Verifying binarized neural networks: Convex relaxations, mixed-integer programming, and consistency

Bochuan Lyu and Joey Huchette*

Department of Computational Applied Mathematics and Operations Research, Rice University
{bl46,joehuchette}@rice.edu

Abstract. We study algorithms for the verification of binarized neural networks. Our main technical contribution is an explicit convex hull description for the output of each neuron in the network, given appropriate bound information (variable bounds and/or integrality) about its multivariate inputs. This description offers the tightest possible convex relaxation for a single neuron in the network and can be easily modified to produce a mixed-integer programming (MIP) formulation for exact verification. However, we illustrate through examples and computations that, due to the structure of the binarized activation function, convex relaxations for verification are intrinsically weak. To alleviate this computationally, we develop a new separation rule for cutting planes that, rather than focusing on improving dual bounds, are instead intended to reduce backtracking. Along the way, we model the verification problem with discrete-valued inputs (e.g. coming from image data) in a slightly different way that is without loss of generality and completely sidesteps the discontinuity problem, without which we may encounter spurious adversarial examples. Finally, we present computational results on standard image classification networks which 1) validate our observations that convex relaxations are poor and standard cutting planes do not strengthen the relaxation meaningfully, but that 2) consistency cuts can nevertheless improve convergence and solve time.[1]

Keywords: Deep Learning · Verification · Mixed-integer Programming.

1 Introduction

Deep neural networks (DNNs) have achieved resounding successes in a wide range of predictive tasks in areas such as image recognition, machine translation, and speech recognition [23]. However, state-of-the-art models often require an enormous amount of storage space and computational power. As a result, it is

* A portion of this work was completed while the author was at Google Research.

[1] We worked on this project mainly in 2020. There is another study with similar results [15]. This draft was written contemporaneously.

often difficult or impossible to deploy trained models in resource-constrained environments like embedded or "smart" devices. To mitigate this, researchers have begun exploring alternative DNN architectures that are much more memory- and compute-efficient while still (hopefully) being capable of high performance. Popular choices include binarized neural networks (BNNs) [17, 33] and ternarized weight networks (TWNs) [16, 25], where the weights and/or the activation values are restricted to take values in $\{-1, +1\}$ and $\{-1, 0, +1\}$, respectively, as well as more general quantized neural networks [18]. Our work will focus on binary activation ternary weight networks (BATWNs), a generalization of BNNs, in which each weight of the network is in $\{-1, 0, +1\}$, and the activations take values in $\{-1, +1\}$.

Due to their exceptional performance in many areas, deep learning systems are rapidly deployed in settings such as self-driving vehicles and facial recognition where failure or malfunction can lead to catastrophic outcomes. As a result, verifying the security and robustness of the deep learning model before deployment is essential [29]. Unfortunately, it has been widely recognized that neural networks can be quite brittle and susceptible to adversarial attacks, wherein very small, targeted changes can radically change the models' behavior [7, 32, 39, 47]. A number of algorithms for either producing adversarial attacks or proving robustness against such attacks have been presented over the intervening years, many based on convex relaxations [38, 40, 43–45, 49], mixed-integer programming [2, 8, 12, 13, 28, 34, 36, 41, 46], or satisfiability and constraint programming methods [1,3–6,19,20,27,31,35,37]; see Liu et al. [26] for a recent survey. Though this research area has been extremely active, most work is targeted or specialized for "standard" neural network architectures, with only a relatively small amount of recent work on verification of binarized networks [1, 19, 21, 30, 31] which are mostly satisfiability (SAT) and constraint programming methods. However, as shown in [30], MIP formulation of binarized networks can be solved in a similar amount of instances as SAT encoding within a limit time and with less average solved time. Also, MIP formulation can be used more straightforwardly in some more scalable relaxed certifiers as [40].

1.1 Contribution

Our contributions can be summarized as follows.

- An explicit description for the tightest possible convex relaxation for each neuron in a BATWN. We provide an explicit inequality description for the set coupling together the output of a single binarized activation and its multivariate input. We present a series of results specialized for different structural information known about the domain of inputs (e.g. variable bounds and/or integrality) that is applicable in different parts of the network. This convex hull result can be used directly as a relaxation for the verification task, or straightforwardly transformed into an "ideal" mixed-integer programming formulation that offers exact verification.

- An illustration that the convex hull is weak, and an algorithmic way to use it anyway. We illustrate, through example and computationally, that convex relaxations offer weak bounds for binarized neural network verification. However, we show that our convex hull description readily provides a large family of consistency cuts that can still be used in a branch-and-bound framework to reduce search tree size.
- A corrective for discrete inputs to avoid spurious adversarial examples. We show that, if the discontinuity in the binarized activation function is not correctly handled, MIP-based methods may potentially yield examples that are falsely labeled as adversarial. We provide a simple fix in the common setting where the input is discrete-valued (e.g. images with finite-bit representations).
- Computational validation. We close with a computational study that illustrates the utility of our approach. We observe that convex relaxations are weak, and that standard cutting planes do not offer much computational improvement. However, our new consistency cuts, separated inside the search tree, can decrease the runtime of a state-of-the-art MIP solver by 12-23% and reduce the size of the search tree by 14-36%, on average.

2 Formulations for BATWNs

We consider BATWNs of the form $\texttt{NN} : X \subseteq \mathbb{R}^{m_0} \to \mathbb{R}^{m_r}$, where r is the number of layers in the network. Then, the output of the network, $x^r = \texttt{NN}(x^0)$, is given by the recursive definition

$$x_i^j = \operatorname{sign}\left(b_i^j + \sum_{k \in I_{j,i}^+} x_k^{j-1} - \sum_{k \in I_{j,i}^-} x_k^{j-1}\right) \quad \forall j \in [r-1], i \in [m_j], \tag{1a}$$

$$x_i^r = b_i^r + \sum_{k \in I_{r,i}^+} x_k^{r-1} - \sum_{k \in I_{r,i}^-} x_k^{r-1} \qquad \forall i \in [m_r], \tag{1b}$$

where $\operatorname{sign}(v) := -1 + \mathbb{1}(v \geq 0) \in \{-1, +1\}$, $[T] := \{1, 2, \ldots, T\}$, and for each layer j, m_j is the number of neurons, b^j is the bias term[2], and $I_{j,i}^+$ and $I_{j,i}^-$ are disjoint subsets of $[m_{j-1}]$ learned in the training process. Each equation in (1) models a single neuron in the BATWN. Note that often the outputs x^r will correspond to logits, or unnormalized probabilities, which may be normalized through a softmax layer that we do not consider here. Also, note that this representation can be readily generalized to handle more complex architectures with, e.g., convolution layers.

