VERIFYING PROPERTIES OF BINARY NEURAL NETWORKS USING SPARSE POLYNOMIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper explores methods for verifying the properties of Binary Neural Networks (BNNs), focusing on robustness against adversarial attacks. Despite their lower computational and memory needs, BNNs, like their full-precision counterparts, are also sensitive to input perturbations. Established methods for solving this problem are predominantly based on Satisfiability Modulo Theories and Mixed-Integer Linear Programming techniques, which often face scalability issues. We introduce an alternative approach using Semidefinite Programming relaxations derived from sparse Polynomial Optimization. Our approach, compatible with continuous input space, not only mitigates numerical issues associated with floating-point calculations but also enhances verification scalability through the strategic use of tighter first-order semidefinite relaxations. We demonstrate the effectiveness of our method in verifying robustness against both $\|.\|_\infty$ and $\|.\|_2$-based adversarial attacks.

1 INTRODUCTION

In the evolving landscape of machine learning, Binary Neural Networks (BNNs) have emerged as an intriguing class of neural networks since their introduction in 2015 (Courbariaux et al., 2015; Hubara et al., 2016). The architectural simplicity and inherent advantages of BNNs, such as reduced memory requirements and lower computational time, have garnered significant attention. As these networks have matured, they have been integrated into a wide range of machine learning applications (Rastegari et al., 2016; Sun et al., 2018; Xiang et al., 2017; Vorabbi et al., 2024).

Despite their great approximation capacities, general deep neural networks are sensitive to input perturbations (Szegedy et al., 2014; Antun et al., 2021). BNNs are not any different in this regard. Indeed, quantized or binarized networks do not necessarily preserve the properties satisfied by their real-precision counterparts (Galloway et al., 2018; Giacobbe et al., 2020). That is why verifying their robustness is one of the crucial aspects of their design and deployment. A classifying network is said to be robust to adversarial attacks if small input perturbations do not cause any misclassifications. Formally speaking, the network verification consists of finding a point $\boldsymbol{x}$ satisfying $P(\boldsymbol{x}) \wedge Q(\mathbf{BNN}(\boldsymbol{x}))$, where P is a pre-condition on $\boldsymbol{x}$, e.g., stating a valid input perturbation, and Q is a post-condition on $y = \mathbf{BNN}(\boldsymbol{x})$, e.g., stating an (undesirable) alteration of the highest output score. If both properties hold for some $\boldsymbol{x}$, the network is not robust.

Compared to much research on the robustness verification of full-precision networks, verification of BNNs is a question yet to be addressed with more attention. Moreover, many formal verification frameworks do not exploit the bit-precise semantics of BNNs (Giacobbe et al., 2020), and in parallel, those BNN-specific approaches for robustness verification often suffer from limited scalability (Lazarus & Kochenderfer, 2022; Narodytska et al., 2020). Two related methodological axes can be identified. Firstly, methods based on Satisfiability Modulo Theories (SAT/SMT), as in Amir et al. (2021); Jia & Rinard (2020a); Narodytska et al. (2018; 2020), encode the BNN verification problem into Boolean formula satisfiability, and leverage modern off-the-shelf solvers to prove robustness or find counterexamples. Secondly, Khalil et al. (2019); Lazarus & Kochenderfer (2022) cast robustness verification as a Mixed Integer Linear Programming (MILP) optimization problem. Both approaches belong to the group of exact verification methods, i.e., they are sound and complete. In this paper, instead of solving the BNN verification problem exactly, we rather examine it through the lens of (Sparse) Polynomial Optimization Problems (POP) (Lasserre, 2015; Magron

& Wang, 2023), that can be approximated with hierarchies of relaxations based on Semidefinite Programming (SDP). Recent frameworks (Chen et al., 2020; 2021; Latorre et al., 2020; Newton & Papachristodoulou, 2023) have demonstrated that sparse variants of SDP hierarchies could certify the robustness of full-precision ReLU neural networks, by efficiently providing accurate bounds for the associated optimization problems.

1.1 CONTRIBUTION

- We exploit the semi-algebraic nature of the sign activation function to encode the BNN verification problem as a POP. We then solve the resulting first-order SDP relaxation of this POP to obtain lower bounds that can certify robustness. In addition, our method overcomes floating-point-related numerical issues that typically compromise the branch and bound process of MILP solvers. To the best of our knowledge, this is the first SDP-based method for BNN verification.
- From the theoretical point of view, we prove that adding tautologies (redundant constraints) to the initial POP encoding leads to first-order SDP relaxations with highly improved accuracy. Knowing that higher-order SDP relaxations quickly become intractable for high-dimensional problems, designing tighter first-order SDP relaxations that exploit the structure of the network is crucial for enhancing the scalability of the method. We show that our bounds can be up to 55% more accurate than those derived from the linear relaxations typically used in traditional MILP algorithms.
- We demonstrate the effectiveness of our method, compatible with continuous input space, in verifying robustness against both $\|.\|_\infty$ and $\|.\|_2$-based adversarial attacks, the latter being much less studied in the BNN verification literature. Our experimental results indicate that, for $\|.\|_\infty$ and $\|.\|_2$ robustness verification problems, our algorithm can provide an average speedup of 4.5 and 11.4 times, respectively. For some severe attacks, the speedup exceeds a factor of 50.

1.2 RELATED WORKS

SDP-based verification methods: Certifying adversarial robustness using SDP relaxations has been first proposed in Raghunathan et al. (2018). Tightening of the SDP bounds via linear reformulations and quadratic constraints has been proposed in Fazlyab et al. (2020); Batten et al. (2021); Lan et al. (2022). Verification can also be tackled by computing upper bounds of Lipschitz constants (Fazlyab et al., 2019; Latorre et al., 2020; Chen et al., 2020). Chordal and correlative sparsity can be exploited (Anton et al., 2024; Newton & Papachristodoulou, 2023; Chen et al., 2021) to design more efficient SDP relaxations. The first-order dual SDP approach developed in Dathathri et al. (2020) enables efficient verification of high-dimensional models. Note that these approaches have been mainly designed to verify full-precision ReLU networks.

BNN verification: Binarized weights and activation functions allow for the BNN verification problem to be exactly encoded into a Boolean expression. However, solving the resulting encoding via SAT solvers is usually quite computationally expensive as the number of involved variables grows very fast with the network size (Narodytska et al., 2018; Narodytska, 2018). Consequently, only small and medium-sized networks can be handled within this framework. In order to improve the scalability, different architectural and training choices have been proposed: inducing sparse weight matrices, achieving neuron stabilization via input bound propagation, direct search instead of unguided jumps for clause-conflict resolution (Narodytska et al., 2020), enforcing sparse patterns to increase the number of shared computations, neuron-factoring (Narodytska et al., 2020; Cheng et al., 2018). The SAT solver proposed in Jia & Rinard (2020a;b) is tailored for BNN verification. By efficiently handling reified cardinality constraints and exploiting balanced weight sparsification, this solver could achieve significantly faster verification on large networks. The SMT-based framework from Amir et al. (2021) extends the well-known Reluplex (Katz et al., 2017) framework to support binary activation functions. Notice that these works are either incompatible with continuous input data or are restricted to $\|.\|_\infty$ perturbations. In contrast, our approach does not require an extra input quantization step and is compatible with more general perturbation regions, including the one defined by the $\|.\|_2$ norm.

The sole nature of BNNs makes them convenient for being directly represented via linear inequalities and binary variables, enabling the verification problem to be cast as a MILP. These properties are exploited in Khalil et al. (2019), where *big-M constraints* model the neuron activations, and a heuristic named *IProp* decomposes the original problem into smaller MILP problems. However, the big-M approach is also well-known to be sensitive to the M parameter (Gurobi Optimization, LLC). Improved bounds for each neuron have been proposed by Han & Goméz (2021) that lead to tighter relaxations, but only on medium-sized networks. This suggests that BNNs might also suffer from a convex relaxation barrier for tight verification (Salman et al., 2019). Another MILP-based approach has been discussed in Lazarus & Kochenderfer (2022); Lazarus et al. (May, 2022), where both input and output domain of a property to be verified are restricted to polytopes. The set-reachability method from Ivashchenko et al. (2023) extends the *star* set and *Image star* approaches from Bak & Duggirala (2017); Tran et al. (2019; 2020). Unlike most other methods, it allows the input space to be continuous. However, our experiments highlight the intrinsic weaknesses of LP relaxations for BNN verification, negatively influencing the bounding step of MILP solving. We argue that SDP bounds would be able to provide significant speedups to these exact solvers, especially for larger networks and more severe attacks. Somewhat related works concern *quantitative* BNN verification, where one tries to estimate how often a given network satisfies or violates some property. In Baluta et al. (2019); Narodytska et al. (2019), quantitative robustness verification is reduced into a model counting problem over a Conjunctive Normal Form (CNF) expression. SAT-based approaches are derived from either (Ordered) Binary or Sentential Decision Diagrams in Shi et al. (2020); Shih et al. (2019); Zhang et al. (2021).

1.3 NOTATIONS AND PRELIMINARIES

We use Roman letters to denote scalars, and boldfaced letters to represent vectors and matrices. If $\boldsymbol{A}$ is a matrix, then $\boldsymbol{A}_{(k,:)}$ denotes its k-th row vector, and $\|\boldsymbol{A}\|_F$ denotes its Frobenius norm. The Hadamard product is denoted by $\odot$, i.e., $(\boldsymbol{A} \odot \boldsymbol{B})_{ij} = \boldsymbol{A}_{ij}\boldsymbol{B}_{ij}$. The ring of n-variate real polynomials (resp. of at most degree d) is denoted by $\mathbb{R}[\boldsymbol{x}]$ (resp. $\mathbb{R}[\boldsymbol{x}]_d$). Let $\Sigma[\boldsymbol{x}]$ be the cone of multivariate Sum Of Squares (SOS) polynomials and $\Sigma[\boldsymbol{x}]_d := \Sigma[\boldsymbol{x}] \cap \mathbb{R}[\boldsymbol{x}]_{2d}$. By $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^n x_i y_i$ we denote the standard inner product of vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. If $k_1, k_2 \in \mathbb{N}$ with $k_1 \leq k_2$, then $[\![k_1, k_2]\!] := \{k_1, \ldots, k_2\}$. Let $\mathbb{B}_{||.||}(\bar{\boldsymbol{x}}, \varepsilon)$ be the $||.||$-ball of radius ε centered at $\bar{\boldsymbol{x}}$. We denote by $\mathbb{S}_+^n$ the set of symmetric positive semidefinite matrices of size n. We recall that $f \in \mathbb{R}[\boldsymbol{x}]_{2d}$ is SOS if and only if a positive semidefinite (PSD) matrix $\boldsymbol{G}$ satisfies $f = \boldsymbol{v}_d^\intercal \boldsymbol{G} \boldsymbol{v}_d$, where $\boldsymbol{v}_d := (1, x_1, \ldots, x_n, \ldots, x_n^d)^\intercal$ is the vector of monomials of degree at most d with size $s(d) := \binom{n+d}{d}$.

Definition 1.1. *Let* $\boldsymbol{g} := (g_j)_{j \in [\![1,m]\!]}$ *and* $\boldsymbol{h} := (h_k)_{k \in [\![1,l]\!]}$ *denote families of polynomial functions. For all* $j \in [\![1, m]\!]$, $k \in [\![1, l]\!]$, *define* $d_j := \lceil \deg(g_j)/2 \rceil$ *and* $\bar{d}_k := \deg(h_k)$. *Then, the* d*-truncated quadratic module* $\mathcal{Q}_d(\boldsymbol{g})$ *generated by* $\boldsymbol{g}$*, and the* d*-truncated ideal* $\mathcal{I}_d(\boldsymbol{h})$ *generated by* $\boldsymbol{h}$ *are*

$$\mathcal{Q}_d(\boldsymbol{g}) := \left\{ \sigma_0 + \sum_{j=1}^m \sigma_j g_j \mid \sigma_0 \in \Sigma[\boldsymbol{x}]_d, \sigma_j \in \Sigma[\boldsymbol{x}]_{d-d_j}, j \in [\![1, m]\!] \right\}, \tag{1}$$

$$\mathcal{I}_d(\boldsymbol{h}) := \left\{ \sum_{k=1}^l \psi_k h_k, \psi_k \in \mathbb{R}[\boldsymbol{x}]_{2d-\bar{d}_k}, k \in [\![1, l]\!] \right\}. \tag{2}$$

2 MAIN INGREDIENTS

2.1 BINARY NEURAL NETWORKS

Let $L \geq 1$ be the number of hidden layers of a classifying BNN, with layer widths being given by $\boldsymbol{n} = (n_0, n_1, \ldots, n_L, n_{L+1})^\intercal \in \mathbb{N}^{L+2}$, where n_0 and n_{L+1} are input and output dimensions. A feed-forward BNN is a mapping from the input region $\mathcal{R}_{n_0} \subset \mathbb{R}^{n_0}$ to the output set $[\![1, n_{L+1}]\!]$ realized via successive compositions of several internal blocks $(\mathbf{B}_i)_{i=1,\ldots,L}$ and an output block $\mathbf{B}_o$:

$$\begin{aligned} \text{BNN} : \mathcal{R}_{n_0} &\to [\![1, n_{L+1}]\!] \\ \boldsymbol{x}^0 &\mapsto \text{BNN}(\boldsymbol{x}^0) := \text{argmax}\left(\mathbf{B}_o(\mathbf{B}_L(\ldots(\mathbf{B}_1(\boldsymbol{x}^0))))\right). \end{aligned} \tag{3}$$

For any $i \in [\![1, L]\!]$, the internal block $\mathbf{B}_i$ implements successively three different operations: affine transformation, batch normalization[1] and point-wise binarization, so that its output vector, denoted

---

[1] Since batch normalization can be understood as another affine transformation, it can be omitted throughout the technical modelling part, w.l.o.g.

by $\boldsymbol{x}^i$, belongs to $\{-1, 1\}^{n_i}$. These operations are described by a set of trainable parameters:

$$\left(\boldsymbol{W}^{[i+1]}, \boldsymbol{b}^{[i+1]}\right)_{i \in [\![0\,,\,L]\!]} \in \{-1, 0, 1\}^{n_{i+1} \times n_i} \times \mathbb{R}^{n_{i+1}}, \tag{4}$$

$$\left(\boldsymbol{\gamma}^{[i]}, \boldsymbol{\beta}^{[i]}, \boldsymbol{\mu}^{[i]}, \boldsymbol{\sigma}^{2,[i]}\right)_{i \in [\![1\,,\,L]\!]} \in \left(\mathbb{R}^{n_i}\right)^4 . \tag{5}$$

Consequently, the output of a neuron $j \in [\![1\,,\,n_i]\!]$ from the hidden layer $i \in [\![1\,,\,L]\!]$ is given by $\boldsymbol{x}^i_j = \operatorname{sign}\left(\boldsymbol{\gamma}^i_j \frac{\left(\left\langle \boldsymbol{W}^{[i]}_{(j:,)}, \boldsymbol{x}^{i-1}\right\rangle + \boldsymbol{b}^{[i]}_j - \boldsymbol{\mu}^{[i]}_j\right)}{\sqrt{\boldsymbol{\sigma}^{2,[i]}_j + \varepsilon}} - \boldsymbol{\beta}^i_j\right)$, with small enough $\varepsilon > 0$. The output block $\mathbf{B}_o$ applies a softmax transformation to the affinely-transformed outputs of the last hidden layer, i.e., for each $j \in [\![1\,,\,n_{L+1}]\!]$, $\boldsymbol{x}^{L+1}_j = \frac{\exp(z_j)}{\sum_{k=1}^{n_{L+1}} \exp(z_k)}$, where $z_j = \boldsymbol{W}^{[L+1]}_{(j:,)} \boldsymbol{x}^L + \boldsymbol{b}^{[L+1]}_j$.

