Tight Verification of Probabilistic Robustness in Bayesian Neural Networks

Ben Batten[*]
Imperial College London

Mehran Hosseini[*]
King's College London

Alessio Lomuscio
Imperial College London

Abstract

We introduce two algorithms for computing tight guarantees on the probabilistic robustness of Bayesian Neural Networks (BNNs). Computing robustness guarantees for BNNs is a significantly more challenging task than verifying the robustness of standard Neural Networks (NNs) because it requires searching the parameters' space for safe weights. Moreover, tight and complete approaches for the verification of standard NNs, such as those based on Mixed-Integer Linear Programming (MILP), cannot be directly used for the verification of BNNs because of the polynomial terms resulting from the consecutive multiplication of variables encoding the weights. Our algorithms efficiently and effectively search the parameters' space for safe weights by using iterative expansion and the network's gradient and can be used with any verification algorithm of choice for BNNs. In addition to proving that our algorithms compute tighter bounds than the SoA, we also evaluate our algorithms against the SoA on standard benchmarks, such as MNIST and CIFAR10, showing that our algorithms compute bounds up to 40% tighter than the SoA.[†]

1 INTRODUCTION

In recent years *Neural Networks* (*NNs*) have been proposed to perform safety-critical functions in applications such as automated driving, medical image processing and beyond (Abiodun et al., 2018; Anwar et al., 2018; Bojarski et al., 2016; Liu et al., 2021b). For machine learning methods to be adopted in these areas, assurance and certification are paramount.

Following the known vulnerabilities of NNs to adversarial attacks (Dalvi et al., 2004; Szegedy et al., 2014; Ilyas et al., 2019; Sharif et al., 2016; Goodfellow et al., 2015), a rapidly growing body of research has focused on methods to verify the safety of NNs (Pulina and Tacchella, 2010; Lomuscio and Maganti, 2017; Katz et al., 2017; Madry et al., 2018; Liu et al., 2021a; Zhang et al., 2023) to identify fragilities before deployment.

To mitigate NN vulnerabilities to adversarial attacks, several methods have been proposed, including adversarial training (Madry et al., 2018; Tramèr et al., 2018; Ganin et al., 2017; Shafahi et al., 2019), and verification-based augmentation (Dreossi et al., 2019). While considerable progress has been made in these areas, present models remain fragile to various input perturbations, thereby limiting their potential applicability in key areas.

In an attempt to overcome this problem, architectures that are inherently more robust have been proposed. Bayesian Neural Networks (BNNs) have arguably emerged as a key class of models intrinsically more resilient to adversarial attacks (Pang et al., 2021; Carbone et al., 2020). This is because BNNs naturally incorporate the randomness present in the posterior distribution (Uchendu et al., 2021).

Although BNNs have been shown to provide a degree of natural robustness, it is crucial to verify their safety and correctness with respect to specifications of interest, including robustness, before deployment. Note that while safety is formulated as a standard decision problem in deterministic NNs, for BNNs, safety is a probabilistic concept. Specifically, given a BNN, an input region and a safe set, the probabilistic robustness problem concerns determining the probability that the input neighbourhood is mapped to a safe set by the BNN (Cardelli et al., 2019a,b; Wicker et al., 2020).

The probabilistic robustness problem for BNNs is also sometimes referred to as the *probabilistic safety prob-*

[*]Equal contribution.

[†]The code is available at `https://github.com/benbatten/TightVerificationOfProbabilisticRobustnessInBNNs`.

lem (Cardelli et al., 2019b). Since the exact computation of probabilistic robustness for BNNs is computationally intractable, several approaches for approximating it have been proposed (Wicker et al., 2020; Berrada et al., 2021; Lechner et al., 2021).

A key challenge in this area is to approximate the probabilistic robustness of BNNs tightly and efficiently. As we demonstrate in Sections 3 and 4, existing methods are either loose or computationally expensive. It therefore remains of great interest to explore new approaches that can derive tighter approximations to probabilistic robustness while remaining computationally effective. To this end, our contributions in this paper can be summarised as follows.

- We introduce two new algorithms, *Pure Iterative Expansion* (*PIE*) and *Gradient-guided Iterative Expansion* (*GIE*). We show that they produce sound lower bounds on the probabilistic robustness of BNNs, which are provably tighter than SoA sampling-based approaches.

- We empirically evaluate our algorithms on MNIST and CIFAR10 benchmarks against the SoA and show that our approach produces tighter approximations of the probabilistic robustness.

- For reproducibility, we provide our code in the supplementary material.

