# Formal Analysis of Deep Binarized Neural Networks

**Nina Narodytska**
VMware Research, USA
nnarodytska@vmware.com

## Abstract

Understanding properties of deep neural networks is an important challenge in deep learning. Deep learning networks are among the most successful artificial intelligence technologies that is making impact in a variety of practical applications. However, many concerns were raised about 'magical' power of these networks. It is disturbing that we are really lacking of understanding of the decision making process behind this technology. Therefore, a natural question is whether we can trust decisions that neural networks make. One way to address this issue is to define properties that we want a neural network to satisfy. Verifying whether a neural network fulfills these properties sheds light on the properties of the function that it represents. In this work, we take the verification approach. Our goal is to design a framework for analysis of properties of neural networks. We start by defining a set of interesting properties to analyze. Then we focus on Binarized Neural Networks that can be represented and analyzed using well-developed means of Boolean Satisfiability and Integer Linear Programming. One of our main results is an exact representation of a binarized neural network as a Boolean formula. We also discuss how we can take advantage of the structure of neural networks in the search procedure.

## 1 Introduction

Deep neural networks have become ubiquitous in machine learning with applications ranging from computer vision to speech recognition and natural language processing. Neural networks demonstrate excellent performance on many practical problems, often beating specialized algorithms for these problems, which led to their rapid adoption in industrial applications. With such a wide adoption, important questions arise regarding our understanding of the decision making process of these neural networks: Is there a way to analyze deep neural networks? Can we explain their decisions? How robust are these networks to perturbations of inputs? How critical is the choice of one architecture over an other? Recently, a new line of research on understanding neural networks has emerged that looks into a wide range of such questions, from interpretability of neural networks to verifying their properties [Bau *et al.*, 2017; Szegedy *et al.*, 2014; Pulina and Tacchella, 2010; Huang *et al.*, 2017; Katz *et al.*, 2017; Cheng *et al.*, 2017b; Narodytska *et al.*, 2017; Leofante *et al.*, 2018].

There are a number of ways to analyze a neural network. One way is to query the network directly, e.g. analyzing of important parts of the input using numerical optimization techniques, extracting interpretable information from the network, e.g. using decision trees, and approximating the network with a simpler function [Simonyan *et al.*, 2013; Ribeiro *et al.*, 2016; Koh and Liang, 2017; Frosst and Hinton, 2017]. These approaches scale to large networks. However, they fail to provide formal guarantees about properties of the network. An alternative approach is based on formal verification techniques. The idea is to encode the network and the property we aim to verify as a formal statement, using ILP, SMT or SAT, for example. If the encoding provides an exact representation of the network then we can study any property related to this network, e.g. how sensitive the network is to perturbations of the input.

In this work we focus on an important class of neural networks: Binarized Neural Networks (BNNs) [Hubara *et al.*, 2016]. These networks have a number of important features that are useful in resource constrained environments, like embedded devices or mobile phones. Firstly, these networks are memory efficient, as their parameters are primarily binary. Secondly, they are computationally efficient as all activations are binary, which enables the use of specialized algorithms for fast binary matrix multiplication. These networks have been shown to achieve performance comparable to traditional deep networks that use floating point precision [Hubara *et al.*, 2016]. Recently, BNNs have been deployed for various embedded applications ranging from image classification [McDanel *et al.*, 2017] to object detection [Kung *et al.*, 2017].

We start by discussing a set of interesting properties of neural network, including properties that relate inputs and outputs of the network, e.g. robustness and invertibility, and properties that relate two networks, like network equivalence. We discuss how binarized neural networks can be represented as Boolean or ILP formulas and how the properties that we identify can be represented in the same formalism. Finally, we consider main challenges that we face in using decision procedures in reasoning about BNNs and how we can potentially address them by exploiting the strutural properties of neural networks.

## 2 Neural Networks

**Notation.** We denote $[m] = \{1, \ldots, m\}$. Vectors are in column-wise fashion, denoted by boldface letters. For a vector $\mathbf{v} \in \mathbb{R}^m$, we use $(v_1, \ldots, v_m)$ to denote its $m$ elements.

