Neuron Pairs in Binarized Neural Networks Robustness Verification via Integer Linear Programming

Dymitr Lubczyk, José Neto

Neuron pairs in binarized neural networks robustness verification via integer linear programming

Dymitr Lubczyk[1] and José Neto[2]

[1] University of Amsterdam,
Science Park 904, 1098 XH Amsterdam, Netherlands
dymitr.lubczyk@student.uva.nl
[2] Samovar, Télécom SudParis, Institut Polytechnique de Paris,
19 place Marguerite Perey, 91120 Palaiseau, France
jose.neto@telecom-sudparis.eu

Abstract. In the context of classification, robustness verification of a neural network is the problem which consists in determining if small changes of inputs lead to a change of their assigned classes. We investigate such a problem on binarized neural networks via an integer linear programming perspective. We namely present a constraint generation framework based on disjunctive programming and complete descriptions of polytopes related to outputs of neuron pairs. We also introduce an alternative relying on specific families of facet defining inequalities. Preliminary experiments assess the performance of the latter approach against recent single neuron convexification results.

Keywords: Disjunctive programming · Cutting-plane · Robustness verification

1 Introduction

Nowadays Deep Neural Networks (DNNs) turn out to be successful in diverse domains such as computer vision, natural language processing, machine translation, etc. (see, e.g., [8,15]). One of the reasons for this is their great expressiveness [10], which comes however at the price of being hard to reason about [21]. The latter motivated research directed towards the assessment and improvement of the robustness of DNNs for their use in the context of critical AI systems. Another important drawback of many DNNs lies in the fact that they resort to important amounts of computational and energy resources. Through the use of binarized weights and a simple activation function, binarized neural networks (BNNs) appear as an interesting option in the context of resource-constrained systems. However, already challenging optimization problems in the context of ReLU DNNs (such as verification or training) may seem even harder for BNNs due to their inherent discrete features. In this paper we focus on robustness verification of BNNs. Our main contributions are

- the presentation of a constraint generation algorithm based on disjunctive programming and complete descriptions established for polytopes related to outputs of neuron pairs,
- the design and evaluation of a constraint generation algorithm relying on specific families of facet defining inequalities to solve a robustness verification problem for BNNs.

The paper is organized as follows. In Section 2 we introduce a robustness verification problem for BNNs and point out works related to ours. Polyhedral results are reported in Section 3. Preliminary computational experiments are presented in Section 4 before we conclude in Section 5. Due to length restrictions proofs are omitted from this extended abstract.

2 Robustness verification problem in BNNs

2.1 Description of BNNs

A BNN is a special type of DNN having for activation function:

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases} \qquad \text{for all } x \in \mathbb{R}.$$

Given a vector $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{\sigma}(\boldsymbol{x}) \in \mathbb{B}^n$, with $\mathbb{B} = \{-1; 1\}$, denotes the vector such that $(\boldsymbol{\sigma}(\boldsymbol{x}))_i = \text{sign}(x_i)$, for all $i \in [n]$, where $[n] = \{1, 2, \ldots, n\}$.

A BNN has some number $s+1$ of ordered layers, the last one being called the *output layer* and the others *hidden layers*. Each layer k, with $k \in [s+1]$, has some number n_k of neurons and an associated weight matrix $\boldsymbol{W^k} \in \mathbb{B}^{n_k} \times \mathbb{B}^{n_{k-1}}$ which is computed during a training phase. Given an input vector $\boldsymbol{y^0} \in \mathbb{B}^{n_0}$, an output vector $\boldsymbol{x^{s+1}} \in \mathbb{Z}^{n_{s+1}}$ is computed using the recursion

$$\begin{aligned} \boldsymbol{x^k} &= \boldsymbol{W^k y^{k-1}}, \forall k \in [s+1] \\ \boldsymbol{y^k} &= \boldsymbol{\sigma}(\boldsymbol{x^k}), \forall k \in [s]. \end{aligned}$$

In the context of classification, each entry of the output vector $\boldsymbol{x^{s+1}}$ is associated to one class and the input $\boldsymbol{y}^0$ is assigned to the class corresponding to the largest entry of $\boldsymbol{x^{s+1}}$. (In what follows, we may assume ties are broken arbitrarily.) Given a BNN $\mathcal{B}$, the output vector obtained with the input $\boldsymbol{y^0}$ will be denoted by $\mathcal{B}(\boldsymbol{y^0})$.

2.2 Robustness verification problem

Let $\mathcal{B}$ represent a BNN as described above, and let $\boldsymbol{z} \in \mathbb{B}^{n_0}$ be an input vector whose class is ℓ and satisfying $(\mathcal{B}(\boldsymbol{z}))_\ell > (\mathcal{B}(\boldsymbol{z}))_j$ for all $j \in [n_{s+1}] \setminus \{\ell\}$. Given a target class $t \in [n_{s+1}] \setminus \{\ell\}$ and a neighborhood $\Omega(\boldsymbol{z})$ of $\boldsymbol{z}$ in $\mathbb{R}^{n_0}$, the BNN $\mathcal{B}$ will be said *locally robust* at $\boldsymbol{z}$ *w.r.t. the target* t if, for all $\boldsymbol{y^0} \in \Omega(\boldsymbol{z})$, the ℓ^{th}

entry of $\mathcal{B}\left(\boldsymbol{y^0}\right)$ is larger than the t^{th} entry. So, checking local robustness of $\mathcal{B}$ at $\boldsymbol{z}$ w.r.t. target t reduces to solving the following problem

