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

Mathias Lechner¹, Đorđe Žikelić², Krishnendu Chatterjee², Thomas A. Henzinger², Daniela Rus¹

¹Massachusetts Institute of Technology (MIT)
²Institute 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 establishes 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 inferring 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 to 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 approximating 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 arithmetics 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_\infty-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_\infty-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 during training and adapts the quantization range according to the moving average. LQ-Net (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 \( f_0 : \mathbb{R}^n \to \mathbb{R}^m \) that consist of several sequentially composed layers \( f_0 = l_1 \circ \ldots \circ l_s \), where layers are parametrized by the vector \( \theta \) of neural network parameters. Quantization is an interpretation of a neural network \( f_0 \) 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 \( I \subseteq \mathbb{R}^n \), we define an interpretation map

\[
\lceil \cdot \rceil : (\mathbb{R}^n \to \mathbb{R}^m) \to (I \to \mathbb{R}^m),
\]

which maps a neural network to its interpretation operating over the input set \( I \). For instance, if \( I = \mathbb{R}^n \) then \( \lceil f_0 \rceil \) is the idealized real arithmetic interpretation of \( f_0 \), whereas \( \lceil f \rceil_{\text{float32}} \) denotes its floating-point 32-bit implementation (Kahan 1996). Given \( k \in \mathbb{N} \), the \( k \)-bit quantization is then an interpretation map \( \lceil \cdot \rceil_{\text{int}-k} \) which uses \( k \)-bit fixed-point arithmetic. We say that \( \lceil f_0 \rceil_{\text{int}-k} \) is a \( k \)-bit quantized neural network (QNN).

The semantics of the QNN \( \lceil f_0 \rceil_{\text{int}-k} \) are defined as follows. Let \( \mathbb{Z}_k = \{0, 1\}^k \) be the set of all bit-vectors of bit-width \( k \). The QNN \( \lceil f_0 \rceil_{\text{int}-k} \) then consists of sequentially composed layers \( \lceil f_0 \rceil_{\text{int}-k} = l_1 \circ \ldots \circ l_s \), where now each layer is a function \( l_i : [\mathbb{Z}_k^n] \to [\mathbb{Z}_k^{n_{i+1}}] \) that operates over \( k \)-bit bitvectors and is defined as follows:

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

\[
x_i'' = \text{round}(x_i', M_i) = \lfloor x_i' \cdot M_i \rfloor,
\]

\[
y_i = \sigma_i(\min(2^{N_i} - 1, x_i'')),
\]

Here, \( w_{ij} \in \mathbb{Z}_k^{n_i} \) and \( b_i \in \mathbb{Z}_k^{n_i} \) for each \( 1 \leq j \leq n_i \) and \( 1 \leq i \leq n_0 \) 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 \( w_{ij} \) 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 \( \sigma_i \) 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 functions, 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.

14966
Adversarial robustness for QNNs. We now formalize the notion of adversarial robustness for QNN classifiers. Let $\|f_\theta\|_{\text{int-k}} : [Z^n_k] \rightarrow [Z^n_k]$ be a k-bit QNN. It naturally defines a classifier with $m$ classes by assuming that it assigns to an input $x \in [Z^n_k]$ a label of the maximal output neuron on input value $x$, i.e., $y = \text{class}(x) = \text{argmax}_{1 \leq i \leq m} \|f_\theta\|_{\text{int-k}}(x)[i]$, 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 $1 \leq i \leq m$ 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 $\epsilon$-adversarially robust at point $x$ if

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

where $\|\cdot\|_{\infty}$ denotes the $L_\infty$-norm. Then, given a finite dataset $D = \{(x_1, y_1), \ldots, (x_D, y_D)\}$ with $x_i \in [Z^n_k]$ and $y_i \in [Z^n_k]$ for each $1 \leq i \leq |D|$, we say that $\|f_\theta\|_{\text{int-k}}$ is $\epsilon$-adversarially robust with respect to the dataset $D$ if it is $\epsilon$-adversarially robust at each datapoint in $D$.

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 $l_{i1}$, $l_{i2}$: $[Z^n_k] \rightarrow [Z^n_k]$ defined as follows:

$$\mu_j = \frac{x_j + x_j}{2}, \quad r_j = \frac{x_j - x_j}{2}$$

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

$$x'_i = \mu_i - r_i \quad x''_i = \mu_i + r_i$$

$$\bar{x}'_i = \text{round}(x'_i, k_i) = |x'_i \cdot M_i|$$

$$\bar{x}''_i = \text{round}(x''_i, k_i) = |x''_i \cdot M_i|$$

$$\bar{y}_i = \max\{0, \min\{2^{N_i} - 1, x''_i\}\}$$

$$\bar{y}'_i = \max\{0, \min\{2^{N_i} - 1, x'_i\}\}$$

As with standard QNNs, $w_{ij}, b_i \in [Z^n_k]$ and $b_i \in [Z^n_k]$ for each $1 \leq j \leq n_i$ and $1 \leq i \leq n_k$ denote the weights and biases of $j$ 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 $\|f_\theta\|_{\text{int-k}}$. 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 interval domain $(x, x')$ with $x = x - \epsilon$ and $x = x + \epsilon$. Next, we propagate the abstract domains through the IBP representation of the network to obtain output bounds $y = \|f_\theta\|_{\text{int-k}}(x, x')$ and $\bar{y} = \|f_\theta\|_{\text{int-k}}(x, x')$. Finally, we know that $\|f_\theta\|_{\text{int-k}}$ is $\epsilon$-adversarially robust at point $x$ if

