Adversarial Robustness Certification for Bayesian Neural Networks

Matthew Wicker[1][0000−0003−0779−3114], Andrea Patane[2][0000−0003−0492−4860], Luca Laurenti[3][0000−0003−1190−6097], and Marta Kwiatkowska(✉)[4][0000−0001−9022−7599]

[1] Imperial College, London, United Kingdom
m.wicker@imperial.ac.uk
[2] School of Computer Science and Statistics, Trinity College Dublin, Ireland
apatane@tcd.ie
[3] Delft Center for Systems and Control (DCSC), TU Delft, Delft, Netherlands
l.laurenti@tudelft.nl
Department of Computer Science, University of Oxford, Oxford, United Kingdom
marta.kwiatkowska@cs.ox.ac.uk

Abstract. We study the problem of certifying the robustness of Bayesian neural networks (BNNs) to adversarial input perturbations. Specifically, we define two notions of robustness for BNNs in an adversarial setting: probabilistic robustness and decision robustness. The former deals with the probabilistic behaviour of the network, that is, it ensures robustness across different stochastic realisations of the network, while the latter provides guarantees for the overall (output) decision of the BNN. Although these robustness properties cannot be computed analytically, we present a unified computational framework for efficiently and formally bounding them. Our approach is based on weight interval sampling, integration and bound propagation techniques, and can be applied to BNNs with a large number of parameters independently of the (approximate) inference method employed to train the BNN. We evaluate the effectiveness of our method on tasks including airborne collision avoidance, medical imaging and autonomous driving, demonstrating that it can compute non-trivial guarantees on medium size images (i.e., over 16 thousand input parameters).

Keywords: Certification · Bayesian Neural Networks · Adversarial Robustness · Classification · Regression · Uncertainty

1 Introduction

While neural networks (NNs) regularly obtain state-of-the-art performance in many supervised machine learning problems [2, 16], they are vulnerable to adversarial attacks, i.e., imperceptible modifications of their inputs that result in an incorrect prediction [46]. Along with several other vulnerabilities [8], the discovery of adversarial examples has made the deployment of NNs in real-world, safety-critical applications increasingly challenging. The design and analysis of

methods that can mitigate such vulnerabilities, or compute provable guarantees on their worst-case behaviour in adversarial conditions, is therefore of utmost importance [48].

While retaining the advantages intrinsic to deep learning, Bayesian neural networks (BNNs), i.e., NNs with a probability distribution placed over their weights and biases [36], enable probabilistically principled evaluation of *model uncertainty*. Because of their ability to model uncertainty [29], the application of BNNs is particularly appealing in safety-critical scenarios, where uncertainty could be taken into account at prediction time to enable safe decision-making [4,12,35,61]. To this end, various techniques have been proposed for the evaluation of BNNs' robustness, including generalisation of gradient-based adversarial attacks [33], statistical verification techniques [13], and formal verification approaches aimed at verifying that the decisions made by a BNN are safe [1, 7] or checking the robustness of the neural networks sampled from the BNN posterior [7, 13, 31]. The increasingly diverse techniques for analysing robustness of Bayesian neural networks have resulted in divergent robustness properties, some directly analysing the stochasticity of the system [13] and others directly adapting robustness specifications from deterministic systems [7]. To the best of our knowledge, there is a lack of systematic, unified approaches for computing formal (i.e., with certified bounds) guarantees on the range of emergent quantitative robustness properties against adversarial input perturbations for BNNs.

In this work, we develop a probabilistic verification framework to quantify the adversarial robustness of BNNs. In particular, we model adversarial robustness as an *input-output specification* defined by a given compact set of input points, $T \subseteq \mathbb{R}^m$, and a given convex polytope output set, $S \subseteq \mathbb{R}^n$ (called a safe set). A neural network satisfies this specification if all points in T are mapped into S. For a particular specification, we focus on two main properties of a BNN of interest for adversarial prediction settings: *probabilistic robustness* [13, 54] and *decision robustness* [7,25]. The former is defined as the probability that a network sampled from the posterior distribution is robust, which thus provides a general measure of the robustness of a BNN. In contrast, *decision robustness* focuses on the decision step, and evaluates the robustness of the optimal decision of a BNN. That is, a BNN satisfies decision robustness if, for all points in T, the expectation of the output of the BNN in the case of regression, or the argmax of the expectation of the softmax for classification, are contained in S.

Unfortunately, evaluating probabilistic and decision robustness for a BNN is not trivial, as it involves computing distributions and expectations of high-dimensional random variables passed through a non-convex function. Nevertheless, we derive a unified algorithmic framework based on computations over the BNN weight space that yields *certified lower* and *upper bounds* for both properties. Specifically, we show that probabilistic robustness is equivalent to the measure, w.r.t. the BNN posterior, of the set of weights for which the resulting deterministic NN is robust. Computing upper and lower bounds for the probability involves sampling compact sets of weights according to the BNN posterior, and propagating each of these weight sets, H, through the neural network ar-

chitecture, jointly with the input region T, to check whether all the networks instantiated by weights in H are safe. To do so, we generalise bound propagation techniques developed for deterministic neural networks to the Bayesian setting and instantiate explicit schemes for Interval Bound Propagation (IBP) and Linear Bound Propagation (LBP) [22]. Similarly, in the case of decision robustness, we show that formal bounds can be obtained by partitioning the weight space into different weight sets, and for each weight set J we employ bound propagation techniques to compute the maximum and minimum of the decision of the NN for any input point in T and any weight in the set J. The resulting extrema are then averaged w.r.t. posterior measure to obtain sound lower and upper bounds on decision robustness.

We empirically validate our framework using case studies from airborne collision avoidance [27], medical image recognition [60], and autonomous driving [44]. We demonstrate that our framework is able to compute sound upper and lower bounds for both notions of robustness for Bayesian neural networks. Moreover, we study the effect of approximate inference, as well as depth and width of the neural network classifier, on our guarantees. We find that our approach, even when using simple interval bound propagation, is able to provide non-trivial certificates of adversarial robustness and predictive uncertainty properties for Bayesian neural networks with four hidden layers and more than 16,000 input dimensions. We additionally use our approach to show how approximate Bayesian posteriors may provide provably robust uncertainty estimation for random noise inputs while failing to provide the same guarantees for more structured classes of out-of-distribution inputs[4].

In summary, this paper makes the following contributions[5].

- We present an algorithmic framework based on convex relaxation techniques for the robustness analysis of BNNs in adversarial settings.
- We derive explicit lower- and upper-bounding procedures based on IBP and LBP for the propagation of input and weight intervals through the BNN posterior function.
- We empirically show that our method can be used to certify BNNs consisting of multiple hidden layers and with hundreds of neurons per layer.

Probabilistic robustness was introduced in [54]. This work extends [54] in several aspects. In contrast to [54], which focused only on probabilistic robustness, here we also tackle decision robustness and embed the calculations for the two properties in a common computational framework. Furthermore, while the method in [54] only computes lower bounds, in this paper we also develop a technique for upper bound computation. Finally, we extend the empirical analysis to include additional datasets, evaluation of convolutional architectures, scalability analysis, as well as certification of out-of-distribution (OOD) uncertainty.

[4] An implementation to reproduce all the experiments can be found at: `https://github.com/matthewwicker/AdversarialRobustnessCertificationForBNNs`

[5] In view of space constraints, additional details are available in Appendix at `https://arxiv.org/abs/2306.13614`

Related Works The vast majority of existing NN verification methods have been developed specifically for deterministic NNs, with approaches including abstract interpretation [22], mixed integer linear programming [20, 40, 47, 58, 64], Monte Carlo search-based frameworks [26, 52, 59], convex relaxation [25, 49, 63] and SAT/SMT [27, 28]. However, these methods cannot be directly applied to BNNs because they all assume that the weights of the network are deterministic, i.e., fixed to a given value, while in the Bayesian setting weights are not fixed, but distributed according to the BNN posterior. Statistical approaches to quantify the robustness of BNNs that are ϵ approximately correct up to a confidence/probability of error bounded by $1 - \delta$, for $\delta > 0$, have been developed in [13, 35]. In contrast, the methods in this paper do not rely on confidence intervals and return guaranteed upper and lower bounds on the true probability that a BNN satisfies a specific property.

Since the publication of our preliminary work [54], other papers have studied the problem of verifying BNN robustness [1, 3, 7, 31, 55, 56]. However, [7] only considers verification of BNNs with weight distributions of bounded support, and consequently does not include Gaussian posterior distributions, which are commonly employed in practice. [1] develops an approach based on dynamic programming to certify decision robustness for BNNs, which improves the precision of BNN verification by performing bound propagation in the latent space of BNNs, rather than working on the space of weights. However, this approach is restricted to decision robustness. Further, [3] develops an approach based on mixed integer linear programming (MILP), which is specific for probabilistic robustness. It is unclear how these approaches could be extended to encompass both probabilistic and decision robustness. In contrast, in this paper we propose a simple and general framework that encompasses both decision and probabilistic robustness, and can be applied to both fully-connected and convolutional neural network architectures. Another related method is [31], which takes a distribution-free approach and considers a dynamical system whose one-step dynamics includes a neural network, and computes the set of weights that satisfy an infinite-horizon safety property. Note that, as the support of a Gaussian distribution is unbounded, similarly to [7], this approach does not support Gaussian posterior distributions over the weights. We also mention [56], which builds on the results of [55] to develop certification for reach-avoid properties of dynamical systems described by BNNs. Finally, [53] considers certifiable robust training and introduces the concept of robust likelihood that we employ in our experimental evaluation.

In the context of Bayesian learning, methods to compute adversarial robustness measures have been explored for Gaussian processes (GPs), both for regression [14] and classification tasks [39,42]. However, because of the non-linearity in NN architectures, GP-based approaches cannot be directly employed for BNNs. Furthermore, the vast majority of approximate Bayesian inference methods for BNNs do not utilise Gaussian approximations over the latent space [10]. In contrast, our method is specifically tailored to take into account the non-linear

nature of BNNs and can be directly applied to a range of approximate Bayesian inference techniques used in the literature.

2 Background on Bayesian Deep Learning

We consider a dataset of $n_{\mathcal{D}}$ independent pairs of inputs and labels, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n_{\mathcal{D}}}$, with $x_i \in \mathbb{R}^m$, where each output $y \in \mathbb{R}^n$ is either a one-hot class vector for classification or a real-valued vector for regression. The aim of Bayesian learning is to learn the function generating $\mathcal{D}$ via a feed forward-neural network $f^w : \mathbb{R}^m \to \mathbb{R}^n$, parameterised by a vector $w \in \mathbb{R}^{n_w}$ containing all its weights and biases. We denote with $f^{w,1}, ..., f^{w,K}$ the K layers of f^w and take the activation function of the ith layer to be $\sigma^{(i)}$, abbreviated to just σ in the case of the output activation.[6] Throughout this paper, we will use $f^w(x)$ to represent pre-activation of the last layer.

Bayesian deep learning starts with a prior distribution, $p(w)$, over the vector $\mathbf{w}$ of random variables associated to the weights. Placing a distribution over the weights defines a stochastic process indexed by the input space, which we denote as $f^{\mathbf{w}}$. Note that we use bold to distinguish the stochastic process parameterised by a random variable, $f^{\mathbf{w}}$, and the deterministic function that results from sampling a single parameter value, f^w. To obtain the posterior distribution, the BNN prior is updated according to the likelihood, $p(\mathcal{D}|w)$, via the Bayes rule, i.e., $p(w|\mathcal{D}) \propto p(\mathcal{D}|w)p(w)$ [9]. The cumulative distribution of $p(w|\mathcal{D})$, which we denote as $P(\cdot)$, is such that for $R \subseteq \mathbb{R}^{n_w}$ we have:

$$P(R) := \int_R p(w|\mathcal{D})dw. \tag{1}$$

The posterior $p(w|\mathcal{D})$ is in turn used to calculate the output of a BNN on an unseen point, x^*. The distribution over outputs is called the posterior predictive distribution and is defined as:

$$p(y^*|x^*, \mathcal{D}) = \int p(y^*|x^*, w)p(w|\mathcal{D})dw. \tag{2}$$

When employing a Bayesian model, the overall final prediction is taken to be a single value, $\hat{y}$, that minimizes the Bayesian risk of an incorrect prediction according to the posterior predictive distribution and a loss function $\mathcal{L}$. Formally, the final decision of a BNN is computed as

$$\hat{y} = \arg\min_{y^*} \int_{\mathbb{R}^n} \mathcal{L}(y, y^*)p(y^*|x^*, \mathcal{D})dy^*. \tag{3}$$

This minimization is the subject of Bayesian decision theory [6], and the final form of $\hat{y}$ depends on the specific loss function $\mathcal{L}$ employed in practice. In this

[6] We assume, for the purposes of linear bound propagation in Appendix D.4, that the activation functions have a finite number of inflection points, which holds for activation functions commonly used in practice [23].

paper, we focus on two standard loss functions widely employed for classification and regression problems[7], described in more detail below.

Classification For classification problems, the 0-1 loss, denoted ℓ_{0-1}, is commonly employed. ℓ_{0-1} assigns a penalty of 0 to the correct prediction, and 1 otherwise. It can be shown that the optimal decision in this case is given by the class for which the predictive distribution obtains its maximum, i.e.:

$$\hat{y} = \underset{i=1,\dots,n}{\arg\max}\, p_i(y^*|x^*, \mathcal{D}) = \underset{i=1,\dots,n}{\arg\max}\, \mathbb{E}_{w\sim p(w|\mathcal{D})}\left[\sigma_i(f^w(x))\right],$$

where σ_i represents the ith output component of the softmax function.

Regression For regression problems, the ℓ_2 loss is generally employed. ℓ_2 assigns a penalty to a prediction according to its ℓ_2 distance from the ground truth. It can be shown that the optimal decision in this case is given by the expected value of the BNN output over the posterior distribution, i.e., $\hat{y} = \mathbb{E}_{w\sim p(w|\mathcal{D})}\left[f^w(x)\right]$. Unfortunately, because of the non-linearity of neural network architectures, the computation of the posterior distribution over the weights, $p(w|\mathcal{D})$, is generally intractable [36]. Hence, various approximation methods have been studied to perform inference with BNNs in practice. Among these, we will consider Hamiltonian Monte Carlo (HMC) [36] and Variational Inference (VI) [10]. While HMC is a sample-based method that involves defining a Markov chain whose invariant distribution is $p_{\mathbf{w}}(w|\mathcal{D})$ [36], VI proceeds by finding a Gaussian approximating distribution over the weight space $q(w) \sim p_{\mathbf{w}}(w|\mathcal{D})$ in a trade-off between approximation accuracy and scalability. For simplicity of notation, in the rest of the paper we will indicate with $p(w|\mathcal{D})$ the posterior distribution estimated by either of the two methods, and clarify the methodological differences when they arise.