In BNNs, replacing vector-matrix multiplications with simpler 1-bit XNOR-count operations comes at a cost: the sign function prevents effective (adversarial) training due to the lack of proper gradient information. For a comprehensive analysis and the latest advancements in training (robust) BNNs, see Qin et al. (2020); Yuan & Agaian (2021); Aspman et al. (2024).

2.2 PROBLEM FORMULATION

Casting BNN verification as an optimization problem requires using an appropriate representation of the non-linear sign$(\cdot)$ activation function. For any $(a, b) \in \mathbb{R}^2$, we have:

$$a = \operatorname{sign}(b) \implies a^2 - 1 = 0, \text{ and } ab \geq 0. \tag{6}$$

This motivates the introduction of a sequence of vector-valued functions $(\boldsymbol{h}_i, \boldsymbol{g}_i)_{i \in [\![1\,,\,L]\!]}$ such that

$$\boldsymbol{x}^i := \operatorname{sign}\left(\boldsymbol{W}^{[i]} \boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}\right) \implies \begin{cases} \boldsymbol{h}_i(\boldsymbol{x}^i) := \boldsymbol{x}^i \odot \boldsymbol{x}^i - \mathbf{1} = \mathbf{0}, & \text{(7a)} \\ \boldsymbol{g}_i(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := \boldsymbol{x}^i \odot (\boldsymbol{W}^{[i]} \boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}) \geq \mathbf{0}, & \text{(7b)} \end{cases}$$

where $\boldsymbol{b}^{[i]} \in \mathbb{R}^{n_i}$ satisfies $\left|\boldsymbol{b}^{[i]}_k\right| < \left\|\boldsymbol{W}^{[i]}_{(k:,)}\right\|_1$ for each $k \in [\![1\,,\,n_i]\!]$. This assumption eliminates the case in which some neurons are either always or never activated, since such neurons have no impact on the verification process. Furthermore, we suppose that the input perturbation region $\boldsymbol{B} \subseteq \mathbb{R}^{n_0}$ can be encoded via positivity conditions on (at most quadratic) polynomials $\boldsymbol{x}^0 \mapsto \boldsymbol{g}_{\boldsymbol{B}}(\boldsymbol{x}^0)$. For example, $\boldsymbol{g}_{\boldsymbol{B}}(\boldsymbol{x}^0) = (\boldsymbol{\varepsilon} + \bar{\boldsymbol{x}} - \boldsymbol{x}^0) \odot (\boldsymbol{\varepsilon} - \bar{\boldsymbol{x}} + \boldsymbol{x}^0)$ corresponds to $\boldsymbol{B} = \mathbb{B}_{||\cdot||_\infty}(\bar{\boldsymbol{x}}, \varepsilon)$, where $\bar{\boldsymbol{x}} \in \mathcal{R}_{n_0}$. Finally, the *standard* form BNN verification problems studied here are:

$$\tau := \begin{cases} \min\limits_{\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L} f(\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L) & \text{(8a)} \\ \text{s.t.} \quad \boldsymbol{h}_i(\boldsymbol{x}^i) = \mathbf{0},\ i \in [\![1\,,\,L]\!], & \text{(8b)} \\ \qquad\ \boldsymbol{g}_i(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) \geq \mathbf{0},\ i \in [\![1\,,\,L]\!], & \text{(8c)} \\ \qquad\ \boldsymbol{g}_{\boldsymbol{B}}(\boldsymbol{x}^0) \geq \mathbf{0}, & \text{(8d)} \end{cases}$$

where f is either a linear or a quadratic function.

Remark 2.1 (Adversarial attacks)**.** *Given some point $\bar{\boldsymbol{x}} \in \mathcal{R}_{n_0}$ whose true label is $\bar{y} \in [\![1\,,\,n_{L+1}]\!]$, we define a k-targeted attack to be any allowed perturbation $\boldsymbol{x}^0$ satisfying (8b)-(8d) for which the output of the network is $k \neq \bar{y}$. If the network is robust, then*

$$f^{\mathrm{adv}}_k(\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L) := \left\langle \boldsymbol{W}^{[L+1]}_{(\bar{y}:,)} - \boldsymbol{W}^{[L+1]}_{(k:,)}, \boldsymbol{x}^L \right\rangle + \boldsymbol{b}^{[L+1]}_{\bar{y}} - \boldsymbol{b}^{[L+1]}_k \tag{9}$$

is always positive. Notice that f^{adv}_k is an affine mapping of the neurons from the last hidden layer. Our framework is suitable for verifying properties of BNNs *other than adversarial robustness (e.g. certain properties describing the ACAS-Xu controller from Katz et al. (2017), or energy conservation in dynamical systems (Qin et al., 2019)). Verification against non-targeted attacks could be achieved via a simple objective function modification.*

2.3 SPARSE POLYNOMIAL OPTIMIZATION

Notice that (8) is an instance of Quadratically Constrained Quadratic Programming (QCQP), involving $n := \sum_{i=0}^{L} n_i$ decision variables and $m := 2\sum_{i=1}^{L} n_i + n_{\boldsymbol{B}}$ constraints, where $n_{\boldsymbol{B}}$ is the number of polynomials (at most quadratic) needed to represent the input region $\boldsymbol{B}$. As such, problem (8) is a special case of Polynomial Optimization (POP) since one minimizes a polynomial f over a feasible set S defined with finitely many polynomial (in)equality constraints. Here $S := \{\boldsymbol{x} \mid \boldsymbol{g}_i(\boldsymbol{x}) \geq \boldsymbol{0}, \boldsymbol{h}_i(\boldsymbol{x}) = \boldsymbol{0},\ \forall i \in [\![1, L]\!],\ \boldsymbol{g}_{\boldsymbol{B}}(\boldsymbol{x}^0) \geq \boldsymbol{0}\}$. Note that an equivalent characterization of the global infimum of f on S is $\tau = \min_{\boldsymbol{x}}\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in S\} = \max_{\lambda}\{\lambda \in \mathbb{R} \mid f - \lambda \geq 0 \text{ on } S\}$. This requires one to efficiently handle the set of polynomials that are nonnegative on S, which is known to be intractable. However, in practice, we can rely on its *tractable* inner approximations based on weighted combinations of elements in $\boldsymbol{g} := \{(\boldsymbol{g}_i)_{i\in[\![1,L]\!]}, \boldsymbol{g}_{\boldsymbol{B}}\}$, the weights being SOS polynomials. By Lasserre (2001), $\tau^d := \sup_{\lambda,\sigma}\{\lambda \in \mathbb{R} \mid f - \lambda - \sigma \in \mathcal{I}_d(\boldsymbol{h}), \sigma \in \mathcal{Q}_d(\boldsymbol{g})\}$ defines a hierarchy of *dense* SDP relaxations whose size increases with d, and such that $\tau^d \uparrow \tau$ as $d \to +\infty$. Moreover, generically the convergence to τ is finite, i.e., $\tau^d = \tau$ for some $d \in \mathbb{N}$ (Nie, 2014).

Example 2.1. *Consider a BNN with* $L = 2$, $(n_0, n_1, n_2, n_3) = (3, 2, 2, 2)$, *with trainable parameters given by* $\left(\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]}, \boldsymbol{W}^{[3]}\right) = \left(\begin{pmatrix} -1 & 1 & 1 \\ -1 & -1 & 1 \end{pmatrix}, \begin{pmatrix} -1 & -1 \\ -1 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 1 \\ -1 & -1 \end{pmatrix}\right)$, *and* $\left(\boldsymbol{b}^{[1]}, \boldsymbol{b}^{[2]}, \boldsymbol{b}^{[3]}\right) = \left(\begin{pmatrix} 1.5 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ -0.5 \end{pmatrix}, \begin{pmatrix} -2 \\ -1 \end{pmatrix}\right)$. *Suppose* $g_{\boldsymbol{B}}(\boldsymbol{x}^0) = 0.2^2 - (\boldsymbol{x}^0 - \bar{\boldsymbol{x}})^\intercal(\boldsymbol{x}^0 - \bar{\boldsymbol{x}})$*, with* $\bar{\boldsymbol{x}} = (0, 0.5, 0)^\intercal$. *The network assigns the label* $\bar{y} = 2$ *since* $\boldsymbol{x}_2^3 = 1 > \boldsymbol{x}_1^3 = -2$. *By Remark 2.1, the affine objective function to minimize becomes* $\boldsymbol{x} \mapsto f_1^{\mathrm{adv}}(\boldsymbol{x}) = -(-\boldsymbol{x}_1^2 - \boldsymbol{x}_2^2 + \boldsymbol{x}_1^2 - \boldsymbol{x}_2^2 - 1 + 2) = 2\boldsymbol{x}_2^2 - 1$. *The corresponding dense SDP relaxation of order* $d \geq 1$ *becomes*

$$\tau^d = \begin{cases} \displaystyle\sup_{\lambda, \{\boldsymbol{G}_{i,j}\}_{i,j=1}^2, \boldsymbol{G}_0, \boldsymbol{G}_{\boldsymbol{B}}} \lambda & \\ \text{s.t.} \quad f_1^{\mathrm{adv}} - \lambda - \sigma \in \mathcal{I}_d(\{\boldsymbol{h}_1, \boldsymbol{h}_2\}), & \text{(10a)} \\ \qquad \sigma = \boldsymbol{v}_d^\intercal \boldsymbol{G}_0 \boldsymbol{v}_d + \displaystyle\sum_{i,j=1}^{2} \boldsymbol{v}_{d-1}^\intercal \boldsymbol{G}_{i,j} \boldsymbol{v}_{d-1} \boldsymbol{g}_i(\cdot)_j + \boldsymbol{v}_{d-1}^\intercal \boldsymbol{G}_{\boldsymbol{B}} \boldsymbol{v}_{d-1} g_{\boldsymbol{B}}, & \text{(10b)} \\ \qquad \boldsymbol{G}_{\boldsymbol{B}}, \boldsymbol{G}_{i,j} \in \mathbb{S}_+^{s(d-1)}, i, j \in [\![1, 2]\!], \boldsymbol{G}_0 \in \mathbb{S}_+^{s(d)}. & \text{(10c)} \end{cases}$$

In addition to SDP conditions from (10c), checking for membership in $\mathcal{I}_d(\{\boldsymbol{h}_1, \boldsymbol{h}_2\})$ *in the line (10a) boils down to imposing linear conditions on the coefficients of the involved polynomials. Those can be obtained after applying the substitution rules* $\boldsymbol{h}_i(\boldsymbol{x}^i) = 0 \iff (\boldsymbol{x}_j^i)^2 = 1$ *for* $i, j \in [\![1, 2]\!]$.

For a fixed relaxation order d, dense SDP relaxations involve $\mathcal{O}(n^{2d})$ equality constraints, which prevents them from being applied to large-scale problems. However, many POP problems, including BNN robustness verification, exhibit important structural sparsity properties, which enables one to build significantly more computationally efficient SDP relaxations (Waki et al., 2006; Wang et al., 2021). For instance, let us suppose that $[\![1, n]\!] =: I_0 = \cup_{k=1}^{p} I_k$ with I_k not necessarily disjoint. The subsets I_k, called *cliques*, correspond to the subsets of variables $\boldsymbol{x}_{I_k} := \{x_i, i \in I_k\}$. An instance of the BNN robustness verification problem of the form (8) exhibits *correlative sparsity* since

- There exist $(f_k)_{k\in[\![1,p]\!]}$ such that $f = \sum_{k=1}^{p} f_k$, with $f_k \in \mathbb{R}[\boldsymbol{x}_{I_k}]$,
- The polynomials $\boldsymbol{g}$ can be split into disjoints sets J_k, such that $\boldsymbol{g}_i(\cdot)_j \in J_k$ if and only if $\boldsymbol{g}_i(\cdot)_j \in \mathbb{R}[\boldsymbol{x}_{I_k}]$. Moreover, $\boldsymbol{g}_{\boldsymbol{B}} \in J_k$ for $k \in [\![1, p]\!]$. Since $\boldsymbol{h}_i(\cdot)_j$ only depends on $\boldsymbol{x}_j^i$, the overall sparsity structure is induced by inequality constraints that mimic the cascading BNN structure.

