Probabilistic Safety for Bayesian Neural Networks

Matthew Wicker* University of Oxford

Luca Laurenti* University of Oxford

Andrea Patane* University of Oxford

Marta Kwiatkowska University of Oxford

Abstract

We study probabilistic safety for Bayesian Neural Networks (BNNs) under adversarial input perturbations. Given a compact set of input points, $T \subseteq \mathbb{R}^m$, we study the probability w.r.t. the BNN posterior that all the points in T are mapped to the same region S in the output space. In particular, this can be used to evaluate the probability that a network sampled from the BNN is vulnerable to adversarial attacks. We rely on relaxation techniques from non-convex optimization to develop a method for computing a lower bound on probabilistic safety for BNNs, deriving explicit procedures for the case of interval and linear function propagation techniques. We apply our methods to BNNs trained on a regression task, airborne collision avoidance, and MNIST, empirically showing that our approach allows one to certify probabilistic safety of BNNs with millions of parameters.

1 INTRODUCTION

Although Neural Networks (NNs) have recently achieved state-of-the-art performance [14], they are susceptible to several vulnerabilities [3], with adversarial examples being one of the most prominent among them [28]. Since adversarial examples are arguably intuitively related to *uncertainty* [17], Bayesian Neural Networks (BNNs), i.e. NNs with a probability distribution placed over their weights and biases [24], have recently been proposed as a potentially more robust learning paradigm [6]. While retaining the advantages intrinsic to deep learning (e.g. representation learning), BNNs also enable principled evaluation of model uncertainty, which can be taken into account at prediction time to enable safe decision making.

Many techniques have been proposed for the evaluation of the robustness of BNNs, including generalisation of NN gradient-based adversarial attacks to BNN posterior distribution [19], statistical verification techniques for adversarial settings [7], as well as pointwise (i.e. for a specific test point x^*) uncertainty evaluation techniques [27]. However, to the best of our knowledge, methods that formally (i.e., with certified bounds) give guarantees on the behaviour of BNNs against adversarial input perturbations in probabilistic settings are missing.

In this paper we aim at analysing *probabilistic safety* for BNNs. Given a BNN, a set $T \subseteq \mathbb{R}^m$ in the input space, and a set $S \subseteq \mathbb{R}^{n_c}$ in the output space, probabilistic safety is defined as the probability that for all $x \in T$ the prediction of the BNN is in S. In adversarial settings, the input region T is built around the neighborhood of a given test point x^*, while the output region S is defined around the BNN prediction on x^*, so that probabilistic safety translates into computing the probability that adversarial perturbations of x^* cause small variations in the BNN output. Note that probabilistic safety represents a probabilistic variant of the notion of safety commonly used to certify deterministic NNs [26].

Unfortunately, computation of probabilistic safety for a BNN over a compact set T and for an output region S is not trivial, as it involves computing the probability that a deterministic NN sampled from the BNN posterior is safe (i.e., all the points in a given set are mapped by the network to a given output set), which is known to be NP-complete [16]. Nevertheless, we derive a method for the computation of a *certified lower bound* for probabilistic safety. In particular, we show that the computation of probabilistic safety for BNNs is equivalent to computing the measure, w.r.t. BNN posterior, of the set of weights for which the resulting deterministic NN is safe, i.e., robust to adversarial perturbations. We compute a subset

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

* Equal Contributions.

of such weights, $\hat{H}$, and build on relaxation techniques from non-linear optimisation to check whether, given a compact set T and a safe output region S, all the networks instantiated by weights in $\hat{H}$ are safe. We provide lower bounds for the case of Interval Bound Propagation (IBP) and Linear Bound Propagation (LBP) for BNNs trained with variational inference (VI) [5]. However, we note that the derived bounds can be extended to other approximate Bayesian inference techniques.

We experimentally investigate the behaviour of our method on a regression task, the VCAS dataset and the MNIST dataset, for a range of properties and different BNN architectures, and applying bounds obtained both via IBP and LBP. We show that our method allows one to compute non-trivial, certified lower bounds on the probabilistic safety of BNNs with millions of weights in a few minutes.

In summary, this paper makes the following main contributions.[1]

- We propose a framework based on relaxation techniques from non-convex optimisation for the analysis of probabilistic safety for BNNs with general activation functions and multiple hidden layers.
- We derive explicit procedures based on IBP and LBP for the computation of the set of weights for which the corresponding NN sampled from the BNN posterior distribution is safe.
- On various datasets we show that our algorithm can efficiently verify multi-layer BNNs with millions of parameters in a few minutes.

Related Work. Most existing certification methods are designed for deterministic neural networks (see e.g., [16]) and cannot be used for verification of Bayesian models against probabilistic properties. In particular, in [30, 32, 13] Interval Bound Propagation (IBP) and Linear Bound Propagation (LBP) approaches have been employed for certification of deterministic neural networks. However, these methods cannot be used for BNNs because they all assume that the weights of the networks are deterministic, i.e fixed to a given value, while in the Bayesian setting we need to certify the BNN for a continuous range of values for weights that are not fixed, but distributed according to the BNN posterior. We extend IBP and LBP to BNNs in Section 4.1.

Bayesian uncertainty estimates have been shown to empirically flag adversarial examples in [25, 27]. However, these approaches only consider pointwise uncertainty estimates, that is, specific to a particular test point. In contrast, probabilistic safety aims at computing the uncertainty for compact subspaces of input points, thus taking into account worst-case adversarial perturbations of the input point when considering its neighbourhood. A probabilistic property analogous to that considered in this paper has been studied for BNNs in [7, 23]. However, the solution methods presented in [7, 23] are statistical, with confidence bounds, and in practice cannot give certified guarantees for the computed values, which are important for safety-critical applications. Our approach, instead building on non-linear optimisation relaxation techniques, computes a certified lower bound.

