Machine Learning (2022) 111:2407–2433
https://doi.org/10.1007/s10994-022-06132-9

Robustness verification of ReLU networks via quadratic programming

Aleksei Kuvshinov[1] · **Stephan Günnemann**[1]

Received: 2 May 2021 / Revised: 1 October 2021 / Accepted: 6 February 2022 /
Published online: 16 March 2022

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 neural net classifier. In this work we present a procedure where we solve a convex quadratic programming (QP) task to obtain a lower bound on the DtDB. This bound is used as a robustness certificate of the classifier around a given sample. We show that our approach provides better or competitive results in comparison with a wide range of existing techniques.

Keywords Machine learning · Robustness verification · Neural networks · Minimal adversarial perturbation · Quadratic programming

1 Introduction

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 neural network classifiers to sensitive tasks that require certification of the obtained results.

In recent years this issue gained a lot of attention, resulting in a large variety of methods tackling tasks ranging from adversarial attacks and defenses against these to robustness verification and robust training. In this work we focus on robustness verification. 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.

Editors: Annalisa Appice, Sergio Escalera, Jose A. Gamez, Heike Trautmann.

✉ Aleksei Kuvshinov
kuvshino@in.tum.de

Stephan Günnemann
guennemann@in.tum.de

[1] Data Analytics and Machine Learning Group, Department of Informatics, Technical University of Munich, Munich, Germany

This problem plays a fundamental role in understanding the behavior of deep classifiers and essentially provides the only reliable way to assess classifier robustness. Unfortunately, its complexity class does not allow 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 the minimum adversarial perturbation of a certain (high) quality cannot be found within polynomial time.

Related work There exist two streams of related work on robustness verification of deep ReLU classifiers. This categorization is based on whether they are solving the verification problem exactly or verifying a bound on the distance to the decision boundary (DtDB).

The first group of methods are **exact verification** 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 (Ehlers 2017) rely on SMT solvers when solving the described task. Bunel et al. (2018) provide 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 to find a tight formulation of the MIP problem or (Jordan et al. 2018) use an algorithm to find the largest ball around the anchor point that touches the decision boundary.

The second popular class of methods for verifying 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 with respect to a certain norm in the input space. All existing 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. (2018a, b), 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 (later extended to quadratic functions by Zhang et al. 2018) to deal with nonlinear activation functions and propagate the layer-wise output bounds until the final layer. Salman et al. (2019) provide a unifying framework for the approaches using neuron-wise relaxations of the activation functions and use the best possible convex relaxation. 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 same group of the inexact verifiers, but deals with **constructing lower bounds** on DtDB without necessarily restricting admissible adversarial points to a given neighborhood. Croce et al. (2019) leverage the piecewise affine nature of the outputs of a ReLU classifier and compute lower bounds on DtDB by assuming that the classifier behaves globally the same way it does in the linear region around the given anchor point. The ϵ-verification task is closely related to this problem since each ϵ-neighborhood that is certified as adversarial-free immediately provides a lower bound on the minimal adversarial perturbation magnitude. It is also a common strategy for the ϵ-verification methods to use 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 is still successfully verified as robust.

Adversarial attacks Constructing misclassified examples that are close to the anchor point can be considered as a complementary research direction to robustness verification since each adversarial example by definition provides an upper bound on the DtDB. Many methods were

proposed to construct such points (Szegedy et al. 2014; Goodfellow et al. 2015; Kurakin et al. 2016; Papernot et al. 2016; Madry et al. 2017; Carlini and Wagner 2017).

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 (Hein and Andriushchenko 2017; Madry et al. 2017; Wong and Kolter 2018; Raghunathan et al. 2018a; Tsuzuku et al. 2018; Wang et al. 2018; Croce et al. 2019). We leave this direction for future work.

1.1 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 the DtDB in l_2-norm with an extension to l_∞-norm. We reach state-of-the-art performance for dense and convolutional networks compared to the bounds obtained from methods based on LP relaxations (CROWN by Zhang et al. 2018 and ConvAdv by Wong and Kolter 2018). Furthermore, our method performs much faster than methods based on SDP relaxations (Raghunathan et al. 2018b), while providing smaller lower bounds. This is a fundamental property due to the difference in computational complexity between SDP and QP tasks.
2. Unlike ϵ-verification techniques, we provide a lower bound on DtDB **without an initial guess and without computing bounds for the neuron activation values** in each layer. If additional information is present allowing the user to bound the distance to any admissible adversarial point from above, we incorporate these upper bounds in our formulation to verify larger regions around the anchor point. Such bounds have to be tight enough to verify non-trivial neighborhoods and play an important role in other relaxation techniques such as the SDP based approaches by Raghunathan et al. (2018b) and Dvijotham et al. (2019). We describe an efficient search method for pre-activation bounds resulting in larger verified regions based on sequential convex quadratic programming (QP).
3. To **analyze the gap in the optimal objective function value between the initial DtDB problem and our relaxation** we establish a connection of DtDB's dual problem to our QP task. It allows us to deconstruct this gap into two components. Moreover, we discuss how we improve the QP formulation to close the gap to DtDB and how we bound one of its components.

The remainder of this paper is organized as follows. In Sect. 2 we introduce the necessary notation. In Sect. 3.1 we formally define the problem of finding the smallest adversarial perturbation and in Sect. 3.2 introduce its QP relaxation QPRel. There we also formulate the dual DtDB problem as the best convex QP relaxation. In Sect. 3.3 we introduce additional linear constraints using bounds on the region of the admissible points around x^0 and summarize our verification procedure. In Sect. 4 we compare our approach to the LP- and SDP-based competitors. We summarize our findings in Sect. 5 and discuss the directions for future work.

2 Notation and idea

We consider a neural network consisting of L linear transformations representing dense, convolutional, skip or average pooling layers and $L-1$ ReLU activations (no ReLU after the last hidden layer). The number of neurons in layer l is denoted as n_l for $l = 0, \ldots, L$, meaning that the data has n_0 features and n_L classes. Furthermore, we present our analysis

for the l_2-norm as perturbation measure since only few available methods are applicable to this setting. To make our method comparable with the approach by Raghunathan et al. (2018b) we propose a generalization to l_∞-setting as well.

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 ith neuron in the lth 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 the observation that for each pair of scalars x and y the following holds (also used by Raghunathan et al. 2018b; Dvijotham et al. 2019 for ϵ-verification).

$$x = \left[y\right]_+ \iff x \geq 0, \quad x - y \geq 0, \quad x(x-y) = 0. \tag{2}$$

This relation allows us to obtain an optimization problem with linear complementarity constraints.

3 Verification as an optimization task

3.1 Formulation of DtDB

For a given sample $\tilde{x}^0$, pre-trained neural network f, predicted label $\tilde{y}$ and adversarial 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 of being classified as y compared to the initial label. This task corresponds to the following optimization problem.

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

where e_i is the ith unit vector in $\mathbb{R}^{n_L}$ and $\|x\|$ denotes the Euclidean norm of x. To compute the distance from $\tilde{x}^0$ to the (full) decision boundary, one needs to compute the solution for all adversarial labels $y = 1, \ldots, n_L$ except $\tilde{y}$. Next we unfold the above optimization problem using (1), where x denotes a container with all variables $x^0, \ldots, x^L$ and $[L]$ is the set $\{1, \ldots, L\}$.