In order to model the function NN, it is sufficient to study the graph of an individual neuron appearing in the definition (1):

$$S(I^+, I^-, D, b) := \left\{(x, y) \in D \times \mathbb{R} : y = \operatorname{sign}\left(b + \sum_{k \in I^+} x_k - \sum_{k \in I^-} x_k\right)\right\}.$$

[2] Note that batch normalization layers can be modeled via the bias terms.

Here, D is some domain for the inputs to this neuron; we will consider different possibilities later. Note that $S(I^+, I^-, D, b)$ may not be closed for certain choices of D due to the discontinuity of the sign function. In the remainder, we will simplify notation and simply use S when the arguments are clear from the context.

We are now prepared to state our central technical result. Notationally, take $\operatorname{supp}(w) \coloneqq \{i \in [n] : w_i \neq 0\}$. For $L, U \in \mathbb{R}^n$ with $L < U$ componentwise, take $\breve{L}_i = L_i + (U_i - L_i)\mathbb{1}[w_i < 0]$ and $\breve{U}_i = U_i + (L_i - U_i)\mathbb{1}[w_i < 0]$ for each $i \in [n]$.

Theorem 1. *Take constants $w \in \mathbb{R}^n$ and $b^1, b^2 \in \mathbb{R}$ with $b^1 \geq b^2$. Consider a box domain $D = [L, U] \subset \mathbb{R}^n$, along with the two subsets of it, $P^1 = \{x \in D : w \cdot x + b^1 \leq 0\}$ and $P^2 = \{x \in D : w \cdot x + b^2 \geq 0\}$. Then $\operatorname{Conv}((P^1 \times \{-1\}) \cup (P^2 \times \{+1\}))$ is equal to the set of all $(x, y) \in D \times [-1, +1]$ satisfying:*

$$\sum_{i \in I} w_i x_i + \sum_{i \notin I} w_i \breve{U}_i + b^2 \geq \frac{1}{2}\left(\sum_{i \in I} w_i \breve{L}_i + \sum_{i \notin I} w_i \breve{U}_i + b^2\right)(1 - y), \forall I \subseteq \operatorname{supp}(w) \tag{2a}$$

$$\sum_{i \in I} w_i (\breve{U}_i - x_i) \geq \frac{1}{2}\left(\sum_{i \in I} w_i \breve{U}_i + \sum_{i \notin I} w_i \breve{L}_i + b^1\right)(1 - y), \forall I \subseteq \operatorname{supp}(w) \tag{2b}$$

Proof. The result is immediate if P^1 or P^2 is empty. Otherwise, an extended formulation [9, Theorem 4.39] describing $\operatorname{Conv}((P^1 \times \{0\}) \cup (P^2 \times \{1\}))$ is

$$\begin{gathered} x = x^1 + x^2, \quad z^1 + z^2 = 1, \quad z^1, z^2 \geq 0 \\ w \cdot x^1 + b^1 z^1 \leq 0, \quad w \cdot x^2 + b^2 z^2 \geq 0 \\ L z^1 \leq x^1 \leq U z^1, \quad L z^2 \leq x^2 \leq U z^2 \end{gathered}$$

We first use the equations to eliminate x^2 and z^2 and rename $x^1 \equiv \tilde{x}$ and $z^1 \equiv z$. Presume for the moment that $w_1 > 0$. Projecting out $\tilde{x}_1$ with Fourier-Motzkin elimination leaves us with all constraints not involving $\tilde{x}_1$, along with:

$$\begin{gathered} L_1 z \leq U_1 z, \quad x_1 \leq U_1, \quad x_1 \geq L_1, \quad L_1(1 - z) \leq U_1(1 - z) \\ w_1 L_1 z + \sum\nolimits_{i>1} w_i \tilde{x}_i + b^1 z \leq 0, \quad w_1(x_1 - U_1(1 - z)) + \sum\nolimits_{i>1} w_i \tilde{x}_i + b^1 z \leq 0 \\ w_1(x_1 - L_1 z) + \sum\nolimits_{i>1} w_i (x_i - \tilde{x}_i) + b^2(1 - z) \geq 0 \\ w_1 U_1(1 - z) + \sum\nolimits_{i>1} w_i (x_i - \tilde{x}_i) + b^2(1 - z) \geq 0. \end{gathered}$$

Those equations on the first line are implied by bounds on x_1 and z and $L \leq U$. We add the remaining four constraints to the system, repeat the Fourier-Motzkin procedure iteratively to eliminate the remaining $\tilde{x}$ variables, giving the desired system. To conclude, we return to the $w_i > 0$ assumption. If instead $w_i = 0$ for some i, the elimination procedure is trivial, giving only variable bounds on x_i.

If $w_i < 0$, introduce an additional variable $\hat{x}_i = -x_i$, eliminate $\hat{x}_i$ instead with corresponding $-w_i > 0$, and then use the equation to eliminate x_i afterwards. The transformation requires us to replace L_i with $\breve{L}_i$ and U_i with $\breve{U}_i$. □

This description requires an exponential number of inequalities; however, we show in Section 3.2 that the separation problem can be solved in linear time.

We emphasize that system (2) is a linear inequality system that offers the tightest convex relaxation for $(P^1 \times \{-1\}) \cup (P^2 \times \{+1\})$. It can also be easily converted into a MIP formulation that exactly models the set. A MIP formulation is ideal if the extreme points of its linear programming (LP) relaxation are integral, and it is sharp if its LP relaxation recovers the convex hull of the set being modeled. An ideal formulation is sharp (though the converse is not necessarily true), and both are highly beneficial to the computational performance [42].

Corollary 1 *The system* $\{(x, y, z) \in D \times [-1, 1] \times \{0, 1\} : (2),\ y = 2z - 1\}$ *is an ideal MIP formulation for* $(P^1 \times \{-1\}) \cup (P^2 \times \{+1\})$.

Proof. Constructing the convex hull of $P^1 \times \{-1\}) \cup (P^2 \times \{+1\})$ in Theorem 1 suffices for building an ideal formulation. □

In the remainder of the section, we will apply Theorem 1 to get convex hull descriptions for binarized neurons in the input ($j = 1$) and intermediate ($1 < j < r$) layers with the box domain X to the network.

