/300	1069	CROWN	0.058	0.016	–	–	–	–

C Proofs

Lemma 1. *Denote the solution of QPRel by $\hat{x}_{qp}$ and the square root of its optimal objective value by $\hat{d}_{qp}$, let $\hat{d}$ be the square root of the optimal objective value of DtDB. The following holds:*

1. *$\hat{d}_{qp} \leq \hat{d}$ and when $c(\hat{x}, \lambda) = 0$ we have $\hat{d}_{qp} = \hat{d}$ and $\hat{x}$ is optimal for DtDB.*
2. *For two non-negative λ^1, λ^2 with $\lambda^1 \leq \lambda^2$ elementwise it holds that $\hat{d}_{qp}(\lambda^1) \leq \hat{d}_{qp}(\lambda^2)$.*

Proof. Assume $\hat{x}_{adv}$ is the optimal solution of DtDB. Then it is an admissible point of QPRel as well and $c(\hat{x}_{adv}, \lambda) = 0$ since $\hat{x}^l_{adv} = W^l \hat{x}^{l-1}_{adv}$ for $l = 1, \dots, L-1$. Since $\hat{x}_{qp}$ is optimal for QPRel and

Table 4: Runtime comparison, train data, l_∞-setting

Setting				Runtime-LB (s)		Runtime-Full (s)		Nr. iterations(-UB)	
Data	L/N	NrPts	Method	mean	std	mean	std	mean	std
MNIST	1/100	1086	QPRel	6.202	1.283	19.104	63.168	2.752	8.231
	1/100	1086	CROWN	0.019	0.005	–	–	–	–
	2/100	965	QPRel	1.392	0.143	4.110	1.738	2.988	1.293
	2/100	965	CROWN	0.029	0.011	–	–	–	–
	1/300	1142	QPRel	10.470	0.854	14.567	7.457	1.400	0.732
	1/300	1142	CROWN	0.054	0.017	–	–	–	–
	1/50	1180	QPRel	2.634	0.916	16.813	19.141	5.818	3.660
	1/50	1180	CROWN	0.016	0.006	–	–	–	–
F-MNIST	1/100	1083	QPRel	5.750	1.314	18.447	66.807	2.651	7.627
	1/100	1083	CROWN	0.018	0.004	–	–	–	–
	2/100	924	QPRel	1.500	0.179	3.496	2.978	2.348	1.954
	2/100	924	CROWN	0.026	0.010	–	–	–	–
	1/300	1094	QPRel	10.636	0.882	11.541	4.031	1.085	0.379
	1/300	1094	CROWN	0.053	0.017	–	–	–	–

Table 5: Runtime comparison, test data, l_∞-setting

Setting				Runtime-LB (s)		Runtime-Full (s)		Nr. iterations(-UB)	
Data	L/N	NrPts	Method	mean	std	mean	std	mean	std
MNIST	1/100	1137	QPRel	6.208	1.250	22.898	75.986	3.145	8.924
	1/100	1137	CROWN	0.020	0.007	–	–	–	–
	2/100	1028	QPRel	1.321	0.145	3.912	1.619	3.002	1.266
	2/100	1028	CROWN	0.029	0.011	–	–	–	–
	1/300	1166	QPRel	10.512	0.844	15.852	8.343	1.515	0.808
	1/300	1166	CROWN	0.048	0.014	–	–	–	–
	1/50	1054	QPRel	2.683	1.041	17.439	16.539	6.102	3.307
	1/50	1054	CROWN	0.016	0.006	–	–	–	–
	1/50	1054	SDPRel	4338.017	949.572	–	–	10	0.000
F-MNIST	1/100	1061	QPRel	5.805	1.193	21.649	73.836	3.107	8.900
	1/100	1061	CROWN	0.020	0.007	–	–	–	–
	2/100	949	QPRel	1.454	0.164	3.407	2.689	2.352	1.827
	2/100	949	CROWN	0.025	0.009	–	–	–	–
	1/300	1069	QPRel	10.686	0.927	11.829	5.051	1.107	0.466
	1/300	1069	CROWN	0.048	0.014	–	–	–	–

