EasyChair Preprint

№ 1000

How Many Bits Does it Take to Quantize Your Neural Network?

Mirco Giacobbe, Thomas A. Henzinger and Mathias Lechner

September 12, 2019

How Many Bits Does it Take to Quantize Your Neural Network?

Mirco Giacobbe, Thomas A. Henzinger, Mathias Lechner
IST Austria

Abstract

Quantization converts neural networks into low-bit fixed-point computations which can be carried out by efficient integer-only hardware, and is standard practice for the deployment of neural networks on real-time embedded devices. However, like their real-numbered counterpart, quantized networks are not immune to malicious misclassification caused by adversarial attacks. We investigate how quantization affects a network's robustness to adversarial attacks, which is a formal verification question. We show that neither robustness nor non-robustness are monotonic with changing the number of bits for the representation and, also, neither are preserved by quantization from a real-numbered network. For this reason, we introduce a verification method for quantized neural networks which, using SMT solving over bit-vectors, accounts for their exact, bit-precise semantics. We built a tool and analyzed the effect of quantization on a classifier for the MNIST dataset. We demonstrate that, compared to our method, existing methods for the analysis of real-numbered networks often derive false conclusions about their quantizations, both when determining robustness and when detecting attacks, and that existing methods for quantized networks often miss attacks. Furthermore, we applied our method beyond robustness, showing how the number of bits in quantization enlarges the gender bias of a predictor for students' grades.

1 Introduction

Deep neural networks are powerful machine learning models, and are becoming increasingly popular in software development. Since recent years, they have pervaded our lives: think about the language recognition system of a voice assistant, the computer vision employed in face recognition or self driving, not to talk about many decision-making tasks that are hidden under the hood. However, this also subjects them to the resource limits that real-time embedded devices impose. Mainly, the requirements are low energy consumption, as they often run on batteries, and low latency, both to maintain user engagement and to effectively interact with the physical world. This translates into specializing our computation by reducing memory footprint and set of instructions, to minimize cache misses avoid costly hardware operations. For this purpose, quantization compresses neural networks, which are traditionally run over 32-bit floating-point arithmetic, into computations that only require bit-wise and integer-only arithmetic over small words, e.g., 8 bits. Quantization is the standard technique for the deployment of neural networks on mobile and embedded devices, and is implemented in TensorFlow Lite (Jacob et al. 2018). In this work, we investigate the robustness of quantized networks to adversarial attacks and, more generally, to formal verification questions.

Adversarial attacks are a well-known vulnerability of neural networks (Szegedy et al. 2013). For instance, a self-driving car can be tricked into confusing a stop with a speed limit sign (Evtimov et al. 2017), or a home automation system can be commanded to deactivate the security camera by a voice reciting a poetry (Schönherr et al. 2019). The attack is carried out by superposing the innocuous input with a crafted perturbation that is imperceptible to humans. Formally, the attack lies within the neighborhood of a known-to-be-innocuous input, according to some notion of distance. The fraction of samples (from a large set of test inputs) that do not admit attacks determines the robustness of the network. We ask ourselves how quantization affects networks' robustness or, dually, how many bits it takes to keep robustness above some specific threshold. This amounts to proving that, for a set of given quantizations and inputs, there does not exists an attack, which is a formal verification question.

The formal verification of neural networks has been addressed either by overapproximating—as it happens in abstract interpretation—the space of outputs given a space of attacks, or by searching—as it happens in SMT-solving—for a variable assignment that witnesses an attack. The first category include methods that relax the neural networks into computations over interval arithmetic (Pulina and Tacchella 2010), treat them as hybrid automata (Xiang, Tran, and Johnson 2018), or abstract them directly by using zonotopes, polyhedra (Gehr et al. 2018), or tailored abstract domains (Singh et al. 2019). Overapproximation-based methods are typically fast, but incomplete: they prove robustness but do not produce attacks. On the other hand, methods based on local gradient descent have turned out to be effective in producing attacks in many cases (Moosavi-Dezfooli, Fawzi, and Frossard 2016), but sacrifice formal completeness. Indeed, the search for adversarial attack is NP-complete even for the simplest (i.e., ReLU) networks (Katz et al. 2017), which motivates the rise of methods based on *Satisfiability Modulo Theory* (SMT) and *Mixed Integer Linear Programming* (MILP). SMT-solvers have been shown not to scale beyond

toy examples (20 hidden neurons) on monolithic encodings (Pulina and Tacchella 2012), but today's specialized techniques can handle real-life benchmarks such as, e.g., neural networks for the MNIST dataset. Specialized tools include DLV (Huang et al. 2017), which subdivides the problem into smaller SMT instances, and Planet (Ehlers 2017), which combines different SAT and LP relaxations. Reluplex takes a step further augmenting LP-solving with a custom calculus for ReLU networks (Katz et al. 2017). On the other side of the spectrum, a recent MILP formulation turned out effective using off-the-shelf solvers (Tjeng, Xiao, and Tedrake 2018). Moreover, it posed the basis for Sherlock (Dutta et al. 2018), which couples local search and MILP, and for a specialized branch and bound algorithm (Bunel et al. 2018).