Methods to compute probabilistic adversarial measures of robustness in Bayesian learning settings have been explored for Gaussian Processes (GPs), both for regression [8] and classification tasks [27, 4]. However, the vast majority of inference methods employed for BNNs do not have a Gaussian approximate distribution in latent/function space, even if they assume a Guassian distribution in the weight space [5]. Hence, certification techniques for GPs cannot be directly employed for BNNs. In fact, because of the non-linearity of the BNN structure, even in this case the distribution of the BNN is not Gaussian in function space. Though methods to approximate BNN inference with GP-based inference have been proposed [10], the guarantees obtained in this way would apply to the approximation and not the actual BNN, and would not provide error bounds. 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.

2 BAYESIAN NEURAL NETWORKS (BNNs)

In this section we review BNNs. We write $f^{\mathbf{w}}(x) = [f_1^{\mathbf{w}}(x), \ldots, f_{n_c}^{\mathbf{w}}(x)]$ to denote a BNN with n_c output units and an unspecified number of hidden layers, where $\mathbf{w}$ is the weight vector random variable. Given a distribution over $\mathbf{w}$ and $w \in \mathbb{R}^{n_w}$, a weight vector sampled from the distribution of $\mathbf{w}$, we denote with $f^w(x)$ the corresponding deterministic neural network with weights fixed to w. Let $\mathcal{D} = \{(x_i, y_i), i \in \{1, ..., N_{\mathcal{D}}\}\}$ be the training set. In Bayesian settings, we assume a prior distribution over the weights, i.e. $\mathbf{w} \sim p(w)$, so that learning amounts to computing the posterior distribution over the weights, $p(w|\mathcal{D})$, via the application of Bayes rule. Unfortunately, because of the non-linearity introduced by the neural network architecture, the computation of the posterior cannot be done analytically [20].

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

In this work we focus on Gaussian Variational Inference (VI) approximations via Bayes by Backprop [5]. In particular, VI proceed by finding a suitable Gaussian approximating distribution $q(w) = \mathcal{N}(w|\mu, \Sigma)$ for the posterior distribution, i.e. such that $q(w) \approx p(w|\mathcal{D})$. The core idea is that the mean and covariance of $q(w)$ are iteratively optimized by minimizing a divergence measure between $q(w)$ and $p(w|\mathcal{D})$. In particular, in Section 4 we give explicit bounds for BNN certification for variational inference; however, we remark that the techniques developed in this paper also extend to other classes of distributions and approximate inference methods, such as Hamiltonian Monte Carlo [24] or dropout [12], with the caveat that in those cases the integral in Eqn. (2) (Section 4) will not be Gaussian and will need to be computed by means of Monte Carlo sampling techniques.

3 PROBLEM FORMULATION

A BNN is a stochastic process whose randomness comes from the weight distribution. Therefore, in order to study robustness, its probabilistic nature should be considered. In this paper we focus on probabilistic safety, which is a widely employed measure to characterize the robustness of stochastic models [1, 18], and also represents a probabilistic generalization of the notion of safety commonly used to guarantee the robustness of deterministic neural networks against adversarial examples [26]. In particular, probabilistic safety for BNNs is defined as follows.

Definition 1. *Let $f^{\mathbf{w}}$ be a BNN, $\mathcal{D}$ a dataset, $T \subset \mathbb{R}^m$ a compact set of input points, and $S \subseteq \mathbb{R}^{n_c}$ a safe set. Then, probabilistic safety is defined as*

$$P_{safe}(T, S) := Prob_{w\sim\mathbf{w}}(\forall x \in T, f^w(x) \in S|\mathcal{D}). \quad (1)$$

$P_{\text{safe}}(T, S)$ is the probability that for all input points in T the output of the BNN belongs to a given output set S. Note that the probabilistic behaviour in $P_{\text{safe}}(T, S)$ is uniquely determined from the distribution over the weights random variable $\mathbf{w}$. In particular, no distribution is assumed in the input space. Hence, $P_{\text{safe}}(T, S)$ represents a probabilistic measure of robustness. We make the following assumption on S.

Assumption 1. *We assume that S is described by n_S linear constraints on the values of the final layer of the BNN, that is,*

$$S = \{y \in \mathbb{R}^{n_c} \,|\, C_S y + d_S \geq 0, C_S \in \mathbb{R}^{n_S \times n_c}, d_S \in \mathbb{R}^{n_S}\}$$

We stress that, as discussed in [13], this assumption encompasses most properties of interest for the verification of neural networks. We refer to $C_S y + d_S \geq 0$ as the *safety specification* associated to the safe set S. Below, in Example 1, we illustrate the notion of probabilistic safety on an example.