Note that with the continuous input domain $D = [L, U]$ and $b^1 = b^2 = b$ in Theorem 1, the points $(x, -1)$ and $(x, 1)$ are both in $(P^1 \times \{0\}) \cup (P^2 \times \{1\})$ if $x \in D$ and $w^T x + b = 0$. We will discuss the drawback of using the continuous input domain in Example 1 of Section 2.2. Thus, we consider a discrete subset of $[L, U]$ as input domain instead in order to model the neurons exactly. [3]

2.1 Neurons in the intermediate layers

We will first consider the intermediate (i.e. $1 < j < r$) layers of a network. Since the inputs of each neuron in these layers are the outputs of sign functions, the input domain for neuron x_i^j is $D = \{-1, +1\}^{m_{j-1}}$.

Proposition 1 *Take* $b \in \mathbb{R}$ *and disjoint, nonempty subsets* $I^+, I^- \subseteq [n]$. *Define* $\gamma = \mathbb{1}[|I^+ \cup I^-| \text{ is odd}] \oplus \mathbb{1}[\lfloor b \rfloor \text{ is odd}]$, *where* $x \oplus y = \mathbb{1}[x \neq y]$ *be the exclusive or operation. Take* $\tau = \lfloor b \rfloor - \gamma$, *and* $\kappa = \lfloor b \rfloor + 2 - \gamma$. *If* $D = \{-1, +1\}^n$, *then* $\mathrm{Conv}(S(I^+, I^-, D, b))$ *is equal to the set of all* $(x, y) \in [-1, +1]^{n+1}$ *satisfying*

$$\sum_{i \in I \cap I^+} x_i - \sum_{i \in I \cap I^-} x_i \geq \frac{|I| - |I^+ \cup I^-| - \tau}{2}(1 + y) - \frac{|I|}{2}(1 - y), \forall I \subseteq I^+ \cup I^- \tag{4a}$$

$$\sum_{i \in I \cap I^+} x_i - \sum_{i \in I \cap I^-} x_i \leq \frac{|I|}{2}(1 + y) - \frac{|I| - |I^+ \cup I^-| + \kappa}{2}(1 - y), \forall I \subseteq I^+ \cup I^- \tag{4b}$$

[3] People will still relax the input domain X to be continuous in practice when applying MIP solver in favor of speeds, even if the input is discrete (e.g. image data) [2, 41].

Proof. Without loss of generality, presume that $\lfloor b \rfloor$ is even. Define

$$Q^1 = \left\{(x,y) \in \mathbb{R}^{n+1} : \sum\nolimits_{k\in I^+} x_k - \sum\nolimits_{k\in I^-} x_k + \kappa \le 0, -1 \le x \le 1, y = -1\right\}$$
$$Q^2 = \left\{(x,y) \in \mathbb{R}^{n+1} : \sum\nolimits_{k\in I^+} x_k - \sum\nolimits_{k\in I^-} x_k + \tau \ge 0, -1 \le x \le 1, y = 1\right\}.$$

Observe that there exists some $x \in D$ such that $\lfloor b \rfloor + \sum_{k\in I^+} x_k = \sum_{k\in I^-} x_k$ if and only if $|I^+ \cup I^-|$ is even. This fact, along with the integrality of D, allows us to conclude that $S = (Q^1 \cup Q^2) \cap (D \times \mathbb{R})$. The result will then follow if we can show that $\mathrm{Conv}(S) = \mathrm{Conv}(Q^1 \cup Q^2)$.

The $\subseteq$ direction is immediate since $D \subseteq \mathrm{Conv}(D)$, so focus on the $\supseteq$ direction. Consider w.l.o.g. Q^1. First, it is not hard to see that Q^1 is polyhedra. Thus, for any $(x,y) \in \mathrm{ext}(Q^1)$, (x,y) is also a basic feasible solution to Q^1. Since the set is n-dimensional, there are $(n+1)$ linearly independent active equations in Q^1. Assume that there exists $i \in (I^+ \cup I^-)$ such that $x_i \notin \{-1,+1\}$. Then, $x_j \in \{-1,+1\}$ for all $j \neq i$ and $\sum_{k\in I^+} x_k - \sum_{k\in I^-} x_k + \kappa = 0$. Since x_j and κ are all integers for any $j \neq i$, then x_i must be an integer and $x_i = 0$. However, κ has the different parity from $(|I^+ \cup I^-| - 1)$. If $x_i = 0$, then, $\sum_{k\in I^+} x_k - \sum_{k\in I^-} x_k + \kappa \neq 0$, which is a contradiction. Thus, $x \in D$ and $(x,y) \in S^1 := Q^1 \cap (D \times \mathbb{R})$. Hence, $\mathrm{ext}(Q^1) \subseteq S^1$ and so $Q^1 \subseteq \mathrm{Conv}(S^1) \subseteq \mathrm{Conv}(S)$. □

Define $\mathcal{N}_d(I^+, I^-, b) := \{(x,y) \in [-1,+1]^{n+1} : (4)\}$.

2.2 Neurons in the input layers

For the first layer (i.e. $j = 1$) in (1), the input domain D could be arbitrarily constrained. We consider the most common case where $D = [L,U]$, i.e. all inputs are continuous with known bounds. The result follows directly from Theorem 1.

Corollary 2 *Take I^+ and I^- as disjoint subsets of $[n]$ and constants $b^1, b^2 \in \mathbb{R}$ with $b^1 \ge b^2$. Consider a box domain $D = [L,U] \subset \mathbb{R}^n$, along with two subsets of it, $P^1 = \{x \in D : \sum_{i\in I^+} x - \sum_{i\in I^-} x + b^1 \le 0\}$ and $P^2 = \{x \in D : \sum_{i\in I^+} x - \sum_{i\in I^-} x + b^2 \ge 0\}$. Then $\mathrm{Conv}((P^1 \times \{-1\}) \cup (P^2 \times \{+1\}))$ is the set of all $(x,y) \in [L,U] \times [-1,+1]$ satisfying:*

$$\begin{aligned}
&(1+y)b^2 + \sum_{i\in I\cap I^+} (2x_i - \breve{L}_i(1-y)) + \sum_{i\in I^+\setminus I} \breve{U}_i(1+y) \\
\ge\ &\sum_{i\in I\cap I^-} (2x_i - \breve{L}_i(1-y)) + \sum_{i\in I^-\setminus I} \breve{U}_i(1+y) \quad \forall I \subseteq I^+ \cup I^- \qquad (5a)\\
&(1-y)b^1 + \sum_{i\in I\cap I^+} (2x_i - \breve{U}_i(1+y)) + \sum_{I^+\setminus I} \breve{L}_i(1-y) \\
\le\ &\sum_{i\in I\cap I^-} (2x_i - \breve{U}_i(1+y)) + \sum_{i\in I^-\setminus I} \breve{L}_i(1-y) \quad \forall I \subseteq I^+ \cup I^- \qquad (5b)
\end{aligned}$$

Define $\mathcal{N}_c(I^+, I^-, b^1, b^2, L, U) := \{(x,y) \in [L,U] \times [-1,1] : (5)\}$.