As in the dense case, a hierarchy of correlatively sparse SDP relaxations is given by $\tau_{\mathrm{cs}}^d := \sup_{\lambda,\sigma}\{\lambda \in \mathbb{R} \mid f - \lambda - \sum_{k=1}^{p} \sigma_k \in \mathcal{I}_d(\boldsymbol{h}), \sigma_k \in \mathcal{Q}_d(\{\boldsymbol{g}_i(\cdot)_j \in J_k\})\}$. Under additional ball constraints (see, e.g., (Magron & Wang, 2023, Assumption 3.1)), one still has $\tau_{\mathrm{cs}}^d \uparrow \tau$ as $d \to \infty$. If $\rho := \max_k |I_k|$, then these relaxations involve $\mathcal{O}(p\rho^{2d})$ equality constraints, yielding a significant improvement when $\rho \ll n$. Apart from this computational gain, we will benefit from the fact that the first-order sparse relaxation is not conservative w.r.t. the dense one, i.e., $\tau_{\mathrm{cs}}^1 = \tau^1$, see, e.g., (Vandenberghe et al., 2015, Theorem 9.2).

3 COMPARISON OF LINEAR PROGRAMMING (LP) AND SDP BOUNDS

Here, we assume that f is linear, e.g., the function f_k^{adv} defined in (9). The goal of this section is to compare the LP and SDP relaxations of the QCQP encoding (8), when the perturbation region is described by $\boldsymbol{B} = \mathbb{B}_{||\cdot||_\infty}(\bar{\boldsymbol{x}}, \varepsilon) = \{\boldsymbol{x} \mid \bar{\boldsymbol{x}} - \boldsymbol{\varepsilon} \leq \boldsymbol{x} \leq \boldsymbol{\varepsilon} + \bar{\boldsymbol{x}}\}$. Let $n \geq 2,\ \boldsymbol{w} \in \{-1,1\}^n$, and $b \in \mathbb{R}$ satisfying $|b| < n$. As pointed out by Amir et al. (2021), an equivalent representation of the set $\{(\boldsymbol{x}, y) \in [-1,1]^n \times \{-1,1\}, y = \mathrm{sign}(\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b)\}$ is given by

$$\left\{(\boldsymbol{x}, y) \in [-1,1]^n \times \{-1,1\}, y \geq \frac{2\left(\langle \boldsymbol{x}, \boldsymbol{w}\rangle + b\right)}{n+b} - 1,\ y \leq \frac{2\left(\langle \boldsymbol{x}, \boldsymbol{w}\rangle + b\right)}{n-b} + 1\right\}. \tag{11}$$

Hence, for each layer $i \in [\![1, L]\!]$, we can replace the quadratic function $\boldsymbol{g}_i$ by two linear functions

$$\begin{cases} \boldsymbol{g}^1_{i,\mathrm{LIN}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\mathrm{nv}(\boldsymbol{W}^{[i]}) + \boldsymbol{b}^{[i]}) \odot (\boldsymbol{x}^i + \mathbf{1}) - 2\left(\boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}\right), & \text{(12a)} \\ \boldsymbol{g}^2_{i,\mathrm{LIN}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\mathrm{nv}(\boldsymbol{W}^{[i]}) - \boldsymbol{b}^{[i]}) \odot (\mathbf{1} - \boldsymbol{x}^i) + 2\left(\boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}\right), & \text{(12b)} \end{cases}$$

where for any $p \times q$ matrix $\boldsymbol{A}$, $\mathrm{nv}(\boldsymbol{A})$ is a p-dimensional column vector whose k-th coordinate is given by $||\boldsymbol{A}_{(k,:)}||_1$. Thus, by encoding the $\mathrm{sign}(\cdot)$ function as explained in (12a)-(12b), we obtain an alternative MILP formulation of the standard BNN verification problem:

$$\tau_{\mathrm{MILP}} := \begin{cases} \min\limits_{\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L} f(\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L) & \\ \text{s.t.} \quad \boldsymbol{g}^1_{i,\mathrm{LIN}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) \geq \mathbf{0},\ i \in [\![1, L]\!], & \text{(13a)} \\ \qquad \boldsymbol{g}^2_{i,\mathrm{LIN}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) \geq \mathbf{0},\ i \in [\![1, L]\!], & \text{(13b)} \\ \qquad \boldsymbol{g}_{\boldsymbol{B}}(\boldsymbol{x}^0) \geq \mathbf{0}, & \text{(13c)} \\ \qquad \boldsymbol{x}^i \in \{-1,1\}^{n_i},\ \ i \in [\![1, L]\!]. & \text{(13d)} \end{cases}$$

By further relaxing the binary constraints in (13d) via $\boldsymbol{g}^0_{i,\mathrm{LIN}} := (\mathbf{1} - \boldsymbol{x}^i, \boldsymbol{x}^i + \mathbf{1})$, we can formulate the corresponding LP relaxation of the MILP problem in (13)[2]:

$$\tau_{\mathrm{LP}} := \begin{cases} \min\limits_{\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L} f(\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L) & \\ \text{s.t.} \quad \boldsymbol{g}^0_{i,\mathrm{LIN}}(\boldsymbol{x}^i) \geq \mathbf{0}, i \in [\![1, L]\!], & \text{(14a)} \\ \qquad (13a) - (13c). & \text{(14b)} \end{cases}$$

Remark 3.1 (Encoding of $\mathrm{sign}(\cdot)$)**.** *In Lazarus & Kochenderfer (2022); Khalil et al. (2019), the authors have also considered a* MILP *encoding of the* BNN *verification problem, based on* $l, u \in \mathbb{R}$ *such that* $l \leq \langle \boldsymbol{x}, \boldsymbol{w}\rangle + b \leq u$ *and*

$$y = \mathrm{sign}(\langle \boldsymbol{x}, \boldsymbol{w}\rangle + b) \implies \frac{4}{u}(\langle \boldsymbol{x}, \boldsymbol{w}\rangle + b) - 3 \leq y \text{ and } y \geq -\frac{4}{l}\left(\langle \boldsymbol{x}, \boldsymbol{w}\rangle + b\right) + 1. \tag{15}$$

For $\boldsymbol{w} \in \{-1,1\}^n$ *and* $u = n + b$*, we conclude that (11) provides a tighter bound than (15) because*

$$y - \left(\frac{4}{n+b}\left(\langle \boldsymbol{x}, \boldsymbol{w}\rangle + b\right) - 3\right) = y - \left(\frac{2}{n+b}\left(\langle \boldsymbol{x}, \boldsymbol{w}\rangle + b\right) - 1\right) + \sum_{k=1}^{n} \frac{\left(1 - w_k x_k\right)^2}{n+b}. \tag{16}$$

Theorem 3.1. *For an arbitrary* BNN *with depth* $L \geq 2$*, there always exists an affine function* $f : \mathbb{R}[\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L] \to \mathbb{R}$ *such that* $\tau_{\mathrm{LP}} > \tau^1 = \tau^1_{\mathrm{cs}}$.

Sketch of proof. Consider $f(\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L) = \boldsymbol{g}^1_{L,\mathrm{LIN}}(\boldsymbol{x}^L, \boldsymbol{x}^{L-1})_j$ for some $j \in [\![1, n_L]\!]$. Then Appendix A.1 explicitly constructs a feasible solution to the dual of the sparse first-order SDP relaxation of (8), yielding the value of the objective function equal to -1, implying $\tau^1_{\mathrm{cs}} = \tau^1 \leq -1$. Since $f(\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L) \geq 0$ is one of the constraints in (14), we deduce that $\tau_{\mathrm{LP}} \geq 0$. ☐

Theorem 3.1 states that the lower bound τ^1 is generally not competitive. On the other hand, the inequality $\tau^2 \geq \tau_{\mathrm{LP}}$ is derived in Appendix A.2, but the second-order SDP relaxation remains computationally inefficient for high-dimensional problems.

---

[2] Replacing the initial sign constraint by $\boldsymbol{g}^1_{1,\mathrm{LIN}}$ and $\boldsymbol{g}^2_{1,\mathrm{LIN}}$ provides a valid encoding only if $\boldsymbol{x}^0 \in [-1,1]^{n_0}$. However, we can always, w.l.o.g, transform the input perturbation region so that it corresponds to $\mathbb{B}_{||\cdot||_\infty}(\mathbf{0}, 1)$.

4 TIGHTENING OF THE FIRST-ORDER SDP RELAXATION

The goal of this section is to propose a more accurate first-order SDP relaxation. We still assume that $\boldsymbol{B} = \mathbb{B}_{||\cdot||_\infty}(\bar{\boldsymbol{x}}, \varepsilon)$. Firstly, notice that the semi-algebraic representation of the subgradient of the ReLU function derived in Chen et al. (2020) provides an alternative encoding of the sign$(\cdot)$ function. Namely, for each $i \in [\![1, L]\!]$, we replace the constraint defined in (7b) by the following two constraints:

$$\begin{cases} \tilde{\boldsymbol{g}}_i^1(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\boldsymbol{x}^i + \mathbf{1}) \odot \left(\boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}\right) \geq \mathbf{0}, & \text{(17a)} \\ \tilde{\boldsymbol{g}}_i^2(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\boldsymbol{x}^i - \mathbf{1}) \odot \left(\boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}\right) \geq \mathbf{0}. & \text{(17b)} \end{cases}$$

Furthermore, we include the following two *redundant* quadratic constraints (tautologies):

$$\begin{cases} \tilde{\boldsymbol{g}}_i^{\mathrm{t1}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\boldsymbol{x}^i + \mathbf{1}) \odot \left(\mathrm{nv}\left(\boldsymbol{W}^{[i]}\right) - \boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1}\right) \geq \mathbf{0}, & \text{(18a)} \\ \tilde{\boldsymbol{g}}_i^{\mathrm{t2}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\mathbf{1} - \boldsymbol{x}^i) \odot \left(\mathrm{nv}\left(\boldsymbol{W}^{[i]}\right) + \boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1}\right) \geq \mathbf{0}, & \text{(18b)} \end{cases}$$

which hold true for $\boldsymbol{x}^{i-1} \in \{-1, 1\}^{n_{i-1}}$ and $\left|\boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1}\right| \leq \mathrm{nv}\left(\boldsymbol{W}^{[i]}\right)$, where $|\cdot|$ should be understood component-wise. Hence, we get an alternative POP encoding of the BNN verification problem:

$$\tau_{\text{tighter}} := \begin{cases} \min\limits_{\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L} f(\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L) & \\ \text{s.t.} \quad \boldsymbol{h}_i(\boldsymbol{x}^i) = \mathbf{0},\ i \in [\![1, L]\!], & \text{(19a)} \\ \qquad \tilde{\boldsymbol{g}}_i^1(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) \geq \mathbf{0},\ \tilde{\boldsymbol{g}}_i^{\mathrm{t1}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) \geq \mathbf{0},\ i \in [\![1, L]\!], & \text{(19b)} \\ \qquad \tilde{\boldsymbol{g}}_i^2(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) \geq \mathbf{0},\ \tilde{\boldsymbol{g}}_i^{\mathrm{t2}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) \geq \mathbf{0},\ i \in [\![1, L]\!], & \text{(19c)} \\ \qquad \boldsymbol{g}_{\boldsymbol{B}}(\boldsymbol{x}^0) \geq \mathbf{0}, & \text{(19d)} \end{cases}$$

and corresponding dense and sparse hierarchies of SDP relaxations, i.e., $(\tau^d_{\text{tighter}})_d$ and $(\tau^d_{\text{tighter,cs}})_d$.

Theorem 4.1. *For any* BNN *verification problem,* $\tau^1_{\text{tighter,cs}} = \tau^1_{\text{tighter}} \geq \tau^1$. *If* $L \geq 2$, *there exists an affine* f *such that the inequality is strict. We also have* $\tau^1_{\text{tighter,cs}} \geq \tau_{\text{LP}}$ *for any* f *affine.*

Theorem 4.1 asserts that adding tautologies (18a) and (18b) is crucial, as they allow to generate a larger first-order quadratic module. See Appendix A.3 for the proof and experimental illustration.

Illustration for the case $L = 2$ (refer to Appendix A.4 for the general case)**:** Generally, we have $n_0 + n_2$ cliques of size $n_1 + 1$, given by $I_k = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}, \boldsymbol{x}^0_k\}$ for $k \in [\![1, n_0]\!]$ and $I_{n_0+k} = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}, \boldsymbol{x}^2_k\}$ for $k \in [\![1, n_2]\!]$. [3]

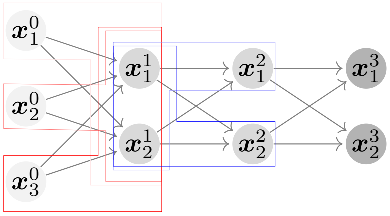


Figure 1: A toy BNN with $L = 2$ and $(n_0, n_1, n_2, n_3) = (3, 2, 2, 2)$. The cliques I_1, I_2, I_3 (red polygons) and I_4, I_5 (blue polygons) are used to compute $\tau^1_{\text{tighter,cs}}$.

---

[3] Notice that, unlike in Newton & Papachristodoulou (2023), this clique structure is not dependent on the input size n_0, which enhances its computational efficiency.

