Verifying Properties of Binary Neural Networks Using Sparse Polynomial Optimization

Srećko Đurašinović, Jianting Yang, Jean-Bernard Lasserre, Victor Magron, Jun Zhao

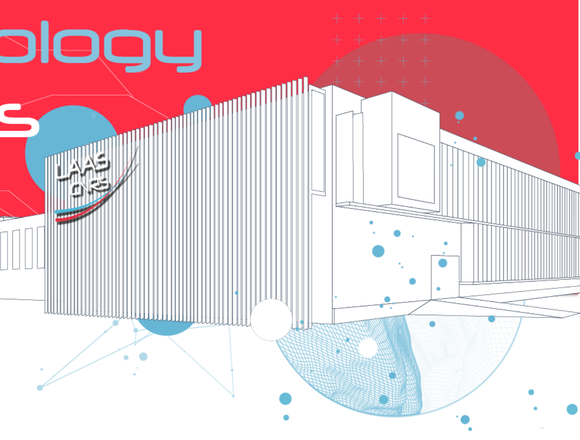
Verifying Properties of Binary Neural Networks Using Sparse Polynomial Optimization

Srećko Đurašinović[1,2]

Joint work with Jianting Yang[1], Jean-Bernard Lasserre[3], Victor Magron[3] and Jun Zhao[2]

[1]CNRS@CREATE, Singapore; [2]Nanyang Technological University, Singapore; [3]LAAS-CNRS

LAAS Research and Technology Days



Summary

This work 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 (SMT) and Mixed-Integer Linear Programming (MILP) techniques, which often face scalability issues. We introduce an alternative approach using Semidefinite Programming (SDP) relaxations derived from sparse Polynomial Optimization (POP). 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.

Keywords: Binary Neural Networks, Robustness Verification, Sparse Polynomial Optimization, Semidefinite Programming

Binary Neural Networks

Let $L \geq 1$ be the number of hidden layers of a classifying BNN, with layer widths being given by $\boldsymbol{n} = (n_0, n_1, \ldots, n_L, n_{L+1})^\mathsf{T} \in \mathbb{N}^{L+2}$, 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 $\mathbf{B}_o$:

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

For any $i \in [\![1, L]\!]$, the internal block $\mathbf{B}_i$ implements successively three different operations: affine transformation, batch normalization and point-wise binarization, so that its output vector, denoted by $\boldsymbol{x}^i$, belongs to $\{-1, 1\}^{n_i}$. These operations are described by a set of trainable parameters:

$$\left(\boldsymbol{W}^{[i+1]}, \boldsymbol{b}^{[i+1]}\right)_{i\in[\![0,L]\!]} \in \{-1, 0, 1\}^{n_{i+1}\times n_i} \times \mathbb{R}^{n_{i+1}}, \ \left(\boldsymbol{\gamma}^{[i]}, \boldsymbol{\beta}^{[i]}, \boldsymbol{\mu}^{[i]}, \boldsymbol{\sigma}^{2,[i]}\right)_{i\in[\![1,L]\!]} \in (\mathbb{R}^{n_i})^4. \tag{2}$$

Steps	Input	Output	Transformation
Linearization	$\boldsymbol{x}^{i-1} \in \{-1, 1\}^{n_{i-1}}$	$\boldsymbol{y} \in \mathbb{R}^{n_i}$	$\boldsymbol{y} = \boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}$
Batch-Normalization	$\boldsymbol{y} \in \mathbb{R}^{n_i}$	$\boldsymbol{z} \in \mathbb{R}^{n_i}$	$z_j = \boldsymbol{\gamma}_j^{[i]}\left(\frac{y_j - \boldsymbol{\mu}_j^{[i]}}{\sqrt{\boldsymbol{\sigma}_j^{2,[i]} + \epsilon}}\right) + \boldsymbol{\beta}_j^{[i]}$
Binarization	$\boldsymbol{z} \in \mathbb{R}^{n_i}$	$\boldsymbol{x}^i \in \{-1, 1\}^{n_i}$	$\boldsymbol{x}^i = \text{sign}(\boldsymbol{z})$

Table 1: Structure of an internal block $\mathbf{B}_i$.