Example 1. *We consider a regression task where we learn a BNN from noisy data centred around the function $y = x^3$, as illustrated in Figure 1. We let $T = [-\epsilon, \epsilon]$ and $S = [-\delta, \delta]$, with $\epsilon = 0.2$ and $\delta = 5$ be two intervals. Then, we aim at computing $P_{safe}(T, S)$, that is, the probability that for all $x \in [-\epsilon, \epsilon]$, $f^{\mathbf{w}}(x) \in [-\delta, \delta]$.*

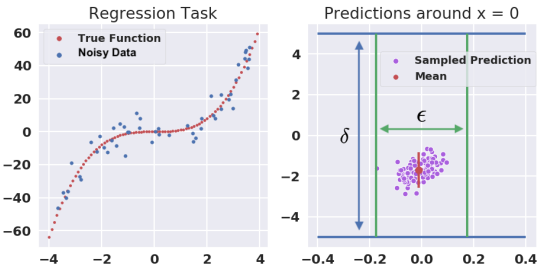

Figure 1: Left: 50 points sampled uniformly from $[-4, 4]$ and their corresponding outputs according to $y = x^3 + \mathcal{N}(0, 5)$. Right: For $T = [-\epsilon, \epsilon]$ and $S = [-\delta, \delta]$, with $\epsilon = 0.2$ and $\delta = 5$, we visualize the property $P_{\text{safe}}(T, S)$. The red dot and bar depicts the mean and the standard deviation of the BNN prediction in $x^* = 0$.

Probabilistic safety can be used in a regression setting to formally account for the uncertainty in the learned model. For instance, it can be employed in a model-based reinforcement learning scenario to ensure the safety of the learned model under uncertain environments [2]. In a classification problem, $P_{\text{safe}}(T, S)$ could be used to evaluate the uncertainty around the model predictions in adversarial settings [11]. We remark that probabilistic safety is also related to other adversarial robustness measures in the literature [9, 4]. In particular, it is straightforward to show (see Proposition 4 in the Supplementary Material in [31]) that

$$P_{\text{safe}}(T, S) \leq \inf_{x\in T} Prob_{w\sim\mathbf{w}}(f^w(x) \in S).$$

Moreover, if for $i \in \{1, ..., n_c\}$ and $a \in \mathbb{R}_{>0}$ we assume that $S = \{y \in \mathbb{R}^{n_c} \,|\, y_i > a\}$, then it holds that

$$aP_{\text{safe}}(T, S) \leq \inf_{x\in T} \mathbb{E}_{w\sim\mathbf{w}}[f_i^w(x)].$$

Approach Outline Probabilistic safety, $P_{\text{safe}}(T, S)$, is not trivial to compute for a given compact set T in the input space and safe set S that satisfies Assumption 1. In fact, already in the case of deterministic neural networks, safety has been shown to pose an NP-complete problem [32]. Therefore, in Section 4 we derive a method for lower-bounding (i.e., for the worst-case analysis) of $P_{\text{safe}}(T, S)$, which can be used for certification of the probabilistic safety of BNNs. We first show that the computation of $P_{\text{safe}}(T, S)$ is equivalent to computing the

maximal set of weights H such that the corresponding deterministic neural network is safe, i.e., $H = \{w \in \mathbb{R}^{n_w} | \forall x \in T, f^w(x) \in S\}$. The computation of H is unfortunately itself not straightforward. Instead, in Section 5, we design a method to compute a subset of safe weights $\hat{H} \subseteq H$, and discuss how $\hat{H}$ can be used to compute a certified lower bound to $P_{\text{safe}}(T, S)$. Our method (detailed in Algorithm 1) works by sampling weights w from the posterior BNN distribution and iteratively building safe weight regions, in the form of disjoint hyper-rectangles, around the sampled weights. This requires a subroutine that, given $\hat{H}$, checks whether it constitutes a safe set of weights or not, that is, verifies if the statement $\hat{H} \subseteq H$ is true. In Section 4.1, we derive two alternative approaches for the solution of this problem, one based on Interval Bound Propagation (IBP) (introduced in Section 4.1.1) and the other on Linear Bound Propagation (LBP) (introduced in Section 4.1.2). These work by propagating the input region T and the weight rectangle $\hat{H}$ through the neural network, in the form of intervals (in the case of IBP) and lines (in the case of LBP) and checking whether the resulting output region is a subset of S, thus deciding if $\hat{H}$ is a safe set of weights or not.

4 BOUNDS FOR PROBABILISTIC SAFETY

We show that the computation of $P_{\text{safe}}(T, S)$ reduces to computing the maximal set of weights for which the corresponding deterministic neural network is safe. To formalize this concept, consider the following definition.

Definition 2. *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 $H = \{w \in \mathbb{R}^{n_w} \,|\, \forall x \in T, f^w(x) \in S\}$. Furthermore, we say that $\hat{H}$ is a safe set of weights from T to S, or simply a safe set of weights, iff $\hat{H} \subseteq H$.*

If $\hat{H}$ is a safe set of weights, then Definition 2 implies that for any $w \in \hat{H}$ the corresponding neural network is safe, i.e., $\forall x \in T, f^w(x) \in S$. Then, a trivial consequence of the definition of probabilistic safety is the following proposition.

Proposition 1. *Let H be the maximal safe set of weights from T to S. Assume that $\mathbf{w} \sim q(w)$. Then, it holds that*

$$\int_H q(w)dw = P_{\textit{safe}}(T, S). \tag{2}$$

Proposition 1 simply translates the safety property from the function space to an integral computation on the weight space. As a consequence of Proposition 1, in the case of $q(w) = \mathcal{N}(w|\mu, \Sigma)$ with diagonal covariance, i.e., of posterior distribution computed by diagonal Gaussian VI, we obtain the following corollary.