$$\max_{\boldsymbol{x^k},\boldsymbol{y^k}} x_t^{s+1} - x_\ell^{s+1} \tag{1a}$$

$$\text{s.t. } \boldsymbol{x^k} = \boldsymbol{W^k y^{k-1}}, \forall k \in [s+1] \tag{1b}$$

$$\boldsymbol{y^k} = \boldsymbol{\sigma}(\boldsymbol{x^k}), \forall k \in [s] \tag{1c}$$

$$\boldsymbol{y^0} \in \Omega(\boldsymbol{z}) \tag{1d}$$

$$\boldsymbol{x^k} \in \mathbb{R}^{n_k}, k \in [s+1] \tag{1e}$$

$$\boldsymbol{y^k} \in \mathbb{B}^{n_k}, k \in \{0\} \cup [s] \tag{1f}$$

$\mathcal{B}$ is locally robust at $\boldsymbol{z}$ w.r.t. target t if and only if the optimal objective of (1) is negative. Note that even though the chosen neighborhood $\Omega(\boldsymbol{z})$ may be convex, problem (1) is not, due to constraints (1c).

2.3 Related work

The importance of robustness verification problems of DNNs stimulated much research efforts leading to the development of solution approaches relying on diverse search strategies, optimization methods and satisfiability modulo theories (SMT), see e.g. [2,4,16,17] and references therein. Among the vast literature in the field, the share dedicated to BNNs seems rather limited.

A BNN can be represented as a Boolean formula, a feature allowing the use of SAT solvers to verify robustness [5,12,19,20]. An SMT based approach extending the Reluplex method [13] is proposed by Amir et al. [1] to support sign activation functions (in addition to ReLU or max-pooling). Khalil et al. [14] design a heuristic to identify adversarial examples for BNNs that is based on the solution of several integer linear programs associated with the layers. This procedure aims at identifying adversarial examples, namely in the context of adversarial training to strengthen robustness of BNNs. However, it does not solve the verification problem exactly and it is rather proposed as an alternative to the solution of an exact formulation of the problem as a mixed integer linear program (MILP) introduced in the same reference but which does not scale to handle large BNNs. Efforts have been dedicated to strengthen relaxations based on such MILP formulations so as to improve performance further by leveraging BNN specificities. Building upon similar techniques to the ones used by Anderson et al. [2] who introduced strong MILP formulations for robustness verification of ReLU based networks, Han and Gómez [9] derive an ideal formulation of a polytope related to the output of a single neuron in BNNs (details are given in §3.1). Their work is concurrent with the one (we became aware of later) by Lyu and Huchette [18], the latter investigating also some extensions, such as handling zero weights (in addition to weights in $\mathbb{B}$). Han and Gómez's work [9] is the closest to and originally motivated ours: overall objective is to investigate MILP techniques and polyhedral structures to take into account correlations between the outputs of pairs of neurons in the same layer of a BNN.

3 Disjunctive programming based view to neuron pairs in BNNs

In this section, after recalling Han and Gómez results about single neuron convexification [9], we present a disjunctive programming based framework to deal with neuron pairs and also present specific families of facet defining inequalities for a polytope related to neuron pairs. Due to length restrictions, we focus on the case when the number of inputs and the number of neurons of each hidden layer in the BNN are even, i.e. n_k is even, for all $k \in \{0\} \cup [s]$. Given a set $X \subseteq \mathbb{R}^n$, $\text{conv}(X)$ stands for its convex hull.

3.1 Single neuron convexification

Let n denote a positive integer. We define the set

$$S_1 = \left\{ (\boldsymbol{y}, t) \in \mathbb{B}^n \times \mathbb{B} \colon t = \text{sign}\left(\sum_{i=1}^{n} y_i \right) \right\}$$

corresponding to all the possible input/output pairs of the sign function with n binarized inputs and all the weights of value one on the inputs. Note that assuming all the weights have value one instead of possibly different values in $\mathbb{B}$ is with no loss of generality (resorting to simple substitutions). Han and Gómez determined a complete description of $\text{conv}(S_1)$

Theorem 1. *[9] If n is even, then the convex hull of S_1 is given by*

$$\frac{n}{2}(t-1) \leq \sum_{i=1}^{n} \min\{y_i, t\} \tag{2a}$$

$$\sum_{i=1}^{n} \max\{y_i, t\} \leq -2 + (n+2)\frac{t+1}{2} \tag{2b}$$

$$(\boldsymbol{y}, t) \in [-1, 1]^{n+1} \tag{2c}$$

Although the number of *linear* inequalities corresponding to (2) is exponential in n, the separation problem can be solved efficiently (in linear time). These inequalities were used in [9] to strengthen a linear relaxation of (1). However, they do not account for potential correlations between the outputs of different neurons. We introduce hereafter a framework aiming at the generation of inequalities taking into account such correlations to strengthen relaxations of (1).

3.2 Disjunctive programming based approach for neuron pairs

Consider the following set defined similar to S_1 but for neuron pairs.

$$S_2 = \left\{ (\boldsymbol{y}, t_1, t_2) \in \mathbb{B}^n \times \mathbb{B} \times \mathbb{B} \colon \begin{array}{l} t_1 = \text{sign}\left(\sum_{i=1}^{n} y_i\right) \\ t_2 = \text{sign}\left(\left(\sum_{i=1}^{k} y_i\right) - \left(\sum_{i=k+1}^{n} y_i\right)\right) \end{array} \right\} \tag{3}$$

where $k \in \{0\} \cup [n]$. Note that any set of the form

$$\left\{ (\boldsymbol{y}, t_1, t_2) \in \mathbb{B}^n \times \mathbb{B} \times \mathbb{B} \colon \begin{array}{l} t_1 = \operatorname{sign}\left(\sum_{i=1}^n w_{1,i} y_i\right) \\ t_2 = \operatorname{sign}\left(\sum_{i=1}^n w_{2,i} y_i\right) \end{array} \right\}$$

with weights $w_{q,j} \in \mathbb{B}$ for all $q \in \{1, 2\}$ and $j \in [n]$, can be represented as (3) (resorting to substitutions and a reordering of the variable indices).