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

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

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

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

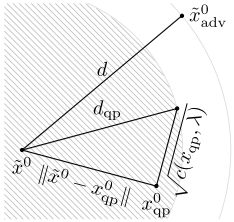

Fig. 1 Setting of the optimal solutions for DtDB $\tilde{x}_{\text{adv}}$ and QPRelx_{qp}

$$x^l - \left(W^l x^{l-1} + b^l\right) \geq 0, \quad x^l \geq 0 \quad \text{for } l \in [L-1]. \tag{4}$$

3.2 QP relaxation

To get rid of the quadratic equality constraints (3) we consider a Lagrangian relaxation of DtDB:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \|x^0 - \tilde{x}^0\|^2 + c(x, \lambda), \quad \text{s.t. } & (e_{\tilde{y}} - e_y)^T \left(W^L x^{L-1} + b^L\right) \leq 0, \\ & x^l - \left(W^l x^{l-1} + b^l\right) \geq 0, \quad x^l \geq 0 \quad \text{for } l \in [L-1], \end{aligned} \tag{QPRel}$$

where for arbitrary vectors $x^0 \in \mathbb{R}^{n_0}, \ldots, x^{L-1} \in \mathbb{R}^{n_{L-1}}$ and $\lambda \in \mathbb{R}^{L-1}_+$ we define

$$c(x, \lambda) := \sum_{l=1}^{L-1} \lambda_l \left(x^l\right)^T \left(x^l - \left(W^l x^{l-1} + b^l\right)\right) \tag{5}$$

as the *propagation gap*. The obtained problem is indeed a QP with linear constraints. We need to clarify two questions. How does the problem QPRel help us with solving DtDB and how do we solve this problem itself efficiently?

3.2.1 QPRel vs. DtDB

QPRel returns robust radius It follows directly from the definition of the Lagrange relaxation QPRel that for arbitrary non-negative λ it holds that:

- if x is feasible for DtDB we have $c(x, \lambda) = 0$, meaning that x equals the vector obtained by propagating x^0 through the neural network as defined in (1),
- if x is feasible for QPRel then $c(x, \lambda) \geq 0$, 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 Fig. 1). We include the proof of Lemma 1 and all other results in "Appendix B".

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

1. $d_{\text{qp}} \leq d$ *and when* $c(x_{\text{qp}}, \lambda) = 0$ *we have* $d_{\text{qp}} = d$ *and* x_{qp} *is optimal for DtDB.*
2. d_{qp} *is monotone with respect to* λ*, that is for two non-negative* λ^1, λ^2 *with* $\lambda^1 \leq \lambda^2$ *elementwise it holds that* $d_{\text{qp}}(\lambda^1) \leq d_{\text{qp}}(\lambda^2)$.

The first result from Lemma 1 ensures that d_{qp} provides a radius of a certified region 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, QPRel becomes non-convex for large values of λ. While one could try to tackle a non-convex QP with proper optimization methods, we address conditions such that QPRel is guaranteed to be convex and can be solved efficiently next.

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. Let $E_l \in \mathbb{R}^{n_l \times n_l}$ be the identity matrix of the corresponding dimension and set $\lambda_0 = 1$. We define $M^\lambda \in \mathbb{R}^{n \times n}$ as the symmetric block tridiagonal matrix with blocks $M^\lambda_{l,l} = 2\lambda_l E_l$ and $M^\lambda_{l,l-1} = -\lambda_l W^l$. Using this matrix we rewrite the objective function from QPRel as (see "Appendix B", Lemma 4 for the proof and definition of the terms)

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T M^\lambda(W) x + x^T B_1(b, \lambda, \tilde{x}^0) + \|\tilde{x}^0\|^2, \quad \text{s.t. } \bar{M}(W)x - B_2(b) \geq 0, \tag{6}$$

where B_1 influences only the linear term and is therefore not relevant in this section. From this reformulation we 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. This 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 a pre-trained neural network 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.*

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

$$\text{(nec. condition)} \quad \lambda_l \leq \frac{4\lambda_{l-1}}{\|W^l\|^2} \quad \text{for } l \geq 1. \tag{8}$$

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

$$\underline{\lambda}_l = 2 \prod_{k=1}^{l} \frac{1}{\|W^k\|^2}, \quad \bar{\lambda}_l = 4^l \prod_{k=1}^{l} \frac{1}{\|W^k\|^2}. \tag{9}$$

Finally, in case with a single hidden layer M^λ *is positive-semi definite even for* $\lambda = \bar{\lambda}$ *from* (8).

We use (7), (8) and our previous results as guidelines for the choice of λ. Since $d_{\text{qp}}(\lambda)$ is monotone in the sense of Lemma 1 we perform a binary search between $\underline{\lambda}$ and $\bar{\lambda}$ to find the point closest to $\bar{\lambda}$ (where QP is non-convex for networks with more than one hidden layer) such that the QP remains convex. We denote the obtained λ by $\hat{\lambda}$. 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 significantly improves the final bounds compared to the bounds obtained when using $\lambda = \underline{\lambda}$ from (7).

Note that this procedure has to be done *once* for a given classifier. $\hat{\lambda}$ is then used to solve QPRel for *all* anchor points and adversarial labels. This is a significant computational advantage compared to SDP-based ϵ-verification methods. For example, Dvijotham et al. (2019) include the dual multipliers as variables in a relaxation of the SDP problem that has to be solved for each combination of the anchor point, adversarial label and verified epsilon.

Relation to the dual of DtDB Since QPRel is a Lagrangean relaxation of a non-convex quadratically constrained QP DtDB, we unavoidably have a gap between their optimal objective values, but get a simpler problem to solve in return. To investigate and approximate the components of that gap, we look onto the relation of DtDB and QPRel from the perspective of duality theory. A similar question was investigated by Salman et al. (2019) for the existing ϵ-verification methods based on neuron-wise LP-relaxations. However, our method does not fall into this category because the relaxation happens jointly for all layers.

Note, that our formulation of DtDB problem contains quadratic equality constraints (3) and therefore has a non-convex admissible set. For the derivation of its dual problem we refer to the complementary material (see "Appendix B") and summarize here the most important result.

Theorem 2 *Solving the Lagrange dual problem of the non-convex DtDB is equivalent to solving the problem*

$$\max_{\lambda \in \mathbb{R}_+^{L-1}} \text{QPRel}(\lambda) \text{ s.t. } M^\lambda \text{ is positive semi-definite,}$$

where we slightly redefine the notation and write $\text{QPRel}(\lambda)$ *for the optimal objective function value of QPRel for the corresponding* λ*. We also denote* λ^* *as the optimal value of* λ *for the above problem.*

Now we are ready to formulate the result that provides a way to estimate how large is the difference between the optimal objective function value of QPRel for $\hat{\lambda}$, constructed using Theorem 1, and the optimal λ^*. The latter is defined by Theorem 2 and would provide the best bound we can get when constraining ourselves to the convex QP relaxations.

Lemma 2 *Denote* λ^* *as the optimal* λ *defined in Theorem 2,* $\hat{\lambda}$ *as* λ *we use for verification,* $\bar{\lambda}$ *as defined in* (9)*,* $c(x, \lambda)$ *as the propagation gap defined in* (5) *and* $\hat{x}_{qp}$ *as the solution of* $\text{QPRel}(\hat{\lambda})$*. Then we get the following upper bound on the possible improvement of QPRel's objective function for a* λ *value that is different from our* $\hat{\lambda}$*:*

$$\max_{\substack{\lambda \geq 0 \\ M^\lambda \text{ psd}}} \left(\text{QPRel}(\lambda) - \text{QPRel}(\hat{\lambda})\right) = \text{QPRel}(\lambda^*) - \text{QPRel}(\hat{\lambda}) \leq c(\hat{x}_{qp}, \bar{\lambda} - \hat{\lambda}).$$

In summary, we have the following relation between the values defined above, where we add -P and -D to the problem name to denote its primal and dual forms respectively:

$$\text{DtDB-P} \geq \text{DtDB-D} = \text{QPRel}(\lambda^*) \geq \text{QPRel}(\hat{\lambda}).$$

We have shown how to find a good $\hat{\lambda}$ and are able to estimate the gap resulting in the second $\geq$ sign as shown in Lemma 2. Additionally, in the next section we describe how to close the duality gap resulting in the first $\geq$ sign by introducing additional constraints to the QPRel problem.