The output block $\mathbf{B}_o$ applies a softmax transformation to the affinely-transformed outputs of the last hidden layer, i.e., for each $j \in [\![1, n_{L+1}]\!]$,

$$\boldsymbol{x}_j^{L+1} = \frac{\exp(z_j)}{\sum_{k=1}^{n_{L+1}} \exp(z_k)}, \text{ where } z_j = \boldsymbol{W}_{(j:,)}^{[L+1]}\boldsymbol{x}^L + \boldsymbol{b}_j^{[L+1]}. \tag{3}$$

Deriving a Tighter First-Order Relaxation - $\tau^1_{\text{tighter,cs}}$

$\Longrightarrow$ For each $i \in [\![1, L]\!]$, we replace the constraint defined in (4b) by the following two constraints:

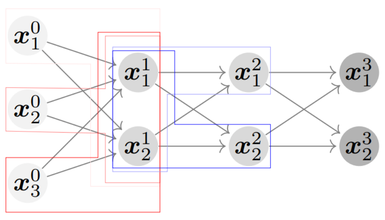
$$\begin{cases} \tilde{\boldsymbol{g}}_i^1(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\boldsymbol{x}^i + \mathbf{1}) \odot \left(\boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}\right) \geq \mathbf{0}, & \text{(6a)} \\ \tilde{\boldsymbol{g}}_i^2(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\boldsymbol{x}^i - \mathbf{1}) \odot \left(\boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}\right) \geq \mathbf{0}. & \text{(6b)} \end{cases}$$

$\Longrightarrow$ Add the following two *redundant* quadratic constraints, i.e., *tautologies*, to the optimization problem (5):

$$\begin{cases} \tilde{\boldsymbol{g}}_i^{\text{t1}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\boldsymbol{x}^i + \mathbf{1}) \odot \left(\text{nv}\left(\boldsymbol{W}^{[i]}\right) - \boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1}\right) \geq \mathbf{0}, & \text{(7a)} \\ \tilde{\boldsymbol{g}}_i^{\text{t2}}(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := (\mathbf{1} - \boldsymbol{x}^i) \odot \left(\text{nv}\left(\boldsymbol{W}^{[i]}\right) + \boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1}\right) \geq \mathbf{0}, & \text{(7b)} \end{cases}$$

Theorem

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 affine f.



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

Figure 1: (left) 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 $\tau^1_{\text{tighter,cs}}$ (right). We let $\tilde{\boldsymbol{g}}_i = \{\tilde{\boldsymbol{g}}_i^1, \tilde{\boldsymbol{g}}_i^2, \tilde{\boldsymbol{g}}_i^{\text{t1}}, \tilde{\boldsymbol{g}}_i^{\text{t2}}\}$.

Sparse SDP for BNN Robustness Verification

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

$$\boldsymbol{x}^i := \text{sign}\left(\boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}\right) \implies \begin{cases} \boldsymbol{h}_i(\boldsymbol{x}^i) := \boldsymbol{x}^i \odot \boldsymbol{x}^i - \mathbf{1} = \mathbf{0}, & \text{(4a)} \\ \boldsymbol{g}_i(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) := \boldsymbol{x}^i \odot (\boldsymbol{W}^{[i]}\boldsymbol{x}^{i-1} + \boldsymbol{b}^{[i]}) \geq \mathbf{0}, & \text{(4b)} \end{cases}$$

Suppose that the input perturbation region $\boldsymbol{B} \subseteq \mathbb{R}^{n_0}$ can be encoded via positivity conditions on (at most quadratic) polynomials $\boldsymbol{x}^0 \mapsto \boldsymbol{g}_{\boldsymbol{B}}(\boldsymbol{x}^0)$. Then, the *standard* form BNN verification problems becomes:

$$\tau := \begin{cases} \min\limits_{\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L} f(\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L) & \text{(5a)} \\ \text{s.t.} \quad \boldsymbol{h}_i(\boldsymbol{x}^i) = \mathbf{0}, \ i \in [\![1, L]\!], & \text{(5b)} \\ \qquad \boldsymbol{g}_i(\boldsymbol{x}^i, \boldsymbol{x}^{i-1}) \geq \mathbf{0}, \ i \in [\![1, L]\!], & \text{(5c)} \\ \qquad \boldsymbol{g}_{\boldsymbol{B}}(\boldsymbol{x}^0) \geq \mathbf{0}, & \text{(5d)} \end{cases}$$

where f is linear/quadratic, e.g., $f = f_k^{\text{adv}}(\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^L) := \langle \boldsymbol{W}_{(\bar{y}:,)}^{[L+1]} - \boldsymbol{W}_{(k:,)}^{[L+1]}, \boldsymbol{x}^L \rangle + \boldsymbol{b}_{\bar{y}}^{[L+1]} - \boldsymbol{b}_k^{[L+1]}$.

Exploiting Sparsity

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 (5) exhibits *correlative sparsity* since

- There exist $(f_k)_{k\in[\![1,p]\!]}$ such that $f = \Sigma_k^p f_k$, 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.