3 Problem Statement

We focus on local specifications defined over an input compact set $T \subseteq \mathbb{R}^m$, which we assume to be a box (axis-aligned linear constraints), and output set $S \subseteq \mathbb{R}^n$ in the form of a convex polytope:

$$S = \{y \in \mathbb{R}^n \,|\, C_S y + d_S \geq 0\}, \tag{4}$$

where $C_S \in \mathbb{R}^{n_S \times n}$ and $d_S \in \mathbb{R}^{n_S}$ are the matrix and vector encoding the polytope constraints, with n_S being the number of output constraints. Throughout the paper we will refer to an input-output set pair, T and S, as defined above, as a *robustness specification*. We note that our formulation of robustness specification captures various important properties used in practice, such as classifier

[7] In Appendix B we discuss how our method can be generalised to other losses commonly employed in practice.

monotonicity [45], adversarial robustness [24,26], and individual fairness [5]. For instance, targeted adversarial robustness for classification, which aims to find an adversarial example belonging to a specified class, can be captured by setting C_S to an $n_S \times n$ matrix of all zeros with a -1 in the diagonal entry corresponding to the true class and a 1 on the diagonal entry corresponding to the target class. Similarly, for regression, one uses C_S to encode the absolute deviation from the target value and d_S to encode the maximum tolerable deviation.

Probabilistic robustness accounts for the probabilistic behaviour of a BNN with respect to a robustness specification.

Definition 1 (Probabilistic robustness). *Given a Bayesian neural network $f^{\mathbf{w}}$, an input set $T \subseteq \mathbb{R}^m$ and an output set $S \subseteq \mathbb{R}^n$, also called safe set of outputs, define probabilistic robustness as*

$$P_{safe}(T, S) := Prob_{w \sim p(w|\mathcal{D})}(\forall x \in T, f^w(x) \in S). \tag{5}$$

Given $\eta \in [0, 1]$, we then say that $f^{\mathbf{w}}$ is probabilistically robust, or safe, for robustness specifications (T, S) with probability at least η iff $P_{safe}(T, S) \geq \eta$.

Probabilistic robustness considers the adversarial behaviour of the model while accounting for the uncertainty arising from the posterior distribution. In particular, $P_{\text{safe}}(T, S)$ quantifies *the proportion* of networks sampled from $f^{\mathbf{w}}$ that satisfy a given input-output specification, and can be used directly as a measure of compliance for Bayesian neural networks [7,17,35]. Exact computation of $P_{\text{safe}}(T, S)$ is hindered by the size and non-linearity of neural networks. Therefore, in this work, we aim to compute provable bounds on probabilistic robustness.

Problem 1 (Bounding probabilistic robustness). Given a Bayesian neural network $f^{\mathbf{w}}$, an input set $T \subseteq \mathbb{R}^m$ and a set $S \subseteq \mathbb{R}^n$ of safe outputs, compute (non-trivial) lower and upper bounds P^L_{safe} and P^U_{safe} such that

$$P^L_{\text{safe}} \leq P_{\text{safe}}(T, S) \leq P^U_{\text{safe}}. \tag{6}$$

3.1 Decision Robustness

While P_{safe} attempts to measure the probability of robustness of neural networks sampled from the BNN posterior, we are often interested in evaluating robustness w.r.t. a specific decision. In order to do so, we consider *decision robustness*, which is computed over the final decision of the BNN. In particular, given a loss function and a decision $\hat{y}$ we have the following.

Definition 2 (Decision robustness). *Consider a Bayesian neural network $f^{\mathbf{w}}$, an input set $T \subseteq \mathbb{R}^m$ and an output set $S \subseteq \mathbb{R}^n$. Assume that the decision for a loss $\mathcal{L}$ for $x \in \mathbb{R}^m$ is given by $\hat{y}(x)$ (Equation 3). Then, the Bayesian decision is considered to be robust if $\forall x \in T$, $\hat{y}(x) \in S$.*

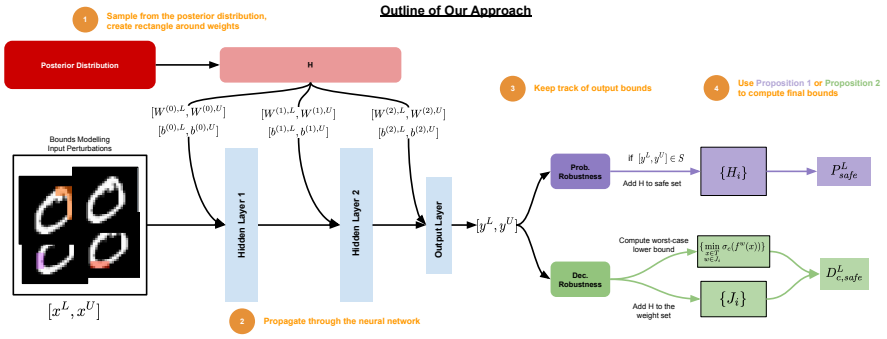

Fig. 1. A diagram illustrating a single iteration of the computational flow for the certification process of a BNN w.r.t. decision robustness (green) and probabilistic robustness (purple). This process is summarised in Algorithm 1.

As discussed in Section 2, since the specific form of the decision depends on the loss function, the definition of decision robustness takes different form depending on whether the BNN is used for classification or for regression. We thus arrive at the following problem.

Problem 2 (Bounding decision robustness). Let $f^{\mathbf{w}}$ be a BNN with posterior distribution $p(w|\mathcal{D})$. Consider a robustness specification (T, S) and assume $\mathcal{L} = \ell_{0-1}$ for classification or $\mathcal{L} = \ell_2$ for regression. We aim at computing (non-trivial) lower and upper bounds D^L_{safe} and D^U_{safe} such that:

$$D^L_{\text{safe}} \leq \mathbb{E}[s(f^w(x))] \leq D^U_{\text{safe}} \quad \forall x \in T,$$

where s corresponds to the likelihood function σ in the case of classification (e.g., the softmax) and simply denotes the identity function in the case of regression.

Problem 2 suggests that, while for regression we can simply bound the expected output of the BNN, for classification we need to bound the predictive posterior to compute bounds on the final decision, i.e., we need to propagate these inside the softmax. This is similar to what is done for deterministic neural networks, where, in the case of classification, the bounds are often computed over the logits, and then used to provide guarantees for the final decision [25].

3.2 Approach Outline

We design an algorithmic framework for computing worst- and best-case bounds (lower and upper bounds, respectively) on local robustness properties for Bayesian neural networks, taking account of both the posterior distribution (P^L_{safe} and P^U_{safe}) and the overall model decision (D^L_{safe} and D^U_{safe}). First, we show how the two robustness properties of Definitions 1 and 2 can be reformulated in terms of computation over weight intervals. This allows us to derive a unified approach,

which enables bounding of the robustness of the BNN posterior (i.e., probabilistic robustness) and that of the overall model decision (i.e., decision robustness) by means of *bound propagation* and *posterior integral* computation over hyper-rectangles. For a discussion of when each bound may be useful see Appendix A.

A visual outline for our framework is presented in Figure 1. The presentation of the framework is organised as follows. We first introduce a general theoretical schema for bounding the robustness quantities of interest (Section 4). We then show how the required integral computations can be achieved for practical Bayesian posterior inference techniques (Section 5.1). This allows us to extend bound propagation techniques to deal with both input variable intervals and intervals over the weight space, which we rely on to instantiate approaches respectively based on Interval Bound Propagation (Section 5.2) and Linear Bound Propagation techniques (Appendix C). Finally, in Section 6, we present an overall algorithm that produces the desired bounds.

4 BNN Adversarial Robustness via Weight Sets

We show how a single computational framework can be leveraged to compute bounds on both definitions of BNN robustness. We start by converting the computation of robustness into the weight space and then define a family of weight intervals that we utilise to bound the integrations required by both definitions. Proofs for the main results in this section are presented in Appendix D.

4.1 Bounding Probabilistic Robustness

We first show that the computation of $P_{\text{safe}}(T, S)$ is equivalent to computing a *maximal* set of safe weights H such that each network associated to weights in H is safe w.r.t. the robustness specification at hand.

Definition 3 (Maximal safe and unsafe sets). *We say that $H \subseteq \mathbb{R}^{n_w}$ is the maximal safe set of weights from T to S, or simply the maximal safe set of weights, iff $H = \{w \in \mathbb{R}^{n_w} \mid \forall x \in T, f^w(x) \in S\}$. Similarly, we say that $K \subseteq \mathbb{R}^{n_w}$ is the maximal unsafe set of weights from T to S, or simply the maximal unsafe set of weights, iff $K = \{w \in \mathbb{R}^{n_w} \mid \exists x \in T, f^w(x) \notin S\}$.*

Intuitively, H and K simply encode the input-output specifications S and T in the BNN weight space. The following lemma, which follows from Equation 5, allows us to relate the maximal sets of weights to probabilistic robustness.

Lemma 1. *Let H and K be the maximal safe and unsafe sets of weights from T to S. Assume that $w \sim p(w|\mathcal{D})$. Then, it holds that*

$$P(H) = \int_H p(w|\mathcal{D})dw = P_{safe}(T, S) = 1 - \int_K p(w|\mathcal{D})dw = 1 - P(K). \quad (7)$$

Unfortunately, an exact computation of sets H and K is infeasible in general and may not be possible to capture using any finite number of sets. However,

we can compute subsets of H and K. Such subsets can then be used to compute upper and lower bounds on the value of $P_{\text{safe}}(T, S)$ by considering subsets of the maximal safe and unsafe weights.

Definition 4 (Safe and unsafe sets). *Given a maximal safe set H or a maximal unsafe set K of weights, we say that $\hat{H}$ and $\hat{K}$ are a safe and unsafe set of weights from T to S iff $\hat{H} \subseteq H$ and $\hat{K} \subseteq K$, respectively.*

Without maximality, we no longer have strict equality in Lemma 1, but we can use $\hat{H}$ and $\hat{K}$ to arrive at bounds on the value of probabilistic robustness. Specifically, we proceed by defining $\hat{H}$ and $\hat{K}$ as the union of a family of disjoint weight intervals, as these can provide flexible approximations of H and K. That is, we consider $\mathcal{H} = \{H_i\}_{i=1}^{n_H}$, with $H_i = [w_i^{L,H}, w_i^{U,H}]$ and $\mathcal{K} = \{K_i\}_{i=1}^{n_K}$, with $K_i = [w_i^{L,K}, w_i^{U,K}]$, such that $H_i \subset H$ and $K_i \subset K$, $\hat{H} = \bigcup_{i=1}^{n_H} H_i$, $\hat{K} = \bigcup_{i=1}^{n_K} K_i$, and $H_i \cap H_j = \emptyset$ and $K_i \cap K_j = \emptyset$, for any $i \neq j$. Hence, as a consequence of Lemma 1, and by the fact that $\hat{H} \subseteq H$ and $\hat{K} \subseteq K$, we obtain the following.

Proposition 1 (Bounds on probabilistic robustness). *Let H and K be the maximal safe and unsafe sets of weights from T to S. Consider two families of pairwise disjoint weight intervals $\mathcal{H} = \{H_i\}_{i=1}^{n_H}$, $\mathcal{K} = \{K_i\}_{i=1}^{n_K}$, where for all i it holds that $H_i \subseteq H$ and $K_i \subseteq K$. Let $\hat{H} \subseteq H$ and $\hat{K} \subseteq K$ be non-maximal safe and unsafe sets of weights, with $\hat{H} = \bigcup_{i=1}^{n_H} H_i$ and $\hat{K} = \bigcup_{i=1}^{n_K} K_i$. Assume that $w \sim p(w|\mathcal{D})$. Then, it holds that*

$$P_{safe}^{L} := \sum_{i=1}^{n_H} P(H_i) \leq P_{safe}(T, S) \leq 1 - \sum_{i=1}^{n_K} P(K_i) =: P_{safe}^{U}, \tag{8}$$

that is, P_{safe}^{L} and P_{safe}^{U} are lower and upper bounds on probabilistic robustness.

Through the use of Proposition 1, we can thus bound probabilistic robustness by performing computation over sets of safe and unsafe intervals.[8] Before explaining in detail how such bounds can be explicitly computed, we first show, in the next section, how a similar derivation leads us to analogous bounds and computations for decision robustness.

4.2 Bounding Decision Robustness

The key difference between our formulation of probabilistic robustness and that of decision robustness is that, for the former, we are only interested in the behaviour of neural networks extracted from the BNN posterior that satisfy the robustness requirements (hence the distinction between H- and K-weight intervals), whereas to compute sound bounds on decision robustness we need to take into account the overall worst-case behaviour of an expected value computed for the BNN predictive distribution. As such, rather than computing safe and

[8] In Appendix E.4 we extend the results to general hyper-rectangles by using the Bonferroni bound.

unsafe sets, we only need a family of weight sets, $\mathcal{J} = \{J_i\}_{i=1}^{n_J}$, which we can rely on for bounding $D_{\text{safe}}(T, S)$. In the following, we explicitly show how to do this for classification with likelihood σ. The bound for regression follows similarly by using the identity function as σ.

Proposition 2 (Bounding decision robustness). *Let $\mathcal{J} = \{J_i\}_{i=1}^{n_J}$, with $J_i \subset \mathbb{R}^{n_w}$, be a family of disjoint weight intervals. Let σ^L and σ^U be vectors that lower- and upper-bound the co-domain of the final activation function, and $c \in \{1, \ldots, m\}$ an index spanning the BNN output dimension. Define:*

$$D^L_{safe,c} := \sum_{i=1}^{n_J} P(J_i) \min_{\substack{x \in T \\ w \in J_i}} \sigma_c(f^w(x)) + \sigma^L \left(1 - \sum_{i=1}^{n_J} P(J_i)\right) \tag{9}$$

$$D^U_{safe,c} := \sum_{i=1}^{n_J} P(J_i) \max_{\substack{x \in T \\ w \in J_i}} \sigma_c(f^w(x)) + \sigma^U \left(1 - \sum_{i=1}^{n_J} P(J_i)\right). \tag{10}$$

Consider $D^L_{safe} = [D^L_{safe,1}, \ldots, D^L_{safe,m}]$ and $D^U_{safe} = [D^U_{safe,1}, \ldots, D^U_{safe,m}]$, then:

$$D^L_{safe} \leq \mathbb{E}_{p(w|\mathcal{D})}[\sigma(f^w(x))] \leq D^U_{safe} \quad \forall x \in T,$$

that is, D^L_{safe} and D^U_{safe} bound the predictive posterior in T.

