Quantization-Aware Interval Bound Propagation for Training Certifiably Robust Quantized Neural Networks

Mathias Lechner[1], Đorđe Žikelić[2], Krishnendu Chatterjee[2], Thomas A. Henzinger[2], Daniela Rus[1]

[1]Massachusetts Institute of Technology (MIT)
[2]Institute of Science and Technology Austria (ISTA)
mlechner@mit.edu

Abstract

We study the problem of training and certifying adversarially robust quantized neural networks (QNNs). Quantization is a technique for making neural networks more efficient by running them using low-bit integer arithmetic and is therefore commonly adopted in industry. Recent work has shown that floating-point neural networks that have been verified to be robust can become vulnerable to adversarial attacks after quantization, and certification of the quantized representation is necessary to guarantee robustness. In this work, we present quantization-aware interval bound propagation (QA-IBP), a novel method for training robust QNNs. Inspired by advances in robust learning of non-quantized networks, our training algorithm computes the gradient of an abstract representation of the actual network. Unlike existing approaches, our method can handle the discrete semantics of QNNs. Based on QA-IBP, we also develop a complete verification procedure for verifying the adversarial robustness of QNNs, which is guaranteed to terminate and produce a correct answer. Compared to existing approaches, the key advantage of our verification procedure is that it runs entirely on GPU or other accelerator devices. We demonstrate experimentally that our approach significantly outperforms existing methods and establish the new state-of-the-art for training and certifying the robustness of QNNs.

Introduction

Quantized neural networks (QNNs) are neural networks that represent their weights and compute their activations using low-bit integer variables. QNNs significantly improve the latency and computational efficiency of inferencing the network for two reasons. First, the reduced size of the weights and activations allows for a much more efficient use of memory bandwidth and caches. Second, integer arithmetic requires less silicon area and less energy to execute than floating-point operations. Consequently, dedicated hardware for running QNNs can be found in GPUs, mobile phones, and autonomous driving computers.

Adversarial attacks are a well-known vulnerability of neural networks that raise concerns about their use in safety-critical applications (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014). These attacks are norm-bounded input perturbations that make the network misclassify samples,

despite the original samples being classified correctly and the perturbations being barely noticeable by humans. For example, most modern image classification networks can be fooled when changing each pixel value of the input image by a few percent. Consequently, researchers have tried to train networks that are provably robust against such attacks. The two most common paradigms of training robust networks are adversarial training (Madry et al. 2018), and abstract interpretation-based training (Mirman, Gehr, and Vechev 2018; Wong and Kolter 2018). Adversarial training and its variations perturb the training samples with gradient descent-based adversarial attacks before feeding them into the network (Madry et al. 2018; Zhang et al. 2019; Wu, Xia, and Wang 2020; Lechner et al. 2021). While this improves the robustness of the trained network empirically, it provides no formal guarantees of the network's robustness due to the incompleteness of gradient descent-based attacking methods, i.e., gradient descent might not find all attacks. Abstract interpretation-based methods avoid this problem by overapproximating the behavior of the network in a forward pass during training. In particular, instead of directly training the network by computing gradients with respect to concrete samples, these algorithms compute gradients of bounds obtained by propagating abstract domains. While the learning process of abstract interpretation-based training is much less stable than a standard training procedure, it provides formal guarantees about the network's robustness. The interval bound propagation (IBP) method (Gowal et al. 2019) effectively showed that the learning process with abstract interpretation can be stabilized when gradually increasing the size of the abstract domains throughout the training process.

Previous work has considered adversarial training and IBP for floating-point arithmetic neural networks, however robustness of QNNs has received comparatively much less attention. Since it was demonstrated by (Giacobbe, Henzinger, and Lechner 2020) that neural networks may become vulnerable to adversarial attacks after quantization even if they have been verified to be robust prior to quantization, one must develop specialized training and verification procedures in order to guarantee robustness of QNNs. Previous works have proposed several robustness *verification* procedures for QNNs (Giacobbe, Henzinger, and Lechner 2020; Baranowski et al. 2020; Henzinger, Lechner, and Žikelić 2021), but none of them consider algorithms for *learning* certifiably robust

QNNs. Furthermore, the existing verification procedures are based on constraint solving and cannot be run on GPU or other accelerating devices, making it much more challenging to use them for verifying large QNNs.

In this work, we present the first abstract interpretation training method for the discrete semantics of QNNs. We achieve this by first defining abstract interval arithmetic semantics that soundly over-approximate the discrete QNN semantics, giving rise to an end-to-end differentiable representation of a QNN abstraction. We then instantiate quantization-aware training techniques within these abstract interval arithmetic semantics in order to obtain a procedure for training certifiably robust QNNs.

Next, we develop a robustness verification procedure which allows us to formally verify QNNs learned via our IBP-based training procedure. We prove that our verification procedure is *complete*, meaning that for any input QNN it will either prove its robustness or find a counterexample. This contrasts the case of abstract interpretation verification procedures for neural networks operating over real arithmetic which are known to be incomplete (Mirman, Baader, and Vechev 2021a). The key advantage of our training and verification procedures for QNNs is that it can make use of GPUs or other accelerator devices. In contrast, the existing verification methods for QNNs are based on constraint solving so cannot be run on GPU.

Finally, we perform an experimental evaluation showing that our method outperforms existing state-of-the-art certified L_∞-robust QNNs. We also elaborate on the limitations of our training method by highlighting how the low precision of QNNs makes IBP-based training difficult.

We summarize our contribution in three points:

- We introduce the first learning procedure for learning robust QNNs. Our learning procedure is based on quantization-aware training and abstract interpretation method.

- We develop the first *complete* robustness verification algorithm for QNNs (i.e., one that always terminates with the correct answer) that runs entirely on GPU or other neural network accelerator devices and make it publicly available.

- We experimentally demonstrate that our method advances the state-of-the-art on certifying L_∞-robustness of QNNs.