The hierarchy of correlatively sparse SDP relaxations is then given by

$$\tau_{\text{cs}}^d := \sup_{\lambda,\sigma}\{\lambda \in \mathbb{R} \mid f - \lambda - \sum_{k=1}^{p} \sigma_k \in \mathcal{I}_d(\boldsymbol{h}), \sigma_k \in \mathcal{Q}_d(\{\boldsymbol{g}_i(\cdot)_j \in J_k\})\}.$$

Some Numerical Results

$\Longrightarrow$ Verification against $\|.\|_\infty$-attacks $\Downarrow$

Model	$\delta_{\|.\|_\infty}$	τ_{LP} cert.	τ_{LP} $t\,(s)$	$\tau^1_{\text{tighter,cs}}$ cert.	$\tau^1_{\text{tighter,cs}}$ $t\,(s)$	$\tau_{\text{Soft-MILP}}$ cert.	$\tau_{\text{Soft-MILP}}$ $t\,(s)$
BNN_1: [784, 500, 500, 10] $w_s = 34.34\%$	0.25	83	0.01	91	3.62	3 | 95	0.04
	0.50	31	0.02	60	6.69	4 | 94	1.21
	1.00	1	0.03	21	10.76	15 | 50	251.90
	1.50	0	0.06	6	38.32	20 | 12	428.24
BNN_2: [784, 500, 500, 10] $w_s = 19.07\%$	0.25	14	0.03	59	11.97	3 | 95	2.23
	0.50	0	0.05	23	42.37	9 | 63	220.53
	0.75	0	0.08	9	139.18	10 | 19	455.61

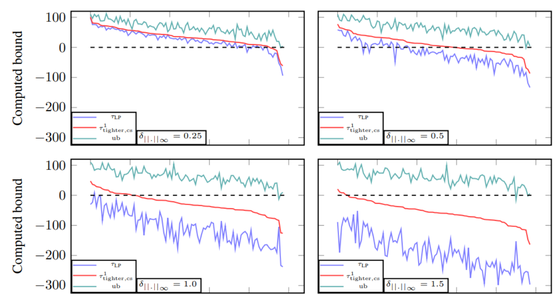


Figure 2: (left) Performance comparison for different models and input regions, given by $\delta_{\|.\|_\infty} = 127.5\epsilon$. (right) Comparing τ_{LP} and $\tau^1_{\text{tighter,cs}}$ bounds for BNN_1 and different $\delta_{\|.\|_\infty}$. The relative improvement is up to 55%.

$\Longrightarrow$ Verification against $\|.\|_2$-attacks $\Downarrow$

$\delta_{\|.\|_2}$	$\tau^1_{\text{tighter,cs}}$ cert.	$\tau^1_{\text{tighter,cs}}$ $t\,(s)$	$\tau_{\text{Soft-MINP}}$ cert.	$\tau_{\text{Soft-MINP}}$ $t\,(s)$
$\text{BNN}_1 : [784, 500, 500, 10], w_s = 34.34\%$				
10	70	5.23	3 | 93	33.35
20	36	19.54	4 | 30	447.11
30	13	34.24	4 | 6	556.07

$\delta_{\|.\|_2}$	$\tau^1_{\text{tighter,cs}}$ cert.	$\tau^1_{\text{tighter,cs}}$ $t\,(s)$	$\tau_{\text{Soft-MINP}}$ cert.	$\tau_{\text{Soft-MINP}}$ $t\,(s)$
$\text{BNN}_2 : [784, 500, 500, 10], w_s = 19.07\%$				
5	81	8.57	2 | 96	3.31
10	46	19.00	3 | 58	272.92
15	27	63.21	4 | 23	475.78

Image Index	$\tau^1_{\text{tighter,cs}}$ bound	$\tau^1_{\text{tighter,cs}}$ $t\,(s)$	$\tau_{\text{Soft-MINP}}$ bound	$\tau_{\text{Soft-MINP}}$ $t\,(s)$
$\delta_{\|.\|_2} = 15$				
4	0.14	47.94	timeout	> 600
14	19.98	39.04	timeout	> 600
20	33.38	88.27	timeout	> 600
48	16.31	47.78	timeout	> 600
57	6.16	25.38	timeout	> 600
69	20.93	67.48	timeout	> 600
86	6.54	115.93	timeout	> 600
89	27.29	29.73	timeout	> 600
92	23.51	57.17	timeout	> 600

Figure 3: (left, center) Performance comparison for different BNN models and input regions, given by $\delta_{\|.\|_2} = 255\epsilon$. (right) Verification against $\|.\|_2$-attacks: illustrating the significant speedup for specific instances.

References

1. Jianting Yang, Srećko Đurašinović, Jean-Bernard Lasserre, Victor Magron and Jun Zhao. *Verifying Properties of Binary Neural Networks Using Sparse Polynomial Optimization.* arXiv preprint arXiv:2405.17049, 2024.
2. Victor Magron and Jie Wang. *Sparse Polynomial Optimization: Theory and Practice.* World Scientific, 2023.
3. Victor Magron and Jie Wang. *TSSOS: a Julia library to exploit sparsity for large-scale polynomial optimization.* arXiv preprint arXiv:2103.00915, 2021.

LAAS-CNRS
/ Laboratoire d'analyse et d'architecture des systèmes du CNRS

Laboratoire conventionné avec
Université de Toulouse