Intuitively, the first term in the bounds of Equations (9) (and similarly(10)) considers the worst-case output for the input set T and each interval J_i, while the second term accounts for the worst-case value of the posterior mass not captured by the family of intervals $\mathcal{J}$. The bound is valid for any family of intervals $\mathcal{J}$. Ideally, however, the partition should be finer around regions of high probability mass of the posterior distribution, as these make up the dominant term in the computation of the posterior predictive. We discuss in Section 5 how we select these intervals in practice so as to empirically obtain non-vacuous bounds.

4.3 Computation of the Lower and Upper Bounds

We now propose a unified approach to computing the lower and upper bounds. We observe that Equations (8), (9) and (10) require the integration of the posterior distribution over weight intervals. While this is in general intractable, we have built the bounds so that H_i, K_i and J_i are axis-aligned hyper-rectangles, and so the computation can be done exactly for commonly used approximate Bayesian inference methods (discussed in detail in Section 5.1).

For the explicit computation of decision robustness, the only missing ingredient is then the computation of the minimum and maximum of $\sigma(f^w(x))$ for $x \in T$ and $w \in J_i$. We do this by bounding the BNN output for any given rectangle, R, in the weight space. That is, we will compute upper and lower bounds y^L and y^U such that:

$$y^L \leq \min_{\substack{x \in T \\ w \in R}} f^w(x) \quad y^U \geq \max_{\substack{x \in T \\ w \in R}} f^w(x), \tag{11}$$

which can then be used to bound $\sigma(f^w(x))$ by simple propagation over the softmax. The derivation of such bounds will be the subject of Section 5.2.

Finally, observe that, whereas for decision robustness we can simply select any weight interval J_i, for probabilistic robustness one needs to make a distinction between safe sets (H_i) and unsafe sets (K_i). It turns out that this can be done by bounding the output of the BNN in each of these intervals. For example, in the case of the safe sets, by definition we have that $\forall w \in H_i, \forall x' \in T$ it follows that $f^w(x') \in S$. By defining y^L and y^U as in Equation (11), we can see that it suffices to check whether $[y^L, y^U] \subseteq S$. Hence, the computation of probabilistic robustness also depends on the computation of such bounds.

Therefore, once we have shown how to compute $P(R)$ for any weight interval and y^L and y^U, the bounds in Proposition 1 and Proposition 2 can be computed explicitly, and we can thus bound probabilistic and decision robustness.

5 Explicit Bound Computation

In this section, we provide details of the computational schema needed to calculate the theoretical bounds presented in Section 4.

5.1 Integral Computation over Weight Intervals

Key to the bound computation is the ability to compute the integral of the posterior distribution over a combined set of weight intervals. Crucially, the shape of the weight sets $\mathcal{H} = \{H_i\}_{i=1}^{n_H}$, $\mathcal{K} = \{K_i\}_{i=1}^{n_K}$ and $\mathcal{J} = \{J_i\}_{i=1}^{n_J}$ is a parameter of the method, which can be leveraged to simplify the integral computation depending on the particular form of the approximate posterior distribution. We build each weight interval as an axis-aligned hyper-rectangle of the form $R = [w^L, w^U]$ for w^L and $w^U \in \mathbb{R}^{n_w}$.

Weight Intervals for Decision Robustness In the case of decision robustness, it suffices to sample any weight interval J_i to compute the bounds we derived in Proposition 2. Clearly, the bound is tighter if the $\mathcal{J}$ family is finer around the area of high probability mass for $p(w|\mathcal{D})$. In order to obtain such a family we proceed as follows. First, we define a *weight margin* $\gamma > 0$, whose role is to parameterise the radius of the weight intervals. We then iteratively sample weight vectors w_i from $p(w|\mathcal{D})$, for $i = 1, \ldots, n_J$, and define $J_i = [w_i^L, w_i^U] = [w_i - \gamma, w_i + \gamma]$. Thus defined weight intervals naturally concentrate around the area of greater density for $p(w|\mathcal{D})$, while asymptotically covering the whole support of the distribution.

Weight Intervals for Probabilistic Robustness On the other hand, for the computation of probabilistic robustness one has to make a distinction between safe and unsafe weight intervals, H_i and K_i. As explained in Section 4.3, this can be done by bounding the output of the BNN in each of these intervals.

For example, in the case of the safe sets, by definition, H_i is safe if and only if $\forall w \in H_i, \forall x' \in T$ we have that $f^w(x') \in S$. Thus, in order to build a family of safe (respectively unsafe) weight intervals H_i (resp. K_i), we proceed as follows. As for decision robustness, we iteratively sample weights w_i from the posterior used to build hyper-rectangles of the form $R_i = [w_i - \gamma, w_i + \gamma]$. We then propagate R_i through the BNN and check whether the output is (resp. is not) a subset of S. The derivation of such bounds on propagation will be the subject of Section 5.2.

Once the family of weights is computed, it remains to compute the cumulative distribution over such sets. The specific computations depend on the particular form of Bayesian approximate inference that is employed. We discuss explicitly the case of Gaussian variational approaches, and of sample-based posterior approximation (e.g., HMC).

Variational Inference For variational approximations, $p(w|\mathcal{D})$ takes the form of a multi-variate Gaussian distribution over the weight space. The resulting computations reduce to the integral of a multi-variate Gaussian distribution over a finite-sized axis-aligned rectangle, which can be computed using standard methods from statistics [15]. In particular, under the common assumption of variational inference with a Gaussian distribution with diagonal covariance matrix [30], i.e., $p(w|\mathcal{D}) = \mathcal{N}(\mu, \Sigma)$, with $\Sigma = \text{diag}(\Sigma_1, \ldots, \Sigma_{n_w})$, we obtain the following result for the posterior integration:

$$P(R) = \int_R p(w|\mathcal{D})dw = \prod_{j=1}^{n_w} \frac{1}{2} \left(\text{erf}\left(\frac{\mu_j - w_i^L}{\sqrt{2\Sigma_j}} \right) - \text{erf}\left(\frac{\mu_j - w_i^u}{\sqrt{2\Sigma_j}} \right) \right). \quad (12)$$

By plugging this into the bound equations for probabilistic robustness and for decision robustness, one obtains a closed-form formula for the bounds given weight set interval families $\mathcal{H}$, $\mathcal{K}$ and $\mathcal{J}$.

Sample-based Approximations In the case of sample-based posterior approximation (e.g., HMC), we have that $p(w|\mathcal{D})$ defines a distribution over a finite set of weights. In this case we can simplify the computations by selecting the weight margin $\gamma = 0$, so that each sampled interval is of the form $R = [w_i, w_i]$ and its probability under the discrete posterior will trivially be:

$$P(R_i) = p(w_i|\mathcal{D}). \quad (13)$$

5.2 Bounding Bayesian Neural Network Output

Given an input set, T, and a weight interval, $R = [w^L, w^U]$, the second key step in computing probabilistic and decision robustness is the bounding of the output of the BNN over R given T. That is, we need to derive methods to compute $[y^L, y^U]$ such that $\forall w \in [w^L, w^U], \forall x' \in T$ it follows that $f^w(x') \in [y^L, y^U]$.

In this section, we consider Interval Bound Propagation (IBP) as a method for computing the desired output set over-approximations, and defer the discussion of Linear Bound Propagation (LBP) to Appendix C. Before discussing IBP in more detail, we first introduce common notation for the rest of the section. We consider feed-forward neural networks of the form:

$$z^{(0)} = x, \quad \zeta_i^{(k+1)} = \sum_{j=1}^{n_k} W_{ij}^{(k)} z_j^{(k)} + b_i^{(k)}, \quad z_i^{(k)} = \sigma(\zeta_i^{(k)}) \tag{14}$$

for $k = 1, \ldots, K$ and $i = 0, \ldots, n_k$, where K is the number of hidden layers, $\sigma(\cdot)$ is a pointwise activation function, $W^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$ and $b^{(k)} \in \mathbb{R}^{n_k}$ are the matrix of weights and vector of biases that correspond to the kth layer of the network, and n_k is the number of neurons in the kth hidden layer. Note that, while Equation (14) is written explicitly for fully-connected layers, convolutional layers can be accounted for by embedding them in fully-connected form [63].

We write $W_{i:}^{(k)}$ for the vector comprising the elements from the ith row of $W^{(k)}$, and similarly $W_{:j}^{(k)}$ for that comprising the elements from the jth column. $\zeta^{(K+1)}$ represents the final output of the network (or the logit in the case of classification networks), that is, $\zeta^{(K+1)} = f^w(x)$. We write $W^{(k),L}$ and $W^{(k),U}$ for the lower and upper bound induced by R for $W^{(k)}$, and $b^{(k),L}$ and $b^{(k),U}$ for the bounds of $b^{(k)}$, for $k = 0, \ldots, K$. Observe that $z^{(0)}$, $\zeta_i^{(k+1)}$ and $z_i^{(k)}$ are all functions of the input point x and of the combined vector of weights $w = [W^{(0)}, b^{(0)}, \ldots, W^{(K)}, b^{(K)}]$. We omit the explicit dependency for simplicity of notation. Finally, we remark that, as both the weights and the input vary in a given set, the middle expression of Equation (14) defines a quadratic form.

Interval Bound Propagation (IBP) IBP has already been employed for fast certification of deterministic neural networks [25]. The only adjustment needed in our setting is that, at each layer, we also need to propagate the interval of the weight matrix $[W^{(k),L}, W^{(k),U}]$ and that of the bias vector $[b^{(k),L}, b^{(k),U}]$. This can be done by noticing that the minimum and maximum of each term of the bi-linear form of Equation (14), that is, of each monomial $W_{ij}^{(k)} z_j^{(k)}$, lies in one of the four corners of the interval $[W_{ij}^{(k),L}, W_{ij}^{(k),U}] \times [z_j^{(k),L}, z_j^{(k),U}]$, and by adding the minimum and maximum values respectively attained by $b_i^{(k)}$. As in the deterministic case, interval propagation through the activation function proceeds by observing that generally employed activation functions are monotonic. This is summarised in the following proposition.

Proposition 3. *Let $f^w(x)$ be the network defined by Equation* (14)*, let for $k = 0, \ldots, K$:*

$$t_{ij}^{(k),L} = \min\{W_{ij}^{(k),L} z_j^{(k),L}, W_{ij}^{(k),U} z_j^{(k),L}, W_{ij}^{(k),L} z_j^{(k),U}, W_{ij}^{(k),U} z_j^{(k),U}\} \tag{15}$$

$$t_{ij}^{(k),U} = \max\{W_{ij}^{(k),L} z_j^{(k),L}, W_{ij}^{(k),U} z_j^{(k),L}, W_{ij}^{(k),L} z_j^{(k),U}, W_{ij}^{(k),U} z_j^{(k),U}\} \tag{16}$$

where $i = 1, \ldots, n_{k+1}$, $j = 1, \ldots, n_k$, $z^{(k),L} = \sigma(\zeta^{(k),L})$, $z^{(k),U} = \sigma(\zeta^{(k),U})$ *and*

$$\zeta^{(k+1),L} = \sum_j t_{:j}^{(k),L} + b^{(k),L}, \quad \zeta^{(k+1),U} = \sum_j t_{:j}^{(k),U} + b^{(k),U}. \tag{17}$$

Then we have that $\forall x \in T$ *and* $\forall w \in R$: $f^w(x) = \zeta^{(K+1)} \in \left[\zeta^{(K+1),L}, \zeta^{(K+1),U}\right]$.

The minima and maxima in Proposition 3 are the tightest possible bounds one can compute on matrix multiplication. A more efficient scheme for this propagation is detailed in [50], which can be seen as an adaptation of [41] to NN operations. Additionally, our approach can be linked to abstract interpretation with simultaneous abstract sets (in our case from the orthotope domain) over inputs and weights [22]. Regardless, [37] shows that both have an over-approximation factor of 1.5. Similar bound formulations have been employed across the deterministic NN certification literature [19, 43, 51, 57]. In Appendix C, we employ linear bounds on Equation 17, which can tighten the bounds computed by our method as shown initially in [54]. In [1] dynamic programming is used to tighten these bounds further, and in [43], outside the context of BNNs, an extension of CROWN is developed for the same problem. We emphasise that, regardless of the propagation or tightening employed, each of these approaches can be seen as an instantiation of the framework provided in this work.

Algorithm 1 Lower Bounds for BNN Probabilistic Robustness

Input: T – Input Region, $f^{\mathbf{w}}$ – Bayesian Neural Network, $p(w|\mathcal{D})$ – Posterior Distribution with variance Σ, N – Number of Samples, γ – Weight margin.
Output: A sound lower bound on $P_{\text{safe}}(T, S)$.

$\mathcal{H} \leftarrow \emptyset$ # $\mathcal{H}$ is a set of known safe weight intervals
$v \leftarrow \gamma \cdot I \cdot \Sigma$ # Elementwise product to obtain width of weight margin
for $i \leftarrow 0$ to N **do**
  $w^{(i)} \sim p(w|\mathcal{D})$
  # Assume weight intervals are built to be disjoint
  $[w^{(i),L}, w^{(i),U}] \leftarrow [w_i - v, w_i + v]$
  # Interval/Linear Bound Propagation, Section 5.2
  $y^L, y^U \leftarrow \texttt{Propagate}(f, T, [w^{(i),L}, w^{(i),U}])$
  if $[y^L, y^U] \subset S$ **then**
    $\mathcal{H} \leftarrow \mathcal{H} \bigcup \{[w^{(i),L}, w^{(i),U}]\}$
  end if
end for
$P_{\text{safe}}^L \leftarrow 0.0$
for $[w^{(i),L}, w^{(i),U}] \in \mathcal{H}$ **do**
  $P_{\text{safe}}^L = P_{\text{safe}}^L + P([w^{(i),L}, w^{(i),U}])$ # Compute safe weight probs, Section 5.1
end for
return P_{safe}^L

6 Complete Bounding Algorithm

In this section, we assemble complete algorithms for the computation of bounds on $P_{\text{safe}}(T, S)$ and $D_{\text{safe}}(T, S)$ based on the results discussed so far, leaving the detailed algorithms to Appendix E. Appendix A discusses further use cases for the bounds. The computational complexity of the algorithm is discussed in Appendix F.