All techniques mentioned above reason about the real-number relaxation of the network. Unfortunately, adversarial attacks computed over the reals are not necessarily attacks on execution architectures, in particular for quantized neural networks. We show that attacks and, more generally, robustness and vulnerability to attacks do not transfer between real and quantized networks and also do not transfer monotonically with the number of bits across quantized networks. As a result, verifying a network with either real or finite precision for the numbers representation may (i) conclude that samples are robust while they admit attacks under a different finite precision (false negative), but also may (ii) find attacks for samples that are instead robust under a different finite precision (false positive); in addition, it may (iii) correctly identify samples as vulnerable but provide invalid attacks, and all three phenomena may happen in either direction and non-monotonically with the number of bits. For this reason, the verification of real-numbered neural networks is inadequate for the analysis of quantized networks, and their analysis needs techniques that account for their exact semantics. Recently, a similar problem has been addressed on binarized neural networks, through SAT-solving (Narodytska et al. 2018). Unfortunately, binarized networks are bound to the special case of 1-bit quantizations. For many-bit quantizations, a methods based on gradient descent has been recently introduced (Zhao et al. 2019). While very efficient (and sound), the method is incomplete and may produce false negatives.

We introduce, for the first time, a complete method for the formal verification of quantized neural networks. Our method accounts for the bit-precise semantics of quantized networks by leveraging the first-order theory of bit vectors without quantifiers (QF_BV) to exactly encode hardware operations such as 2'complementation, bit-shift, integer arithmetic with overflow. On the technical side, we encode multiply-add operations in a balanced fashion, which enabled the SMT-solver to scale up to our benchmarks. As a result, we obtain a monolithic encoding of the verification problem into a first-order logic formula, amenable to modern bit-precise SMT-solving. We built a tool using the SMT-solver Boolector (Niemetz, Preiner, and Biere 2014), we evaluated its performance, compared—favorably—its effective soundness and completeness against Reluplex and gradient descent for quantized networks, while assessing the effect of quantization for different networks and verification questions.

We measured the robustness to attacks of a neural classifier involving 890 neurons and trained after the MNIST dataset (handwritten digits), for quantizations between 6 and 10 bits. First, we observed that our SMT encoding with balancing could compute every attacks within 16 hours, with median time of 3h 41m, while the naive (linear) encoding timed-out on all instances beyond 6 bits. Second, we experimentally confirmed that both Reluplex and gradient descent for quantized networks can produce false conclusions about quantized networks; in particular, spurious results occurred consistently more frequently as the number of bits in quantization lowered. Finally, we discovered that, to achieve an acceptable level of robustness, it takes a higher bit quantization than assessed by standard accuracy measures.

Finally, we applied our method beyond the property of robustness to attacks. We estimated the effect of quantization upon the gender bias emerging from quantized predictors for students' performance in mathematics exams. More precisely, we computed the maximum predictable grade gap between any two student with identical features except for the gender. The experiment showed that a substantial gap existed and was proportionally enlarged by quantization: the lower the number bits the larger the gap.

We summarize our contribution in five points. First, we show that the robustness of quantized neural networks is non-monotonic in the number of bits and that is independent of the robustness of its real-numbered counterpart. Second, we introduce the first complete method for the verification of quantized neural networks. Third, we built a tool and used it to demonstrate that our method, unlike a naive approach, can verify networks with hundreds of neurons. Fourth, we also show that our method determines both robustness and vulnerability of quantized networks more accurately than existing methods, in particular for low-bit quantizations. Fifth, we illustrate how quantization affects the robustness of neural networks, not only with respect to adversarial attacks, but also with respect to other verification questions, specifically fairness in machine learning.

2 Quantization of Feed-forward Networks

A feed-forward neural network consists of a finite set of *neurons* $x_1, \ldots, x_k$ partitioned into a sequence of layers: an *input layer* with n neurons, followed by one or many *hidden layers*, finally followed by an *output layer* with m neurons. Every pair of neurons x_j and x_i in respectively subsequent layers is associated with a *weight* coefficient $w_{ij} \in \mathbb{R}$; if the layer of x_j is not subsequent to that of x_i, then we assume $w_{ij} = 0$. Every hidden or output neuron x_i is associated with a *bias* coefficient $b_i \in \mathbb{R}$. The semantics of the neural network gives to each neuron a real value: upon a valuation for the neurons in the input layer, every other neuron x_i assumes its value according to the update rule

$$x_i = \text{ReLU-}N(b_i + \sum_{j=1}^{k} w_{ij} x_j), \tag{1}$$

where $\text{ReLU-}N \colon \mathbb{R} \to \mathbb{R}$ is the *activation function*. Altogether, the neural network implements a function $f \colon \mathbb{R}^n \to$

$\mathbb{R}^m$ whose result corresponds to the valuation for the neurons in the output layer.