$\hat{x}_{adv}$ is just its admissible point we get that

$$\hat{d}^2 = \|\hat{x}^0_{adv} - \tilde{x}^0\|^2 = \|\hat{x}^0_{adv} - \tilde{x}^0\|^2 + c(\hat{x}_{adv}, \lambda) \geq \|\hat{x}^0_{qp} - \tilde{x}^0\|^2 + c(\hat{x}_{qp}, \lambda) = \hat{d}^2_{qp}$$

proving the first claim. The second one follows from the fact that $c(x, \lambda)$ for a given x is a linear function of λ:

$$c(x, \lambda) = \lambda^T \begin{pmatrix} c(x, e_1) \\ \vdots \\ c(x, e_{L-1}) \end{pmatrix}$$

where each $c(x, e_l) = \left(x^l\right)^T \left(x^l - \left(W^l x^{l-1} + b^l\right)\right)$ is non-negative for admissible x because of the non-negativity constraints (4). Therefore the claim follows immediately from the assumption that

$\lambda_l^1 \leq \lambda_l^2$ for all l

$$c(x, \lambda^1) = \sum_{l=1}^{L-1} \lambda_l^1 c(x, e_l) \leq \sum_{l=1}^{L-1} \lambda_l^2 c(x, e_l) = c(x, \lambda^2).$$

□

Lemma 2. *Objective function of QPRel can be reformulated as*

$$d_{qp}(\lambda, x) = x^T M^\lambda x + x^T \bar{b}(\lambda, \tilde{x}^0) + \bar{c}(\tilde{x}^0)$$

where terms $\bar{b}$ and $\bar{c}$ don't depend on x.

Proof. The proof is done by sorting the quadratic, linear and constant terms in the objective function of the initial formulation.

$$\begin{aligned} d_{qp}(\lambda, x) &= \|x^0 - \tilde{x}^0\|^2 + \sum_{l=1}^{L-1} \lambda_l \left(x^l\right)^T \left(x^l - \left(W^l x^{l-1} + b^l\right)\right) \\ &= \left(x^0\right)^T x^0 - 2\left(x^0\right)^T \tilde{x}^0 + \|\tilde{x}^0\|^2 + \sum_{l=1}^{L-1} \lambda_l \Big(\left(x^l\right)^T x^l - \left(x^l\right)^T W^l x^{l-1} + \left(x^l\right)^T b^l \Big) \\ &= \underbrace{\sum_{l=0}^{L-1} \lambda_l \left(x^l\right)^T x^l - \sum_{l=1}^{L-1} \lambda_l \left(x^l\right)^T W^l x^{l-1}}_{\text{quadratic term}} \underbrace{-2\left(x^0\right)^T \tilde{x}^0 + \sum_{l=1}^{L-1} \lambda_l \left(x^l\right)^T b^l + \|\tilde{x}^0\|^2}_{\text{linear and constant terms}} \end{aligned}$$

From the quadratic term we can identify now the blocks of M^λ

$$\begin{aligned} &\text{diagonal: } M_{l,l}^\lambda = \lambda_l E_l && l = 0, \dots, L-1, \\ &\text{sub-diagonal: } M_{l+1,l}^\lambda = \left(M_{l,l+1}^\lambda\right)^T = -\frac{1}{2}\lambda_l W^l && l = 0, \dots, L-2. \end{aligned}$$

□

Theorem 1. *Let $W^1, \dots, W^{L-1}$ be the weights of an arbitrary pre-trained NN and $\|W\|$ the spectral norm of an arbitrary matrix. Then the following two conditions for λ provide correspondingly a sufficient and a necessary criterion for the matrix M^λ to be positive semi-definite.*

$$\text{(suf. condition)} \qquad \lambda_1 \leq \frac{2\lambda_0}{\|W^l\|^2} \text{ and } \lambda_l \leq \frac{\lambda_{l-1}}{\|W^l\|^2} \qquad \text{for } l = 2, \dots, L-1 \tag{7}$$

$$\text{(nec. condition)} \qquad \lambda_l \leq \frac{4\lambda_{l-1}}{\|W^l\|^2} \qquad \text{for } l = 1, \dots, L-1 \tag{8}$$

Further we define $\underline{\lambda}$ and $\bar{\lambda}$ that correspondingly satisfy conditions (7) *and* (8) *with equality (note that $\lambda_0 = 1$ uniquely determines those values). In case with a single hidden layer M^λ with $\lambda = \bar{\lambda}$ from* (8) *is guaranteed to be positive-semi definite.*

Proof. Let the assumptions hold and x be an arbitrary vector from $\mathbb{R}^n$. First we prove the *sufficient condition* by deriving a lower bound on $x^T M^\lambda x$ that is non-negative if (7) holds.