6.1 Lower-bounding Algorithm

We provide a step-by-step outline for how to compute lower bounds on $P_{\text{safe}}(T, S)$ in Algorithm 1. We start (line 1) by initialising the family of safe weight sets $\mathcal{H}$ to be the empty set and by scaling the weight margin with the posterior weight scale (line 2). We then iteratively (line 3) proceed by sampling weights from the posterior distribution (line 4), building candidate weight boxes (line 6), and propagating the input and weight box through the BNN (line 8). We next check whether the propagated output set is inside the safe output region S, and, if so, update the family of weights $\mathcal{H}$ to include the weight box currently under consideration (lines 9 and 10). Finally, we rely on the results in Section 5.1 to compute the overall probabilities over all the weight sets in $\mathcal{H}$, yielding a valid lower bound for $P_{\text{safe}}(T, S)$. For clarity of presentation, we assume that all the weight boxes that we sample in lines 4–6 are pairwise disjoint, as this simplifies the probability computation. The general case with overlapping weight boxes relies on the Bonferroni bound and is given in Appendix E.4.

The algorithm for the computation of a lower bound on $D_{\text{safe}}(T, S)$ (listed in the Appendix E as Algorithm 2) proceeds in an analogous way, but without the need to perform the check in line 9, and by adjusting line 15 to the formula from Proposition 2.

6.2 Upper-bounding Algorithm

Upper-bounding $P_{\text{safe}}(T, S)$ and $D_{\text{safe}}(T, S)$ follows the same computational flow as Algorithm 1. The algorithms for the computation of upper bounds on probabilistic and decision robustness are listed respectively as Algorithm 3 and 4 in Appendix E. We again proceed by sampling a rectangle around the weights, propagate bounds through the NN, and compute the probabilities of weight intervals. The key change to the algorithm to allow upper bound computation involves computing the best case, rather than the worst case, for y for decision robustness (line 12 in Algorithm 3) and ensuring that the entire interval $[y^L, y^U] \notin S$ (line 18) for probabilistic robustness.

7 Experiments

In this section we experimentally validate our framework on a variety of tasks, including airborne collision avoidance, medical imaging, and autonomous driving applications. We mainly focus on verifying the adversarial robustness and

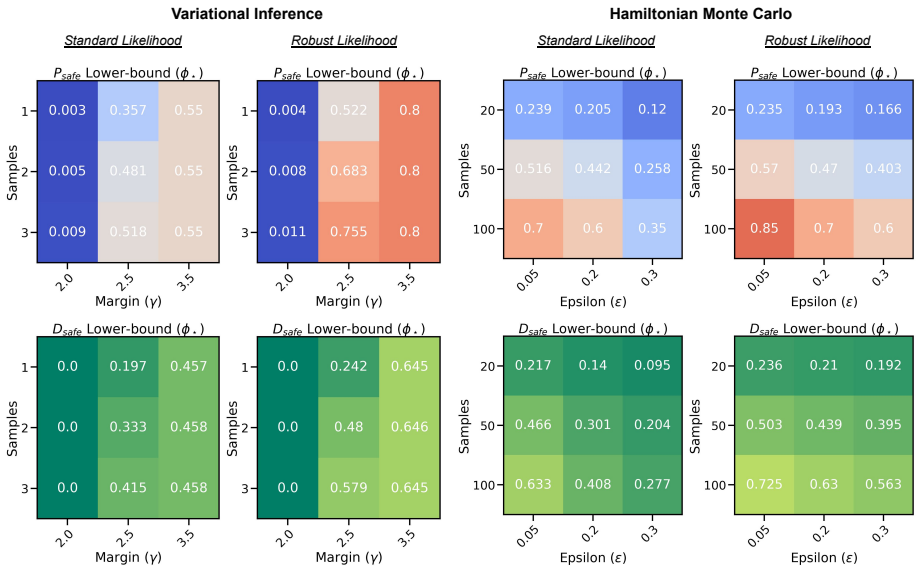

Fig. 2. Top Row: Lower bounds on P_{safe}. **Bottom Row:** Lower bounds on D_{safe}. **Left Two Columns:** Bound values for VI-inferred BNN averaged over 1000 test-set examples using various likelihoods, number of samples, and weight-margin values. **Right Two Columns:** Bound values for HMC-inferred BNN averaged over 1000 test-set examples using various likelihoods, number of samples, and values of ϵ.

uncertainty of classification problems that use the 0-1 loss. For a discussion of how our framework applies to a wider class of specifications see Appendix A, and Appendix B for an extension to other decision rules. In each case study, we take the input set to be the interval $T_\epsilon(x) := [x - \epsilon, x + \epsilon]$, where $\epsilon \geq 0$ is a parameter that we vary in our experiments. For all experiments, S is the set of all vectors where the true class is returned. Experiments are run on a server equipped with 2x AMD EPYC 9334 CPUs and 2x NVIDIA L40 GPUs. Details on training hyper-parameters can be found in Appendix G.

7.1 Airborne Collision Avoidance

We start with the airborne collision avoidance benchmark, which is commonly used to evaluate the robustness of neural network controllers in a safety-critical scenario [27, 28]. In particular, we consider the horizontal collision avoidance scenario (HCAS) from [27], and work with a single hidden layer neural network with 125 hidden neurons trained both using Variational Online Gauss Newton (VOGN) [30] and Hamiltonian Monte Carlo (HMC) [36]. We infer posteriors using both the standard likelihood and the robust likelihood proposed in [53]. In Figure 2 we study the guarantees that our method is able to provide for each combination of the inference method and likelihood. We plot the lower

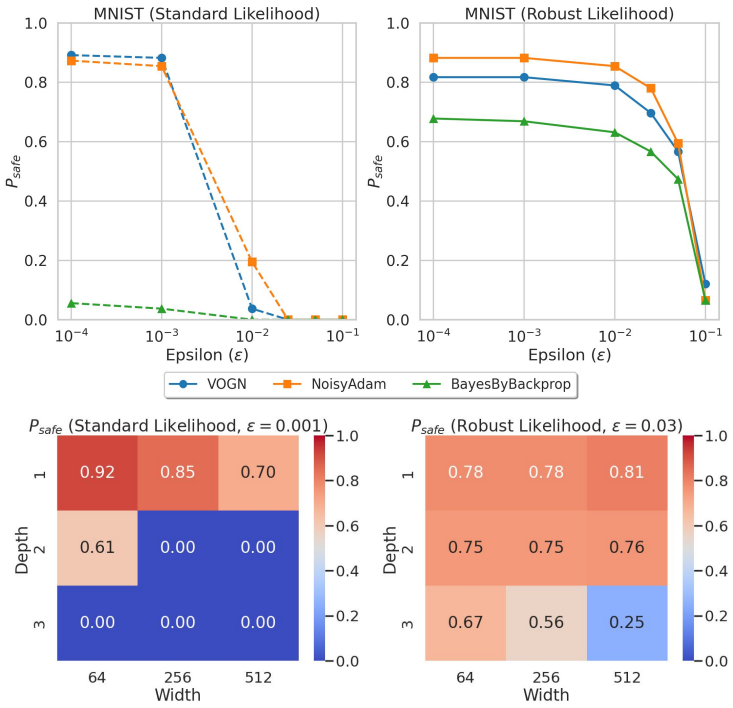

Fig. 3. Top Row: Computed lower bound values on P_{safe} for robust-likelihood VOGN posterior (right) and standard VOGN posterior (left). **Bottom Row:** Computed lower bound P_{safe} values for the VOGN posterior while varying depth and width parameters of the BNN architecture.

bound on P_{safe} and D_{safe} resulting from Algorithm 1 averaged over 1000 test-set samples. In each plot we show the effect of varying the critical parameters of our algorithm, including the number of samples and, for VOGN, the width of the weight margin γ, as defined in Section 5. As expected, in all cases, we find that taking more samples and using a higher weight margin consistently yields a higher lower bound. HMC requires significantly more samples to cover the probability mass as there is no margin parameter when certifying probability mass functions, i.e., probability distributions with discrete support. Thus, each sample covers a fixed, small amount of mass, while even one sample from the VOGN posterior, with a suitable weight margin, is able to give non-trivial lower bounds, e.g., 0.8 in the case of a P_{safe} lower bound for the the robust likelihood BNN in Figure 2. The fact that higher ϵ values lead to smaller values of the lower bound is also expected, as larger ϵ implies a greater radius for the initial set T.

7.2 Image Classification

We now turn our attention to image classification, considering first the widely used MNIST benchmark with 28 by 28 pixel grey-scale images [32] and then two safety-critical tasks from medical image classification and autonomous driving.

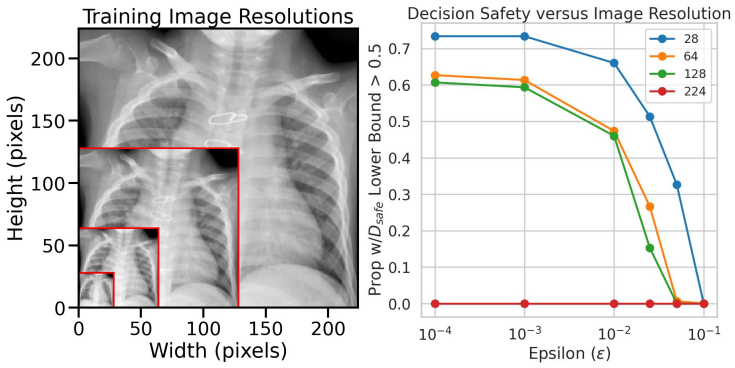

Fig. 4. Left: Different training image resolutions on a training image sample from PneumoniaMNIST. **Right:** Our computed lower bounds on D_{safe}, which correspond to adversarial robustness certificates as we vary the resolution fed into a VOGN-inferred BNN.

MNIST Digit Recognition In Figure 3, we present two plots certifying (via lower bounds on P_{safe}) a single hidden layer neural network with 100 hidden neurons with parameters inferred using VOGN [30], BayesByBackprop [10] and NoisyAdam [62], using both robust and standard likelihoods as for the airborne collision avoidance case study. In the top row of Figure 3, we plot the computed lower bounds as we increase the value of ϵ. For the posterior inferred by each inference method using the standard likelihood, we observe that our method is only able to certify low values of P_{safe}, even for small values of ϵ, e.g., 0.001. However, for the robust likelihood posteriors, we are able to certify non-trival robustness guarantees even at $\epsilon = 0.1$. Additionally, we observe that BayesByBackprop [10] has consistently lower certified values of P_{safe}. We hypothesise that this is due to BayesByBackprop having a higher variance posterior, which in turn results in the propagation of wider weight intervals that can introduce significant approximation.

In the bottom half of Figure 3, we study how our lower bounds on P_{safe} change as we increase the depth and width of the neural network architecture. For this study we exclusively employ VOGN, but, as previously, still utilise the standard (left) and robust (right) likelihoods. We find that, for the standard likelihood, we are able to obtain high lower bounds (greater than 0.7) for all one-layer networks regardless of width, but struggle with increasing depths. For the

posteriors inferred using the robust likelihood, we observe that the lower bounds produced by our approach only begin to decrease when the depth reaches three layers with significant width. We additionally highlight that, for the posteriors inferred using the robust likelihood, we use a much larger ϵ (=0.03) compared to what is used to get non-trivial bounds in the standard training case ($\epsilon = 0.001$).

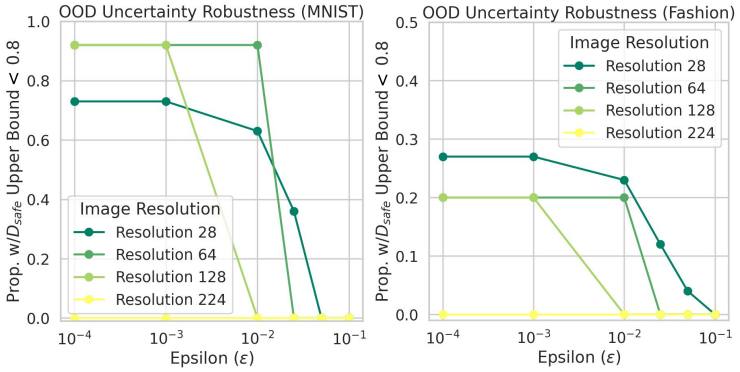

Fig. 5. Computing upper bounds on D_{safe} to certify robust uncertainty estimates from posteriors inferred on PneumoniaMNIST. **Left:** Uncertainty certificates for PneumoniaMNIST posterior on MNIST dataset. **Right:** Uncertainty certificates for PneumoniaMNIST posterior on FashionMNIST dataset.

Medical Image Classification We now turn our attention to a more realistic safety-critical application from the medical image classification domain. In particular, we study the PneumoniaMNIST dataset from the MedMNIST suite of benchmarks [60]. PneumoniaMNIST is a dataset of greyscale images of chest X-rays that pose a binary classification problem, with one class representing normal chest X-rays and the other class presenting with pneumonia. In the most recent iteration of the MedMNIST benchmark, an option for different resolutions is provided ranging from 28 by 28, the same resolution as MNIST, up to 224 by 224, the same resolution as the popular, large-scale ImageNet dataset [18]. In the left-hand-side plot of Figure 4, we visualize the significant differences between these input dimensionalities. We use these datasets to study how well our certification approaches scale with increasing input dimensionality. We work with a four-layer convolutional architecture with two 2D convolution layers, an average pooling layer, and a final fully-connected layer consisting of 50 neurons. For each network studied in this section, we use the robust likelihood of [53] in order to obtain non-trivial certifications. Additionally, we turn our attention to bounding decision robustness, D_{safe}, rather than probabilistic robustness, P_{safe}, employed for MNIST evaluation. Decision robustness is more appropriate here due to the safety-critical nature of pneumonia classification, compared to hand-

written digit classification. In particular, we begin by computing lower bounds on D_{safe}, which in turn allows us to compute adversarial robustness certificates commensurate with those computed for deterministic neural networks. We find (see the right-hand-side plot of Figure 4) that an increase in resolution corresponds to a significant decrease in the lower bounds computed by our approach, which is a result of greater approximation introduced by bound propagation techniques. Nevertheless, on images with 128 by 128 resolution, our guarantees continue to provide non-trivial bounds.