The rest of the paper is organised as follows. In Subsection 1, we review the related literature on NN and BNN verification. In Section 2, we formally define BNNs, the probabilistic robustness problem for BNNs, and other related terminology that we use throughout the paper. We discuss the details of our approach in Section 3 by introducing the PIE and GIE algorithms for exploring the set of safe weights and computing lower bounds for the probabilistic robustness of BNNs in Subsection 3.2. We also formally show that our approach outperforms the SoA sampling-based method in Subsection 3.3 by producing provably tighter bounds. We evaluate the performance of our approach against SoA on the MNIST and CIFAR10 datasets in Section 4.

Related Work

Mixed-Integer Linear Programming (*MILP*) and *Bound Propagation* (*BP*) are two common techniques used for the verification of standard NNs. In MILP-based approaches, the verification problem for NNs is encoded as an MILP problem (Lomuscio and Maganti, 2017; Hosseini and Lomuscio, 2023; Tjeng et al., 2019), which can be solved using MILP solvers. A common assumption in MILP-based approaches is that activations are piecewise-linear.

Propagation-based approaches differ from MILP-based approaches in that they consider a relaxed network and propagate bounds on each node's output through the network's layers to solve the verification problem (Gowal et al., 2018; Singh et al., 2019, 2018; Wang et al., 2018).

SoA approaches for the verification of NNs combine MILP and BP to achieve complete and scalable verification for NNs. For instance, (Kouvaros and Lomuscio, 2021) uses bound propagation alongside *adaptive splitting* (Henriksen and Lomuscio, 2020; Botoeva et al., 2020) to reduce the number of binary variables in the resulting MILP problem and thus further speed up the verification process. An important and well-studied safety specification in NN verification is *robustness*, i.e., a model's resilience in returning the same output for all values in a small neighbourhood of a given input. An example approach devoted only to checking robustness is (Cardelli et al., 2019b), where they compute upper bounds on the robustness of BNNs. Closely related to this, (Cardelli et al., 2019a) considers the problem of computing the probability of whether, for a given input to a BNN, there exists another input within a bounded set around the input such that the network's prediction differs for the two points and provides an approach to approximate this probability and compute statistical guarantees for it.

In the same line of research (Carbone et al., 2020) shows that the vulnerability of BNNs to gradient-based attacks arises as a result of the data lying on a lower-dimensional submanifold of the ambient space.

(Wicker et al., 2020) considers a set-up similar to this paper, using a sampling-based approach to bound the probabilistic robustness of BNNs. In this approach, they sample orthotopes from the posterior distribution and use *Interval Bound Propagation* (*IBP*) and *Linear Bound Propagation* (*LBP*) to determine whether a given orthotope is safe. By repeating this process, they obtain a robustness certificate and a lower bound for the probabilistic robustness of the BNN.

Berrada et al. (2021) consider the *adversarial robustness* of BNNs. They use the notion of Lagrangian duality, which replaces standard Lagrangian multipliers with functional multipliers. For an optimal choice of functional multipliers, this approach leads to exact verification, i.e., the exact computation of probabilistic robustness. Nevertheless, this may not be practical. The authors show that for specific classes of multipliers, however, one can obtain lower bounds for the probabilistic robustness.

Adams et al. (2023) introduced an algorithm to verify the adversarial robustness of BNNs. They use Dynamic Programming alongside bound propagation and convex relaxations to compute lower and upper bounds on BNN predictions for a given input set. Wicker et al. (2023) build on the algorithm first presented in (Wicker et al., 2020) to provide upper bounds on the probabilistic robustness of BNNs in addition to lower bounds. Moreover, they extend the algorithmic formulation in (Wicker et al., 2020) to handle the specification proposed in (Berrada et al., 2021) on posterior distributions with unbounded support.

(Lechner et al., 2021) uses MILP to obtain robustness guarantees for infinite time-horizon safety and robustness of BNNs. However, this approach can only be used for small networks. (Wijk et al., 2022) obtains robustness guarantees for credal BNNs via constraint relaxation over probabilistic circuits.

2 PRELIMINARIES

Throughout this paper, we denote the set of real numbers by $\mathbb{R}$, use italics, w, to refer to scalers in $\mathbb{R}$, bold italics, $\boldsymbol{w}$, for vectors in $\mathbb{R}^n$, and bold letters, $\mathbf{w}$, to refer to random variables in $\mathbb{R}^n$. For an orthotope $\mathcal{O} \subset \mathbb{R}^n$ with two opposing corners $\boldsymbol{w}_1, \boldsymbol{w}_2$, we denote $\mathcal{O}$ by $[\boldsymbol{w}_1, \boldsymbol{w}_2]$.