The activation function governs the firing logic of the neurons, layer by layer, by introducing non-linearity in the system. Among the most popular activation functions are purely non-linear functions, such as the tangent hyperbolic and the sigmoidal function, and piece-wise linear functions, better known as *Rectified Linear Units* (ReLU) (Nair and Hinton 2010). ReLU consists of the function that takes the positive part of its argument, i.e., $\mathrm{ReLU}(x) = \max\{x, 0\}$. We consider the variant of ReLU that imposes a cap value N, known as ReLU-N (Krizhevsky and Hinton 2010). Precisely

$$\mathrm{ReLU}\text{-}N(x) = \min\{\max\{x, 0\}, N\}, \tag{2}$$

which can be alternatively seen as a concatenation of two ReLU functions (see Eq. 10). As a consequence, the real-numbered version of all neural networks we treat are full-fledged ReLU networks, whose verification is amenable to state-of-the-art verification tools including Reluplex.

Quantizing consists of converting a neural network over real numbers, which is normally deployed on floating-point architectures, into a neural network over integers, whose semantics corresponds to a computation over fixed-point arithmetic (Jacob et al. 2018). Specifically, fixed-point arithmetic can be carried out by integer-only architectures and possibly over small words, e.g., 8 bits. All numbers are represented in 2's complement over B bits words and F bits are reserved to the fractional part: we call the result a *B-bits quantization in QF arithmetic*. More concretely, the conversion follows from the rounding of weight and bias coefficients to the F-th digit, namely $\bar{b}_i = \mathrm{rnd}(2^F b_i)$ and $\bar{w}_{ij} = \mathrm{rnd}(2^F w_{ij})$ where $\mathrm{rnd}(\cdot)$ stands for any rounding to an integer. Then, the fundamental relation between a quantized value $\bar{a}$ and its real counterpart a is

$$a \approx 2^{-F}\bar{a}. \tag{3}$$

Consequently, the semantics of a quantized neural network corresponds to the update rule in Eq. 1 after substituting of x, w, and b with the respective approximants $2^{-F}\bar{x}$, $2^{-F}\bar{w}$, and $2^{-F}\bar{b}$. Namely, the semantics amounts to

$$\bar{x}_i = \mathrm{ReLU}\text{-}(2^F N)(\bar{b}_i + \mathrm{int}(2^{-F}\sum_{j=1}^{k} \bar{w}_{ij}\bar{x}_j)), \tag{4}$$

where $\mathrm{int}(\cdot)$ truncates the fractional part of its argument or, in other words, rounds towards zero. In summary, the update rule for the quantized semantics consists of four parts. The first part, i.e., the linear combination $\sum_{j=1}^{k} \bar{w}_{ij}\bar{x}_j$, propagates all neurons values from the previous layer, obtaining a value with possibly $2B$ fractional bits. The second scales the result by 2^{-F} truncating the fractional part by, in practice, applying an arithmetic shift to the right of F bits. Finally, the third applies the bias $\bar{b}$ and the fourth clamps the result between 0 and $2^F N$. As a result, a quantize neural network realizes a function $f\colon \mathbb{Z}^n \to \mathbb{Z}^m$, whose evaluation relies on integer-only hardware operations.

We assume all intermediate values, e.g., of the linear combination, to be fully representable as, coherently with the common execution platforms (Jacob et al. 2018), we always allocate enough bits for under and overflow not to happen. Hence, any loss of precision from the respective real-numbered network happens exclusively, at each layer, as a consequence of rounding the result of the linear combination to F fractional bits. Notably, rounding causes the robustness to adversarial attacks of quantized networks with different quantization levels to be independent of one another, and independent of their real counterpart.

Figure 1: Adversarial attack.

3 Robustness is Non-monotonic in the Number of Bits

A neural classifier is a neural network that maps a n-dimensional input to one out of m classes, each of which is identified by the output neuron with the largest value, i.e., for the output values $z_1, \ldots, z_m$, the choice is given by

$$\mathrm{class}(z_1, \ldots, z_m) = \arg\max_i z_i. \tag{5}$$

For example, a classifier for handwritten digits takes in input the pixels of an image and returns 10 outputs $z_0, \ldots, z_9$, where the largest indicates the digit the image represents. An adversarial attack is a perturbation for a sample input

original + perturbation = attack

that, according to some notion of closeness, is indistinguishable from the original, but tricks the classifier into inferring an incorrect class. The attack in Fig. 1 is indistinguishable from the original by the human eye, but induces our classifier to assign the largest value to z_3, rather than z_9, misclassifying the digit as a 3. For this example, misclassification happens consistently, both on the real-numbered and on the respective 8-bits quantized network in Q4 arithmetic. Unfortunately, attacks do not necessarily transfer between real and quantized networks and neither between quantized networks for different precision. More generally, attacks and, dually, robustness to attacks are non-monotonic with the number of bits.

