Certified Quantization Strategy Synthesis for Neural Networks*

Yedi Zhang[1], Guangke Chen[2], Fu Song[3,4], Jun Sun[5](✉), and Jin Song Dong[1]

[1] National University of Singapore, Singapore 117417, Singapore
[2] ShanghaiTech University, Shanghai 201210, China
[3] Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[4] Nanjing Institute of Software Technology, Nanjing 211135, China
[5] Singapore Management University, Singapore 178902, Singapore
junsun@smu.edu.sg

Abstract. Quantization plays an important role in deploying neural networks on embedded, real-time systems with limited computing and storage resources (e.g., edge devices). It significantly reduces the model storage cost and improves inference efficiency by using fewer bits to represent the parameters. However, it was recently shown that critical properties may be broken after quantization, such as robustness and backdoor-freeness. In this work, we introduce the first method for synthesizing quantization strategies that verifiably maintain desired properties after quantization, leveraging a key insight that quantization leads to a data distribution shift in each layer. We propose to compute the preimage for each layer based on which the preceding layer is quantized, ensuring that the quantized reachable region of the preceding layer remains within the preimage. To tackle the challenge of computing the exact preimage, we propose an MILP-based method to compute its under-approximation. We implement our method into a tool Quadapter and demonstrate its effectiveness and efficiency by providing certified quantization that successfully preserves model robustness and backdoor-freeness.

1 Introduction

While deep neural networks (DNNs) have achieved notable success in various application domains [5, 31], their deployment on resource-constrained, embedded,

* This study was funded by the National Natural Science Foundation of China (62072309), CAS Project for Young Scientists in Basic Research (YSBR-040), ISCAS New Cultivation Project (ISCAS-PYFX-202201), ISCAS Fundamental Research Project (ISCAS-JCZD-202302), the Ministry of Education, Singapore under its Academic Research Fund Tier 3 (Award ID: MOET32020-0004), and the Ministry of Education, Singapore under its Academic Research Fund Tier 3 (Award ID: MOE-MOET32020-0003). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

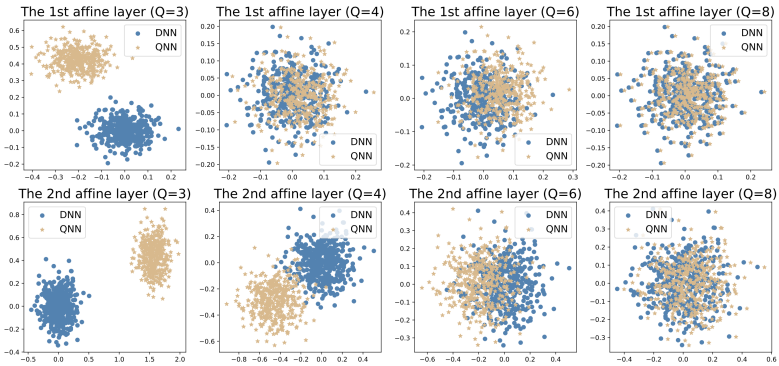


Fig. 1. Visualized data distribution shift using 400 random samples centered around an input image. These inputs are processed through both a DNN (trained on MNIST [20]) and its counterparts quantized with bit-width $Q \in \{4, 6, 8, 10\}$. The resulting high-dimensional convex shapes are visualized in 2D. The blue and brown scatters demonstrate the distribution of output values of each affine layer of the DNN and QNNs.

real-time systems is currently impeded by their substantial demand for computing and storage resources [27]. Quantization is one of the most popular and promising techniques to address this issue [8, 39]. By storing the full-precision values in a DNN (such as parameters and/or activation values) into low bit-width fixed-point numbers, quantization facilitates the compression of a DNN and leads to a quantized neural network (QNN), making the network more efficient.

While a lot of techniques have been proposed to minimize the loss of accuracy induced by quantization [8, 15, 21, 22, 32, 33, 42, 44, 48], an important side-effect of quantization is overlooked, that is the risk of breaking desired critical properties, e.g., robustness [24, 41] and backdoor-freeness [13, 26, 34, 55], thereby raising great concerns, especially when they are deployed in safety-critical applications. While quantization-aware training techniques have been proposed to improve the robustness for a given fixed quantization strategy [23, 24, 41, 43], they fail to provide robustness guarantees. Therefore, it becomes imperative to devise a quantization strategy synthesis technique, ensuring that the resulting QNNs retain specific desired properties. Noting that although various verification methods for QNNs have been proposed [3, 9, 12, 52–54], they exclusively focus on post-hoc analyses rather than synthesis, namely, these methods merely verify or falsify the properties but offer no solutions for those that are falsified.

Contributions. In this work, we propose the first quantization strategy synthesis method, named Quadapter, such that the desired properties are verifiably maintained by the quantization. Given a DNN $\mathcal{N}$ and a property $\langle \mathcal{I}, \mathcal{O} \rangle$ where $\mathcal{I}$ and $\mathcal{O}$ are the pre- and post-condition for the input and output, our general idea is first to compute the preimage of each layer w.r.t. the output region formed by $\mathcal{O}$. Then, considering the typical data distribution shift caused by quantization in each layer (cf. Fig. 1), we identify the minimal bit-width for each layer such that the shifted quantized reachable region w.r.t. $\mathcal{I}$ always remains within the

corresponding preimage. This method allows us to derive a quantization strategy for the entire network, preserving the desired property $\langle \mathcal{I}, \mathcal{O} \rangle$ after quantization.

A key technical question is how to represent and compute the preimage for each layer effectively and efficiently. In this work, we propose to compute an under-approximation of the preimage for each layer and represent it by adapting the abstract domain of DeepPoly [40]. Specifically, we devise a novel Mixed Integer Linear Programming (MILP) based method to propagate the (approximate) preimage layer-by-layer in a backward fashion, where we encode the affine transformation and activation function precisely as linear constraints and compute under-approximate preimage via MILP solving.

We implement our methods as an end-to-end tool Quadapter and extensively evaluate our tool on a large set of synthesis tasks for DNNs trained using two widely used datasets MNIST [20] and Fashion-MNIST [46], where the number of hidden layers varies from 2 to 6 and the number of neurons in each hidden layer varies from 100 to 512. The experimental results demonstrate the effectiveness and efficiency of Quadapter in synthesizing certified quantization strategies to preserve robustness and backdoor-freeness. The quantization strategy synthesized by Quadapter generally preserves the accuracy of the original DNNs (with only minor degradation). We also show that by slightly relaxing the under-approximate preimages of the hidden layers (without sacrificing the overall soundness), Quadapter can synthesize quantization strategies with much smaller bit-widths while preserving the desired properties and accuracy.

The remainder of this paper is organized as follows. Section 2 gives the preliminaries and formulates the problem. Section 3 presents the details of our approach and Section 4 demonstrates its applications. Section 5 reports our experimental results. We discuss related work in Section 6 and finally, Section 7 concludes. The source code for our tool, along with the benchmarks, is available in [50], which also includes a long version of the paper containing all missing proofs, design choices, implementation details, and additional experimental results.

2 Preliminaries

We denote by $\mathbb{R}$ the set of real numbers. Given an integer n, let $[n] := \{1, \ldots, n\}$ and $\mathbb{R}^n$ be the set of the n-tuples of real numbers. We use **bold lowercase letters** (e.g., $\mathbf{x}$) and **BOLD UPPERCASE** (e.g., $\mathbf{W}$) to denote vectors and matrices. We denote by $\mathbf{W}_{i,:}$ (resp. $\mathbf{W}_{:,i}$) the i-th row (resp. column) vector of the matrix $\mathbf{W}$, and by $\mathbf{x}_j$ (resp. $\mathbf{W}_{i,j}$) the j-th entry of the vector $\mathbf{x}$ (resp. $\mathbf{W}_{i,:}$). $\mathbf{M}$ denotes an extremely large number.

A *Deep Neural Network* (DNN) with $2d$ layers is a function $\mathcal{N} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{2d}}$ such that $\mathcal{N} = f_{2d} \circ \cdots \circ f_1$, where $f_1 : \mathbb{R}^{n_0} \to \mathbb{R}^{n_1}$ is the input layer, $f_{2d} : \mathbb{R}^{n_{2d-1}} \to \mathbb{R}^{n_{2d}}$ is the output layer, and the others are hidden layers. The hidden layers alternate between affine layers $f_{2i} : \mathbb{R}^{n_{2i-1}} \to \mathbb{R}^{n_{2i}}$ and activation layers $f_{2i+1} : \mathbb{R}^{n_{2i}} \to \mathbb{R}^{n_{2i+1}}$ for $i \in [d-1]$. The semantics of each layer is defined as follows: $\mathbf{x}^1 = f_1(\mathbf{x}) = \mathbf{x}$, $\mathbf{x}^{2i} = f_{2i}(\mathbf{x}^{2i-1}) = \mathbf{W}^{2i}\mathbf{x}^{2i-1} + \mathbf{b}^{2i}$ for $i \in [d]$ and $\mathbf{x}^{2i+1} = f_{2i+1}(\mathbf{x}^{2i}) = \mathrm{ReLU}(\mathbf{x}^{2i})$ for $i \in [d-1]$, where $\mathbf{W}^{2i}$ and $\mathbf{b}^{2i}$ are the

weight matrix and the bias vector of the $2i$-th layer, $n_0 = n_1$ and $n_{2i} = n_{2i+1}$ for $i \in [d-1]$. Note that for the sake of presentation, we regard affine and activation layers separately as hidden layers, some prior work regards the composition of an affine layer and an activation layer as one hidden layer, e.g., [4, 25, 38]. Given a DNN $\mathcal{N}$ with $2d$ layers, we use $\mathcal{N}_{[i:j]} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_j}$ to denote the composed function $f_j \circ \cdots \circ f_i$. By $\mathcal{N}(\mathcal{I})$ (resp. $\mathcal{N}(\mathcal{I})_g$), we refer to the output region of the network $\mathcal{N}$ (resp. neuron $\mathbf{x}_g^{2d}$) w.r.t. the input region $\mathcal{I}$.