Discontinuity Observe that, due to the discontinuity in the sign function at zero, $\mathcal{N}_c$ cannot be immediately converted to a valid MIP formulation in the same way as in Corollary 1. In particular, $\{(x, y) \in \mathcal{N}_c : 2y - 1 \in \{0, 1\}\}$ is a valid MIP formulation for the closure $\mathrm{cl}(\mathrm{Conv}(S))$, which, in general, is not equal to $\mathrm{Conv}(S)$. At first blush, this may seem to be benign, and indeed the closure is used in the BNN verification literature [21]. However, the following example shows that conflating the two sets can potentially lead to incorrect answers for the verification problem.

Example 1. Consider a simple BNN with one-dimensional input ($m_0 = 1$), one hidden layer with two neurons ($m_1 = 2$), and a two-dimensional output layer ($m_2 = 2$). Take $x_1^1 = x_2^1 = \mathrm{sign}(x_1^0 - 0.5)$ as the output of the hidden layer, and $x_1^2 = x_1^1 + 1$ and $x_2^2 = x_2^1$ as the final output of the network. For the verification task, we minimize $x_1^2 - x_2^2$ over the domain $D = [0.5 - \epsilon, 0.5 + \epsilon]$ for a small $\epsilon > 0$.

It is easy to see that, since $x_1^1 = x_2^1$, the objective value will be 1 for any feasible solution. However, consider the potential formulation $(x_1^0, x_1^1), (x_1^0, x_2^1) \in N_C(\{1\}, \emptyset, -0.5, -0.5, 0, 1)$ along with $\{-1, +1\}$ constraints on the outputs of the hidden layer, which is equivalent to the system

$$\begin{aligned} \frac{1}{2}(1 + x_1^1) + 2x_1^0 \geq 0, \quad \frac{1}{2}(1 + x_2^1) + 2x_1^0 \geq 0 \\ -\frac{3}{2} - \frac{1}{2}x_1^1 + 2x_1^0 \leq 0, \quad -\frac{3}{2} - \frac{1}{2}x_2^1 + 2x_1^0 \leq 0 \\ x^0 \in [0.5 - \epsilon, 0.5 + \epsilon], \quad x^1 \in \{-1, +1\}^2. \end{aligned}$$

It is not hard to see that $x^0 = 0.5$ and $x^1 = (-1, +1)$ is feasible for this system, and yields an objective value -1. In other words, $x^0 = 0.5$ is a false adversarial example: the MIP formulation believes that it is adversarial, when in fact the network is robust under the verification task.

For many practical settings, the input for NNs is discrete-valued. For example, a standard image classification network might consider input images represented with 8-bit channels, meaning that the inputs take values in $\{0, ..., 255\}$. In this case with $D = [L, U] \cap \mathbb{Z}^{m_0}$, the graph of sign function with discrete input is closed, and so that we can formulation it with MIP directly by slightly altering the problem data.[4]

Proposition 2 *Take $D = [L, U] \cap \mathbb{Z}^{m_0}$, $b \in \mathbb{R}$, and I^+ and I^- as disjoint subsets of $[n]$. Then an ideal MIP formulation for $S(I^+, I^-, D, b)$ is:*

$$(x, y) \in \mathcal{N}_c(I^+, I^-, \lfloor b \rfloor + 1, \lfloor b \rfloor, \lceil L \rceil, \lfloor U \rfloor), \quad 2y - 1 \in \{0, 1\}.$$

Proof. □

[4] It is straightforward to handle the case where the input domain is normalized by scaling each component; this is often done to accommodate network training.

2.3 Formulating the Entire Neural Network

We are now prepared to present a formulation for an entire BATWN. We will proceed by applying our formulations for single neurons to each neuron in the network, and then composing the resulting formulation by identifying inputs and outputs according to the network structure.

Theorem 2. *Take* NN *as described by* (1)*, along with input constraints* $x^0 \in X := [L, U] \cap \mathbb{Z}^{m_0}$*. Then, a MIP formulation for* $x^r = \texttt{NN}(x^0)$ *is*

$$(x^0, x_i^1) \in \mathcal{N}_c(I_{1,i}^+, I_{1,i}^-, \lfloor b_i^1 \rfloor + 1, \lfloor b_i^1 \rfloor, \lceil L \rceil, \lfloor U \rfloor) \qquad \forall i \in [m_1] \quad \text{(6a)}$$

$$(x^j, x_i^{j+1}) \in \mathcal{N}_d(I_{j+1,i}^+, I_{j+1,i}^-, b_i^{j+1}) \qquad \forall i \in [m_{j+1}], \forall j \in [r-2] \quad \text{(6b)}$$

$$x_i^r = b_i^r + \sum_{k \in I_{r,i}^+} x_k^{r-1} - \sum_{k \in I_{r,i}^-} x_k^{r-1} \qquad \forall i \in [m_r] \quad \text{(6c)}$$

$$2x^j - 1 \in \{0,1\}^{m_j} \qquad \forall j \in [r-1]. \quad \text{(6d)}$$

Proof. Theorem 2 combines Propositions 1 and 2, and Corollary 2 all together to model NN. By Corollary 2 and Proposition 2, (6a) and corresponding constraints in (6d) provide ideal formulations of the neurons in input layer. By Proposition 1, (6b) and corresponding constraints in (6d) provide ideal formulations of the neurons in intermediate layers. The equations in (6c) formulate the output layer. □

Observe that it is straightforward to accommodate other types of input domains (e.g. continuous inputs, side constraints) by modifying (6a) suitably. Also, note again that each $\mathcal{N}_c$ and $\mathcal{N}_d$ will require an exponential number of constraints to describe. We now turn our attention to develop a formulation with a linear number of constraints and obtain cuts in (6) in linear time.