In addition to computing lower bounds on D_{safe} to certify the adversarial robustness of our trained posteriors, we also compute upper bounds on D_{safe} to provide certificates that our posterior is provably, robustly uncertain on given out-of-distribution inputs. To study this, we use the MNIST dataset as well as the FashionMNIST dataset (consisting of greyscale, 28 by 28, images of clothing items) as out-of-distribution examples for pneumonia classification. We then consider an example *uncertain* if the maximum value of the posterior predictive distribution is less than 0.8 (an arbitrary, user-definable threshold, which may require calibration to the specific setting). In Figure 5 we plot the proportion of test-set inputs for which the inferred posterior is robustly uncertain on MNIST (left plot) and FashionMNIST (right plot). For very small values of ϵ, we notice that the network is much more robustly uncertain on MNIST examples then on FashionMNIST examples. Further, we find that, similarly to robustness certification, we are unable to certify any non-trivial uncertainty properties for images with 224 by 224 resolution.

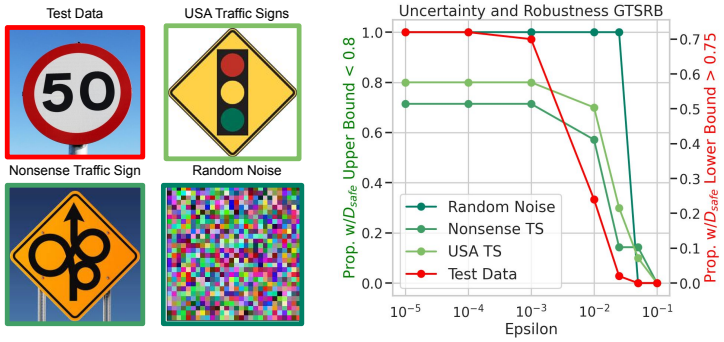

Fig. 6. Analysis of BNN inferred on GTSRB dataset. **Left:** Example in-distribution image (top left) and out-of-distribution images. **Right:** Adversarial robustness certificates (red) and uncertainty certificates (shades of green) using lower and upper bounds on D_{safe} respectively for different levels of ϵ.

Traffic Sign Recognition Classification Our final safety-critical case study comes from autonomous navigation using the German Traffic Sign Recognition

Benchmark (GTSRB) [44]. In particular, we study a three-class subset of the GTSRB dataset with a three-layer CNN model with parameters inferred using the robust likelihood and VOGN. In Figure 6 we plot an example of the 50 km/h sign (an in-distribution image) and different examples from three different out-of-distribution datasets: United States Traffic Signs, Nonsense Traffic Signs, and random noise. The first two are small sets of images curated from royalty free image databases online and the third is sampled from a unit normal distribution. Using each of these datasets, we study both adversarial robustness (ensuring a sufficiently high D_{safe} lower bound) and uncertainty properties (ensuring sufficiently low D_{safe} upper bound) of the trained network that achieves 96% test-set accuracy. In the right-hand-side plot of Figure 6 (in red), we show that our method is able to compute non-trivial adversarial robustness guarantees up to $\epsilon = 0.001$. In various shades of green, we show that the uncertainty guarantees we compute are also non-trivial for similar values of ϵ.

8 Conclusion

In this work, we introduced a computational framework for evaluating robustness properties of BNNs operating under adversarial settings. In particular, we have discussed how probabilistic robustness and decision robustness can be upper- and lower-bounded via a combination of posterior sampling, integral computation over boxes and bound propagation techniques. We have detailed how to compute these properties for the case of HMC and VI posterior approximation, and how to instantiate the bounds for interval and linear bound propagation techniques. We emphasise that the framework presented is general and can be adapted to different inference techniques, and to most of the verification techniques employed for deterministic neural networks. The main limitation of the approach presented here arises directly from the Bayesian nature of the underlying model, i.e., the need to bound and partition at the weight space level (which is not needed for deterministic neural networks, with the weight fixed to a specific value). Nevertheless, the methods presented here provide the first general-purpose, formal technique for the verification of probabilistic and decision robustness, as well as uncertainty quantification, in Bayesian neural networks, systematically evaluated on a range of tasks and network architectures. We hope this can serve as a sound basis for future practical applications in safety-critical scenarios.

Acknowledgments. This project received funding from the ERC under the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No. 834115). MK further acknowledges funding from ELSA: European Lighthouse on Secure and Safe AI project (grant agreement No. 101070617 under UK guarantee). Preliminary work on this paper was done while Matthew Wicker, Andrea Patane and Luca Laurenti were at the University of Oxford funded by FUN2MODEL.

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

References

1. Adams, S., Patane, A., Lahijanian, M., Laurenti, L.: BNN-DP: robustness certification of Bayesian neural networks via dynamic programming. In: ICML. pp. 133–151. PMLR (2023)
2. Aggarwal, R., Sounderajah, V., Martin, G., Ting, D.S., Karthikesalingam, A., King, D., Ashrafian, H., Darzi, A.: Diagnostic accuracy of deep learning in medical imaging: A systematic review and meta-analysis. NPJ digital medicine **4**(1), 1–23 (2021)
3. Batten, B., Hosseini, M., Lomuscio, A.: Tight verification of probabilistic robustness in Bayesian neural networks. In: AISTATS (2024)
4. Bekasov, A., Murray, I.: Bayesian adversarial spheres: Bayesian inference and adversarial examples in a noiseless setting. arXiv preprint arXiv:1811.12335 (2018)
5. Benussi, E., Patane, A., Wicker, M., Laurenti, L., Kwiatkowska, M.: Individual fairness guarantees for neural networks. In: IJCAI (2022)
6. Berger, J.O.: Statistical decision theory and Bayesian analysis. Springer Science & Business Media (2013)
7. Berrada, L., Dathathri, S., Dvijotham, K., Stanforth, R., Bunel, R.R., Uesato, J., Gowal, S., Kumar, M.P.: Make sure you're unsure: A framework for verifying probabilistic specifications. In: NeurIPS. vol. 34 (2021)
8. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition **84**, 317–331 (2018)
9. Bishop, C.: Neural networks for pattern recognition. Oxford University Press, USA (1995)
10. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural networks. In: ICML (2015)
11. Bonferroni, C.: Teoria statistica delle classi e calcolo delle probabilita. Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze **8**, 3–62 (1936)
12. Carbone, G., Wicker, M., Laurenti, L., Patane, A., Bortolussi, L., Sanguinetti, G.: Robustness of Bayesian neural networks to gradient-based attacks. In: NeurIPS. vol. 33, pp. 15602–15613 (2020)
13. Cardelli, L., Kwiatkowska, M., Laurenti, L., Paoletti, N., Patane, A., Wicker, M.: Statistical guarantees for the robustness of Bayesian neural networks. In: IJCAI (2019)
14. Cardelli, L., Kwiatkowska, M., Laurenti, L., Patane, A.: Robustness guarantees for Bayesian inference with Gaussian processes. In: AAAI (2018)
15. Chang, S.H., Cosman, P.C., Milstein, L.B.: Chernoff-type bounds for the Gaussian error function. IEEE Transactions on Communications **59**(11), 2939–2944 (2011)
16. Chen, L., Lin, S., Lu, X., Cao, D., Wu, H., Guo, C., Liu, C., Wang, F.Y.: Deep neural network based vehicle and pedestrian detection for autonomous driving: a survey. IEEE Transactions on Intelligent Transportation Systems **22**(6), 3234–3246 (2021)
17. De Palma, G., Kiani, B., Lloyd, S.: Adversarial robustness guarantees for random deep neural networks. In: ICML. pp. 2522–2534. PMLR (2021)
18. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
19. Doherty, A., Wicker, M., Laurenti, L., Patane, A.: Individual fairness in Bayesian neural networks. arXiv preprint arXiv:2304.10828 (2023)

20. Dvijotham, K., Garnelo, M., Fawzi, A., Kohli, P.: Verification of deep probabilistic models. arXiv preprint arXiv:1812.02795 (2018)
21. Gal, Y.: Uncertainty in deep learning. Ph.D. thesis, University of Cambridge (2016)
22. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: Ai2: Safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE S&P. pp. 3–18. IEEE (2018)
23. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
24. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
25. Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., Kohli, P.: On the effectiveness of interval bound propagation for training verifiably robust models. In: SecML 2018 (2018)
26. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: CAV. pp. 3–29. Springer (2017)
27. Julian, K.D., Kochenderfer, M.J.: Guaranteeing safety for neural network-based aircraft collision avoidance systems. DASC (2019)
28. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: CAV (2017)
29. Kendall, A., Gal, Y.: What uncertainties do we need in Bayesian deep learning for computer vision? In: NeurIPS (2017)
30. Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., Srivastava, A.: Fast and scalable Bayesian deep learning by weight-perturbation in adam. In: ICML. pp. 2611–2620. PMLR (2018)
31. Lechner, M., Žikelić, D., Chatterjee, K., Henzinger, T.: Infinite time horizon safety of Bayesian neural networks. In: NeurIPS. vol. 34, pp. 10171–10185 (2021)
32. LeCun, Y.: The MNIST database of handwritten digits (1998)
33. Liu, X., Li, Y., Wu, C., Hsieh, C.J.: Adv-BNN: Improved adversarial defense through robust Bayesian neural network. In: ICLR (2019)
34. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I convex underestimating problems. Mathematical programming pp. 147–175 (1976)
35. Michelmore, R., Wicker, M., Laurenti, L., Cardelli, L., Gal, Y., Kwiatkowska, M.: Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In: ICRA (2019)
36. Neal, R.M.: Bayesian learning for neural networks. Springer Science & Business Media (2012)
37. Nguyen, H.D.: Efficient implementation of interval matrix multiplication. In: PARA. pp. 179–188. Springer (2012)
38. Nix, D.A., Weigend, A.S.: Estimating the mean and variance of the target probability distribution. In: ICNN. vol. 1, pp. 55–60. IEEE (1994)
39. Patane, A., Blaas, A., Laurenti, L., Cardelli, L., Roberts, S., Kwiatkowska, M.: Adversarial robustness guarantees for Gaussian processes. Journal of Machine Learning Research **23** (2022)
40. Raghunathan, A., Steinhardt, J., Liang, P.S.: Semidefinite relaxations for certifying robustness to adversarial examples. In: NeurIPS. vol. 31 (2018)
41. Rump, S.M.: Fast and parallel interval arithmetic. BIT Numerical Mathematics **39**, 534–554 (1999)
42. Smith, M.T., Grosse, K., Backes, M., Alvarez, M.A.: Adversarial vulnerability bounds for Gaussian process classification. arXiv preprint arXiv:1909.08864 (2019)
43. Sosnin, P., Müller, M., Baader, M., Tsay, C., Wicker, M.: Certified robustness to data poisoning in gradient-based training. arXiv preprint arXiv:2406.05670 (2024)

44. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural networks **32**, 323–332 (2012)
45. Stanforth, R., Gowal, S., Mann, T., Kohli, P., et al.: A dual approach to scalable verification of deep networks. arXiv preprint arXiv:1803.06567 (2018)
46. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: ICLR (2014)
47. Tjeng, V., Xiao, K., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. arXiv preprint arXiv:1711.07356 (2017)
48. Wei, T., Liu, C.: Safe control with neural network dynamic models. In: Learning for Dynamics and Control Conference. pp. 739–750. PMLR (2022)
49. Weng, T.W., Zhang, H., Chen, H., Song, Z., Hsieh, C.J., Boning, D., Dhillon, I.S., Daniel, L.: Towards fast computation of certified robustness for relu networks. In: ICML (2018)
50. Wicker, M.: Adversarial robustness of Bayesian neural networks. Ph.D. thesis, University of Oxford (2021)
51. Wicker, M., Heo, J., Costabello, L., Weller, A.: Robust explanation constraints for neural networks. arXiv preprint arXiv:2212.08507 (2022)
52. Wicker, M., Huang, X., Kwiatkowska, M.: Feature-guided black-box safety testing of deep neural networks. In: TACAS. pp. 408–426. Springer (2018)
53. Wicker, M., Laurenti, L., Patane, A., Chen, Z., Zhang, Z., Kwiatkowska, M.: Bayesian inference with certifiable adversarial robustness. In: AISTATS. pp. 2431–2439. PMLR (2021)
54. Wicker, M., Laurenti, L., Patane, A., Kwiatkowska, M.: Probabilistic safety for Bayesian neural networks. In: UAI. pp. 1198–1207. PMLR (2020)
55. Wicker, M., Laurenti, L., Patane, A., Paoletti, N., Abate, A., Kwiatkowska, M.: Certification of iterative predictions in Bayesian neural networks. In: UAI. pp. 1713–1723. PMLR (2021)
56. Wicker, M., Laurenti, L., Patane, A., Paoletti, N., Abate, A., Kwiatkowska, M.: Probabilistic reach-avoid for Bayesian neural networks. Artificial Intelligence (2024)
57. Wicker, M., Sosnin, P., Janik, A., Müller, M., Weller, A., Tsay, C., Tsay, C.: Certificates of differential privacy and unlearning for gradient-based training. arXiv preprint arXiv:2406.13433 (2024)
58. Wong, E., Kolter, Z.: Provable defenses against adversarial examples via the convex outer adversarial polytope. In: ICML. pp. 5286–5295. PMLR (2018)
59. Wu, M., Wicker, M., Ruan, W., Huang, X., Kwiatkowska, M.: A game-based approximate verification of deep neural networks with provable guarantees. Theoretical Computer Science **807**, 298–329 (2020)
60. Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., Ni, B.: MedMNIST v2-a large-scale lightweight benchmark for 2D and 3D biomedical image classification. Scientific Data **10**(1), 41 (2023)
61. Yuan, M., Wicker, M., Laurenti, L.: Gradient-free adversarial attacks for Bayesian neural networks. In: AABI (2020)
62. Zhang, G., Sun, S., Duvenaud, D., Grosse, R.: Noisy natural gradient as variational inference. In: ICML. pp. 5852–5861. PMLR (2018)
63. Zhang, H., Weng, T.W., Chen, P.Y., Hsieh, C.J., Daniel, L.: Efficient neural network robustness certification with general activation functions. In: NeurIPS. pp. 4939–4948 (2018)
64. Zhang, X., Wang, B., Kwiatkowska, M.: Provable preimage under-approximation for neural networks. In: TACAS. pp. 3–23. Springer (2024)

Appendix

This appendix provides additional details of the proposed robustness evaluation methodology, bounding algorithms, proofs of the main results, as well as hyper-parameters and network architectures for reproducing our experiments.

A Use Cases for Lower and Upper Bounds

In this section we expand on the methodology for robustness certification supported by the bounding algorithms. Specifically, we present the use cases for each bound we derive in this paper and highlight their importance. We also discuss various kinds of uncertainty measures (full definitions and discussion in [21]), as well as how one can obtain certification on these quantities. A summary of the suggested use cases is given in Table 1.