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 such that, for $i \in \{1, .., M\}$, $\hat{H}_i = [l_1^i, u_1^i] \times ... \times [l_{n_w}^i, u_{n_w}^i]$ and $\hat{H}_i \cap \hat{H}_j = \emptyset$ for $i \neq j$. Then, it holds that*

$$P_{\textit{safe}}(T, S) \geq \sum_{i=1}^{M} \prod_{j=1}^{n_w} \frac{1}{2} \left(\textit{erf}\left(\frac{\mu_j - l_j^i}{\sqrt{2\Sigma_j}} \right) - \textit{erf}\left(\frac{\mu_j - u_j^i}{\sqrt{2\Sigma_j}} \right) \right).$$

Proposition 1 and Corollary 1 guarantee that the computation of P_{safe} is equivalent to the characterization of H, the maximal safe set of weights, and a lower bound for P_{safe} can be computed by considering the union of M safe sets of weights. In what follows, in Section 4.1, we derive a framework to efficiently check if a given set of weights is safe. Then, in Section 5 we present a method to generate safe sets of weights, which will be integrated in an algorithm for the computation of $P_{\text{safe}}(S, T)$ by making use of Proposition 1.

4.1 SAFETY COMPUTATION

In this subsection we derive schemes for checking whether a given hyper-rectangle, $\hat{H}$, in the weight space is such that $\hat{H} \subseteq H$, that is, for $S = \{y \in \mathbb{R}^{n_c} \,|\, C_S y + d_S \geq 0, C_S \in \mathbb{R}^{n_S \times n_c}, d_S \in \mathbb{R}^{n_S}\}$, we want to check whether $C_S f^w(x) + d_S \geq 0, \quad \forall x \in T, \forall w \in \hat{H}$. This is equivalent to check:

$$\min_{w \in \hat{H}, x \in T} C_S f^w(x) + d_S \geq 0. \tag{3}$$

In the following we show how IBP (Section 4.1.1) and LBP (Section 4.1.2) can be used to find a lower bound on the solution of the problem posed by Equation (3). The basic principles behind the two methods are depicted in Figure 2 for an illustrative one-dimensional case (IBP shown in plots (a)–(c), LBP in plots (d)–(f)). Given a bounding box in the input of each BNN layer (plot (a), as for a deterministic NN), and an interval in the weight space, $\hat{H}$ (plot (b), which is due to the fact that the BNN has probabilistic weights), IBP propagates the two bounding boxes, as detailed in Section 4.1.1, to obtain a bounding box on the network output (plot (c)). The process is then iterated for each layer. In LBP, instead, the linear function that bounds the input at each layer (plot (d)) is combined with the weight space interval $\hat{H}$ (plot (e)) to obtain a linear bound on the layer output (plot (f)) as detailed in Section 4.1.2. Intuitively, as LBP allows for a linear bound, it mimics more closely the behaviour of the network, thus giving better bounds, albeit at an increased computational cost. This is further investigated in the experiments in Section 6.

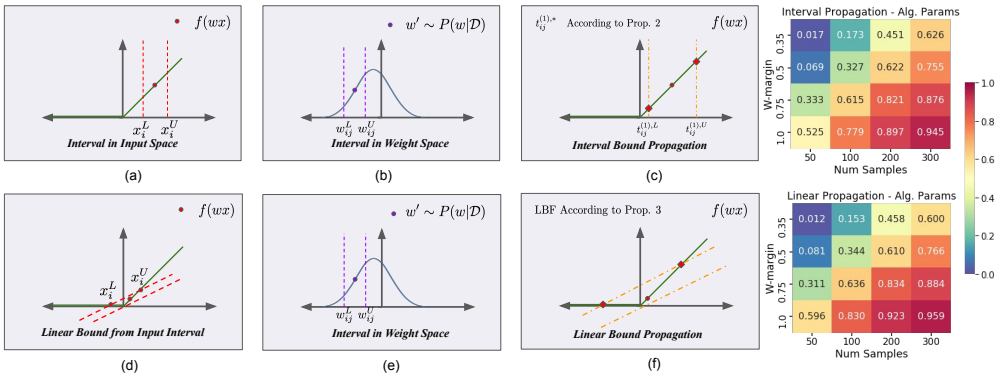

Figure 2: Example of IBP (first row) and LBP (second row) for BNNs. For IBP we consider an interval in the input space (a) together with an interval in the weight space (b). The two intervals are combined to obtain an interval in the output, which is guaranteed to contain the network output (c). In the LBP case, the input bound and the weight interval are combined to obtain linear bounds that contain the network output at any layer (d)-(e)-(f). In the last column we show the application of Algorithm 1 for the computation of the property illustrated in Example 1. We consider different values of the parameters required by the algorithm for both IBP and LBP. Notice that LBP tends to give tighter bounds (i.e. higher values) compared to IBP.

Before discussing IBP and LBP in detail, we first introduce common notation for the rest of the section. We consider fully-connected networks of the form:[2]

$$z^{(0)} = x \tag{4}$$

$$\zeta_i^{(k+1)} = \sum_{j=1}^{n_k} W_{ij}^{(k)} z_j^{(k)} + b_i^{(k)} \quad i = 0, \ldots, n_{k+1} \tag{5}$$

$$z_i^{(k)} = \sigma(\zeta_i^{(k)}) \quad i = 0, \ldots, n_k \tag{6}$$

for $k = 1, \ldots, K$, 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. 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 $\hat{H}$ for $W^{(k)}$ and $b^{(k),L}$ and $b^{(k),U}$ for those of $b^{(k)}$, for $k = 0, \ldots, K$. Notice 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, Equation (5) defines a quadratic form.

4.1.1 Interval Bound Propagation

IBP has already been employed for fast certification of deterministic neural networks [13]. For a deterministic network, the idea is to propagate the input box around x, i.e., $T = [x^L, x^U]$,[3] through the first layer, so as to find values $z^{(1),L}$ and $z^{(1),U}$ such that $z^{(1)} \in [z^{(1),L}, z^{(1),U}]$, and then iteratively propagate the bound through each consecutive layer for $k = 1, \ldots, K$. The final box constraint in the output layer can then be used to check for the property of interest [13]. The only adjustment needed in our setting (that is, for the bounding of Equation (3)) is that at each layer we also need to propagate the interval on the weight matrix $[W^{(k),L}, W^{(k),U}]$ and that on 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 (5), 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, which permits the application of Equation (6) to the bounding interval. This is summarised in the following proposition.