In the rest of this section, we define BNNs and briefly discuss common approaches for training them in Subsection 2.1; we then define the robustness problem for BNNs and a few related concepts, which we will use in Section 3, in Subsection 2.2.

2.1 Bayesian Neural Networks

Definition 2.1 (Bayesian Neural Network). Given a dataset, $\mathcal{D}$, let $f_{\mathbf{w}} : \mathbb{R}^m \to \mathbb{R}^n$ be a *Bayesian neural network* with $\mathbf{w}$ a vector random variable consisting of the parameters of the network. Assuming a prior distribution over the parameters, $\mathbf{w} \sim p(\boldsymbol{w})$, training amounts to computing the posterior distribution, $p(\boldsymbol{w}|\mathcal{D})$, by the application of Baye's rule. For a vector, $\boldsymbol{w}$, sampled from the posterior, $p(\boldsymbol{w}|\mathcal{D})$, we denote the deterministic neural network with parameters $\boldsymbol{w}$ by $f_{\boldsymbol{w}}$. We use $n_{\boldsymbol{w}}$ to refer to the dimension of the parameters space of the network.

In BNN literature, it is customary to assume the prior and the posterior both have normal distribution; nevertheless, our *Pure Iterative Expansion* (*PIE*) algorithm works for arbitrary distributions, and our *Gradient-guided Iterative Expansion* (*GIE*) algorithm works for all continuous distributions. In the case of normal distribution, training requires finding the mean

$\boldsymbol{\mu} \in \mathbb{R}^{n_{\boldsymbol{w}}}$ and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{n_{\boldsymbol{w}} \times n_{\boldsymbol{w}}}$ of the parameters such that $p(\boldsymbol{w}|\mathcal{D}) \approx q(\boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We refer to the parameters' variance by $\boldsymbol{\sigma}^2 = \mathrm{diag}(\boldsymbol{\Sigma})$. When the parameters are independent, i.e., when the covariance matrix $\boldsymbol{\Sigma}$ is diagonal, we denote a normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ by $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. Similarly, we denote an arbitrary probability distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ by $\mathcal{P}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$.

Despite the similarities between standard, deterministic NNs and BNNs, training BNNs poses a greater challenge since computing the posterior of a BNN is computationally infeasible due to non-linearities in the network (MacKay, 1992). Some of the approaches for calculating the posterior distribution include *Variational Inference* (*VI*) (Blundell et al., 2015), *Markov Chain Monte Carlo* (*MCMC*) (Hastings, 1970), combinations of VI and MCMC (Salimans et al., 2015), or other approximate approaches. For a comparison of the approaches see (Jospin et al., 2022).

2.2 The Probabilistic Robustness Problem for BNNs

Let us start by defining the probabilistic robustness problem for BNNs.

Definition 2.2 (Probabilistic Robustness Problem). Given a BNN $f_{\mathbf{w}} : \mathbb{R}^m \to \mathbb{R}^n$, a *robustness specification* for $f_{\mathbf{w}}$ is a tuple $\varphi = \langle \mathcal{X}, \mathcal{S} \rangle$, where $\mathcal{X} \subseteq \mathbb{R}^m$ is an ϵ-ball $B_\epsilon(\boldsymbol{x})$ around a given $\boldsymbol{x} \in \mathbb{R}^m$ with respect to a norm $\|\cdot\|_p$ and $\mathcal{S} \subseteq \mathbb{R}^n$ is a half-space

$$\mathcal{S} = \{\boldsymbol{y} : \boldsymbol{a}^\mathsf{T} \cdot \boldsymbol{y} \geq b\}, \tag{1}$$

for some $\boldsymbol{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$. The *probabilistic robustness problem* concerns determining the probability that for all inputs in $\mathcal{X}$ the output of $f_{\mathbf{w}}$ is in $\mathcal{S}$, i.e.,

$$P_{\mathrm{safe}}(\mathcal{X}, \mathcal{S}) = P_{\boldsymbol{w} \sim \mathbf{w}}(\forall \boldsymbol{x} \in \mathcal{X}, f_{\boldsymbol{w}}(\boldsymbol{x}) \in \mathcal{S}).$$

We refer to $\mathcal{X}$ as the *input constraint* set and $\mathcal{S}$ as the *output constraint* set. The probabilistic robustness problem is sometimes referred to as the *probabilistic safety problem* (Cardelli et al., 2019b) or the *adversarial robustness problem* (Berrada et al., 2021) for BNNs; in the same light, φ is also referred to as the *safety specification*.

For a classification task, when the goal is to verify for a correct class c against a target class c', we fix $\boldsymbol{a}_c = 1$, $\boldsymbol{a}_{c'} = -1$, and $\boldsymbol{a}_i = 0$ for, $i \neq c, c'$ in Equation (1).