A *Quantized Neural Network* (QNN) is structurally identical to a DNN but uses fixed-point values for its parameters and/or layer outputs. In this work, we focus on QNNs where only parameters are quantized using the most hardware-efficient quantization scheme, i.e., *signed power-of-two quantization* [33].

A *quantization configuration* ξ is a pair $\langle Q, F \rangle$, where Q denotes the total bit-width and F denotes the bit-width for the fractional part of the value. Given a quantization configuration ξ and a real-valued number u, its fixed-point counterpart $\hat{u}$ is defined as $\hat{u} = \min(\max(\frac{\lfloor u \cdot 2^F \rceil}{2^F}, -2^{Q-1}), 2^{Q-1} - 1)$, where $\lfloor \cdot \rceil$ is the round-to-nearest operator. Given a DNN $\mathcal{N} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{2d}}$ with $2d$ layers and a set of quantization configurations for affine and output layers $\Xi = \{\xi_1, \ldots, \xi_d\}$, its quantized version $\widehat{\mathcal{N}} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{2d}}$ is a composed function as $\widehat{\mathcal{N}} = \hat{f}_{2d} \circ \cdots \circ \hat{f}_1$, where each layer is defined the same as that in the DNN $\mathcal{N}$ except that the parameters $\mathbf{W}^{2i}$ and $\mathbf{b}^{2i}$ for $i \in [d]$ from the DNN $\mathcal{N}$ are quantized into fixed-point values $\widehat{\mathbf{W}}^{2i}$ and $\widehat{\mathbf{b}}^{2i}$ in the QNN $\widehat{\mathcal{N}}$ according to the quantization configuration ξ_i. In this work, we call the set Ξ a *quantization strategy* of the DNN $\mathcal{N}$.

Definition 1. *Given a DNN $\mathcal{N} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{2d}}$, a property of $\mathcal{N}$ is a pair $\langle \phi, \psi \rangle$ where ϕ is a pre-condition over the input $\mathbf{x} \in \mathbb{R}^{n_0}$ and ψ is a post-condition over the output $\mathbf{y} = \mathcal{N}(\mathbf{x}) \in \mathbb{R}^{n_{2d}}$. $\mathcal{N}$ satisfies the property $\langle \phi, \psi \rangle$, denoted by $\mathcal{N} \models \langle \phi, \psi \rangle$, if $\phi(\mathbf{x}) \Rightarrow \psi(\mathcal{N}(\mathbf{x}))$ holds for any input $\mathbf{x} \in \mathbb{R}^{n_0}$.*

Following prior work [49], we assume that the pre-condition ϕ and post-condition ψ are expressible by polyhedra, namely, $\mathcal{I}$ and $\mathcal{O}$, respectively. It is reasonable since, for typical properties such as robustness, both conditions can be effectively represented by a set of linear constraints. For simplicity, we will use $\langle \mathcal{I}, \mathcal{O} \rangle$ to denote the property directly. We are now ready to define our problem.

Definition 2. *Given a DNN $\mathcal{N}$ and a property $\langle \mathcal{I}, \mathcal{O} \rangle$ such that $\mathcal{N} \models \langle \mathcal{I}, \mathcal{O} \rangle$, the problem of certified quantization strategy synthesis is to find a quantization strategy Ξ such that $\widehat{\mathcal{N}} \models \langle \mathcal{I}, \mathcal{O} \rangle$, where $\widehat{\mathcal{N}}$ is the QNN obtained from $\mathcal{N}$ under the quantization strategy Ξ.*

Review of DeepPoly. The core idea of DeepPoly is to (approximately) represent the transformation of each layer using an abstract transformer, and compute lower/upper bounds for the output of each neuron. Fix a neuron $\mathbf{x}_j^i$, its abstract element $\mathcal{A}_j^{i,\sharp}$ is given by a tuple $\langle \mathbf{a}_j^{i,\leq}, \mathbf{a}_j^{i,\geq}, l_j^i, u_j^i \rangle$, where $\mathbf{a}_j^{i,\leq}$ (resp. $\mathbf{a}_j^{i,\geq}$) is a symbolic lower (resp. upper) bound in the form of a linear combination of variables from its preceding layers, l_j^i (resp. u_j^i) is the concrete lower (resp. upper) bound of $\mathbf{x}_j^i$. We denote by $\mathbf{a}^{i,\leq}$ (resp. $\mathbf{a}^{i,\geq}$) the vector of symbolic bounds $\mathbf{a}_j^{i,\leq}$

(resp. $\mathbf{a}_j^{i,\geq}$) of the neurons $\mathbf{x}_j^i$'s in the i-th layer. The concretization of $\mathcal{A}_j^{i,\sharp}$ is defined as $\gamma(\mathcal{A}_j^{i,\sharp}) = \{\mathbf{x}_j^i \in \mathbb{R} \mid \mathbf{a}_j^{i,\leq} \leq \mathbf{x}_j^i \leq \mathbf{a}_j^{i,\geq}\}$. By repeatedly substituting each variable $x_{j'}^{i'}$ in $\mathbf{a}_j^{i,\leq}$ (resp. $\mathbf{a}_j^{i,\geq}$) using $\mathbf{a}_{j'}^{i',\leq}$ or $\mathbf{a}_{j'}^{i',\geq}$ according to the coefficient of $x_{j'}^{i'}$, until no further substitution is possible, $\mathbf{a}_j^{i,\leq}$ (resp. $\mathbf{a}_j^{i,\geq}$) will be a linear combination over the input variables of the DNN. We denote by $f_j^{i,\leq}$ and $f_j^{i,\geq}$ the resulting linear combinations of $\mathbf{a}_j^{i,\leq}$ and $\mathbf{a}_j^{i,\geq}$. Then, the concrete lower bound l_j^i (resp. concrete upper bound u_j^i) of the neuron $\mathbf{x}_j^i$ can be derived using the input region $\mathcal{I}$ and $f_j^{i,\leq}$ (resp. $f_j^{i,\geq}$). All the abstract elements $\mathcal{A}_j^{i,\sharp}$ are required to satisfy the domain invariant: $\gamma(\mathcal{A}_j^{i,\sharp}) \subseteq [l_j^i, u_j^i]$. We denote by $\mathcal{A}_j^i$ the abstract element $\langle f_j^{i,\leq}, f_j^{i,\geq}, l_j^i, u_j^i\rangle$. For an affine function $\mathbf{x}^i = \mathbf{W}^i\mathbf{x}^{i-1} + \mathbf{b}^i$, the abstract affine transformer sets $\mathbf{a}^{i,\leq} = \mathbf{a}^{i,\geq} = \mathbf{W}^i\mathbf{x}^{i-1} + \mathbf{b}^i$. Given the abstract element $\mathcal{A}_j^{i,\sharp} = \langle \mathbf{a}_j^{i,\leq}, \mathbf{a}_j^{i,\geq}, l_j^i, u_j^i\rangle$ of the neuron $\mathbf{x}_j^i$, $\mathcal{A}_j^{i+1,\sharp}$ of the neuron $\mathbf{x}_j^{i+1} = \mathrm{ReLU}(\mathbf{x}_j^i)$ have three cases as follows, where $\lambda_j^i = \frac{u_j^i}{u_j^i - l_j^i}$: i) if $l_j^i \geq 0$, then $\mathbf{a}_j^{i+1,\leq} = \mathbf{a}_j^{i+1,\geq} = \mathbf{x}_j^i$, $l_j^{i+1} = l_j^i$, $u_j^{i+1} = u_j^i$; ii) if $u_j^i \leq 0$, then $\mathbf{a}_j^{i+1,\leq} = \mathbf{a}_j^{i+1,\geq} = l_j^{i+1} = u_j^{i+1} = 0$; iii) if $l_j^i u_j^i < 0$, then $\mathbf{a}_j^{i+1,\geq} = \lambda_j^i(\mathbf{x}_j^i - l_j^i)$, $\mathbf{a}_j^{i+1,\leq} = \kappa \cdot \mathbf{x}_j^i$ where $\kappa \in \{0, 1\}$ such that the area of resulting shape by $\mathbf{a}_j^{i+1,\leq}$ and $\mathbf{a}_j^{i+1,\geq}$ is minimal, $l_j^{i+1} = \kappa \cdot l_j^i$ and $u_j^{i+1} = u_j^i$.

3 Our Approach

In the following, we fix a DNN $\mathcal{N}$ with $2d$ layers and a property $\langle \mathcal{I}, \mathcal{O}\rangle$.

3.1 Foundation of Quadapter

Consider a function f and an output set Y, the *preimage* $f^{-1}(Y)$ of the output set Y for f is the set $\{x \mid f(x) \in Y\}$. An *under-approximation* of $f^{-1}(Y)$ is a set $\mathcal{P}$ such that $\mathcal{P} \subseteq f^{-1}(Y)$.

Definition 3. *A set $\mathfrak{P} = \{\mathcal{P}^{2i} \mid i \in [d-1]\}$ is an under-approximate preimage of the output region $\mathcal{O}$ for the DNN $\mathcal{N}$ if for every $i \in [d-1]$, $\mathcal{P}^{2i} \subseteq \mathcal{N}^{-1}_{[2i+1:2d]}(\mathcal{O})$.*

Intuitively, $\mathcal{P}^{2i}$ (resp. $\mathcal{P}_j^{2i}$) is the preimage of the activation layer f_{2i+1} (resp. neuron $\mathbf{x}_j^{2i+1}$) w.r.t. the output region $\mathcal{O}$. Since it suffices to consider preimages of the activation layers in the set $\mathfrak{P}$ for computing bit-widths of affine layers, the preimages of the affine layers are excluded.