3 Analysis of the formulation

3.1 The big-M variant

Although (6a) and (6b) requires an exponential number of constraints, we only need to keep a small number of the constraints to keep the formulation valid. Take as $\hat{\mathcal{N}}_c$ (respectively $\hat{\mathcal{N}}_d$) as the relaxation of $\mathcal{N}_c$ (resp. $\mathcal{N}_d$) constructed by omiting all constraints from (5a) and (5b) (resp. (4a) and (4b)) except those corresponding to $I = I_{j,i}^+ \cup I_{j,i}^-$.

Corollary 3 *The system* (6) *is still a valid MIP formulation for* $x^r = \texttt{NN}(x^0)$ *if we replace each* $\mathcal{N}_c$ *and* $\mathcal{N}_d$ *with* $\hat{\mathcal{N}}_c$ *and* $\hat{\mathcal{N}}_d$*, respectively.*
Proof. □

We can best understand $\hat{\mathcal{N}}_c$ and $\hat{\mathcal{N}}_d$ as big-M formulations for individual neurons. See Figure 1 for an illustration of the difference between $\mathcal{N}_c$ and $\hat{\mathcal{N}}_c$ for one particular set of problem data. As we observe in the picture, the big-M formulations will, in general, be neither sharp nor ideal.

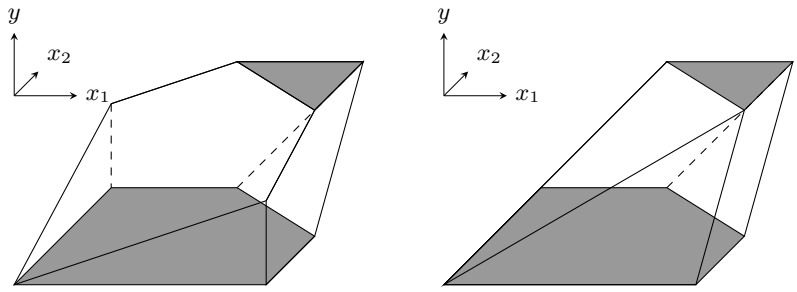

Fig. 1: A binarized activation neuron with two dimensional continuous input. **(Left)** The feasible region for a big-M formulation, and **(Right)** the tightest possible convex relaxation, where $D = [-3, 1]$ and $S = \{(x, -1) : x_1 + x_2 < 0, x \in D\} \cup \{(x, 1) : x_1 + x_2 \geq 0, x \in D\}$.

3.2 Separation

Thankfully, it is straightforward to separate over the inequalities (2a) and (2b) in $\mathcal{O}(n)$ computational time.

Proposition 3 *For $(\hat{x}, \hat{y}) \in D \times [-1, +1]$, one of the most violated constraints (in terms of absolute violation) in the family* (2a) *are associated with the set $I = \{k \in \text{supp}(w) : 2(w_i\hat{x}_i - w_i\breve{U}_i) < (w_i\breve{L}_i - w_i\breve{U}_i)(1 - \hat{y})\}$. Similarly, one of the most violated constraints in the family* (2b) *are associated with the set $I = \{k \in \text{supp}(w) : 2w_i(\breve{U}_i - \hat{x}_i) < (w_i\breve{U}_i - w_i\breve{L}_i)(1 - \hat{y})\}$.*

Proof. In order to find the most violated constraints in (2a), we need to solve the following optimization problem

$$
\begin{aligned}
&\underset{I \subseteq \text{supp}(w)}{\arg\min} \sum_{i \in I} w_i\hat{x}_i + \sum_{i \notin I} w_i\breve{U}_i + b^2 - \frac{1}{2}\left(\sum_{i \in I} w_i\breve{L}_i + \sum_{i \notin I} w_i\breve{U}_i + b^2\right)(1 - \hat{y}) \\
= &\underset{I \subseteq \text{supp}(w)}{\arg\min} \sum_{i \in I} w_i\hat{x}_i + \sum_{i \notin I} w_i\breve{U}_i - \frac{1}{2}\left(\sum_{i \in I} w_i\breve{L}_i + \sum_{i \notin I} w_i\breve{U}_i\right)(1 - \hat{y}) \\
= &\underset{I \subseteq \text{supp}(w)}{\arg\min} \sum_{i \in I}(2w_i\hat{x}_i - w_i\breve{L}_i(1 - \hat{y})) + \sum_{i \notin I}(2w_i\breve{U}_i - w_i\breve{U}_i(1 - \hat{y}))
\end{aligned}
$$

Thus, it is not hard to see that in one of the optimal solution I^*, $i \in I^*$ if $(2w_i\hat{x}_i - w_i\breve{L}_i(1 - \hat{y})) < (2w_i\breve{U}_i - w_i\breve{U}_i(1 - \hat{y}))$.

Similarly, for (2b),

$$\begin{aligned}
&\operatorname*{arg\,min}_{I \subseteq \operatorname{supp}(w)} \sum_{i \in I} w_i(\breve{U}_i - \hat{x}_i) - \frac{1}{2}\left(\sum_{i \in I} w_i \breve{U}_i + \sum_{i \notin I} w_i \breve{L}_i + b^1\right)(1-\hat{y}) \\
&= \operatorname*{arg\,min}_{I \subseteq \operatorname{supp}(w)} \sum_{i \in I} w_i(\breve{U}_i - \hat{x}_i) - \frac{1}{2}\left(\sum_{i \in I} w_i \breve{U}_i + \sum_{i \notin I} w_i \breve{L}_i\right)(1-\hat{y}) \\
&= \operatorname*{arg\,min}_{I \subseteq \operatorname{supp}(w)} \sum_{i \in I} 2w_i()
\end{aligned}$$

□

Proposition 4 *For $(\hat{x}, \hat{y}) \in D \times [-1, +1]$, the most violated constraints (in terms of absolute violation) in the families* (4a) *and* (5a) *are associated with the set $I = \{k \in I^+ : \hat{x}_k < \hat{y}\} \cup \{k \in I^- : -\hat{x}_k < \hat{y}\}$. Similarly, the most violated constraints in the families* (4b) *and* (5b) *are associated with the set $I = \{k \in I^+ : \hat{x}_k > \hat{y}\} \cup \{k \in I^- : -\hat{x}_k > \hat{y}\}$.*

Along with the discussion in Section 3.1, this result suggests that the most practical way to use formulation (6) is in a cutting plane fashion. In particular, start with the big-M formulation given by $\hat{\mathcal{N}}_c$ and $\hat{\mathcal{N}}_d$, and then generate violated constraints from the exponentially large families describing $\mathcal{N}_c$ and $\mathcal{N}_d$ as needed.