We define the supervised image classification problem that we focus on. We are given a set of training images drawn from an unknown distribution $\nu$ over $\mathbb{Z}^n$, where $n$ represents the "size" of individual images. Each image is associated with a label generated by an unknown function $\mathrm{L} : \mathbb{Z}^n \to [s]$, where $[s] = \{1, \ldots, s\}$ is a set of possible labels. During the training phase, given a labeled training set, the goal is to learn a neural network classifier that can be used for inference: given a new image drawn from the same distribution $\nu$, the classifier should predict its true label. During the inference phase, the network is *fixed*. In this work, we study properties of such fixed networks generated post training. Let $\mathcal{X}$ denote the domain from which inputs are drawn. For example, in the case of images, $\mathcal{X} = \mathbb{Z}^n$.

## 3 Analysis of Neural Networks

In this section, we define several important properties of neural networks, ranging from robustness to properties related to network structure. We consider a general feedforward neural network denoted by F. Let $\mathrm{F}(\mathbf{x})$ represent the output of F on input $\mathbf{x}$ and $\ell_{\mathbf{x}} = \mathrm{L}(\mathbf{x})$ be the ground truth label of $\mathbf{x}$. For example, $\mathbf{x}$ can be an image of a bus and $\ell_{\mathbf{x}}$ its ground truth label, i.e. 'bus'.

### 3.1 Robustness

Robustness is an important property that guards the network against tampering of its outcome by perturbing the inputs. Robustness is by far the most researched notion in formal methods literature of verification of neural networks [Katz *et al.*, 2017; Huang *et al.*, 2017; Cheng *et al.*, 2017a; Bunel *et al.*, 2017; Fischetti and Jo, 2017]. It is also known as vulnerability to adversarial attacks in the neural networks literature [Szegedy *et al.*, 2014; Goodfellow *et al.*, 2015]. We make two simplifications compared to the related work. First, we look at the robustness property in the context of the classification problem. However, these definitions can be extended to networks with richer outputs. Second, we consider the $L_\infty$ norm to measure distance for simplicity. [Leofante *et al.*, 2018] give a survey of neural networks properties from the formal verification point of view in the general case.

There are two forms of robustness that are widely considered in the literature: global and local robustness. Global robustness means that for any valid input, there is no small perturbation that can change the decision of the network on this input. Global robustness is a strong property that is challenging to verify for many applications.

**Definition 1** (Global Robustness)**.** *A feedforward neural network* F *is globally* $\epsilon$*-robust if for any* $\mathbf{x}$*,* $\mathbf{x} \in \mathcal{X}$ *and* $\tau$*,* $\|\tau\|_\infty \leq \epsilon$ *we have that* $\mathrm{F}(\mathbf{x} + \tau) = \ell_{\mathbf{x}}$.

Local robustness is a property that is defined for a single input $\mathbf{x}$. It is a much weaker property that can be efficiently checked for small realistic inputs.

**Definition 2** (Local Robustness)**.** *A feedforward neural network* F *is locally* $\epsilon$*-robust for an input* $\mathbf{x}$*,* $\mathbf{x} \in \mathcal{X}$*, if there does not exist* $\tau$*,* $\|\tau\|_\infty \leq \epsilon$*, such that* $\mathrm{F}(\mathbf{x} + \tau) \neq \ell_{\mathbf{x}}$.

There are many variants of robustness that can be positioned between local and global robustness. One example is to define a relaxation of the global robustness property by allowing a violation of the property on a small fraction of inputs, that comes from the notion of universal adversarial attacks [Moosavi-Dezfooli *et al.*, 2016].

### 3.2 Invertibility

Invertibility of the neural network is an interesting property that recently was considered in the verification literature [Ehlers, 2017; Korneev *et al.*, 2018]. The main idea is to explore a set of inputs that map to a given output. For example, what the inputs of the network are (if exist) that map to a given output. In general, we need to define declarative constraints on the inputs otherwise a lot of noisy images will be generated. Let $C(\mathcal{X})$ denote the constrained domain of inputs. These constraints come from the practical application.

**Definition 3** (Local Invertibility)**.** *A feedforward neural network* F *is locally invertible for an output* $\mathbf{s}$ *if there exists* $\mathbf{x}$*,* $\mathbf{x} \in C(\mathcal{X})$*, such that* $\mathrm{F}(\mathbf{x}) = \mathbf{s}$.