Computational experiments on small instances illustrate the fact that the size of an ideal description of S_2 may be significantly much larger (in terms of the number of inequalities) than S_1, at least when restricted to the original space of variables (see Table 1).

Table 1. Number of facet-defining inequalities of $\operatorname{conv}(S_2)$ for $n \in \{4, 6, 8, 10\}$ and $k \in \{0\} \cup [n]$ obtained with PORTA [6]. The number of facets of $\operatorname{conv}(S_1)$ for $n = 4, 6, 8, 10$ is $26, 78, 274, 1046$, respectively.

n \ k	0	1	2	3	4	5	6	7	8	9	10
10	1299	2191	10793	11455	25485	27276	21769	19107	3300	1092	1046
8	345	555	1799	2097	2923	2650	857	292	274		
6	99	149	311	337	223	87	78				
4	33	39	35	21	26						

This may suggest that families of inequalities that are valid for $\operatorname{conv}(S_2)$ could contribute to strengthen further relaxations of (1) already including (2). So far we could not derive a complete description of $\operatorname{conv}(S_2)$ and we alternatively present hereafter a disjunctive programming based approach to solve the separation problem w.r.t. $\operatorname{conv}(S_2)$ in polynomial time. (Given a polyhedron $P \subseteq \mathbb{R}^n$ and a point $\boldsymbol{x} \in \mathbb{R}^n$, the corresponding separation problem is to determine whether $\boldsymbol{x} \in P$, and, if not, to find an inequality that is valid for P and violated by $\boldsymbol{x}$.) The proposed method is based on complete descriptions of polytopes derived from disjunctions of S_2 using the following inequalities.

$$\sum_{i=1}^{n} y_i \geq 0 \tag{t_1+}$$

$$\sum_{i=1}^{n} y_i \leq -2 \tag{t_1-}$$

$$\left(\sum_{i=1}^{k} y_i\right) - \left(\sum_{i=k+1}^{n} y_i\right) \geq 0 \tag{t_2+}$$

$$\left(\sum_{i=1}^{k} y_i\right) - \left(\sum_{i=k+1}^{n} y_i\right) \leq -2 \tag{t_2-}$$

Inequalities (t_1+) and (t_1-) (resp. (t_2+) and (t_2-)) lead to a disjunction of S_2 w.r.t. the sign of t_1 (resp. t_2). Note that, due to the assumed even parity of n, the right-hand side can be decreased from -1 to -2 in (t_1-) and (t_2-). The four inequalities above lead us to consider the sets

$$Z_{\bullet\blacktriangleright} = \left\{ (\boldsymbol{y}, t_1, t_2) \in \mathbb{B}^n \times \mathbb{B} \times \mathbb{B} \colon \begin{array}{l} \boldsymbol{y} \text{ satisfies inequalities } (t_1\bullet) \text{ and } (t_2\,\blacktriangleright) \\ t_1 = \operatorname{sign}\left(\sum_{i=1}^{n} y_i\right) \\ t_2 = \operatorname{sign}\left(\left(\sum_{i=1}^{k} y_i\right) - \left(\sum_{i=k+1}^{n} y_i\right)\right) \end{array} \right\}$$

with $\bullet, \blacktriangleright \in \{+,-\}$. An important property which was used in [9] to get an ideal description of $\operatorname{conv}(S_1)$ was the fact that the matrix corresponding to the system of constraints composed of $-1 \le t \le 1$, $-\mathbf{1} \le \boldsymbol{y} \le \mathbf{1}$ and (t_1+) or (t_1-) is totally unimodular. This, however, no longer holds if either inequality (t_2+) or (t_2-) is added to such a system. Anyhow, we can show that a simple ideal description can still be determined for all of the above sets $Z_{\bullet\blacktriangleright}$.

Proposition 1. *Assuming n is even the following holds.*

$$\operatorname{conv}(Z_{++}) = \left\{ (\boldsymbol{y}, 1, 1) : \begin{array}{l} (t_1+), (t_2+), \boldsymbol{y} \in [-1,1]^n \\ \sum_{i=1}^{k} y_i \ge 1 \textit{ if } k \textit{ is odd} \end{array} \right\}$$

$$\operatorname{conv}(Z_{+-}) = \left\{ (\boldsymbol{y}, 1, -1) : \begin{array}{l} (t_1+), (t_2-), \boldsymbol{y} \in [-1,1]^n \\ \sum_{i=k+1}^{n} y_i \ge 2 \textit{ if } k \textit{ is even} \end{array} \right\}$$

$$\operatorname{conv}(Z_{-+}) = \left\{ (\boldsymbol{y}, -1, 1) : \begin{array}{l} (t_1-), (t_2+), \boldsymbol{y} \in [-1,1]^n \\ \sum_{i=k+1}^{n} y_i \le -2 \textit{ if } k \textit{ is even} \end{array} \right\}$$

$$\operatorname{conv}(Z_{--}) = \left\{ (\boldsymbol{y}, -1, -1) : \begin{array}{l} (t_1-), (t_2-), \boldsymbol{y} \in [-1,1]^n \\ \sum_{i=1}^{k} y_i \le -3 \textit{ if } k \textit{ is odd and } k \ge 3 \end{array} \right\}$$