We give a prototypical example for the non-monotonicity of quantized networks in Fig. 2. The network consists of 7 neurons with all weight and bias coefficients fully representable in Q1. For every hidden and output neurons in the top row, we show, respectively from top to bottom, the valuations for the Q3, Q2, and Q1 quantizations of the network (following Eq. 4); in particular, we show their real counterpart $\bar{x}/2^F$ (as in Eq. 3). We evaluate all quantizations and obtain that the valuations for the top output neuron are non-monotonic with the number of fractional bits; in fact, the Q1 dominates the Q3 which dominates the Q2 output. Coincidentally, the valuations for the Q3 quantization correspond to the valuations with real-number precision (i.e., never undergo truncation), indicating that also real and quantized networks are similarly incomparable. Notably, all phenomena

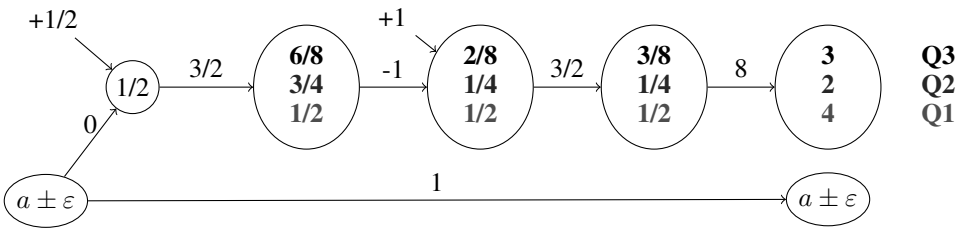


Figure 2: Neural network with non-monotonic robustness w.r.t. its Q1, Q2, and Q3 quantizations.

occur both for quantized networks with rounding towards zero (as we show in the example), and with rounding to the nearest, which is naturally non-monotonic (e.g., 5/16 rounds to 1/2, 1/4, and 3/8 with, resp., Q1, Q2, and Q3). Besides, non-monotonicity of the output causes non-monotonicity of robustness.

An input constitutes an adversarial attack when the output layer violates the maximality of the neuron associated with the class of the original sample. For instance, the attack of Fig. 1 constitutes an attack because it violates $z_3 < z_9$, and 9 is the original class. As for the example in Fig. 2, we suppose the original sample is 3/2 and its class is associated with the top output neuron; notably, the original sample is correctly classified by all quantizations. Assuming attacks can only lay in the neighboring interval $3/2 \pm 1$, we obtain that the Q2 network admits an attack, e.g., the input 5/2. Conversely, for the same neighboring interval, neither of the Q1 and Q3 networks admit an attack, showing that robustness is non-monotonic. Dually, also non-robustness is non-monotonic as, for the sample 9/2 (whose class corresponds to the bottom neuron) and the interval $9/2 \pm 2$, Q2 is robust while both Q3 and Q1 are vulnerable, even though their specific attacks do not always coincide as for, e.g., 7/2.

Robustness and non-robustness are non-monotonic in the number of bits for quantized networks. As a consequence, verifying a high-bits quantization, or a real-valued network, may derive false conclusions about a target lower-bits quantization, in either direction. Specifically, for the question as for whether an attack exists, we may have both (i) false negatives, i.e., the verified network is robust but the target network admits an attack, and (ii) false positives, i.e., the verified network is vulnerable while the target network robust. In addition, we may also have (iii) true positives with invalid attacks, i.e., both are vulnerable but the found attack do not transfer to the target network. For these reasons, we introduce a verification method that accounts for the bit-precise semantics of quantized neural network.

4 Verification of Quantized Networks using Bit-precise SMT-solving

Bit-precise SMT-solving comprises various technologies for deciding the satisfiability of first-order logic formulae, whose variables are interpreted as bit-vectors of fixed size. In particular, it produces satisfying assignments (if any exist) for formulae that include bitwise and arithmetic operators, whose semantics corresponds to that of hardware architectures. For instance, we can encode bit-shifts, 2's complementation, multiplication and addition with overflow, signed and unsigned comparisons. More precisely, this is the quantifier-free first-order theory of bit-vectors (i.e., QF_BV), which we employ to produce a monolithic encoding of the verification problem for quantized neural networks.

A verification problem for the neural networks $f_1, \ldots, f_K$ consists of checking the validity of a statement of the form

$$\varphi(\vec{y}_1, \ldots, \vec{y}_K) \implies \psi(f_1(\vec{y}_1), \ldots, f_K(\vec{y}_K)), \quad (6)$$

where φ is a predicate over the inputs and ψ over the outputs of all networks; in other words, it consists of checking an input–output relation, which generalizes various verification questions, including robustness to adversarial attacks and fairness in machine learning, which we treat in Sec. 5. For the purpose of SMT solving, we encode the verification problem in Eq. 6, which is a validity question, by its dual satisfiability question

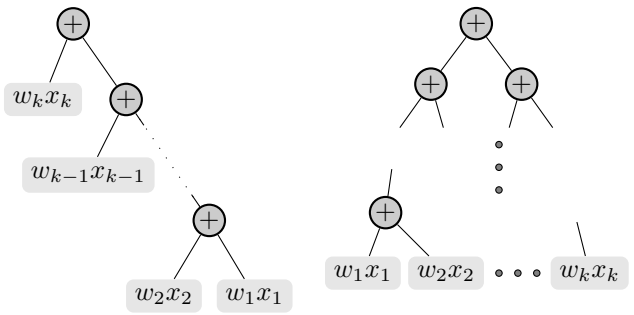
$$\varphi(\vec{y}_1, \ldots, \vec{y}_K) \wedge \bigwedge_{i=1}^{K} f_i(\vec{y}_i) = \vec{z}_i \ \wedge \ \neg\psi(\vec{z}_1, \ldots, \vec{z}_K), \quad (7)$$

whose satisfying assignments constitute counterexamples for the contract. The formula consists of three conjuncts: the rightmost constraints the input within the assumption, the leftmost forces the output to violate the guarantee, while the one in the middle relates inputs and outputs by the semantics of the neural networks.