3.3 Improving bounds via additional linear constraints

The initial DtDB problem and its relaxation QPRel do not require bounds on pre-activation values $W^l x^{l-1} + b^l$ frequently used in ϵ-verification approaches. However, if available, these can improve our relaxation. That is, we can additionally bound the admissible set of QPRel by

$$\underline{a}^l \leq W^l x^{l-1} + b^l \leq \bar{a}^l \quad \text{for } l = 1, \dots, L-1 \tag{10}$$

given some bounds $\underline{a}^l, \bar{a}^l \in \mathbb{R}^{n_l}$ for layer l. Moreover, we include the following linear constraint on each neuron i in layer l as also widely used in other verification methods for ReLU networks (Ehlers 2017; Wong and Kolter 2018; Dvijotham et al. 2019; Salman et al. 2019).

$$-\bar{a}_i^l (W^l x^{l-1} + b^l)_i + (\bar{a}_i^l - \underline{a}_i^l) x_i^l \leq -\bar{a}_i^l \underline{a}_i^l. \tag{11}$$

Note that constraints (10) and (11) are linear and therefore the new relaxation is still a QP.

Before continuing the discussion how we exploit these bounds, we first introduce the notation of a *proper bound propagation mapping*. We need this to ensure that the resulting solution of QPRel with these additional constraints is still a lower bound on DtDB. For a fixed anchor point and network weights consider a mapping from a bound in the input layer $\gamma \in \mathbb{R}_+$ to the bounds $\underline{a}^l(\gamma), \bar{a}^l(\gamma) \in \mathbb{R}^{n_l}$. We call this mapping a proper bound propagation mapping if

1. **bounds are valid** for all x^0 with $\|\tilde{x}^0 - x^0\| \leq \gamma$ inequalities (10) hold for the corresponding pre-activation values in each layer as defined in (1) and
2. **bounds are monotone** for arbitrary $\gamma_1 \leq \gamma_2$ in each hidden layer l of the network there holds $\bar{a}^l(\gamma_2) \geq \bar{a}^l(\gamma_1) \geq \underline{a}^l(\gamma_1) \geq \underline{a}^l(\gamma_2)$.

In our experiments we deploy the bound propagation technique by Wong and Kolter (2018) to obtain bounds $\underline{a}^l, \bar{a}^l$ since it satisfies these properties and is computationally efficient.

Lemma 3 *When using a proper bound propagation mapping, the following holds for the square root of the optimal objective function value $d_{\text{qp}}(\gamma)$ of QPRel (we drop the dependence on λ since it is now fixed) solved with the additional constraints (10) and (11) using pre-activation bounds $\underline{a}^l(\gamma), \bar{a}^l(\gamma)$.*

1. *$d_{\text{qp}}(\gamma_1) \geq d_{\text{qp}}(\gamma_2)$ if $\gamma_1 \leq \gamma_2$, i.e. $d_{\text{qp}}(\gamma)$ is monotonically decreasing, where we say that $d_{\text{qp}}(\gamma) = \infty$ if the corresponding QPRel with (10) and (11) is infeasible,*
2. *if $d_{\text{qp}}(\gamma) \leq \gamma$ then $d_{\text{qp}}(\gamma)$ is a lower bound on DtDB (which might not be the case otherwise, see "Appendix B" for details).*

Guided by the results of Lemma 3 we apply binary search to find the smallest γ that is still providing us with a lower bound $d_{\text{qp}}(\gamma)$ on the smallest adversarial perturbation (the smaller the value of γ, the better the resulting bound). In each step we solve a convex QP and increase γ if QPRel is infeasible, that is current bounds $\underline{a}^l(\gamma), \bar{a}^l(\gamma)$ are too tight, or if $d_{\text{qp}}(\gamma) > \gamma$ since in this case we do not have a certificate for $d_{\text{qp}}(\gamma)$ to be a valid lower bound on DtDB. Otherwise we set the current γ as the right boundary of the search interval and proceed with a smaller value of γ. The whole procedure is summarized in Algorithm 1.

Algorithm 1: Robustness Verification

Input: QP solver, neural network, anchor point $\tilde{x}^0$, predicted label $\tilde{y}$, initial large enough $\bar{\gamma}$, thresholds c_γ and c_λ as well as the maximum number of iterations n_γ.
Output: A distance d_{qp} such that for any x^0 with $\|\tilde{x}^0 - x^0\| \leq d_{\text{qp}}$ the neural networks predicts the same label as for $\tilde{x}^0$.
STEP 1: initialization
Compute $\bar{\lambda}$, $\underline{\lambda}$ from (9).
Using binary search between $\bar{\lambda}$, $\underline{\lambda}$ find the largest λ up to the tolerance c_λ such that M^λ is positive definite, denote the output by $\hat{\lambda}$.
STEP 2: verification
Set $d_{qp} = \infty$.
for $y \neq \tilde{y}$ **do**
  Set $i = 0$, $\gamma_l = 0$ and $\gamma_u = \bar{\gamma}$.
  repeat
    Set $\gamma = \frac{1}{2}(\gamma_u - \gamma_l)$.
    From the neural network, $\tilde{x}^0$, $\hat{\lambda}$ and γ construct the QPRel including constraints (10) and (11).
    Apply the QP solver on QPRel.
    If feasible denote d as the square root of the optimal objective function value, else $d = \infty$.
    if $d \leq \gamma$ **then**
      Tighten the constraints, set $\gamma_u = \gamma$ (continue searching in $[\gamma_l, \gamma]$).
      Set $d_{\text{qp},y} = d$ (see Lemma 3, point 2).
    else
      Loosen the constraint, set $\gamma_l = \gamma$ (continue searching in $[\gamma, \gamma_u]$).
    end if
  until $\gamma_u - \gamma_l < c_\gamma$ or $i = n_\gamma$
  To achieve that $d_{\text{qp}} = \min_{y \neq \tilde{y}} d_{\text{qp},y}$ update d_{qp} if necessary:
  if $d_{\text{qp},y} < d_{\text{qp}}$ **then**
    Set $d_{\text{qp}} = d_{\text{qp},y}$.
  end if
end for
Return: d_{qp}

3.4 l_∞-Setting

For comparison with the SDP-based approach by Raghunathan et al. (2018b) we show how we apply our method 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:

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

Note that the quadratic constraints do not harm the complexity since they describe a convex cone and can be handled by the QP-solvers. While this formulation is of a similar structure as the QPRel (quadratic objective as well as linear and 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 (6), 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 $x^0 \in \mathbb{R}^{n_0}$ as well as $x^1 = \alpha W^1 x^0$ for some $0 < \alpha < 1$ 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.

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

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. In Sect. 4 we obtain the results in the l_∞-setting by solving this problem with $\mu = (2n_0)^{-1}$.

4 Experiments

For each considered sample we apply the procedure described in Sect. 3.3, Algorithm 1 including tightening of the relaxation by introducing additional linear constraints (10) and (11). $\hat{\lambda}$ is chosen for each classifier according to Theorem 1 and the discussion afterwards such that a relative accuracy of at least $c_\lambda = 10^{-4}$ is achieved during the binary search in each λ_l. For the values of other parameters in Algorithm 1 we choose for all tests $c_\gamma = 10^{-8}$ and $n_\gamma = 10$. Other methods are tested with the default settings as provided in the corresponding repositories. For ConvAdv by Wong and Kolter (2018) we use the maximum of 200 iterations during Newton's method for the networks **D8**, **D8R**, **C**, **CR** (see below) and 20 otherwise. To solve the QP tasks or verify that they are infeasible we use Gurobi (Gurobi Optimization 2018).