Proof. We just report here the proof for $\operatorname{conv}(Z_{++})$ since it is similar for the other convex hulls. Let T denote the set

$$T = \left\{ (\boldsymbol{y}, 1, 1) : \begin{array}{l} (t_1+), (t_2+), \boldsymbol{y} \in [-1,1]^n \\ \sum_{i=1}^{k} y_i \ge 1 \text{ if } k \text{ is odd} \end{array} \right\}.$$

Firstly, we show $Z_{++} \subseteq T$ which implies then $\operatorname{conv}(Z_{++}) \subseteq T$ because T is convex. Let $(\overline{\boldsymbol{y}}, 1, 1) \in Z_{++}$. By definition, $\overline{\boldsymbol{y}}$ satisfies (t_1+) and (t_2+), implying $\sum_{i=1}^{k} \overline{y}_i \ge 0$. If k is odd, then, due to $\overline{\boldsymbol{y}} \in \mathbb{B}^n$, we have $\sum_{i=1}^{k} \overline{y}_i \ge 1$, and thus $(\overline{\boldsymbol{y}}, 1, 1) \in T$.

We now prove $T \subseteq \operatorname{conv}(Z_{++})$. Obviously, $T \cap \mathbb{B}^{n+2} \subseteq Z_{++}$. It is then sufficient to prove that all the extreme points of T belong to $\mathbb{B}^{n+2}$. Let $(\hat{\boldsymbol{y}}, 1, 1)$ denote an extreme point of T. Then $\hat{\boldsymbol{y}}$ must verify with equality n linearly independent inequalities from the system $\mathcal{S}$ composed of (t_1+),(t_2+), $-\mathbf{1} \le \boldsymbol{y} \le \mathbf{1}$, and, if k is odd: $\sum_{i=1}^{k} y_i \ge 1$. We can distinguish the following cases.

– Case 1: k is even, or k is odd with $\sum_{i=1}^{k} \hat{y}_i > 1$.
  - Subcase 1.1: $\mathbf{1}^\top \hat{\boldsymbol{y}} = 0$. Then $\hat{\boldsymbol{y}}$ is also an extreme point of the polyhedron $Q_{11} \subseteq \mathbb{R}^n$ defined by the system: $\mathbf{1}^\top \boldsymbol{y} = 0, \sum_{i=k+1}^{n} y_i \leq 0, -\mathbf{1} \leq \boldsymbol{y} \leq \mathbf{1}$, which is totally unimodular. Thus $\hat{\boldsymbol{y}}$ is integral, has at least $n-2$ entries in $\mathbb{B}$ and at most two zero entries. Since n is even and $\mathbf{1}^\top \hat{\boldsymbol{y}} = 0$, the number of zero entries cannot be odd. If $\hat{\boldsymbol{y}}$ has two zero entries: $\hat{y}_p = \hat{y}_q = 0$ with $(p,q) \in [n]^2$, $p \neq q$, then necessarily $\sum_{i=k+1}^{n} \hat{y}_i = 0$ and (using our assumptions on k with $\mathbf{1}^\top \hat{\boldsymbol{y}} = 0$), k must be even and either $\{p,q\} \subseteq [k]$ or $\{p,q\} \subseteq \{k+1,\ldots,n\}$. Now, let $\hat{\boldsymbol{y}}^{\mathbf{1}} = \hat{\boldsymbol{y}} + \epsilon\,(\boldsymbol{e_p} - \boldsymbol{e_q})$ and $\hat{\boldsymbol{y}}^{\mathbf{2}} = \hat{\boldsymbol{y}} - \epsilon\,(\boldsymbol{e_p} - \boldsymbol{e_q})$, where $\boldsymbol{e_i}$ stands for the i^{th} unit vector and $\epsilon \in]0, \frac{1}{2}[$. We can check that $\hat{\boldsymbol{y}}^{\mathbf{1}}$ and $\hat{\boldsymbol{y}}^{\mathbf{2}}$ belong to Q_{11} and $\hat{\boldsymbol{y}} = \frac{1}{2}\left(\hat{\boldsymbol{y}}^{\mathbf{1}} + \hat{\boldsymbol{y}}^{\mathbf{2}}\right)$, i.e. a contradiction with $\hat{\boldsymbol{y}}$ being an extreme point of Q_{11}. Consequently, $\hat{\boldsymbol{y}}$ has no zero entries and $\hat{\boldsymbol{y}} \in \mathbb{B}^n$.
  - Subcase 1.2: $\mathbf{1}^\top \hat{\boldsymbol{y}} > 0$. Then $\hat{\boldsymbol{y}}$ is an extreme point of the polyhedron defined by the system: (t_2+), $-\mathbf{1} \leq \boldsymbol{y} \leq \mathbf{1}$, which is totally unimodular. Thus, $\hat{\boldsymbol{y}}$ is integral with at least $n-1$ entries in $\mathbb{B}$ and at most one entry with value zero. If $\hat{\boldsymbol{y}}$ has one entry equal to zero, then (t_2+) must be verified with equality, which is not possible because n is assumed to be even. So, $\hat{\boldsymbol{y}} \in \mathbb{B}^n$.