Proposition 2. *Let $f^w(x)$ be the network defined by the set of Equations* (4)–(6)*, let for $k = 0, \ldots, K$:*

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

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

[2] CNNs can be considered by applying the approach of [32].

[3] In the general case, in which T is not a box already, we first compute a bounding box $R = [x^L, x^U]$ such that $T \subset R$, and propagate R with IBP, which yields a worst-case analysis.

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

$$\zeta^{(k+1),L} = \sum_j t_{:j}^{(k),L} + b^{(k),L}$$

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

Then we have that $\forall x \in T$ *and* $\forall w \in \hat{H}$*:*

$$f^w(x) = \zeta^{(K+1)} \in \left[\zeta^{(K+1),L}, \zeta^{(K+1),U}\right].$$

The proposition above, whose proof is in the Supplementary Material (found in [31]), yields a bounding box for the output of the neural network in T and $\hat{H}$. This can be used directly to find a lower bound for Equation (3), and hence checking whether $\hat{H}$ is a safe set of weights. As noticed in [13] and discussed in the Supplementary Material, for BNNs a slightly improved IBP bound can be obtained by eliding the last layer with the linear formula of the safety specification of set S.

4.1.2 Linear Bound Propagation

We now discuss how LBP can be used to lower-bound the solution of Equation (3), 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, LBFs in the deterministic case are computed by bounding the activation functions, and propagating the bounds through the affine function that defines each layer.

Similarly, in our setting, given T in the input space and $\hat{H}$ 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 for Equation (6), as discussed in [32].

Lemma 1. *Let* $f^w(x)$ *be defined by Equations* (4)–(6)*. 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 (5) can be derived using McCormick's inequalities [22]:

$$W_{ij}^{(k)} z_j^{(k)} \geq W_{ij}^{(k),L} z_j^{(k)} + W_{ij}^{(k)} z_j^{(k),L} - W_{ij}^{(k),L} z_j^{(k),L} \tag{7}$$

$$W_{ij}^{(k)} z_j^{(k)} \leq W_{ij}^{(k),U} z_j^{(k)} + W_{ij}^{(k)} z_j^{(k),L} - W_{ij}^{(k),U} z_j^{(k),L} \tag{8}$$

for every $i = 1, \ldots, n_k$, $j = 1, \ldots, n_{k-1}$ and $k = 1, \ldots, K$. The bounds of Equations (7)–(8) can thus be used in Equation (5) to obtain LBFs on the pre-activation function of the following layer, i.e. $\zeta^{(k+1)}$. The final linear bound can be obtained by iterating the application of Lemma 1 and Equations (7)–(8) through every layer. This is summarised in the following proposition, which is proved in the Supplementary Material (see [31]) along with an explicit construction of the LBFs.

Proposition 3. *Let* $f^w(x)$ *be the network defined by the set of Equations* (4)–(6)*. Then for every* $k = 0, \ldots, K$ *there exist lower and upper LBFs on the pre-activation function of the form:*

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

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

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 the Supplementary Material [31].*

Now 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* $\forall w \in \hat{H}$*:*

$$f^w(x) = \zeta^{(K+1)} \in \left[\zeta^{(K+1),L}, \zeta^{(K+1),U}\right].$$

Again, while the lower and upper bounds on $f^w(x)$ computed by Proposition 3 can be directly used to check whether $\hat{H}$ is a safe set of weights, an improved bound can be obtained by directly considering the linear constraint form of S when computing the LBFs. This is further described in the Supplementary Material [31].

Algorithm 1 Probabilistic Safety for BNNs

Input: T – compact input region, S – safe set, C_S, d_S – output constraints, $f^{\mathbf{w}}$ – BNN, $\mathbf{w} \sim \mathcal{N}(\cdot|\mu, \Sigma)$ – weight posterior, Σ assumed to be diagonal, N – number of samples, γ – weight margin.

Output: Safe lower bound on $P_{\text{safe}}(T, S)$.

1: $\hat{H} \leftarrow \emptyset$ {$\hat{H}$ is the set of safe weights}
2: **for** $i \leftarrow 0$ to N **do**
3: $\quad w' \sim \mathcal{N}(\cdot|\mu, \Sigma)$
4: $\quad [w^L, w^U] \leftarrow [w' - \gamma\text{diag}(\Sigma), w' + \gamma\text{diag}(\Sigma)]$
5: $\quad y^L, y^U \leftarrow \texttt{Method}(f, T, [w^L, w^U])$ {IBP/LBP}
6: $\quad$ **if** CheckProperty(C_S, d_S, y^L, y^U) **then**
7: $\quad\quad H \leftarrow \hat{H} \bigcup \{[w^L, w^U]\}$
8: $\quad$ **end if**
9: **end for**
10: $\hat{H} \leftarrow$ MergeOverlappingRectangles($\hat{H}$)
11: $p \leftarrow 0.0$
12: **if** $\hat{H} \neq \emptyset$ **then**
13: $\quad$ **for** $[w^L, w^U] \in \hat{H}$ **do**
14: $\quad\quad p \leftarrow p + \prod_{i=1}^{n_w} \frac{1}{2} \left(\text{erf}\left(\frac{\mu_i - w_i^L}{\sqrt{2\Sigma_i}}\right) - \text{erf}\left(\frac{\mu_i - w_i^U}{\sqrt{2\Sigma_i}}\right) \right)$
15: $\quad$ **end for**
16: **end if**
17: return p