Datasets and classifiers The experiments are performed using the MNIST (LeCun et al. 1999) and Fashion-MNIST (Xiao et al. 2017) datasets as well as the tabular datasets IRIS (3 classes, 4 features) and WINE (2 classes, 12 features) from Dua and Graff (2017) scaled

Table 1 Clean accuracy

Model	MNIST	FMNIST
D2	0.9753	0.8847
D2R (l_2)	0.8518	0.7184
D2R (l_∞)	0.8722	0.7440
D4	0.9758	0.8873
D4R (l_2)	0.8169	0.7092
D4R (l_∞)	0.8916	0.7324
D8	0.9649	0.8802
D8R (l_2)	0.7482	0.7894
D8R (l_∞)	0.6843	0.6856
C	0.9897	0.9136
CR (l_2)	0.8323	0.7383
CR (l_∞)	0.9835	0.7688
	IRIS	WINE
D2	0.9733	0.9858
D2R (l_2)	0.9349	0.7845
D2R (l_∞)	0.9232	0.7538

such that the feature values lie in [0, 1] interval. For each of the datasets we use the correctly classified samples from 120 train points to evaluate the verification approaches.

For classification we take ReLU networks consisting of dense and convolutional linear layers. The architectures we used for the image datasets are named **D2**, **D4**, **D8** (dense networks containing 2, 4 and 8 hidden layers consisting of 50 neurons each with an exception for the last 4 layers in **D8** that have 20 neurons each) and **C**. We use similar structures of the networks as Wong and Kolter (2018) to enable easier comparison. The latter consists of two convolutional layers with 4×4 windows, a stride of 2 as well as 16 and 32 output channels correspondingly, followed by two dense layers with input/output dimensions of 1568/100 and finally 100/10. For each architecture we use normally trained classifiers as well as robustly trained ones (indicated by suffix **R**, e.g. **CR**) using the method by Wong and Kolter (2018) with $\epsilon = 1.58$ in l_2-setting and $\epsilon = 0.1$ in l_∞-setting. For the tabular datasets we use a dense network with two hidden layers with 10 neurons called **D2** and different ϵ values in l_2-setting: 0.113 for IRIS and 0.195 for WINE (and the same $\epsilon = 0.1$ in l_∞-setting). The weights as well as the project code are available at github.com/Aleksei-Kuvshinov/QPRel. In Table 1 we show the clean accuracy of the trained networks on the corresponding test sets.

Competitors We compare our approach QPRel with the following 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 dense networks only since its implementation did not support convolutional layers), and SDPRel by Raghunathan et al. (2018b) based on a SDP relaxation solved by MOSEK.

Metrics The results on MNIST and Fashion-MNIST for the l_2- and l_∞ setting are shown in Tables 3 and 4 correspondingly. We show the results on the tabular data in Table 2. We run the methods for each of the considered samples and report the following metrics.

Table 2 Experiment results, tabular data

IRIS and WINE, l_2			AvgBound	MedRelDiff	ϵ to hit 50%
Dataset	Network	Method		to QPRel (%)	LB-verified
IRIS	D2	QPRel	0.24	–	**0.206**
		CROWN	0.24	+ 2.17	0.202
		ConvAdv	0.23	+ 0.63	0.197
WINE	D2	QPRel	0.18	–	**0.179**
		CROWN	0.18	+ 1.76	0.176
		ConvAdv	0.18	+ 1.19	0.174
IRIS	D2R	QPRel	**0.24**	–	**0.225**
		CROWN	0.24	+ 0.09	0.225
		ConvAdv	0.24	+ 1.00	0.223
WINE	D2R	QPRel	**0.62**	–	**0.614**
		CROWN	0.46	+ 34.9	0.455
		ConvAdv	0.45	+ 37.2	0.449
IRIS and WINE, l_∞			AvgBound	MedRelDiff	ϵ to hit 50%
Dataset	Network	Method		to QPRel (%)	LB-verified
IRIS	D2	QPRel	0.12	–	0.106
		CROWN	0.13	−3.16	**0.110**
		ConvAdv	**0.13**	−6.99	0.109
WINE	D2	QPRel	0.06	–	0.055
		CROWN	**0.06**	−6.06	**0.059**
		ConvAdv	0.06	−4.38	0.058
IRIS	D2R	QPRel	0.14	–	0.144
		CROWN	**0.15**	−8.50	**0.157**
		ConvAdv	0.15	−6.48	0.157
WINE	D2R	QPRel	**0.19**	–	**0.190**
		CROWN	0.15	+ 27.3	0.150
		ConvAdv	0.15	+ 28.0	0.150

(1) AvgBound the average value of the bounds obtained from QPRel and the corresponding competitor (the best value marked bold if at least 5% larger than the worst one). To asses the impact of introducing additional linear constraints using a bound propagation method as described in Sect. 3.3 we report the lower bounds obtained by solving QPRel without constraints (10) and (11) in the last column ***AvgBound (no BndProp)*** in Tables 3 and 4. ***(2) MedRelDiff to QPRel***: the median of the *relative* difference between the bounds (e.g. QPRel minus CROWN and then divided by CROWN). *Positive* values for the *lower* bounds mean our bounds are better in average over the samples. ***(3) ϵ to hit 50% LB-verified***: the number of samples with an adversarial-free radius of ϵ is monotonically decreasing in ϵ. Therefore, to assess the performance of a verification procedure like QPRel or CROWN we report the smallest ϵ such that exactly 50% of the samples are successfully verified. The larger this value, the better (the largest values marked bold).

l_2-setting, state-of-the-art bounds For all considered architectures the lower bounds computed by QPRel are tighter in comparison to the competitors in average (see Table 3,

Table 3 Experiment results, image data, l_2-setting

MNIST, l_2		AvgBound	MedRelDiff	ϵ to hit 50%	AvgBound
Network	Method		to QPRel (%)	LB-verified	(no BndProp)
D2	QPRel	**0.38**	–	**0.371**	0.37
	CROWN	0.25	+ 52.8	0.235	–
	ConvAdv	0.23	+ 66.0	0.217	–
D2R	QPRel	**1.77**	–	**1.681**	1.39
	CROWN	1.60	+ 11.6	1.561	–
	ConvAdv	1.44	+ 25.6	1.361	–
D4	QPRel	**0.20**	–	**0.200**	0.09
	CROWN	0.17	+ 17.9	0.169	–
	ConvAdv	0.17	+ 24.3	0.159	–
D4R	QPRel	**1.67**	–	1.689	0.69
	CROWN	1.63	+ 6.5	**1.770**	–
	ConvAdv	1.46	+ 19.1	1.649	–
D8	QPRel	0.17	–	**0.167**	0.03
	CROWN	0.16	+ 3.2	0.159	–
	ConvAdv	0.16	+ 3.2	0.159	–
D8R	QPRel	**1.78**	–	**2.034**	0.18
	CROWN	1.61	+ 10.8	1.780	–
	ConvAdv	1.49	+ 22.9	1.628	–
C	QPRel	**0.13**	–	**0.140**	0.01
	ConvAdv	0.06	+ 108.1	0.064	–
CR	QPRel	**1.57**	–	**1.686**	0.02
	ConvAdv	1.37	+ 20.3	1.488	–
FMNIST, l_2		AvgBound	MedRelDiff	ϵ to hit 50%	AvgBound
Network	Method		to QPRel (%)	LB-verified	(no BndProp)
D2	QPRel	**0.31**	–	**0.277**	0.19
	CROWN	0.21	+ 42.5	0.202	–
	ConvAdv	0.20	+ 55.1	0.192	–
D2R	QPRel	**2.22**	–	**2.290**	1.43
	CROWN	2.11	+ 4.1	2.197	–
	ConvAdv	1.92	+ 15.4	1.916	–
D4	QPRel	**0.19**	–	**0.181**	0.03
	CROWN	0.17	+ 13.7	0.158	–
	ConvAdv	0.16	+ 20.1	0.152	–
D4R	QPRel	**2.12**	–	**2.154**	0.51
	CROWN	2.05	+ 3.4	2.019	–
	ConvAdv	1.89	+ 14.6	1.840	–
D8	QPRel	**0.14**	–	**0.134**	0.01
	CROWN	0.14	+ 7.3	0.128	–
	ConvAdv	0.14	+ 7.3	0.128	–
D8R	QPRel	**2.03**	–	**2.225**	0.10
	CROWN	1.88	+ 6.9	1.947	–
	ConvAdv	1.74	+ 16.3	1.761	–