– Case 2: k is odd and $\sum_{i=1}^{k} \hat{y}_i = 1$.
  - Subcase 2.1: $\mathbf{1}^\top \hat{\boldsymbol{y}} = 0$. Then $\hat{\boldsymbol{y}}$ is also an extreme point of the polyhedron Q_{21} defined by the system: $\sum_{i=1}^{k} y_i = 1$, $\sum_{i=k+1}^{n} y_i = -1$, $-\mathbf{1} \leq \boldsymbol{y} \leq \mathbf{1}$. Since the matrix defining this system is totally unimodular we can deduce that $\hat{\boldsymbol{y}}$ is integral with at least $n-2$ entries in $\mathbb{B}$ and at most 2 entries with value zero. Since $\mathbf{1}^\top \hat{\boldsymbol{y}} = 0$ and n is even, the number of zero entries must be even. If $\hat{\boldsymbol{y}}$ has two zero entries $\hat{y}_p = \hat{y}_q = 0$, with $(p,q) \in [n]^2$, $p \neq q$, then the equations in the definition of Q_{21} imply that either $\{p,q\} \subseteq [k]$ or $\{p,q\} \subseteq \{k+1,\ldots,n\}$. But either case leads to a contradiction, similar to Subcase 1.1 above. Thus, $\hat{\boldsymbol{y}} \in \mathbb{B}^n$.
  - Subcase 2.2: $\mathbf{1}^\top \hat{\boldsymbol{y}} > 0$. Then, $\hat{\boldsymbol{y}}$ is an extreme point of the polyhedron Q_{22} defined by the system: $\sum_{i=1}^{k} y_i = 1$, $\sum_{i=k+1}^{n} y_i \leq 1$, $-\mathbf{1} \leq \boldsymbol{y} \leq \mathbf{1}$, which is totally unimodular. We can deduce that $\hat{\boldsymbol{y}}$ is integral with at least $n-2$ entries in $\mathbb{B}$ and at most 2 entries with value zero. Using the odd parity of k, we can deduce that $\hat{\boldsymbol{y}}$ cannot have exactly one entry with value zero. If $\hat{\boldsymbol{y}}$ has two zero entries $\hat{y}_p = \hat{y}_q = 0$, with $(p,q) \in [n]^2$, $p \neq q$, then the equation in the definition of Q_{22} implies that either $\{p,q\} \subseteq [k]$ or $\{p,q\} \subseteq \{k+1,\ldots,n\}$. The rest of the proof is similar to Subcase 1.1 above.

□

Note that $\text{conv}\,(S_2) = \text{conv}\left(\cup_{\bullet,\blacktriangleright \in \{+,-\}} \text{conv}\,(Z_{\bullet\blacktriangleright})\right) = \text{conv}\left(\cup_{\bullet,\blacktriangleright \in \{+,-\}} Z_{\bullet\blacktriangleright}\right)$. Thus, the derivation of an extended formulation of $\text{conv}\,(S_2)$ from Proposition 1 with disjunctive programming techniques is straightforward. The latter can be used to design a separation procedure w.r.t. $\text{conv}\,(S_2)$.

3.3 Separation procedure w.r.t. conv (S_2)

We describe hereafter a generic procedure to solve the separation problem w.r.t. the convex hull of a finite number of nonempty polytopes and that we used in our experiments w.r.t. conv (S_2).

Let $(\mathcal{P}_i)_{i=1}^{\ell}$ denote a finite family of polytopes in $\mathbb{R}^n$ with

$$\mathcal{P}_i = \{\boldsymbol{x} \in \mathbb{R}^n \colon \boldsymbol{A_i x} \leq \boldsymbol{b_i}\} \text{ for all } i \in [\ell],$$

with $\boldsymbol{A_i} \in \mathbb{R}^{m_i \times n}$, $\boldsymbol{b_i} \in \mathbb{R}^{m_i}$, $m_i \in \mathbb{N}$. Let $\mathcal{P}$ stand for the convex hull of the union of the polytopes $(\mathcal{P}_i)_{i=1}^{\ell}$: $\mathcal{P} = \text{conv}\left(\cup_{i=1}^{\ell} \mathcal{P}_i\right)$. The problem of determining whether some given point $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ belongs to $\mathcal{P}$ reduces to solving the linear program

$$(\texttt{SEP}) \begin{cases} \min_{\boldsymbol{\lambda}, \boldsymbol{y}, \boldsymbol{\epsilon}^+, \boldsymbol{\epsilon}^-} \sum_{i=1}^{n} \epsilon_i^+ + \epsilon_i^- \\ \hat{\boldsymbol{x}} = \sum_{i=1}^{\ell} \boldsymbol{y_i} + \boldsymbol{\epsilon}^+ - \boldsymbol{\epsilon}^- \\ \boldsymbol{A_i y_i} \leq \lambda_i \boldsymbol{b_i}, i = 1, \ldots, \ell \\ \sum_{i=1}^{\ell} \lambda_i = 1 \\ \boldsymbol{\lambda} \in \mathbb{R}_+^{\ell}, \boldsymbol{\epsilon}^+, \boldsymbol{\epsilon}^- \in \mathbb{R}_+^n, \\ \boldsymbol{y_i} \in \mathbb{R}^{m_i}, i = 1, \ldots, \ell. \end{cases}$$

One can check that $\hat{\boldsymbol{x}} \in \mathcal{P}$ holds if and only if the optimal objective value of (SEP) is zero. Consider then the dual problem:

$$(\texttt{DSEP}) \begin{cases} \max_{\boldsymbol{c}, \boldsymbol{\alpha}, \gamma} \boldsymbol{c}^\top \hat{\boldsymbol{x}} + \gamma \\ \boldsymbol{c} = \boldsymbol{A_i}^\top \boldsymbol{\alpha_i}, \forall i \in [\ell] \\ \boldsymbol{\alpha_i}^\top \boldsymbol{b_i} + \gamma \leq 0, \forall i \in [\ell] \\ \boldsymbol{c} \in [-1; 1]^n, \gamma \in \mathbb{R}, \boldsymbol{\alpha_i} \in \mathbb{R}_+^{m_i}. \end{cases}$$