5 ALGORITHM

In Algorithm 1 we illustrate our framework for lower-bounding of $P_{\text{safe}}(T, S)$ under the simplifying assumption that Σ is diagonal. The computational complexity of this algorithm for both IBP and LBP is discussed in the Supplementary Material [31]. In lines 1–10 the algorithm computes $\hat{H}$, which is a safe set weights whose union is a subset of H, the maximal safe set of weights (see Definition 1). In general, H is not a connected set. Hence, we build $\hat{H}$ as a union of hyper-rectangles (line 7), each of which is safe. Each candidate safe hyper-rectangle is generated as follows: we sample a weight realisation, w, from the weights posterior (line 3) and expand each of its dimensions by a value computed by multiplying the variance of each weight with a proportional factor γ, which we refer to as the *weight margin* (line 4). The trade-off is that greater values of γ will yield larger regions, and hence potentially a larger safe region $\hat{H}$, though the chances that a wide interval $[w^L, w^U]$ will fail the certification test of line 6 are higher. This process is repeated N times to generate multiple safe weight rectangles in disconnected regions of the weight space. Empirically, we found this heuristic strategy to yield good candidate sets.[4] In line 4–7 we check if the given hyper-rectangle is safe by using either IBP or LBP, and by checking the safety specification (line 6) as detailed in the Supplementary Material. All safe rectangles are added to $\hat{H}$ in line 7. In line 10 we merge overlapping rectangles generated in the main algorithm loop, to guarantee that $\hat{H}$ is a set of non-overlapping hyper-rectangles of weights (as for Corollary 1). This is done simply by iterating over the previously computed safe rectangles, and by merging them iteratively over each dimension. Finally, in lines 12–15, by employing Corollary 1, we compute a lower bound for $P_{\text{safe}}(S, T)$. The following theorem, which is a straightforward consequence of Proposition 1, 2, 3 and Corollary 1, guarantees that Algorithm 1 returns a certified lower bound for $P_{\text{safe}}(S, T)$.

Theorem 1. *Let p be as computed by Algorithm 1. Then, it holds that*

$$p \leq P_{safe}(S, T).$$

In the general case, when a non-Gaussian distribution or full covariance matrix is employed (e.g. with SWAG [21] or HMC) the only modifications to make to the algorithm are in line 4, where an estimation of the weight variance can be used, and in line 14, which needs to be computed with Monte Carlo techniques. In this case, the weights sampled during integration can be utilised in line 3 of Algorithm 1. We expect that different posterior approximation methods with non-diagonal distributions will yield different probabilistic safety profiles, but stress that our method and its computational complexity is independent of the inference method used.

Example 2. *Lower bounds for the property discussed in Example 1 are depicted in Figure 2 (rightmost plots). We analyse the influence of parameters involved in Algorithm 1 (the number of samples N and the weight margin γ), and the use of either IBP or LBP. As expected, we obtain higher values (that is, a tighter bound) as the number of samples increases, as this implies a better chance of finding a safe weight interval, and we observe similar behaviour for γ. Notice also that the bounds provided by LBP (bottom row) are always slightly more tight than those provided by IBP (top row).*

6 EXPERIMENTAL RESULTS

We explore the empirical performance of our framework. We begin with an extended analysis of the regression setting introduced in Example 1. We then turn our attention to an airborne collision avoidance scenario (VCAS) [16, 29]. Finally, we explore the scalability of our approach on the MNIST dataset, and show that we compute non-trivial lower bounds on probabilistic safety even for networks with over 1M parameters.

Regression Task In Figure 3 we explore the regression problem introduced in Example 1. We train a BNN with 128 hidden neurons for 10 epochs. We check the per-

[4] Note that the algorithm output is a lower bound of $P_{\text{safe}}(T, S)$ independently of the heuristic used for building $\hat{H}$.

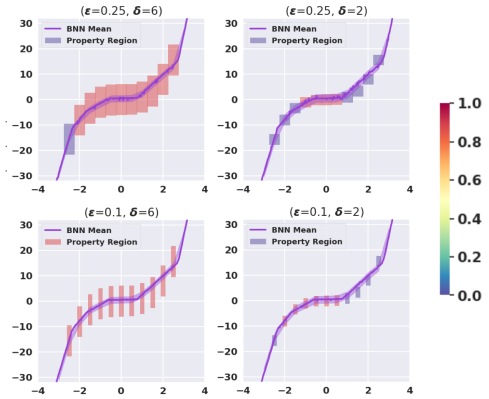

Figure 3: Probabilistic safety for the regression task on various input regions. Each box in the plot represents a safety specification and is colored with the lower bound returned by our method on the probability that the specification is met. The purple region represents 2 standard deviations about the mean of the BNN.

formance of our methodology on this BNN under different properties by considering many input regions T and output sets S. In particular, we extend the property in Example 1 to multiple intervals along the $x-$axis for different values of ϵ and δ and use LBP for the certification of safe weights regions. Namely, we explore four different property specifications for the combination of $\epsilon \in \{0.1, 0.25\}$ and $\delta \in \{2, 6\}$, where every box in the plot represents both an input (range along the x axis) and output region (range along the y axis) and is colored accordingly with the lower bound obtained (which hence represents a lower bound on the probability that the samples from the BNN will remain inside that box for each specific range of x-axis values). Naturally, we obtain higher values in regions in which the BNN is flat. Also the bounds decreases as soon as ϵ or δ increases as these imply a tighter property specification.