3.3 Convex relaxations are intrinsically weak

As shown in Figure 1, the ideal formulation can provide a tighter relaxation than the big-M formulation. However, we will observe that convex relaxations of BATWNs are intrinsically weak. In the following example, we show that even if we apply the ideal formulation for each neuron, the output of the network can be completely uncoupled from the input in a verification problem.

Example 2. Consider a simple BATWN with three-dimensional input ($m_0 = 3$), two hidden layer with four and two neurons ($m_1 = 4, m_2 = 2$), and a two-dimensional output layer ($m_3 = 2$). Take $x_1^1 = x_2^1 = x_3^1 = x_4^1 = \operatorname{sign}(x_1^0 - x_2^0 + x_3^0)$ as the outputs of the first hidden layer, $x_1^2 = \operatorname{sign}(-x_1^1 - x_2^1)$ and $x_2^2 = \operatorname{sign}(-x_3^1 - x_4^1)$ as the outputs of the second hidden layer, and then $x_1^3 = x_1^2 + 1, x_2^3 = x_2^2$ as the outputs of the network. Simple interval arithmetic shows that the outputs (x_1^3, x_2^3) is constrained to lie within the box $[0, 2] \times [-1, +1]$. For our given network, the formulation (6) is

$$\begin{aligned}
&(x^0, x_i^1) \in \mathcal{N}_c(\{1,3\}, \{2\}, 1, 0, (-1,-1,-1), (+1,+1,+1)), \quad \forall i \in \{1,2,3,4\} \\
&(x^1, x_1^2) \in \mathcal{N}_d(\emptyset, \{1,2\}, 0), \quad (x^1, x_1^2) \in \mathcal{N}_d(\emptyset, \{3,4\}, 0) \\
&x_1^3 = x_1^2 + 1, \quad x_2^3 = x_2^2
\end{aligned}$$

It is possible to show that, if we fix $x^0 = (0, 0, 0)$, then any

$$(x^1, x^2) \in \{((\alpha, \alpha, \beta, \beta), (1 - 2\alpha, 1 - 2\beta)) : \alpha, \beta \in \{0, 1\}\}$$

satisfies the above constraints. Therefore, by convexity we have shown that for a single input point $x^0 = (0, 0, 0)$, the output can take any possible value. In other words, the strongest possible convex relaxation offers no additional strength beyond interval arithmetic.

3.4 Consistency

The previous example illustrates that our approach of generating cutting planes from $\mathcal{N}_c$ and $\mathcal{N}_d$, as described in Section 3.2, may not greatly improve the relaxation bound. However, cutting planes are still potentially computationally useful inside the search tree, where they can enforce consistency by excluding infeasible partial assignments and reducing backtracking [10, 11].

Definition 1 (Definition 5 [10]) *A polyhedron $\mathcal{S} \subseteq [0,1]^k$ is* sequentially k-consistent *if, for each $v \in \{0,1\}^{k-1}$, $\{z \in \mathcal{S} : z_{1:(k-1)} = v\} \subseteq \mathcal{S} \cap \{0,1\}^k$.*

Our MIP formulation for discrete-input neurons is consistent (after an affine transformation). Intuitively, the following result says that, after the inputs are fixed, the output are completely determined.

Proposition 5 *The system $\{(2x-1, 2y-1) \in [0,1]^{n+1} : (x,y) \in \mathcal{N}_d\}$ is sequentially $(n+1)$-consistent.*

Proof. By Definition 1, we want to prove that $y \in \{-1, +1\}$, i.e. $(2y-1) \in \{0,1\}$, if $(x,y) \in \mathcal{N}_d$ and $x \in \{-1,1\}^n$, i.e. $(2x-1) \in \{0,1\}^n$. Let $\hat{x}$ be an arbitrary point in $\{-1,+1\}^n$. We only prove the case that $\sum_{i \in I^+} \hat{x}_i - \sum_{i \in I^-} \hat{x}_i + b \geq 0$; the other case that $\sum_{i \in I^+} \hat{x}_i - \sum_{i \in I^-} \hat{x}_i + b < 0$ follows the same manner. Because $\mathcal{N}_d = \mathrm{Conv}(S)$, we know that $(\hat{x}, +1) \in \mathcal{N}_d$. Now, we want to prove that $(\hat{x}, \hat{y}) \notin \mathcal{N}_d$ for any $\hat{y} < 1$. Let $\hat{y} = \arg\min\{y : (\hat{x}, y) \in \mathcal{N}_d\}$. Then, $(\hat{x}, \hat{y})$ must be an extreme point of $\mathrm{Conv}(S)$ because $\hat{x} \in \mathrm{ext}([-1,+1]^n)$. Thus, $(\hat{x}, \hat{y}) \in S$, which forces $\hat{y} = 1$. □

We can potentially say even more: depending on the values, we may only need to fix a subset of the input variables in order to derive the consistency property.

Proposition 6 *Take $b \in \mathbb{R}$, disjoint subsets $I^+, I^- \subseteq [n]$, some $J \subseteq [n]$, and $v \in \{-1,+1\}^{|J|}$. Given $(\hat{x}, \hat{y}) \in \mathcal{N}_d(I^+, I^-, b)$ satisfying $\hat{x}_J = v$, then:*

1. *If $b + \sum_{i \in I^+ \cap J} v_i - \sum_{i \in I^- \cap J} v_i \geq |(I^+ \cup I^-) \setminus J|$, then $\hat{y} = 1$.*
2. *If $b + \sum_{i \in I^+ \cap J} v_i - \sum_{i \in I^- \cap J} v_i < -|(I^+ \cup I^-) \setminus J|$, then $\hat{y} = -1$.*

Proof. We prove only Statement 1; the second follows in a similar manner. Since $\hat{x} \in \{x \in [-1,+1]^n : x_J = v\}$, any such $\hat{x}$ can be written as a convex combination of points in the set $C = \{x \in \{-1,+1\}^n : x_J = v\}$. Therefore, it is sufficient to prove that $\hat{y}$ must be 1 when $\hat{x} \in C$ and $b + \sum_{i \in I^+ \cap J} v_i - \sum_{i \in I^- \cap J} v_i \geq |(I^+ \cup I^-) \setminus J|$. Since $|\hat{x}_i| \leq 1$, $\sum_{i \in I^+ \setminus J} \hat{x}_i - \sum_{i \in I^- \setminus J} \hat{x}_i \geq -|(I^+ \cup I^-) \setminus J|$. Thus, $b + \sum_{i \in I^+} \hat{x}_i - \sum_{i \in I^-} \hat{x}_i \geq 0$. As $\hat{x}$ is an extreme point of $[-1,1]^n$, we apply Proposition 5 to conclude that $\hat{y} = 1$. □

3.5 Separating consistency cuts

The conditions of Proposition 6 are quite similar to the constraints defining $\mathcal{N}_d$. Using this connection, we can derive an alternative separation rule that, inside of the search tree, will enforce the consistency property.