Note that for any feasible solution $(\boldsymbol{c}, \boldsymbol{\alpha}, \gamma)$ of (DSEP) the inequality $\boldsymbol{c}^\top \boldsymbol{x} \leq -\gamma$ is valid for $\mathcal{P}_i, \forall i \in [\ell]$. Let $Z^{\star}_{\texttt{DSEP}}$ denote the optimal objective of (DSEP). Using strong duality in linear programming, the separation problem w.r.t. $\mathcal{P}$ reduces to solving (DSEP): either $Z^{\star}_{\texttt{DSEP}} = 0$ and in that case $\hat{\boldsymbol{x}} \in \mathcal{P}$. Otherwise $Z^{\star}_{\texttt{DSEP}} > 0$ and a violated inequality is given by an optimal solution $(\overline{\boldsymbol{c}}, \overline{\boldsymbol{\alpha}}, \overline{\gamma})$ of (DSEP): $\overline{\boldsymbol{c}}^\top \hat{\boldsymbol{x}} > -\gamma$. Taking for $\mathcal{P}_i$ the polytopes $\text{conv}\,(Z_{\bullet\blacktriangleright})$ with $\bullet, \blacktriangleright \in \{+, -\}$, the approach described above leads to the next result.

Proposition 2. *The separation w.r.t.* conv (S_2) *can be solved in polynomial time.*

3.4 Facet defining inequalities

Designing a cutting-plane algorithm based on the separation procedure described above (§3.3) to generate constraints can lead to poor performance in terms of computational time, as we could observe in preliminary experiments. This led us to consider the alternative of proceeding to the separation over specific families of inequalities. We introduce hereafter four families of inequalities stemming from studies based on the framework described in §3.2-3.3 and that we used in our

experiments. We also provide sufficient conditions for these inequalities to be facet defining and study the corresponding separation problem. In what follows, we assume $n \geq 4$, n even.

Proposition 3. *The following inequalities, together with the specified conditions on the set I are valid for* $\operatorname{conv}(S_2)$.

$$\left(|I| - \left\lceil \frac{k-1}{2} \right\rceil\right)(t_1 + t_2) - \sum_{i \in I} y_i \leq |I|, I \subseteq [k], |I| \geq \left\lceil \frac{k-1}{2} \right\rceil \tag{5}$$

$$\left(\left\lceil \frac{k-3}{2} \right\rceil - |I|\right)(t_1 + t_2) + \sum_{i \in I} y_i \leq |I|, I \subseteq [k], |I| \geq \left\lceil \frac{k-3}{2} \right\rceil \tag{6}$$

$$\left(|I| - \left\lceil \frac{n-k-2}{2} \right\rceil\right)(t_1 - t_2) - \sum_{i \in I} y_i \leq |I|, I \subseteq [n] \backslash [k], |I| \geq \left\lceil \frac{n-k-2}{2} \right\rceil \tag{7}$$

$$\left(\left\lceil \frac{n-k-2}{2} \right\rceil - |I|\right)(t_1 - t_2) + \sum_{i \in I} y_i \leq |I|, I \subseteq [n] \backslash [k], |I| \geq \left\lceil \frac{n-k-2}{2} \right\rceil \tag{8}$$

Proposition 4. *The following properties hold.*

(i) Assume $k < \frac{n}{2}$, and let $I \subseteq [k]$ such that $|I| > \lceil \frac{k-1}{2} \rceil$. Then (5) is facet defining for $\operatorname{conv}(S_2)$.

(ii) Assume $3 \leq k < \frac{n}{2}$, and let $I \subseteq [k]$ such that $|I| > \lceil \frac{k-3}{2} \rceil$. Then (6) is facet defining for $\operatorname{conv}(S_2)$.

(iii) Assume $\frac{n}{2} + 1 < k \leq n - 2$, and let $I \subseteq [n] \setminus [k]$ such that $|I| > \lceil \frac{n-k-2}{2} \rceil$. Also assume $k < n - 2$ if $|I| > 1$. Then (7)-(8) are facet defining for $\operatorname{conv}(S_2)$.

Proposition 5. *The separation problem w.r.t. (5)-(8) can be solved in polynomial time.*

4 Computational experiments

In this section we provide preliminary computational results to assess the performance of constraint generation procedures relying on results from [9] and the families of inequalities (5)-(8) to verify robustness of BNNs.

4.1 Evaluated methods and setup

We consider solving (1) with a constraint generation algorithm, starting with the relaxation:

$$\max_{\boldsymbol{x^{s+1}}, \boldsymbol{y^k}} x_t^{s+1} - x_\ell^{s+1} \tag{9a}$$

$$\text{s.t. } \boldsymbol{x^{s+1}} = \boldsymbol{W^{s+1}} \boldsymbol{y^s} \tag{9b}$$

$$\|\boldsymbol{y^0} - \boldsymbol{z}\|_1 \leq \epsilon \tag{9c}$$

$$\boldsymbol{x^{s+1}} \in \mathbb{R}^{n_{s+1}} \tag{9d}$$

$$\boldsymbol{y^k} \in \mathbb{B}^{n_k}, k \in \{0\} \cup [s] \tag{9e}$$

for some fixed value $\epsilon > 0$. Starting with (9), two options to generate constraints at each iteration are considered:

- `single`: for each inequality of type (2a) and (2b), we check if one is violated, and if so, one with largest violation is generated (for each type).
- `approx`: in addition to the procedure `single`, for each pair of neurons and for each of the four types of inequalities (5)-(8), we check if one is violated, and if so, one with largest violation is generated (for each type).