Proposition 1. *Let $\widehat{\mathcal{N}}^{2i}$ be a network obtained from $\mathcal{N}$ by quantizing the first $2i$ layers. If $\mathfrak{P} = \{\mathcal{P}^{2i} \mid i \in [d-1]\}$ is an under-approximate preimage of the output region $\mathcal{O}$ for the DNN $\mathcal{N}$, then $\widehat{\mathcal{N}}^{2i}_{[1:2i]}(\mathcal{I}) \subseteq \mathcal{P}^{2i} \Rightarrow \widehat{\mathcal{N}}^{2i} \models \langle \mathcal{I}, \mathcal{O}\rangle$.* □

Intuitively, Proposition 1 states that regardless of the quantization configurations of the first $2i$ layers, the property $\langle \mathcal{I}, \mathcal{O}\rangle$ is always preserved in the resulting

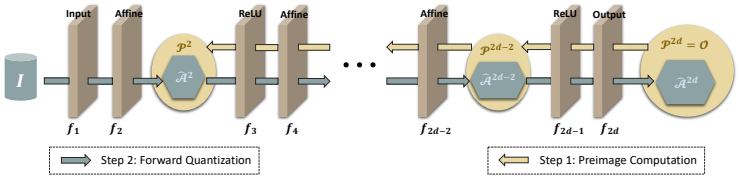


Fig. 2. An overview of our method.

QNN, as long as the reachable region of the quantized layer $\hat{f}_{2i}$ w.r.t. the input region $\mathcal{I}$ remains within the preimage $\mathcal{P}^{2i}$. This proposition allows us to repeatedly compute a quantization configuration ξ_i for each layer f_{2i} ($i \in [d]$), from the first affine layer to the output layer, that guarantees the reachable region of each quantized layer $\hat{f}_{2i}$ remains within its respective preimage $\mathcal{P}^{2i}$. Putting all the quantization configurations of the affine layers and the output layer together yields a quantization strategy Ξ that preserves the desired property $\langle \mathcal{I}, \mathcal{O} \rangle$.

However, it is non-trivial to compute the preimages $\mathcal{N}^{-1}_{[2i+1:2d]}(\mathcal{O})$ from the functions $\mathcal{N}^{-1}_{[2i+1:2d]}$ for $i \in [d-1]$. To resolve this issue, we propose to repeatedly compute a preimage $\mathcal{P}^{2i}$ of each activation layer f_{2i+1} starting from the output layer to the first activation layer by analyzing the function $\mathcal{N}^{-1}_{[2i+1:2i+2]}$ instead of the function $\mathcal{N}^{-1}_{[2i+1:2d]}$, according to the following proposition.

Proposition 2. *Let* $\mathfrak{P} = \{\mathcal{P}^{2i} \mid i \in [d-1]\}$ *be a set such that for every* $i \in [d-1]$, *i) if* $i = d-1$, $\mathcal{P}^{2i} \subseteq \mathcal{N}^{-1}_{[2i+1:2i+2]}(\mathcal{O})$; *ii) if* $i \leq d-2$, $\mathcal{P}^{2i} \subseteq \mathcal{N}^{-1}_{[2i+1:2i+2]}(\mathcal{P}^{2i+2})$. $\mathfrak{P}$ *is an under-approximate preimage of the output region* $\mathcal{O}$ *for the DNN* $\mathcal{N}$. □

3.2 Overview of Quadapter

Let $\mathcal{P}^{2d} = \mathcal{O}$. The overall workflow of Quadapter is depicted in Fig. 2 which consists of the following two steps:

- **Step 1: Preimage Computation.** We first compute an under-approximate preimage $\mathcal{P}^{2d-2}$ for the output layer s.t. $\mathcal{P}^{2d-2} \subseteq \mathcal{N}^{-1}_{[2d-1:2d]}(\mathcal{O})$, and then propagate it through the network until reaching the first affine layer. Finally, we obtain the under-approximate preimage $\mathfrak{P} = \{\mathcal{P}^{2i} \mid i \in [d-1]\}$ for the DNN $\mathcal{N}$ (the yellow part);
- **Step 2: Forward Quantization.** We then conduct a forward quantization procedure layer-by-layer to find a quantization configuration $\xi_i = \langle Q_i, F_i \rangle$ with minimal bit-width Q_i for each layer f_{2i}, ensuring that the reachable region characterized by the quantized abstract element $\widehat{\mathcal{A}}^{2i}$ (the blue part) is included in the preimage $\mathcal{P}^{2i}$, i.e., $\gamma(\widehat{\mathcal{A}}^{2i}) \subseteq \mathcal{P}^{2i}$ for $1 \leq i \leq d$.

The overall algorithm is given in Alg. 1. Given a DNN $\mathcal{N}$, a property $\langle \mathcal{I}, \mathcal{O} \rangle$, and the minimum (resp. maximum) fractional bit-width $\mathfrak{B}_l$ (resp. $\mathfrak{B}_u$) for each layer, we first apply DeepPoly on the DNN $\mathcal{N}$ w.r.t. input region $\mathcal{I}$ to obtain the abstract elements $\mathcal{A}^{2i}$ for $i \in [d]$. Then, the first for-loop computes the preimage

Algorithm 1: *Certified_Quantization*$(\mathcal{N}, \mathcal{I}, \mathcal{O}, \mathfrak{B}_l, \mathfrak{B}_u)$

```
Apply DeepPoly on the DNN 𝒩 w.r.t. ℐ to obtain abstract elements {𝒜^{2i} | 1 ≤ i ≤ d};
Let 𝒫^{2d} = 𝒪 and 𝒩̂ = 𝒩;
for i = d − 1 to 1 do
    𝒫^{2i} = UnderPreImage(𝒩, 𝒜^{2i}, 𝒫^{2i+2});        //get 𝒫^{2i} s.t. 𝒫^{2i} ⊆ 𝒩^{-1}_{[2i+1:2i+2]}(𝒫^{2i+2})
for i = 1 to d do
    ξ_i = ⊥;
    ℑ = the minimal bit-width to encode integer parts of W^{2i} and b^{2i} without overflow;
    for F = 𝔅_l to 𝔅_u do
        Quantize W^{2i}, b^{2i} w.r.t. ξ̆_i = ⟨F + ℑ, F⟩ on 𝒩̂ to obtain 𝒩̂^{2i};
        Apply DeepPoly on 𝒩̂^{2i}_{[1:2i]} w.r.t. ℐ to obtain 𝒜̂^{2i};
        if γ(𝒜̂^{2i}) ⊆ 𝒫^{2i} then
            ξ_i = ξ̆_i; 𝒩̂ = 𝒩̂^{2i};        //accept ξ̆_i and update quantized parameters
            break
    if ξ_i == ⊥ then return UNKNOWN;
return Ξ = {ξ_1, ..., ξ_d};
```

by invoking the function $\texttt{UnderPreImage}(\mathcal{N}, \mathcal{A}^{2i}, \mathcal{P}^{2i+2})$ which propagates $\mathcal{P}^{2i+2}$ to the preceding activation layer and returns the approximate preimage $\mathcal{P}^{2i}$ with $\mathcal{P}^{2i} \subseteq \mathcal{N}^{-1}_{[2i+1:2i+2]}(\mathcal{P}^{2i+2})$. The second for-loop performs a forward quantization procedure, where the i-th iteration is used to compute the quantization configuration ξ_i for layer f_{2i}. First, we obtain the minimal bit-width $\mathfrak{I}$ for the integer part of weights and biases to prevent overflow. Then, we iterate through all the possible configurations $\check{\xi}_i = \langle F + \mathfrak{I}, F\rangle$ by varying the fractional bit-width F from the smallest one $\mathfrak{B}_l$ to the largest one $\mathfrak{B}_u$. For each $F \in [\mathfrak{B}_l, \mathfrak{B}_u]$, we compute a partially quantized DNN $\widehat{\mathcal{N}}^{2i}$, where only the first i affine layers (and the output layer) are quantized using $\xi_1, \cdots, \xi_{i-1}, \check{\xi}_i$. Next, we apply DeepPoly on $\widehat{\mathcal{N}}^{2i}_{[1:2i]}$ w.r.t. the input region $\mathcal{I}$ to obtain the abstract element $\widehat{\mathcal{A}}^{2i}$ of the quantized layer $\hat{f}_{2i}$, resulting in reachable region as the blue part in Fig. 2. We then check whether this reachable region is strictly contained in the preimage $\mathcal{P}^{2i}$, i.e., $\gamma(\widehat{\mathcal{A}}^{2i}) \subseteq \mathcal{P}^{2i}$. If this is the case, we update ξ_i as $\check{\xi}_i$, stop the iteration, and proceed to find the quantization configuration ξ_{i+1} for the next layer f_{2i+2}. If there is no such quantization configuration, we return `UNKNOWN`.

Below, we present the details of function $\texttt{UnderPreImage}(\mathcal{N}, \mathcal{A}^{2i}, \mathcal{P}^{2i+2})$ and the method of checking the condition $\gamma(\widehat{\mathcal{A}}^{2i}) \subseteq \mathcal{P}^{2i}$. We first introduce the template of preimage $\mathcal{P}^{2i}$ utilized in this work.

3.3 Template $\mathcal{T}^{2i}$ of Preimage $\mathcal{P}^{2i}$

Given the abstract elements $\mathcal{A}^{2i} = \{\mathcal{A}^{2i}_j \mid j \in [n_{2i}]\}$ of the neurons in the layer f_{2i}, where $\mathcal{A}^{2i}_j = \langle f^{2i,\leq}_j, f^{2i,\geq}_j, l^{2i}_j, u^{2i}_j\rangle$, we define the template $\mathcal{T}^{2i}$ of the preimage $\mathcal{P}^{2i}$ as $\bigwedge_{j\in[n_{2i}]} \mathcal{T}^{2i}_j$, where $\mathcal{T}^{2i}_j = \{\mathbf{x}^{2i}_j \in \mathbb{R} \mid f^{2i,\leq}_j - \alpha^{2i}_j \leq \mathbf{x}^{2i}_j \leq f^{2i,\geq}_j + \beta^{2i}_j\}$, $\alpha^{2i}_j = \beta^{2i}_j = (\frac{u^{2i}_j - l^{2i}_j}{2})\chi^{2i}$, and χ^{2i} is an additional variable over the domain $\mathbb{R}$. Intuitively, $\mathcal{T}^{2i}_j$ is a scaling of $\mathcal{A}^{2i}_j$ using the scaling variable χ^{2i} and step $\frac{u^{2i}_j - l^{2i}_j}{2}$. Thus, $\mathcal{T}^{2i}_j$ is $\mathcal{A}^{2i}_j$ when $\chi^{2i} = 0$, and is super-region (resp. sub-region) of $\mathcal{A}^{2i}_j$ when $\chi^{2i} > 0$ (resp. $\chi^{2i} < 0$).