Consider some $(\hat{x}, \hat{y}) \in [-1, +1]^{n+1}$ and take J is the set of indices of x that has already been fixed at the current node in the search tree. Then, the new cut generated at this point in the family (4a) is associated with set $I = \{k \in (I^+ \cap J) : \hat{x}_k < \hat{y}\} \cup \{k \in (I^- \cap J) : -\hat{x}_k < \hat{y}\}$. Similarly, the new cut at this point in the family (4b) can be found with set $I = \{k \in (I^+ \cap J) : \hat{x}_k > \hat{y}\} \cup \{k \in (I^- \cap J) : -\hat{x}_k > \hat{y}\}$.

Note the connection to Proposition 6: if the cuts separated in this manner satisfy the conditions in Statement 2 or 1, respectively, the cut will fix the output to -1 or $+1$, respectively. For this reason, we refer to the cutting planes separated in this fashion as consistency cuts.

4 Computational Results

We now turn to computational experiments to validate our new methods. We study the verification of image classification networks on the standard MNIST digit dataset [24]. The inputs are 28×28 8-bit grayscale images [5] (normalized to lie in $[0, 1]^{28\times 28}$), and the 10 outputs correspond to each digit 0 to 9. Given a trained BATWN, $\texttt{NN} : [0, 1]^{28\times 28} \to \mathbb{R}^{10}$, an input image $x^0 \in [0, 1]^{28\times 28}$ with true label i, and a neighborhood radius $\epsilon > 0$, we solve the optimization problem $\min_{||\tilde{x}-x^0||_\infty \leq \epsilon} \texttt{NN}(\tilde{x})_i - \texttt{NN}(\tilde{x})_j$. If the optimal objective value is greater than 0, then the network is robust around x^0 with respect to class j. Otherwise, an optimal solution x^* is an "adversarial example" that induces the network to give an incorrect prediction.

We train a series of models with two fully connected layers with binarized activation functions, along with an output on which a softmax is applied. The networks are trained with Larq [14], which is built on top of TensorFlow. The network is trained with L1 regularization, first with Adam [22] for 5 epochs as a warm start and then for another 10 epochs with Adadelta [48]. For each model, we randomly generate 100 instances by sampling a target image from the test set and then randomly selecting a target class $j \neq i$. In more detail, we have:

- Model 1: 2 layers of 200 neurons each. Train accuracy: 93.48%. Test accuracy: 92.78%. Weight sparsity per layer: 50.37%, 98.25%, and 0%, respectively.
- Model 2: 2 layers of 200 neurons each. Train accuracy: 92.78%. Test accuracy: 92.20%. Weight sparsity per layer: 72.29%, 98.69%, and 64.45%, respectively.
- Model 3: 2 layers of 500 neurons each. Train accuracy: 95.44%. Test accuracy: 94.82%. Weight sparsity per layer: 55.13%, 99.36%, and 0%, respectively.

[5] We relax the input domain to be continuous as [2, 40, 41] even if the input domain should be discrete.

We note that each model attains the target accuracy range of 90% to 95% mentioned in [21].

We use Gurobi v9.0.0 as the MIP solver, with four threads on a server with 16 GB of RAM and Intel(R) Xeon(R) W-2102 CPU with 4 cores @ 2.90GHz. We set a time limit of 30 minutes for each instance. We compare the performances of four methods in the experiments:

- No Cuts: Gurobi with all cutting planes turned off.
- No Cuts + (6): Gurobi with all (internal) cutting planes turned off, and with the consistency cuts.
- Default: Gurobi's default configuration.
- Default + (6): Gurobi's default configuration, with consistency cuts added.

Consistency cut separation entails performing the separation described in Section 3.5 for the discrete layers in cut callbacks. At each node in the search tree, we submit up to 6 violated consistency cuts per neuron in the discrete layers.

Models	Methods	Time (s)	Nodes	Gap (%)	Solved	Winners
	No Cuts	45.10	100964	24.19	98	28
Model 1	No Cuts + (6)	**39.74**	86557	**0.00**	**100**	**66**
$\epsilon = 13/255$	Default	79.12	84909	32.11	95	4
	Default + (6)	80.89	**75694**	29.96	95	2
	No Cuts	55.78	249988	**20.69**	90	24
Model 2	No Cuts + (6)	**43.69**	164857	23.20	**92**	**64**
$\epsilon = 20/255$	Default	102.31	148816	27.17	85	1
	Default + (6)	96.97	**139500**	26.19	85	3
	No Cuts	13.78	35860	32.72	96	25
Model 3	No Cuts + (6)	**11.89**	**25100**	**31.91**	**97**	**68**
$\epsilon = 6/255$	Default	24.33	26840	44.58	94	2
	Default + (6)	23.90	27975	41.07	95	2
	No Cuts	86.84	169206	51.00	82	18
Model 3	No Cuts + (6)	**68.19**	**108898**	**43.07**	**83**	**63**
$\epsilon = 8/255$	Default	157.15	118229	73.25	72	0
	Default + (6)	157.79	119050	83.62	73	3

Table 1: Computational results of all the models.

We summarize the computational results in Table 1. For performance metrics, we use: "Time" as the run time considering only those instances which terminate before the time limit (shifted geometric mean with a shift of 10); "Nodes" as the number of nodes enumerated in the search tree, again considering only those instances which terminate before the time limit (arithmetic mean); "Gap" as the MIP gap percentage reported for those instances that terminate at the time limit (shifted geometric mean with a shift of 0.01) ; "Solved" as the number of instances that terminate before the time limit; and "Winners" as the number of instances on which the given method terminates at optimality in the least time.