Definition 2.3 (Set of Safe Weights). The *set of safe weights* for a BNN with respect to a specification $\varphi = \langle \mathcal{X}, \mathcal{S} \rangle$, is defined as

$$\mathcal{C} = \{\boldsymbol{w} \in \mathbb{R}^{n_{\boldsymbol{w}}} : \forall \boldsymbol{x} \in \mathcal{X}, f_{\boldsymbol{w}}(\boldsymbol{x}) \in \mathcal{S}\}. \tag{2}$$

Proposition 2.1. Given a BNN $f_{\mathbf{w}}$, such that $\mathbf{w} \sim q(\boldsymbol{w})$ for a probability density function q, and a safety specification $\langle \mathcal{X}, \mathcal{S} \rangle$, we have that

$$P_{\text{safe}}(\mathcal{X}, \mathcal{S}) = \int_{\mathcal{C}} q(\boldsymbol{w}) d\boldsymbol{w}.$$

Since solving the robustness problem for deterministic networks is NP-complete (Katz et al., 2017; Sinha et al., 2017), verifying whether a given $\boldsymbol{w} \in \mathcal{C}$ is also NP-complete. Therefore, it is infeasible to calculate P_{safe} exactly even for moderately sized networks. In the rest of this paper we put forward two algorithms for calculating lower-bound approximations of P_{safe} in a provably tighter way than previous approaches.

3 COMPUTING THE PROBABILISTIC ROBUSTNESS FOR BNNS

In Subsection 2.2 we discussed why calculating P_{safe} is computationally infeasible. Besides empirical sampling, the current approaches for computing P_{safe} rely on sampling orthotopes $\hat{\mathcal{C}}_i \subseteq \mathbb{R}^{n_{\boldsymbol{w}}}$ with fixed sizes, whose edges are proportional to the standard deviations of the weights of the network, and verifying whether $f_{\hat{\mathcal{C}}_i}(\mathcal{X}) \subseteq \mathcal{S}$ through *Linear Bound Propagation* (LBP) (Wicker et al., 2020). This enables the computation of an under-approximation $\hat{\mathcal{C}} = \bigcup \hat{\mathcal{C}}_i \subseteq \mathcal{C}$, which, in turn, allows computing the lower bound

$$p_{\text{safe}} = \int_{\hat{\mathcal{C}}} q(\boldsymbol{w}) d\boldsymbol{w} \leq P_{\text{safe}}(\mathcal{X}, \mathcal{S}).$$

One weakness of this approach is that it uses orthotopes of the same size for all specifications. For example, whilst for a given input to a BNN it may be possible to fit large orthotopes around the weights's mean and use LBP to verify the safety of these orthotopes, for another input to the same network this may not necessarily be the case, and must resort to smaller orthotopes, which in high dimensions result in obtaining minuscule probabilistic robustness lower bounds, a phenomenon demonstrated in Example 3.2.

To mitigate this problem, we introduce two approaches that allow us to dynamically adjust the size of orthotopes $\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_s$. The *Pure Iterative Expansion* (*PIE*) approach, which we introduce in Subsection 3.1, allows us to dynamically *scale* the size of and orthotope $\hat{\mathcal{C}}_i$, and therefore, cover a larger volume in $\mathcal{C}$.

The more sophisticated *Gradient-guided Iterative Expansion* (*GIE*) approach, which we introduce in Subsection 3.2, not only allows the use of dynamic scaling, but also expands the orthotopes further in the dimen-

sions that allow $\hat{\mathcal{C}}_i$'s to remain in $\mathcal{C}$, according to the posterior's gradient, $\nabla_{\mathbf{w}} f_{\mathbf{w}}$.

Both PIE and GIE can be accompanied by LBP or other approaches. Since our iterative expansion approach depends on iteratively expanding $\hat{\mathcal{C}}_i$'s and verifying their safety, we use LBP, as described and implemented in (Wicker et al., 2020), because of its significant computational efficiency.