Table 3 (continued)

FMNIST, l_2		AvgBound	MedRelDiff	ϵ to hit 50%	AvgBound
Network	Method		to QPRel (%)	LB-verified	(no BndProp)
C	QPRel	**0.09**	–	**0.101**	0.00
	ConvAdv	0.07	+ 28.1	0.082	–
CR	QPRel	**1.64**	–	**1.760**	0.02
	ConvAdv	1.57	+ 7.6	1.553	–

AvgBound and *MedRelDiff*) and for the networks with a smaller number of hidden layers even for most individual images. Naturally, this results in larger values of *ϵ to hit 50% LB-verified* as well. It seems that the competitors tend to underestimate robustness of the considered networks, especially if it was not trained robustly. For the normally trained convolutional network **C** on MNIST we were able to improve the competitor's lower bounds by a factor of 2 in average. In contrast to other verification procedure that can not easily verify networks that were not robustly trained, our method is applicable to normally trained networks as well.

While this improvement of the verifiable radius comes at higher computational cost (QPRel is about one order of magnitude slower than the LP-competitors) due to a fundamental difference in complexity of the LP- and QP-tasks, the average runtime per sample is still only seconds or less for the dense and multiple minutes for the convolutional networks. We present a detailed runtime comparison in "Appendix A".

In the last column of Table 3, we report the lower bound obtained when solving QPRel *without* introducing additional constraints as described in Sect. 3.3. We observe that the relaxation becomes less tight for networks with more layers and if it was trained robustly. We suppose that when the number of layers L becomes larger the binary search between $\underline{\lambda}$ and $\bar{\lambda}$ (see Theorem 1 and the discussion afterwards) in a higher dimensional space results in a point far from the optimal Lagrange multipliers. Especially the last $\underline{\lambda}_{L-1}$ and $\bar{\lambda}_{L-1}$ defined in (9) become small such that the gap between x^{L-1} and $W^{L-1}x^{L-2} + b^{L-1}$ has only a very limited effect on the objective function of QPRel. That results in an undesired optimal solution of QPRel with a large propagation gap. At that point, by introducing additional linear constraints [especially (11)] we prohibit this behavior by bounding the propagation gap for the set of feasible points. Overall, incorporating additional linear constraints by using bounds on ReLU's input has proven to significantly improve our relaxation and the resulting lower bounds.

l_∞-setting, comparison with SDP-relaxations In order to compare our method with the work done by Raghunathan et al. (2018b) we generalize QPRel to the l_∞-setting as described in Sect. 3.4. Note, that the resulting relaxation is looser than the initial QPRel for the l_2-setting since we bound the l_∞-distance from below to make the problem quadratic and convex. To compute the largest ϵ such that the SDP verification succeeds we perform a binary search on the [0, 1] interval. Since this approach takes longer to run we test it only on the networks **D2** and **D2R** trained with $\epsilon = 0.1$ (MNIST data).

In l_∞ setting our bounds are about 3 times smaller than the ones of SDPRel (see Table 4, *MedRelDiff to QPRel*)—though computed three orders of magnitude faster (see "Appendix A"). This shows that the QP relaxation is less suited than the competitors for obtaining tight bounds in l_∞-setting as already indicated by the arguments above due to the nature of the quadratic relaxation, but trades this off by much better efficiency compared to SDPRel.

Table 4 Experiment results, image data, l_∞-setting

MNIST, l_∞		AvgBound	MedRelDiff	ϵ to hit 50%	AvgBound
Network	Method		to QPRel (%)	LB-verified	(no BndProp)
D2	QPRel	0.01	–	0.014	0.01
	CROWN	0.01	+ 19.6	0.011	–
	ConvAdv	0.01	+ 20.7	0.011	–
	SDPRel	**0.03**	−43.5	**0.024**	–
D2R	QPRel	0.05	–	0.048	0.04
	CROWN	0.10	−54.4	0.105	–
	ConvAdv	0.10	−50.5	0.099	–
	SDPRel	**0.16**	−68.2	**0.165**	–
D4	QPRel	**0.01**	–	0.009	0.01
	CROWN	0.01	+ 8.2	0.008	–
	ConvAdv	**0.01**	+ 0.0	**0.009**	–
D4R	QPRel	0.04	–	0.038	0.03
	CROWN	**0.11**	−63.3	**0.106**	–
	ConvAdv	0.10	−60.8	0.098	–
D8	QPRel	< 0.01	–	0.003	< 0.01
	CROWN	**0.01**	−60.9	**0.008**	–
	ConvAdv	0.01	−64.0	0.007	–
D8R	QPRel	0.01	–	0.006	< 0.01
	CROWN	**0.09**	−93.9	**0.093**	–
	ConvAdv	**0.09**	−93.9	**0.093**	–
C	QPRel	0.01	–	0.002	< 0.01
	ConvAdv	**0.01**	−101.6	**0.005**	–
CR	QPRel	0.04	–	0.038	< 0.01
	ConvAdv	**0.09**	−61.4	**0.091**	–
FMNIST, l_∞		AvgBound	MedRelDiff	ϵ to hit 50%	AvgBound
Network	Method		to QPRel (%)	LB-verified	(no BndProp)
D2	QPRel	**0.01**	–	**0.011**	0.01
	CROWN	0.01	+ 11.1	0.010	–
	ConvAdv	0.01	+ 11.7	0.010	–
D2R	QPRel	0.05	–	0.052	0.04
	CROWN	**0.13**	−64.1	**0.136**	–
	ConvAdv	0.13	−61.5	0.128	–
D4	QPRel	0.01	–	0.008	< 0.01
	CROWN	0.01	+ 2.2	0.008	–
	ConvAdv	**0.01**	−5.8	0.008	–
D4R	QPRel	0.06	–	0.059	0.02
	CROWN	**0.13**	−54.7	**0.133**	–
	ConvAdv	0.12	−51.1	0.125	–
D8	QPRel	< 0.01	–	0.004	< 0.01
	CROWN	0.01	−27.4	0.007	–
	ConvAdv	**0.01**	−27.5	**0.007**	–

Table 4 (continued)

FMNIST, l_∞		AvgBound	MedRelDiff	ϵ to hit 50%	AvgBound
Network	Method		to QPRel (%)	LB-verified	(no BndProp)
D8R	QPRel	0.05	–	0.060	< 0.01
	CROWN	**0.11**	−49.0	**0.112**	–
	ConvAdv	**0.11**	−49.0	**0.112**	–
C	QPRel	< 0.01	–	0.001	< 0.01
	ConvAdv	**0.01**	−162.0	**0.004**	–
CR	QPRel	0.02	–	0.004	< 0.01
	ConvAdv	**0.08**	−373.3	**0.086**	–

5 Conclusion and future work

We presented a novel approach to solve the problem of approximating the minimal adversarial perturbations for ReLU networks based on a convex QP relaxation of DtDB. We show that the lower bounds computed with QPRel 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. The obtained lower bounds in the l_2-setting show state-of-the-art results allowing to certify larger radia around the data samples as adversarial free.