We observe that our proposed method "No Cuts + (6)" performs the best on a supermajority of instances. It is the fastest on roughly two thirds of all instances across the four families, with an average solve time reduction of 11.88% to 23.47% compared with the second best method. It also reduces the the node count against its baseline ("No Cuts") on average by 14.27% to 35.64%. We observe that the "Default" method is considerably slower than pure enumeration. This seems to corroborate our observation in Section 3.3 by showing that standard cutting planes aimed at strengthening the relaxation will be relatively ineffective for a problem such as this one, where convex relaxations are intrinsically weak. However, our computational results suggest that cutting planes can nonetheless still be computationally useful if we instead focus on using them to enforce consistency in the search tree.

References

1. Amir, G., Wu, H., Barrett, C., and Katz, G. An SMT-based approach for verifying binarized neural networks. *arXiv preprint arXiv:2011.02948* (2020).
2. Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., and Vielma, J. P. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming* (2020), 1–37.
3. Bartolini, A., Lombardi, M., Milano, M., and Benini, L. Neuron constraints to model complex real-world problems. In *International Conference on Principles and Practice of Constraint Programming* (2011), Springer, pp. 115–129.
4. Bartolini, A., Lombardi, M., Milano, M., and Benini, L. Optimization and controlled systems: A case study on thermal aware workload dispatching. In *Twenty-Sixth AAAI Conference on Artificial Intelligence* (2012).
5. Bunel, R., Lu, J., Turkaslan, I., Kohli, P., Torr, P., and Mudigonda, P. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research 21*, 2020 (2020).
6. Bunel, R. R., Turkaslan, I., Torr, P., Kohli, P., and Mudigonda, P. K. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems* (2018), pp. 4790–4799.
7. Carlini, N., and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy* (2017), IEEE, pp. 39–57.
8. Cheng, C.-H., Nührenberg, G., and Ruess, H. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis* (2017), Springer, pp. 251–268.
9. Conforti, M., Cornuéjols, G., Zambelli, G., et al. *Integer programming*, vol. 271. Springer, 2014.
10. Davarnia, D., and Hooker, J. N. Consistency for 0–1 programming. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research* (2019), Springer, pp. 225–240.
11. Davarnia, D., Rajabalizadeh, A., and Hooker, J. Achieving consistency with cutting planes.
12. Dutta, S., Jha, S., Sankaranarayanan, S., and Tiwari, A. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium* (2018), Springer, pp. 121–138.
13. Fischetti, M., and Jo, J. Deep neural networks and mixed integer linear optimization. *Constraints 23*, 3 (2018), 296–309.
14. Geiger, L., and Team, P. Larq: An open-source library for training binarized neural networks. *Journal of Open Source Software 5*, 45 (2020), 1746.
15. Han, S., and Gómez, A. Single-neuron convexifications for binarized neural networks. *Technical Report* (2021).
16. He, Z., Gong, B., and Fan, D. Optimize deep convolutional neural network with ternarized weights and high accuracy. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2019), IEEE, pp. 913–921.
17. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *Advances in Neural Information Processing Systems* (2016), pp. 4107–4115.
18. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research 18*, 1 (2017), 6869–6898.

19. Jia, K., and Rinard, M. Efficient exact verification of binarized neural networks. *arXiv preprint arXiv:2005.03597* (2020).
20. Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification* (2017), Springer, pp. 97–117.
21. Khalil, E. B., Gupta, A., and Dilkina, B. Combinatorial attacks on binarized neural networks. In *International Conference on Learning Representations (ICLR)* (2019).
22. Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
23. LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature 521*, 7553 (2015), 436–444.
24. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.
25. Li, F., Zhang, B., and Liu, B. Ternary weight networks. *arXiv preprint arXiv:1605.04711* (2016).
26. Liu, C., Arnon, T., Lazarus, C., Barrett, C., and Kochenderfer, M. J. Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758* (2019).
27. Lombardi, M., and Gualandi, S. A lagrangian propagator for artificial neural networks in constraint programming. *Constraints 21*, 4 (2016), 435–462.
28. Lomuscio, A., and Maganti, L. An approach to reachability analysis for feed-forward ReLU neural networks. *arXiv preprint arXiv:1706.07351* (2017).
29. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations* (2018).
30. Narodytska, N., Kasiviswanathan, S. P., Ryzhyk, L., Sagiv, M., and Walsh, T. Verifying properties of binarized deep neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).
31. Narodytska, N., Zhang, H., Gupta, A., and Walsh, T. In search for a sat-friendly binarized neural network architecture. In *International Conference on Learning Representations* (2020).
32. Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 427–436.
33. Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision* (2016), Springer, pp. 525–542.
34. Say, B., Wu, G., Zhou, Y. Q., and Sanner, S. Nonlinear hybrid planning with deep net learned transition models and mixed-integer linear programming. In *the Twenty-Sixth International Joint Conference on Artificial Intelligence* (2017), pp. 750–756.
35. Schweidtmann, A. M., and Mitsos, A. Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications 180*, 3 (2019), 925–948.
36. Serra, T., and Ramalingam, S. Empirical bounds on linear regions of deep rectifier networks. In *the Thirty-Fourth AAAI Conference on Artificial Intelligence* (2020), pp. 5628–5635.

37. Serra, T., Tjandraatmadja, C., and Ramalingam, S. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning* (2018), PMLR, pp. 4558–4566.
38. Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages 3*, POPL (2019), 1–30.
39. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations* (2014).
40. Tjandraatmadja, C., Anderson, R., Huchette, J., Ma, W., Patel, K., and Vielma, J. P. The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. *arXiv preprint arXiv:2006.14076* (2020).
41. Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations* (2019).
42. Vielma, J. P. Mixed integer linear programming formulation techniques. *Siam Review 57*, 1 (2015), 3–57.
43. Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Daniel, L., Boning, D., and Dhillon, I. Towards fast computation of certified robustness for relu networks. In *Proceedings of Machine Learning Research* (Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018), vol. 80, PMLR, pp. 5276–5285.
44. Wong, E., and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning* (2018), PMLR, pp. 5286–5295.
45. Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems* (2018), pp. 8400–8409.
46. Xiao, K., Tjeng, V., Shafiullah, N. M., and Madry, A. Training for faster adversarial robustness verification via inducing ReLU stability. In *International Conference on Learning Representations (ICLR)* (2019).
47. Yuan, X., He, P., Zhu, Q., and Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems 30*, 9 (2019), 2805–2824.
48. Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).
49. Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems* (2018), pp. 4939–4948.