3.1 Pure Dynamic Scaling

Given a BNN, $f_{\mathbf{w}}$, and safety specification, $\varphi = \langle \mathcal{X}, \mathcal{S} \rangle$, remember that we denote the set of safe weights of $f_{\boldsymbol{w}}$ with respect to φ as $\mathcal{C}$. The pure sampling and LBP approach (Wicker et al., 2020) starts by the under-approximation $\hat{\mathcal{C}} = \{\}$ of $\mathcal{C}$ and samples $\boldsymbol{w}_0 \sim q(\boldsymbol{w})$. It then defines $\hat{\mathcal{C}}_0 = [\boldsymbol{w}_0 \pm \lambda \boldsymbol{\sigma}]$, where $\lambda > 0$ is the scaling factor and $\boldsymbol{\sigma}$ is the parameters' standard deviation vector. Then, using IBP or LBP, it checks whether $\hat{\mathcal{C}}_0 \subseteq \mathcal{C}$. If this is the case, $\hat{\mathcal{C}} = \hat{\mathcal{C}} \cup \hat{\mathcal{C}}_0$; otherwise, $\hat{\mathcal{C}}_0$ is discarded. This process is repeated until a given number of weights s have been tried.

As we briefly discussed in the beginning of Section 3 and demonstrate in Section 4, the lower bounds computed using a pure sampling approach are not tight, because of the rigidity of the sampled orthotopes.

The first and most straightforward solution to this problem that we propose is to instead use dynamically-scaled orthotopes, as outlined in Algorithm 1. Wherein, after sampling an orthotope $\hat{\mathcal{C}}_i = [\boldsymbol{w}_i \pm \lambda \boldsymbol{\sigma}] \subseteq \mathcal{C}$ and verifying it (line 8 in Algorithm 1), we expand the orthotope to $\hat{\mathcal{C}}_i = [\boldsymbol{w}_i \pm 2\lambda \boldsymbol{\sigma}]$ rather than immediately moving on to the next sampled parameter vector, as in the pure sampling approach. After the initial expansion, we again check $\hat{\mathcal{C}}_i \subseteq \hat{\mathcal{C}}$. If this is the case, we can repeat the same process for $\hat{\mathcal{C}}_i = [\boldsymbol{w}_i \pm 3\lambda \boldsymbol{\sigma}]$. We repeat this, until $\hat{\mathcal{C}}_i \not\subseteq \mathcal{C}$.

Example 3.1 demonstrates the advantage of the PIE approach over the pure sampling-based approach.

Example 3.1. Consider a BNN with two Bayesian layers, $f_{\boldsymbol{w}}(x) = \text{ReLU}(w_2 \cdot \text{ReLU}(w_1 \cdot x))$, where $w, w_2 \sim \mathcal{N}(0, 1)$ and no bias terms. For the input constraint set $\mathcal{X} = \{1\}$ and the output constraint set $\mathcal{S} = \{y \in \mathbb{R} : y \leq 1\}$, the set of safe weights is $\mathcal{C} = \left\{ (w_1 \quad w_2)^{\mathsf{T}} \in \mathbb{R}^2 : \text{ReLU}(w_2 \cdot \text{ReLU}(w_1)) \leq 1 \right\}$. The set $\mathcal{C}$ is shown blue in Figures 1a to 1c.

For the same sampled weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_4 \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and scaling factor $\lambda = 1$, areas covered by the pure sampling approach, PIE approach for 2 iterations, and GIE approach (cf. Subsection 3.2) for 2 iterations (see Subsection 3.2) are shown in Figures 1a, 1b, and 1c, respectively. We observe that the GIE approach (Figure 1c) has the greatest coverage of $\mathcal{C}$, followed by the

---

Algorithm 1: Certifying BNNs using PIE

1: **Input:** BNN $f_{\mathbf{w}}$, $\mathbf{w} \sim \mathcal{P}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, safety specification $\varphi = \langle \mathcal{X}, \mathcal{S} \rangle$, number of samples $s \in \mathbb{N}$, and scaling factor $\lambda > 0$.
2: **Output:** $\hat{\mathcal{C}}$ and p_{safe}.
3: $\hat{\mathcal{C}} = \{\}$
4: **for** $i \leftarrow 1$ to s **do**
5: $\quad \boldsymbol{w} \sim \mathcal{P}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
6: $\quad j = 1$
7: $\quad \bar{\mathcal{C}} = \{\}$
8: $\quad$ **while** $LBP([\boldsymbol{w} \pm j\lambda\boldsymbol{\sigma}], \mathcal{X}) \subseteq \mathcal{S}$ **do**
9: $\quad\quad \bar{\mathcal{C}} = [\boldsymbol{w} \pm j\lambda\boldsymbol{\sigma}]$
10: $\quad\quad j = j + 1$
11: $\quad$ **end while**
12: $\quad \hat{\mathcal{C}} = \hat{\mathcal{C}} \cup \bar{\mathcal{C}}$
13: **end for**
14: **Return** $\hat{\mathcal{C}}$ and $\int_{\hat{\mathcal{C}}} q(\boldsymbol{w}) d\boldsymbol{w}$