Table 1. Use cases for each lower and upper bound presented in this paper.

Property	*Application*	*Bound for Certification*
Correctness	Classification	Lower & Upper on $D_{\text{safe}}(T, S)$
Aleatoric Uncert.	Classification	Upper on $D_{\text{safe}}(T, S)$
Epistemic Uncert. (OOD)	Classification	Lower & Upper on $P_{\text{safe}}(T, S)$
Correctness	Regression	Lower & Upper on $D_{\text{safe}}(T, S)$
Aleatoric Uncert.	Regression	Lower & Upper on $D_{\text{safe}}(T, S)$
Epistemic Uncert. (OOD)	Regression	Lower & Upper on $P_{\text{safe}}(T, S)$

Correctness One of the most widely studied properties in NN robustness is that of "correctness" [28], which requires prediction of the NN to match the ground truth even in the face of adversarial perturbations. For classification, as discussed in the main text, correctness boils down to checking that, for all adversarial perturbations, the argmax of the softmax output remains the same. For regression, due to the continuous nature of outputs, correctness involves establishing a range of outputs that correspond to the tolerable error. Given that correctness relies on the ultimate decision of the BNN in either the classification or regression, we use upper and lower bounds on the posterior predictive expectation (i.e., D^L_{safe} and D^U_{safe}). To prove that classification is correct, one must prove that the lower bound of the true class softmax probability is higher than the upper bound of all other classes softmax probability, which implies:

$$\forall x \in T, \quad \arg\max \mathbb{E}\big[\sigma(f^w(x))\big] = c$$

For regression one must use upper and lower bounds in order to show that output prediction lies within tolerable error. For this, one needs to check that the end points of the decision, $[D^L_{\text{safe}}, D^L_{\text{safe}}]$, are contained within the tolerance region.

Aleatoric Uncertainty Measures of aleatoric uncertainty are input-dependent and come from the noise within the data observation process [21]. For classification, the aleatoric uncertainty is usually measured as $\max_{i\in[n]} \mathbb{E}_{p(w|\mathcal{D})}\sigma(f^w(x))_i$. This is also termed the 'confidence.' For regression, one can predict both the mean and variance of a Gaussian likelihood, where the variance represents the aleatoric uncertainty [38]. Computing bounds on the posterior predictive mean allows us to ensure that a point has sufficiently high (or low) aleatoric uncertainty. For classification, D^L_{safe} represents a lower bound on $\mathbb{E}_{p(w|\mathcal{D})}\sigma(f^w(x))_i$, and thus D^L_{safe} allows us to bound aleatoric uncertainty. For regression, the same holds, except it is only the bound D^L_{safe} in the dimension corresponding to the predicted variance.

Epistemic Uncertainty Model or epistemic uncertainty measures the uncertainty from the lack of data at training time. We expect that epistemic uncertainty is high for out-of-distribution samples. Epistemic uncertainty is measured as the spread of prediction from various models under the posterior distribution. To measure this, it is natural to consider the variance of the posterior predictive distribution. Given an out-of-distribution input x, one can certify that P_{safe} is not sufficiently high for any class. This guarantees that there does not exist one class that the BNN maps all of its predictions into, and thus guarantees that the BNN is uncertain. By checking P_{safe} across in- and out-of-distribution points, modellers can certify that their BNN is well calibrated with respect to epistemic uncertainty.

B Certifying Further Decision Rules

As discussed in the main paper, decision robustness is clearly dependent on the function used for Bayesian decisions for the given learning model. In the main paper we have given explicit results for the two standard losses, ℓ_{0-1} for classification and ℓ_2 for regression. However, with some minor adjustments, our method can be employed for different losses too. In this section we give the details for the ℓ_1 loss for regression and weighted loss for classification.

B.1 Bounding Decisions for the ℓ_1 Loss

For the ℓ_1 decision loss, it is known that the *median* of the posterior predictive distribution is the value that minimizes the loss. Thus, we must bound the median, defined as usual to be $m(Z) := x \iff \int_{-\infty}^{x} p_Z(v)dv = 0.5$. Assuming $\sum_{i=1}^{N} P(J_i) = 1.0$, we can arrive at a lower bound by picking y_m^L to be our median lower bound such that $\sum_{i=1}^{m} P(J_i) \leq 0.5$ but $\sum_{i=1}^{m+1} P(J_i) \geq 0.5$. One can similarly find an upper bound via this routine by first computing upper bounds for each weight rectangle and then picking y_m^U such that $\sum_{i=1}^{m} P(J_i) \geq 0.5$ but $\sum_{i=0}^{m-1} P(J_i) \leq 0.5$. When the condition $\sum_{i=1}^{N} P(J_i) = 1.0$ does not hold, we can modify the procedure to obtain valid bounds on the median. We assume

that $\sum_{i=1}^{N} P(J_i) = 1.0 - \eta$ for any η such that $0.5 > \eta > 0$. Then we pick the lower bound to the median to be y_m^L such that $\eta + \sum_{i=1}^{m} P(J_i) \leq 0.5$ but $\eta + \sum_{i=1}^{m+1} P(J_i) \geq 0.5$. This yields a valid bound on the median. Similar formulas can be computed for the upper bound, by relying on the laws of complementary probabilities.

B.2 Bounding Decisions for the 0-K Loss

In some safety-critical decision-making problems, particularly in medical diagnosis, predicting one class is associated with with greater risk (formally, loss) than predicting another. In this case, the 0-1 loss is made more general and is defined as the 0-K loss, which assigns a penalty of 0 to the correct prediction, and K_i otherwise, where i indexes the classes. Thus, the posterior expected losses in a binary classification case are $K_0 p(y_0|x, \mathcal{D})$ and $K_1 p(y_1|x, \mathcal{D})$. In this scenario the decision rule is not to take the argmax as before, but to predict class i if the $p(y_i|x, \mathcal{D}) > \frac{K_i}{\sum_{i=0}^{n_c} K_i}$, which is straightforward in our framework. To certify this decision rule it suffices to check that $D^L_{\text{safe},i} \geq \frac{K_i}{\sum_{i=0}^{n_c} K_i}$. We refer interested readers to Section 4.4.3 of [6] for more in-depth discussion.

C Linear bound propagation (LBP)

We now discuss how LBP can be used to lower-bound the BNN output over T and R as an alternative to IBP. In LBP, instead of propagating bounding boxes, one finds lower and upper linear bounding functions (LBFs) for each layer and then propagates them through the network. As the bounding function has an extra degree of freedom w.r.t. the bounding boxes obtained through IBP, LBP usually yields tighter bounds, though at an increased computational cost. Since in deterministic networks non-linearity comes only from the activation functions, in the deterministic case LBFs are computed by bounding the activation functions and propagating the bounds through the affine function that defines each layer.

In our setting, given T in the input space and R for the first layer in the weight space, we start with the observation that LBFs can be obtained and propagated through commonly employed activation functions as discussed in [63].

Lemma 2. *Let $f^w(x)$ be defined by Equation (14). For each hidden layer $k = 1, \ldots, K$, consider a bounding box in the pre-activation function, i.e. such that $\zeta_i^{(k)} \in [\zeta_i^{(k),L}, \zeta_i^{(k),U}]$ for $i = 1, \ldots, n_k$. Then there exist coefficients $\alpha_i^{(k),L}$, $\beta_i^{(k),L}$, $\alpha_i^{(k),U}$ and $\beta_i^{(k),U}$ of lower and upper LBFs on the activation function such that for all $\zeta_i^{(k)} \in [\zeta_i^{(k),L}, \zeta_i^{(k),U}]$ it holds that: $\alpha_i^{(k),L}\zeta_i^{(k)} + \beta_i^{(k),L} \leq \sigma(\zeta_i^{(k)}) \leq \alpha_i^{(k),U}\zeta_i^{(k)} + \beta_i^{(k),U}$.*

The lower and upper LBFs can thus be minimised and maximised to propagate the bounds of $\zeta^{(k)}$ in order to compute a bounding interval $[z^{(k),L}, z^{(k),U}]$ for $z^{(k)} = \sigma(\zeta^{(k)})$. Then, LBFs for the monomials of the bi-linear form of Equation (14) can be derived using McCormick's inequalities [34]:

$$W_{ij}^{(k)} z_j^{(k)} \geq W_{ij}^{(k),L} z_j^{(k)} + W_{ij}^{(k)} z_j^{(k),L} - W_{ij}^{(k),L} z_j^{(k),L} \tag{18}$$

$$W_{ij}^{(k)} z_j^{(k)} \leq W_{ij}^{(k),U} z_j^{(k)} + W_{ij}^{(k)} z_j^{(k),L} - W_{ij}^{(k),U} z_j^{(k),L} \tag{19}$$

for every $i = 1, \ldots, n_k$, $j = 1, \ldots, n_{k-1}$ and $k = 1, \ldots, K$. The final linear bound can be obtained by iterating the application of Lemma 2 and Equations (18)–(19) through every layer. This is summarised in the following proposition, which is proved in Appendix D along with an explicit construction of the LBFs.

Proposition 4. *Let $f^w(x)$ be the network defined by Equation (14). Then for every $k = 0, \ldots, K$ there exists lower and upper LBFs on the pre-activation function of the form:*

$$\zeta_i^{(k+1)} \geq \mu_i^{(k+1),L} \cdot x + \sum_{l=0}^{k-1} \langle \nu_i^{(l,k+1),L}, W^{(l)} \rangle + \nu_i^{(k,k+1),L} \cdot W_{i:}^{(k)} + \lambda_i^{(k+1),L}$$

$$\zeta_i^{(k+1)} \leq \mu_i^{(k+1),U} \cdot x + \sum_{l=0}^{k-1} \langle \nu_i^{(l,k+1),U}, W^{(l)} \rangle + \nu_i^{(k-1,k+1),U} \cdot W_{i:}^{(k)} + \lambda_i^{(k+1),U}$$

for $i = 1, \ldots, n_{k+1}$, where $\langle \cdot, \cdot \rangle$ is the Frobenius product between matrices, $\cdot$ is the dot product between vectors, and the explicit formulas for the LBF coefficients, i.e., $\mu_i^{(k+1),L}$, $\nu_i^{(l,k+1),L}$, $\lambda_i^{(k+1),L}$, $\mu_i^{(k+1),U}$, $\nu_i^{(l,k+1),U}$, are given in Appendix D.4. Let $\zeta_i^{(k),L}$ and $\zeta_i^{(k),U}$, respectively, be the minimum and the maximum of the right-hand side of the two equations above; then we have that $\forall x \in T$ and $\forall w \in R$: $f^w(x) = \zeta^{(K+1)} \in \left[\zeta^{(K+1),L}, \zeta^{(K+1),U}\right]$.

D Proofs

In this section we provide proofs for the main theoretical results stated in the paper.

D.1 Lemma 1

Proof. By the definition of the maximal safe weight set we have $w \in H \iff \forall x \in T, f^w(x) \in S$. Moreover, we have that the probability of a weight being in such a set is given as $Prob_{w \sim p(w|\mathcal{D})}(w \in H) = \int_H p(w|\mathcal{D}) dw$. By making explicit the definition of H, together these two give us

$$Prob_{w \sim p(w|\mathcal{D})}(\forall x \in T, f^w(x) \in S) = \int_H p(w|\mathcal{D}) dw.$$

The second equality stated in the lemma follows directly from the latter result and the property of complementary probabilities, with $w \in H$ and $w \in K$ being two complementary events.

D.2 Proposition 2

Proof. We prove the results explicitly for the lower bound; the derivation of the upper bound is analogous. Consider the minimisation over T of the expected value computed over the posterior distribution of Problem 2 for output index $c \in \{1, \ldots, m\}$:

$$\min_{x \in T} \mathbb{E}_{p(w|\mathcal{D})}[\sigma_c(f^w(x))] = \min_{x \in T} \int \sigma_c(f^w(x)) p(w|\mathcal{D}) dw.$$

Let $I = \mathbb{R}^{n_w} \setminus \bigcup_{i=1}^{n_J} J_i$ Since the weight intervals in $\mathcal{J}$ are disjointed we can rely on the linearity of integrals to obtain:

$$\min_{x \in T} \int \sigma_c(f^w(x)) p(w|\mathcal{D}) dw = \min_{x \in T} \left(\sum_{i=1}^{n_J} \int_{J_i} \sigma_c(f^w(x)) p(w|\mathcal{D}) \right.$$
$$\left. + \int_I \sigma_c(f^w(x)) p(w|\mathcal{D}) \right).$$

We notice that, for every x, $\int_{J_i} \sigma_c(f^w(x)) p(w|\mathcal{D}) \geq \min_{w \in J_i} \sigma_c(f^w(x)) \int_{J_i} p(w|\mathcal{D})$. By combining this result with the above chain of equalities, and further relying on the property of minimum, we obtain that:

$$\min_{x \in T} \left(\sum_{i=1}^{n_J} \int_{J_i} \sigma_c(f^w(x)) p(w|\mathcal{D}) + \int_I \sigma_c(f^w(x)) p(w|\mathcal{D}) \right) \geq$$
$$\sum_{i=1}^{n_J} \int_{J_i} p(w|\mathcal{D}) dw \min_{\substack{x \in T \\ w \in J_i}} \sigma_c(f^w(x)) +$$
$$\sigma^L \left(1 - \sum_{i=1}^{n_J} \int_{J_i} p(w|\mathcal{D}) dw \right) = D^L_{\text{safe},c},$$

which proves the theorem statement.

D.3 Proposition 3

Proof. The bounding box can be computed iteratively in the number of hidden layers of the network, K. We show how to compute the lower bound of the bounding box; the computation for the maximum is analogous.

