Certifiably Quantisation-Robust Training and Inference of Neural Networks

Hue Dang
Lero, Trinity College Dublin

Matthew Robert Wicker
Imperial College London

Goetz Botterweck
Lero, Trinity College Dublin

Andrea Patane
Lero, Trinity College Dublin

Abstract

We tackle the problem of computing guarantees for the robustness of neural networks against quantisation of their inputs, parameters and activation values. In particular, we pose the problem of bounding the worst-case discrepancy between the original neural network and all possible quantised ones parametrised by a given maximum quantisation diameter $\epsilon > 0$ over a finite dataset. To achieve this, we first reformulate the problem in terms of bilinear optimisation, which can be solved for provable bounds on the robustness guarantee. We then show how a quick scheme based on interval bound propagation can be developed and implemented during training so to allow for the learning of neural networks robust against a continuous family of quantisation techniques. We evaluated our methodology on a variety of architectures on datasets such as MNIST, F-MNIST and CIFAR10. We demonstrate how non-trivial bounds on guaranteed accuracy can be obtained on several architectures and how quantisation robustness can be significantly improved through robust training.

1 INTRODUCTION

Deep learning, particularly in the form of Neural Networks (NNs), has achieved state-of-the-art and even human-level performances in a variety of tasks (Goodfellow et al., 2016). Applications such as self-driving cars (Ni et al., 2020), automatic diagnosis (Vasilakos et al., 2016), space-AI (Rech, 2024) etc. appear today as tantalisingly close to the reach of automated technology. However, despite their great promises, the successful deployment of NNs in industrial applications is still hindered by many practical considerations, including computational efficiency (Chinchali et al., 2018), constrained resources on embedded devices (Chen et al., 2020), sustainability and energy consumptions (Patterson et al., 2021), and the overall fragility of these models to small changes in their parameters or environment (Huang et al., 2018; Kurd et al., 2007; Forsberg et al., 2020).

Quantisation techniques (Paupamah et al., 2020) potentially offer a way out of the conundrum. By carefully quantising the parameters and inputs of a NN (at training or inference time), it is, in fact, possible to reduce their computational footprint while keeping as much of their predictive capacity as possible. Unfortunately though, the behaviour of quantised NNs can vary greatly when compared to their full-precision counter-parts (Hashemi et al., 2017; Ferianc et al., 2021; Guo, 2018; Sung et al., 2015). While techniques for their formal analysis exists (Li et al., 2021; Zhang et al., 2023b,a; Henzinger et al., 2021), those are quantisation-scheme specific and generally require the development of new schemes and the performing of new analyses every time a model needs to be deployed in different settings (Chmiel et al., 2020), and, to the best of our knowledge, a technique for the systematic verification of quantisation schemes is still missing. The development of such a technique would allow not only for the computation of hard guarantees on the worst-case performances of a NN when deployed, but also – when used at training time – the learning of NNs which are certifiably robust across a variety of commonly employed quantisation schemes.

In this paper we develop robustness guarantees on the behaviour of NNs across different quantisation techniques. The resulting guarantees are *worst-case*, in that they aim at computing the largest behavioural change of a NN across infinitely-many quantisation schemes in a given test dataset. Specifically, for a given quantisation diameter $\epsilon > 0$, encoding the maximum

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

change performed by a family of quantisation schemes $\mathscr{Q}$, we pose the problem of computing the maximum discrepancy in the output behaviour of the neural network subjected to any quantisation $\mathcal{Q} \in \mathscr{Q}$. Unfortunately the resulting optimisation problem is NP-hard, involving changes not only on the input space of the neural network (as in the case of adversarial examples (Katz et al., 2017)) but also in the weights, biases and activation values of the model.

Nevertheless, by relying on recent advances in non-convex global optimisation for deep learning (Chiu and Zhang, 2023; Lyu et al., 2020; Wicker et al., 2020) we show how safe bounds on the network performances can be efficiently computed. In particular we extend two optimisation schemes to work under the settings at hand. First we show how the problem of robustness against quantisation in neural networks can be reformulated in terms of a bilinear optimisation problem, which can be efficiently solved using standard techniques implemented by Gurobi (Cheng and Li, 2022). We then show, how Interval Bound Propagation (IBP) techniques can be derived to compute (coarse) robustness bounds in terms only of arithmetic operations. Crucially, this allows us to embed their computation into the SGD algorithm to obtain quantisation-robust (Chmiel et al., 2020) training schemes akin to those used for adversarial learning (Gowal et al., 2018).

We evaluate our method on an array of different architectures trained on the MNIST, F-MNIST and CIFAR10 datasets. We showcase how our verification technique can obtain non-vacuous bounds across different settings and for commonly used quantisation strengths, and find that certifiably quantisation-robust training can dramatically increase the verified accuracy of the resulting NN, upward to +97% when compared to standard training.

In summary, our main contributions are:

- We pose the problem of formal robustness against a family of quantisation schemes and show how bounds can be computed using global optimisation techniques.
- We develop an IBP scheme that can be employed at training time to obtain NNs certifiably-robust against different quantisation techniques. Our method can produce a model that is not only robust to different quantisation techniques but also easier to verify.
- We empirically demonstrate the effectiveness of our method in computing verification bounds on the quantisation robustness across different fully-connected architectures and in increasing NN's quantisation-robustness through training.

2 RELATED WORK

A number of algorithms have been developed for verification of quantised neural networks (Paulsen et al., 2020; Li et al., 2021; Zhang et al., 2023b,a; Henzinger et al., 2021; Cooke et al., 2023; Huang et al., 2024). These however work by assuming that a given quantisation scheme with a given precision is being employed, and generally function by computing the difference between the original neural network and the one specific quantised version of the latter. Therefore a new verification routine needs to be developed, implemented and run every time the NN is quantised with a different technique and precision (Chmiel et al., 2020). Instead, our method works by verifying the behaviour of all possible quantised networks, up to a given maximum quantisation diameter and is transparent to the details of the quantisation algorithm used. That is, we verify an infinite amount of neural networks which are ϵ-close to each other. Crucially this enables one to train neural networks which are quantisation-robust.

While quantisation-aware training techniques exist (Jacob et al., 2018), those either apply to, again, a specific quantisation scheme (Lechner et al., 2023; Song et al., 2020), or do not come with guarantees (Subia-Waud and Dasmahapatra, 2024; Liu et al., 2021; Chmiel et al., 2020). The latter references are the most related to the current work, as they tackle the problem of quantisation-robust training transparent to the details of the quantisation algorithm used. Liu et al. (2021) model quantisation as adversarial noise parameterised by a given ϵ (as we do in Section 3), and adapt sharpness-aware optimisation for the training of NNs which are intrinsically more robust to quantisation. Similarly, Chmiel et al. (2020) modify the NN training loss with a kurtosis regulariser that encourages the learning of less sensitive models. Subia-Waud and Dasmahapatra (2024) instead rely on Bayesian learning to model quantisations in terms of uncertainty in their parameters. While all these works aim at developing NNs robust against a variety of quantisation techniques, they lack verification guarantees.

Another related area is that of sensitivity to weight perturbations in NNs (Weng et al., 2020; Tsai et al., 2021; Cheney et al., 2017; Wu et al., 2020), though quantisation generally affects also the activation values of a neural network, which is not accounted by weight perturbation analysis. An explicit reference to weight quantisation is indeed made by Weng et al. (2020), who propose an algorithm based on the alternating direction method of multipliers to compute a certified region around the weights of a neural network. While their technique is not specific to a given quantisation precision, their method still relies on a given quantisation scheme. Furthermore, it cannot be used to train a quantisation-robust network but only to estimate how robust a given network is. Instead we develop differentiable bounds, transparent on the quantisation scheme, and that crucially can be used at training time.

3 PROBLEM FORMULATION

We focus on feed-forward NNs, which without loss of generality we assume to be fully-connected.[1] Such a NN is a function $\mathscr{N} : \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_{out}}$ defined iteratively over K layers by the following computations:

$$z^{(0)}(x) = x \tag{1}$$

$$h^{(k)}(x) = W^{(k)} z^{(k-1)}(x) + b^{(k)} \tag{2}$$

$$z^{(k)}(x) = \sigma(h^{(k)}(x)), \tag{3}$$

for $k = 1, \ldots, K$. In the set of equations above, $W^{(k)}$ are the weights and $b^{(k)}$ the biases, that we collectively identify using the parameter w and call the parameters or weights of the NN. σ is the activation function, and $\mathscr{N}(x) = z^{(K)}(x)$ is the final output of the NN – which corresponds to its prediction in the case of regression and to a vector of class probabilities in the case of classification.