The semantics of the network consists of the bit-level translation of the update rule in Eq. 4 over all neurons, which we encode in the formula

$$\bigwedge_{i=1}^{k} x_i = \text{ReLU-}(2^F N)(x'_i) \wedge \ x'_i = \bar{b}_i + \texttt{ashr}(x''_i, F) \wedge x''_i = \sum_{j=1}^{k} \bar{w}_{ij} x_j. \quad (8)$$

Each conjunct in the formula employs three variables x, x', and x'' and is made of three, respective, parts. The first part accounts for the operation of clamping between 0 and $2^F N$, whose semantics is given by the formula $\text{ReLU-}M(x) = \texttt{ite}(\texttt{sign}(x), 0, \texttt{ite}(x \geq M, M, x))$. Then, the second part accounts for the operations of scaling and biasing. In particular, it encodes the operation of rounding by truncation scaling, i.e., $\text{int}(2^{-F}x)$, as an arithmetic shift to the right. Finally, the last part accounts for the propagation of values



Linear layout (a) Balanced layout (b)

Figure 3: Abstract syntax trees for alternative encodings of a long linear combination.

from the previous layer, which, despite the obvious optimization of pruning away all monomials with null coefficient, often consists of long linear combinations, whose exact semantic amounts to a sequence of multiply-add operations over an accumulator; particularly, encoding it requires care in choosing variables size and association layout.

The size of the bit-vector variables determines whether overflows can occur. In particular, since every monomial $w_{ij}x_j$ consists of the multiplication of two B-bits variables, its result requires $2B$ bits in the worst case; since summation increases the value linearly, its result requires a logarithmic amount of extra bits in the number of summands (regardless of the layout). Provided that, we avoid overflow by using variables of $2B + \log k$ bits, where k is the number of summands.

The association layout is not unique and, more precisely, varies with the order of construction of the long summation. For instance, naively associating to the right (or to the left) produces a linear layout, as in Fig. 3a. For the verification of quantized networks, we associate by recursively splitting the sum into equal parts, producing a *balanced layout* as in Fig. 3b. While linear and balanced layouts are semantically equivalent, we have observed that, in practice, the second impacted—positively—the performance of the SMT-solver as we discuss in Sec. 5, where we also compare against other methods and investigate different verification questions.

5 Experimental Results

The MNIST dataset consists of 70,000 handwritten digits represented by 28-by-28 pixel images with a single 8-bit grayscale channel. Each sample belongs to exactly one category $\{0, 1, \dots 9\}$, which a machine learning model must predict from the raw pixel values. The MNIST set is split into 60,000 training and 10,000 test samples.

We trained a neural network classifier on MNIST, following a *post-training quantization* scheme (Jacob et al. 2018). First, we trained, using TensorFlow with floating-point precision, a network composed of 784 inputs, 2 hidden layers of size 64, 32 with ReLU-7 activation function and 10 outputs, for a total of 890 neurons. The classifier yielded a *standard accuracy*, i.e., the ratio of samples that are correctly classified out of all samples in the testing set, of 94.7% on the floating-point architecture. Afterwards, we quantized the network with various bit sizes, with the exception of imposing the input layer to be always quantized in 8 bits, i.e., the original precision of the samples. The quantized networks required at least Q3 with 7 total bits to obtain an accuracy above 90% and Q5 with 10 bits to reach 94%. For this reason, we focused our study to the quantizations from 6 and the 10 bits in, respectively, Q2 to Q6 arithmetic.

Robust accuracy or, more simply, *robustness* measure the ratio of robust samples: for the distance $\varepsilon > 0$, a sample a is robust when, for all its perturbations y within that distance, the classifier $\text{class} \circ f$ chooses the original class $c = \text{class} \circ f(a)$. In other words, a is robust if, for all $\vec{y}$

$$|a - \vec{y}|_\infty \leq \varepsilon \implies c = \text{class} \circ f(\vec{y}), \tag{9}$$

where, in particular, the right-hand side can be encoded as $\bigwedge_{j=1}^{m} z_j \leq z_c$, for $\vec{z} = f(\vec{y})$. Robustness is a validity question as in Eq. 6 and any witness for the dual satisfiability question constitutes an adversarial attack. We checked the robustness of our selected networks over the first 300 test samples from the dataset with $\varepsilon = 1$ on the first 200 and $\varepsilon = 2$ on the next 100; in particular, we used the SMT-solver Boolector (Niemetz, Preiner, and Biere 2014), off-the-shelf and without any specific configuration.