3.4 Details of Function `UnderPreImage`

We present an MILP-based method to implement $\texttt{UnderPreImage}(\mathcal{N}, \mathcal{A}^{2i}, \mathcal{P}^{2i+2})$. Given the abstract element $\mathcal{A}^{2i}$ and preimage $\mathcal{P}^{2i+2}$, we construct a maximization problem with objective function χ^{2i} subject to the constraints $\mathcal{T}^{2i} \subseteq \mathcal{N}^{-1}_{[2i+1:2i+2]}(\mathcal{P}^{2i+2})$, where $\mathcal{T}^{2i}$ is the template of $\mathcal{P}^{2i}$ with the scaling variable χ^{2i}. The solution, i.e., the value of χ^{2i}, yields the tightest under-approximate preimage $\mathcal{P}^{2i}$ such that $\mathcal{P}^{2i} \subseteq \mathcal{N}^{-1}_{[2i+1:2i+2]}(\mathcal{P}^{2i+2})$. Hence, the key is addressing $\mathcal{T}^{2i} \subseteq \mathcal{N}^{-1}_{[2i+1:2i+2]}(\mathcal{P}^{2i+2})$, for which we present an MILP-based method. We first express $\mathcal{T}^{2i} \subseteq \mathcal{N}^{-1}_{[2i+1:2i+2]}(\mathcal{P}^{2i+2})$ as the following maximization problem:

$$\text{maximize } \chi^{2i} \text{ s.t. } \mathcal{N}_{[2i+1:2i+2]}(\mathcal{T}^{2i}) \subseteq \mathcal{P}^{2i+2}. \tag{1}$$

However, Problem (1) is not an MILP, due to the "forall"-type of constraints. To address this issue, we construct the following minimization problem:

$$\text{minimize } \chi^{2i} \text{ s.t. } \mathbf{x}^{2i+2} \in \mathcal{N}_{[2i+1:2i+2]}(\mathcal{T}^{2i}) \wedge \mathbf{x}^{2i+2} \notin \mathcal{P}^{2i+2}. \tag{2}$$

Intuitively, given the solution to Problem (2), e.g., $\chi^{2i,*}_{\min}$, we can always get a value for χ^{2i} by subtracting an extremely small value from $\chi^{2i,*}_{\min}$. The resulting value of χ^{2i} is close to the optimal solution of Problem (1), within a negligible margin of error. Such a transformation to an "existential" constraint provides an alternative way for handling $\mathcal{T}^{2i} \subseteq \mathcal{N}^{-1}_{[2i+1:2i+2]}(\mathcal{P}^{2i+2})$, allowing the problem to be effectively tackled within the MILP framework.

Suppose $\mathcal{T}^{2i}_j = \{\mathbf{x}^{2i}_j \in \mathbb{R} \mid f^{2i,\leq}_j - \alpha^{2i}_j \leq \mathbf{x}^{2i}_j \leq f^{2i,\geq}_j + \beta^{2i}_j\}$ for $j \in [n_{2i}]$ and $\mathcal{P}^{2i+2}_k = \{\mathbf{x}^{2i+2}_k \in \mathbb{R} \mid f^{2i+2,\leq}_k - a^{2i+2}_k \leq \mathbf{x}^{2i+2}_k \leq f^{2i+2,\geq}_k + b^{2i+2}_k\}$ for $k \in [n_{2i+2}]$ and $i \leq d-2$. We reformulate Problem (2) as the following MILP problem:

$$\text{minimize } \chi^{2i} \text{ s.t. } \Psi_{\in\mathcal{I}} \cup \Psi_{\mathcal{T}^{2i}} \cup \Psi_{\mathcal{T}^{2i+1}} \cup \Psi_{\mathcal{T}^{2i+2}} \cup \Psi_{\notin\mathcal{P}^{2i+2}}, \tag{3}$$

where $\Psi_{\in\mathcal{I}}$ and $\Psi_{\notin\mathcal{P}^{2d}}$ will be given in Section 4 which entail $\mathbf{x} \in \mathcal{I}$ and $\mathbf{x}^{2d} \notin \mathcal{P}^{2d}$ respectively, as they depend on the property $\langle \mathcal{I}, \mathcal{O} \rangle$. $\Psi_{\mathcal{T}^{2i}}$, $\Psi_{\mathcal{T}^{2i+1}}$, $\Psi_{\mathcal{T}^{2i+2}}$, and $\Psi_{\notin\mathcal{P}^{2i+2}}$ are defined as follows ($\{\eta^{2i+1}_j, \eta^{2i+2}_j, \zeta^{2i+2}_j\}$ are Boolean variables):

- $\Psi_{\mathcal{T}^{2i}} = \{f^{2i,\leq}_j - \alpha^{2i}_j \leq \mathbf{x}^{2i}_j \leq f^{2i,\geq}_j + \beta^{2i}_j \mid j \in [n_{2i}]\}$ expressing template $\mathcal{T}^{2i}$;
- $\Psi_{\mathcal{T}^{2i+1}} = \{\mathbf{x}^{2i+1} \geq 0, \mathbf{x}^{2i+1} \geq \mathbf{x}^{2i}, \mathbf{x}^{2i+1} \leq \mathbf{M} \cdot \eta^{2i+1}_j, \mathbf{x}^{2i+1} \leq \mathbf{x}^{2i} + \mathbf{M} \cdot (1 - \eta^{2i+1}_j) \mid j \in [n_{2i+1}]\}$ encoding the activation layer f_{2i+1} (cf. [54]);
- $\Psi_{\mathcal{T}^{2i+2}} = \{\mathbf{x}^{2i+2}_j = \mathbf{W}^{2i+2}_{j,:}\mathbf{x}^{2i+1} + \mathbf{b}^{2i+2}_j \mid j \in [n_{2i+2}]\}$ encoding the affine layer f_{2i+2} (cf. [54]). Note that $\Psi_{\mathcal{T}^{2i}}$, $\Psi_{\mathcal{T}^{2i+1}}$ and $\Psi_{\mathcal{T}^{2i+2}}$ together express the condition $\mathbf{x}^{2i+2} \in \mathcal{N}_{[2i+1:2i+2]}(\mathcal{T}^{2i})$.
- $\Psi_{\notin\mathcal{P}^{2i+2}} = \left\{ \begin{array}{c} \mathbf{x}^{2i+2}_j > f^{2i+2,\geq}_j + b^{2i+2}_j + \mathbf{M} \cdot (\eta^{2i+2}_j - 1), \\ \mathbf{x}^{2i+2}_j \leq f^{2i+2,\geq}_j + b^{2i+2}_j + \mathbf{M} \cdot \eta^{2i+2}_j, \\ \mathbf{x}^{2i+2}_j \geq f^{2i+2,\leq}_j - a^{2i+2}_j - \mathbf{M} \cdot \zeta^{2i+2}_j, \\ \mathbf{x}^{2i+2}_j < f^{2i+2,\leq}_j - a^{2i+2}_j - \mathbf{M} \cdot (\zeta^{2i+2}_j - 1), \\ j \in [n_{2i+2}] \wedge \sum_{k=1}^{n_{2i+2}} \left(\eta^{2i+2}_k + \zeta^{2i+2}_k\right) \geq 1 \end{array} \right\}$ expressing the condition $\mathbf{x}^{2i+2} \notin \mathcal{P}^{2i+2}$.

Theorem 1. *Problems (2) and (3) are equivalent.* □

3.5 Checking $\gamma(\widehat{\mathcal{A}}^{2i}) \subseteq \mathcal{P}^{2i}$

Fix the abstract elements $\widehat{\mathcal{A}}^{2i} = \{\widehat{\mathcal{A}}_j^{2i} \mid j \in [n_{2i}]\}$ for the quantized layer $\hat{f}_{2i}$ with $\widehat{\mathcal{A}}_j^{2i} = \langle \hat{f}_j^{2i,\leq}, \hat{f}_j^{2i,\geq}, \hat{l}_j^{2i}, \hat{u}_j^{2i} \rangle$, we have $\gamma(\widehat{\mathcal{A}}_j^{2i}) = \{\mathbf{x}_j^{2i} \in \mathbb{R} \mid \hat{f}_j^{2i,\leq} \leq \mathbf{x}_j^{2i} \leq \hat{f}_j^{2i,\geq}\}$. Let $\mathcal{P}_j^{2i} = \{\mathbf{x}_j^{2i} \in \mathbb{R} \mid f_j^{2i,\leq} - a_j^{2i} \leq \mathbf{x}_j^{2i} \leq f_j^{2i,\geq} + b_j^{2i}\}$ for $j \in [n_{2i}]$ be the preimage obtained by the function `UnderPreImage` for $i \leq d-1$, where a_j^{2i} and b_j^{2i} are real-valued numbers.

Since reformulating the problem of checking $\gamma(\widehat{\mathcal{A}}^{2i}) \subseteq \mathcal{P}^{2i}$ into an MILP problem directly is infeasible due to its inherent nature of "forall"-type constraint, we instead check the negation of this statement.