The above equations are implicitly given in infinite precision arithmetic. Quantisation affects all the individual steps by rounding all the intermediate computations of the neural network to its corresponding quantised value. In particular, a general quantisation function can be modelled with a function $\mathcal{Q} : \mathbb{R} \to \mathcal{D}$ such that $\mathcal{D} \subset \mathbb{R}$ is a finite set, and for each $t \in \mathbb{R}$, $\mathcal{Q}(t) = q$ represents its quantised value.

Definition 1 (Quantisation diameter) *Given a quantisation function $\mathcal{Q}$ we call diameter of the quantisation (ϵ) over a bounded support $T \subset \mathbb{R}$ the maximum loss of precision produced by the quantisation, i.e.:*

$$\epsilon = diam(\mathcal{Q}) = \max_{x \in T} |\mathcal{Q}(x) - x|.$$

Notice that, in the case of quantisation for NNs, the quantisation function $\mathcal{Q}$ does not need to be the same for each layer. It is, in fact, customary in the literature to use different precisions for different layers and parameters, typically for weights, biases, activations, and inputs. Practically, this would result in having different quantisation diameters for the different quantisation schemes and precisions employed in the network.

For simplicity of presentation, we here assume the existence of a function $\mathcal{Q}_w$ for quantising all the weights, $\mathcal{Q}_a$ for the activation, and $\mathcal{Q}_{in}$ for the input. However, the results presented in the paper can be generalised in a straightforward manner. With an abuse of notation, we use $\mathcal{Q} = [\mathcal{Q}_w, \mathcal{Q}_a, \mathcal{Q}_{in}]$ to refer to the overall quantisation strategy employed for a given NN, and $\mathcal{Q}(\mathscr{N})$ to refer to the quantised NN.

In this work, we focus on two complementary problems. In the first one, we aim at providing guarantees for a trained NN on the maximum discrepancy between its predictions and its quantised versions up to a given diameter.

Problem 1 (Quantisation Robustness) *Consider a neural network $\mathscr{N}$, and three non-negative quantisation diameters ϵ_w, ϵ_a, and ϵ_{in}. Define: $\mathscr{Q} = \{\mathcal{Q} = [\mathcal{Q}_w, \mathcal{Q}_a, \mathcal{Q}_{in}] \,|\, diam(\mathcal{Q}_w) \leq \epsilon_w,\ diam(\mathcal{Q}_a) \leq \epsilon_a,\ diam(\mathcal{Q}_{in}) \leq \epsilon_{in}\}$, that is, the set of all viable quantisations of $\mathscr{N}$ with the upper diameter bounded by the given thresholds. Given $x \in T$, with $T \subset \mathbb{R}^{n_{in}}$ the input space of the NN, find:*

$$\hat{q} = \max_{\mathcal{Q} \in \mathscr{Q}} |\mathscr{N}(x) - \mathcal{Q}(\mathscr{N})(x)|, \tag{4}$$

that is the maximum discrepancy possible between the neural network $\mathscr{N}$ computed in x and all the possible neural networks resulting from the quantisations included in $\mathscr{Q}$.

The maximisation problem above is non-convex, global and over an infinite set of input points and neural networks. While it cannot be solved exactly, here we will develop techniques for computing an upper bound on it.We remark that a solution to Problem 1 enables one to compute verification guarantees against a desired maximum variation in the behaviour of the quantised network, similar to those employed for adversarial robustness (Meng et al., 2022). In particular, in the case of classification, a guarantee of the resulting accuracy can be computed by comparing the minimum for the class confidence corresponding to the ground truth to the maximum for that of all remaining classes.

The second problem we tackle instead, concerns training a NN such that it is robust against quantisations. Similarly to methods from adversarial learning (Gowal et al., 2018), we modify the loss function to account for quantisations up to a given strength. Below we give the formulation in the case of classification loss.

Problem 2 (Quantisation-Robust Training) *Consider an NN $\mathscr{N}$, a training set D, and maximum diameters ϵ_w, ϵ_a, and ϵ_{in}. Let $\lambda \in [0, 1]$ be a constant, $x \in T$ an input point with $T \subset \mathbb{R}^{n_{in}}$ the input space of the NN and y its class label. Redefine the NN's loss function as:*

$$L_{qr}(\mathscr{N}(x), y, \mathscr{Q}) = \lambda L(\mathscr{N}(x), y) + (1 - \lambda) L(\hat{z}_{\mathscr{Q}}, y)$$

[1]Convolutional layers can, in fact, be reformulated as fully-connected layers (Boopathy et al., 2019).

where $\hat{z}_{\mathscr{Q}}$ is defined as:

$$\hat{z}_{\mathscr{Q},j} = \begin{cases} \min_{\mathcal{Q}\in\mathscr{Q}} \mathcal{Q}(\mathscr{N})_j(x) & \textit{if } j = y \\ \max_{\mathcal{Q}\in\mathscr{Q}} \mathcal{Q}(\mathscr{N})_j(x) & \textit{otherwise} \end{cases}.$$

Find:

$$w^{qr} = \arg\min_{w} \sum_{i=1}^{|D|} L_{qr}(\mathscr{N}(x_i), y_i, \mathscr{Q}))).$$

Here, $|D|$ denotes the cardinality of the dataset, $\mathcal{Q}(\mathscr{N})_j(x)$ for $j = 1, \ldots, n_{out}$ represent the components of the output vector $\mathcal{Q}(\mathscr{N})(x)$. The expressions $\max_{\mathcal{Q}\in\mathscr{Q}} \mathcal{Q}(\mathscr{N})j(x)$ and $\min_{\mathcal{Q}\in\mathscr{Q}} \mathcal{Q}(\mathscr{N})_j(x)$ denote the maximum and minimum values, respectively, of each output component over the set of quantisations $\mathscr{Q}$. Notice that the optimisation problem over the loss function is highly non-linear and high-dimensional. Nevertheless, we show how a differentiable upper bound to the quantised-robust loss function L_{qr} can be found in an efficient way, so that standard tools for backpropagation and gradient descent can be used for performing training of the neural network with only a marginal overhead.

4 METHODOLOGY

The computation of robustness guarantees against quantisation at training and inference time requires the solution of a non-convex optimisation problem on the inputs, parameters and activations of the neural network. In this section, we show how safe bounds on the optimal values can be computed. We will then demonstrate how they can be implemented in a straightforward manner inside the gradient descent computations at training time.

4.1 Bilinear Optimisation for Quantisation Robustness

Solving Problem 1 boils down to the solution of two optimisation problems for each of the output dimensions of the NN under consideration. That is we aim at solving $\max_{\mathcal{Q}\in\mathscr{Q}} \mathcal{Q}(\mathscr{N})_j(x)$ and $\min_{\mathcal{Q}\in\mathscr{Q}} \mathcal{Q}(\mathscr{N})_j(x)$ for every $j = 1, \ldots, n_{out}$. This, unfortunately, is in general a highly non-linear optimisation problem. However, we show how it can be encoded into a bilinear programme using standard non-convex relaxation techniques. Crucially, bilinear programmes can be solved efficiently and with provable guarantees using standard tools such as Gurobi.

The main relaxation needed is the linearisation of the activation function of Equation (3). This can be performed using standard techniques developed in the field of verification of neural networks (Lyu et al., 2020; Wicker et al., 2020), and results in a set of lower and upper bounds for each activation function in the architecture:[2]

Definition 2 (Activation Bound) *For a given layer $k \in \{1, \ldots, K-1\}$ and neuron $i \in \{1, \ldots, n_k\}$, we denote with $\alpha^L_{k,i}, \beta^L_{k,i}, \alpha^U_{k,i}$ and $\beta^U_{k,i}$ the coefficients associated to linear lower and upper bounds computed on Equation (3), i.e. such that:*

$$\alpha^L_{k,i} h^{(k)}_i + \beta^L_{k,i} \leq z^{(k)}_i \leq \alpha^U_{k,i} h^{(k)}_i + \beta^U_{k,i}$$

Notice that in the specific case of ReLU the encoding can be done exactly without resorting to linear bounds. Specifically, given upper and lower bound on the pre-activation $h^{(k)}_i$, which we denote as $h^{(k),L}_i$ and $h^{(k),U}_i$, then one can encode ReLU exactly using the following set of linear and integer constraints (Tjeng et al., 2017):

$$\begin{aligned} z^{(k)}_i = \mathrm{ReLU}(h^{(k)}_i) \iff & (z^{(k)}_i \leq h^{(k)}_i - h^{(k),L}_i(1-a)) \\ & \wedge (z^{(k)}_i \geq h^{(k)}_i) \\ & \wedge (z^{(k)}_i \leq h^{(k),U}_i \cdot a) \\ & \wedge (z^{(k)}_i \geq 0) \wedge (a \in \{0,1\}), \end{aligned}$$

where a is an auxiliary binary variable.