A related problem here is how to enumerate multiple, preferably diverse by some measure, inputs of the network that map to a given output.

### 3.3 Network Equivalence

We consider is equivalence of networks. Informally, two networks $\mathrm{F}_1$ and $\mathrm{F}_2$ are equivalent if they generate same outputs on all inputs drawn from the domain $\mathcal{X}$.

**Definition 4** (Network Equivalence)**.** *Two feedforward neural networks* $\mathrm{F}_1$ *and* $\mathrm{F}_2$ *are equivalent if for all* $\mathbf{x} \in \mathcal{X}$*,* $\mathrm{F}_1(\mathbf{x}) = \mathrm{F}_2(\mathbf{x})$.

An important case of using network equivalence is certifying a *network alteration*. Consider a scenario where a part of the trained network has been altered to form a new network. This change could arise due to model reduction operations that are commonly performed on deep networks to make them amenable to execution on resource-constrained devices [Reagen *et al.*, 2017] or they could arise from other sources of noise including adversarial corruption of the network. The question now is whether the altered network is equivalent to the original network?

Next we consider a class of networks that we focus on in this work.

## 4 Binarized Neural Networks

A binarized neural network (BNN) is a feedforward network where weights and activations are predominantly binary [Hubara *et al.*, 2016]. It is convenient to describe the structure of a BNN in terms of composition of blocks of layers rather than individual layers. Each block consists of a collection of linear and non-linear transformations. Blocks are assembled sequentially to form a BNN.

| Structure of $k$th Internal block, $\text{BLK}_k : \{-1,1\}^{n_k} \rightarrow \{-1,1\}^{n_{k+1}}$ on input $\mathbf{x}_k \in \{-1,1\}^{n_k}$ | |
|---|---|
| LIN | $\mathbf{y} = A_k\mathbf{x}_k + \mathbf{b}_k$ , where $A_k \in \{-1,1\}^{n_{k+1}\times n_k}$ and $\mathbf{b}_k \in \mathbb{R}^{n_{k+1}}$ |
| BN | $z_i = \alpha_{k_i}\left(\frac{y_i-\mu_{k_i}}{\sigma_{k_i}}\right) + \gamma_{k_i}$, where $\mathbf{y} = (y_1,\dots,y_{n_{k+1}})$, and $\alpha_{k_i},\gamma_{k_i},\mu_{k_i},\sigma_{k_i} \in \mathbb{R}$. Assume $\sigma_{k_i} > 0$. |
| BIN | $\mathbf{x}_{k+1} = \text{sign}(\mathbf{z})$ where $\mathbf{z} = (z_1,\dots,z_{n_{k+1}}) \in \mathbb{R}^{n_{k+1}}$ and $\mathbf{x}_{k+1} \in \{-1,1\}^{n_{k+1}}$ |

| Structure of Output Block, $\text{O} : \{-1,1\}^{n_d} \rightarrow [s]$ on input $\mathbf{x}_d \in \{-1,1\}^{n_d}$ | |
|---|---|
| LIN | $\mathbf{w} = A_d\mathbf{x}_d + \mathbf{b}_d$, where $A_d \in \{-1,1\}^{s\times n_d}$ and $\mathbf{b}_d \in \mathbb{R}^s$ |
| ARGMAX | $o = \text{argmax}(\mathbf{w})$, where $o \in [s]$ |

Table 1: Structure of internal and outputs blocks, which stacked together form a binarized neural network. In the training phase, there might be an additional *hard tanh* layer after batch normalization. $A_k$ and $b_k$ are parameters of the LIN layer, whereas $\alpha_{k_i},\gamma_{k_i},\mu_{k_i},\sigma_{k_i}$ are parameters of the BN layer. $\mu$'s and $\sigma$'s correspond to mean and standard deviation computed in the training phase. The BIN layer is parameter free.

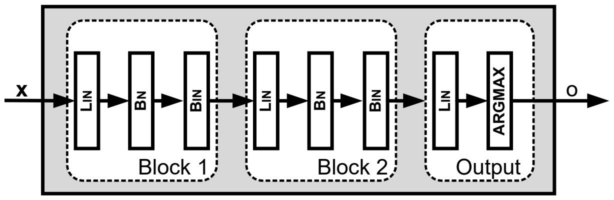

Figure 1: A schematic view of a binarized neural network. The internal blocks also have an additional hard tanh layer.