For the BNN from Example 2.1, depicted in Figure 1, most SOS multipliers are non-negative real numbers if $d = 1$. With $\tilde{\boldsymbol{g}}_i = \{\tilde{\boldsymbol{g}}_i^1, \tilde{\boldsymbol{g}}_i^2, \tilde{\boldsymbol{g}}_i^{\mathrm{t}1}, \tilde{\boldsymbol{g}}_i^{\mathrm{t}2}\}$, the tighter first-order sparse SDP relaxation writes:

$$\tau^1_{\text{tighter,cs}} = \begin{cases} \sup\limits_{\lambda, \{\boldsymbol{\sigma_g}\}_{\boldsymbol{g} \in \tilde{\boldsymbol{g}}_i, i \in [\![1,2]\!]}, \sigma_{\boldsymbol{B}}, \{\boldsymbol{G}_k\}_{k=1}^5} \quad \lambda \\ \text{s.t.} \quad f_1^{\text{adv}} - \lambda - \sigma \in \mathcal{I}_1(h), & \text{(20a)} \\ \qquad \sigma(\boldsymbol{x}) = \sum_{i=1}^{2} \sum_{\boldsymbol{g} \in \tilde{\boldsymbol{g}}_i} \mathbf{1}^\intercal (\boldsymbol{\sigma_g} \odot \boldsymbol{g}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})) + \sum_{k=1}^{5} \sigma_{0,k}(\boldsymbol{x}_{I_k}) + \sigma_{\boldsymbol{B}} g_{\boldsymbol{B}}(\boldsymbol{x}^0), & \text{(20b)} \\ \qquad \sigma_{0,k}(\boldsymbol{x}_{I_k}) = \boldsymbol{v}_1(\boldsymbol{x}_{I_k})^\intercal \boldsymbol{G}_k \boldsymbol{v}_1(\boldsymbol{x}_{I_k}), \boldsymbol{G}_k \in \mathbb{S}_+^{|I_k|+1}, k \in [\![1,5]\!], & \text{(20c)} \\ \qquad \boldsymbol{\sigma_g} \geq \mathbf{0}, \boldsymbol{g} \in \tilde{\boldsymbol{g}}_i, i \in [\![1,2]\!], \qquad \sigma_{\boldsymbol{B}} \geq 0. & \text{(20d)} \end{cases}$$

5 NUMERICAL EXPERIMENTS

In this section, we provide numerical results for BNN robustness verification problems with respect to different perturbations. All experiments are run on a desktop with a 12-core i7-12700 2.10 GHz CPU and 32GB of RAM. The tightened first-order sparse SDP relaxation is modeled with TSSOS (Magron & Wang, 2021) and solved with Mosek (Andersen & Andersen, 2000). Gurobi (Gurobi Optimization, LLC, 2023) is used to solve MILP. BNNs were trained on standard benchmark datasets, using Larq Geiger & Team (2020). The full experimental setup is detailed in Appendix B.

5.1 ROBUSTNESS AGAINST $\|.\|_\infty$ ATTACKS

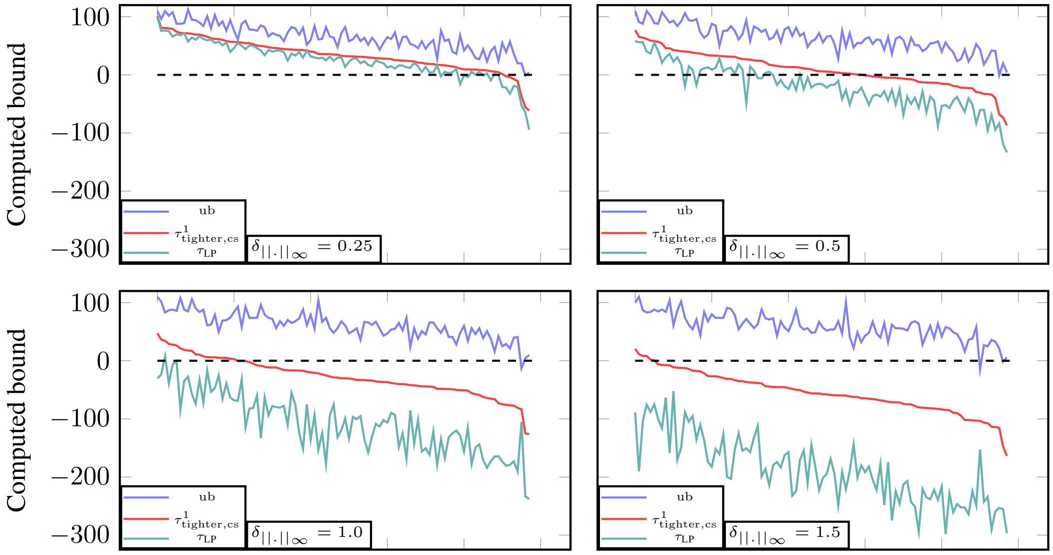
MNIST dataset was used to train two sparse networks - BNN$_1$ and BNN$_2$, where sparsity refers to *weights sparsity* defined by $w_s =: \frac{1 - \sum_{i=0}^{L} \|\boldsymbol{W}^{[i+1]}\|_F^2}{\sum_{i=0}^{L} n_i n_{i+1}}$. We assess the performance of our method (number of solved cases (cert.) and verification time $t\,(s)$) in verifying robustness of the first 100 test instances. Obtained results, see Table 1, are compared with both LP (14) and MILP (13) methods, where the latter was converted into a feasibility problem by adding $f(\cdot) \leq 0$ to the constraint set.

Our method is on average 15.75% (for BNN$_1$) and 25.67% (for BNN$_2$) less conservative than the LP-based method. Moreover, larger input regions do not significantly affect performance, compared to MILP, where the impact on running time is much more severe. This is partially due to the coarseness of LP bounds, see Figure 2, and is even more prominent for densely connected networks.

Table 1: Performance comparison for different models and input regions, given by $\delta_{||.||_\infty} = 127.5\epsilon$ (data were scaled to $[-1, 1]^{784}$). We use | to separate the number of MILP-solver-certified non-robust and robust instances, within a time limitation of 600 s. The runtime in parentheses refers to the average runtime over the instances that our method verified successfully.

Model	$\delta_{\|\|.\|\|_\infty}$	τ_{LP}		$\tau^1_{\text{tighter,cs}}$		$\tau_{\text{Soft-MILP}}$	
		cert.	$t\,(s)$	cert.	$t\,(s)$	cert.	$t\,(s)$
BNN$_1$: $[784, 500, 500, 10]$ $w_s = 34.34\%$	0.25	83	0.01	91	3.62 (3.58)	3 | 95	0.04 (0.04)
	0.50	31	0.02	60	6.69 (6.50)	4 | 94	1.21 (0.06)
	1.00	1	0.03	21	10.76 (8.22)	15 | 50	251.90 (1.37)
	1.50	0	0.06	6	38.32 (26.99)	20 | 12	428.24 (191.95)
BNN$_2$: $[784, 500, 500, 10]$ $w_s = 19.07\%$	0.25	14	0.03	59	11.97 (10.64)	3 | 95	2.23 (0.69)
	0.50	0	0.05	23	42.37 (24.21)	9 | 63	220.53 (13.24)
	0.75	0	0.08	9	139.18 (52.01)	10 | 19	455.61 (186.54)

Figure 2: Comparing τ_{LP} and $\tau^1_{\text{tighter,cs}}$ bounds for BNN_1 and different $\delta_{||.||_\infty}$. Each subplot x-axis represents image indices sorted in the descending order of $\tau^1_{\text{tighter,cs}}$ values. The upper bound ub is obtained by random sampling. The relative improvement over LP is estimated through $\frac{\tau^1_{\text{tighter,cs}} - \tau_{\text{LP}}}{\text{ub} - \tau_{\text{LP}}}$. On average, $\tau^1_{\text{tighter,cs}}$ bounds are $21, 33, 46$ and 53 percent more accurate, respectively.



SDP methods provide trusted bounds: The discontinuity of the $\text{sign}(\cdot)$ activation function can exacerbate floating-point errors, leading to significant numerical inaccuracies in the Mixed Integer Linear Programming (MILP) solving process. For instance, if a node's value after linear transformations is $-1.95 \cdot 10^{-14}$, assigning the value of -1 to this node is unreliable, as floating-point errors could easily flip the sign. Such incorrect sign values can critically affect the feasibility of the solution and the overall bound. To address this, we introduce a margin of 10^{-7} for sign determination, which we call Soft-MILP. In contrast, for the SDP-based method, we can easily enclose the errors with interval arithmetic as in Magron et al. (2015), to obtain a rigorously valid lower bound.

5.2 ROBUSTNESS AGAINST $||.||_2$ ATTACKS

We replace the constraint (13c) with $\boldsymbol{g_B}$ such that $\boldsymbol{B} = \mathbb{B}_{||.||_2}(\bar{\boldsymbol{x}}, \varepsilon) \cap [-1, 1]^{784}$. The resulting Mixed Integer Non-Linear Programming (MINP) problem, also solved using Gurobi, and its optimal value are referred to as $\tau_{\text{Soft-MINP}}$.

Table 2: Performance comparison for $||.||_2$-verification, where $\delta_{||.||_2} = 255\varepsilon$.

$\delta_{\vert\vert.\vert\vert_2}$	$\tau^1_{\text{tighter,cs}}$ cert.	$t\ (s)$	$\tau_{\text{Soft-MINP}}$ cert.	$t\ (s)$
$\text{BNN}_1 : [784, 500, 500, 10], w_s = 34.34\%$				
10	70	5.23 (5.02)	3 | 93	33.35 (5.75)
20	36	19.54 (15.20)	4 | 30	447.11 (278.38)
30	13	34.24 (18.08)	4 | 6	556.07 (467.16)
$\text{BNN}_2 : [784, 500, 500, 10], w_s = 19.07\%$				
5	81	8.57 (8.50)	2 | 96	3.31 (1.18)
10	46	19.00 (15.44)	3 | 58	272.92 (106.25)
15	27	63.21 (36.73)	4 | 23	475.78 (293.96)

As displayed in Table 2, when the perturbation region is small, the exact MINP method can verify more instances. However, for more severe attacks, our method certifies robustness for almost the same number of instances, but 11.4 times faster on average (for more details, refer to Appendix B).

6 CONCLUSION AND FUTURE WORKS

In this work, we studied SDP relaxations associated with polynomial optimization problems to verify the properties of BNNs with continuous input space. We demonstrated the ability of our method to verify robustness against both $\|.\|_\infty$ and $\|.\|_2$ attacks. The proposed method efficiently exploits the inherent sparse structure of a given BNN and generally provides much less conservative bounds than the LP-based method. Moreover, its running time does not scale exponentially with either the size of the network or the perturbation region.

Our method relies on the interior-point SDP solvers, and thus inherits their limitations. One direction for future improvements could be based on the embedding of automatic differentiation to either improve existing interior-point SDP solvers or create new ones, like in Dathathri et al. (2020).

Moreover, our experimental results suggest that one could replace LP relaxations with SDP relaxations in branch-and-bound/branch-and-cut algorithms, such as the ones implemented in Gurobi. Based on our experiments, an alternative SDP-based relaxation embedded within general-purpose MILP/MINP solvers would significantly accelerate the exact verification process, especially for larger input perturbation regions.

Better bounding at comparable time cost could significantly improve the overall performance of such solvers, which is another exciting topic of future research.

REFERENCES

Guy Amir, Haoze Wu, Clark Barrett, and Guy Katz. An SMT-based approach for verifying binarized neural networks. In *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 203–222. Springer International Publishing, 2021. ISBN 978-3-030-72013-1.

Erling D Andersen and Knud D Andersen. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*, pp. 197–232. Springer, 2000.

Xue Anton, Lars Lindemann, and Rajeev Alur. Chordal sparsity for SDP-based neural network verification. *Automatica*, 161:111487, 2024. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2023.111487. URL `https://www.sciencedirect.com/science/article/pii/S0005109823006568`.

V. Antun, N.M. Gottschling, A.C Hansen, and B. Adcok. Deep learning in scientific computing: Understanding the instability mystery. *SIAM News*, 54(2), 2021. March 01.

Johannes Aspman, Georgios Korpas, and Jakub Marecek. Taming binarized neural networks and mixed-integer programs. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 10935–10943. AAAI Press, 2024. doi: 10.1609/AAAI.V38I10.28968. URL `https://doi.org/10.1609/aaai.v38i10.28968`.

Stanley Bak and Parasara Sridhar Duggirala. Simulation-equivalent reachability of large linear systems with inputs. In *Computer Aided Verification*, pp. 401–420, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63387-9.

Teodora Baluta, Shiqi Shen, Shweta Shinde, Kuldeep S. Meel, and Prateek Saxena. Quantitative verification of neural networks and its security applications. In *CCS '19*, pp. 1249–1264, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3354245. URL `https://doi.org/10.1145/3319535.3354245`.

Ben Batten, Panagiotis Kouvaros, Alessio Lomuscio, and Yang Zheng. Efficient neural network verification via layer-based semidefinite relaxations and linear cuts. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2184–2190. International Joint Conferences on Artificial Intelligence Organization, 2021. doi: 10.24963/ijcai.2021/301. URL `https://doi.org/10.24963/ijcai.2021/301`.

Tong Chen, J.-B Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic Optimization for Bounding Lipschitz Constants of ReLU Networks. *Proceeding of Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Tong Chen, J.-B Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic Representation of Monotone Deep Equilibrium Models and Applications to Certification. *Proceeding of Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2021.

Chih-Hong Cheng, Georg Nührenberg, Chung-Hao Huang, and Harald Ruess. Verification of binarized neural networks via inter-neuron factoring. In *Verified Software. Theories, Tools, and Experiments*, pp. 279–290. Springer International Publishing, 2018. ISBN 978-3-030-03592-1.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/3e15cc11f979ed25912dff5b0669f2cd-Paper.pdf`.

Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang, and Pushmeet Kohli. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5318–5331. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/397d6b4c83c91021fe928a8c4220386b-Paper.pdf`.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

Mahyar Fazlyab, Manfred Morari, and George Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, PP:1–1, 12 2020. doi: 10.1109/TAC.2020.3046193.

Angus Galloway, Graham W. Taylor, and Medhat Moussa. Attacking binarized neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HkTEFfZRb`.

Lukas Geiger and Plumerai Team. Larq: An open-source library for training binarized neural networks. *Journal of Open Source Software*, 5(45):1746, 2020. doi: 10.21105/joss.01746. URL `https://doi.org/10.21105/joss.01746`.

Mirco Giacobbe, Thomas A. Henzinger, and Mathias Lechner. How many bits does it take to quantize your neural network? In *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 79–97, Cham, 2020. Springer International Publishing. ISBN 978-3-030-45237-7.

Gurobi Optimization, LLC. Dealing with big-M constraints. `https://www.gurobi.com/documentation/current/refman/dealing_with_big_m_constra.html`. Accessed: 17 May 2024.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL `https://www.gurobi.com`.

Shaoning Han and Andrés Goméz. Single-neuron convexification for binarized neural networks. *Optimization Online*, 2021. URL `https://optimization-online.org/2021/05/8419/`.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper_files/paper/2016/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf`.