We can now formulate the resulting full bilinear programme for the computation of quantisation robustness. Let ζ, ω, and ξ denote auxiliary variables for the quantised activation values, weights and inputs. Consider a compact set $T \subset \mathbb{R}^{n_{in}+n_w+n_a}$ bounding all possible quantised values of inputs, weights and activation, where n_a is the overall number of neurons in the NN. Then the resulting bilinear formulation (minimisation case) is:

$$\begin{aligned} \min \quad & h^{(K)}_\iota \qquad (5) \\ \text{subject to:} \quad & \text{for } k = 1, \ldots, K,\ i = 1, \ldots n_k : \\ & h^{(k)}_i = \sum_{j=1}^{n_{k-1}} \omega^{(k)}_{ij} \zeta^{(k-1)}_j + b^{(k)}_i \\ & \alpha^L_{k,i} h^{(k)}_i + \beta^L_{k,i} \leq z^{(k)}_i \leq \alpha^U_{k,i} h^{(k)}_i + \beta^U_{k,i} \\ & z^{(k)}_i - \epsilon_a \leq \zeta^{(k)}_i \leq z^{(k)}_i + \epsilon_a \\ & w^{(k)}_{ij} - \epsilon_w \leq \omega^{(k)}_{ij} \leq w^{(k)}_{ij} + \epsilon_w \\ & x - \epsilon_{in} \leq \xi \leq x + \epsilon_{in}, \quad \xi = \zeta^{(0)} \\ & (\xi, \zeta^{(k)}, \omega^{(k)}) \in T. \end{aligned}$$

Notice that, albeit similar in form, the above optimisation problem is significantly different than those con-

[2]Under the common assumption that the activation function has only a finite number of discontinuities.

structed in the context of adversarial robustness. Crucially, it is not only the inputs that are variable, but also the activation values and weights in each layer. Specifically, the constants in the Equation 5 are ϵ_w, ϵ_a, ϵ_{in}, x, the α coefficients, the β coefficients and the $w_i j^{(k)}$. The rest are all variables to the optimisation problem. This implies not only a significantly higher number of variables, but most importantly it turns the constraints of the problem – even in its relaxed form – into non-linear constraints. Fortunately, the introduction of auxiliary variables for the activation functions at each layer, disentangles their explicit dependence on the weights of previous layers, so that the resulting constraints are bilinear in form. Therefore, the resulting optimisation problem is a bilinear programme, for which standard optimisation techniques can be used.

Remark 1 *While the solution of the bilinear programme can be performed using provable bounds (i.e., upper and lower bounds on the optimum value) and converges to the true solution of the optimisation problem, because of the relaxation of the activation function of Definition 2 the final result will be a lower bound to the true minimum. Therefore our method is sound but in general not complete (except for the special case of ReLU networks). A complete method can be developed by using piece-wise linear approximations of the activation functions employed, and refining them iteratively until a desired error tolerance is reached (Benussi et al., 2022).*

Solving Problem 1, requires one to iteratively compute maximum and minimum of $h_\iota^{(K)}$ for $\iota = 1, \dots, n_{out}$. This can then be used to obtain an upper bound on Problem 1, by computing the worst-case softmax resulting from the logit bounds. To summarise this section's result, we obtain the following.

Proposition 1 *Consider a NN architecture of Equations* (1)–(3)*, a family of quantisation schemes $\mathscr{Q}$ with maximum diameters $\epsilon_{in}, \epsilon_w$ and ϵ_a, consider an input point x and a compact region T over the input-weight-activation space. For $\iota = 1, \dots, n_{out}$, let $h_\iota^{(K),L}$ and $h_\iota^{(K),U}$ be the solution to the minimisation problem in Equation* (5) *and its corresponding maximisation problem. Define the vector $\hat{q}^U \in \mathbb{R}^{n_{out}}$ as the vector with entries $\hat{q}_\iota^U = \max\{\mathscr{N}_\iota(x) - h^{(K),L}, h^{(K),U} - \mathscr{N}_\iota(x)\}$. Then:*

$$\hat{q} \leq \hat{q}^U,$$

i.e., $\hat{q}^U$ upper-bounds the solution of Problem 1.

4.2 Bound Propagation for Quantisation-Robust Training

Albeit tight, the bound resulting from solving the bilinear programme in Section (4.1) is computationally expensive and non-differentiable. Both of these factors make it difficult to incorporate into training. Thus, solving Problem 2 requires us to develop an alternative method to regularise training. Methods from sharpness-aware minimization may be considered, however, they do not provide worst-case guarantees and therefore, may not directly improve the model's robustness w.r.t. Problem 1. Instead, we consider methods similar to those investigated in adversarial training, namely, interval bound propagation.

In the adversarial robustness literature, interval bound propagation (IBP), proceeds by defining an interval around inputs, i.e., $[x-\epsilon, x+\epsilon]$ for some fixed $\epsilon > 0$ and then defines a propagation procedure such that one can pass the interval through the neural network architecture to arrive at an interval over the output space: $[y^L, y^U]$ such that $\forall x' \in [x-\epsilon, x+\epsilon], \mathscr{N}(x) \in [y^L, y^U]$. Though interval bound propagation is sound, differentiable, and efficient this sort of procedure cannot directly be applied to either Problem 1 nor Problem 2 as only input bounds are propagated. Recent developments in IBP have also developed tools for propagating both interval and weight intervals (Wicker et al., 2021, 2023) however, these do not consider the quantisation of activation functions and therefore would not soundly be able to solve Problem 1.

For this purpose, we derive a straightforward extension of the standard IBP formulation so to jointly propagate the entire quantisation radius through Equations (1)–(3) with intervals $[x - \epsilon_{in}, x + \epsilon_{in}], [z - \epsilon_a, z + \epsilon_a]$ and $[w-\epsilon_w, w+\epsilon_w]$. To do so, we leverage standard results from bound propagation and abstract interpretation (Rump, 1999; Mirman et al., 2018). We state our quantisation interval bound propagation below, which is proved in Section C of the Supplementary Material.

Proposition 2 *Consider a NN architecture as defined in Equations* (1)–(3)*, a family of quantisation schemes $\mathscr{Q}$ with maximum diameters $\epsilon_{in}, \epsilon_w$ and ϵ_a, consider an input point x and denoting the value of ϵ_w corresponding to each weight and bias with W^{k,ϵ_w} and b^{k,ϵ_w} respectively. We can propagate the quantisation diameters through the network:*

$$z^{(0),\mu}(x) = z^{(0)}(x) \tag{6}$$

$$z^{(0),r}(x) = \epsilon_{in} \tag{7}$$

$$h^{(k),\mu}(x) = W^{(k)} z^{(k-1),\mu}(x) + b^{(k)} \tag{8}$$

$$\begin{aligned} h^{(k),r}(x) = {} & |W^{(k)}| z^{(k-1),r}(x) + W^{(k),\epsilon_w} |z^{(k-1),\mu}(x)| \\ & + W^{(k),\epsilon_w} z^{(k-1),r}(x) + b^{(k),\epsilon_w} \end{aligned} \tag{9}$$

$$z^{(k),L}(x) = \sigma(h^{(k),\mu}(x) - h^{(k),r}(x)) - \epsilon_a \quad (10)$$
$$z^{(k),U}(x) = \sigma(h^{(k),\mu}(x) + h^{(k),r}(x)) + \epsilon_a \quad (11)$$
$$z^{(k),\mu}(x) = \left(z^{(k),L}(x) + z^{(k),U}(x)\right)/2 \quad (12)$$
$$z^{(k),r}(x) = \left(z^{(k),U}(x) - z^{(k),L}(x)\right)/2 \quad (13)$$

such that $\forall \mathcal{Q} \in \mathscr{Q}, \mathcal{Q}(\mathcal{N}(x)) \in [z^{(K),L}(x), z^{(K),U}(x)]$.

We can observe that the computations proposed in Proposition 2 not only satisfy the desired soundness property i.e., $\forall \mathcal{Q} \in \mathscr{Q}, \mathcal{Q}(\mathcal{N}(x)) \in [z^{(K),L}(x), z^{(K),U}(x)]$, but additionally, that each operation used (addition, subtraction, integer division, matrix multiplication, and application of the activation function) is easily differentiable. The computational complexity of computing these bounds is roughly four times the complexity of a standard forward pass with respect to the network making. Thus, using Proposition 2 we are able to employ gradient descent to approximately solve Problem 2 at only $4\times$ the computational cost. Notice that Proposition 2 can also be used to provide a quick (albeit looser) solution to Problem 1, which we also do in the experiments in Section 5.