$$\begin{aligned}
x^T M^\lambda x &= \sum_{l=0}^{L-1} \lambda_l \|x^l\|^2 - \sum_{l=1}^{L-1} \lambda_l \left(x^l\right)^T W^l x^{l-1} \\
&= \frac{\lambda_0}{2}\|x^0\|^2 + \frac{\lambda_{L-1}}{2}\|x^{L-1}\|^2 \\
&\quad + \sum_{l=1}^{L-1} \frac{\lambda_l}{2}\|x^l\|^2 - \lambda_l \left(x^l\right)^T W^l x^{l-1} + \frac{\lambda_{l-1}}{2}\|x^{l-1}\|^2 \\
&= \frac{\lambda_0}{2}\|x^0\|^2 + \frac{\lambda_{L-1}}{2}\|x^{L-1}\|^2 \\
&\quad + \sum_{l=1}^{L-1} \frac{\lambda_l}{2}\|x^l\|^2 - \lambda_l \left(x^l\right)^T W^l x^{l-1} + \frac{\lambda_l}{2}\|W^l x^{l-1}\|^2 \\
&\quad + \sum_{l=1}^{L-1} \frac{\lambda_{l-1}}{2}\|x^{l-1}\|^2 - \frac{\lambda_l}{2}\|W^l x^{l-1}\|^2 \\
&= \frac{\lambda_0}{2}\|x^0\|^2 + \frac{\lambda_{L-1}}{2}\|x^{L-1}\|^2 + \sum_{l=1}^{L-1} \frac{\lambda_l}{2}\|x^l - W^l x^{l-1}\|^2 \\
&\quad + \sum_{l=1}^{L-1} \frac{\lambda_{l-1}}{2}\|x^{l-1}\|^2 - \frac{\lambda_l}{2}\|W^l x^{l-1}\|^2 \\
&= \frac{\lambda_{L-1}}{2}\|x^{L-1}\|^2 + \sum_{l=1}^{L-1} \frac{\lambda_l}{2}\|x^l - W^l x^{l-1}\|^2 + \\
&\quad + \left(\lambda_0\|x^0\|^2 - \frac{\lambda_1}{2}\|W^1 x^0\|^2\right) + \sum_{l=2}^{L-1} \frac{\lambda_{l-1}}{2}\|x^{l-1}\|^2 - \frac{\lambda_l}{2}\|W^l x^{l-1}\|^2 \\
&\geq \frac{\lambda_{L-1}}{2}\|x^{L-1}\|^2 + \sum_{l=1}^{L-1} \frac{\lambda_l}{2}\|x^l - W^l x^{l-1}\|^2 + \\
&\quad + \frac{1}{2}\left(2\lambda_0 - \lambda_1\|W^1\|^2\right)\|x^0\|^2 + \frac{1}{2}\sum_{l=2}^{L-1}\left(\lambda_{l-1} - \lambda_l\|W^l\|^2\right)\|x^{l-1}\|^2
\end{aligned}$$

where we applied the sub-multiplicativity property of the spectral norm, i.e. $\|W^l x^{l-1}\| \leq \|W^l\|\|x^{l-1}\|$, to obtain the last inequality. We see that under the assumption (7) on λ and W's it holds that

$$\lambda_1\|W^1\|^2 \leq 2\lambda_0 \text{ and } \lambda_l\|W^l\|^2 \leq \lambda_{l-1} \text{ for } l = 2, \ldots, L-1$$

and the lower bound on $x^T M^\lambda x$ in the last line is a sum of non-negative terms meaning that $x^T M^\lambda x \geq 0$ for all $x \in \mathbb{R}^n$.

To prove the *necessary condition* consider for each $l = 1, \ldots, L-1$ a special vector $\tilde{x}$ (we don't explicitly label it as dependent on l to avoid overloaded notation) which is everywhere zero except

$$\tilde{x}^{l-1} := \arg\max_{x \in \mathbb{R}^{n_{l-1}}} \frac{\|W^l x\|}{\|x\|} \text{ and } \tilde{x}^l := \frac{1}{2}W^l x^{l-1}.$$