Mykhailo Ivashchenko, Sung Woo Choi, Luan Viet Nguyen, and Hoang-Dung Tran. Verifying binary neural networks on continuous input space using star reachability. In *2023 IEEE/ACM 11th International Conference on Formal Methods in Software Engineering (FormaliSE)*, pp. 7–17, 2023. doi: 10.1109/FormaliSE58978.2023.00009.

Kai Jia and Martin Rinard. Efficient exact verification of binarized neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020a. Curran Associates Inc. ISBN 9781713829546.

Kai Jia and Martin C. Rinard. Exploiting verified neural networks via floating point numerical error. *CoRR*, abs/2003.03021, 2020b. URL `https://arxiv.org/abs/2003.03021`.

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*, pp. 97–117, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63387-9.

Elias B. Khalil, Amrita Gupta, and Bistra Dilkina. Combinatorial attacks on binarized neural networks. In *International Conference on Learning Representations (ICLR)*, 2019. URL `https://arxiv.org/abs/1810.03538`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `https://api.semanticscholar.org/CorpusID:6628106`.

Jianglin Lan, Yang Zheng, and Alessio Lomuscio. Tight neural network verification via semidefinite relaxations and linear reformulations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7272–7280, 2022.

Jean-Bernard Lasserre. Global Optimization with Polynomials and the Problem of Moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001. doi: 10.1137/S1052623400366802. URL `https://doi.org/10.1137/S1052623400366802`.

Jean-Bernard Lasserre. Convergent SDP-Relaxations in Polynomial Optimization with Sparsity. *Siam Journal on Optimization*, 17:263–272, 01 2006. doi: 10.1137/05064504X.

Jean-Bernard Lasserre. *An Introduction to Polynomial and Semi-Algebraic Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2015. ISBN 9781316240397. URL `https://books.google.com.sg/books?id=pZXsDAAAQBAJ`.

Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of Neural Networks via sparse polynomial optimization. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rJe4_xSFDB`.

Christopher Lazarus and Mykel J. Kochenderfer. A mixed integer programming approach for verifying properties of binarized neural networks. *International Joint Conference on Artificial Intelligence (IJCAI), AI Safety Workshop*, 2022. doi: 10.48550/arXiv.2203.07078. URL `https://doi.org/10.48550/arXiv.2203.07078`.

Christopher Lazarus, Mykel J. Kochenderfer, Stephen Boyd, and Mert Pilanci. Trustworthy machine learning by efficiently verifying compressed models. *https://purl.stanford.edu/rs658gj7336.*, May, 2022.

Victor Magron and Jie Wang. TSSOS: a Julia library to exploit sparsity for large-scale polynomial optimization. *ArXiv*, abs/2103.00915, 2021. URL `https://api.semanticscholar.org/CorpusID:232076082`.

Victor Magron and Jie Wang. *Sparse Polynomial Optimization: Theory and Practice*. Series on Optimization and Its Applications. World Scientific (Europe), 2023. doi: 10.1142/q0382. URL `https://www.worldscientific.com/doi/abs/10.1142/q0382`.

Victor Magron, Xavier Allamigeon, Stéphane Gaubert, and Benjamin Werner. Formal proofs for Nonlinear Optimization. *Journal of Formalized Reasoning*, 8(1):1–24, 2015.

Nina Narodytska. Formal analysis of deep binarized neural networks. In *IJCAI*, pp. 5692–5696, 2018.

Nina Narodytska, Shiva Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. Verifying properties of binarized deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Nina Narodytska, Aditya Shrotri, Kuldeep S. Meel, Alexey Ignatiev, and Joao Marques-Silva. Assessing heuristic machine learning explanations with model counting. In *Theory and Applications of Satisfiability Testing – SAT 2019*, pp. 267–278, Cham, 2019. Springer International Publishing. ISBN 978-3-030-24258-9.

Nina Narodytska, Hongce Zhang, Aarti Gupta, and Toby Walsh. In search for a SAT-friendly binarized neural network architecture. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SJx-j64FDr`.

Matthew Newton and Antonis Papachristodoulou. Sparse polynomial optimisation for neural network verification. *Automatica*, 157:111233, 2023. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2023.111233. URL `https://www.sciencedirect.com/science/article/pii/S0005109823003941`.

Jiawang Nie. Optimality conditions and finite convergence of Lasserre's hierarchy. *Mathematical programming*, 146:97–121, 2014.

Chongli Qin, Krishnamurthy (Dj) Dvijotham, Brendan O'Donoghue, Rudy Bunel, Robert Stanforth, Sven Gowal, Jonathan Uesato, Grzegorz Swirszcz, and Pushmeet Kohli. Verification of non-linear specifications for neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyeFAsRctQ`.

Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, 2020.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 10900–10910, 2018. URL `https://proceedings.neurips.cc/paper/2018/hash/29c0605a3bab4229e46723f89cf59d83-Abstract.html`.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *Computer Vision – ECCV 2016*, pp. 525–542, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/246a3c5544feb054f3ea718f61adfa16-Paper.pdf`.

Weijia Shi, Andy Shih, Adnan Darwiche, and Arthur Choi. On tractable representations of binary neural networks. In *International Conference on Principles of Knowledge Representation and Reasoning*, 2020. URL `https://api.semanticscholar.org/CorpusID:214802814`.

Andy Shih, Adnan Darwiche, and Arthur Choi. Verifying binarized neural networks by angluin-style learning. In *Theory and Applications of Satisfiability Testing – SAT 2019*, pp. 354–370, Cham, 2019. Springer International Publishing. ISBN 978-3-030-24258-9.

Siyang Sun, Yingjie Yin, Xingang Wang, De Xu, Wenqi Wu, and Qingyi Gu. Fast object detection based on binary deep convolution neural networks. *CAAI Transactions on Intelligence Technology*, 3(4):191–197, 2018. doi: https://doi.org/10.1049/trit.2018.1026. URL `https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/trit.2018.1026`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6199`.

Dung Tran, Diego Manzanas Lopez, Patrick Musau, Xiaodong Yang, Viet Luan, Luan Nguyen, Weiming Xiang, and Taylor Johnson. Star-based reachability analysis of deep neural networks. In *23rd International Symposium on Formal Methods*, pp. 670–686, 2019.

Hoang-Dung Tran, Stanley Bak, Weiming Xiang, and Taylor T. Johnson. Verification of Deep Convolutional Neural Networks Using ImageStars. In *Computer Aided Verification*, pp. 18–42, Cham, 2020. Springer International Publishing. ISBN 978-3-030-53288-8.

Lieven Vandenberghe, Martin S Andersen, et al. Chordal graphs and semidefinite optimization. *Foundations and Trends® in Optimization*, 1(4):241–433, 2015.

Lorenzo Vorabbi, Davide Maltoni, and Stefano Santi. On-device learning with binary neural networks. In *Image Analysis and Processing - ICIAP 2023 Workshops*, pp. 39–50, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-51023-6.

Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Muramatsu. Sums of Squares and Semidefinite Program Relaxations for Polynomial Optimization Problems with Structured Sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006. doi: 10.1137/050623802. URL `https://doi.org/10.1137/050623802`.

Jie Wang, Victor Magron, and Jean-Bernard Lasserre. TSSOS: A Moment-SOS Hierarchy That Exploits Term Sparsity. *SIAM Journal on Optimization*, 31(1):30–58, 2021. doi: 10.1137/19M1307871. URL `https://doi.org/10.1137/19M1307871`.

Xu Xiang, Yanmin Qian, and Kai Yu. Binary deep neural networks for speech recognition. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pp. 533–537. ISCA, 2017. doi: 10.21437/INTERSPEECH.2017-1343. URL `https://doi.org/10.21437/Interspeech.2017-1343`.

Chunyu Yuan and Sos S. Agaian. A comprehensive review of binary neural network. *Artificial Intelligence Review*, pp. 1–65, 2021. URL `https://api.semanticscholar.org/CorpusID:238743860`.

Yedi Zhang, Zhe Zhao, Guangke Chen, Fu Song, and Taolue Chen. BDD4BNN: A BDD-based quantitative analysis framework for binarized neural networks. In *Computer Aided Verification*, pp. 175–200, Cham, 2021. Springer International Publishing. ISBN 978-3-030-81685-8.

A THEORETICAL DISCUSSION

A.1 DUAL SIDE OF POLYNOMIAL OPTIMIZATION RELAXATIONS - MOMENT HIERARCHIES

We have discussed computing a global minimum of a multivariate polynomial using SDP relaxations obtained by interpreting the requirement for polynomials to be positive on sets defined with finitely many (in)equalities.

It is also possible to derive the corresponding dual relaxations. Notice that a polynomial f in the variable $\boldsymbol{x} = (x_1, \ldots, x_n)$ can also be written as $f = \sum_{\alpha \in \mathcal{A}} f_\alpha \boldsymbol{x}^\alpha$ with $\mathcal{A} \subset \mathbb{N}^n$ and $f_\alpha \in \mathbb{R}$, where $\boldsymbol{x}^\alpha = x_1^{\alpha_1} \dots x_n^{\alpha_n}$. Then the *moment hierarchy* Lasserre (2001) for the POP introduced in Section 2.3 corresponds to:

$$\tau^d_{\text{mom}} := \begin{cases} \inf\limits_{\boldsymbol{y}} L_{\boldsymbol{y}}(f) & \\ \text{s.t.} \quad \boldsymbol{M}_d(\boldsymbol{y}) \succeq 0, & (21\text{a}) \\ \qquad \boldsymbol{M}_{d-1}(\boldsymbol{g}_i(\cdot)_j \boldsymbol{y}) \succeq 0, i \in [\![1, L]\!], j \in [\![1, n_i]\!], & (21\text{b}) \\ \qquad \boldsymbol{M}_{d-1}(\boldsymbol{h}_i(\cdot)_j \boldsymbol{y}) = 0, i \in [\![1, L]\!], j \in [\![1, n_i]\!], & (21\text{c}) \\ \qquad \boldsymbol{M}_{d-1}(\boldsymbol{g}_{\boldsymbol{B}}(\cdot)_k \boldsymbol{y}) \succeq 0, k \in [\![1, n_{\boldsymbol{B}}]\!], & (21\text{d}) \\ \qquad y_0 = 1, & (21\text{e}) \end{cases}$$

where $\boldsymbol{y} = (y_\alpha)_\alpha$ is a sequence indexed by $\alpha \in \mathbb{N}^n$ and $L_{\boldsymbol{y}}$ the linear functional defined by

$$f \mapsto L_{\boldsymbol{y}}(f) := \sum_\alpha f_\alpha y_\alpha. \tag{22}$$

For $d \in \mathbb{N}$, $\boldsymbol{M}_d(\boldsymbol{y})$ denotes the *moment matrix* of order d associated with $\boldsymbol{y}$ and defined as follows

$$\boldsymbol{M}_d(\boldsymbol{y})(\beta, \gamma) := L_{\boldsymbol{y}}(\boldsymbol{x}^\beta \boldsymbol{x}^\gamma) = y_{\beta+\gamma}, \quad \forall \beta, \gamma \in \mathbb{N}^n_d. \tag{23}$$

Similarly, for $g = \sum_\alpha g_\alpha \boldsymbol{x}^\alpha \in \mathbb{R}[\boldsymbol{x}]$, $\boldsymbol{M}_d(g\boldsymbol{y})$ denotes the *localizing matrix* of order d associated with g and $\boldsymbol{y}$, defined as follows

$$\boldsymbol{M}_d(g\boldsymbol{y})(\beta, \gamma) := L_{\boldsymbol{y}}(g \boldsymbol{x}^\beta \boldsymbol{x}^\gamma) = \sum_\alpha g_\alpha y_{\alpha+\beta+\gamma}, \quad \forall \beta, \gamma \in \mathbb{N}^n_d. \tag{24}$$

When the correlative sparsity is present, analogous hierarchies of sparse moment relaxations can be derived as well (Lasserre, 2006; Magron & Wang, 2023). Such reasoning was adopted in the following proof of Theorem 3.1.

Proof of Theorem 3.1. Let $j \in [\![1, n_L]\!]$ and $c_j^L = \mathrm{nv}(\boldsymbol{W}^{[L]})_j + \boldsymbol{b}_j^{[L]}$ and define $f(\boldsymbol{x}^0, \boldsymbol{x}^1, \dots, \boldsymbol{x}^L) := \boldsymbol{x}_j^L - \left(\frac{2}{c_j^L}\left(\left\langle \boldsymbol{W}^{[L]}_{(j:,)}, \boldsymbol{x}^{L-1}\right\rangle + \boldsymbol{b}_j^{[L]}\right) - 1\right)$. Then,

$$\boldsymbol{M} = \begin{array}{c} \\ 1 \\ \boldsymbol{x}^{L-1} \\ \boldsymbol{x}_j^L \end{array} \begin{array}{c} \begin{array}{ccc} 1 & \boldsymbol{x}^{L-1} & \boldsymbol{x}_j^L \end{array} \\ \left[\begin{array}{ccc} 1 & \left(\boldsymbol{W}^{[L]}_{(j:,)}\right)^\intercal & 0 \\ \boldsymbol{W}^{[L]}_{(j:,)} & \boldsymbol{W}^{[L]}_{(j:,)} \left(\boldsymbol{W}^{[L]}_{(j:,)}\right)^\intercal & \mathbf{0} \\ 0 & \mathbf{0} & 1 \end{array} \right] \end{array} \tag{25}$$

represents a feasible moment matrix for the sparse first-order moment relaxation of (8), yielding the value of the objective function equal to -1 (because $||\boldsymbol{W}^{[L]}_{(j:,)}||_2^2 = \mathrm{nv}(\boldsymbol{W}^{[L]})_j$), implying that $\tau^1 \leq -1$. However, since $f(\boldsymbol{x}^0, \boldsymbol{x}^1, \dots, \boldsymbol{x}^L) \geq 0$ is one of the constraints in (14), we deduce that $\tau_{\text{LP}} \geq 0$. $\square$

As a direct consequence of Theorem 3.1, we have the following corollary:

Corollary A.1.1. *Consider an arbitrary BNN with depth $L \geq 2$, and let $j \in [\![1, n_L]\!]$. Then,*

$$\boldsymbol{g}^1_{L,\mathit{LIN}}(\cdot)_j \notin \mathcal{Q}_1(\{\boldsymbol{g}_L(\cdot)_j\}), \tag{26}$$

thus the set $\left\{\boldsymbol{g}^0_{i,\mathit{LIN}}, \boldsymbol{g}^1_{i,\mathit{LIN}}, \boldsymbol{g}^2_{i,\mathit{LIN}}, i \in [\![1, L]\!]\right\}$ is not included in $\mathcal{Q}_1(\{\boldsymbol{g}_1, \dots, \boldsymbol{g}_L, \boldsymbol{g}_{\boldsymbol{B}}\})$.