Let $\Phi_{\notin \mathcal{P}^{2i}}$ be the following set of the linear constraints:

$$\Phi_{\notin \mathcal{P}^{2i}} = \Psi_{\in \mathcal{I}} \cup \left\{ \begin{array}{c} f_j^{2i,\geq} + b_j^{2i} + \mathbf{M} \cdot (\eta_j^{2i} - 1) < \hat{f}_j^{2i,\geq} \leq f^{2i,\geq} + b_j^{2i} + \mathbf{M} \cdot \eta_j^{2i}, \\ f^{2i,\leq} - a_j^{2i} - \mathbf{M} \cdot \zeta_j^{2i} \leq \hat{f}_j^{2i,\leq} < f^{2i,\leq} - a_j^{2i} - \mathbf{M} \cdot (\zeta_j^{2i} - 1), \\ j \in [n_{2i}], \qquad \sum_{k=1}^{n_{2i}} \left(\eta_k^{2i} + \zeta_k^{2i}\right) \geq 1 \end{array} \right\}$$

where η_j^{2i} and ζ_j^{2i} are two additional Boolean variables, and $\Psi_{\in \mathcal{I}}$ and $\Phi_{\notin \mathcal{P}^{2d}}$ will be given in Section 4 such that $\Psi_{\in \mathcal{I}}$ entails $\mathbf{x} \in \mathcal{I}$ and $\neg \Phi_{\notin \mathcal{P}^{2d}}$ entails $\gamma(\widehat{\mathcal{A}}^{2d}) \subseteq \mathcal{P}^{2d}$ respectively, as they depend on the property $\langle \mathcal{I}, \mathcal{O} \rangle$.

Theorem 2. *If $\Phi_{\notin \mathcal{P}^{2i}}$ does not hold, then $\gamma(\widehat{\mathcal{A}}^{2i}) \subseteq \mathcal{P}^{2i}$.* □

4 Applications: Robustness and Backdoor-freeness

4.1 Certified Quantization for Robustness

We use Alg. 1 to synthesize quantization strategies for preserving robustness.

Definition 4. *Let $\mathcal{N} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{2d}}$ be a DNN, $\mathcal{I}_{\mathbf{u}}^r = \{\mathbf{x} \in \mathbb{R}^{n_0} \mid ||\mathbf{x} - \mathbf{u}||_\infty \leq r\}$ be a perturbation region around an input $\mathbf{u} \in \mathbb{R}^{n_0}$, and $\mathcal{O}_g = \{\mathbf{x}^{2d} \in \mathbb{R}^{n_{2d}} \mid argmax(\mathbf{x}^{2d}) = g\}$ be the output region corresponding to a specific class g. Then, $\langle \mathcal{I}_{\mathbf{u}}^r, \mathcal{O}_g \rangle$ is a (local) robustness property of the DNN $\mathcal{N}$.*

We now give the encoding details that are not covered in Section 3, i.e., $\Psi_{\in \mathcal{I}}$ and $\Psi_{\notin \mathcal{P}^{2d}}$ in Problem (3), and $\Phi_{\notin \mathcal{P}^{2d}}$ in Section 3.5 for the property $\langle \mathcal{I}_{\mathbf{u}}^r, \mathcal{O}_g \rangle$[6]:

- $\Psi_{\in \mathcal{I}} = \{\max(\mathbf{u}_j - r, 0) \leq \mathbf{x}_j \leq \min(\mathbf{u}_j + r, 1) \mid j \in [n_0]\}$ specifying the feasible input range $\mathcal{I}_{\mathbf{u}}^r$;
- $\Psi_{\notin \mathcal{P}^{2d}} = \left\{ \begin{array}{c} \mathbf{x}_g^{2d} + \mathbf{M} \cdot (\eta_j^{2d} - 1) \leq \mathbf{x}_j^{2d} \leq \mathbf{x}_g^{2d} + \mathbf{M} \cdot \eta_j^{2d}, \\ j \in [n_{2d}] \setminus g, \qquad \sum_{k \in [n_{2d}] \setminus g} \eta_k^{2d} \geq 1 \end{array} \right\}$ stating $\mathbf{x}^{2d} \notin \mathcal{O}_g$, i.e., $\text{argmax}(\mathbf{x}^{2d}) \neq g$, where η_j^{2d} is a Boolean variable;
- $\Phi_{\notin \mathcal{P}^{2d}} = \left\{ \begin{array}{c} \hat{f}_g^{2d,\leq} + \mathbf{M} \cdot (\eta_j^{2d} - 1) \leq \hat{f}_j^{2d,\geq} \leq \hat{f}_g^{2d,\leq} + \mathbf{M} \cdot \eta_j^{2d}, \\ j \in [n_{2d}] \backslash g, \qquad \sum_{k \in [n_{2d}] \setminus g} \eta_k^{2d} \geq 1 \end{array} \right\}$ whose unsatisfiability ensuring $\gamma(\widehat{\mathcal{A}}^{2d}) \subseteq \mathcal{O}_g$, where η_j^{2d} is a Boolean variable.

[6] For simplicity, we assume that the output layer of $\mathcal{N}$ has a unique maximum value for any given input. This assumption can be avoided by adapting $\Psi_{\notin \mathcal{P}^{2d}}$ and $\Phi_{\notin \mathcal{P}^{2d}}$.

The soundness of the algorithm is captured by the theorem below.

Theorem 3. $\Psi_{\in\mathcal{I}} \Leftrightarrow \mathbf{x} \in \mathcal{I}^r_{\mathbf{u}}$, $\Psi_{\notin\mathcal{P}^{2d}} \Leftrightarrow \mathbf{x}^{2d} \notin \mathcal{O}_g$, $\neg\Phi_{\notin\mathcal{P}^{2d}} \Rightarrow \gamma(\widehat{\mathcal{A}}^{2d}) \subseteq \mathcal{O}_g$. □

4.2 Certified Quantization for Backdoor-freeness

Given a DNN $\mathcal{N} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{2d}}$ and an input $\mathbf{u} \in \mathbb{R}^{n_0}$, assume that the 2D-shape of $\mathbf{u}$ is a rectangle (h_u, w_u) (i.e., $n_0 = h_u \times w_u$). A backdoor trigger is any 2D input $\mathbf{s} \in \mathbb{R}^{h_s \times w_s}$ with a shape of rectangle (h_s, w_s) such that $h_s \leq h_u$ and $w_s \leq w_u$. We use $\mathbf{u}[x, y]$ to denote the element located in the x-th row and y-th column within the 2D-input $\mathbf{u}$. Let (h_p, w_p) denote the position of (i.e., the top-left corner of) the trigger $\mathbf{s}$ such that $h_p + h_s \leq h_u$ and $w_p + w_s \leq w_u$. Then, $\mathbf{u}^{\mathbf{s}}$ is the stamped input where $\mathbf{u}^{\mathbf{s}}[x, y] = \mathbf{s}[x - h_p, y - w_p]$ if $h_p \leq x \leq h_p + h_s \wedge w_p \leq y \leq w_p + w_s$, and $\mathbf{u}^{\mathbf{s}}[x, y] = \mathbf{u}[x, y]$ otherwise.

Definition 5. *Let $\mathcal{N} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{2d}}$ be a DNN, (h_s, w_s), (h_p, w_p), t, and θ be the shape, position, target class, and attack success rate of potential triggers. Then, the DNN $\mathcal{N}$ satisfies the backdoor-freeness property if there does not exist a backdoor trigger $\mathbf{s}$ which has an attack success rate of at least θ, i.e, the probability of $\mathcal{N}(\mathbf{u}^{\mathbf{s}}) = t$ for any $\mathbf{u} \in \mathbb{R}^{n_0}$ is at least θ [37].*

Given an input $\mathbf{u} \in \mathbb{R}^{n_0}$, let $\langle \mathcal{I}^B_{\mathbf{u}}, \mathcal{O}^B_t \rangle$ be a property such that $\mathcal{I}^B_{\mathbf{u}} = \{\mathbf{u}^{\mathbf{s}} \in \mathbb{R}^{n_0} \mid \mathbf{s} \in \mathbb{R}^{h_s \times w_s}$ is any trigger at position $(h_p, w_p)\}$ and $\mathcal{O}^B_t = \{\mathbf{x}^{2d} \in \mathbb{R}^{n_{2d}} \mid \text{argmax}(\mathbf{x}^{2d}) \neq t\}$. Intuitively, $\langle \mathcal{I}^B_{\mathbf{u}}, \mathcal{O}^B_t \rangle$ entails that no trigger exists whereby the input $\mathbf{u}$, once stamped, would be classified as class t.

The overall algorithm is given in Alg. 2 by applying a hypothesis testing (a type I/II error σ/ϱ and a half-width of the indifference region δ), i.e., the SPRT algorithm [1]. The while loop first keeps randomly selecting a set of K properties and collects the preimage with the highest value of the scaling variable of the first affine layer, along with the property, until one of the hypotheses is accepted. When the null hypothesis H_0 is accepted (line 9), we try to find a shared quantization strategy for all the properties collected before, following Alg. 1, with the innermost for-loop to traverse all properties. Due to space limitations, details of the hypothesis testing and input parameters are explained in [50].

We now give the encoding details that are not covered in Section 3, i.e., $\Psi_{\in\mathcal{I}}$ and $\Psi_{\notin\mathcal{P}^{2d}}$ in Problem (3), and $\Phi_{\notin\mathcal{P}^{2d}}$ in Section 3.5 for the property $\langle \mathcal{I}^B_{\mathbf{u}}, \mathcal{O}^B_t \rangle$:

- $\Psi_{\in\mathcal{I}} = \left\{ \begin{array}{c} 0 \leq \mathbf{x}[a, b] \leq 1 \text{ if } h_p \leq a \leq h_p + h_s \wedge w_p \leq b \leq w_p + w_s, \\ \mathbf{x}[a, b] = \mathbf{u}[a, b] \text{ otherwise} \end{array} \right\}$;
- $\Psi_{\notin\mathcal{P}^{2d}} = \{\mathbf{x}^{2d}_t \geq \mathbf{x}^{2d}_j \mid j \in [n_{2d}]\}$;
- $\Phi_{\notin\mathcal{P}^{2d}} = \{\hat{f}^{2d,\leq}_j \leq \hat{f}^{2d,\geq}_t \mid j \in [n_{2d}] \setminus t\}$.