Algorithm 2: Certifying BNNs using GIE

1: **Input:** BNN $f_{\mathbf{w}}$, $\mathbf{w} \sim \mathcal{P}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, safety specification $\varphi = \langle \mathcal{X}, \mathcal{S} \rangle$, number of samples $s \in \mathbb{N}$, scaling factor $\lambda > 0$, and gradient-based scaling factor $\rho > 0$.
2: **Output:** $\hat{\mathcal{C}}$ and p_{safe}.
3: $\hat{\mathcal{C}} = \{\}$
4: **for** $i \leftarrow 1$ to s **do**
5: $\quad \boldsymbol{w} \sim \mathcal{P}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
6: $\quad \mathcal{E}^- = \{\boldsymbol{e}_i \in \mathbb{R}^{n_w} : \nabla_{\mathbf{w}}(\boldsymbol{a}^\intercal \cdot f_{\boldsymbol{w}})(\boldsymbol{x}) \cdot \boldsymbol{e}_i < 0\}$
7: $\quad \mathcal{E}^+ = \{\boldsymbol{e}_i \in \mathbb{R}^{n_w} : \nabla_{\mathbf{w}}(\boldsymbol{a}^\intercal \cdot f_{\boldsymbol{w}})(\boldsymbol{x}) \cdot \boldsymbol{e}_i > 0\}$
8: $\quad \mathcal{E}^0 = \{\boldsymbol{e}_i \in \mathbb{R}^{n_w} : \nabla_{\mathbf{w}}(\boldsymbol{a}^\intercal \cdot f_{\boldsymbol{w}})(\boldsymbol{x}) \cdot \boldsymbol{e}_i = 0\}$
9: $\quad \boldsymbol{v}^+ = \lambda\boldsymbol{\sigma}(1 + \rho \sum_{\boldsymbol{e}_i \in \mathcal{E}^+ \cup \mathcal{E}^0} \boldsymbol{e}_i)$
10: $\quad \boldsymbol{v}^- = \lambda\boldsymbol{\sigma}(1 + \rho \sum_{\boldsymbol{e}_i \in \mathcal{E}^- \cup \mathcal{E}^0} \boldsymbol{e}_i)$
11: $\quad j = 1$
12: $\quad \bar{\mathcal{C}} = \{\}$
13: $\quad$ **while** $LBP([\boldsymbol{w}_i - j\boldsymbol{v}^-, \boldsymbol{w}_i + j\boldsymbol{v}^+], \mathcal{X}) \subseteq \mathcal{S}$ **do**
14: $\quad\quad \bar{\mathcal{C}} = [\boldsymbol{w}_i - j\boldsymbol{v}^-, \boldsymbol{w}_i + j\boldsymbol{v}^+]$
15: $\quad\quad j = j + 1$
16: $\quad$ **end while**
17: $\quad \hat{\mathcal{C}} = \hat{\mathcal{C}} \cup \bar{\mathcal{C}}$
18: **end for**
19: **Return** $\hat{\mathcal{C}}$ and $\int_{\hat{\mathcal{C}}} q(\boldsymbol{w}) d\boldsymbol{w}$

---

the PIE approach (Figure 1b), which has greater coverage than the pure sampling approach (Figure 1a).

3.2 Gradient-Guided Dynamic Scaling

The goal of this subsection is to improve Algorithm 1 even further by using the gradient of the network, $\nabla_{\mathbf{w}} f_{\mathbf{w}}$, to maximise our coverage of the safe set of weights, $\mathcal{C}$, as summarised in Algorithm 2. Recall that for a given point in the parameter space, $\boldsymbol{w} \in \mathcal{C}$, our goal is to explore $\mathcal{C}$ starting from $\boldsymbol{w}$. Instead of expanding $\hat{\mathcal{C}}_i$ uniformly in all dimensions, we can use the parameter-wise BNN gradient, $\nabla_{\mathbf{w}} f_{\mathbf{w}}$, to expand $\hat{\mathcal{C}}_i$ further in dimensions that do not violate $\hat{\mathcal{C}}_i \subseteq \mathcal{C}$.