Consider the k-th network layer, for $k = 0, \ldots, K$, we want to find for $i = 1, \ldots n_{k+1}$:

$$\min_{\substack{W^{(k)}_{i:} \in [W^{(k),L}_{i:}, W^{(k),U}_{i:}] \\ z^{(k)} \in [z^{(k),L}, z^{(k),U}] \\ b^{(k)}_i \in [b^{(k),L}_i, b^{(k),U}_i]}} z^{(k+1)}_i = \sigma \left(\sum_{j=1}^{n_k} W^{(k)}_{ij} z^{(k)}_j + b^{(k)}_i \right).$$

As the activation function σ is monotonic, it suffices to find the minimum of: $\sum_{j=1}^{n_k} W_{ij}^{(k)} z_j^{(k)} + b_i^{(k)}$. Since $W_{ij}^{(k)} z_j^{(k)}$ is a bi-linear form defined on an hyper-rectangle, it follows that it obtains its minimum in one of the four corners of the rectangle $[W_{ij}^{(k),L}, W_{ij}^{(k),U}] \times [z_j^{(k),L}, z_j^{(k),U}]$.
Let $t_{ij}^{(k),L} = \min\{W_{ij}^{(k),L} z_j^{(k),L}, W_{ij}^{(k),U} z_j^{(k),L}, W_{ij}^{(k),L} z_j^{(k),U}, W_{ij}^{(k),U} z_j^{(k),U}\}$ we hence have:

$$\sum_{j=1}^{n_k} W_{ij}^{(k)} z_j^{(k)} + b_i^{(k)} \geq \sum_{j=1}^{n_k} t_{ij}^{(k),L} + b_i^{(k),L} =: \zeta_i^{(k+1),L}.$$

Thus for every $W_{i:}^{(k)} \in [W_{i:}^{(k),L}, W_{i:}^{(k),U}]$, $z^{(k)} \in [z^{(k),L}, z^{(k),U}]$ and $b_i^{(k)} \in [b_i^{(k),L}, b_i^{(k),U}]$ we have:

$$\sigma\left(\sum_{j=1}^{n_k} W_{ij}^{(k)} z_j^{(k)} + b_i^{(k)}\right) \geq \sigma\left(\zeta_i^{(k+1),L}\right)$$

that is $z_i^{(k+1),L} = \sigma\left(\zeta_i^{(k+1),L}\right)$ is a lower bound to the solution of the minimisation problem posed above.

D.4 Proposition 4

We first state the following lemma that follows directly from the definition of linear functions:

Lemma 3. *Let $f^L(t) = \sum_j a_j^L t_j + b^L$ and $f^U(t) = \sum_j a_j^U t_j + b^U$ be lower and upper LBFs to a function $g(t)$ $\forall t \in \mathcal{T}$, i.e., $f^L(t) \leq g(t) \leq f^U(t)$ $\forall t \in \mathcal{T}$. Consider two real coefficients $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$. Define*

$$\bar{a}_j^L = \begin{cases} \alpha a_j^L \ if \alpha \geq 0 \\ \alpha a_j^U \ if \alpha < 0 \end{cases} \quad \bar{b}^L = \begin{cases} \alpha b^L + \beta \ if \alpha \geq 0 \\ \alpha b^U + \beta \ if \alpha < 0 \end{cases} \tag{20}$$

$$\bar{a}_j^U = \begin{cases} \alpha a_j^U \ if \alpha \geq 0 \\ \alpha a_j^L \ if \alpha < 0 \end{cases} \quad \bar{b}^U = \begin{cases} \alpha b^U + \beta \ if \alpha \geq 0 \\ \alpha b^L + \beta \ if \alpha < 0 \end{cases} \tag{21}$$

Then:

$$\bar{f}^L(t) := \sum_j \bar{a}_j^L t_j + \bar{b}^L \leq \alpha g(t) + \beta \leq \sum_j \bar{a}_j^U t_j + \bar{b}^U =: \bar{f}^U(t)$$

That is, LBFs can be propagated through linear transformation by redefining the coefficients through Equations (20)–(21).

We now prove Proposition 4 iteratively on $k = 1, \ldots, K$, that is, for $i = 1, \ldots, n_k$ there exist $f_i^{(k),L}(x, W)$ and $f_i^{(k),U}(x, W)$ lower and upper LBFs such

that:

$$\zeta_i^{(k)} \geq f_i^{(k),L}(x, W) := \mu_i^{(k),L} \cdot x + \sum_{l=0}^{k-2} \langle \nu_i^{(l,k),L}, W^{(l)} \rangle + \nu_i^{(k-1,k),L} \cdot W_{i:}^{(k-1)} + \lambda_i^{(k),L} \tag{22}$$

$$\zeta_i^{(k)} \leq f_i^{(k),U}(x, W) := \mu_i^{(k),U} \cdot x + \sum_{l=0}^{k-2} \langle \nu_i^{(l,k),U}, W^{(l)} \rangle + \nu_i^{(k-1,k),U} \cdot W_{i:}^{(k-1)} + \lambda_i^{(k),U} \tag{23}$$

and iteratively find valid values for the LBFs coefficients, i.e., $\mu_i^{(k),L}$, $\nu_i^{(l,k),L}$, $\lambda_i^{(k),L}$, $\mu_i^{(k),U}$, $\nu_i^{(l,k),U}$ and $\lambda_i^{(k),U}$.

For the first hidden-layer we have that $\zeta_i^{(1)} = \sum_j W_{ij}^{(0)} x_j + b_i^{(0)}$. By inequality (18) and using the lower bound for $b_i^{(0)}$ we have:

$$\begin{aligned} \zeta_i^{(1)} &\geq \sum_j \left(W_{ij}^{(0),L} x_j + W_{ij}^{(0)} x_j^L - W_{ij}^{(0),L} x_j^L \right) + b_i^{(0),L} \\ &= W_{i:}^{(0),L} \cdot x + W_{i:}^{(0)} \cdot x^L - W_{i:}^{(0),L} \cdot x^L + b_i^{(0),L} \end{aligned}$$

which is a lower LBF on $\zeta^{(1)}$. Similarly, using Equation (19) we obtain:

$$\zeta_i^{(1)} \leq W_{i:}^{(0),U} \cdot x + W_{i:}^{(0)} \cdot x^L - W_{i:}^{(0),U} \cdot x^L + b_i^{(0),U}$$

which is an upper LBF on $\zeta^{(1)}$. By setting:

$$\begin{aligned} \mu_i^{(1),L} &= W_{i:}^{(0),L} \quad , \quad \mu_i^{(1),U} = W_{i:}^{(0),U} \\ \nu_i^{(0,1),L} &= z^{(0),L} \quad , \quad \nu_i^{(0,1),U} = x^L \\ \lambda_i^{(1),L} &= -W_{i:}^{(0),L} \cdot x^L + b_i^{(0),L} \\ \lambda_i^{(1),U} &= -W_{i:}^{(0),U} \cdot x^L + b_i^{(0),U} \end{aligned}$$

we obtain LBFs $f_i^{(1),L}(x, W)$ and $f_i^{(1),U}(x, W)$ of the form (22)–(23).

Given the validity of Equations (22)–(23) up to a certain k, we now show how to compute the LBF for layer $k+1$. Specifically, given $f_i^{(k),L}(x, W)$ and $f_i^{(k),U}(x, W)$ we explicitly compute $f_i^{(k+1),L}(x, W)$ and $f_i^{(k+1),U}(x, W)$. Let $\zeta_i^{(k),L} = \min f_i^{(k),L}(x, W)$ and $\zeta_i^{(k),U} = \max f_i^{(k),U}(x, W)$ be the minimum and maximum of the two LBFs (which can be computed analytically as the functions are linear). For Lemma 2 there exists a set of coefficients such that $z_i^{(k)} = \sigma(\zeta_i^{(k)}) \geq \alpha_i^{(k),L} \zeta_i^{(k)} + \beta_i^{(k),L}$. By Lemma 3 we know that there exists $\bar{f}_i^{(k),L}(x, W)$ with coefficients $\bar{\mu}_i^{(k),L}$, $\bar{\nu}_i^{(l,k),L}$, $\bar{\lambda}_i^{(k),L}$ obtained through Equations 20–21 such that:

$$z_i^{(k)} \geq \alpha_i^{(k),L} f_i^{(k),L}(x, W) + \beta_i^{(k),L} \geq \bar{f}_i^{(k),L}(x, W)$$

that is $\bar{f}_i^{(k),L}(x,W)$ is a lower LBF on $z_i^{(k)}$ with coefficients $\bar{\mu}_i^{(k),L}$, $\bar{\nu}_i^{(l,k),L}$, $\bar{\lambda}_i^{(k),L}$. Analogously, let $\bar{f}_i^{(k),U}(x,W)$ be the upper LBF on $z_i^{(k)}$ computed in a similar way.

Consider now the bi-linear layer $\zeta_i^{(k+1)} = \sum_j W_{ij}^{(k)} z_j^{(k)} + b_i^{(k)}$. From Equation (18) we know that: $W_{ij}^{(k)} z_j^{(k)} \geq W_{ij}^{(k),L} z_j^{(k)} + W_{ij}^{(k)} z_j^{(k),L} - W_{ij}^{(k),L} z_j^{(k),L}$. By applying Lemma 3 with $\alpha = W_{ij}^{(k),L}$ and $\beta = 0$ we know that there exists a lower LBF $\hat{f}_{ij}^{(k),L}(x,W)$ with a set of coefficients $a_{ij}^{(k),L}$, $b_{ij}^{(l,k),L}$ and $c_{ij}^{(k),L}$ computed applying Equations (20)–(21) to $\bar{\mu}_i^{(k),L}$, $\bar{\nu}_i^{(l,k),L}$, $\bar{\lambda}_i^{(k),L}$ such that: $W_{ij}^{(k),L} z_j^{(k)} \geq \hat{f}_{ij}^{(k),L}(x,W)$. Hence we have:

$$
\begin{aligned}
&\zeta_i^{(k+1)} = \sum_j W_{ij}^{(k)} z_j^{(k)} + b_i^{(k)} \geq \sum_j \big(W_{ij}^{(k),L} z_j^{(k)} + \\
&W_{ij}^{(k)} z_j^{(k),L} - W_{ij}^{(k),L} z_j^{(k),L} \big) + b_i^{(k),L} \geq \\
&\sum_j \hat{f}_{ij}^{(k),L}(x,W) + \sum_j W_{ij}^{(k)} z_j^{(k),L} - \\
&\sum_j W_{ij}^{(k),L} z_j^{(k),L} + b_i^{(k),L} = \\
&\sum_j \big(a_{ij}^{(k),L} \cdot x + \sum_{l=0}^{k-2} \langle b_{ij}^{(l,k),L}, W^{(l)} \rangle \\
&+ b_{ij}^{kl-1,k),L} \cdot W_{j:}^{(k-1)} + c_{ij}^{(k),L} \big) + \\
&W_{i:}^{(k)} \cdot z^{(k),L} - W_{i:}^{(k),L} z^{(k),L}.
\end{aligned}
$$

By setting

$$
\begin{aligned}
&\mu_i^{(k+1),L} = \sum_j a_{ij}^{(k),L} \\
&\nu_i^{(l,k+1),L} = \sum_j b_{ij}^{(l,k),L} \quad k = 0, \ldots, l-2 \\
&\nu_i^{(k-1,k+1),L} = b_i^{(k-1,k),L} \\
&\nu_i^{(k,k+1),L} = z^{(k),L} \\
&\lambda_i^{(k+1),L} = \sum_j c_{ij}^{(k),L} - W_{i:}^{(k),L} \cdot z^{(k),L} + b_i^{(k),L}
\end{aligned}
$$

and re-arranging the elements in the above inequality, we finally obtain:

$$
\begin{aligned}
&\zeta_i^{(k+1)} \geq \mu_i^{(k+1),L} \cdot x + \sum_{l=0}^{k-1} \langle \nu_i^{(l,k+1),L}, W^{(l)} \rangle + \\
&\nu_i^{(k,k+1),L} \cdot W_{i:}^{(k)} + \lambda_i^{k+1),L} =: f_i^{(k+1),L}(x,W)
\end{aligned}
$$

which is of the form of Equation (22) for the lower LBF for the $k+1$-th layer. Similarly, an upper LBF of the form of Equation (23) can be obtained by using Equation (19) in the chain of inequalities above.

E Algorithms and Discussion

Algorithm 2 Lower Bounds for $D_{\text{safe}}(T, S)$

Input: T – Compact Input Region, $f^{\mathbf{w}}$ – Bayesian Neural Network, $p(w|\mathcal{D})$ – Posterior Distribution, N – Number of Samples, γ – Weight margin.
Output: A sound lower bound on $D_{\text{safe}}(T, S)$.

1: # $\mathcal{J}$ is an arbitrary set of weight intervals
2: $\mathcal{J} \leftarrow \emptyset$
3: # $\hat{\Psi}$ is a set of worst-case predicted outputs
4: $\hat{\Psi} \leftarrow \emptyset$
5: # Element-wise products to get width of weight margin.
6: $v \leftarrow \gamma \cdot I \cdot \Sigma$
7: **for** $i \leftarrow 0$ to N **do**
8: $\quad w^{(i)} \sim p(w|\mathcal{D})$
9: $\quad$ # Assume weight intervals are built to be disjoint
10: $\quad [w^{(i),L}, w^{(i),U}] \leftarrow [w_i - v, w_i + v]$
11: $\quad$ # *Interval/Linear Bound Propagation, Section 5.2*
12: $\quad y^L, y^U \leftarrow \texttt{Propagate}(f, T, [w^{(i),L}, w^{(i),U}])$
13: $\quad$ # *Output worst-case see Section VI-F*
14: $\quad y^{\text{worst}} \leftarrow \texttt{Output-Worst}([y^L, y^U])$
15: $\quad \mathcal{J} \leftarrow \mathcal{J} \bigcup \{[w^{(i),L}, w^{(i),U}]\}, \quad \hat{\Psi} \leftarrow \hat{\Psi} \bigcup \{y^{\text{worst}}\}$
16: **end for**
17: $y_{\text{mean}} \leftarrow 0.0$; $p_{\text{total}} \leftarrow 0.0$
18: **for** $i \leftarrow 0$ to $|\mathcal{J}_i|$ **do**
19: $\quad$ # Mult. weight probs and output bounds.
20: $\quad y_{\text{mean}} = y_{\text{mean}} + \hat{\Psi}_i P(\mathcal{J}_i)$
21: $\quad p_{\text{total}} = p_{\text{total}} + P(\mathcal{J}_i)$
22: **end for**
23: # Complete the bound according to Proposition 2.
24: $D^L_{\text{safe}} = y_{\text{mean}} + (1 - p_{\text{total}})\sigma^L$
25: return D^L_{safe}

E.1 Lower Bound on $D_{\text{safe}}(T, S)$

In Algorithm 2, we provide step-by-step pseudocode for lower-bounding $D_{\text{safe}}(T, S)$. One can notice that the algorithm follows a similar computational flow to Algorithm 1 in the main text. Namely, on lines 2 and 4 we establish the sets that we will keep track of (weight intervals and worst-case outputs, respectively). Then,

in lines 7–16, we iteratively sample pairwise disjoint weight intervals and compute their worst-case outputs. On line 14, a key modification is added compared to the lower bound on $P_{\text{safe}}(T, S)$, which is the computation of the worst-case output. In the case of softmax classification we have that `Output-Worst` takes the form:

$$\texttt{Output-Worst}([y^L, y^U]) = \frac{\exp(y_c^L)}{\exp(y_c^L) + \sum_{l \neq c}^{n_c} \exp(y_l^U)} \tag{24}$$

That is, the lower bound for the true class and the upper bound for all other classes. For regression $\texttt{Output-Worst} = y^L$. Both formuals represent the worst-case output and satisfy the conditions needed for Proposition 2 in the main text. Finally, in lines 17-22 we compute the necessary components for our bound in Proposition 2 and complete the bound on line 24. Overall, the computational complexity of this algorithm is exactly the same as the lower bound on probabilistic safety and in practice the computational times are only fractionally different.