Theorem 4. *(1) $\Psi_{\in\mathcal{I}} \Leftrightarrow \mathbf{x} \in \mathcal{I}^B_{\mathbf{u}}$, $\Psi_{\notin\mathcal{P}^{2d}} \Leftrightarrow \mathbf{x}^{2d} \notin \mathcal{O}^B_t$, $\neg\Phi_{\notin\mathcal{P}^{2d}} \Rightarrow \gamma(\widehat{\mathcal{A}}^{2d}) \subseteq \mathcal{O}^B_t$, and (2) there is sufficient evidence (subject to type I error σ and type II error ϱ) that there are no backdoor attacks with the featured triggers within the QNN obtained by Alg. 2.* □

Algorithm 2: $CQ\_Backdoor(\mathcal{N}, \mathfrak{B}_l, \mathfrak{B}_u, (h_s, w_s), (h_p, w_p), t, \theta, K, \epsilon, \sigma, \varrho, \delta)$

Let $\mathcal{P}^{2d} = \mathcal{O}_t^B$, $\widehat{\mathcal{N}} = \mathcal{N}$, $All_{\mathcal{I}} = \emptyset$, $All_{\mathfrak{P}} = \emptyset$, $n = z = 0$;
Let $p_0 = 1 - \theta^K + \delta$, $p_1 = 1 - \theta^K - \delta$;
while *true* **do**
  $n = n + 1$;
  Randomly select a set of K properties $X = \{\langle \mathcal{N}^B_{\mathbf{u_1}}, \mathcal{O}_t^B \rangle, \ldots, \langle \mathcal{N}^B_{\mathbf{u_K}}, \mathcal{O}_t^B \rangle\}$;
  Compute under-approximate preimage for each property in X (cf. Alg. 1), and let $\langle \mathcal{I}^B_{\mathbf{u}^*}, \mathcal{O}_t^B \rangle$ be the property with the highest value of the scaling variable χ^{2*} for layer f_2 and $\mathfrak{P}^*$ be the corresponding under-approximate preimage;
  if $\chi^{2*} \geq \epsilon$ **then**
    $z = z + 1$; $All_{\mathcal{I}}$.append($\mathcal{I}^B_{\mathbf{u}^*}$); $All_{\mathfrak{P}}$.append($\mathfrak{P}^*$);
  if $\frac{p_1^z}{p_0^z} \times \frac{(1-p_1)^{n-z}}{(1-p_0)^{n-z}} \leq \frac{\varrho}{1-\sigma}$ **then**
    for $i = 1$ to d **do**
      $\xi_i = \perp$;
      Let $\mathfrak{I}$ be the minimal bit-width to encode integer parts of $\mathbf{W}^{2i}$ and $\mathbf{b}^{2i}$ without overflow;
      for $F = \mathfrak{B}_l$ to $\mathfrak{B}_u$ **do**
        Quantize $\mathbf{W}^{2i}$, $\mathbf{b}^{2i}$ w.r.t. $\check{\xi}_i = (F + \mathfrak{I}, F)$ on $\widehat{\mathcal{N}}$ and obtain $\widehat{\mathcal{N}}^{2i}$;
        for $k = 1$ *to* z **do**
          Apply DeepPoly on $\widehat{\mathcal{N}}^{2i}_{[1:2i]}$ w.r.t. $All_{\mathcal{I}}[k]$ and obtain $\widehat{\mathcal{A}}^{2i,k}$;
          if $\gamma(\widehat{\mathcal{A}}^{2i,k}) \subseteq All_{\mathfrak{P}}[k][i]$ *is* UNSAT **then**
            break *//jump to line 13 for next iteration of F*
        $\xi_i = \check{\xi}_i$; $\widehat{\mathcal{N}} = \widehat{\mathcal{N}}^{2i}$; *//accept $\check{\xi}_i$ and update quantized parameters*
        break *//jump to line 10 to quantize next layer f_{2i+2}*
      if $\xi_i == \perp$ **then return** UNKNOWN;
    return $\Xi = \{\xi_1, \ldots, \xi_d\}$
  else if $\frac{p_1^z}{p_0^z} \times \frac{(1-p_1)^{n-z}}{(1-p_0)^{n-z}} \leq \frac{1-\varrho}{\sigma}$ **then**
    return UNKNOWN;

Table 1. Benchmarks of DNNs on MNIST and Fashion-MNIST.

Accuracy	P1: 2×100	P2: 4×100	P3: 6×100	P4: 4×512
MNIST	97.79%	97.63%	97.39%	98.17%
Fashion-MNIST	87.86%	88.45%	87.22%	88.7%

5 Evaluation

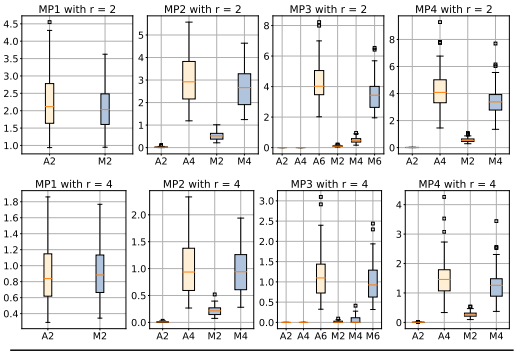
We have implemented our methods as a tool Quadapter with Gurobi [11] as the back-end MILP solver. To address the numerical stability problem using big-M, we use alternative formulations for the ReLU activation function and tighter bounds for other big-M. Details refer to [50]. All experiments are run on a machine with Intel(R) Xeon(R) Platinum 8375C CPU@2.90GHz, using 30 threads in total. The time limit for each task is 2 hours.

Benchmarks. We train 8 DNNs using the MNIST [20] and Fashion-MNIST [46] datasets based on their popularity in previous verification studies with comparable size [9,12,19,36,37]. To evaluate the performance of Quadapter, these DNNs vary in architectures, whose details are given in Table 1, where $x \times y$ means that the network has x hidden layers and y neurons per each hidden layer. Hereafter, we use MPx (resp. FPx) with $x \in \{1, 2, 3, 4\}$ to denote the network of architecture Px trained using MNIST (resp. Fashion-MNIST).

5.1 Performance of `UnderPreImage` Function

We evaluate the effectiveness and efficiency of the MILP-based method introduced in Section 3.4 for computing the under-approximate preimage of DNNs MPx with $x \in \{1, 2, 3, 4\}$ for robustness properties. Specifically, we randomly select 50 inputs from the test set of MNIST and set the perturbation radius as $r \in \{2, 4\}$, resulting in a total of 400 robustness properties, each of which can be certified using DeepPoly. The time limit for each computation task is 2 hours. We also implement an abstraction-based method (ABS) to compute the preimages for comparative analysis. Details refer to [50].

The results are depicted in Fig. 3. The boxplot shows the distribution of the values of the scaling variables obtained by the two methods for each layer, where Ax and Mx denote the results of layer f_x obtained by the ABS and MILP methods, respectively. (Note that some Ax and Mx may be missing because the DNN has no f_x layer.) The table reports the average computation time in seconds, where (i) indicates the number of tasks that run out of time in 2 hours. We find that compared to the MILP method, the ABS method tends to obtain significantly smaller values for scaling variables in earlier layers, albeit requiring less time. It is mainly attributed to the inherent over-approximation in the abstract transformers. Note that the scaling variable for the last affine layer returned by the ABS method is typically larger than that obtained via the MILP method. However, we argue that the scaling variables of preceding layers are more significant, with larger values being preferable for a successful forward quantization process subsequently. Therefore, we opt for the MILP method to implement `UnderPreImage`, despite its longer execution time. Integrating both methods is an interesting direction for future work.



Perturbation Radius	MP1		MP2		MP3		MP4	
	ABS	MILP	ABS	MILP	ABS	MILP	ABS	MILP
$r = 2$	4.80	7.18	10.89	24.86	15.47	38.00	54.98	565.0
$r = 4$	4.80	9.13	11.52	35.73	16.33	49.86	58.48	1,763(5)

Fig. 3. Results of preimage computation.

Unsurprisingly, we also observe the decrease of scaling variables as r increases or the layer index decreases. The former is attributed to the enlargement of the reachable region of each neuron with an increasing r, leading to a diminution in the theoretical range of the amplification. The latter is because we propagate the preimage towards the input layer and the preimage returned by `UnderPreImage` increasingly under-approximates the ground truth. Additionally, we find a more pronounced impact of the number of layers in a DNN on the scaling, as opposed to the impact of the number of neurons per each layer. For example, when $r = 4$, while the scaling of the last affine layer is similar across MP2, MP3, and MP4,

Table 2. Certified quantization strategy synthesis results for robustness.

Network	Quadapter with $(\mathfrak{B}_l, \mathfrak{B}_l) = (1, 16)$						Quadapter* with $(\mathfrak{B}_l, \mathfrak{B}_l) = (2, 16)$					
	#S	#F	Bit-width	Acc.	PTime(s)	QTime(s)	#S	#F	Bit-width	Acc.	PTime(s)	QTime(s)
MP1	250	0	(6,3)	95.57%	8.17	10.80	250	0	(4,4)	96.59%	8.75	3.96
MP2	248	2	(8,6,3)	94.11%	30.49	29.18	249	1	(5,4,4)	96.35%	31.60	13.38
MP3	175	75	(11,9,6,3)	95.47%	39.55	58.63	208	32	(8,5,4,4)	96.08%	42.37	78.22
MP4	228	0	(8,6,3)	94.48%	1,066	160.2	227	0	(4,4,4)	96.97%	1,066	32.99
FP1	250	0	(6,4)	78.54%	6.93	10.48	250	0	(4,4)	83.89%	7.80	3.63
FP2	249	1	(8,6,3)	79.43%	29.82	28.86	248	2	(5,4,4)	84.56%	33.13	11.39
FP3	180	70	(11,9,6,3)	74.23%	36.90	59.45	222	26	(7,5,4,4)	85.74%	39.71	39.44
FP4	221	2	(8,7,3)	75.98%	564.0	160.7	220	2	(4,4,4)	83.07%	565.3	64.23

a notable divergence is observed as the preimage computation progresses to the preceding layer, i.e., the scaling of f_4 in MP3 largely diminishes compared to that of f_2 in MP2 and MP4, and even approaches zero in some tasks. We conjecture that as the DNN gets deeper and r gets larger, DeepPoly shows enhanced efficacy in its symbolic propagation such that the region delineated by $\mathcal{A}^{2i+2}$ becomes significantly tighter compared to the region confined by $\mathcal{N}_{[2i+1:2i+2]}(\mathcal{A}^{2i})$. Finally, we find that the preimage computation time is predominantly impacted by the number of neurons per each layer (e.g., MP2 vs MP4).