Related Work

Abstract interpretations for neural networks   Abstract interpretation is a method for overapproximating the semantics of a computer program in order to make its formal analysis feasible (Cousot and Cousot 1977). Abstract interpretation executes program semantics over abstract domains instead of concrete program states. The method has been adapted to the robustness certification of neural networks by computing bounds on the outputs of neural networks (Wong and Kolter 2018; Gehr et al. 2018; Tjeng, Xiao, and Tedrake 2019). For instance, polyhedra (Katz et al. 2017; Ehlers 2017; Gehr et al. 2018; Singh et al. 2019; Tjeng, Xiao, and Tedrake 2019), intervals (Gowal et al. 2019) , hybrid automata (Xiang, Tran, and Johnson 2018), zonotopes (Singh et al. 2018), convex

relaxations (Dvijotham et al. 2018; Zhang et al. 2020; Wang et al. 2021), and polynomials (Zhang et al. 2018b) have been used as abstract domains in the context of neural network verification. Abstract interpretation has been shown to be most effective for verifying neural networks when directly incorporating them into gradient descent-based training algorithms by optimizing the obtained output bounds as the loss function (Mirman, Gehr, and Vechev 2018; Wong and Kolter 2018; Gowal et al. 2019; Zhang et al. 2020).

Most of the abstract domains discussed above exploit the piecewise linear structure of neural networks, e.g., linear relaxations such as polytopes and zonotopes. However, linear relaxations are less suited for QNNs due to their piecewise-constant discrete semantics.

Verification of quantized neural networks   The earliest work on the verification of QNNs has focused on binarized neural networks (BNNs), i.e., 1-bit QNNs (Hubara et al. 2016). In particular, (Narodytska et al. 2018) and (Cheng et al. 2018) have reduced the problem of BNN verification to boolean satisfiability (SAT) instances. Using modern SAT solvers, the authors were able to verify formal properties of BNNs. (Jia and Rinard 2020) further improve the scalability of BNNs by specifically training networks that can be handled by SAT-solvers more efficiently. (Amir et al. 2021; Lazarus and Kochenderfer 2022) developed satisfiability modulo theories (SMT) approaches for BNN verification.

Verification for many bit QNNs was first reported in (Giacobbe, Henzinger, and Lechner 2020) by reducing the QNN verification problem to quantifier-free bit-vector satisfiability modulo theory (QF_BV SMT). The SMT encoding was further improved in (Henzinger, Lechner, and Žikelić 2021) by removing redundancies from the SMT formulation. (Baranowski et al. 2020) introduced fixed-point arithmetic SMT to verify QNNs. The works of (Sena et al. 2021, 2022) have studied SMT-based verification for QNNs as well. Recently, (Mistry, Saha, and Biswas 2022) proposed encoding of the QNN verification problem into a mixed-integer linear programming (MILP) instance. IntRS (Lin et al. 2021) considers the problem of certifying adversarial robustness of quantized neural networks using randomized smoothing. IntRS is limited to L_2-norm bounded attacks and only provides statistical instead of formal guarantees compared to our approach.

Decision procedures for neural network verification Early work on the verification of floating-point neural networks has employed off-the-shelf tools and solvers. For instance, (Pulina and Tacchella 2012; Katz et al. 2017; Ehlers 2017) employed SMT-solvers to verify formal properties of neural networks. Similarly, (Tjeng, Xiao, and Tedrake 2019) reduces the verification problem to mixed-integer linear programming instances. Procedures better tailored to neural networks are based on branch and bound algorithms (Bunel et al. 2018). In particular, these algorithms combine incomplete verification routines (bound) with divide-and-conquer (branch) methods to tackle the verification problem. The speedup advantage of these methods comes from the fact that the bounding methods can be easily implemented on GPU and other accelerator devices (Serre et al. 2021; Wang et al. 2021).

Quantization-aware training   There are two main strategies for training QNNs: post-training quantization and quantization-aware training (Krishnamoorthi 2018). In post-training quantization, a standard neural network is first trained using floating-point arithmetic, which is then translated to a quantized representation by finding suitable fixed-point format that makes the quantized interpretation as close to the original network as possible. Post-training quantization usually results in a drop in the accuracy of the network with a magnitude that depends on the specific dataset and network architecture.

To avoid a significant reduction in accuracy caused by the quantization in some cases, *quantization-aware training (QAT)* models the imprecision of the low-bit fixed-point arithmetic already during the training process, i.e., the network can adapt to a quantized computation during training. The rounding operations found in the semantics of QNNs are non-differentiable computations. Consequently, QNNs cannot be directly trained with stochastic gradient descent. Researchers have come up with several ways of circumventing the problem of non-differentiable rounding. The most common approach is the straight-through gradient estimator (STE) (Bengio, Léonard, and Courville 2013; Hubara et al. 2017). In the forward pass of a training step, the STE applies rounding operations to computations involved in the QNN, i.e., the weights, biases, and arithmetic operations. However, in the backward pass, the rounding operations are removed such that the error can backpropagate through the network. The approach of (Gupta et al. 2015) uses stochastic rounding that randomly selects one of the two nearest quantized values for a given floating-point value. Relaxed quantization (Louizos et al. 2019) generalizes stochastic rounding by replacing the probability distribution over the nearest two values with a distribution over all possible quantized values. DoReFA-Net (Zhou et al. 2016) combines the straight-through gradient estimator and stochastic rounding to train QNN with high accuracy. The authors observed that quantizing the first and last layer results in a significant drop in accuracy, and therefore abstain from quantizing these two layers.

Instead of having a fixed pre-defined quantization range, i.e., fixed-point format, more recent QAT schemes allow learning the quantization range. PACT (Choi et al. 2018) treats the maximum representable fixed-point value as a free variable that is learned via stochastic gradient descent using a straight-through gradient estimation. The approach of (Jacob et al. 2018) keeps a moving average of the values ranges during training and adapts the quantization range according to the moving average. LQ-Nets (Zhang et al. 2018a) learn an arbitrary set of quantization levels in the form of a set of coding vectors. While this approach provides a better approximation of the real-valued neural network than fixed-point-based quantization formats, it also prevents the use of efficient integer arithmetic to run the network. MobileNet (Howard et al. 2019) is a specialized network architecture family for efficient inference on the ImageNet dataset and employs quantization as one technique to achieve this target. HAWQ-V3 (Yao et al. 2021) dynamically assigns the number of bits of each layer to either 4-bit, 8-bit, or 32-bit depending on how numerically sensitive the layer is. EfficientNet-lite

(Tan and Le 2019) employs a neural architecture search to automatically find a network architecture that achieves high accuracy on the ImageNet dataset while being fast for inference on a CPU.