Airborne Collision Avoidance We empirically evaluate probabilistic safety for the vertical collision avoidance system dataset (VCAS) [15]. The task of the original NN is to take as input the information about the geometric layout (heading, location, and speed) of the ownship and intruder, and return a warning if the ownship's current heading puts it on course for a near midair collision (NMAC). There are four input variables describing the scenario (Figure 4) and nine possible advisories corresponding to nine output dimensions. Each output is assigned a real-valued *reward*. The maximum reward advisory indicates the safest warning given the current intruder geometry. The three most prominent advisories are clear of conflict (COC), descend at a rate $\geq$ 1500 ft/min (DES1500), and climb at a rate $\geq$ 1500 ft/min (CLI1500). We train a BNN with one hidden layer

Method	Property	P_{safe}	Time (s)	Num. Samples
IBP	ϕ_1	0.9739	136	10500
	ϕ_2	0.9701	117	9000
	ϕ_3	0.9999	26	2000
	ϕ_4	0.9999	26	2000
LBP	ϕ_1	0.9798	723	10500
	ϕ_2	0.9867	628	9000
	ϕ_3	0.9999	139	2000
	ϕ_4	0.9999	148	2000

Table 1: VCAS probabilistic safety. We see that, though LBP is 5 times slower than IBP, it gives slightly tighter lower bounds, similarly to the observations in Figure 2.

with 512 hidden neurons that focuses on the situation in which the ownship's previous warning was COC, where we would like to predict if the intruder has moved into a position which requires action. This scenario is represented by roughly 5 million entries in the VCAS dataset and training our BNN with VI results in test accuracy of 91%. We use probabilistic safety to evaluate whether the network is robust to four properties, referred to as ϕ_1, ϕ_2, ϕ_3 and ϕ_4, which comprise a probabilistic extension of those considered for NNs in [16, 29]. Properties ϕ_1 and ϕ_2 test the consistency of DES1500 and CLI1500 advisories: given a region in the input space, ϕ_1 and ϕ_2 ensure that the output is constrained such that DES1500 and CLI1500 are the maximal advisories for all points in the region, respectively. On the other hand, ϕ_3 and ϕ_4 test that, given a hyper-rectangle in the input space, no point in the hyper-rectangle causes DES1500 or CLI1500 to be the maximal advisory. The properties we test are depicted in the centre and right plot of Figure 4. In Table 1 we report the results of the verification of the above properties, along with their computational times and the number of weights sampled for the verification. Our implementation of Algorithm 1 with both LBP and IBP is able to compute a lower bound for probabilistic safety with these properties in a few hundreds of seconds.[5] Notice that, also in this case, LBP gives tighter bounds than IBP, though it takes around 5 times longer to run, which is a similar trade-off to what is observed for deterministic NNs [32].

Image Classification on MNIST We train several BNNs on MNIST to study how our method scales with the number of neurons, considering networks that are hundreds of times larger than those for the VCAS dataset. For this analysis we consider image classification with the MNIST dataset. In order to model manipulations of an image we use the l_∞-norm ϵ-ball around

[5]Note that in the case of ϕ_1 and ϕ_2 the input set T is composed of three disjoint boxes. Our framework can be used on such sets by computing probabilistic safety on each box and then combining the results together via the union bound.

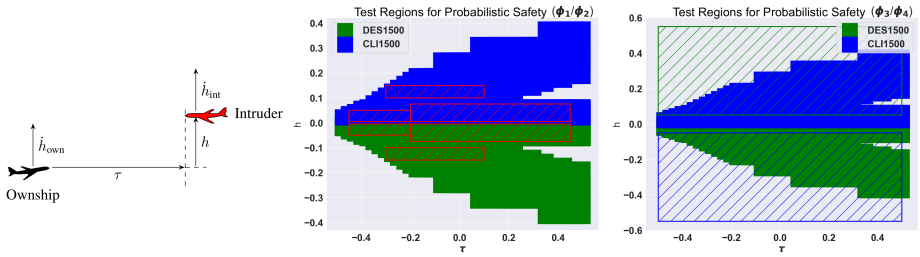

Figure 4: VCAS encounter geometry and properties under consideration. Left: Taken from [15], a visualization of the encounter geometry and the four variables that describe it (distance τ, ownship heading $\dot{h}_{\text{own}}$, intruder heading $\dot{h}_{\text{int}}$, and vertical separation h). Center: Visualization of ground truth labels (in color); red boxes indicate hyper-rectangles that make up the input areas for property ϕ_1 (red boxes in the blue area) and ϕ_2 (red boxes in the green area). Right: Hyper-rectangle for visualization of properties ϕ_3 and ϕ_4: we ensure that DES1500 is not predicted in the green striped box and CLI1500 is not predicted in the blue striped box.

test points. For all manipulations of magnitude up to ϵ, we would like to check that the classification remains the same as that given by the (argmax of the) posterior predictive distribution. This can be done by first taking the classification according to the posterior on the test point x^*. Let i be the predicted class index, then we create a $n_{\text{C}} \times n_{\text{C}}$ matrix, C_S, where the ith column is populated with ones and the diagonal is populated with negative ones, save for the ith entry which is set to one. Finally, ensuring that all entries of the vectors given by $C_S f^w(x)$ for all $x \in T$ are positive indicates that the classification does not change.