The solution in Proposition 2 is not without its limitations, however. The bound on the matrix multiplication used in Equation 9 is known to introduce an over-approximation factor of about $1.5\times$ (Rump, 1999). When used in the context of propagation through neural network architectures, this over-approximation means that the bounds grow loosely super-linearly with an increase in the depth of the neural network (Wicker et al., 2020, 2024). This approximation can be limited by leveraging linear approximations of activation functions Sosnin et al. (2024) though at a larger computational cost.

5 RESULTS

We empirically investigate the suitability of the Bilinear Programme and IBP formulations we derived to solve Problem 1 and 2. For this purpose, we design a set of experiments looking at how the bounding techniques and the maximum discrepancy ($\hat{q}$ in Problem 1) scale with the width and the depth of the neural network (Section 5.1). We then look at the effect of quantisation-robust training (i.e., Problem 2) on guaranteed accuracy, performing comparisons on an array of $\epsilon_{in}, \epsilon_w$ and ϵ_a (Section 5.2). Additionally, we look at the specific results of verified accuracy across three different quantisation diameters applied to the NN: 6-bit, 8-bit and 10-bit quantisation (Section 5.3). Finally, we conduct an empirical analysis of the effects of quantisation on our IBP-trained models, focusing on worst-case discrepancy and model performance. Specifically, we compute the worst-case discrepancy across various quantisation precisions in the F-MNIST dataset and compare the quantised accuracy in the CIFAR10 dataset of our training method with the quantisation-aware training techniques in Chmiel et al. (2020) and Esser et al. (2019) (Section 5.4).

All the NNs have been training on the MNIST LeCun et al. (2010), F-MNIST Xiao et al. (2017) and CIFAR10 Krizhevsky et al. (2009) datasets using PyTorch. All the experiments have been carried out using a laptop with an Intel Ultra7-155H-1.40GHz, 32GB RAM, NVIDIA GeForce RTX 4070 Laptop GPU. [3]

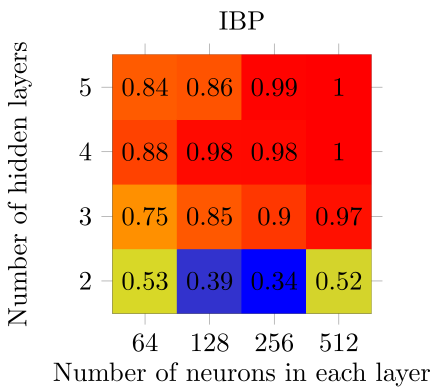
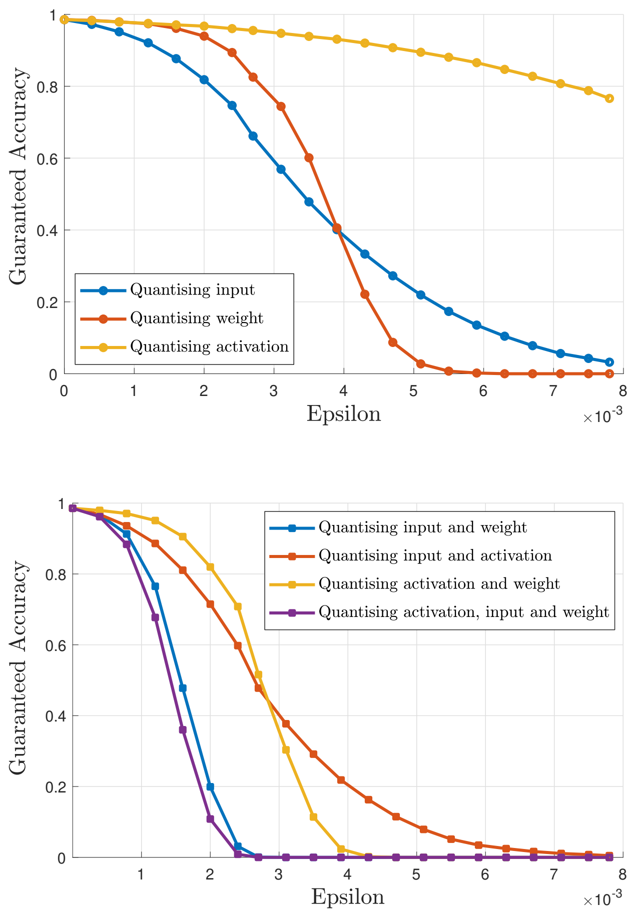


Figure 1: The worst-case discrepancy between the original neural network and all possible networks within the perturbation region computed by IBP for varying numbers of hidden layers and neurons.

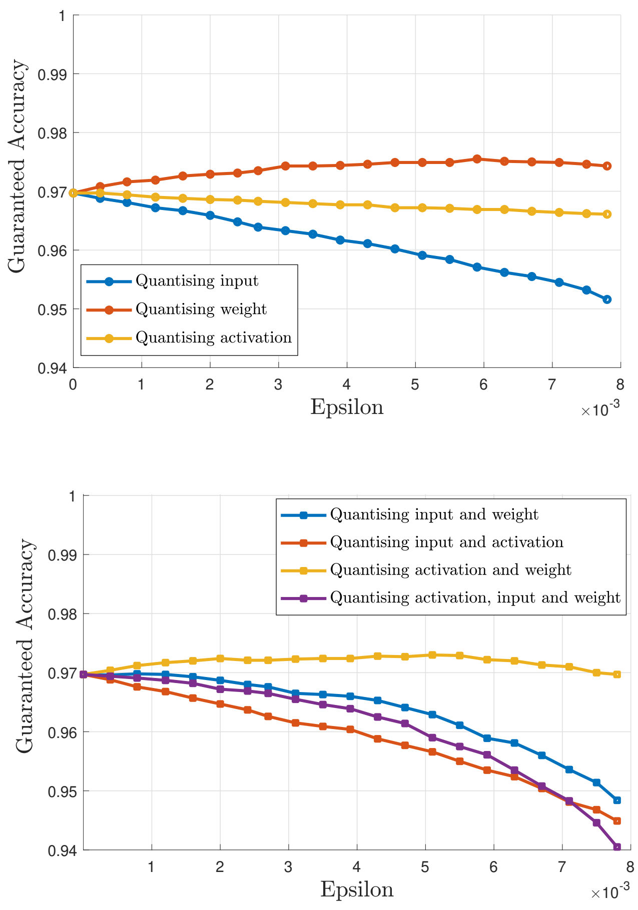
5.1 Verification of Quantisation Robustness

We bound the maximum discrepancy between the original and the worst-case quantised neural network on a range of fully-connected models trained with 2, 3, 4 and 5 hidden layers, and with 64, 128, 256 and 512 hidden neurons per layer. We evaluate the worst-case discrepancy by computing the maximum of $\hat{q}$ over 100 test images from the MNIST dataset. The results for this analysis in the case of IBP bounds are given in the heatmap in Figure 1, using maximum quantisation diameter $\epsilon_{in} = \epsilon_w = \epsilon_a = 1/1023$, roughly corresponding to a generic 10-bit quantisation scheme. Notice that in all situations, except from the case of 512 hidden neurons per layer, with 4 and 5 hidden layers (corresponding to $\approx$ 1.45M network parameters, 784 inputs and 2560 activations to be bounded), IBP is able to obtain non-vacuous (i.e. < 1) bounds. We also observe the general behaviour one would expect, with

[3]The source code used for the experiments can be found at *https://github.com/danghue18/Certifiably-Quantisation-Robust-training-and-inference.git*



(a) Normal training



(b) Robust training

Figure 2: Guaranteed accuracy computed by IBP method of the model (a) trained normally and (b) trained robustly (i.e., solving Problem 2).

the worst-case $\hat{q}$ increasing in value as the number of neurons and the number of layers increases (with one single exception in the bottom left corner, likely due to different sensitivity of the NN obtained). Interestingly we observe that, albeit having more of an impact on the overall number of parameters, increasing the width of the NN has less impact on the final bound than the number of layers. This is probably due, on the one hand, to the increased looseness of IBP as more propagation is needed, and likely to the fact that an increase in the number of layers magnifies the effect of small differences in the quantisations of early layers.

We further compare the results obtained by IBP with those from the Bilinear Program run until convergence, on architectures with 2 hidden layers containing 64 and 128 neurons per layer. For the architecture with two hidden layers and 64 neurons per layer, the upper bound is 0.30 when computed using the bilinear programming method and 0.53 when using the IBP method. The upper bounds for the architecture with 128 neurons per layer are 0.29 (bilinear programming) and 0.39 (IBP). The average time to obtain an IBP bound for each sample in the MNIST test set is 0.005s, while the significantly higher result for bilinear programming is 60 minutes in a 2-hidden layer MLP.