**Internal Block.** Each internal block (denoted as BLK) in a BNN performs a series of transformations over a binary input vector and outputs a binary vector. While the input and output of a BLK are binary vectors, internal layers of BLK can produce real-valued intermediate outputs. A common construction of an internal BLK (taken from [Hubara *et al.*, 2016]) is composed of three main operations:[1] a linear transformation (LIN), batch normalization (BN), and binarization (BIN). Table 1 presents the formal definition of these transformations. The first step is a linear (affine) transformation of the input vector. The linear transformation can be based on a fully connected layer or a convolutional layer. The linear transformation is followed by a scaling which is performed with a batch normalization operation [Ioffe and Szegedy, 2015]. Finally, binarization is performed using the sign function to obtain a binary output vector. Figure 1 shows two BLKs connected sequentially.

**Output Block.** The output block (denoted as O) produces the classification decision for a given image. It consists of two layers (see Table 1). The first layer applies a linear (affine) transformation that maps its input to a vector of integers, one for each output label class. This is followed by a ARGMAX layer, which outputs the index of the largest entry in this vector as the predicted label.

**Network of Blocks.** BNN is a deep feedforward network formed by assembling a sequence of internal blocks and an output block. Suppose we have $d-1$ blocks, $\text{BLK}_1,\dots,\text{BLK}_{d-1}$ that are placed consecutively, so the output of a block is an input to the next block in the list. Let $\mathbf{x}_k$ be the input to $\text{BLK}_k$ and $\mathbf{x}_{k+1}$ be its output. The input of the first block is the input of the network. We assume that the input of the network is a vector of integers, which is true for the image classification task if images are in the standard RGB format. Note that these integers can be encoded with binary values $\{-1,1\}$ using a standard encoding. Therefore, we keep notations uniform for all layers by assuming that inputs are all binary. The output of the last layer, $\mathbf{x}_d \in \{-1,1\}^{n_d}$, is passed to the output block O to obtain the label.

[1]In the training phase, there is an additional *hard tanh* layer after batch normalization layer that is omitted in the inference phase [Hubara *et al.*, 2016].

**Definition 5** (Binarized Neural Network)**.** *A binarized neural network* BNN : $\{-1,1\}^n \rightarrow [s]$ *is a feedforward network that is composed of* $d$ *blocks,* $\text{BLK}_1,\dots,\text{BLK}_{d-1},\text{O}$. *Formally, given an input* $\mathbf{x}$,

$$\text{BNN}(\mathbf{x}) = \text{O}(\text{BLK}_{d-1}(\dots\text{BLK}_1(\mathbf{x})\dots)).$$

## 5 Progress in Formal Analysis of BNNs

We overview results that we have obtained so far on analysis of BNNs [Narodytska *et al.*, 2017; Korneev *et al.*, 2018].

### 5.1 Encoding of BNNs

Our first contribution is to propose an *exact* encoding of BNNs as a Boolean formula in the sense that all valid pairs of inputs and outputs of a given network are exactly solutions of the Boolean formula. To the best of our knowledge, this is the first work on verifying properties of deep neural networks using an exact Boolean encoding of the network. Independently, a similar encoding was proposed by [Cheng *et al.*, 2017b]. As we mentioned above, while the input and the output of each block are binary vectors, the intermediate values are real. The key insight was that we should consider a composition of functions rather than functions of individual layers separately. Using this approach, we showed that a safe rounding can be performed and the network can be encoded as a set of reified cardinality constraints. In turn, these constraints can be compactly encoded into a Boolean formula using one of the commonly used encodings, e.g. we used the sequential counters encoding [Sinz, 2005]. Hence, we can use powerful SAT solvers to perform property verification.

### 5.2 Robustness of BNNs

We considered a problem of local robustness of BNNs. To be able to verify this property, first, we encoded it as a Boolean formula. Second, to check the feasibility of the approach we performed a series of experiments on three small datasets, MNIST and it variants. We trained a medium size binarized network to perform classification. Then we encoded it as a Boolean formula and added the verification condition formula. We showed that if the resulting formula is unsatisfiable then the local robustness property holds for the input image.