Evaluation. Using IBP, we are able to certify that the probabilistic safety of more than half of the tested 100 images is greater than 0.9 in the case of a 2 hidden layer BNN with 256 nodes per layer. In Figure 5, we compare the lower bounds obtained with our approach with an empirical estimate obtained by sampling 500 posterior weights from the BNN posterior, so as to evaluate the tightness of the bounds obtained in practice. The results are given for the average over the same 100 images employed for the results reported in Figure 5. For 1 layer BNNs we use $\epsilon = 0.025$ and for 2 layer BNNs we use $\epsilon = 0.001$. Notice that tight bounds can be obtained even for BNNs with almost a million of parameters, in particular, for a two-layer BNN with 512 neurons per layer, we have that our bound is within 5% of the empirical results.

7 CONCLUSION

We considered probabilistic safety for BNNs, a worst-case probabilistic adversarial robustness measure, which can be used to certify a BNN against adversarial perturbations. We developed an algorithmic framework for the computation of probabilistic safety based on techniques from non-convex optimization, which computes a certified lower bound of the measure. On experiments on various datasets we showed that our methods allows one to certify BNNs with general activation functions, multiple layers, and millions of parameters.

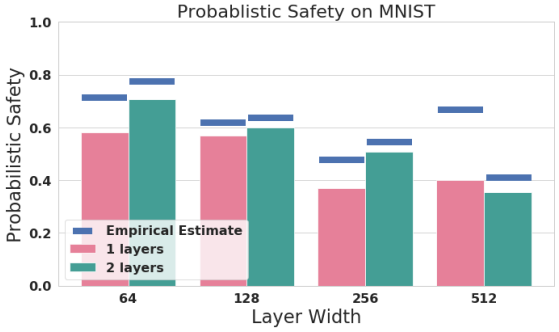

Figure 5: Performance of our framework on BNN architectures with varying numbers of neurons. The height of each bar represents the mean of the probabilistic safety value computed on 100 test set images.

Acknowledgements This project was funded by the EU's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie (grant agreement No. 722022), the ERC under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 834115) and the EPSRC Programme Grant on Mobile Autonomy (EP/M019918/1).

References

[1] Abate, A., Prandini, M., Lygeros, J., and Sastry, S. (2008). Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734.

[2] Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. In *NeurIPS*.

[3] Biggio, B. and Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331.

[4] Blaas, A., Laurenti, L., Patane, A., Cardelli, L., Kwiatkowska, M., and Roberts, S. (2020). Adversarial robustness guarantees for classification with gaussian processes. *AISTATS*.

[5] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *ICML*.

[6] Carbone, G., Wicker, M., Laurenti, L., Patane, A., Bortolussi, L., and Sanguinetti, G. (2020). Robustness of Bayesian neural networks to gradient-based attacks. *arXiv preprint arXiv:2002.04359*.

[7] Cardelli, L., Kwiatkowska, M., Laurenti, L., Paoletti, N., Patane, A., and Wicker, M. (2019). Statistical guarantees for the robustness of Bayesian neural networks. *IJCAI*.

[8] Cardelli, L., Kwiatkowska, M., Laurenti, L., and Patane, A. (2018). Robustness guarantees for Bayesian inference with Gaussian processes. In *AAAI*.

[9] Dvijotham, K., Garnelo, M., Fawzi, A., and Kohli, P. (2018). Verification of deep probabilistic models. *arXiv preprint arXiv:1812.02795*.

[10] Emtiyaz Khan, M., Immer, A., Abedi, E., and Korzepa, M. (2019). Approximate inference turns deep networks into Gaussian processes. *NeurIPS*.

[11] Gal, Y. (2016). *Uncertainty in deep learning*. PhD thesis, University of Cambridge.

[12] Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059.

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

[14] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

[15] Julian, K. D. and Kochenderfer, M. J. (2019). Guaranteeing safety for neural network-based aircraft collision avoidance systems. *DASC*.

[16] Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In *CAV*.

[17] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? In *NeurIPS*.

[18] Laurenti, L., Lahijanian, M., Abate, A., Cardelli, L., and Kwiatkowska, M. (2020). Formal and efficient synthesis for continuous-time linear stochastic hybrid processes. *IEEE Transactions on Automatic Control*.

[19] Liu, X., Li, Y., Wu, C., and Hsieh, C.-J. (2019). Adv-bnn: Improved adversarial defense through robust Bayesian neural network. *ICLR*.

[20] MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

[21] Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). A simple baseline for Bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143.

[22] McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part i convex underestimating problems. *Mathematical programming*, pages 147–175.

[23] Michelmore, R., Wicker, M., Laurenti, L., Cardelli, L., Gal, Y., and Kwiatkowska, M. (2019). Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. *ICRA*.

[24] Neal, R. M. (2012). *Bayesian learning for neural networks*. Springer Science & Business Media.

[25] Rawat, A., Wistuba, M., and Nicolae, M.-I. (2017). Adversarial phenomenon in the eyes of Bayesian deep learning. *arXiv preprint arXiv:1711.08244*.

[26] Ruan, W., Huang, X., and Kwiatkowska, M. (2018). Reachability analysis of deep neural networks with provable guarantees. *IJCAI*.

[27] Smith, L. and Gal, Y. (2018). Understanding measures of uncertainty for adversarial example detection.

[28] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. *ICLR*.

[29] Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. (2018). Formal security analysis of neural networks using symbolic intervals. In *USENIX Security 18*.

[30] Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Boning, D., Dhillon, I. S., and Daniel, L. (2018). Towards fast computation of certified robustness for relu networks. *ICML*.

[31] Wicker, M., Laurenti, L., Patane, A., and Kwiatkowska, M. (2020). Probabilistic safety for Bayesian neural networks. *arXiv:2004.10281*.

[32] Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. (2018). Efficient neural network robustness certification with general activation functions. In *NeurIPS*, pages 4939–4948.