5.2 Quantisation-Robust Training

We evaluate how quantisation-robust training affects the verified accuracy of an NN, across different ranges of quantisations. To do so, we train a neural network with 5 hidden layers and 256 neurons per layer on the MNIST dataset, and separately analyse the effect of quantisation on inputs, weights, activations, their pairwise combinations, and the full quantisation of the network, for ϵ ranging from 0 (full precision model) to 2/255 (corresponding to 8-bit quantisation).

The results for this analysis are plotted in Figure 2, contrasting the results for normal training (left plots) and quantisation-robust training (right plots). Notice that because of the significantly different range in guaranteed accuracy obtained, the y-axis differs between the plots ($[0, 1]$ vs. $[0.94, 1]$). Notice how robust training significantly impacts the verified accuracy, with the performances of normal training quickly decreasing as ϵ increases, while robust training obtains stable robust accuracy results (always greater than 0.94). In some situations, for example, when pairwise quantisation effects are analysed, normal training quickly reaches 0% verification accuracy, while with robust training, the guaranteed accuracy decreases only by no more than 3%.

Interestingly the analysis also gives insight into the relative impact of quantisation of different aspects of the NN on the overall output behaviour. While the difference is not remarkable in robust training, in the context of normal training, both the input and the weight have a significantly higher impact than the quantisation of activation functions. Intuitively, this is because the quantisation of the inputs has a downstream effect on all the successive computations of the network, and the weights are by far the ones introducing the highest quantity of approximations in terms of the number of parameters involved.

5.3 Verified accuracy

We compute the bounds of the verified accuracy of IBP-trained models obtained by the verification methods of IBP and bilinear programming. These verified accuracies are shown in Figure 3, computed with the quantisation diameters equivalent to 6-bit, 8-bit, and 10-bit weight quantisation (quantising weights only) on a 3-hidden-layer model with 256 neurons on each layer. Note that in Figures 3 and 4, we use the labels 6-bit, 8-bit, and 10-bit to represent all networks parameterised by these quantisation diameters. Interestingly, while the difference in the bound quality between bilinear programming and IBP is marked before without robust training, the latter has the effect of making the IBP bounds much tighter.

We then compare the behaviour of robust training and normal training (standard training without robustness considerations) across different weight quantisation schemes. We perform this analysis by having bilinear programming run until convergence on a ReLU network (3 hidden layers with 256 neurons on each layer) so the verified accuracies obtained are exact. The result of this analysis is given in Figure 4, and shows how our quantisation-robust training increases the robustness across different quantisation strengths with only one training necessary. Notably, the average verification time per image in the MNIST test set is 13 seconds for the robustly trained model, whereas the normal model requires 68 seconds per image. Additional comparisons of verified accuracy and runtime across different quantisation schemes are presented in Table 2 of the Supplementary Material. Furthermore, we compare the verified accuracy of robust and normal models using the IBP verification method on other datasets, including CIFAR-10, MNIST, Fashion-MNIST, and SVHN (Table 1 of the Supplementary Material), showing that our IBP training method can obtain models that are not only robust but also quicker and more effective to verify.

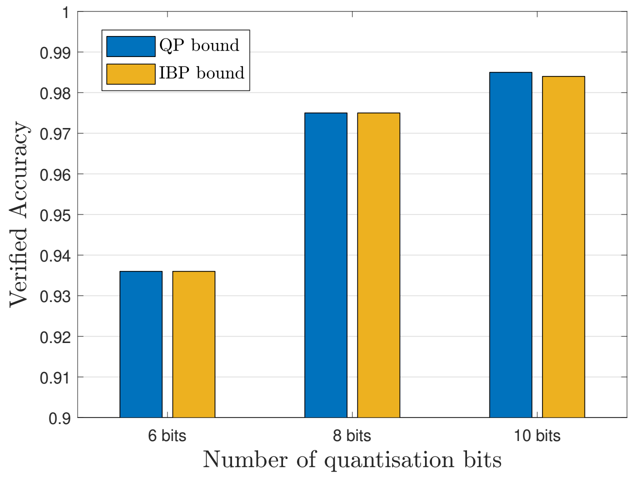


Figure 3: Verified accuracy computed on models trained via solution of Problem 2 with IBP and bilinear programming.

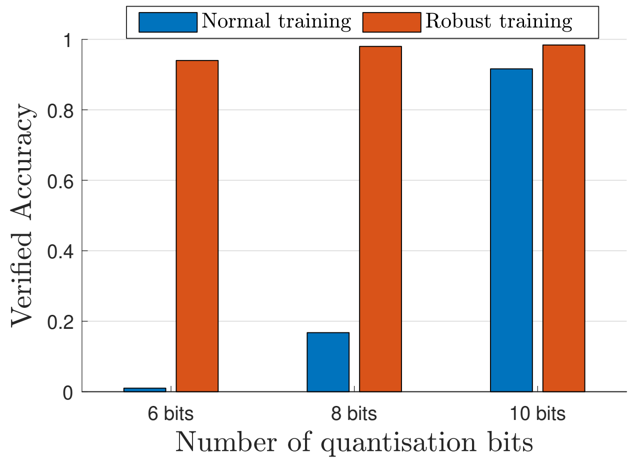


Figure 4: Comparison of normal training and robust training on verified accuracy computed by bilinear optimisation.

5.4 Analysis on Different Quantisation Strengths

We compare our IBP-trained model with the QAT-trained model obtained using the Learned Step Size Quantisation (LSQ) technique Esser et al. (2019), and the QAT-KURE-trained model, which follows the approach in Chmiel et al. (2020), combining LSQ with a kurtosis regularization term to improve robustness to quantisation. Our IBP-trained model achieved a nominal accuracy of 56.13% on CIFAR10 test dataset, while the QAT-trained and QAT-KURE-trained models, specifically designed for 8-bit quantisation, achieved 55.14% and 56.12%, respectively. We applied a symmetric quantisation scheme with 4-bit inputs and 4-bit activations, evaluating quantisation accuracy while varying weight bit widths from 2 to 16 bits. The results in Figure 5 demonstrate that our IBP training method more effectively preserves model performance across different quantisation schemes and precisions than methods tailored for a specific scheme.

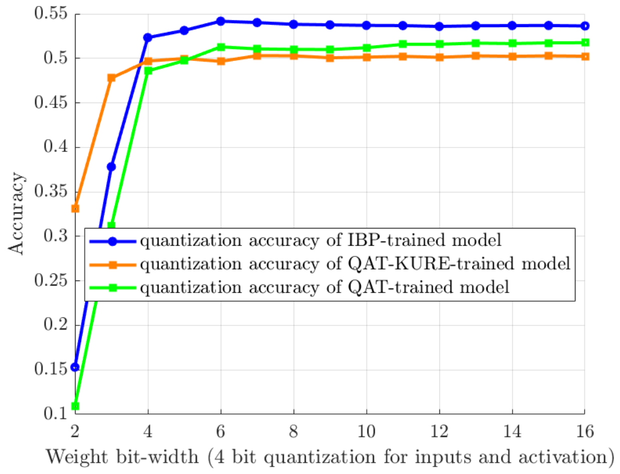


Figure 5: Quantisation accuracy across varying weight bit-widths on CIFAR10.

Finally, in Figure 6, we analyse the impact of quantisation-robust training on the empirical worst-case discrepancy observed (i.e., computed not through verification but by testing the quantisation scheme on the full dataset and reporting the observed worst-case) on a NN trained on the F-MNIST dataset for a varying number of quantisation bits and comparing normal training with robust training using different ϵ.

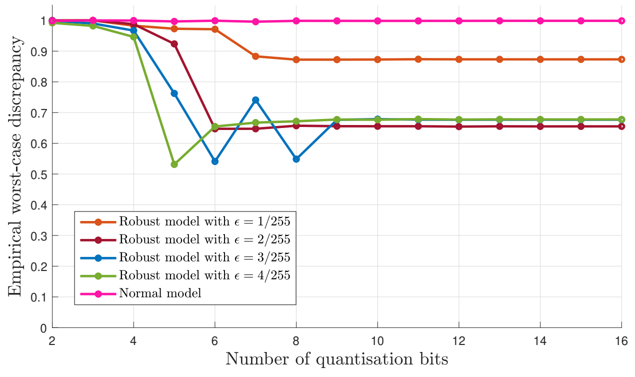


Figure 6: Empirical worst-case discrepancy on F-MNIST.

5.5 Discussion

Verifying an infinite family of possible quantisations requires treating all weights and activations of the neural network as variables, significantly increasing the optimisation problem's complexity. The scalability of the bilinear optimisation method is inherently constrained by the number of variables and constraints, which grow with network size and depth. Hence, applying this method until convergence becomes challenging for larger networks and more complex datasets. Although we have shown that IBP-guided training tightens the bounds and improves verification efficiency, IBP bounds tend to become looser as the number of network layers increases, potentially affecting the convergence of training. Exploring methods to combine the strengths of empirical testing and formal verification is a promising direction for improving the scalability of both verification and training methods.