For the majority of benchmarks we showed that the property does not hold and a perturbation that leads to its violation exists. However, for some benchmark images we showed that these images are certifiably $\epsilon$-robust.

The main lesson we learn from this work is that verification of neural networks is a challenging problem for modern decision procedures. Even a medium size network with an input of size 784 results in large formulas that are hard to tackle using a SAT solver. One observation we made is that we can exploit the structure of these encodings to solve the resulting SAT formulas more efficiently based on the idea of *counterexample-guided* search [Clarke *et al.*, 2000; McMillan, 2005; McMillan, 2003]. Namely, the SAT encoding follows the modular structure of the network. Let us illustrate our approach with a simple network consisting of two internal blocks and an output block as in Figure 1. Suppose we want to verify the local robustness property of the network . The network can be encoded as a conjunction of two Boolean formulas: Gen (generator) that encodes the first block of the network, and Ver (verifier) that encodes the rest of the network. The Gen and Ver are embedded in a counterexample-guided search procedure and they communicate via variables shared by the two formulas. This allows us to guide the search procedure and improve performance on some benchmarks.

### 5.3 Invertibility of BNNs

We considered the problem of invertibility of BNNs [Korneev *et al.*, 2018]. We started from a trained BNN that takes an image of porous media and outputs a vector of parameters that describe its physical properties. Images of porous media[2] are black and white images that represent an abstraction of the physical structure. Solid parts are encoded as a set of connected black pixels; a void area is encoded a set of connected white pixels. The given BNN represents an approximation of a partial differential equation solver for computing dispersion coefficients for the given geometry of a porous medium.

We considered the problem of invertibility of a BNN: Given an output vector, can we construct an input image subset to some additional constraints? The physical meaning is to synthesize a new porous media with the given set of properties where properties are defined by the values of dispersion coefficients. In this work we demonstrated that invertibility problem for BNNs can be encoded as an integer linear program where all variables are integers and used ILP and SMT solvers to tackle this problem. We were able to generate images for a small dataset with 16 by 16 pixels images given 3 layered neural network.

## 6 Future work

The area of the formal verification of neural networks is just emerging [Pulina and Tacchella, 2010; Pulina and Tacchella, 2012; Bastani *et al.*, 2016; Huang *et al.*, 2017; Katz *et al.*, 2017; Bunel *et al.*, 2017; Cheng *et al.*, 2017a; Dutta *et al.*, 2017; Tjeng and Tedrake, 2017; Narodytska *et al.*, 2017; Leofante *et al.*, 2018]. There are a number of interesting research directions. First, it is important to build new decision procedures that are tailored for solving problems of verification of neural networks. For example, a promising research direction is to take advantage of the modular structure of neural networks that is naturally preserved in the encoding. Second, we observe that so far research on verification of neural networks is focused on the discriminative problem, e.g. the classification problem. However, to the best of our knowledge, there is no work on analysing generative models formally, like neural networks produced with the generative adversarial framework [Goodfellow *et al.*, 2014]. For example, we need to understand what interesting properties to analyze for these structures are. The third promising research direction is using formal analysis to increase our understanding of the decision making process of neural networks, for example, extracting explanations that support neural network decisions.

[2]Specifically, we are looking at a transitionally periodic "unit cell" of porous medium assuming that porous medium has a periodic structure [Hornung, 1997].

## References

[Bastani *et al.*, 2016] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring Neural Net Robustness with Constraints. In *NIPS'16*, 2016.

[Bau *et al.*, 2017] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. *CoRR*, abs/1704.05796, 2017.

[Bunel *et al.*, 2017] Rudy Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan Kumar. Piecewise linear neural network verification: A comparative study. *CoRR*, abs/1711.00455, 2017.

[Cheng *et al.*, 2017a] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *Automated Technology for Verification and Analysis*, volume 10482, pages 251–268, 2017.

[Cheng *et al.*, 2017b] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Verification of binarized neural networks. *CoRR*, abs/1710.03107, 2017.

[Clarke *et al.*, 2000] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *CAV'00*, pages 154–169. Springer, 2000.