Our experiments serve two purposes. The first is evaluating scalability and precision of our approach. As for scalability, we study how encoding layout, i.e., linear or balanced, and number of bits affect the runtime of the SMT-solver. As for precision, we measured the gap between our method and both Reluplex (Katz et al. 2017), a formal verifier for real-numbered networks, and the IFGSM algorithm (Zhao et al. 2019), with respect to the accuracy at identifying robust and vulnerable samples. The second purpose of our experiments is evaluating the effect of quantization on the robustness to attacks of our MNIST classifier and, with an additional experiment, measuring the effect of quantization over the gender fairness of a student grades predictor, demonstrating also the expressivity of our method beyond adversarial attacks.

5.1 Scalability and performance

We ran all our experiments on an Intel Xeon W-2175 CPU, with 64GB memory and 16 hours of time budget per problem instance, which we encoded in the two variants with, resp., linear and balanced layout. With linear layout the solver timed-out on all instances but the smallest networks (6 bits), while with balanced layout it checked all instances with an overall median runtime of 3h 41m and, as shown in Tab. 1, roughly doubling at every bits increase, as also confirmed by the histogram in Fig. 4.

Bits	6	7	8	9	10
Linear	3h 25m	oot	oot	oot	oot
Balanced	18m	1h 29m	3h 41m	5h 34m	8h 58m

Table 1: Median runtimes for bit-exact robustness checks.

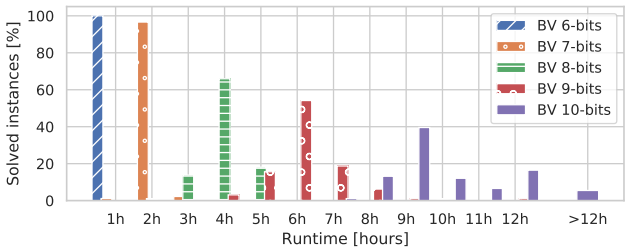


Figure 4: Runtimes for bit-exact robustness checks.

Our results demonstrate that using a balanced association layout improves the performance of the SMT-solver so as to scale up to networks beyond 7 and up to 10 bits, while the linear encoding turned out to be ineffective. Besides, our method tackled networks with 890 neurons which, while small compared to state-of-the-art image classification models, already pose challenging benchmarks for the formal verification task. In the real-numbered world, for instance, off-the-shelf solvers could initially tackle up to 20 neurons (Pulina and Tacchella 2010), and modern techniques, while faster, are often evaluated on networks not beyond 1000 neurons (Katz et al. 2017; Bunel et al. 2018).

Additionally, we pushed our method to its limits, refining our MNIST network to a four-layers deep Convolutional network (2 Conv + 2 Fully-connected layers) with a total of 2238 neurons, which achieved a test accuracy of 98.56%. While for the 6-bits quantization we proved robustness for 99% of the tested samples within a median runtime of 3h 39min, for 7-bits and above all instances timed-out. Notably, Reluplex also failed on the real-numbered version, reporting numerical instability.

5.2 Comparison to other methods

Looking at existing methods for verification, one has two options to verify quantized neural networks: Verifying the real-valued network and hoping the property is preserved when quantizing the network, or relying on incomplete methods and hoping no counterexample is missed. A question that emerges is how accurately are these two approaches for verifying robustness of a quantized network.

To answer this question we used Reluplex (Katz et al. 2017) to prove robustness of the real-valued network. Additionally, we compated against the Iterative Fast Gradient Sign Method (IFGSM), which has recently been proposed to generate ℓ_∞-bounded adversarial attacks for quantized networks (Zhao et al. 2019); notably, IFGSM is incomplete in the sense that it may miss attacks.

Regarding the real-numbered encoding, Reluplex accepts only pure ReLU networks. For this reason, we translate our ReLU-N networks into functionally equivalent ReLU networks, by translating each layer with

$$\text{ReLU-}N(W \cdot \vec{x} + \vec{b}) = \\ \text{ReLU}\Big(-I \cdot \text{ReLU}(-W \cdot \vec{x} - \vec{b} + N)\Big). \quad (10)$$

Out of the 300 samples, at least one method timed out on 56 samples, leaving us with 244 samples whose result was computed over all networks. Tab. 2 depicts how frequently the robustness property could be transferred from the real-valued network to the quantized networks. Non surprisingly, we observed the trend that when increasing the precision of the network, the error between the quantized model and the real-valued model decreases. However, even for the 10-bit model, in 0.8% of the tested samples, verifying the real-valued model leads to a wrong conclusion about the robustness of the quantized network. Moreover, our results show the existence of samples where the 10-bit network is robustness while the real-valued is attackable and vice versa. The invalid attacks illustrate that the higher the precision of the quantization, the more targeted attacks need to be. For instance, while 94% of attacks generated for the real-valued network were also attacks for the 7-bit model, this percentage decrease to 80% for the 10-bit network.