6 CONCLUSIONS

We derived a verification scheme based on bilinear programming for robustness against a range of quantisation, and further demonstrated how IBP can be leveraged to devise a certifiably quantisation-robust training scheme. In experimental analyses, we have shown the effectiveness of our method across different fully-connected architecture on the MNIST, F-MNIST and CIFAR10 datasets. Future work will look into extending the verification framework from the analysis of quantisation to a complete analysis of all soft-errors and numerical cancellations that can affect a NN at deployment time.

Acknowledgements

We special thanks anonymous reviewers for their valuable comments to improve our work. We would like to thank Doan Ngoc Phu and Dinh Viet Cuong for their help with the experiments. This publication has emanated from research jointly funded by EU Horizon Europe 2021–2027 framework programme, Marie Skłodowska-Curie Actions, Grant Agreement No. 101072456 and Taighde Éireann – Research Ireland under grant number 13/RC/2094_2 (Lero).

References

Benussi, E., Patane, A., Wicker, M., Laurenti, L., and Kwiatkowska, M. (2022). Individual fairness guarantees for neural networks. In *31st International Joint Conference on Artificial Intelligence, IJCAI 2022*, pages 651–658. International Joint Conferences on Artificial Intelligence (IJCAI).

Boopathy, A., Weng, T.-W., Chen, P.-Y., Liu, S., and Daniel, L. (2019). Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3240–3247.

Chen, Y., Zheng, B., Zhang, Z., Wang, Q., Shen, C., and Zhang, Q. (2020). Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions. *ACM Computing Surveys (CSUR)*, 53(4):1–37.

Cheney, N., Schrimpf, M., and Kreiman, G. (2017). On the robustness of convolutional neural networks to internal architecture and weight perturbations. *arXiv preprint arXiv:1703.08245*.

Cheng, X. and Li, X. (2022). Discretization and global optimization for mixed integer bilinear programming. *Journal of Global Optimization*, 84(4):843–867.

Chinchali, S. P., Cidon, E., Pergament, E., Chu, T., and Katti, S. (2018). Neural networks meet physical networks: Distributed inference between edge devices and the cloud. In *Proceedings of the 17th ACM workshop on hot topics in networks*, pages 50–56.

Chiu, H.-M. and Zhang, R. Y. (2023). Tight certification of adversarially trained neural networks via nonconvex low-rank semidefinite relaxations. In *International Conference on Machine Learning*, pages 5631–5660. PMLR.

Chmiel, B., Banner, R., Shomron, G., Nahshan, Y., Bronstein, A., Weiser, U., et al. (2020). Robust quantization: One model to rule them all. *Advances in neural information processing systems*, 33:5308–5317.

Cooke, W., Mo, Z., and Xiang, W. (2023). Guaranteed quantization error computation for neural network model compression. In *2023 IEEE International Conference on Industrial Technology (ICIT)*, pages 1–4. IEEE.

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. (2019). Learned step size quantization. *arXiv preprint arXiv:1902.08153*.

Ferianc, M., Maji, P., Mattina, M., and Rodrigues, M. (2021). On the effects of quantisation on model uncertainty in bayesian neural networks. In *Uncertainty in Artificial Intelligence*, pages 929–938. PMLR.

Forsberg, H., Lindén, J., Hjorth, J., Månefjord, T., and Daneshtalab, M. (2020). Challenges in using neural networks in safety-critical applications. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–7. IEEE.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT Press.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. (2018). On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*.

Guo, Y. (2018). A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*.

Hashemi, S., Anthony, N., Tann, H., Bahar, R. I., and Reda, S. (2017). Understanding the impact of precision quantization on the accuracy and energy of neural networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1474–1479. IEEE.

Henzinger, T. A., Lechner, M., and Žikelić, D. (2021). Scalable verification of quantized neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3787–3795.

Huang, P., Wu, H., Yang, Y., Daukantas, I., Wu, M., Zhang, Y., and Barrett, C. (2024). Towards efficient verification of quantized neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21152–21160.

Huang, X., Kroening, D., Kwiatkowska, M., Ruan, W., Sun, Y., Thamo, E., Wu, M., and Yi, X. (2018). Safety and trustworthiness of deep neural networks: A survey. *arXiv preprint arXiv:1812.08342*, page 151.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D.

(2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 97–117. Springer.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kurd, Z., Kelly, T., and Austin, J. (2007). Developing artificial neural networks for safety critical systems. *Neural Computing and Applications*, 16:11–19.

Lechner, M., Žikelić, D., Chatterjee, K., Henzinger, T. A., and Rus, D. (2023). Quantization-aware interval bound propagation for training certifiably robust quantized neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14964–14973.

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. att labs.

Li, J., Drummond, R., and Duncan, S. R. (2021). Robust error bounds for quantised and pruned neural networks. In *Learning for Dynamics and Control*, pages 361–372. PMLR.

Liu, J., Cai, J., and Zhuang, B. (2021). Sharpness-aware quantization for deep neural networks. *arXiv preprint arXiv:2111.12273.*

Lyu, Z., Ko, C.-Y., Kong, Z., Wong, N., Lin, D., and Daniel, L. (2020). Fastened crown: Tightened neural network robustness certificates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5037–5044.

Meng, M. H., Bai, G., Teo, S. G., Hou, Z., Xiao, Y., Lin, Y., and Dong, J. S. (2022). Adversarial robustness of deep neural networks: A survey from a formal verification perspective. *IEEE Transactions on Dependable and Secure Computing.*

Mirman, M., Gehr, T., and Vechev, M. (2018). Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pages 3578–3586. PMLR.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. (2018). The street view house numbers (svhn) dataset. *Technical Report.*

Ni, J., Chen, Y., Chen, Y., Zhu, J., Ali, D., and Cao, W. (2020). A survey on theories and applications for self-driving cars based on deep learning methods. *Applied Sciences*, 10(8):2749.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., and Dean, J. (2021). Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350.*

Paulsen, B., Wang, J., and Wang, C. (2020). Reludiff: Differential verification of deep neural networks. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 714–726.

Paupamah, K., James, S., and Klein, R. (2020). Quantisation and pruning for neural network compression and regularisation. In *2020 International SAUPEC/RobMech/PRASA Conference*, pages 1–6. IEEE.

Rech, P. (2024). Artificial neural networks for space and safety-critical applications: Reliability issues and potential solutions. *IEEE Transactions on Nuclear Science.*

Rump, S. M. (1999). Fast and parallel interval arithmetic. *BIT Numerical Mathematics*, 39:534–554.

Song, C., Fallon, E., and Li, H. (2020). Improving adversarial robustness in weight-quantized neural networks. *arXiv preprint arXiv:2012.14965.*

Sosnin, P., Müller, M. N., Baader, M., Tsay, C., and Wicker, M. (2024). Certified robustness to data poisoning in gradient-based training. *arXiv preprint arXiv:2406.05670.*

Subia-Waud, C. and Dasmahapatra, S. (2024). Probabilistic weight fixing: Large-scale training of neural network weight uncertainties for quantisation. *Advances in Neural Information Processing Systems*, 36.

Sung, W., Shin, S., and Hwang, K. (2015). Resiliency of deep neural networks under quantization. *arXiv preprint arXiv:1511.06488.*

Tjeng, V., Xiao, K., and Tedrake, R. (2017). Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356.*

Tsai, Y.-L., Hsu, C.-Y., Yu, C.-M., and Chen, P.-Y. (2021). Formalizing generalization and adversarial robustness of neural networks to weight perturbations. *Advances in Neural Information Processing Systems*, 34:19692–19704.

Vasilakos, A. V., Tang, Y., Yao, Y., et al. (2016). Neural networks for computer-aided diagnosis in medicine: a review. *Neurocomputing*, 216:700–708.

Weng, T.-W., Zhao, P., Liu, S., Chen, P.-Y., Lin, X., and Daniel, L. (2020). Towards certificated model robustness against weight perturbations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6356–6363.

Wicker, M., Laurenti, L., Patane, A., Chen, Z., Zhang, Z., and Kwiatkowska, M. (2021). Bayesian inference with certifiable adversarial robustness. In *International Conference on Artificial Intelligence and Statistics*, pages 2431–2439. PMLR.

Wicker, M., Laurenti, L., Patane, A., and Kwiatkowska, M. (2020). Probabilistic safety for bayesian neural networks. In *Conference on uncertainty in artificial intelligence*, pages 1198–1207. PMLR.