From Equation (2), we have that

$$\mathcal{C} = \{\boldsymbol{w} \in \mathbb{R}^{n_w} : \boldsymbol{a}^\intercal \cdot f_{\boldsymbol{w}}(B_\epsilon(\boldsymbol{x})) \geq b\},$$

for some $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{x} \in \mathbb{R}^m$, $b \in \mathbb{R}$, and $\epsilon > 0$. To maximise the volume of a candidate orthotope, $\hat{\mathcal{C}}_i$, while staying within the bounds of $\mathcal{C}$, we propose expanding further in directions with *zero* or *positive gradients* in $\nabla_{\boldsymbol{w}} \boldsymbol{a}^\intercal \cdot f_{\mathbf{w}}(\boldsymbol{x})$. Concretely, let

$$\mathcal{E}^- = \{\boldsymbol{e}_i \in \mathbb{R}^{n_w} : \nabla_{\mathbf{w}}(\boldsymbol{a}^\intercal \cdot f_{\boldsymbol{w}})(\boldsymbol{x}) \cdot \boldsymbol{e}_i < 0\},$$
$$\mathcal{E}^+ = \{\boldsymbol{e}_i \in \mathbb{R}^{n_w} : \nabla_{\mathbf{w}}(\boldsymbol{a}^\intercal \cdot f_{\boldsymbol{w}})(\boldsymbol{x}) \cdot \boldsymbol{e}_i > 0\},$$
$$\mathcal{E}^0 = \{\boldsymbol{e}_i \in \mathbb{R}^{n_w} : \nabla_{\mathbf{w}}(\boldsymbol{a}^\intercal \cdot f_{\boldsymbol{w}})(\boldsymbol{x}) \cdot \boldsymbol{e}_i = 0\},$$

where $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{n_w}\}$ is the standard basis for the $\mathbb{R}^{n_w}$.

Intuitively, we can cover more of $\mathcal{C}$ by moving further in directions that take us away from the boundary at $\boldsymbol{a}^\intercal \cdot f_{\boldsymbol{w}}(\boldsymbol{x}) = b$. In particular, when the activations considered are ReLU, where we have that $\nabla_{\mathbf{w}}(\boldsymbol{a}^\intercal \cdot f_{\boldsymbol{w}_i})(\boldsymbol{x}) \cdot \boldsymbol{e}_i = 0$ for all i's corresponding to a weight feeding directly to an inactive node. Moreover, the value of a node is a locally linear function of the weights that are input to that node. Therefore, we have the freedom of moving in the directions corresponding to these weights without changing the output of the network, and hence, remaining inside $\mathcal{C}$. We note that this assumption only holds locally, and with larger deviations in the parameter space, the activation pattern of the network will change. However, we observe that assuming local linearity yields a substantial increase in the volume of verifiable $\hat{\mathcal{C}}_i$'s.

With this knowledge, let us define

$$\boldsymbol{e}^+ = \sum_{\boldsymbol{e}_i \in \mathcal{E}^+ \cup \mathcal{E}^0} \boldsymbol{e}_i, \quad \text{and} \quad \boldsymbol{e}^- = \sum_{\boldsymbol{e}_i \in \mathcal{E}^- \cup \mathcal{E}^0} \boldsymbol{e}_i.$$

Now, instead of starting from the orthotope $\hat{\mathcal{C}}_i = [\boldsymbol{w}_i \pm \lambda\boldsymbol{\sigma}]$ as in the PIE approach, we start from $\hat{\mathcal{C}}_i = [\boldsymbol{w}_i - \lambda\boldsymbol{\sigma}(1 + \rho\boldsymbol{e}^-), \boldsymbol{w}_i + \lambda\boldsymbol{\sigma}(1 + \rho\boldsymbol{e}^+)]$, where $\rho > 0$ is the *gradient-based scaling factor*. Then, we check whether $\hat{\mathcal{C}}_i \subseteq \hat{\mathcal{C}}$ (line 12 in Algorithm 2) and iteratively expand $\hat{\mathcal{C}}_i$, similarly to the PIE approach, until $\hat{\mathcal{C}}_i \nsubseteq \mathcal{C}$.

3.3 Theoretical Comparison

Here we compare Algorithms 1 and 2 against the SoA sampling-based approach (Wicker et al., 2020) and show that both algorithms introduced here produce provably tighter lower bounds for the probabilistic ro-
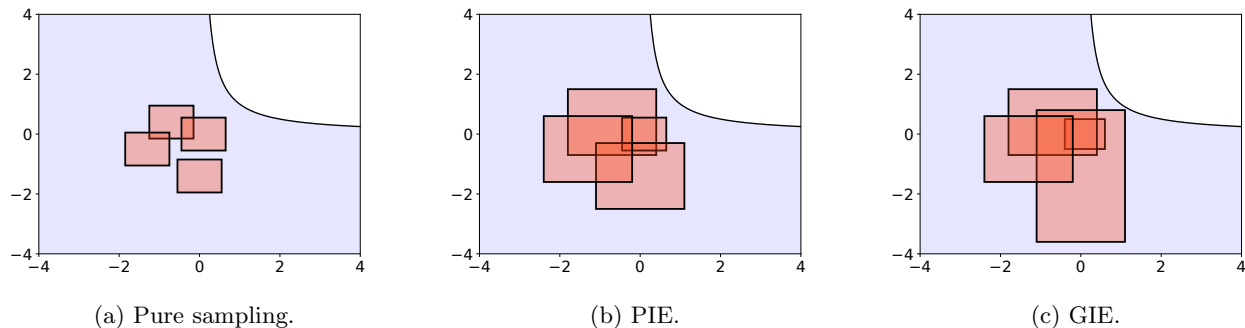
(a) Pure sampling. (b) PIE. (c) GIE.