Preliminaries

Quantized neural networks (QNNs)   Feedforward neural networks are functions $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$ that consist of several sequentially composed layers $f_\theta = l_1 \circ \ldots \circ l_s$, where layers are parametrized by the vector θ of neural network parameters. Quantization is an interpretation of a neural network f_θ that evaluates the network over a fixed point arithmetic and operates over a restricted set of bitvector inputs (Smith et al. 1997), e.g. 4 or 8 bits. Formally, given an admissible input set $\mathcal{I} \subseteq \mathcal{R}^n$, we define an interpretation map

$$[\![\cdot]\!]_\mathcal{I} : (\mathbb{R}^n \to \mathbb{R}^m) \to (\mathcal{I} \to \mathbb{R}^m),$$

which maps a neural network to its interpretation operating over the input set $\mathcal{I}$. For instance, if $\mathcal{I} = \mathbb{R}^n$ then $[\![f_\theta]\!]_\mathbb{R}$ is the idealized real arithmetic interpretation of f_θ, whereas $[\![f]\!]_{\text{float32}}$ denotes its floating-point 32-bit implementation (Kahan 1996). Given $k \in \mathbb{N}$, the k-bit *quantization* is then an interpretation map $[\![\cdot]\!]_{\text{int-}k}$ which uses k-bit fixed-point arithmetic. We say that $[\![f_\theta]\!]_{\text{int-}k}$ is a k-bit *quantized neural network (QNN)*.

The semantics of the QNN $[\![f_\theta]\!]_{\text{int-}k}$ are defined as follows. Let $[\mathbb{Z}]_k = \{0, 1\}^k$ be the set of all bit-vectors of bit-width k. The QNN $[\![f_\theta]\!]_{\text{int-}k}$ then also consists of sequentially composed layers $[\![f_\theta]\!]_{\text{int-}k} = l_1 \circ \ldots \circ l_s$, where now each layer is a function $l_i : [\mathbb{Z}]_k^{n_i} \to [\mathbb{Z}]_k^{n_{i+1}}$ that operates over k-bit bitvectors and is defined as follows:

$$x_i' = \sum_{j=1}^{n_i} w_{ij} x_j + b_i, \tag{1}$$

$$x_i'' = \text{round}(x_i', M_i) = \lfloor x_i' \cdot M_i \rfloor, \qquad \text{and} \tag{2}$$

$$y_i = \sigma_i(\min\{2^{N_i} - 1, x_i''\}), \tag{3}$$

Here, $w_{i,j} \in [\mathbb{Z}]_k^{n_i}$ and $b_i \in [\mathbb{Z}]_k^{n_i}$ for each $1 \le j \le n_i$ and $1 \le i \le n_0$ denote the weights and biases of f which are also bitvectors of appropriate dimension. Note that it is a task of the training procedure to ensure that trained weights and biases are bitvectors, see below. In eq. (1), the linear map defined by weights $w_{i,j}$ and biases b_i is applied to the input values x_j. Then, eq. (2) multiplies the result of eq. (1) by M_i and takes the floor of the obtained result. This is done in order to scale the result and round it to the nearest valid fixed-point value, for which one typically uses M_i of the form 2^{-k} for some integer k. Finally, eq. (3) applies an activation function σ_i to the result of eq. (2) where the result is first "cut-off" if it exceeds $2^{N_i} - 1$, i.e., to avoid integer overflows, and then passed to the activation function. We restrict ourselves to *monotone* activation functions, which will be necessary for our IBP procedure to be correct. This is still a very general assumption which includes a rich class of activation function, e.g. ReLU, sigmoid or tanh activation functions. Furthermore, similarly to most quantization-aware training procedures our method assumes that it is provided with *quantized* versions of these activation functions that operate over bit-vectors.

Adversarial robustness for QNNs We now formalize the notion of adversarial robustness for QNN classifiers. Let $[\![f_\theta]\!]_{\text{int-}k} : [\mathbb{Z}]_k^n \to [\mathbb{Z}]_k^m$ be a k-bit QNN. It naturally defines a classifier with m classes by assuming that it assigns to an input $x \in [\mathbb{Z}]_k^n$ a label of the maximal output neuron on input value x, i.e. $y = \text{class}(x) = \text{argmax}_{1 \le i \le m}[\![f_\theta]\!]_{\text{int-}k}(x)[i]$ with $[\![f_\theta]\!]_{\text{int-}k}(x)[i]$ being the value of the i-th output neuron on input value x. If the maximum is attained at multiple output neurons, we assume that argmax picks the smallest index $1 \le i \le m$ for which the maximum is attained.

Intuitively, a QNN is adversarially robust at a point x if it assigns the same class to every point in some neighbourhood of x. Formally, given $\epsilon > 0$, we say that $[\![f_\theta]\!]_{\text{int-}k}$ is ϵ-*adversarially robust* at point x if

$$\forall x' \in [\mathbb{Z}]_k^n . \ ||x - x'||_\infty < \epsilon \Rightarrow \text{class}(x') = \text{class}(x),$$

where $||\cdot||_\infty$ denotes the L_∞-norm. Then, given a finite dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_{|\mathcal{D}|}, y_{|\mathcal{D}|})\}$ with $x_i \in [\mathbb{Z}]_k^n$ and $y_i \in [\mathbb{Z}]_k^m$ for each $1 \le i \le |\mathcal{D}|$, we say that $[\![f_\theta]\!]_{\text{int-}k}$ is ϵ-*adversarially robust* with respect to the dataset $\mathcal{D}$ if it is ϵ-adversarially robust at each datapoint in $\mathcal{D}$.

Quantization-aware Interval Bound Propagation

In this section, we introduce an end-to-end differentiable abstract interpretation method for training certifiably robust QNNs. We achieve this by extending the interval bound propagation (IBP) method of (Gowal et al. 2019) to the discrete semantic of QNNs. Our resulting quantization-aware interval bound propagation method (QA-IBP) trains an interval arithmetic abstraction of a QNN via stochastic gradient descent by propagating upper and lower bounds for each layer instead of concrete values.