Due to length restrictions we only report results for three configurations of BNNs: 64×2, 128×2 and 256×1, where the first number denotes the number of neurons per hidden layer (common to all hidden layers) and the second number is the number of layers. Each BNN has 784 inputs and 10 outputs (each one corresponding to a digit). The BNNs have been trained on the MNIST dataset as described in [7], using the methodology from [11]. The training process of BNNs was conducted on the DAS-5 cluster [3]. All networks have been trained to an accuracy rate of approximately 75%. 25 images from the MNIST dataset are used when performing robustness verification, and the target class is always selected so that it differs from the predicted class. The reported results are always averaged over this set of instances. The reported results were obtained using a computer with an Apple M1 processor and 8GB of RAM. Gurobi 10 (with default options) is used to solve the optimization problems.

4.2 Computational results

We first evaluate the efficiency of the contraint generation methods to determine (with certainty) the robustness status of a BNN, while restricting the number of iterations to 20 and considering different values for the parameter ϵ defining the neighborhood in (9): $\epsilon \in \{11, 12, \ldots, 20\}$. By one *iteration* we mean the application of the separation procedures for each neuron (w.r.t. (2a)-(2b)), and also for each pair of neurons in the case of `approx` (w.r.t. (5)-(8)).

Figure 1 (resp. 2) displays the verification accuracy, i.e. the proportion of images for which the robustness status could be settled depending on ϵ (resp. the objective value after 20 iterations).

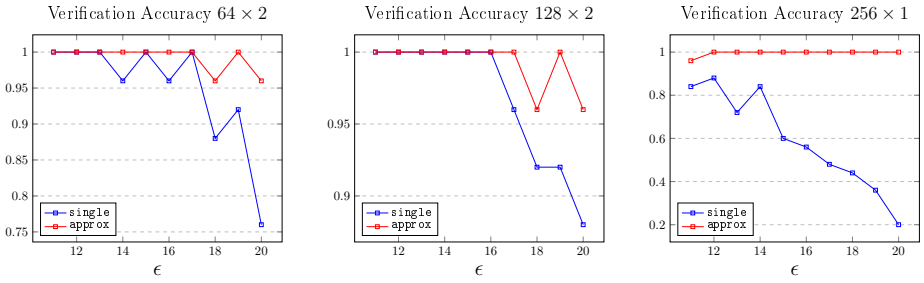

Fig. 1. Verification accuracy

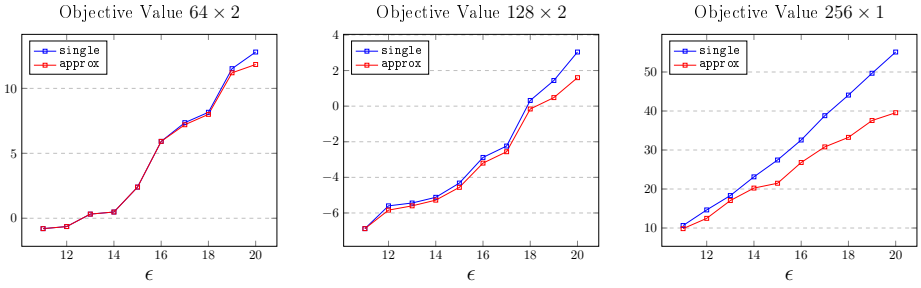

Fig. 2. Objective value after 20 iterations (starting with (9))

The method `approx` clearly improves the verification abilities of the solver for the considered configurations of BNNs. Its overall verification accuracy - averaged over all epsilons and networks - equals 99.3%, whereas the single method verifies 86.1% with especially poor performance for 256×1 BNNs. It is important to notice that when starting with the formulation (9) and keeping the integrality constraints the used integer programming solver may add many cuts (such as Gomory cuts). In order to better assess the potential improvement of `approx` over `single`(i.e. independently of cuts added by the solver), in what follows we report results obtained by relaxing the integrality contraints of (9).

Another observation from experiments we carried out is that `approx` may be much more time consuming than `single`. This led us to investigate an alternative constraint generation strategy denoted by `approx-`q with $q \in \{1, 5\}$. It differs from `approx` by the fact that it generates at most q inequalities per neuron and per iteration. We report in Figures and the evolution of the objective value of the continuous relaxation depending on the number of iterations and time respectively, within the limit of 20 iterations and for $\epsilon = 5$. The ratio of the

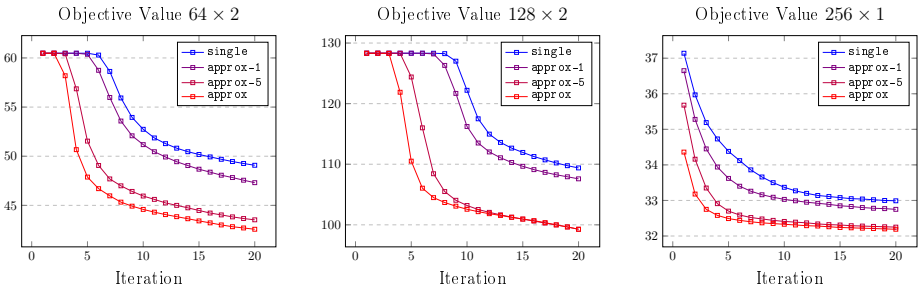

Fig. 3. Evolution of objective value of continuous relaxation (starting with (9)) w.r.t. the number of iterations

number of cuts added compared with `single` is in the following ranges: $[1.3, 1.57]$ for `approx-1`, $[2.45, 2.5]$ for `approx-5` and $[5.6, 8.15]$ for `approx`. The objective value obtained wth `approx-5` within 20 iterations tends to be close to `approx` but with fewer cuts added. On the other hand, it seems that `approx-1` delivers only a slight improvement over `single`, and this is even more stressed for deeper