$$\bar{y}_i > \bar{y}'_i, \quad \forall i = \text{class}(x) \text{ and } \forall j \neq \text{class}(x).$$

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

$$\text{fake.quant}(x_i, k_i) := \text{round}(x_i, k_i)$$

$$\frac{\partial \text{fake.quant}(x_i)}{\partial x_i} := 1.$$
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 $2\epsilon$ away, we prove that there exists a QNN with ReLU activations that is $\epsilon$-robust with respect to the dataset. Furthermore, we provide an upper bound on the number of neurons that the QNN must contain. We assume $2\epsilon$ distance simply for the $\epsilon$-neighbourhoods of datapoints to be disjoint so that robustness cannot impose contradicting classification conditions.

Note that this result does not hold for real arithmetic feed-forward 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 $D = \{(x_1, y_1), \ldots, (x_D, y_D)\}$ with $x_i \in \mathbb{Z}_k^1$ and $y_i \in \mathbb{Z}_k$ for each $1 \leq i \leq |D|$. Suppose that $|x_i - x_j|_{\infty} \geq 2\epsilon$ for each $i \neq j$. Then, there exists a QNN $f_{\text{int}, k}$ which is $\epsilon$-robust with respect to the dataset $D$ and which consists of $O(|D| \cdot \lceil \epsilon \cdot 2^k + 1 \rceil)$ 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 $B_i(x_i) = \{x' \in \mathbb{Z}_k^1 \mid ||x' - x_i||_{\infty} < \epsilon\}$ is finite and consists of at most $2\epsilon \cdot 2^k + 1$ bit-vectors for each $1 \leq i \leq |D|$. Hence, constructing a QNN $f_{\text{int}, k}$ that is $\epsilon$-robust with respect to the dataset $D$ is equivalent to constructing a QNN that correctly classifies $D' = \bigcup_{i=1}^{|D|} \{ (x', y_i) \mid x' \in B_i(x_i) \}$ which consists of $O(|D| \cdot \lceil \epsilon \cdot 2^k + 1 \rceil)$ datapoints. In the extended version of the paper (Lechner et al. 2022), we then design the QNN $f_{\text{int}, k}$ which 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 $D = \{(x_1, y_1), \ldots, (x_D, y_D)\}$ with $x_i \in \mathbb{Z}_k^1$ and $y_i \in \mathbb{Z}_k$ for each $1 \leq i \leq |D|$. Then, there exists a QNN $f_{\text{int}, k}$ which correctly classifies the dataset, i.e. $f_{\text{int}, k}(x_i) = y_i$ for each $1 \leq i \leq |D|$, and which consists of $O(|D|)$ 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 functions 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 $\epsilon$-robustness of a QNN with respect to some finite dataset $D$ and $\epsilon > 0$ is equivalent to verifying $\epsilon$-robustness of the QNN at each datapoint in $D$, Algorithm 1 only takes as inputs a QNN $f_{\text{int}, k}$ that operates over bit-vectors of bit-width $k$, a single datapoint $x \in \mathbb{Z}_k^1$, and a robustness radius $\epsilon > 0$. It then returns either ROBUST if $f_{\text{int}, k}$ is verified to be $\epsilon$-robust at $x \in \mathbb{Z}_k^1$, or VULNERABLE if an adversarial attack $||x' - x||_{\infty} < \epsilon$ with class($x'$) = class($x$) is found.