First, we replace each layer l_i with two functions $\underline{l_i}, \overline{l_i}$: $[\mathbb{Z}]_k^{n_i} \to [\mathbb{Z}]_k^{n_0}$ defined as follows:

$$\mu_j = \frac{\overline{x_j} + \underline{x_j}}{2} \qquad\qquad r_j = \frac{\overline{x_j} - \underline{x_j}}{2} \qquad (4)$$

$$\mu_i = \sum_{j=1}^{n_i} w_{ij}\mu_j + b_i \qquad r_i = \sum_{j=1}^{n_i} |w_{ij}|r_j \quad (5)$$

$$\underline{x_i'} = \mu_i - r_i \qquad\qquad \overline{x_i'} = \mu_i + r_i \qquad (6)$$

$$\underline{x_i''} = \text{round}(\underline{x_i'}, k_i) = \lfloor \underline{x_i'} \cdot M_i \rfloor \qquad (7)$$

$$\overline{x_i''} = \text{round}(\overline{x_i'}, k_i) = \lfloor \overline{x_i'} \cdot M_i \rfloor \qquad (8)$$

$$\underline{y_i} = \max\{0, \min\{2^{N_i} - 1, \underline{x_i''}\}\} \qquad (9)$$

$$\overline{y_i} = \max\{0, \min\{2^{N_i} - 1, \overline{x_i''}\}\}. \qquad (10)$$

As with standard QNNs, $w_{i,j} \in [\mathbb{Z}]_k^{n_i}$ and $b_i \in [\mathbb{Z}]_k^{n_i}$ for each $1 \le j \le n_i$ and $1 \le i \le n_0$ denote the weights and biases of f which are also bit-vectors of appropriate dimension. By the sequential composition of all layers of the QNN we get the IBP representation of the QNN in the form of two functions $[\![f_\theta]\!]_{\text{int-}k}$ and $\overline{[\![f_\theta]\!]_{\text{int-}k}}$. For a given input sample x and $\epsilon > 0$ we can use the IBP representation of $[\![f_\theta]\!]_{\text{int-}k}$ to potentially prove the adversarial robustness of the QNN. In particular, the input sample defines an abstract
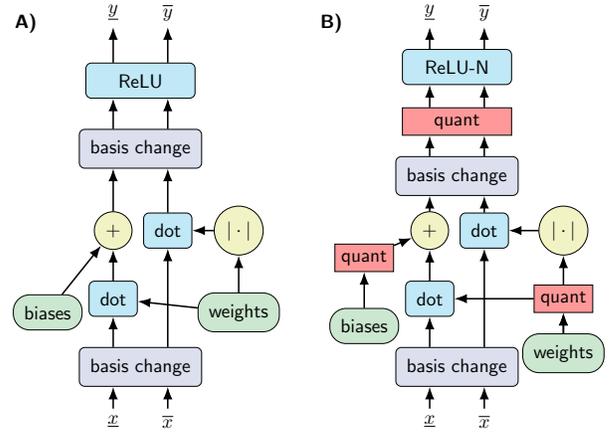
Figure 1: Illustration of how the interval bound propagation training graph is affected by quantization-aware training. A) Standard IBP inference graph. B) IBP inference path with fake quantization operations inserted to model quantized weights, biases, and computations.

interval domain $(\underline{x}, \overline{x})$ with $\underline{x} = x - \epsilon$ and $\overline{x} = x + \epsilon$. Next, we propagate the abstract domains through the IBP representation of the network to obtain output bounds $\underline{y} = [\![f_\theta]\!]_{\text{int-}k}(\underline{x}, \overline{x})$ and $\overline{y} = \overline{[\![f_\theta]\!]_{\text{int-}k}}(\underline{x}, \overline{x})$. Finally, we know that $[\![f_\theta]\!]_{\text{int-}k}$ is ϵ-*adversarially robust* at point x if

$$\underline{y_i} > \overline{y_j}, \qquad \text{for } i = \text{class}(x) \text{ and } \forall j \ne \text{class}(x). \quad (11)$$

End-to-end differentiation We modify the IBP representation of QNNs described above to allow an end-to-end differentiation necessary for a stochastic gradient descent-based learning algorithm. In particular, first, we apply the straight-through gradient estimator to propagate the error backward through the rounding operations in Eq. 8. We do this by replacing the non-differentiable rounding operation with the fake quantization function

$$\text{fake\_quant}(x_i) := \text{round}(x_i, k_i) \qquad (12)$$

$$\frac{\partial \text{ fake\_quant}(x_i)}{\partial x_i} := 1. \qquad (13)$$

We also add fake quantization operations around the weights and biases in Eq. 6. The modified training graph is visualized in Figure 1.

For a single training sample (x, j), we define the per-sample training loss

$$L(\underline{y}, \overline{y}, j) = \sum_{i \ne j} (\overline{y_i} - \underline{y_j}) \mathbb{1}[\underline{y_j} - \overline{y_i} \le 0], \qquad (14)$$

where $\underline{y}, \overline{y}$ are the QA-IBP output bounds QNN with respect to the input domain $(x - \epsilon, x + \epsilon)$ and j corresponds to the label, i.e., class j. The loss term encourages the QA-IBP to produce output bounds that prove adversarial robustness of the QNN $[\![f_\theta]\!]_{\text{int-}k}$ with respect to the input x and adversarial radius ϵ.

Existence of robust QNNs We conclude this section by presenting an interesting result on the existence of robust QNNs. In particular, given $\epsilon > 0$ and a finite dataset of 1-dimensional bit-vectors whose any two distinct datapoints are at least 2ϵ away, we prove that there exists a QNN with ReLU activations that is ϵ-robust with respect to the dataset. Furthermore, we provide an upper bound on the number of neurons that the QNN must contain. We assume 2ϵ distance simply for the ϵ-neighbourhoods of datapoints to be disjoint so that robustness cannot impose contradicting classification conditions.

Note that this result *does not* hold for real arithmetic feedforward neural networks with ReLU or any other continuous activation functions. Indeed, it was observed in (Mirman, Baader, and Vechev 2021b, Corollary 5.12) that a dataset $\{(-2, -1), (0, 1), (2, -1)\}$ cannot be 1-robustly classified by a feed-forward neural network which uses continuous activation functions. The intuition behind this impossibility result for real arithmetic networks is that a classifier would have to be a continuous function that correctly classifies all points in some open neighbourhoods of $x = -1$ and $x = 1$.