[Dutta *et al.*, 2017] Souradeep Dutta, Susmit Jha, Sriram Sanakaranarayanan, and Ashish Tiwari. Output range analysis for deep neural networks. *CoRR*, abs/1709.09130, 2017.

[Ehlers, 2017] Rudiger Ehlers. Formal verification of piecewise linear feed-forward neural networks. In *ATVA*, pages 269–286, 2017.

[Fischetti and Jo, 2017] Matteo Fischetti and Jason Jo. Deep neural networks as 0-1 mixed integer linear programs: A feasibility study. *CoRR*, abs/1712.06174, 2017.

[Frosst and Hinton, 2017] Nicholas Frosst and Geoffrey E. Hinton. Distilling a neural network into a soft decision tree. In Tarek R. Besold and Oliver Kutz, editors, *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017, Bari, Italy, November 16th and 17th, 2017.*, volume 2071. CEUR-WS.org, 2017.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[Goodfellow *et al.*, 2015] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR'15*, 2015.

[Hornung, 1997] Ulrich Hornung, editor. *Homogenization and Porous Media*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[Huang *et al.*, 2017] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety Verification of Deep Neural Networks. In *CAV'17*, Lecture Notes in Computer Science, pages 3–29. Springer, 2017.

[Hubara *et al.*, 2016] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks. In *Advances in Neural Information Processing Systems 29*, pages 4107–4115. Curran Associates, Inc., 2016.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML'157*, pages 448–456, 2015.

[Katz *et al.*, 2017] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *CAV'17*, pages 97–117, 2017.

[Koh and Liang, 2017] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 2017.

[Korneev *et al.*, 2018] Svyatoslav Korneev, Nina Narodytska, Luca Pulina, Armando Tacchella, Nikolaj Bjorner, and Mooly Sagiv. Constrained image generation using binarized neural networks with decision procedures. *CoRR*, abs/1802.08795, 2018.

[Kung *et al.*, 2017] Jaeha Kung, David Zhang, Gooitzen Van der Wal, Sek Chai, and Saibal Mukhopadhyay. Efficient Object Detection Using Embedded Binarized Neural Networks. *Journal of Signal Processing Systems*, pages 1–14, 2017.

[Leofante *et al.*, 2018] Francesco Leofante, Nina Narodytska, Luca Pulina, and Armando Tacchella. Automated verification of neural networks: Advances, challenges and perspectives. *CoRR*, abs/1805.09938, 2018.

[McDanel *et al.*, 2017] Bradley McDanel, Surat Teerapittayanon, and H. T. Kung. Embedded binarized neural networks. In *EWSN*, pages 168–173. Junction Publishing, Canada / ACM, 2017.

[McMillan, 2003] Kenneth McMillan. Interpolation and SAT-based model checking. In *CAV'03*, volume 3, pages 1–13. Springer, 2003.

[McMillan, 2005] Kenneth L. McMillan. Applications of craig interpolants in model checking. In *TACAS*, volume 3440 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.

[Moosavi-Dezfooli *et al.*, 2016] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *arXiv preprint arXiv:1610.08401*, 2016.

[Narodytska *et al.*, 2017] Nina Narodytska, Shiva Prasad Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. Verifying properties of binarized deep neural networks. *CoRR*, abs/1709.06662, 2017.

[Pulina and Tacchella, 2010] Luca Pulina and Armando Tacchella. An abstraction-refinement approach to verification of artificial neural networks. In *CAV*, pages 243–257, 2010.

[Pulina and Tacchella, 2012] Luca Pulina and Armando Tacchella. Challenging SMT solvers to verify neural networks. *AI Commun.*, 25(2):117–135, 2012.

[Reagen *et al.*, 2017] Brandon Reagen, Robert Adolf, Paul N. Whatmough, Gu-Yeon Wei, and David M. Brooks. *Deep Learning for Computer Architects*. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2017.

[Ribeiro *et al.*, 2016] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM, 2016.

[Simonyan *et al.*, 2013] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[Sinz, 2005] Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. Technical report, University of Tubingen, 2005.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[Tjeng and Tedrake, 2017] Vincent Tjeng and Russ Tedrake. Verifying neural networks with mixed integer programming. *CoRR*, abs/1711.07356, 2017.