With our contribution we make a step towards robustness verification of deep ReLU-based classifiers. While the proposed theoretical framework is applicable to any linear transformations including dense, convolutional and average pooling layers as well as skip connections, it requires a different analysis when a non-ReLU activation functions are used (except leaky ReLU). To be able to apply the approach on a wider class of networks it should be generalized to popular architectures beyond ReLU activations. Last but not least, excellent results that our method demonstrated for the verification task indicate an intriguing research direction toward robust training. Based on our certificates the next step towards robust training would be an approach that uses the solution of QPRel to make an update step resulting in larger certified neighborhood for the correctly classified samples. As our approach does not require a predefined ϵ, that additional regularization acts individually for each sample depending on its current robust neighborhood.

Appendix

A Runtime

Tables 5, 6, 7 and 8 (see "Appendix C") show the average runtime and its standard deviation for the considered experiments. During the binary search procedure we apply for SDPRel we always make 10 bisection steps. Furthermore, we speed up this approach by

Table 5 Runtime comparison, MNIST, l_2

MNIST, l_2			Runtime-LB (s) sequential QPRel	
Network	NrPts	Method	Mean	Std
D2	119	QPRel	6.231	3.996
D2	119	CROWN	0.092	0.026
D2	119	ConvAdv	0.829	0.017
D2R	99	QPRel	5.285	1.144
D2R	99	CROWN	0.104	0.030
D2R	99	ConvAdv	0.735	0.020
D4	120	QPRel	6.267	0.996
D4	120	CROWN	0.312	0.017
D4	120	ConvAdv	1.886	0.045
D4R	96	QPRel	12.514	1.649
D4R	96	CROWN	0.346	0.033
D4R	96	ConvAdv	1.426	0.103
D8	119	QPRel	6.134	1.013
D8	119	CROWN	0.644	0.032
D8	119	ConvAdv	13.889	0.820
D8R	75	QPRel	13.160	2.953
D8R	75	CROWN	0.620	0.138
D8R	75	ConvAdv	10.153	2.118
C	120	QPRel	2370.008	824.183
C	120	ConvAdv	143.156	11.727
CR	94	QPRel	214.818	91.502
CR	94	ConvAdv	48.880	3.656

modifying MOSEK parameters[1] (see Table 9 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 than when the solver would reach a true optimum.

All tasks necessary for the computation of bounds on DtDB for one sample are run on four CPUs (including the solution of QPs and SDPs with Gurobi and MOSEK respectively). Column *Runtime-LB (s), sequential QPRel* shows the runtime of the whole bound improvement procedure as described in Sect. 3.3. From the comparison of QPRel and the LP-based approaches we see the clear advantage of the latter since they do not involve any optimization task. However, especially in the l_2-setting this advantage comes in cost of the verification properties as discussed in Sect. 4. On the other hand, SDPRel with a binary search provides better bounds, but is about three orders of magnitude slower than QPRel.

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

Table 6 Runtime comparison, MNIST, l_∞

MNIST, l_∞			Runtime-LB (s) sequential QPRel	
Network	NrPts	Method	Mean	Std
D2	119	QPRel	5.459	0.782
D2	119	CROWN	0.088	0.030
D2	119	ConvAdv	0.858	0.018
D2	119	SDPRel	5094.572	974.972
D2R	104	QPRel	4.724	0.805
D2R	104	CROWN	0.093	0.020
D2R	104	ConvAdv	0.836	0.013
D2R	104	SDPRel	5443.324	1121.064
D4	120	QPRel	6.593	1.112
D4	120	CROWN	0.256	0.025
D4	120	ConvAdv	2.015	0.024
D4R	103	QPRel	11.106	3.700
D4R	103	CROWN	0.308	0.021
D4R	103	ConvAdv	1.905	0.094
D8	119	QPRel	8.358	1.692
D8	119	CROWN	0.523	0.060
D8	119	ConvAdv	15.502	1.004
D8R	75	QPRel	12.024	2.543
D8R	75	CROWN	0.629	0.062
D8R	75	ConvAdv	10.281	2.072
C	120	QPRel	3647.024	322.421
C	120	ConvAdv	201.347	23.439
CR	94	QPRel	564.413	231.521
CR	94	ConvAdv	141.203	11.073

B Proofs

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

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

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

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

Table 7 Runtime comparison, F-MNIST, train data, l_2

F-MNIST, l_2			Runtime-LB (s) sequential QPRel	
Network	NrPts	Method	Mean	Std
D2	117	QPRel	4.836	0.685
D2	117	CROWN	0.096	0.020
D2	117	ConvAdv	0.825	0.038
D2R	87	QPRel	5.196	1.007
D2R	87	CROWN	0.118	0.030
D2R	87	ConvAdv	0.738	0.043
D4	116	QPRel	5.715	0.765
D4	116	CROWN	0.305	0.033
D4	116	ConvAdv	1.629	0.059
D4R	88	QPRel	11.618	2.079
D4R	88	CROWN	0.349	0.033
D4R	88	ConvAdv	1.918	0.038
D8	117	QPRel	6.470	1.269
D8	117	CROWN	0.626	0.046
D8	117	ConvAdv	14.123	0.999
D8R	89	QPRel	12.992	4.952
D8R	89	CROWN	0.730	0.050
D8R	89	ConvAdv	11.818	0.843
C	112	QPRel	2805.849	955.243
C	112	ConvAdv	177.451	15.203
CR	90	QPRel	352.121	138.573
CR	90	ConvAdv	57.378	9.209

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

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

□

Theorem 3 *Let $W^1, \ldots, W^{L-1}$ be the weights of a pre-trained neural network and $\|W\|$ the spectral norm of an arbitrary matrix. Then the following two conditions for λ provide*

Table 8 Runtime comparison, F-MNIST, l_∞

F-MNIST, l_∞			Runtime-LB (s) sequential QPRel	
Network	NrPts	Method	Mean	Std
D2	117	QPRel	4.836	0.685
D2	117	CROWN	0.096	0.020
D2	117	ConvAdv	0.825	0.038
D2R	87	QPRel	5.196	1.007
D2R	87	CROWN	0.118	0.030
D2R	87	ConvAdv	0.738	0.043
D4	116	QPRel	5.715	0.765
D4	116	CROWN	0.305	0.033
D4	116	ConvAdv	1.629	0.059
D4R	88	QPRel	11.618	2.079
D4R	88	CROWN	0.349	0.033
D4R	88	ConvAdv	1.918	0.038
D8	117	QPRel	6.470	1.269
D8	117	CROWN	0.626	0.046
D8	117	ConvAdv	14.123	0.999
D8R	89	QPRel	12.992	4.952
D8R	89	CROWN	0.730	0.050
D8R	89	ConvAdv	11.818	0.843
C	110	QPRel	5372.089	562.702
C	110	ConvAdv	252.012	29.203
CR	86	QPRel	1252.235	422.278
CR	86	ConvAdv	185.257	15.502

Table 9 MOSEK parameters we use to run SDPRel and their default values

Parameter	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}

correspondingly a sufficient and a necessary criterion for the matrix M^λ to be positive semi-definite.

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

$$\text{(nec. condition)} \quad \lambda_l \leq \frac{4\lambda_{l-1}}{\|W^l\|^2} \quad \text{for } l \geq 1. \tag{8}$$

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

$$\underline{\lambda}_l = 2\prod_{k=1}^{l}\frac{1}{\|W^k\|^2}, \quad \bar{\lambda}_l = 4^l\prod_{k=1}^{l}\frac{1}{\|W^k\|^2}. \tag{9}$$

Finally, in case with a single hidden layer M^λ *is positive-semi definite even for* $\lambda = \bar{\lambda}$ *from* (8).