Theorem 1. *Let* $\epsilon > 0$ *and let* $\mathcal{D} = \{(x_1, y_1), \ldots, (x_{|\mathcal{D}|}, y_{|\mathcal{D}|})\}$ *with* $x_i \in [\mathbb{Z}]_k^1$ *and* $y_i \in [\mathbb{Z}]_k^1$ *for each* $1 \le i \le |\mathcal{D}|$. *Suppose that* $||x_i - x_j||_\infty \ge 2\epsilon$ *for each* $i \ne j$. *Then, there exists a QNN* $[\![f_\theta]\!]_{\text{int-}k}$ *which is ϵ-robust with respect to the dataset $\mathcal{D}$ and which consists of* $\mathcal{O}(|\mathcal{D}| \cdot \lceil \epsilon \cdot 2^k + 1 \rceil)$ *neurons.*

The proof of Theorem 1 is provided in the extended version of the paper (Lechner et al. 2022). It starts with an observation that the set $\mathcal{B}_\epsilon(x_i) = \{x' \in [\mathbb{Z}]_k^1 \mid ||x' - x_i||_\infty < \epsilon\}$ is finite and consists of at most $\lceil 2\epsilon \cdot 2^k + 1 \rceil$ bit-vectors for each $1 \le i \le |\mathcal{D}|$. Hence, constructing a QNN $[\![f_\theta]\!]_{\text{int-}k}$ that is ϵ-robust with respect to the dataset $\mathcal{D}$ is equivalent to constructing a QNN that correctly classifies $\mathcal{D}' = \cup_{i=1}^{|\mathcal{D}|}\{(x', y_i) \mid x' \in \mathcal{B}_\epsilon(x_i)\}$ which consists of $\mathcal{O}(|\mathcal{D}| \cdot \lceil \epsilon \cdot 2^k + 1 \rceil)$ datapoints. In the extended version of the paper (Lechner et al. 2022), we then design the QNN $[\![f_\theta]\!]_{\text{int-}k}$ that correctly classifies a dataset and consists of at most linearly many neurons in the dataset size. Our construction also implies the following corollary.

Corollary 1. *Let* $\mathcal{D} = \{(x_1, y_1), \ldots, (x_{|\mathcal{D}|}, y_{|\mathcal{D}|})\}$ *with* $x_i \in [\mathbb{Z}]_k^1$ *and* $y_i \in [\mathbb{Z}]_k^1$ *for each* $1 \le i \le |\mathcal{D}|$. *Then, there exists a QNN* $[\![f_\theta]\!]_{\text{int-}k}$ *which correctly classifies the dataset, i.e.* $[\![f_\theta]\!]_{\text{int-}k}(x_i) = y_i$ *for each* $1 \le i \le |\mathcal{D}|$, *and which consists of* $\mathcal{O}(|\mathcal{D}|)$ *neurons.*

A Complete Decision Procedure for QNN Verification

In the previous section, we presented a quantization-aware training procedure for QNNs with robustness guarantees which was achieved by extending IBP to quantized neural network interpretations. We now show that IBP can also be used towards designing a *complete* verification procedure for already trained feed-forward QNNs. By completeness, we mean that the procedure is guaranteed to return either that the QNN is robust or to produce an adversarial attack.

There are two important novel aspects of the verification procedure that we present in this section. First, to the best of our knowledge this is the first complete robustness verification procedure for QNNs that is applicable to networks with non-piecewise linear activation functions. Existing constraint solving based methods that reduce verification to SMT-solving are complete but they only support piecewise linear activation fucntions such as ReLU (Krizhevsky and Hinton 2010). These could in theory be extended to more general activation functions by considering more expressive satisfiability modulo theories (Clark and Cesare 2018), however this would lead to inefficient verification procedures and our experimental results in the following section already demonstrate the significant gain in scalability of our IBP-based methods as opposed to SMT-solving based methods for ReLU networks. Second, we note that while our IBP-based verification procedure is complete for QNNs, in general it is known that existing IBP-based verification procedures for real arithmetic neural networks are *not complete* (Mirman, Baader, and Vechev 2021a). Thus, our result leads to an interesting contrast in IBP-based robustness verification procedures for QNNs and for real arithmetic neural networks.

Verification procedure We now describe our robustness verification procedure for QNNs. Its pseudocode is shown in Algorithm 1. Since verifying ϵ-robustness of a QNN with respect to some finite dataset $\mathcal{D}$ and $\epsilon > 0$ is equivalent to verifying ϵ-robustness of the QNN at each datapoint in $\mathcal{D}$, Algorithm 1 only takes as inputs a QNN $[\![f_\theta]\!]_{\text{int-}k}$ that operates over bit-vectors of bit-width k, a single datapoint $x \in [\mathbb{Z}]_k^n$ and a robustness radius $\epsilon > 0$. It then returns either ROBUST if $[\![f_\theta]\!]_{\text{int-}k}$ is verified to be ϵ-robust at $x \in [\mathbb{Z}]_k^n$, or VULNERABLE if an adversarial attack $||x' - x||_\infty < \epsilon$ with $\text{class}(x') = \text{class}(x)$ is found.

The algorithm proceeds by initializing a stack D of abstract intervals to contain a single element $\{(x - \epsilon \cdot \mathbf{1}, x + \epsilon \cdot \mathbf{1})\}$, where $\mathbf{1} \in [\mathbb{Z}]_k^n$ is a unit bit-vector of bit-width k. Intuitively, D contains all abstract intervals that may contain concrete adversarial examples but have not yet been processed by the algorithm. The algorithm then iterates through a loop which in each loop iteration processes the top element of the stack. Once the stack is empty and the last loop iteration terminates, Algorithm 1 returns ROBUST.

In each loop iteration, Algorithm 1 pops an abstract interval $(\underline{x}, \overline{x})$ from D and processes it as follows. First, it uses IBP for QNNs that we introduced before to propagate $(\underline{x}, \overline{x})$ in order to compute an abstract interval $(\underline{y}, \overline{y})$ that overapproximates the set of all possible outputs for a concrete input point in $(\underline{x}, \overline{x})$. The algorithm then considers three cases. First, if $(\underline{y}, \overline{y})$ does not violate Equation (11) which characterizes violation of robustness by a propagated abstract interval, Algorithm 1 concludes that the abstract interval $(\underline{x}, \overline{x})$ does not contain an adversarial example and it proceeds to processing the next element of D. Second, if $(\underline{y}, \overline{y})$ violates Equation (11), the algorithm uses projected gradient descent restricted to $(\underline{x}, \overline{x})$ to search for an adversarial example and returns VULNERABLE if found. Note that the adversarial attack is generated with respect to the quantization-aware representation of the network, thus ensuring that the input