Bits	True negatives	False negatives	False positives	True positives	Invalid attacks
6	66.4%	25.0%	3.3%	5.3%	8%
7	84.8%	6.6%	1.6%	7.0%	6%
8	88.5%	2.9%	0.4%	8.2%	10%
9	91.0%	0.4%	0.4%	8.2%	20%
10	91.0%	0.4%	0.4%	8.2%	20%

Table 2: Transferability of robustness from real-valued network to quantized model.

Next, we compared how well incomplete methods are suited to reason about the robustness of quantized neural networks. We employed IFGSM to attack the 244 test samples for which we obtained the ground-truth robustness and measure how often IFGSM is correct about assessing the robustness of the network. For the sake of completeness, we perform the same analysis for the real-valued network.

Bits	True negatives	False negatives	False positives	True positives
6	69.7%	1.2 %	-	30.3%
7	86.5%	1.6 %	-	13.5%
8	88.9%	0.8 %	-	11.1%
9	91.4%	0.8 %	-	8.6 %
10	91.4%	0 %	-	8.6 %
$\mathbb{R}$	91.4%	0 %	-	8.6 %

Table 3: Transferability of unsound robustness (IFGSM (Zhao et al. 2019)) to ground-truth robustness (ours)

Our results in Tab. 3 present the trend that with higher precision, e.g. 10-bits or reals, incomplete methods provide a stable estimate about the robustness of the network, i.e. IFGSM was able to find attacks for all non-robust samples. However, for lower precision levels, IFGSM missed a substantial amount of attacks, i.e. for the 7-bit network IFGSM could not find a valid attack for 10% of the non-robust samples.

5.3 The effect of quantization on robustness

In Tab. 3 we show how standard accuracy and robust accuracy degrade on our MNIST classifier when increasing the compression level. The data indicates a constant discrepancy between standard accuracy and robustness; for real numbered networks, a similar fact was already known in the literature (Tsipras et al. 2019): we empirically confirm that observation for our quantized networks, whose discrepancy fluctuated between 3 and 4% across all precision levels. Besides, while an acceptable, larger than 90%, standard accuracy was achieved at 7 bits, an equally acceptable robustness was achieved at 9 bits.

Precision	6	7	8	9	10	$\mathbb{R}$
Standard	73.4%	91.8%	92.2%	94.3%	95.5%	94.7%
Robust	69.7%	86.5%	88.9%	91.4%	91.4%	91.4%

Table 4: Accuracy of the MNIST classifiers on the 244 test samples that are checked for robustness.

One relationship not shown in Tab. 3 is that these 4% of non-robust samples are not equal for across quantization levels. For instance, we observed samples that are robust for 7-bit network but attackable when quantizing with 9- and 10-bits. Conversely, there are attacks for the 7-bit networks that are robust samples in the 8-bit network.

5.4 Network specifications beyond robustness

Concerns have been raised that decisions of a ML system could discriminate towards certain groups due to a bias in the training data (Barocas, Hardt, and Narayanan 2017). A key issue in quantifying fairness is that neural networks are black-boxes, and it is hard to explain how each input contributes to certain decisions.

We trained a network on a publicly available dataset consisting of 1000 students' personal information and academic test scores. The personal features include gender, parental level of education, lunch plans, and whether the student took a preparation course for the test, all of which are discrete variables. We train a predictor for students' math scores, which is a discrete variable between 0 and 100. Notably, the dataset contains a potential source for gender bias: the mean math score among females is 63.63, among males is 68.73.

The network we trained is composed of 2 hidden layers with 64 and 32 units respectively. We use an 7-bit quantization-aware training scheme, achieving a 4.14% mean absolute error, i.e., the difference between predicted and actual math score, on the test set.

The network is *fair* if the gender of a person influences the predicted math score by at most the bias β. In other words, checking fairness amounts to verifying that

$$\bigwedge_{i \neq \text{gender}} s_i = t_i \wedge s_{\text{gender}} \neq t_{\text{gender}} \implies |f(\vec{s}) - f(\vec{t})| \leq \beta, \tag{11}$$

is valid over the variables $\vec{s}$ and $\vec{t}$, which respectively model two students for which gender differs but all other features are identical—we call them twin students. When we encode the dual formula, we encode two copies of the semantics of same network: to one copy we give one student $\vec{s}$ and take the respective grade g, to the other we give its twin $\vec{t}$ and take grade h; precisely, we check for the unsatisfiability the negation of formula in Eq. 11. Then, we compute a tight upper bound for the bias, that is the maximum possible change in predicted score for any two twin. To compute the tightest bias, we progressively increase β until our encoded formula becomes unsatisfiable.

We measure mean test error and gender bias of the 6- to the 10-bits quantization of the networks. We show the results in Tab. 5. The test error was stable between 4.1 and 4.6% among all quantizations, showing that the change in precision did not affect the quality of the network in a way that was perceivable by standard measures. However, our formal analysis confirmed a gender bias in the network, producing twins with a 15 to 21 difference in predicted math score. Surprisingly, the bias monotonically increased as the precision level in quantization lowered, indicating to us that quantization plays a role in determining the bias.