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.

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

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

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

and the lower bound on $x^T M^\lambda x$ obtained above 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

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

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 sufficiency of (8) if the considered network contains one hidden layer. To do so we can reuse the last computation and obtain now for arbitrary $x \in \mathbb{R}^n$

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

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

Theorem 4 *Solving the Lagrange dual problem of the non-convex DtDB is equivalent to solving the problem*

$$\max_{\lambda \in \mathbb{R}_+^{L-1}} \text{QPRel}(\lambda) \text{ s.t. } M^\lambda \text{ is positive semi-definite.}$$

Proof We start with the following formulation of DtDB, where we use $\leq$ instead of the equality to formulate the complementarity constraints. This is possible because of the fact that both components of the product are non-negative due to the other constraints, that are rewritten using the matrix notation from Lemma 4.

$$\min_{x \in \mathbb{R}^n} \|x^0 - \tilde{x}^0\|^2, \quad \text{s.t. } \left(x^l\right)^T \left(x^l - \left(W^l x^{l-1} + b^l\right)\right) \leq 0 \quad \text{for } l \in [L-1],$$
$$\bar{M} x - B_2 \geq 0.$$

To formulate the Lagrange dual we introduce the non-negative Lagrange multipliers λ and μ. We obtain the following formulation of the primal problem

$$\min_{x \in \mathbb{R}^n} \max_{\substack{\lambda \geq 0 \\ \mu \geq 0}} \|x^0 - \tilde{x}^0\|^2 + c(x, \lambda) + \mu^T (B_2 - \bar{M} x),$$

and by switching the order of the optimization tasks we arrive at the dual task (again, we use the notation from Lemma 4 to rewrite the objective).

$$\max_{\substack{\lambda \geq 0 \\ \mu \geq 0}} \min_{x \in \mathbb{R}^n} \frac{1}{2} x^T M^\lambda x + x^T (B_1 - \bar{M}^T \mu) + \|\tilde{x}^0\|^2 + \mu^T B_2.$$

Note, that for λ such that M^λ is not positive semi-definite there exists x such that $x^T M^\lambda x < 0$. Therefore, the inner optimization task is unbounded in this case. That means we can introduce the desired constraint on λ and solve the convex QP explicitly, obtaining the following equivalent formulation of the dual.

$$\max_{\substack{\lambda \geq 0 \\ \mu \geq 0}} -\frac{1}{2}(B_1 - \bar{M}^T \mu)^T \left(M^\lambda\right)^{-1} (B_1 - \bar{M}^T \mu) + \|\tilde{x}^0\|^2 + \mu^T B_2,$$
$$\text{s.t. } M^\lambda \text{ is positive semi-definite.}$$

By splitting the maximization task in two we obtain

$$\max_{\substack{\lambda \geq 0 \\ M^\lambda \text{ psd}}} \max_{\mu \geq 0} -\frac{1}{2}(B_1 - \bar{M}^T \mu)^T \left(M^\lambda\right)^{-1} (B_1 - \bar{M}^T \mu) + \|\tilde{x}^0\|^2 + \mu^T B_2,$$

where the inner task is a convex QP. Therefore, it can be transformed to its dual without introducing the duality gap. Following the steps we have done backwards (now with a fixed λ) we obtain exactly the QPRel problem as the dual of the inner optimization problem. That concludes the proof since we arrive at the formulation from the claim. □

Lemma 2 *Denote λ^* as the optimal λ defined in Theorem 2, $\hat{\lambda}$ as λ we use for verification, $\bar{\lambda}$ as defined in (9), $c(x, \lambda)$ as the propagation gap defined in (5) and $\hat{x}_{qp}$ as the solution of* QPRel($\hat{\lambda}$). *Then we get the following upper bound on the possible improvement of QPRel's objective function for a λ value that is different from our $\hat{\lambda}$:*

$$\max_{\substack{\lambda \geq 0 \\ M^\lambda \text{ psd}}} \left(\text{QPRel}(\lambda) - \text{QPRel}(\hat{\lambda})\right) = \text{QPRel}(\lambda^*) - \text{QPRel}(\hat{\lambda}) \leq c(\hat{x}_{qp}, \bar{\lambda} - \hat{\lambda}).$$

Proof The first equality holds due to the definition of λ^*. From there we proceed as follows.

$$\begin{aligned}
\text{QPRel}(\lambda^*) - \text{QPRel}(\hat{\lambda}) &\leq \text{QPRel}(\bar{\lambda}) - \text{QPRel}(\hat{\lambda}) \\
&= \left(\min_x \|x^0 - \tilde{x}^0\| + c(x, \bar{\lambda})\right) - \left(\|\hat{x}^0 - \tilde{x}^0\| + c(\hat{x}, \hat{\lambda})\right) \\
&\leq \|\hat{x}^0 - \tilde{x}^0\| + c(\hat{x}, \bar{\lambda}) - \left(\|\hat{x}^0 - \tilde{x}^0\| + c(\hat{x}, \hat{\lambda})\right) = c(\hat{x}, \bar{\lambda} - \hat{\lambda}).
\end{aligned}$$

To prove the first inequality note that M^{λ^*} is positive semi-definite and therefore λ^* satisfies the necessary condition (8) from Theorem 1 meaning that $\lambda^* \leq \bar{\lambda}$ elementwise. Due to the fact that QPRel is monotone with respect to λ (see Lemma 1) we get that $\text{QPRel}(\lambda^*) \leq \text{QPRel}(\bar{\lambda})$. For the second inequality instead of the minimum objective function value we just use the value of $\text{QPRel}(\bar{\lambda})$ evaluated at $\hat{x}$. Finally, the last equation holds due to the linearity of $c(x, \lambda)$ with respect to λ. □

Lemma 3 *When using a proper bound propagation mapping, the following holds for the square root of the optimal objective function value $d_{\text{qp}}(\gamma)$ of QPRel (we drop the*

dependence on λ since it is now fixed) *solved with the additional constraints* (10) *and* (11) *using pre-activation bounds* $\underline{a}^l(\gamma), \bar{a}^l(\gamma)$.

1. $d_{\text{qp}}(\gamma_1) \geq d_{\text{qp}}(\gamma_2)$ *if* $\gamma_1 \leq \gamma_2$, *i.e.* $d_{\text{qp}}(\gamma)$ *is monotonically decreasing, where we say that* $d_{\text{qp}}(\gamma) = \infty$ *if the corresponding QPRel with* (10) *and* (11) *is infeasible*,
2. *if* $d_{\text{qp}}(\gamma) \leq \gamma$ *then* $d_{\text{qp}}(\gamma)$ *is a lower bound on DtDB*.

Proof **First claim** follows directly from the fact that QPRel with constraints (10) and (11) has a larger feasible set if the additional constraints are constructed using a larger value for γ. Assume that $\gamma_1 \leq \gamma_2$ and constraints (10), (11) hold for bounds $\underline{a}^l(\gamma_1)$ and $\bar{a}^l(\gamma_1)$, then they hold for $\underline{a}^l(\gamma_2)$ and $\bar{a}^l(\gamma_2)$ automatically.

For (10), if $\underline{a}^l(\gamma_1) \leq W^l x^{l-1} + b^l \leq \bar{a}^l(\gamma_1)$ then $\underline{a}^l(\gamma_2) \leq W^l x^{l-1} + b^l \leq \bar{a}^l(\gamma_2)$ since $\underline{a}^l(\gamma_2) \leq \underline{a}^l(\gamma_1)$ and $\bar{a}^l(\gamma_1) \leq \bar{a}^l(\gamma_2)$. The latter is true due to the fact that proper bound propagation mapping is monotonic.