The algorithm proceeds by initializing a stack $D$ of abstract intervals to contain a single element $\{x - \epsilon \cdot 1, x + \epsilon \cdot 1\}$, where $1 \in \mathbb{Z}_k^1$ 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 $(x, \pi)$ from $D$ and processes it as follows. First, it uses IBP for QNNs that we introduced before to propagate $(x, \pi)$ in order to compute an abstract interval $(y, \overline{y})$ that overapproximates the set of all possible outputs for a concrete input point in $(x, \pi)$. The algorithm then considers three cases. First, if $(y, \overline{y})$ does not violate Equation (11) which characterizes violation of robustness by a propagated abstract interval, Algorithm 1 concludes that the abstract interval $(x, \pi)$ does not contain an adversarial example and it proceeds to processing the next element of $D$. Second, if $(y, \overline{y})$ violates Equation (11), the algorithm uses projected gradient descent restricted to $(x, \pi)$ 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 $\|f_0\|_{\text{int},k}$, datapoint $x \in \mathbb{Z}_K^n$, robustness radius $\epsilon > 0$
2: **Output** ROBUST or VULNERABLE
3: $D \leftarrow \{(x - \epsilon \cdot 1, x + \epsilon \cdot 1)\}$
4: while $D \neq \emptyset$ do
5:     $(\bar{x}, \bar{\pi}) \leftarrow$ pop item from $D$
6:     $(y, \bar{y}) \leftarrow \|f_0\|_{\text{int},k}(\bar{x}, \bar{\pi}), \|f_0\|_{\text{int},k}(\bar{x}, \bar{\pi})$ propagated via IBP
7:     if $(y, \bar{y})$ violates Equation (11) then
8:         Try to generate adversarial example using projected gradient descent restricted to $(\bar{x}, \bar{\pi})$
9:         if adversarial example found then
10:        return VULNERABLE
11:     else
12:         $(x', \overline{x'}), (x'', \overline{x''}) \leftarrow$ partition of $(\bar{x}, \bar{\pi})$ into two abstract subintervals
13:         $D \leftarrow D \cup \{(x', \overline{x'}), (x'', \overline{x''})\}$
14:     end if
15: end while
16: return ROBUST

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

$$i^* = \arg\max_{1 \leq i \leq n} (\pi[i] - \bar{x}[i])$$

and splitting the abstract interval $(\bar{x}, \bar{\pi})$ along the $i^*$-th dimension into two abstract subintervals $(x', \overline{x'}), (x'', \overline{x''})$, 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 $\|f_0\|_{\text{int},k}$ is $\epsilon$-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 class($x'$) $\neq$ 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.

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 $\epsilon$ 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 $L_{\infty}$-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

---

1https://github.com/mlech26/qquantization_aware_ibp
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 Fashion-MNIST. We run our setup with several values for the weight decay rate. Similar to above, we linearly scale $\epsilon$ from 0 to 4 during training and report the certified robust accuracy obtained by QA-IBP with an $\epsilon$ 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.

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 certifiably 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 $L_\infty$-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.

### Table 1: Certified robust accuracy of a convolutional neural network trained with QA-IBP compared to existing methods for certifying $L_\infty$-robustness of quantized neural networks reported in the literature.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST $\epsilon$ = 0</th>
<th>MNIST $\epsilon$ = 1</th>
<th>MNIST $\epsilon$ = 4</th>
<th>Fashion-MNIST $\epsilon$ = 0</th>
<th>Fashion-MNIST $\epsilon$ = 1</th>
<th>Fashion-MNIST $\epsilon$ = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA-IBP (ours)</td>
<td>99.2%</td>
<td>98.8%</td>
<td>95.6%</td>
<td>86.3%</td>
<td>80.0%</td>
<td>59.8%</td>
</tr>
<tr>
<td>QF_BV SMT (Giacobbe, Henzinger, and Lechner 2020)</td>
<td>97.1%</td>
<td>92%</td>
<td>0%</td>
<td>85.6%</td>
<td>44%</td>
<td>0%</td>
</tr>
<tr>
<td>QF_BV SMT (Henzinger, Lechner, and Žikelić 2021)</td>
<td>97.1%</td>
<td>99%*</td>
<td>53%</td>
<td>85.6%</td>
<td>76%</td>
<td>36%</td>
</tr>
</tbody>
</table>

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 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.

<table>
<thead>
<tr>
<th>Weight decay</th>
<th>CIFAR-10 $\epsilon$ = 0</th>
<th>CIFAR-10 $\epsilon$ = 1</th>
<th>CIFAR-10 $\epsilon$ = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \cdot 10^{-4}$</td>
<td>30.0%</td>
<td>28.4%</td>
<td>22.4%</td>
</tr>
<tr>
<td>$5 \cdot 10^{-5}$</td>
<td>39.6%</td>
<td>34.1%</td>
<td>20.6%</td>
</tr>
<tr>
<td>$1 \cdot 10^{-5}$</td>
<td>83.1%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>
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

This work was supported in part by the ERC-2020-AdG 101020093, ERC CoG 863818 (FoRM-SMArt) and the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 665385. Research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The research was also funded in part by the AI2050 program at Schmidt Futures (Grant G-22-63172) and Capgemini SE.

## References


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


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.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST $\epsilon = 1$</th>
<th>MNIST $\epsilon = 4$</th>
<th>Fashion-MNIST $\epsilon = 1$</th>
<th>Fashion-MNIST $\epsilon = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBP + Projected gradient descent</td>
<td>3.51%</td>
<td>3.96%</td>
<td>3.96%</td>
<td>18.51%</td>
</tr>
</tbody>
</table>