E.2 Upper Bound on $P_{\text{safe}}(T, S)$

We provide pseudocode for the computation of the upper bound on $P_{\text{safe}}(T, S)$ in Algorithm 3. To do this we compute unsafe weight sets. We wish to determine that a weight interval is unsafe, i.e., the logical negation of our safety property: $\neg(f^w(x) \in S \ \forall x \in T) = (\exists x \ s.t. \ f^w(x) \notin S)$. Notice that, unlike the procedure for computing safety, here we do not need to jointly propagate a weight-space interval together with the full input specification T, as we only need to find a single x which causes the entire weight interval to be mapped outside of S, and note that every $x \in T$ returns a valid bound. Finding an x that violates the property is identical to the formulation for adversarial examples. Thus, in order to test if there exists a single input that causes the weight interval to be unsafe, we leverage the developments in adversarial attacks in order to attack each sampled weight w_i (line 10 of Algorithm 3).

E.3 Upper Bound on $D_{\text{safe}}(T, S)$

We provide pseudocode for the computation of the upper bound on $D_{\text{safe}}(T, S)$ in Algorithm 4. The main modification to this algorithm is a change from computing the worst-case output to computing the best-case output. This is done with the `Output-Best` function. In the case of softmax classification `Output-Best` takes the form:

$$\texttt{Output-Best}(y^L, y^U) = \frac{exp(y_c^U)}{exp(y_c^U) + \sum_{l \neq c}^{n_c} exp(y_l^L)} \tag{25}$$

and for regression, $\texttt{Output-Best} = y^U$.

Algorithm 3 Upper Bounding $P_{\text{safe}}(T, S)$

Input: T – Input Set, S – Safe Set, $f^{\mathbf{w}}$ – Bayesian Neural Network, $\mathbf{w}$ – Posterior Distribution, N – Number of Samples, γ – Weight Margin.
Output: Safe upper bound on $P_{\text{safe}}(T, S)$.

$\mathcal{K}$ is a set of known unsafe weight intervals
$\mathcal{K} \leftarrow \emptyset$
Element-wise products to get width of weight margin.
$v \leftarrow \gamma \cdot I \cdot \Sigma$
for $i \leftarrow 0$ to N **do**
  $w^{(i)} \sim p(w|\mathcal{D})$
  # Assume weight intervals are built to be disjoint
  $[w^{(i),L}, w^{(i),U}] \leftarrow [w_i - v, w_i + v]$
  # *FGSM/PGD*
  $x_{\text{adv}} \leftarrow \texttt{Attack}(f, w_i, T)$
  # *Interval/Linear Bound Propagation*
  $y^L, y^U \leftarrow \texttt{Propagate}(f, x_{\text{adv}}, [w^{(i),L}, w^{(i),U}])$
  if $\forall y \in [y^L, y^U] y \notin S$) **then**
    $\mathcal{K} \leftarrow \mathcal{K} \bigcup \{[w^{(i),L}, w^{(i),U}]\}$
  end if
end for
$P_{\text{unsafe}} \leftarrow 0.0$
for $i = 0..|\mathcal{K}|$ **do**
  $P_{\text{unsafe}} = P_{\text{unsafe}} + P(\mathcal{K}_i)$
end for
$P^U_{\text{safe}} = 1 - P_{\text{unsafe}}$
return P^U_{safe}

Algorithm 4 Upper Bounding $D_{\text{safe}}(T, S)$

Input: T – Input Set, $f^{\mathbf{w}}$ – Bayesian Neural Network, $p(w|\mathcal{D})$ – Posterior Distribution, N – Number of Samples, γ – Weight margin.
Output: A sound lower bound on $D_{\text{safe}}(T, S)$.

$\mathcal{H}$ is a set of known safe weight intervals
$\mathcal{J} \leftarrow \emptyset$
$\hat{\Psi}$ is a set of best-case predicted outputs
$\hat{\Psi} \leftarrow \emptyset$
Element-wise products to get width of weight margin.
$v \leftarrow \gamma \cdot I \cdot \Sigma$
for $i \leftarrow 0$ to N **do**
  $w^{(i)} \sim p(w|\mathcal{D})$
  $[w^{(i),L}, w^{(i),U}] \leftarrow [w_i - v, w_i + v]$
  # Interval/Linear Bound Propagation, Section 5.2
  $y^L, y^U \leftarrow \texttt{Propagate}(f, T, [w^{(i),L}, w^{(i),U}])$
  # Output upperbound see Eq (25)
  $y^{\text{upper}} \leftarrow \texttt{Output-Best}([y^L, y^U])$
  $\mathcal{J} \leftarrow \mathcal{J} \bigcup \{[w^{(i),L}, w^{(i),U}]\}, \quad \hat{\Psi} \leftarrow \hat{\Psi} \bigcup \{y^{\text{upper}}\}$
end for
$y_{\text{mean}} \leftarrow 0.0$; $p_{\text{total}} \leftarrow 0.0$
for $i \leftarrow 0$ to N **do**
  # Mult. weight probs and output bounds
  $y_{\text{mean}} = y_{\text{mean}} + \hat{\Psi}_i P(\mathcal{H}_i)$
  $p_{\text{total}} = p_{\text{total}} + P(\mathcal{H}_i)$
end for
# Complete the bound according to Proposition 2.
$D^U_{\text{safe}} = y_{\text{mean}} + (1 - p_{\text{total}})\sigma^U$
return D^U_{safe}

E.4 Bonferroni Bounds for Overlapping Weight Intervals

A key challenge of Proposition 1 in the case of variational inference is ensuring that the hyper-rectangles are pairwise disjoint (i.e., $\hat{H}_i \cap \hat{H}_j = \emptyset$). If this is not the case, then enforcing independence can be computationally tricky, as the relative complement of two or more hyper-rectangles is not necessarily a hyper-rectangle. While one could modify the sampling procedure to reject overlapping intervals, or could devise a scheme for sampling pairwise disjoint hyper-rectangles, for high values of N and for a large number of parameters this becomes computationally intensive. To solve this, we highlight that the disjoint union of two or more hyper-rectangles is necessarily a hyper-rectangle. Therefore, we can employ Bonferroni inequalities [11] to get upper and lower bound on the posterior probability of non-disjoint hyper-rectangles:

Corollary 1. *Assume that Σ, the covariance matrix of the posterior distribution of the weights, is diagonal with diagonal elements $\Sigma_1, ..., \Sigma_{n_w}$. Let $\hat{H}_1, ..., \hat{H}_M$ be M safe sets of weights not necessary satisfying $\hat{H}_i \cap \hat{H}_j = \emptyset$ and let the probability*

of any k of these safe sets simultaneously occurring be defined as:

$$S_k := \bigsqcup_{i_1<...<i_k} H_{i_1} \sqcup ... \sqcup H_{i_k}$$

We then have that for any even integer v and odd integer u that the probability of the weights under the posterior is bounded:

$$\sum_{j=1}^{v}(-1)^j Prob(S_j) \leq Prob(\hat{H}_1, ..., \hat{H}_M) \leq \sum_{j=1}^{u}(-1)^j Prob(S_j)$$

where $Prob(S_j)$ is computed according to Corollary 2 as S_j is a single hyper-rectangle.

Now that we can compute if a weight interval is guaranteed to be safe and can obtain a lower bound to the posterior probability covered by many weight intervals, we can combine these subroutines into algorithms for computing the required probability bounds. For bounds on decision robustness we need to consider the upper or lower bound output in conjunction with this probability. Recall that the upper or lower bound output determined by `Output-Worst` or `Output-Best` described in Appendix E and the upper and lower bounds are stored such that the output bound of $\mathcal{J}_i$ is stored in $\hat{\Psi}_i$. To get a lower bound we modify the above corollary to be:

$$\sum_{j=1}^{v}(-1)^j Prob(S_j) \max\{\hat{\Psi}_i\}_{i=1}^{j} \leq \sum_{i=1}^{M} \hat{\Psi}_i Prob(\hat{J}_i). \tag{26}$$

To get an upper bound we use:

$$\sum_{i=1}^{M} \hat{\Psi}_i Prob(\hat{J}_i) \leq \sum_{j=1}^{u}(-1)^j Prob(S_j) \min\{\hat{\Psi}_i\}_{i=1}^{j}$$

Here we can use the max operator for our lower bound and min for our upper bound as every value in the set $\{\hat{\Psi}_i\}_{i=1}^{M}$ is a valid output bound for the disjoint union of hyper-rectangles.

F Computational Complexity

Calculations for probabilistic robustness and decision robustness follow the same computational flow and include bounding of the neural network output, sampling from the posterior distribution, and computation of integrals over boxes on the input and weight space.

Regarding Algorithm 1 (or equivalently Algorithm 2 for decision robustness), it is clear that the computational complexity scales linearly with the number of samples, N, taken from the posterior distribution. Observe that, in order to obtain a tight bound on the integral computation, N needs to be large enough such

that N samples of the posterior with width γ span an area of high probability mass for $p(w|\mathcal{D})$. Unfortunately, this means that, for a given approximation error magnitude, N needs to scale quadratically on the number of hidden neurons. Given the sampling of the hyper-rectangles, computation of the integral over the weight boxes proceeds through Equations (12) and (13). The integration over the weight boxes requires constant time for HMC (though a good quality HMC posterior approximation scales with the number of parameters) and $\mathcal{O}(n_w)$ for VI. The final step needed for the method is bound propagation, which clearly differs when using IBP or LBP. In particular, the cost of performing IBP is $\mathcal{O}(K\hat{n}\hat{m})$, where K is the number of hidden layers and $\hat{n} \times \hat{m}$ is the size of the largest weight matrix $W^{(k)}$, for $k = 0, \ldots, K$. LBP is, on the other hand, $\mathcal{O}(K^2\hat{n}\hat{m})$. Overall, the time complexity for certifying a VI BNN with IBP is therefore $\mathcal{O}(Nn_wK\hat{n}\hat{m})$, and similar formulas can be obtained for alternative combinations of inference and propagation techniques that are employed. We remark that, while sound, the bounds we compute are not guaranteed to converge to the true values of $P_{\text{safe}}(T, S)$ and $D_{\text{safe}}(T, S)$ in the limit of the number of samples N because of the introduction of over-approximation errors during bound propagation.

G All Experimental Parameters

In this section we detail all of the hyper-parameters for each network trained and describe the convolutional architectures used in our experiments.

G.1 Horizontal Collision Avoidance (HCAS)

Infer.	**Epochs**	**Batch**	**Depth**	**Width**	**LR**	**Loss**	**Burn_in**	**Decay**	**Prior**	ϵ	λ
VOGN	75	512	1	125	0.05	Standard	-	-	2.5	-	-
VOGN	75	512	1	125	0.15	Robust	-	-	2.5	0.15	0.5
HMC	75	-	1	125	0.05	Standard	20	0.3	2.5	-	-
HMC	75	-	1	125	0.05	Robust	1	0.3	2.5	0.15	0.5

Table 2. Experimental Configurations for Horizontal Collision Avoidance

G.2 MNIST Handwritten Digits

Inference	Epochs	Batchsize	Depth	Width	LR	Decay	Loss	ϵ	λ	Prior
VOGN	40	128	1	125	0.1	0.1	Standard	-	-	1.0
VOGN	40	512	1	125	0.1	0.1	Robust	0.05	0.25	1.0
NA	40	512	1	125	0.025	0.1	Standard	-	-	1.0
NA	40	512	1	125	0.025	0.1	Robust	0.05	0.75	1.0
BBB	40	512	1	125	0.075	0.01	Standard	-	-	0.5
BBB	40	512	1	125	0.08	0.01	Robust	0.05	0.75	0.5

Table 3. Experimental Configurations for MNIST

G.3 German Traffic Sign Recognition Benchmark

In the German Traffic Sign Recognition Benchmark (GTSRB) experiment, we use Variational Online Gaussian Newton (VOGN) inference. The model is trained for 50 epochs with a batch size of 32. The learning rate is set to 0.001, and no decay is applied (decay rate is 0.00). The loss function is the robust loss, specifically configured with ϵ set to 0.01 and λ set to 0.1, to enhance the model's robustness. The model is designed to classify traffic signs into 3 different classes. Additionally, a prior of 1.0 is applied in the model configuration to regularize the learning process and prevent overfitting. The architecture of the neural network used in this experiment is a sequential model constructed as follows. The first layer is a convolutional layer (Conv2D) with 4 filters, each of size 4×4, and ReLU activation function. This layer takes an input of shape $30 \times 30 \times 3$ (30x30 pixel images with 3 color channels). Following this, an average pooling layer (AveragePooling2D) with a pool size of 4×4 is applied to reduce the spatial dimensions of the feature maps. The output is then flattened into a 1D vector by the Flatten layer, which is then fed into a dense layer (Dense) with 50 units and ReLU activation function. The final layer is another dense layer with the number of units equal to the number of classes (3 in this case) and a softmax activation function, which outputs the class probabilities.

G.4 PneumoniaMNIST

In the PneumoniaMNIST experiment, we again utilize Variational Online Gaussian Newton (VOGN) inference over 30 epochs with a batch size of 128. The learning rate is set to 0.075 with no decay applied (decay rate is 0.0). The loss function is the robust loss to enhance model resilience, configured with ϵ set to 0.015 and λ set to 0.25. The dataset used is PneumoniaMNIST, which consists of images with an input dimension of 28x28 pixels. A prior of 1.0 is incorporated to regularize the learning process. The neural network architecture employed in this experiment is a sequential model constructed as follows. The first layer is a convolutional layer (Conv2D) with 10 filters, each of size 4×4, and ReLU activation function. Following this, another Conv2D layer with the same configuration

is added. An average pooling layer (AveragePooling) is applied to downsample the feature maps. The output is then flattened into a 1D vector by the Flatten layer, followed by a dense layer (Dense) with 50 units and ReLU activation function. The final layer is a dense layer with 2 units and softmax activation function, which outputs the probabilities for the two classes.