For M^λ in order to be positive semi-definite it has to satisfy

$$\begin{aligned}
0 \leq \tilde{x}^T M^\lambda \tilde{x} &= \begin{pmatrix}\tilde{x}^{l-1} \\ \tilde{x}^l\end{pmatrix}^T \begin{pmatrix}\lambda_{l-1}E_{l-1} & -\frac{1}{2}\lambda_l\left(W^l\right)^T \\ -\frac{1}{2}\lambda_l W^l & \lambda_l E_l\end{pmatrix} \begin{pmatrix}\tilde{x}^{l-1} \\ \tilde{x}^l\end{pmatrix} \\
&= \lambda_{l-1}\|\tilde{x}^{l-1}\|^2 - \lambda_l\left(\tilde{x}^l\right)^T W^l \tilde{x}^{l-1} + \lambda_l\|\tilde{x}^l\|^2 \\
&= \lambda_{l-1}\|\tilde{x}^{l-1}\|^2 - \frac{1}{4}\lambda_l\|W^l\tilde{x}^{l-1}\|^2 = \left(\lambda_{l-1} - \frac{1}{4}\lambda_l\|W^l\|^2\right)\|\tilde{x}^{l-1}\|^2
\end{aligned}$$

which results in the necessary condition (8) as stated above. It remains to prove the sufficiency of (8) if the considered network contains one hidden layer. For that we can reuse the last computation and obtain now for an arbitrary $x \in \mathbb{R}^n$ that

$$\begin{aligned} x^T M^{\bar{\lambda}} x &= \lambda_0 \|x^0\|^2 - \lambda_1 \left(x^1\right)^T W^1 x^0 + \lambda_1 \|x^1\|^2 \\ &= \lambda_0 \|x^0\|^2 - \frac{1}{4}\lambda_1 \|W^1 x^0\|^2 + \lambda_1 \|\frac{1}{2} W^1 x^0 - x^1\|^2 \\ &\geq \left(\lambda_0 - \frac{1}{4}\lambda_1 \|W^1\|^2\right) \|x^0\|^2 + \lambda_1 \|\frac{1}{2} W^1 x^0 - x^1\|^2. \end{aligned}$$

We see that the last term remains non-negative in case of $\lambda_1 = \frac{4\lambda_0}{\|W^1\|^2}$ for all x. □

D TABLES

Table 6: MOSEK parameters we used to run **SDPRel** and their default values

Parameter [1]	New value	Default value
MSK_IPAR_NUM_THREADS	1	0
MSK_DPAR_INTPNT_CO_TOL_MU_RED	10^{-4}	10^{-8}
MSK_DPAR_INTPNT_CO_TOL_REL_GAP	10^{-4}	10^{-8}
MSK_DPAR_INTPNT_CO_TOL_INFEAS	10^{-6}	10^{-12}
MSK_DPAR_INTPNT_CO_TOL_DFEAS	10^{-4}	10^{-8}
MSK_DPAR_INTPNT_CO_TOL_PFEAS	10^{-4}	10^{-8}

[1] `https://docs.mosek.com/9.0/pythonapi/parameters.html` contains the full list of parameters including their description.

Table 7: Results for train data, l_2-perturbations (same as Table 1)

Setting				MedRelDiff	ϵ to hit 50%	VerRatio	MedRelDiff
Data	L/N	NrPts	Method	to QPRel (%)	VerRatio	worst (%)	UB-LB (%)
MNIST	1/100	1086	QPRel-LB	–	**0.246**	**82.8**	**13.4**
	1/100	1086	CROWN	**+82.4**	0.129	8.1	1033.7
	2/100	965	QPRel-LB	–	**1.174**	**99.3**	**<0.01**
	2/100	965	ConvAdv	**+30.1**	0.891	23.3	277.9
	1/300	1142	QPRel-LB	–	**0.243**	**94.0**	**5.5**
	1/300	1142	CROWN	**+149.1**	0.101	10.9	627.8
	1/50	1180	QPRel-LB	–	**0.872**	**82.4**	**14.3**
	1/50	1180	CROWN	**+67.0**	0.506	8.8	419.4
F-MNIST	1/100	1083	QPRel-LB	–	**0.252**	**81.9**	**17.2**
	1/100	1083	CROWN	**+49.1**	0.159	15.9	678.8
	2/100	924	QPRel-LB	–	**0.332**	**83.0**	**20.3**
	2/100	924	ConvAdv	**+68.0**	0.205	22.2	375.1
	1/300	1094	QPRel-LB	–	**0.183**	**78.8**	**23.1**
	1/300	1094	CROWN	**+56.6**	0.122	12.8	902.2