Wicker, M., Sosnin, P., Janik, A., Müller, M. N., Weller, A., and Tsay, C. (2024). Certificates of differential privacy and unlearning for gradient-based training. *arXiv preprint arXiv:2406.13433*.

Wicker, M. R., Heo, J., Costabello, L., and Weller, A. (2023). Robust explanation constraints for neural networks. In *The Eleventh International Conference on Learning Representations*.

Wu, D., Xia, S.-T., and Wang, Y. (2020). Adversarial weight perturbation helps robust generalization. *Advances in neural information processing systems*, 33:2958–2969.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Zhang, J., Zhou, Y., and Saab, R. (2023a). Post-training quantization for neural networks with provable guarantees. *SIAM Journal on Mathematics of Data Science*, 5(2):373–399.

Zhang, Y., Song, F., and Sun, J. (2023b). Qebverif: Quantization error bound verification of neural networks. In *International Conference on Computer Aided Verification*, pages 413–437. Springer.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable]: Yes. See section 3 and 4
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable]: Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable]: Yes
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable]: Yes
   (b) Complete proofs of all theoretical results. [Yes/No/Not Applicable]: Yes
   (c) Clear explanations of any assumptions. [Yes/No/Not Applicable]: Yes
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable]: Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable]: Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes/No/Not Applicable]: iple times). Not Applicable
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes/No/Not Applicable]: Yes
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes/No/Not Applicable]: Yes
   (b) The license information of the assets, if applicable. [Yes/No/Not Applicable]: Yes
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable]: No
   (d) Information about consent from data providers/curators. [Yes/No/Not Applicable]: Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable]: Not Applicable
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Yes/No/Not Applicable]: Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable]: Not Applicable

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable]: Not Applicable

A Experimental details

In this section of the appendix, we highlight the details of the experiments discussed in the main paper.

A.1 Dataset

Most of the experiments have been carried out using the MNIST LeCun et al. (2010) and F-MNIST Xiao et al. (2017) datasets. For both dataset, we convert image pixel values from the range [0, 255] to [0, 1] and do not apply any other preprocessing techniques.

We also use CIFAR10 Krizhevsky et al. (2009) and SVHN Netzer et al. (2018) for the additional experiments described in Section B.3 of the Supplementary Material.

For CIFAR-10, the training data was preprocessed with random 32x32 cropping with padding of 4, random horizontal flipping, and normalization using a mean of (0.4914, 0.4822, 0.4465) and a standard deviation of (0.2023, 0.1994, 0.2010). For the test data, only normalization was applied. We applied random cropping and normalization for the SVHN training data with a mean and standard deviation of (0.5, 0.5, 0.5). For the testing data, we used the same normalization without cropping.

A.2 quantisation-robust training parameters

Normal training For all datasets, the neural networks were trained using the stochastic gradient descent (SGD) algorithm with a momentum of 0.9 and a weight decay of 5×10^{-4}. The initial learning rate is set to 0.1 and is decayed by a factor of 10 after epochs 50, 75, and 100. Each model was trained for 125 epochs with a batch size of 100 on an NVIDIA GeForce RTX 4070 Laptop GPU.

Robust training with IBP For quantisation-robust training using IBP we also used the SGD algorithm with the same hyperparameters as defined in the normal training process. For robust training, to optimise NN's loss function $L_{qr}(\mathscr{N}(x), y, \mathscr{Q})$ with maximum diameters ϵ_w, ϵ_a and ϵ_{in} in training set $\mathcal{D}$, i.e.,

$$L_{qr}(\mathscr{N}(x), y, \mathscr{Q}) = \lambda L(\mathscr{N}(x), y) + (1-\lambda) L(\hat{z}_{\mathscr{Q}}, y),$$

we leverage a learning curriculum by scheduling the values of λ and ϵ. We use the term ϵ to collectively refer to ϵ_w, ϵ_a, and ϵ_{in}.

MNIST: We warm up the training process with $\lambda = 1$ in the first 2400 individual steps of gradient descent (training steps) and fix $\lambda = 0.5$ in the later stage. Simultaneously, we linearly increase the value of ϵ from 0 to the required maximum diameters in 24000 training steps.

FMNIST: We use the same schedules for λ and ϵ as in MNIST.

SVHN: We warm up λ in 6000 training steps and linearly increase ϵ in 15000 training steps.

CIFAR-10: We warm up λ in 12000 training steps and linearly increase ϵ in 12000 training steps.

Figure 7 illustrates the convergence process of the model when using the above strategies for ϵ and λ. The model consists of 5 hidden layers and 256 neurons per layer. We observe that while initially, the bounds are quite loose, resulting in a large robust lossduring the early stages, they do become tighter while the training progresses.

A.3 Verified Accuracy Experiments

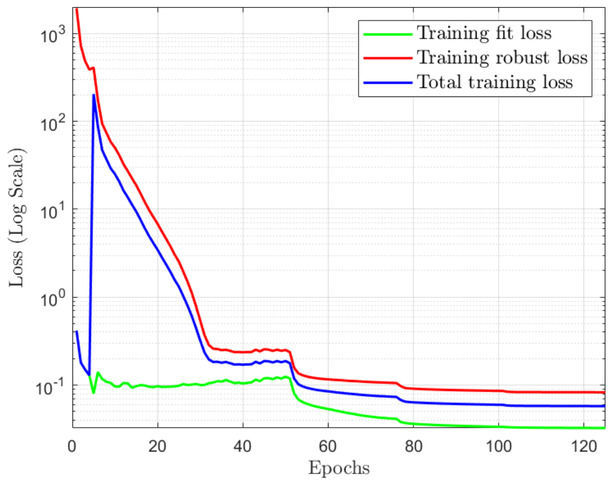
We evaluate the verified accuracy on a subset of the MNIST test set using bounds computed by the IBP method and the bilinear programming method. The model has three hidden layers, each containing 256 neurons. For each test sample, the optimization problem is solved using Gurobi with a 30-minute time limit. If Gurobi fails to find a solution within the allotted time or the optimization is unsuccessful due to an out-of-memory error, the sample is classified as non-robust.

B Additional Experiments

In this section of the Supplementary Material, we report additional experiments to those discussed in the main paper.

B.1 Analysis on the Effect of Lambda in quantisation-Robust Training

We train a neural network with 3 hidden layers and 256 neurons per layer on the MNIST dataset with the maximum diameters ϵ_w, ϵ_a and ϵ_{in} equal to 1/64, 1/32 and 1/32, respectively, which are roughly corresponding to a generic 6bit quantisation scheme. We analyse the effect of λ on the convergence ability and guaranteed accuracy of the model. Figure 8 shows that the model converges across all values of λ, however, the value of λ needs to be chosen carefully for the good balance between verified accuracy and nominal accuracy (which is the accuracy of the full precision model). Empirically, we found the value of $\lambda = 0.5$ works well on different values of ϵ and datasets.



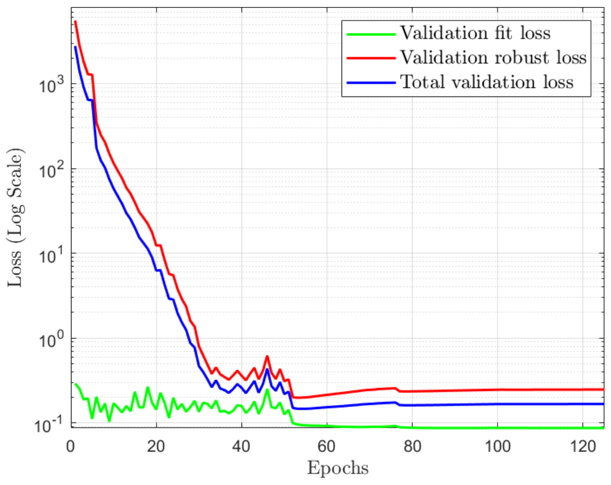
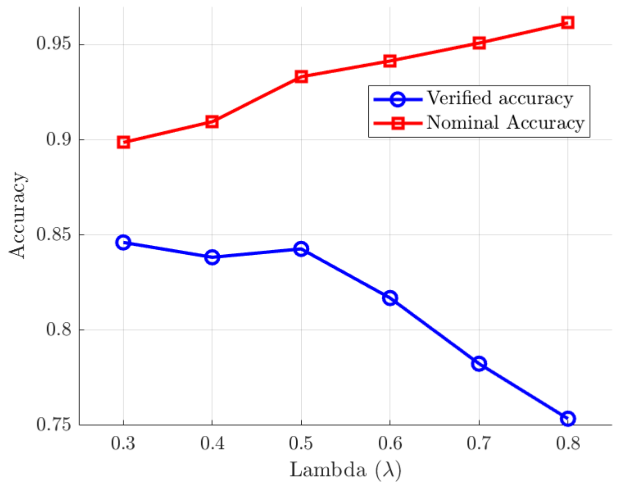


Figure 7: Model Convergence during the Training and Validation Processes



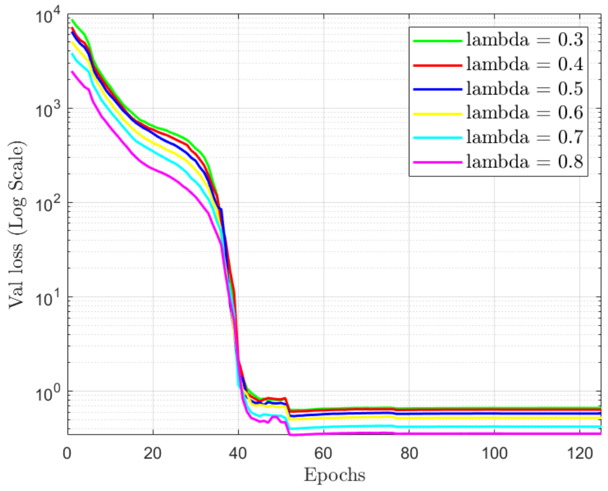


Figure 8: Effect of lambda on the guaranteed accuracy and convergence of the model.