A.2 BOUNDS FROM THE SECOND-ORDER RELAXATION

Theorem A.2.1. *If $(\boldsymbol{x}^0, \boldsymbol{x}^1, \dots, \boldsymbol{x}^L) \mapsto f(\boldsymbol{x}^0, \boldsymbol{x}^1, \dots, \boldsymbol{x}^L)$ is affine, we have $\tau \geq \tau^2 \geq \tau_{\text{LP}}$. Consequently, any dual-feasible solution of (14) yields a valid SOS decomposition for the second-order SDP relaxation of (8).*

Proof of Theorem A.2.1. We proceed by analyzing the feasible sets of each problem. Firstly, both problems share the same input perturbation region given by (13c). For the remaining constraints, let

us consider three different cases:
(i) Single node bounds: Let $i \in [\![1, L]\!]$ and $j \in [\![1, n_i]\!]$. Then, the following equations on $\mathbb{R}[\boldsymbol{x}^i]$ hold true:

$$\boldsymbol{g}^0_{i,\mathrm{LIN}}(\boldsymbol{x}^i)_j = \frac{1}{2}(1-\boldsymbol{x}^i_j)^2 + \frac{1}{2}\boldsymbol{h}_i(\boldsymbol{x}^i)_j, \text{ and } \boldsymbol{g}^0_{i,\mathrm{LIN}}(\boldsymbol{x}^i)_{n_i+j} = \frac{1}{2}(\boldsymbol{x}^i_j+1)^2 + \frac{1}{2}\boldsymbol{h}_i(\boldsymbol{x}^i)_j. \tag{27}$$

Thus $\{\boldsymbol{g}^0_{i,\mathrm{LIN}}, i \in [\![1, L]\!]\} \subset \mathcal{Q}_2(\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_L, \boldsymbol{g}_{\boldsymbol{B}}\})$.
(ii) Interaction between two adjacent hidden layers: Fix $i \in [\![2, L]\!]$ and $j \in [\![1, n_i]\!]$. Let us define $c^i_{\pm,j} := \mathrm{nv}(\boldsymbol{W}^{[i]})_j \pm \boldsymbol{b}^{[i]}_j > 0$. If we apply the substitution rules $\boldsymbol{x}^{i-1} \odot \boldsymbol{x}^{i-1} = \mathbf{1}$ and $(\boldsymbol{x}^i_j)^2 = 1$, then the following equations always hold true on $\mathbb{R}[\boldsymbol{x}^{i-1}, \boldsymbol{x}^i_j]$:

$$\boldsymbol{g}^1_{i,\mathrm{LIN}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j = \frac{(1-\boldsymbol{x}^i_j)^2}{2c^i_{+,j}}\boldsymbol{g}_i(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j + \frac{(1+\boldsymbol{x}^i_j)^2}{4c^i_{+,j}}\sum_{k=1}^{n_{i-1}}\left(1-\boldsymbol{W}^{[i]}_{j,k}\boldsymbol{x}^{i-1}_k\right)^2, \tag{28}$$

$$\boldsymbol{g}^2_{i,\mathrm{LIN}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j = \frac{(1+\boldsymbol{x}^i_j)^2}{2c^i_{-,j}}\boldsymbol{g}_i(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j + \frac{(1-\boldsymbol{x}^i_j)^2}{4c^i_{-,j}}\sum_{k=1}^{n_{i-1}}\left(1+\boldsymbol{W}^{[i]}_{j,k}\boldsymbol{x}^{i-1}_k\right)^2. \tag{29}$$

Equations (28) and (29) certify that $\{\boldsymbol{g}^1_{i,\mathrm{LIN}}, \boldsymbol{g}^2_{i,\mathrm{LIN}}, i \in [\![1, L]\!]\} \subset \mathcal{Q}_2(\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_L, \boldsymbol{g}_{\boldsymbol{B}}\})$.

(iii) Interaction between the input layer and the first hidden layer: For any $j \in [\![1, n_1]\!]$, let us define $c^1_{\pm,j} := \mathrm{nv}(\boldsymbol{W}^{[1]})_j \pm \boldsymbol{b}^{[1]}_j > 0$. It follows that

$$\boldsymbol{g}^1_{1,\mathrm{LIN}}(\boldsymbol{x}^1, \boldsymbol{x}^0)_j = \frac{(1-\boldsymbol{x}^1_j)^2}{2c^1_{+,j}}\boldsymbol{g}_1(\boldsymbol{x}^0, \boldsymbol{x}^1)_j + \frac{(1+\boldsymbol{x}^1_j)^2}{2c^1_{+,j}}\sum_{k=1}^{n_0}\left(\left|\boldsymbol{W}^{[1]}_{j,k}\right| - \boldsymbol{W}^{[1]}_{j,k}\boldsymbol{x}^0_k\right), \tag{30}$$

$$\boldsymbol{g}^2_{1,\mathrm{LIN}}(\boldsymbol{x}^1, \boldsymbol{x}^0)_j = \frac{(1+\boldsymbol{x}^1_j)^2}{2c^1_{-,j}}\boldsymbol{g}_1(\boldsymbol{x}^0, \boldsymbol{x}^1)_j + \frac{(1-\boldsymbol{x}^1_j)^2}{2c^1_{-,j}}\sum_{k=1}^{n_0}\left(\left|\boldsymbol{W}^{[1]}_{j,k}\right| + \boldsymbol{W}^{[1]}_{j,k}\boldsymbol{x}^0_k\right), \tag{31}$$

provide the $\mathcal{Q}_2\left(\{\boldsymbol{g}_1(\cdot)_j\}\right)$-based representations for $\boldsymbol{g}^1_{1,\mathrm{LIN}}(\cdot)_j$ and $\boldsymbol{g}^2_{1,\mathrm{LIN}}(\cdot)_j$, valid after substituting $(\boldsymbol{x}^1_j)^2 = 1$. □

Example A.2.1 (Illustration of Theorem A.2.1)**.** *This theorem indicates that any dual-feasible solution of (14) provides a feasible solution for the second-order SOS relaxation (10) of QCQP (8). For instance, let us consider the toy BNN from Example (2.1). Recall that we aim to minimize the linear function* $f^{\mathrm{adv}}_1(\boldsymbol{x}) = 2\boldsymbol{x}^2_2 - 1$. *Since the LP relaxation problem has following constraints:*

$$-\frac{4\,\boldsymbol{x}^1_2}{3} + \frac{4\,\boldsymbol{x}^1_1}{3} + \boldsymbol{x}^2_2 \geq -\frac{5}{3},\ -\boldsymbol{x}^1_1 \geq -1,\ \boldsymbol{x}^1_2 \geq -1,\ \boldsymbol{x}^2_2 \geq -1, \tag{32}$$

the expression

$$\begin{aligned}\left(2\boldsymbol{x}^2_2 - 1\right) + 4 = \frac{3}{10}\left(-\frac{4\,\boldsymbol{x}^1_2}{3} + \frac{4\,\boldsymbol{x}^1_1}{3} + \boldsymbol{x}^2_2 + \frac{5}{3}\right) + \frac{2}{5}\left(-\boldsymbol{x}^1_1 + 1\right)\\ + \frac{2}{5}\left(-\boldsymbol{x}^1_2 + 1\right) + \frac{17}{10}\left(\boldsymbol{x}^2_2 + 1\right) \geq 0\end{aligned} \tag{33}$$

gives one feasible dual solution of the LP relaxation problem. From this dual solution, we can recover the following SOS decomposition, certifying that $f^{\mathrm{adv}}_1 \geq -3$*:*

$$\begin{aligned}\left(2\boldsymbol{x}^2_2 - 1\right) + 4 = \frac{3}{10}\left(\frac{2}{3}(\boldsymbol{x}^2_2 - 1)\left(\boldsymbol{x}^1_2 - \boldsymbol{x}^1_1 - \frac{1}{2}\right) + \frac{2}{3}(1+\boldsymbol{x}^2_2)(2 - \boldsymbol{x}^1_2 + \boldsymbol{x}^1_1)\right)\\ + \frac{2}{5}\cdot\frac{1}{2}\left(1-\boldsymbol{x}^1_1\right)^2 + \frac{2}{5}\cdot\frac{1}{2}\left(1+\boldsymbol{x}^1_2\right)^2 + \frac{17}{10}\cdot\frac{1}{2}\left(1+\boldsymbol{x}^2_2\right)^2.\end{aligned} \tag{34}$$

A.3 IMPORTANCE OF TAUTOLOGIES

Proof of Theorem 4.1. Since all the variables are bounded, we have $\tau^1_{\text{tighter,cs}} = \tau^1_{\text{tighter}}$. Moreover, the inequality $\tau^1_{\text{tighter,cs}} \geq \tau^1$ is a consequence of the identity

$$\boldsymbol{g}_i(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) = \frac{1}{2}\tilde{\boldsymbol{g}}_i^1(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) + \frac{1}{2}\tilde{\boldsymbol{g}}_i^2(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}),\ i \in [\![2, L]\!]. \tag{35}$$

In order to prove that $\tau^1_{\text{tighter,cs}} \geq \tau_{\text{LP}}$ holds, we show that all linear functions $\boldsymbol{g}^1_{i,\text{LIN}}, \boldsymbol{g}^2_{i,\text{LIN}}, i \in [\![1, L]\!]$, involved in the constraints (13a) and (13b) belong to $\mathcal{Q}_1(\{\tilde{\boldsymbol{g}}_i^1, \tilde{\boldsymbol{g}}_i^2, \tilde{\boldsymbol{g}}_i^{\text{t1}}, \tilde{\boldsymbol{g}}_i^{\text{t2}}, i \in [\![1, L]\!]\})$. We consider two different scenarios:

(i) Let $i \in [\![2, L]\!]$, $j \in [\![1, n_i]\!]$ and $c^i_{\pm,j} := \text{nv}(\boldsymbol{W}^{[i]})_j \pm \boldsymbol{b}^{[i]}_j > 0$. When we apply the substitution rules $\boldsymbol{x}^{i-1} \odot \boldsymbol{x}^{i-1} = \mathbf{1}$ and $(\boldsymbol{x}^i_j)^2 = 1$, then the following equations always hold true on $\mathbb{R}[\boldsymbol{x}^{i-1}, \boldsymbol{x}^i_j]$:

$$\boldsymbol{g}^1_{i,\text{LIN}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j = \frac{1}{c^i_{+,j}}\tilde{\boldsymbol{g}}_i^2(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j + \frac{1}{c^i_{+,j}}\tilde{\boldsymbol{g}}_i^{\text{t1}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j, \tag{36}$$

$$\boldsymbol{g}^2_{i,\text{LIN}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j = \frac{1}{c^i_{-,j}}\tilde{\boldsymbol{g}}_i^1(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j + \frac{1}{c^i_{-,j}}\tilde{\boldsymbol{g}}_i^{\text{t2}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1})_j. \tag{37}$$

(ii) Secondly, for any $j \in [\![1, n_1]\!]$, and $c^1_{\pm,j} = \text{nv}(\boldsymbol{W}^{[1]})_j \pm \boldsymbol{b}^{[1]}_j > 0$, substituting $(\boldsymbol{x}^1_j)^2 = 1$ yields the following equalities on $\mathbb{R}[\boldsymbol{x}^0, \boldsymbol{x}^1_j]$:

$$\boldsymbol{g}^1_{1,\text{LIN}}(\boldsymbol{x}^1, \boldsymbol{x}^0)_j = \frac{1}{c^1_{+,j}}\boldsymbol{g}_2(\boldsymbol{x}^1, \boldsymbol{x}^0)_j + \frac{1}{c^1_{+,j}}\tilde{\boldsymbol{g}}_1^{\text{t2}}(\boldsymbol{x}^1, \boldsymbol{x}^0)_j, \tag{38}$$

$$\boldsymbol{g}^2_{1,\text{LIN}}(\boldsymbol{x}^1, \boldsymbol{x}^0)_j = \frac{1}{c^1_{+,j}}\boldsymbol{g}_1(\boldsymbol{x}^1, \boldsymbol{x}^0)_j + \frac{1}{c^1_{+,j}}\tilde{\boldsymbol{g}}_1^{\text{t1}}(\boldsymbol{x}^1, \boldsymbol{x}^0)_j, \tag{39}$$

certifying that $\boldsymbol{g}^1_{1,\text{LIN}}(\boldsymbol{x}^0, \boldsymbol{x}^1)_j$ and $\boldsymbol{g}^2_{1,\text{LIN}}(\boldsymbol{x}^0, \boldsymbol{x}^1)_j$ belong to $\mathcal{Q}_1(\{\tilde{\boldsymbol{g}}_i^1, \tilde{\boldsymbol{g}}_i^2, \tilde{\boldsymbol{g}}_i^{\text{t1}}, \tilde{\boldsymbol{g}}_i^{\text{t2}}, i \in [\![1, L]\!]\})$.

Finally, Theorem 3.1 implies that there exists an affine f such that $\tau^1_{\text{tighter,cs}} \geq \tau_{\text{LP}} > \tau^1$.