Quantization level	Mean test error	Tightest bias upper bound
6 bits	4.46	21
7 bits	4.14	21
8 bits	4.37	16
9 bits	4.38	15
10 bits	4.59	15

Table 5: Results for the formal analysis of the gender bias of a students' grade predictor.

6 Conclusion

We introduced the first complete method for the verification of quantized neural networks which, by SMT solving over bit-vectors, accounts for their bit-precise semantics. We demonstrated, both theoretically and experimentally, that bit-precise reasoning is necessary to accurately ensure the robustness to adversarial attacks of a quantized network. We showed that robustness and non-robustness are non-monotonic in the number of bits for the numerical representation and that, consequently, the analysis of high-bits or real-numbered networks may derive false conclusions about their lower-bits quantizations. Experimentally, we confirmed that real-valued solvers significantly produce spurious results, in particular for low-bit quantizations. Additionally, we showed that gradient descent may also miss attacks on low-bit networks. Besides, we showed that quantization indeed affects, not only robustness, but also other properties of neural networks, such as fairness. We also demonstrated that, using our balanced encoding, off-the-shelf SMT-solving could tackle networks with hundreds of neurons which, despite hitting the limits of current solvers, poses an encouraging baseline for future research.

Acknowledgments

This research was supported in part by the Austrian Science Fund (FWF) under grants S11402-N23(RiSE/SHiNE) and Z211-N23 (Wittgenstein Award).

References

Barocas, S.; Hardt, M.; and Narayanan, A. 2017. Fairness in machine learning. In *Proceeding of NIPS*.

Bunel, R. R.; Turkaslan, I.; Torr, P. H. S.; Kohli, P.; and Mudigonda, P. K. 2018. A unified view of piecewise linear neural network verification. In *NeurIPS*, 4795–4804.

Dutta, S.; Jha, S.; Sankaranarayanan, S.; and Tiwari, A. 2018. Output range analysis for deep feedforward neural networks. In *NFM*, volume 10811 of *Lecture Notes in Computer Science*, 121–138. Springer.

Ehlers, R. 2017. Formal verification of piece-wise linear feed-forward neural networks. In *ATVA*, volume 10482 of *Lecture Notes in Computer Science*, 269–286. Springer.

Evtimov, I.; Eykholt, K.; Fernandes, E.; Kohno, T.; Li, B.; Prakash, A.; Rahmati, A.; and Song, D. 2017. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945* 1.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. T. 2018. AI2: safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy*, 3–18. IEEE.

Huang, X.; Kwiatkowska, M.; Wang, S.; and Wu, M. 2017. Safety verification of deep neural networks. In *CAV (1)*, volume 10426 of *Lecture Notes in Computer Science*, 3–29. Springer.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A. G.; Adam, H.; and Kalenichenko, D. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, 2704–2713. IEEE Computer Society.

Katz, G.; Barrett, C. W.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *CAV (1)*, volume 10426 of *Lecture Notes in Computer Science*, 97–117. Springer.

Krizhevsky, A., and Hinton, G. 2010. Convolutional deep belief networks on cifar-10. *Unpublished manuscript* 40(7).

Moosavi-Dezfooli, S.; Fawzi, A.; and Frossard, P. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *CVPR*, 2574–2582. IEEE Computer Society.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814. Omnipress.

Narodytska, N.; Kasiviswanathan, S. P.; Ryzhyk, L.; Sagiv, M.; and Walsh, T. 2018. Verifying properties of binarized deep neural networks. In *AAAI*, 6615–6624. AAAI Press.

Niemetz, A.; Preiner, M.; and Biere, A. 2014. Boolector 2.0. *JSAT* 9:53–58.

Pulina, L., and Tacchella, A. 2010. An abstraction-refinement approach to verification of artificial neural networks. In *CAV*, volume 6174 of *Lecture Notes in Computer Science*, 243–257. Springer.

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

Schönherr, L.; Kohls, K.; Zeiler, S.; Holz, T.; and Kolossa, D. 2019. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In *accepted for Publication, NDSS*.

Singh, G.; Gehr, T.; Püschel, M.; and Vechev, M. T. 2019. An abstract domain for certifying neural networks. In *POPL*. ACM.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2013. Intriguing properties of neural networks. *CoRR* abs/1312.6199.

Tjeng, V.; Xiao, K. Y.; and Tedrake, R. 2018. Evaluating robustness of neural networks with mixed integer programming.

Tsipras, D.; Santurkar, S.; Engstrom, L.; Turner, A.; and Madry, A. 2019. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*.

Xiang, W.; Tran, H.; and Johnson, T. T. 2018. Output reachable set estimation and verification for multilayer neural networks. *IEEE Trans. Neural Netw. Learning Syst.* 29(11):5777–5783.

Zhao, Y.; Shumailov, I.; Mullins, R.; and Anderson, R. 2019. To compress or not to compress: Understanding the interactions between adversarial attacks and neural network compression. In *SysML Conference*.