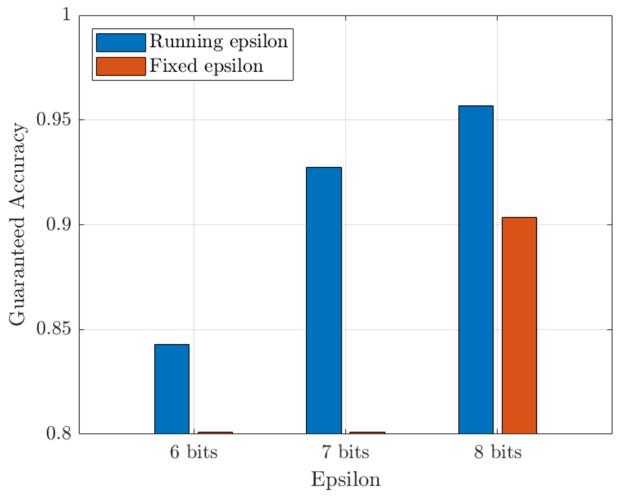
B.2 Analysis on the Effect of Epsilon Scheduling Strategy in quantisation-Robust Training

We compare the guaranteed accuracy and the convergence of the model over different values of ϵ in two cases: linearly increased to ϵ_{train} (running epsilon) and fixed as ϵ_{train} from the start (fixed epsilon).

Figure 9 illustrates that models trained without the epsilon schedule fail to converge when using an ϵ equivalent to 6-bit and 7-bit quantisation, due to the loss function's lack of convergence. Implementing the running epsilon strategy mitigates the impact of large robust loss, thereby stabilizing the training process.

B.3 Verified accuracy on other datasets

Table 1 provides additional results of verified accuracy testing in different datasets. We use the sample value of ϵ for the maximum diameters ϵ_w, ϵ_a and ϵ_{in}. MNIST model has 2 hidden layers with 256 neurons each. The model used for Fashion-MNIST, SVHN, and CIFAR-10 dataset has 5 hidden layers with 256-128-64-32-16 neurons, respectively. The verified accuracy is computed using 10,000 images in the MNIST test set. Dash "–" indicates these models obtain 0% verified accuracy and we were unable to verify them using the IBP verification method. These results show that our IBP training method can obtain models which are not only robust but also easier to verify.



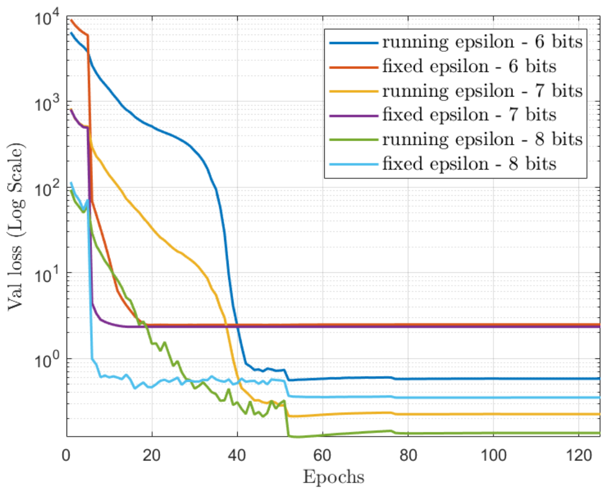


Figure 9: Effect of epsilon strategy on the guaranteed accuracy and convergence of model

Table 1: Verified Accuracy using IBP Verification Method

Dataset	**Epsilon**	**Model trained with IBP**	**Model trained normally**
MNIST	1/255	97.63%	55.56%
	2/255	96.35%	5.40%
	4/255	93.85%	--
	8/255	91.11%	--
Fashion_MNIST	1/255	82.69%	--
	2/255	78.72%	--
	3/255	74.59%	--
	4/255	70.97%	--
	7/255	51.79%	--
SVHN	1/511	69.36%	--
	1/255	55.02%	--
CIFAR10	1/511	36.56%	--
	1/255	22.57%	--

B.4 Runtime

In Table 2, we compare normal training and robust training in terms of verified accuracy, computed using bilinear optimization. The bounds were calculated on a laptop equipped with an Intel Ultra7-155H (1.40 GHz), 32 GB of RAM, 16 physical cores, and 22 logical processors. The model used has three hidden layers with 256 neurons per layer (approximately 335K network parameters, 784 inputs, and 768 activations to be bounded). The time limit for solving the verification problem is 30 minutes. We observe that most samples cannot be verified within 30 minutes when models are trained normally, while models trained with IBP can be verified more quickly and efficiently.

C Proof of IBP robust training method

In Proposition 2 of the main paper, we perform interval bound propagation using the propagation equations specific to the midpoint-radius representation of intervals, as discussed by (Rump, 1999; Mirman et al., 2018).

We give here a proof of the propagation equations given in Equations 6- 13

The equations are based on the midpoint-radius representation of an interval. Specifically given an interval $[z^L, z^U]$, this can also be equivalently represented by using its midpoint $z^\mu = (z^U + z^L)/2$ and its radius $z^r = (z^U - z^L)/2$. Similarly, we can represent the intervals over the weights using the midpoint $W^\mu = W$

Table 2: Additional verified accuracy and runtime comparison

Model	**Epsilons**	**Equivalent quantisation**	**Number of samples**	**Robust**	**Non-robust**	**Time-out**	**Verified**	**Average runtime(s)**
IBP	1/64-0-0	6 bit input	1182	1062	120	0	89.98%	39
Normal	1/64-0-0	6 bit input	60	10	0	50	16.67%	1800
IBP	1/256-0-0	8 bit input	515	502	13	0	97.47%	65
Normal	1/256-0-0	8 bit input	200	194	4	0	96%	84
IBP	1/64-1/32-1/32	6 bits	50	30	20	15	60%	787
Normal	1/64-1/32-1/32	6 bits	50	0	0	50	0%	1800

of each weight interval and its radius $W^r = W^{k,\epsilon_w}$.

We here give the explicit formulas for the propagation through the first layer – the computations for the subsequent layers follow iteratively from this. We find that the input interval has the midpoint $z^{(0),\mu} = z^{(0)}(x)$ and radius $z^{(0),r} = \epsilon_{in}$. The first layer is $h^{(1)} = W^{(1)}z^{(0)} + b^{(1)}$ and $z^{(1)} = \sigma(h^{(1)})$. Using the equations for multiplication and addition from Definition 2.1 in (Rump, 1999), we obtain:

$$h^{(1),\mu} = W^{(1)}z^{(0),\mu} + b^{(k)}$$

and

$$h^{(1),r} = |W^{(1)}|z^{(0),r} + W^{(1),\epsilon_w}|z^{(0),\mu}| + W^{(1),\epsilon_w}z^{(0),r} + b^{k,\epsilon_w}.$$

As proved in Proposition 2.3 of (Rump, 1999) this is an over-approximation of the propagated bound.

Propagating the upper and lower bounds of the preactivation through the activation function, which is monotonic and hence preserves the maximum and minimum, and are adding the error due to the quantisation of the activation, we have:

$$z^{(1),L}(x) = \sigma(h^{(1),\mu}(x) - h^{(1),r}(x)) - \epsilon_a$$
$$z^{(1),U}(x) = \sigma(h^{(1),\mu}(x) + h^{(1),r}(x)) + \epsilon_a$$

Finally, Equations 12 - 13 are obtained by converting the minimum and maximum of the interval over the activation values back to the midpoint-radius representation, so that it can be used iteratively for the computations of the following layer.