$\square$

Remark A.3.1. *We insist that tautologies (18a) and (18b) are both important and necessary. If we consider the optimization problem without tautologies, where only (17a) and (17b) are used to replace (7b), then we have the following strict containment relationship:*

$$\mathcal{Q}_1(\{\boldsymbol{g}_{\boldsymbol{B}}, \tilde{\boldsymbol{g}}_i^1, \tilde{\boldsymbol{g}}_i^2, i \in [\![1, L]\!]\}) \subsetneqq \mathcal{Q}_1(\{\boldsymbol{g}_{\boldsymbol{B}}, \tilde{\boldsymbol{g}}_i^1, \tilde{\boldsymbol{g}}_i^2, \tilde{\boldsymbol{g}}_i^{t1}, \tilde{\boldsymbol{g}}_i^{t2}, i \in [\![1, L]\!]\}). \tag{40}$$

Indeed, following the same reasoning as in the proof of Theorem 3.1, we obtain that

$$\boldsymbol{M} = \begin{array}{c|ccc|} & 1 & \boldsymbol{x}^{L-1} & \boldsymbol{x}^L_j \\ 1 & 1 & a\left(\boldsymbol{W}^{[L]}_{(j:,)}\right)^\intercal & 0 \\ \boldsymbol{x}^{L-1} & a\boldsymbol{W}^{[L]}_{(j:,)} & \boldsymbol{W}^{[L]}_{(j:,)}\left(\boldsymbol{W}^{[L]}_{(j:,)}\right)^\intercal & t\boldsymbol{W}^{[L]}_{(j:,)} \\ \boldsymbol{x}^L_j & 0 & t\left(\boldsymbol{W}^{[L]}_{(j:,)}\right)^\intercal & 1 \end{array} \tag{41}$$

with $a = \frac{1}{2}\sqrt{2 - \frac{\left(\boldsymbol{b}^{[L]}_j\right)^2}{\left(\text{nv}(\boldsymbol{W}^{[L]})_j\right)^2}} - \frac{\boldsymbol{b}^{[L]}_j}{2\,\text{nv}(\boldsymbol{W}^{[L]})_j}$ *and* $t = \sqrt{1-a^2}$ *is a feasible moment matrix for which the value of the objective function is* $-\frac{\text{nv}(\boldsymbol{W}^{[L]})_j}{\text{nv}(\boldsymbol{W}^{[L]})_j + \boldsymbol{b}^{[L]}_j}\left(\sqrt{2 - \frac{\left(\boldsymbol{b}^{[L]}_j\right)^2}{\left(\text{nv}(\boldsymbol{W}^{[L]})_j\right)^2}} - 1\right) < 0$. *To prove that* $\boldsymbol{M}$ *is positive semidefinite, we observe that* M *has a* n_{L-1}*-dimensional null space corresponding to the direct sum of the sets* A *and* B*, where*

$$\begin{aligned} A &:= \left\{(0, \boldsymbol{v}, 0)^\intercal \in \mathbb{R}^{n_{L-1}+2} \mid \left\langle \boldsymbol{W}^{[L]}_{(j:,)}, \boldsymbol{v} \right\rangle = 0\right\}, \\ B &:= \left\{\alpha\left(-a\,\text{nv}(\boldsymbol{W}^{[L]})_j, \boldsymbol{W}^{[L]}_{(j:,)}, -t\,\text{nv}(\boldsymbol{W}^{[L]})_j\right)^\intercal \mid \alpha \in \mathbb{R}\right\}. \end{aligned} \tag{42}$$

Thus, $\boldsymbol{M}$ has n_{L-1} eigenvalues equal to zero, and one eigenvalue equal to 1 corresponding to the eigenvector $(-t, \mathbf{0}, a)^\intercal$. Since the trace of $\boldsymbol{M}$ is $\mathrm{nv}(\boldsymbol{W}^{[L]})_j + 2$, we know that the maximal eigenvalue of $\boldsymbol{M}$ is $\mathrm{nv}(\boldsymbol{W}^{[L]})_j + 1$.

Details from the previous discussion can be summarized in the following corollary:

Corollary A.3.1. *Let $i \in [\![2, L]\!]$ and $j \in [\![1, n_i]\!]$. If the substitution rule $(\boldsymbol{x}^i_j)^2 = 1$ is applied on the space $\mathbb{R}[\boldsymbol{x}^{i-1}, \boldsymbol{x}^i_j]$,*

$$\boldsymbol{g}^1_{i,LIN}(\cdot)_j \notin \mathcal{Q}_1(\{\boldsymbol{g}_i(\cdot)_j, \boldsymbol{g}^0_{i,LIN}(\cdot)_j, \boldsymbol{g}^0_{i,LIN}(\cdot)_{n_i+j}\}). \tag{43}$$

Moreover, at least one of the polynomials defining the constraints from (17a)-(18b) is not in $\mathcal{Q}_1(\{\boldsymbol{g}_i(\cdot)_j, \boldsymbol{g}^0_{i,LIN}(\cdot)_j, \boldsymbol{g}^0_{i,LIN}(\cdot)_{n_i+j}\})$, and consequently

$$\mathcal{Q}_1(\{\boldsymbol{g}_i(\cdot)_j, \boldsymbol{g}^0_{i,LIN}(\cdot)_j, \boldsymbol{g}^0_{i,LIN}(\cdot)_{n_i+j}\}) \subsetneqq \mathcal{Q}_1(\{\tilde{\boldsymbol{g}}^1_i, \tilde{\boldsymbol{g}}^2_i, \tilde{\boldsymbol{g}}^{t1}_i, \tilde{\boldsymbol{g}}^{t2}_i, i \in [\![1, L]\!]\}). \tag{44}$$

Numerical experiments, simultaneously illustrating both scalability and bound superiority of $\tau^1_{\mathrm{tighter,cs}}$, are reported in Table 3.

Table 3: Behaviour of bounds from the first-order sparse relaxations on randomly generated $\|.\|_\infty$-verification instances. Parameter s represents row sparsity, i.e., the number of non-zero elements in each of the weight matrices rows, while "dense" indicates fully populated weight matrices. The relative τ_{LP} running time being significantly smaller, we omit it from the table.

Network size and sparsity	bound			*t (s)*	
	$\tau^1_{\mathrm{tighter,cs}}$	τ^1_{QCQP}	τ_{LP}	$\tau^1_{\mathrm{tighter,cs}}$	τ^1_{QCQP}
$[300, 100, 100]$, dense	**-24.71**	-78.64	-78.83	413.51	427.28
$[500, 100, 100]$, dense	**-27.17**	-78.58	-77.89	795.41	721.81
$[300, 100, 100, 100]$, dense	**-19.49**	-77.06	-76.81	739.21	992.66
$[500, 100, 100, 100]$, dense	**-32.35**	-82.35	-81.94	928.16	964.56
$[1000, 100, 100, 100]$, dense	**-26.19**	-81.70	-81.32	1764.67	1659.13
$[300, 100, 100, 100, 100]$, dense	**-30.58**	-80.59	-80.42	966.76	1523.92
$[500, 100, 100, 100, 100]$, dense	**-16.19**	-77.61	-77.25	1341.33	1498.51
$[1000, 100, 100, 100, 100]$, dense	**-18.31**	-76.32	-75.89	1959.30	2237.44
$[300, 100, 100, 100, 100, 100]$, dense	**-28.62**	-80.95	-80.99	1012.18	1803.31
$[500, 100, 100, 100, 100, 100]$, dense	**-16.04**	-78.10	-78.12	1255.16	1641.50
$[1000, 100, 100, 100, 100, 100]$, dense	**-32.54**	-83.93	-82.84	2225.89	2866.54
$[500, 100, 100]$, $s = 10$	**-80.84**	-94.59	-98.16	12.53	14.90
$[800, 200, 200]$, $s = 10$	**-181.52**	-194.96	-198.97	218.72	186.77
$[1000, 100, 100]$, $s = 10$	**-81.13**	-94.72	-98.18	13.44	12.55
$[1000, 200, 200]$, $s = 10$	**-174.13**	-192.79	-198.61	243.87	191.24
$[1000, 300, 300]$, $s = 10$	**-242.60**	-283.95	-297.99	1203.41	1093.90
$[1000, 100, 100, 100]$, $s = 10$	**-72.79**	-92.98	-97.78	69.70	77.63
$[1000, 100, 100, 100, 100]$, $s = 10$	**-81.81**	-95.30	-98.30	327.83	319.04
$[1000, 500, 500]$, $s = 3$	**-476.01**	-495.79	-499.78	520.61	502.97
$[2000, 500, 500]$, $s = 3$	**-466.91**	-492.96	-499.69	238.35	225.89
$[2000, 300, 300, 300]$, $s = 3$	**-283.58**	-296.90	-299.75	469.02	309.20
$[2000, 1000, 1000]$, $s = 2$	**-944.68**	-989.77	-999.83	207.37	245.58
$[3000, 1000, 1000]$, $s = 2$	**-943.68**	-989.65	-999.83	81.96	84.25
$[3000, 2000, 1000]$, $s = 2$	**-982.09**	-996.68	-999.97	662.55	648.97
$[2000, 500, 500, 500]$, $s = 2$	**-484.19**	-497.30	-499.90	144.28	146.75
$[3000, 500, 500, 500]$, $s = 2$	**-480.42**	-496.66	-499.88	137.56	159.70

As it can be observed in Table 3, the bound $\tau^1_{\mathrm{tighter,cs}}$ is consistently better, across all instances. Moreover, the running time of $\tau^1_{\mathrm{tighter,cs}}$ is relatively stable, even for networks with large number of neurons, akin to small-to-medium-sized convolutional networks.

A.4 CLIQUES STRUCTURE DISCUSSION

Let us describe in details the clique structure of the first-order sparse SDP relaxation of the problem (19). We emphasize that this structure is valid only for the first-order sparse SDP relaxations, and regardless of the norm describing the input perturbation region.
We recall that a set of cliques $(I_k)_{k\in[\![1,p]\!]}$ is said to satisfy the RIP property is

$$(I_{k+1} \cap \cup_{j\leq k} I_j) \subset I_i, \text{ for some } i \leq k. \tag{45}$$

The RIP property (45) is one of the essential requirements for the convergence of the sparse SOS-based hierarchy of SDP relaxations (see Assumption 3.1 in Magron & Wang (2023)).

Theorem A.4.1. *The clique structure of the standard BNN robustness verification problem is such that:*

- *If $L = 1$, there are n_0 cliques of size n_1+1, given by $I_k = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}, \boldsymbol{x}^0_k\}$, $k \in [\![1, n_0]\!]$.*
- *If $L = 2$, there are $n_0 + n_2$ cliques of size $n_1 + 1$, given by $I_k = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}, \boldsymbol{x}^0_k\}$ for $k \in [\![1, n_0]\!]$ and $I_{n_0+k} = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}, \boldsymbol{x}^2_k\}$ for $k \in [\![1, n_2]\!]$.*
- *If $L \geq 3$, there are $L - 2 + n_0 + n_L$ cliques in total, and the maximum clique size is given by $\max\{n_1 + 1, n_{L-1} + 1, \max_{j\leq L-2}\{n_j + n_{j+1}\}\}$. The cliques are given by $I_k = \{\boldsymbol{x}^k_1, \ldots, \boldsymbol{x}^k_{n_k}, \boldsymbol{x}^{k+1}_1, \ldots, \boldsymbol{x}^{k+1}_{n_{k+1}}\}$ for $k \in [\![1, L-2]\!]$, $I_{L-2+k} = \{\boldsymbol{x}^0_k, \boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}\}$ for $k \in [\![1, n_0]\!]$ and $I_{L-2+n_0+k} = \{\boldsymbol{x}^{L-1}_1, \ldots, \boldsymbol{x}^{L-1}_{n_{L-1}}, \boldsymbol{x}^L_k\}$ for $k \in [\![1, n_L]\!]$.*

In any of the three cases, the defined clique structure satisfies the RIP property.

Proof of Theorem A.4.1. We consider different cases, depending on the depth of the BNN:

- $L = 1$:

  All the nodes from the hidden layer form a separate clique with each individual input node, resulting in n_0 cliques of size $n_1 + 1$. Thus, we can set $I_k = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}, \boldsymbol{x}^0_k\}$, for $k \in [\![1, n_0]\!]$. The RIP property is clearly satisfied, as the nodes from the hidden layer appear in each clique.

- $L = 2$:

  All the nodes from the first hidden layer form a separate clique with each individual input node, resulting in n_0 cliques of size $n_1 + 1$. Thus, we can set $I_k = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}, \boldsymbol{x}^0_k\}$, for $k \in [\![1, n_0]\!]$. Moreover, they also form a separate clique with each individual node from the second hidden layer, resulting in n_2 cliques of size $n_1 + 1$. Thus, we can set $I_{n_0+k} = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}, \boldsymbol{x}^2_k\}$, for $k \in [\![1, n_2]\!]$. The RIP property is satisfied as the first hidden layer nodes appear in each clique.

- $L \geq 3$:

  For every $k \in [\![1, L-2]\!]$, a pair $(k, k+1)$ of adjacent hidden layers forms a clique $I_k = \{\boldsymbol{x}^k_1, \ldots, \boldsymbol{x}^k_{n_k}, \boldsymbol{x}^{k+1}_1, \ldots, \boldsymbol{x}^{k+1}_{n_{k+1}}\}$ of size $n_k + n_{k+1}$. Furthermore, all the nodes from the first hidden layer form a separate clique with each individual input node, resulting in n_0 cliques of size $n_1 + 1$. Thus, we can set $I_{L-2+k} = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}, \boldsymbol{x}^0_k\}$, for $k \in [\![1, n_0]\!]$. Finally, all the nodes from the penultimate hidden layer form a separate clique with each individual node from the last hidden layer, resulting in n_L cliques of size $n_{L-1} + 1$. Thus, we can set $I_{L-2+n_0+k} = \{\boldsymbol{x}^{L-1}_1, \ldots, \boldsymbol{x}^{L-1}_{n_{L-1}}, \boldsymbol{x}^L_k\}$, for $k \in [\![1, n_L]\!]$.