Table 8: Results for test data, l_2-perturbations

Setting				MedRelDiff	ϵ to hit 50%	VerRatio	MedRelDiff
Data	L/N	NrPts	Method	to QPRel (%)	VerRatio	worst (%)	UB-LB (%)
MNIST	1/100	1137	QPRel-LB	–	**0.268**	**81.9**	**14.6**
	1/100	1137	CROWN	**+84.6**	0.146	6.7	1097.2
	2/100	1028	QPRel-LB	–	**1.251**	**99.4**	**0.0**
	2/100	1028	ConvAdv	**+30.2**	0.959	23.6	281.8
	1/300	1166	QPRel-LB	–	**0.267**	**93.9**	**5.9**
	1/300	1166	CROWN	**+154.6**	0.111	7.0	658.5
	1/50	1212	QPRel-LB	–	**0.937**	**80.0**	**15.2**
	1/50	1212	CROWN	**+68.5**	0.562	7.4	421.8
F-MNIST	1/100	1061	QPRel-LB	–	**0.258**	**79.8**	**19.2**
	1/100	1061	CROWN	**+50.3**	0.164	13.7	703.0
	2/100	949	QPRel-LB	–	**0.355**	**79.9**	**21.3**
	2/100	949	ConvAdv	**+71.6**	0.213	19.3	375.2
	1/300	1069	QPRel-LB	–	**0.194**	**77.0**	**24.7**
	1/300	1069	CROWN	**+59.3**	0.122	10.6	890.9

Table 9: Results for train data, l_∞-perturbations

Setting				MedRelDiff	ϵ to hit 50%	VerRatio	MedRelDiff
Data	L/N	NrPts	Method	to QPRel (%)	VerRatio	worst (%)	UB-LB (%)
MNIST	1/100	1086	QPRel-LB	–	0.010	5.8	1478.1
	1/100	1086	CROWN	**+33.4**	0.010	**13.6**	**882.5**
	2/100	965	QPRel-LB	–	0.057	5.3	2484.0
	2/100	965	ConvAdv	**+16.8**	0.057	**32.4**	**220.4**
	1/300	1142	QPRel-LB	–	0.011	3.6	1691.4
	1/300	1142	CROWN	**+104.3**	0.011	**13.0**	**575.1**
	1/50	1180	QPRel-LB	–	0.033	1.6	3344.0
	1/50	1180	CROWN	**+2.2**	0.033	**16.0**	**286.8**
F-MNIST	1/100	1083	QPRel-LB	–	0.010	6.9	1520.8
	1/100	1083	CROWN	**+5.5**	0.010	**17.8**	**599.5**
	2/100	924	QPRel-LB	–	0.013	3.6	1855.8
	2/100	924	ConvAdv	**+17.1**	0.013	**33.1**	**341.6**
	1/300	1094	QPRel-LB	–	0.007	3.0	1579.8
	1/300	1094	CROWN	**+11.0**	0.007	**14.2**	**741.8**

Table 10: Results for test data, l_∞-perturbations

Setting				MedRelDiff	ϵ to hit 50%	VerRatio	MedRelDiff
Data	L/N	NrPts	Method	to QPRel (%)	VerRatio	worst (%)	UB-LB (%)
MNIST	1/100	1137	QPRel-LB	–	0.011	7.2	1439.0
	1/100	1137	CROWN	**+33.6**	0.011	**10.5**	**946.5**
	2/100	1028	QPRel-LB	–	**0.115**	5.2	2480.9
	2/100	1028	ConvAdv	**+16.5**	0.058	**28.7**	**218.6**
	1/300	1166	QPRel-LB	–	0.011	2.4	1704.3
	1/300	1166	CROWN	**+105.7**	0.011	**8.1**	**610.7**
	1/50	1054	QPRel-LB	–	0.037	1.8	3462.1
	1/50	1054	CROWN	+2.3	0.037	15.5	280.8
	1/50	1054	SDPRel	**-54.5**	**0.074**	**51.1**	**68.1**
F-MNIST	1/100	1061	QPRel-LB	–	0.011	6.8	1520.9
	1/100	1061	CROWN	**+5.5**	0.011	**19.3**	**605.4**
	2/100	949	QPRel-LB	–	0.012	2.5	1938.1
	2/100	949	ConvAdv	**+18.2**	0.012	**28.6**	**344.7**
	1/300	1069	QPRel-LB	–	0.007	3.9	1543.1
	1/300	1069	CROWN	**+12.6**	0.007	**12.9**	**736.9**