---

Algorithm 1: QNN robustness verification procedure

1: **Input** QNN $\llbracket f_\theta \rrbracket_{\text{int-}k}$, datapoint $x \in [\mathbb{Z}]_k^n$, robustness radius $\epsilon > 0$
2: **Output** ROBUST or VULNERABLE
3: $D \leftarrow \{(x - \epsilon \cdot \mathbf{1}, x + \epsilon \cdot \mathbf{1})\}$
4: **while** $D \neq \{\}$ **do**
5: $\quad (\underline{x}, \overline{x}) \leftarrow$ pop item from D
6: $\quad (\underline{y}, \overline{y}) \leftarrow \underline{\llbracket f_\theta \rrbracket_{\text{int-}k}}(\underline{x}, \overline{x}), \overline{\llbracket f_\theta \rrbracket_{\text{int-}k}}(\underline{x}, \overline{x})$ propagated via IBP
7: $\quad$ **if** $(\underline{y}, \overline{y})$ violates Equation (11) **then**
8: $\quad\quad$ Try to generate adversarial example using projected gradient descent restricted to $(\underline{x}, \overline{x})$
9: $\quad\quad$ **if** adversarial example found **then**
10: $\quad\quad\quad$ **return** VULNERABLE
11: $\quad\quad$ **else**
12: $\quad\quad\quad (\underline{x'}, \overline{x'}), (\underline{x''}, \overline{x''}) \leftarrow$ partition of $(\underline{x}, \overline{x})$ into two abstract subintervals
13: $\quad\quad\quad D \leftarrow D \cup \{(\underline{x'}, \overline{x'}), (\underline{x''}, \overline{x''})\}$
14: $\quad\quad$ **end if**
15: $\quad$ **end if**
16: **end while**
17: **return** ROBUST

---

space corresponds to valid quantized inputs. Third, if $(\underline{y}, \overline{y})$ violates Equation (11) but the adversarial attack could not be found by projected gradient descent, the algorithm refines the abstract interval $(\underline{x}, \overline{x})$ by splitting it into two smaller subintervals. This is done by identifying

$$i^* = \text{argmax}_{1 \le i \le n}(\overline{x}[i] - \underline{x}[i])$$

and splitting the abstract interval $(\underline{x}, \overline{x})$ along the i^*-th dimension into two abstract subintervals $(\underline{x'}, \overline{x'}), (\underline{x''}, \overline{x''})$, which are both added to the stack D.

Correctness, termination and completeness The following theorem establishes that Algorithm 1 is complete, that it terminates on every input and that it is a complete robustness verification procedure. The proof is provided in the extended version of the paper (Lechner et al. 2022).

Theorem 2. *If Algorithm 1 returns* ROBUST *then* $\llbracket f_\theta \rrbracket_{int\text{-}k}$ *is ϵ-robust at $x \in [\mathbb{Z}]_k^n$. On the other hand, if Algorithm 1 returns* VULNERABLE *then there exists an adversarial attack* $||x' - x||_\infty < \epsilon$ *with* $\text{class}(x') \neq \text{class}(x)$. *Therefore, Algorithm 1 is correct. Furthermore, Algorithm 1 terminates and is guaranteed to return an output on any input. Since Algorithm 1 is correct and it terminates, we conclude that it is also complete.*

Experiments

We perform an experimental evaluation to assess the effectiveness of our quantization-aware interval bound propagation (QA-IBP). In particular, we first use our training procedure to train two QNNs. We then use our complete verification procedure to verify robustness of trained QNNs and we compare our method to the existing verification methods for QNNs of (Giacobbe, Henzinger, and Lechner 2020) and (Henzinger,

Lechner, and Žikelić 2021). Our full experimental setup and code can be found on GitHub [1].

We train two CNNs with QA-IBP using an 8-bit quantization scheme for both weights and activations on the MNIST (LeCun et al. 1998) and Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017) datasets. We quantize all layers; however, we note that our approach is also compatible with mixing quantized and non-quantized layers. Our MNIST network consists of five convolutional layers, followed by two fully-connected layers. Our Fashion-MNIST network contains three convolutional layers and two fully-connected layers. We apply a fixed pre-defined fixed-point format on both weights and activations. Further details on the network architectures can be found in the extended version of the paper (Lechner et al. 2022). We use the Adam optimizer (Kingma and Ba 2015) with a learning rate of 10^{-4} with decoupled weight decay of 10^{-4} (Loshchilov and Hutter 2019) and a batch size of 512. We train our networks for a total of 5,000,000 gradient steps on a single GPU. We linearly scale the value of ϵ during the QA-IBP training from 0 to 4 and apply the elision of the last layer optimization as reported in (Gowal et al. 2019). We pre-train our networks for 5000 steps in their non-IBP representation with a learning rate of $5 \cdot 10^{-4}$.

After training, we certify all test samples using Algorithm 1 with a timeout of 20s per sample. We certify for L_∞-robustness with radii 1 and 4 to match the experimental setup of (Henzinger, Lechner, and Žikelić 2021). We report and compare the certified robust accuracy of the trained networks to the values reported in (Henzinger, Lechner, and Žikelić 2021).

The results in Table 1 demonstrate that our QA-IBP significantly improves the certified robust accuracy for QNNs on both datasets.

Regarding the runtime of the verification step, both (Giacobbe, Henzinger, and Lechner 2020) and (Henzinger, Lechner, and Žikelić 2021) certified a smaller model compared to our evaluation. In particular, (Giacobbe, Henzinger, and Lechner 2020) report a mean runtime of their 8-bit model to be over 3 hours, and (Henzinger, Lechner, and Žikelić 2021) report the mean runtime of their 6-bit model to be 90 and 49 seconds for MNIST/Fashion-MNIST, respectively. Conversely, our method was evaluated with a timeout of 20 seconds and tested on larger networks than the two existing approaches, showing the efficiency of our approach.

Ablation Analysis

In this section, we assess the effectiveness of Algorithm 1 for certifying robustness compared to an incomplete verification based on the QA-IBP output bounds alone. In particular, we perform an ablation and compare Algorithm 1 to an incomplete verification baseline. Our baseline consists of checking the bounds obtained by QA-IBP as an incomplete verifier combined with projected gradient descent (PGD) as an incomplete falsifier, i.e., we try to certify robustness via QA-IBP and simultaneously try to generate an adversarial attack using PGD. In particular, this baseline resembles a version of Algorithm 1 without any branching into subdomains. We carry our

---

[1]https://github.com/mlech26l/quantization_aware_ibp

Method	MNIST			Fashion-MNIST		
	$\epsilon = 0$	$\epsilon = 1$	$\epsilon = 4$	$\epsilon = 0$	$\epsilon = 1$	$\epsilon = 4$
QF_BV SMT (Giacobbe, Henzinger, and Lechner 2020)	97.1%	92%	0%	85.6%	44%	0%
QF_BV SMT (Henzinger, Lechner, and Žikelić 2021)	97.1%	**99%**[*]	53%	85.6%	76%	36%
QA-IBP (ours)	99.2%	**98.8%**	**95.6%**	86.3%	**80.0%**	**59.8%**

Table 1: Certified robust accuracy of a convolutional neural network trained with QA-IBP compared to existing methods for certifying L_∞-robustness of quantized neural networks reported in the literature. [*] Note that (Henzinger, Lechner, and Žikelić 2021) certified only a subset of the test set due to a high per-sample runtime of their approach. Due to this choice they reported a higher $\epsilon = 1$ robust accuracy than clean accuracy.