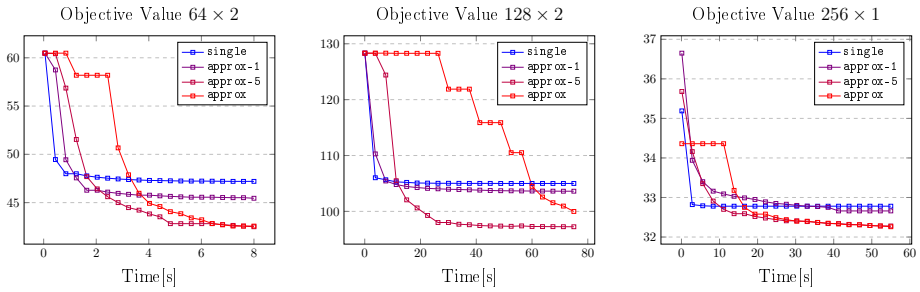

Fig. 4. Evolution of objective value of continuous relaxation (starting with (9)) w.r.t. computation time

networks (i.e. instances with two hidden layers for the results reported here). `single` or `approx-1` appear to converge much faster than the other methods but they are not able to reach the bounds of the same quality.

5 Conclusion

In this paper we addressed a robustness verification problem for BNNs via a constraint generation algorithm. We namely introduced a constraint generation framework relying on disjunctive programming and complete descriptions established for polytopes defining a special disjunction related to the outputs of neuron pairs. Considering the limitations of the latter approach due to high computation times we proposed an alternative constraint generation algorithm relying on specific families of facet defining inequalities. Our preliminary computational results illustrate improvements in terms of verification accuracy over recent convexification results for a single neuron. Ongoing research is directed towards alternative constraint generation strategies and further polyhedral studies related to the outputs of two or more neurons.

References

1. Amir, G., Wu, H., Barrett, C., Katz, G.: An SMT-based approach for verifying binarized neural networks. In: Groote, J.F., Larsen, K.G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 203–222. Springer International Publishing, Cham (2021)
2. Anderson, R., Huchette, J., Ma, W., , Tjandraatmadja, C., Vielma, J.P.: Strong mixed-integer programming formulations for trained neural networks. Mathematical Programming **183**, 3–39 (2020)
3. Bal, H., Epema, D., de Laat, C., van Nieuwpoort, R., Romein, J., Seinstra, F., Snoek, C., Wijshoff, H.: A medium-scale distributed system for computer science research: Infrastructure for the long term. Computer **49**(5), 54–63 (2016)
4. Bunel, R., Turkaslan, I., Torr, P.H.S., Kohli, P., Mudigonda, P.K.: A unified view of piecewise linear neural network verification. In: Neural Information Processing Systems (2017)

5. Cheng, C., Nührenberg, G., Ruess, H.: Verification of binarized neural networks. CoRR **abs/1710.03107** (2017), http://arxiv.org/abs/1710.03107
6. Christof, T., Löbel, A.: Porta - polyhedron representation transformation algorithm. Available at https://porta.zib.de/
7. Deng, L.: The MNIST database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)
8. Goldberg, Y.: A primer on neural network models for natural language processing. Journal of Artificial Intelligence Research **57**, 345–420 (2016)
9. Han, S., Gómez, A.: Single-neuron convexifications for binarized neural networks, University of Southern California (2021). Available at https://optimization-online.org/?p=17148
10. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**(5), 359–366 (1989)
11. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 29. Curran Associates, Inc. (2016)
12. Jia, K., Rinard, M.C.: Efficient exact verification of binarized neural networks. CoRR **abs/2005.03597** (2020), https://arxiv.org/abs/2005.03597
13. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: a calculus for reasoning about deep neural networks. Formal Methods in System Design **60**, 87–116 (2022), https://doi.org/10.1007/s10703-021-00363-7
14. Khalil, E.B., Gupta, A., Dilkina, B.: Combinatorial attacks on binarized neural networks. In: International Conference on Learning Representations (ICLR) (2019)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012)
16. Lin, W., Yang, Z., Chen, X., Zhao, Q., Li, X., Liu, Z., He, J.: Robustness verification of classification deep neural networks via linear programming. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11410–11419 (2019)
17. Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., Kochenderfer, M.J.: Algorithms for Verifying Deep Neural Networks (2021)
18. Lyu, B., Huchette, J.: Verifying binarized neural networks: Convex relaxations, mixed-integer programming, and consistency, https://bochuanbob.github.io/BNN_MIP.pdf
19. Narodytska, N., Kasiviswanathan, S., Ryzhyk, L., Sagiv, M., Walsh, T.: Verifying properties of binarized deep neural networks. In: AAAI Conference on Artificial Intelligence (AAAI) (2018)
20. Narodytska, N., Zhang, H., Gupta, A., Walsh, T.: In search for a sat-friendly binarized neural network architecture. In: International Conference on Learning Representations (ICLR) (2020)
21. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (ICLR) (2014)