For (11) assume that $\underline{a}^l_i(\gamma_1) < 0 < \bar{a}^l_i(\gamma_1)$. Otherwise the only admissible values for x^l_i and $(W^l x^{l-1} + b^l)_i$ satisfy $x^l_i = \max((W^l x^{l-1} + b^l)_i, 0)$ and are obviously admissible in case of the less restrictive bounds $\underline{a}^l_i(\gamma_2)$ and $\bar{a}^l_i(\gamma_2)$ as well. With this assumption (11) can be equivalently reformulated as

$$x^l_i \leq \frac{\bar{a}^l_i}{\bar{a}^l_i - \underline{a}^l_i}\big((W^l x^{l-1} + b^l)_i - \underline{a}^l_i\big),$$

where the right hand side is increasing in $\bar{a}^l_i$ and decreasing in $\underline{a}^l_i$ (as long as $\underline{a}^l_i < 0 < \bar{a}^l_i$). Therefore it remains true if we replace $\underline{a}^l_i, \bar{a}^l_i$ by any less restrictive bounds.

To prove the **second claim** we denote as before the square root of the optimal objective function of DtDB by d_{adv}, that is the distance to the closest adversarial, and the corresponding solution by $x_{\text{adv}} \in \mathbb{R}^n$. Consider two cases.

If $\gamma > d_{\text{adv}} = \|\tilde{x}^0 - x^0_{\text{adv}}\|$, then x_{adv} is an admissible point for QPRel with additional linear constraints due to the first property of a proper bound propagation mapping. The objective function value at this point is d^2_{adv} and therefore the minimum $d_{\text{qp}}(\gamma)^2$ cannot be larger (similar to the argumentation in Lemma 1).

If $\gamma \leq d_{\text{adv}}$, we directly follow that $d_{\text{qp}}(\gamma) \leq d_{\text{adv}}$ from the assumption. Note that at this point the assumption $d_{\text{qp}}(\gamma) \leq \gamma$ plays a crucial role making it possible to prove that d_{qp} is a valid lower bound on d_{adv}. □

Lemma 4 *Objective function of QPRel is equal to*

$$\frac{1}{2}x^T M^\lambda(W)x + x^T B_1(b, \lambda, \tilde{x}^0) + \|\tilde{x}^0\|^2,$$

where B_1 *does not depend on* x, $\lambda_0 = 1$ *and*

$$M^\lambda_{l,l} = \lambda_l E_l \quad \text{for } l = 0, \ldots, L-1,$$
$$M^\lambda_{l,l-1} = -\frac{1}{2}\lambda_l W^l \quad \text{for } l = 1, \ldots, L-1,$$
$$M^\lambda_{l-1,l} = -\frac{1}{2}\lambda_l (W^l)^T \quad \text{for } l = 1, \ldots, L-1.$$

Proof The proof is done by sorting the quadratic, linear and constant terms in the objective function:

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

From the quadratic term we can identify the blocks of M^λ as claimed. □

C Tables

All tables can be found on pp. 24–27.

Author Contributions SG, AK: Conceptualization; SG, AK: Methodology; SG, AK: Formal analysis and investigation; AK: Writing—original draft preparation; SG, AK: Writing—review and editing; SG: Supervision.

Funding Open Access funding enabled and organized by Projekt DEAL. This research was supported by the BMW AG.

Data availability The authors provide references to all data and material used in this work.

Code availability Custom code is provided including the installation instructions. It requires installation of the gurobi solver, academic licenses are available at gurobi.com.

Declarations

Conflict of interest Not applicable, the authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethical approval The authors approve that the research presented in this paper is conducted following the principles of ethical and professional conduct.

Informed consent The authors consent to participate in ECML PKDD 2022 conference.

Consent for publication Not applicable, the authors use publicly available data only and provide the corresponding references.

References

Bunel, R. R., Turkaslan, I., Torr, P., Kohli, P., & Mudigonda, P. K. (2018). A unified view of piecewise linear neural network verification. *Advances in Neural Information Processing Systems, 31,* 4790–4799.

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

Croce, F., Andriushchenko, M., & Hein, M. (2019). Provable robustness of ReLU networks via maximization of linear regions. In *AISTATS*.

Dua, D., & Graff, C. (2017). UCI machine learning repository. http://archive.ics.uci.edu/ml.

Dutta, S., Jha, S., Sankaranarayanan, S., & Tiwari, A. (2018). Output range analysis for deep feedforward neural networks. In *NASA formal methods* (pp. 121–138). https://doi.org/10.1007/978-3-319-77935-5_9.

Dvijotham, K., Stanforth, R., Gowal, S., Mann, T. A., & Kohli, P. (2018). A dual approach to scalable verification of deep networks. In *Proceedings of the conference on uncertainty in artificial intelligence*. http://auai.org/uai2018/proceedings/papers/204.pdf.

Dvijotham, K., Stanforth, R., Gowal, S., Qin, C., De, S., & Kohli, P. (2019). Efficient neural network verification with exactness characterization. In *Proceedings of the conference on uncertainty in artificial intelligence*. http://auai.org/uai2019/proceedings/papers/164.pdf.

Ehlers, R. (2017). Formal verification of piece-wise linear feed-forward neural networks. In *Automated technology for verification and analysis* (pp. 269–286). Springer

Goodfellow, I., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International conference on learning representations*. arXiv:1412.6572.

Gurobi Optimization L. (2018). *Gurobi optimizer reference manual*. http://www.gurobi.com.

Hein, M., & Andriushchenko, M. (2017). Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in Neural Information Processing Systems, 30,* 2266–2276.

Jordan, M., Lewis, J., & Dimakis, A. G. (2018). Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. In *Advances in neural information processing systems 33*.

Katz, G., Barrett, C., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer aided verification* (pp. 97–117).

Kurakin, A., Goodfellow, I. J., & Bengio, S. (2016). Adversarial examples in the physical world. In *International conference on learning representations*. arXiv:1607.02533.

LeCun, Y., Cortes, C., & Burges, C. J. (1999). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

Lomuscio, A., & Maganti, L. (2017). An approach to reachability analysis for feed-forward ReLU neural networks. arXiv e-prints arXiv:1706.07351.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:170606083.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)* (pp. 372–387).

Raghunathan, A., Steinhardt, J., & Liang, P. (2018a). Certified defenses against adversarial examples. In *International conference on learning representations*.

Raghunathan, A., Steinhardt, J., & Liang, P. S. (2018b). Semidefinite relaxations for certifying robustness to adversarial examples. *Advances in Neural Information Processing Systems, 31,* 10877–10887.

Salman, H., Yang, G., Zhang, H., Hsieh, C. J., & Zhang, P. (2019). A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems, 32,* 9832–9842.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In *International conference on learning representations*. arXiv:1312.6199.

Tjeng, V., Xiao, K., & Tedrake, R. (2017). Evaluating robustness of neural networks with mixed integer programming. arXiv preprint arXiv:171107356.

Tsuzuku, Y., Sato, I., & Sugiyama, M. (2018). Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *Advances in Neural Information Processing Systems, 31,* 6541–6550.

Wang, S., Chen, Y., Abdou, A., & Jana, S. (2018). MixTrain: Scalable training of verifiably robust neural networks. arXiv preprint arXiv:181102625.

Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C. J., Daniel, L., Boning, D., & Dhillon, I. (2018). Towards fast computation of certified robustness for ReLU networks. In *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 5276–5285).

Wong, E., & Kolter, Z. (2018). Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 5286–5295).

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. CoRR arXiv:1708.07747.

Zhang, H., Weng, T. W., Chen, P. Y., Hsieh, C. J., & Daniel, L. (2018). Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems, 31,* 4939–4948.