  Therefore, there are $L - 2 + n_0 + n_L$ cliques in total, and the maximum clique size is given by $\max\{n_1+1, n_{L-1}+1, \max_{j\leq L-2}\{n_j + n_{j+1}\}\}$. Moreover, given this particular clique ordering (enumeration), the RIP property is satisfied. Indeed,

  - If $k \in [\![2, L-2]\!]$, then
    $$I_k \cap (\cup_{j\leq k-1} I_j) = \{\boldsymbol{x}^{k-1}_1, \ldots, \boldsymbol{x}^{k-1}_{n_{k-1}}\} \subset I_{k-1}. \tag{46}$$
  - If $k \in [\![L-2+1, L-2+n_0]\!]$, then
    $$I_k \cap (\cup_{j\leq k-1} I_j) = \{\boldsymbol{x}^1_1, \ldots, \boldsymbol{x}^1_{n_1}\} \subset I_1. \tag{47}$$

- If $k \in [\![L-2+n_0+1, L-2+n_0+n_L]\!]$, then

$$I_k \cap (\cup_{j \leq k-1} I_j) = \{\boldsymbol{x}_1^{L-1}, \ldots, \boldsymbol{x}_{n_{L-1}}^{L-1}\} \subset I_{L-2}. \tag{48}$$

Thus, the RIP property holds in all three cases, which concludes the proof. □

An alternative clique structure, independent of the depth of the BNN, is to consider the cliques $\{\boldsymbol{x}_j^{i-1}, \boldsymbol{x}_k^i\}_{1 \leq j \leq n_{i-1}, 1 \leq k \leq n_i}^{1 \leq i \leq L}$. This option results in $\sum_{i=1}^{L} n_{i-1} \times n_i$ cliques of size two, since every node is coupled with all the other nodes form adjacent layers. The RIP property always holds in this case if cliques are enumerated in an order that mirrors the feed-forward structure of the BNN. Computations are generally faster for this clique structure, but the obtained bounds are much coarser.

B EXPERIMENTAL SETUP

Training details: All neural networks have been trained on MNIST dataset (data were scaled to belong to $[-1, 1]^{784}$) using Larq (Geiger & Team, 2020), an open-source Python library for training neural networks with quantized (binarized) weights and activation functions.

The training process, lasting 300 epochs, has consisted of minimizing the sparse categorical cross-entropy, using the Adam optimizer (Kingma & Ba, 2014). The learning rate has been handled by the exponential decay learning scheduler, whose initial value has been set to be 0.001. SteTern quantizer with different threshold values has been used to induce different sparsities of the weights matrices. Network parameters have been initialized from a uniform distribution. Batch normalization layers have been added to all but the output layer.
All networks have achieved over 95% accuracy on the test set.

Optimization details: Semidefinite Programming (SDP) problems have been assembled using TSSOS (Wang et al., 2021), a specialized Julia library for polynomial optimization. Correlative sparsity has been exploited by using the CS="MF" option, which generates an approximately smallest chordal extension of the identified clique structure. The SDP values have been computed via the interior-point solver Mosek (Andersen & Andersen, 2000), and the SDP solving time was recorded.

Since computing the exact value of the objective function for MILP and MINP problems is highly computationally demanding, we transform those problems into satisfiability problems by setting the objective function to be a constant function equal to zero, and adding $f(\cdot) \leq 0$ to the set of constraints. Infeasibility of the resulting problem provides a certificate of robustness. These problems have then been solved using Gurobi, where the time limit was set to 600 seconds. The upper bound presented in Figure 2 has been determined by randomly sampling $10,000$ points from the uniform distribution over the perturbation region and documenting the lowest objective function value observed at those points.

More detailed experimental results: We provide detailed verification results against $||.||_\infty$ (see Table 4 and Table 5) and $||.||_2$ (see Table 6 and Table 7) attacks. Those results illustrate that the solving time of general branch and bound algorithms could be significantly improved if the LP bounds would be replaced by tighter SDP bounds, such as $\tau^1_{\text{tighter,cs}}$. The improvement would be even more noticeable for $||.||_2$ verification.

Finally, due to their increased computational complexity, the experiments from Table 3 were run on a server with a 26-core Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz and a RAM of 756GB.

Table 4: Verification against $\|.\|_\infty$-attacks: detailed numerical results for BNN_1 : $[784, 500, 500, 10], w_s = 34.34\%$. For example, image 72 illustrates that MILP methods can not handle severe attacks. Likewise, image 28 confirms the potential benefits of accommodating $\tau^1_{\text{tighter,cs}}$ within the branch and bound algorithms. Note that MILP implementation only solves the attack feasibility problem, since computing exact bounds would result in much more timeouts.

Image Index	$\tau^1_{\text{tighter,cs}}$ bound	$\tau^1_{\text{tighter,cs}}$ $t\ (s)$	$\tau_{\text{Soft-MILP}}$ bound	$\tau_{\text{Soft-MILP}}$ $t\ (s)$
$\delta_{\|\|.\|\|_\infty} = 1.25$				
14	1.08	11.53	infeasible	552.99
24	5.82	15.43	timeout	> 600
26	0.31	7.18	infeasible	4.07
28	29.86	14.90	infeasible	29.08
31	7.54	10.08	infeasible	11.67
52	12.17	7.53	infeasible	0.49
55	11.05	6.99	infeasible	3.33
57	3.23	8.69	infeasible	6.41
72	29.43	9.75	infeasible	1.36
83	17.29	14.07	infeasible	44.62
100	19.99	6.64	infeasible	0.30
$\delta_{\|\|.\|\|_\infty} = 1.50$				
28	11.18	70.62	timeout	> 600
52	0.48	18.35	infeasible	7.18
55	2.25	16.33	infeasible	7.42
72	20.65	28.52	infeasible	15.68
83	8.37	17.68	infeasible	519.23
100	8.32	10.39	infeasible	2.21
$\delta_{\|\|.\|\|_\infty} = 1.75$				
28	0.20	42.46	timeout	> 600
72	13.50	37.79	timeout	> 600

Table 5: Verification against $\|.\|_\infty$-attacks: detailed numerical results for BNN_2 : $[784, 500, 500, 10], w_s = 19.07\%$. Remarkable improvements can be observed for images $20, 28, 29$ and 33.

Image Index	$\tau^1_{\text{tighter,cs}}$ bound	$\tau^1_{\text{tighter,cs}}$ $t\ (s)$	$\tau_{\text{Soft-MILP}}$ bound	$\tau_{\text{Soft-MILP}}$ $t\ (s)$
$\delta_{\|\|.\|\|_\infty} = 0.75$				
11	19.54	42.31	infeasible	13.39
20	8.72	107.36	timeout	> 600
26	14.90	37.25	infeasible	10.61
28	14.26	75.15	infeasible	227.07
29	7.48	47.33	infeasible	200.02
33	7.31	49.35	timeout	> 600
55	25.14	28.96	infeasible	4.67
72	35.33	35.80	infeasible	6.85
83	22.14	44.61	infeasible	16.26

Table 6: Verification against $\|.\|_2$-attacks: detailed numerical results for $\text{BNN}_1 : [784, 500, 500, 10], w_s = 34.34\%$. Notice that the number of timeouts is significant. On the other hand, the verification time associated to $\tau^1_{\text{tighter,cs}}$ remains relatively stable.

Image Index	$\tau^1_{\text{tighter,cs}}$		$\tau_{\text{Soft-MINP}}$	
	bound	$t\ (s)$	bound	$t\ (s)$
$\delta_{\|.\|_2} = 20$				
4	4.20	12.79	timeout	> 600
10	0.71	16.41	infeasible	564.03
11	6.05	13.91	infeasible	376.86
12	14.34	8.00	infeasible	181.41
13	5.03	21.96	timeout	> 600
14	34.12	15.05	infeasible	47.93
15	4.35	18.58	timeout	> 600
23	9.91	9.13	infeasible	0.40
24	37.52	12.37	infeasible	354.42
26	32.32	11.12	infeasible	0.71
28	58.43	22.14	infeasible	203.53
31	27.73	8.31	infeasible	1.05
48	12.48	16.48	timeout	> 600
49	14.85	15.72	timeout	> 600
52	34.45	10.35	infeasible	0.07
55	42.34	5.32	infeasible	0.08
56	15.68	11.15	timeout	> 600
57	30.32	6.16	infeasible	0.48
59	17.66	27.45	timeout	> 600
61	8.08	15.45	infeasible	102.50
65	5.20	8.99	timeout	> 600
69	12.73	19.38	infeasible	272.38
70	6.37	15.82	infeasible	292.86
72	46.78	15.88	infeasible	0.15
73	18.78	11.73	infeasible	14.15
77	22.70	30.85	infeasible	480.90
80	11.89	12.12	infeasible	72.41
83	43.12	20.10	infeasible	60.96
86	9.89	14.54	infeasible	531.24
89	18.74	17.25	infeasible	229.98
91	11.59	18.89	infeasible	505.02
92	13.55	34.95	timeout	> 600
94	7.13	14.75	infeasible	62.56
98	2.64	13.02	infeasible	165.10
99	13.27	12.16	infeasible	100.43
100	42.32	8.95	infeasible	0.09
$\delta_{\|.\|_2} = 30$				
14	11.81	17.54	timeout	> 600
24	15.04	28.94	timeout	> 600
26	15.48	7.28	infeasible	108.11
28	35.08	40.25	timeout	> 600
31	6.89	16.39	timeout	> 600
52	16.78	10.86	infeasible	191.03
55	21.80	10.49	infeasible	555.08
57	12.04	10.93	timeout	> 600
72	35.81	12.45	infeasible	353.96
73	3.59	10.44	timeout	> 600
77	3.45	37.65	timeout	> 600
83	26.76	22.78	timeout	> 600
100	25.51	9.14	infeasible	64.95

Table 7: VerificatioN against $\|.\|_2$-attacks: detailed numerical results for $\mathrm{BNN}_2 : [784, 500, 500, 10], w_s = 19.07\%$. Images 57 and 89 demonstrate that $\tau^1_{\mathrm{tighter,cs}}$ can typically certify robustness more than times faster.

Image Index	$\tau^1_{\mathrm{tighter,cs}}$		$\tau_{\text{Soft-MINP}}$	
	bound	$t\ (s)$	bound	$t\ (s)$
$\delta_{\|.\|_2} = 15$				
2	2.74	31.34	infeasible	279.66
4	0.14	47.94	timeout	> 600
11	50.95	17.32	infeasible	12.99
14	19.98	39.04	timeout	> 600
20	33.38	88.27	timeout	> 600
23	13.43	19.91	infeasible	253.64
26	40.90	16.40	infeasible	12.57
28	40.87	45.39	infeasible	73.84
29	39.00	23.42	infeasible	9.52
31	15.00	15.89	infeasible	59.66
33	30.05	51.76	infeasible	134.40
48	16.31	47.78	timeout	> 600
55	53.65	12.11	infeasible	0.25
57	6.16	25.38	timeout	> 600
61	2.29	19.44	infeasible	253.44
69	20.93	67.48	timeout	> 600
71	18.96	17.43	infeasible	88.33
72	61.78	14.47	infeasible	1.40
73	25.72	14.60	infeasible	62.32
77	2.48	37.19	infeasible	510.44
83	47.72	23.62	infeasible	81.36
86	6.54	115.93	timeout	> 600
87	22.33	69.43	infeasible	344.06
88	15.48	23.39	infeasible	250.97
89	27.29	29.73	timeout	> 600
92	23.51	57.17	timeout	> 600
100	15.40	20.03	infeasible	108.08

Experimental results for more complex data sets and larger networks: To further demonstrate the versatility of our method, we provide additional illustrative experiments for $\text{BNN}_3 : [3072, 5000, 800, 10], w_s = 55.97\%$, achieving the test accuracy of 47.66% on CIFAR-10 data set.

Table 8: Verifying robustness of BNN_3 on CIFAR-10, for an input region determined by $\delta_{||.||_\infty} = 0.2/255$. Data were scaled to $[-1, 1]^{3072}$, and a time limitation of 3600 s was used. We present the results of the robustness verification queries on the correctly classified images from the first 40 images from the test data set.

Image Index	$\tau^1_{\text{tighter,cs}}$		$\tau_{\text{Soft-MILP}}$	
	bound	t (s)	bound	t (s)
1	20.46 (robust)	**1179.55**	timeout	> 3600
2	7.83 (robust)	**571.08**	timeout	> 3600
7	-30.28 (unknown)	380.10	timeout	> 3600
8	-68.50 (unknown)	1721.79	feasible (not robust)	1.71
10	-62.31 (unknown)	1330.03	feasible (not robust)	87.46
11	-142.50 (unknown)	650.79	feasible (not robust)	0.11
12	-94.92 (unknown)	1115.18	feasible (not robust)	0.071
14	-103.36 (unknown)	493.80	feasible (not robust)	0.067
15	-71.09 (unknown)	2090.44	feasible (not robust)	57.09
18	-97.96 (unknown)	1469.67	feasible (not robust)	0.08
19	31.06 (robust)	344.70	infeasible (robust)	9.02
20	-25.66 (unknown)	823.54	timeout	> 3600
21	-90.54 (unknown)	670.94	feasible (not robust)	0.59
24	-29.08 (unknown)	629.50	timeout	> 3600
27	-83.76 (unknown)	766.95	feasible (not robust)	0.54
29	-110.26 (unknown)	873.51	feasible (not robust)	0.07
30	-60.63 (unknown)	1407.68	feasible (not robust)	1.39
31	-29.51 (unknown)	1190.56	timeout	> 3600
33	timeout	> 3600	timeout	> 3600
34	-73.15 (unknown)	569.46	feasible (not robust)	0.51
35	36.62 (robust)	**657.73**	timeout	> 3600
40	-52.82 (unknown)	650.79	timeout	> 3600

As illustrated in Table 8, our approach demonstrates comparable performance even on larger datasets, such as CIFAR-10, and for larger networks involving nearly 9000 neurons.

Specifically, $\tau^1_{\text{tighter,cs}}$ proves the robustness of images 1, 2 and 35 at least 3x, 6x, and 5.5x faster, respectively. In contrast, the low quality of LP bounds prevents MILP from providing an answer within a one-hour time limit. These additional experimental results further validate that our method consistently provides high-quality lower bounds, even for large-scale problems, which is consistent with the results for BNN_1 and BNN_2.

However, for instances 8 and 10, our method is unable to provide an answer, while MILP can efficiently certify non-robustness. This difference arises due to the fact that MILP based methods do not aim to solve the optimization problems to global optimality but instead focus on determining the feasibility of an adversarial attack, which is inherently less computationally demanding.

We believe that incorporating our tighter SDP bounds within the MILP framework could enhance the ability of MILP methods to certify *robustness* in more complex cases. This represents a promising direction for future research.