5.2 Certified Quantization for Robustness

We evaluate Quadapter in terms of robustness properties on all the networks listed in Table 1 with the fractional bit-width range $[\mathfrak{B}_l, \mathfrak{B}_u] = [1, 16]$. For each network, we randomly select 50 inputs from the test set of the respective dataset and set the perturbation radius as $r \in \{1, 2, 3, 4, 5\}$. It results in a total of 250 synthesis tasks for each network, each of which can be certified by DeepPoly.

The results are reported in Columns 2 to 7 in Table 2. Columns (#S) and (#F) list the number of quantization successes and quantization failures due to small values of scaling variables. Column (Bit-width) lists the average bit-width for each layer within the quantization strategies synthesized by Quadapter and Column (Acc.) lists the average accuracy of the resulting QNNs. Columns (PTime) and (QTime) show the average execution time in seconds for the preimage computation and forward quantization procedures, respectively. Overall, Quadapter solves almost all the tasks of MPx and FPx for $x \in \{1, 2\}$, and most tasks of MP4 and FP4, where all timeout cases occur in the preimage computation process. For MP3 and FP3, all quantization failures are due to the excessively small preimage returned by `UnderPreImage`, posing a great challenge in finding a feasible quantization strategy, which requires that the quantized region must be strictly included within the preimage. Given the distribution shift phenomenon shown in Fig. 1, we hypothesize that it may be alleviated by relaxing such "strict-inclusion" requirement on the early layer quantization while not compromising soundness. Thus, we next relax the restriction by permitting the quantized regions of some portion of neurons, e.g., 25%, in each affine layer (except the output layer to guarantee the soundness of the approach) to deviate from the preimage returned by `UnderPreImage`. Note that, when using the

Network	$(h_s, w_s) = (3, 3)$					$(h_s, w_s) = (5, 5)$				
	#S	#F	Bit-width	Acc.	Time(s)	#S	#F	Bit-width	Acc.	Time(s)
MP1	49	0	(12,4)	96.87%	474.9	44	6	(10,4)	96.94%	595.4
MP2	43	7	(15,8,4)	97.06%	1,100	26	24	(12,8,4)	97.09%	1,293
FP1	50	0	(12,4)	85.29%	466.7	46	4	(9,4)	85.38%	511.4
FP2	40	10	(13,7,4)	86.61%	1,114	32	18	(11,8,4)	86.71%	1,063

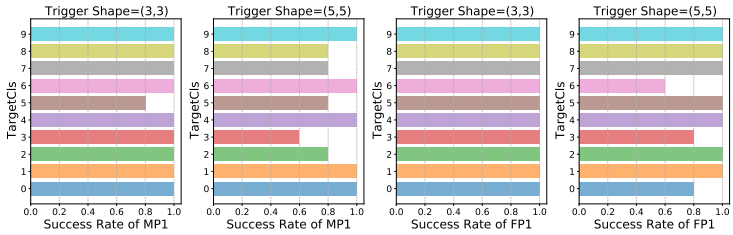


Fig. 4. Certified quantization strategies synthesis results for backdoor-freeness.

relaxed version of our tool, named Quadapter*, we set $\mathfrak{B}_l = 2$ to circumvent situations where the use of the smallest bit-width (specifically, 1-bit), while theoretically yielding a viable solution for the current layer, may lead to a lack of feasible quantization for subsequent layers. Experimental results are shown in Columns 8 to 13 in Table 2. We observe that Quadapter* usually synthesizes quantization strategies with smaller bit-widths for earlier layers, larger bit-widths for the last later, better accuracy, and solves more tasks on average. While the accuracy drops slightly, it also slightly drops using the same but non-certified quantization scheme and our certified quantization achieved comparable accuracy [50].

5.3 Certified Quantization for Backdoor-freeness

We evaluate Quadapter in terms of backdoor-freeness on MP1, MP2, FP1 and FP2. For each network, we randomly select 5 trigger positions and consider all the 10 output classes as target labels of the backdoor attacks with two shapes of triggers, i.e., $h_s = w_s = 3$ and $h_s = w_s = 5$, resulting in $5 \times 10 \times 2 = 100$ backdoor-freeness properties. Following [37], we set the input parameters of Alg. 2 as $(\mathfrak{B}_l, \mathfrak{B}_u) = (2, 16)$, $\theta = 0.9$, $K = 5$, $\epsilon = 0.01$, and $\sigma = \varrho = \delta = 0.05$. Note that these parameters do not affect the soundness of Alg. 2.

The results are given in Fig. 4. We observe that for $(h_s, w_s) = (3, 3)$, Quadapter solves almost all the tasks of MP1 and FP1, and most tasks on MP2 and FP2. For $(h_s, w_s) = (5, 5)$, over half of the tasks are solved by Quadapter. All the quantization failures (due to small values of scaling variables) may be solvable with the relaxed version of Quadapter which is left as future work. The histogram shows the distribution of target classes in the solved tasks on MP1 and FP1, where the x-axis gives the synthesis success rate. We also observe that Quadapter is more likely to successfully find certified quantization strategies w.r.t. target classes $\{0, 1, 4, 6, 9\}$ on MP1 and target classes $\{1, 2, 4, 5, 7, 8, 9\}$ on FP1, compared to its efficacy w.r.t. other classes. Due to the black-box nature, we currently cannot explain the discrepancy in performance between target classes.

6 Related Work

Numerous methods have been proposed to verify (local) robustness of DNNs (e.g., [7,10,17,40,45,47]) and QNNs (e.g., [9,12,14,19,52–54]). Recently, backdoor-freeness verification for DNNs has been explored leveraging a similar hypothesis testing method [37]. Methods for verifying quantization error bound [30,35,36,51] and Top-1 equivalence [16] between DNNs and QNNs have also been proposed. Except for [16], these works only verify properties without adjusting quantization strategies for falsified properties. The concurrent work [16] iteratively searches for a quantization strategy and verifies Top-1 equivalence after quantization, refining strategies if equivalence is violated. However, it does not support general properties (e.g., backdoor freeness or robustness of multi-label classification [6]). Additionally, [16] requires frequent equivalence verification, which is computationally expensive and inefficient (e.g., networks with 100 neurons in 20 minutes). Comparison experiments are given in [50].

The primary contribution of this work is the first certified quantization strategy synthesis approach utilizing preimage computation as a crucial step. Hence, any (under-approximate) preimage computation methods can be integrated. [28] introduced an exact preimage computation method that, while precise, is impractical due to its exponential time complexity. The inverse abstraction approach [4] circumvents the intractability of exact preimage computation by using symbolic interpolants [2] for compact symbolic abstractions of preimages. However, it still faces scalability issues due to the complexity of the interpolation process. [18,49] considered over-approximate preimages, which are unsuitable for our purpose.

Quantization-aware training has been studied to improve robustness for a given fixed quantization strategy [19, 23, 24, 41, 43], but only [19] provides robustness guarantees by lifting abstract interpretation-based training [29] from DNNs to QNNs. In contrast, our work aims to obtain a better quantification strategy for preserving given properties. Thus, our work is orthogonal to and could be combined with them. We leave this as interesting future work.

7 Conclusion

In this work, we have presented a pioneering method Quadapter to synthesize a fine-grained quantization strategy such that the desired properties are preserved within the resulting quantized network. We have implemented our methods as an end-to-end tool and conducted extensive experiments to demonstrate the effectiveness and efficiency of Quadapter in preserving robustness and backdoor-freeness properties. For future work, it would be interesting to explore the adaptation of Quadapter to other activation functions and network architectures, towards which this work makes a significant step.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Transactions on Modeling and Computer Simulation **28**(1), 1–39 (2018)
2. Albarghouthi, A., McMillan, K.L.: Beautiful interpolants. In: Proceedings of the 25th International Conference on Computer Aided Verification. pp. 313–329. Springer (2013)
3. Amir, G., Wu, H., Barrett, C.W., Katz, G.: An SMT-based approach for verifying binarized neural networks. In: Proceedings of the 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). vol. 12652, pp. 203–222 (2021). https://doi.org/10.1007/978-3-030-72013-1_11
4. Dathathri, S., Gao, S., Murray, R.M.: Inverse abstraction of neural networks using symbolic interpolation. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI). pp. 3437–3444 (2019). https://doi.org/10.1609/AAAI.V33I01.33013437
5. Dong, S., Wang, P., Abbas, K.: A survey on deep learning and its applications. Comput. Sci. Rev. **40**, 100379 (2021). https://doi.org/10.1016/J.COSREV.2021.100379
6. Eleftheriadis, C., Kekatos, N., Katsaros, P., Tripakis, S.: On neural network equivalence checking using SMT solvers. In: Proceedings of the 20th International Conference on Formal Modeling and Analysis of Timed Systems. vol. 13465, pp. 237–257 (2022)
7. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.T.: AI^2: safety and robustness certification of neural networks with abstract interpretation. In: Proceedings of the 2018 IEEE Symposium on Security and Privacy. pp. 3–18 (2018)
8. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. In: Low-Power Computer Vision, pp. 291–326. Chapman and Hall/CRC (2022)
9. Giacobbe, M., Henzinger, T.A., Lechner, M.: How many bits does it take to quantize your neural network? In: Proceedings of the 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 79–97 (2020). https://doi.org/10.1007/978-3-030-45237-7_5
10. Guo, X., Wan, W., Zhang, Z., Zhang, M., Song, F., Wen, X.: Eager falsification for accelerating robustness verification of deep neural networks. In: Proceedings of the 32nd IEEE International Symposium on Software Reliability Engineering. pp. 345–356 (2021)
11. Gurobi: A most powerful mathematical optimization solver. https://www.gurobi.com/ (2018)
12. Henzinger, T.A., Lechner, M., Zikelic, D.: Scalable verification of quantized neural networks. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI). pp. 3787–3795 (2021). https://doi.org/10.1609/AAAI.V35I5.16496
13. Hong, S., Panaitescu-Liess, M., Kaya, Y., Dumitras, T.: Qu-anti-zation: Exploiting quantization artifacts for achieving adversarial outcomes. In: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS). pp. 9303–9316 (2021)
14. Huang, P., Wu, H., Yang, Y., Daukantas, I., Wu, M., Zhang, Y., Barrett, C.W.: Towards efficient verification of quantized neural networks. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence. pp. 21152–21160 (2024). https://doi.org/10.1609/AAAI.V38I19.30108

15. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A.G., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2704–2713 (2018)
16. Jr., J.B.P.M., de Lima Filho, E.B., Bessa, I., Manino, E., Song, X., Cordeiro, L.C.: Counterexample guided neural network quantization refinement. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **43**(4), 1121–1134 (2024). https://doi.org/10.1109/TCAD.2023.3335313
17. Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: Proceedings of the 29th International Conference on Computer Aided Verification. pp. 97–117 (2017)
18. Kotha, S., Brix, C., Kolter, J.Z., Dvijotham, K., Zhang, H.: Provably bounding neural network preimages. Advances in Neural Information Processing Systems **36** (2024)
19. Lechner, M., Žikelić, Đ., Chatterjee, K., Henzinger, T.A., Rus, D.: Quantization-aware interval bound propagation for training certifiably robust quantized neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). pp. 14964–14973 (2023). https://doi.org/10.1609/AAAI.V37I12.26747
20. LeCun, Y., Cortes, C.: Mnist handwritten digit database (2010)
21. Li, Z., Ni, B., Zhang, W., Yang, X., Gao, W.: Performance guaranteed network acceleration via high-order residual quantization. In: IEEE International Conference on Computer Vision (ICCV). pp. 2603–2611 (2017). https://doi.org/10.1109/ICCV.2017.282
22. Lin, D.D., Talathi, S.S., Annapureddy, V.S.: Fixed point quantization of deep convolutional networks. In: Proceedings of the 33nd International Conference on Machine Learning (ICML). pp. 2849–2858 (2016)
23. Lin, H., Lou, J., Xiong, L., Shahabi, C.: Integer-arithmetic-only certified robustness for quantized neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR). pp. 7808–7817. IEEE (2021). https://doi.org/10.1109/ICCV48922.2021.00773
24. Lin, J., Gan, C., Han, S.: Defensive quantization: When efficiency meets robustness. In: International Conference on Learning Representations (2018)
25. Liu, J., Xing, Y., Shi, X., Song, F., Xu, Z., Ming, Z.: Abstraction and refinement: Towards scalable and exact verification of neural networks. CoRR **abs/2207.00759** (2022)
26. Ma, H., Qiu, H., Gao, Y., Zhang, Z., Abuadbba, A., Xue, M., Fu, A., Zhang, J., Al-Sarawi, S.F., Abbott, D.: Quantization backdoors to deep learning commercial frameworks. IEEE Transactions on Dependable and Secure Computing (2023). https://doi.org/10.1109/TDSC.2023.3271956
27. Marco, V.S., Taylor, B., Wang, Z., Elkhatib, Y.: Optimizing deep learning inference on embedded systems through adaptive model selection. ACM Trans. Embed. Comput. Syst. **19**(1), 2:1–2:28 (2020). https://doi.org/10.1145/3371154
28. Matoba, K., Fleuret, F.: Exact preimages of neural network aircraft collision avoidance systems. In: Proceedings of the Workshop on Machine Learning for Engineering Modeling, Simulation, and Design. pp. 1–9 (2020)
29. Mirman, M., Gehr, T., Vechev, M.T.: Differentiable abstract interpretation for provably robust neural networks. In: Proceedings of the 35th International Conference on Machine Learning. vol. 80, pp. 3575–3583 (2018)
30. Mohammadinejad, S., Paulsen, B., Deshmukh, J.V., Wang, C.: DiffRNN: Differential verification of recurrent neural networks. In: Proceedings of the 19th In-

ternational Conference on Formal Modeling and Analysis of Timed Systems. pp. 117–134 (2021)
31. Musa, A.A., Hussaini, A., Liao, W., Liang, F., Yu, W.: Deep neural networks for spatial-temporal cyber-physical systems: A survey. Future Internet **15**(6), 199 (2023). https://doi.org/10.3390/FI15060199
32. Nagel, M., Amjad, R.A., Van Baalen, M., Louizos, C., Blankevoort, T.: Up or down? adaptive rounding for post-training quantization. In: Proceedings of the 37th International Conference on Machine Learning (ICML). vol. 119, pp. 7197–7206 (2020)
33. Nagel, M., Fournarakis, M., Amjad, R.A., Bondarenko, Y., van Baalen, M., Blankevoort, T.: A white paper on neural network quantization. arXiv preprint arXiv:2106.08295 (2021)
34. Pan, X., Zhang, M., Yan, Y., Yang, M.: Understanding the threats of trojaned quantized neural network in model supply chains. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC). pp. 634–645 (2021). https://doi.org/10.1145/3485832.3485881
35. Paulsen, B., Wang, J., Wang, C.: Reludiff: Differential verification of deep neural networks. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). pp. 714–726. IEEE (2020)
36. Paulsen, B., Wang, J., Wang, J., Wang, C.: NeuroDiff: scalable differential verification of neural networks using fine-grained approximation. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. pp. 784–796 (2020)
37. Pham, L.H., Sun, J.: Verifying neural networks against backdoor attacks. In: Proceedings of the 34th International Conference on Computer Aided Verification (CAV). pp. 171–192 (2022). https://doi.org/10.1007/978-3-031-13185-1_9
38. Prabhakar, P., Afzal, Z.R.: Abstraction based output range analysis for neural networks. In: Proceedings of the Annual Conference on Neural Information Processing Systems. pp. 15762–15772 (2019)
39. Rokh, B., Azarpeyvand, A., Khanteymoori, A.: A comprehensive survey on model quantization for deep neural networks in image classification. ACM Trans. Intell. Syst. Technol. **14**(6), 97:1–97:50 (2023). https://doi.org/10.1145/3623402
40. Singh, G., Gehr, T., Püschel, M., Vechev, M.T.: An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages (POPL) **3**, 41:1–41:30 (2019). https://doi.org/10.1145/3290354
41. Song, C., Fallon, E., Li, H.: Improving adversarial robustness in weight-quantized neural networks. arXiv preprint arXiv:2012.14965 (2020)
42. Song, X., Sun, Y., Mustafa, M.A., Cordeiro, L.C.: QNNRepair: Quantized neural network repair. In: Proceedings of the 21st International Conference on Software Engineering and Formal Methods. vol. 14323, pp. 320–339 (2023)
43. Tang, Z., Dong, Y., Su, H.: Error-silenced quantization: Bridging robustness and compactness. In: Proceedings of the Workshop on Artificial Intelligence Safety (AISafety@IJCAI) (2020)
44. Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., Cheng, J.: Two-step quantization for low-bit neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4376–4384 (2018). https://doi.org/10.1109/CVPR.2018.00460
45. Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C., Kolter, J.Z.: Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In: Proceedings of the Annual Conference on Neural Information Processing Systems. pp. 29909–29921 (2021)

46. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. CoRR **abs/1708.07747** (2017)
47. Yang, P., Li, R., Li, J., Huang, C., Wang, J., Sun, J., Xue, B., Zhang, L.: Improving neural network verification through spurious region guided refinement. In: Proceedings of 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 389–408 (2021)
48. Zhang, D., Yang, J., Ye, D., Hua, G.: Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In: Proceedings of the European conference on computer vision (ECCV). pp. 365–382 (2018)
49. Zhang, X., Wang, B., Kwiatkowska, M.: On preimage approximation for neural networks. arXiv preprint arXiv:2305.03686 (2023)
50. Zhang, Y., Chen, G., Song, F., Sun, J., Dong, J.S.: Certified quantization strategy synthesis for neural networks. `https://github.com/zhangyedi/Quadapter` (2024)
51. Zhang, Y., Song, F., Sun, J.: Qebverif: Quantization error bound verification of neural networks. In: Proceedings of the 35th International Conference on Computer Aided Verification. vol. 13965, pp. 413–437 (2023). `https://doi.org/10.1007/978-3-031-37703-7_20`
52. Zhang, Y., Zhao, Z., Chen, G., Song, F., Chen, T.: BDD4BNN: A bdd-based quantitative analysis framework for binarized neural networks. In: Proceedings of the 33rd International Conference on Computer Aided Verification (CAV). pp. 175–200 (2021). `https://doi.org/10.1007/978-3-030-81685-8_8`
53. Zhang, Y., Zhao, Z., Chen, G., Song, F., Chen, T.: Precise quantitative analysis of binarized neural networks: A bdd-based approach. ACM Trans. Softw. Eng. Methodol. **32**(3), 62:1–62:51 (2023). `https://doi.org/10.1145/3563212`
54. Zhang, Y., Zhao, Z., Chen, G., Song, F., Zhang, M., Chen, T., Sun, J.: Qvip: An ilp-based formal verification approach for quantized neural networks. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 82:1–82:13 (2022). `https://doi.org/10.1145/3551349.3556916`
55. Zhu, Y., Peng, H., Fu, A., Yang, W., Ma, H., Al-Sarawi, S.F., Abbott, D.: Towards robustness evaluation of backdoor defense on quantized deep learning model. Available at SSRN: `https://ssrn.com/abstract=4578346`