Weight decay	CIFAR-10		
	$\epsilon = 0$	$\epsilon = 1$	$\epsilon = 4$
$1 \cdot 10^{-4}$	30.0%	28.4%	**22.4%**
$5 \cdot 10^{-5}$	39.6%	**34.1%**	20.6%
$1 \cdot 10^{-5}$	**83.1%**	0%	0%

Table 2: Robust accuracy of our convolutional neural network trained with QA-IBP on the CIFAR-10 dataset with various values for the weight decay. The results express the *robustness-accuracy tradeoff* (Tsipras et al. 2018), i.e., the empirical observation made for non-quantized neural networks that we can have a high robustness or a high accuracy but not both at the same time.

ablation analysis on the two networks of our first experiment. We report the number of samples where the verification approach could not determine whether the network is robust on the sample or not. We use a timeout of 20s per instance when running our algorithm.

The results shown in Table 3 indicate that our algorithm is indeed improving on the number of samples for which robustness could be decided. However, the improvement is relatively small, suggesting that the QA-IBP training stems from most of the observed gains.

Limitations

In this section, we aim to scale our QA-IBP beyond the two gray-scale image classification tasks studied before to the CIFAR-10 dataset (Krizhevsky and Hinton 2010). Our setup consists of the same convolutional neural network as used for the Fasion-MNIST. We run our setup with several values for the weight decay rate. Similar to above, we linearly scale ϵ from 0 to 4 during training and report the certified robust accuracy obtained by QA-IBP with an ϵ of 1 and 4 and the clean accuracy.

The results in Table 2 express the robustness-accuracy tradeoff, i.e., the observed antagonistic relation between clean accuracy and robustness, which has been extensively studied in non-quantized neural networks (Zhang et al. 2019; Bubeck and Sellke 2021). Depending on the weight decay value, different points on the trade-off were observed. In particular, our training procedure either obtains a network that has an acceptable accuracy but no robustness or a certifiable robust network with a significantly reduced clean accuracy.

We also trained larger models on all datasets (MNIST, Fashion-MNIST, and CIFAR-10), but observed the same behavior of having a high accuracy but no robustness when training, i.e., as in row with $1 \cdot 10^{-5}$ in Table 2.

We found the underlying reason for this behavior to be activations of internal neurons that clamp to the minimum and maximum value of the quantization range in their QA-IBP representation but not in their standard representation. Consequently, the activation gradients become zero during QA-IBP due to falling in the constant region of the activation function. This effect is specific to quantized neural networks due to the upper bound on the representative range of values. Our observation hints that future research needs to look into developing dynamic quantization ranges or weight decay schedules that can adapt to both the standard and the QA-IBP representation of a QNN.

Conclusion

In this paper, we introduced quantization-aware interval bound propagation (QA-IBP), the first method for training certifiably robust QNNs. We also present a theoretical result on the existence and upper bounds on the needed size of a robust QNN for a given dataset of 1-dimensional datapoints. Moreover, based on our interval bound propagation method, we developed the first complete verification algorithm for QNNs that may be run on GPUs. We experimentally showed that our training scheme and verification procedure advance the state-of-the-art on certifying L_∞-robustness of QNNs.

We also demonstrated the limitations of our method regarding training stability and convergence. Particularly, we found that the boundedness of the representable value range of QNNs compared to standard networks leads to truncation of the abstract domains, which in turn leads to gradients becoming zero. Our results suggest that dynamic quantization schemes that adapt their quantization range to the abstract domains instead of the concrete activation values of existing quantization schemes may further improve the certified robust accuracy of quantized neural networks.

Nonetheless, our work serves as a new baseline for future research. Promising directions on how to improve upon QA-IBP and potentially overcome its numerical challenges is to adopt advanced quantization-aware training techniques. For instance, dynamical quantization ranges (Choi et al. 2018; Jacob et al. 2018), mixed-precision layers (Zhou et al. 2016; Yao et al. 2021), and automated architecture search (Tan and Le 2019) have shown promising results for standard training

Method	MNIST		Fashion-MNIST	
	$\epsilon = 1$	$\epsilon = 4$	$\epsilon = 1$	$\epsilon = 4$
IBP + Projected gradient descent	**0.35%**	3.51%	3.96%	19.23%
Algorithm 1	**0.35%**	**3.47%**	**3.93%**	**18.51%**

Table 3: Percentage of samples where the robustness of the network can be determined, i.e., certified or falsified, by the method (lower is better). Algorithm timeout was set to 20s per instance.

QNNs and might enhance QA-IBP-based training procedures as well. Moreover, further improvements may be feasible by adapting recent advances in IBP-based training methods for non-quantized neural networks (Müller et al. 2022) to our quantized IBP variant.

Acknowledgments

References

Amir, G.; Wu, H.; Barrett, C.; and Katz, G. 2021. An SMT-based approach for verifying binarized neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*.

Baranowski, M. S.; He, S.; Lechner, M.; Nguyen, T. S.; and Rakamaric, Z. 2020. An SMT Theory of Fixed-Point Arithmetic. In *International Joint Conference on Automated Reasoning (IJCAR)*.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*.

Bubeck, S.; and Sellke, M. 2021. A Universal Law of Robustness via Isoperimetry. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Bunel, R. R.; Turkaslan, I.; Torr, P.; Kohli, P.; and Mudigonda, P. K. 2018. A Unified View of Piecewise Linear Neural Network Verification. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Cheng, C.-H.; Nührenberg, G.; Huang, C.-H.; and Ruess, H. 2018. Verification of Binarized Neural Networks via Inter-Neuron Factoring. In *Working Conference on Verified Software: Theories, Tools, and Experiments (VSTTE)*.

Choi, J.; Wang, Z.; Venkataramani, S.; Chuang, P. I.-J.; Srinivasan, V.; and Gopalakrishnan, K. 2018. PACT: Parameterized Clipping Activation for Quantized Neural Networks. *arXiv preprint arXiv:1805.06085*.

Clark, B.; and Cesare, T. 2018. Satisfiability Modulo Theories. In Clarke, E. M.; Henzinger, T. A.; Veith, H.; and Bloem, R., eds., *Handbook of Model Checking*. Springer.

Cousot, P.; and Cousot, R. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Symposium on Principles of Programming Languages (POPL)*.

Dvijotham, K.; Stanforth, R.; Gowal, S.; Mann, T. A.; and Kohli, P. 2018. A Dual Approach to Scalable Verification of Deep Networks. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Ehlers, R. 2017. Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*.

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 (S&P)*.

Giacobbe, M.; Henzinger, T. A.; and Lechner, M. 2020. How Many Bits Does it Take to Quantize Your Neural Network? In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*.

Gowal, S.; Dvijotham, K. D.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T.; and Kohli, P. 2019. Scalable Verified Training for Provably Robust Image Classification. In *IEEE International Conference on Computer Vision (ICCV)*.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep Learning with Limited Numerical Precision. In *International Conference on Machine Learning (ICML)*.

Henzinger, T. A.; Lechner, M.; and Žikelić, Đ. 2021. Scalable Verification of Quantized Neural Networks. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. 2019. Searching for mobilenetv3. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2017. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *The Journal of Machine Learning Research (JMLR)*.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jia, K.; and Rinard, M. 2020. Efficient exact verification of binarized neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Kahan, W. 1996. IEEE Standard 754 for Binary Floating-Point Arithmetic. *Lecture Notes on the Status of IEEE*.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *International Conference on Computer Aided Verification (CAV)*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Krishnamoorthi, R. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.

Krizhevsky, A.; and Hinton, G. 2010. Convolutional Deep Belief Networks on CIFAR-10. University of Toronto Preprint.

Lazarus, C.; and Kochenderfer, M. J. 2022. A mixed integer programming approach for verifying properties of binarized neural networks. *arXiv preprint arXiv:2203.07078*.

Lechner, M.; Hasani, R. M.; Grosu, R.; Rus, D.; and Henzinger, T. A. 2021. Adversarial Training is Not Ready for Robot Learning. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Lechner, M.; Žikelić, Đ.; Chatterjee, K.; Henzinger, T. A.; and Rus, D. 2022. Quantization-aware Interval Bound Propagation for Training Certifiably Robust Quantized Neural Networks. *arXiv preprint arXiv:2211.16187*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. GradientBased Learning Applied to Document Recognition. *Proceedings of the IEEE*.

Lin, H.; Lou, J.; Xiong, L.; and Shahabi, C. 2021. Integer-arithmetic-only Certified Robustness for Quantized Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*.

Louizos, C.; Reisser, M.; Blankevoort, T.; Gavves, E.; and Welling, M. 2019. Relaxed Quantization for Discretized Neural Networks. In *International Conference on Learning Representations (ICLR)*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*.

Mirman, M.; Baader, M.; and Vechev, M. 2021a. The Fundamental Limits of Interval Arithmetic for Neural Networks. *arXiv preprint arXiv:2112.05235*.

Mirman, M.; Baader, M.; and Vechev, M. T. 2021b. The Fundamental Limits of Interval Arithmetic for Neural Networks. *CoRR*, abs/2112.05235.

Mirman, M.; Gehr, T.; and Vechev, M. 2018. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In *International Conference on Machine Learning (ICML)*.

Mistry, S.; Saha, I.; and Biswas, S. 2022. An MILP Encoding for Efficient Verification of Quantized Deep Neural Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11): 4445–4456.

Müller, M. N.; Eckert, F.; Fischer, M.; and Vechev, M. 2022. Certified Training: Small Boxes are All You Need. *arXiv preprint arXiv:2210.04871*.

Narodytska, N.; Kasiviswanathan, S.; Ryzhyk, L.; Sagiv, M.; and Walsh, T. 2018. Verifying Properties of Binarized Deep Neural Networks. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Pulina, L.; and Tacchella, A. 2012. Challenging SMT solvers to verify neural networks. *AI Communications*.

Sena, L.; Song, X.; Alves, E.; Bessa, I.; Manino, E.; Cordeiro, L.; et al. 2021. Verifying Quantized Neural Networks using SMT-Based Model Checking. *arXiv preprint arXiv:2106.05997*.

Sena, L. H. C.; et al. 2022. *Automated verification and refutation of quantized neural networks*. Master's thesis, Universidade Federal do Amazonas.

Serre, F.; Müller, C.; Singh, G.; Püschel, M.; and Vechev, M. 2021. Scaling Polyhedral Neural Network Verification on GPUs. In *Proc. Machine Learning and Systems (MLSys)*.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2018. Fast and Effective Robustness Certification. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Singh, G.; Gehr, T.; Püschel, M.; and Vechev, M. 2019. An Abstract Domain for Certifying Neural Networks. In *Symposium on Principles of Programming Languages (POPL)*.

Smith, S. W.; et al. 1997. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Pub. San Diego.

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

Tan, M.; and Le, Q. V. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning (ICML)*.

Tjeng, V.; Xiao, K. Y.; and Tedrake, R. 2019. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *International Conference on Learning Representations (ICLR)*.

Tsipras, D.; Santurkar, S.; Engstrom, L.; Turner, A.; and Madry, A. 2018. Robustness May Be at Odds with Accuracy. In *International Conference on Learning Representations (ICLR)*.

Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.-J.; and Kolter, J. Z. 2021. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Wong, E.; and Kolter, Z. 2018. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *International Conference on Machine Learning (ICML)*.

Wu, D.; Xia, S.-T.; and Wang, Y. 2020. Adversarial weight perturbation helps robust generalization. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Xiang, W.; Tran, H.-D.; and Johnson, T. T. 2018. Output Reachable Set Estimation and Verification for Multi-Layer Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*.

Yao, Z.; Dong, Z.; Zheng, Z.; Gholami, A.; Yu, J.; Tan, E.; Wang, L.; Huang, Q.; Wang, Y.; Mahoney, M.; et al. 2021. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning (ICML)*. PMLR.

Zhang, D.; Yang, J.; Ye, D.; and Hua, G. 2018a. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks. In *European Conference on Computer Vision (ECCV)*.

Zhang, H.; Chen, H.; Xiao, C.; Gowal, S.; Stanforth, R.; Li, B.; Boning, D. S.; and Hsieh, C. 2020. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. In *International Conference on Learning Representations (ICLR)*.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018b. Efficient Neural Network Robustness Certification with General Activation Functions. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; El Ghaoui, L.; and Jordan, M. 2019. Theoretically Principled Trade-off between Robustness and Accuracy. In *International Conference on Machine Learning (ICML)*.

Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; and Zou, Y. 2016. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *arXiv preprint arXiv:1606.06160*.