Figure 1: A comparison between the area of the set of safe weights $\mathcal{C}$ in Example 3.1 that is covered by the pure sampling (1a), PIE after two iterations (1b), and GIE after two iterations (1c) for the same initial weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_4 \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. See Section 4 for detailed comparisons, including comparison with equal compute.

bustness of BNNs. We start this by comparing Algorithms 1 and 2 against the static orthotopes approach of Wicker et al. (2020). We show that Algorithms 1 and 2 always provides tighter lower bounds than the static orthotopes approach. We provide an intuition for the three approaches in Figure 2.

Proposition 3.1. For every BNN, $f_{\mathbf{w}}$, and specification, $\phi = \langle \mathcal{X}, \mathcal{S} \rangle$, let p be the probabilistic robustness of $f_{\mathbf{w}}$ with respect to ϕ, and p_s, p_p, and p_g be the probabilistic robustness lower bounds computed with the same sampled weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_s$ using the pure sampling, PIE, and GIE approaches, respectively. Then we have that $p_s \leq p_p, p_g \leq p$.

Proof. It is trivial that $p_s, p_p, p_g \leq p$. We show that $p_s \leq p_p$. Let $\hat{\mathcal{C}}_s$ and $\hat{\mathcal{C}}_v$ be the robustness certificates computed using $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_s$ by the pure sampling approach (Wicker et al., 2020) and the PIE approach (Algorithm 1), respectively. For each $\boldsymbol{w}_i$, the pure sampling approach checks whether $LBP([\boldsymbol{w}_i \pm \lambda \boldsymbol{\sigma}], \mathcal{X}) \subseteq \mathcal{S}$. This is also checked in line 8 of Algorithm 1. Therefore, if $[\boldsymbol{w}_i \pm \lambda \boldsymbol{\sigma}] \subseteq \hat{\mathcal{C}}_s$, then $[\boldsymbol{w}_i \pm \lambda \boldsymbol{\sigma}] \subseteq \hat{\mathcal{C}}_v$. Thus, $\hat{\mathcal{C}}_s \subseteq \hat{\mathcal{C}}_v$ and $p_s = \int_{\hat{\mathcal{C}}_s} f_{\boldsymbol{w}}(\boldsymbol{x}) d\boldsymbol{w} \leq \int_{\hat{\mathcal{C}}_v} f_{\boldsymbol{w}}(\boldsymbol{x}) d\boldsymbol{w} = p_p$. The proof for $p_s \leq p_g$ is similar. $\square$

Intuitively, one may expect that $p_p \leq p_g$; however, this does not necessarily hold. For instance, if the gradient-based scaling factor ρ is tool large, then $\boldsymbol{w} + j\boldsymbol{v}^+$ may well lie within the set of unsafe weights.

As presented in Section 4, both PIE and GIE allow us to compute bounds for the probabilistic robustness of BNNs that are significantly tighter than the SoA. Before presenting their performance, we would like to emphasise the theoretical significance of the iterative expansion, used in both approaches in Example 3.2.

Example 3.2. Consider a BNN with k nodes and posterior $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, the best lower bound that can be com-

puted using a single orthotope is when the orthotope is centred at $\boldsymbol{\mu}$. For orthotopes used in the pure sampling approach, this is an orthotope of the form $\mathcal{O} = [\boldsymbol{\mu} \pm \lambda \boldsymbol{\sigma}]$ (cf. Algorithm 1). For a fixed $\lambda > 0$, the tightest lower bound that can be obtained for probabilistic robustness is p_λ^k, where $p_\lambda = Prob_{x \sim \mathcal{N}(0,1)}(|x| \leq \lambda)$. This means that the probabilistic robustness approximations computed using only pure sampling vanishes exponentially as the number of parameters of the network increases. This cannot be addressed by increasing the number of sampled orthotopes, as exponentially many orthotopes would be needed.

To better understand this effect, we observe that for a BNN with only 10^3 parameters and $\lambda = 3$, this gives us a lower bound of $p_3^{1,000} \approx 0.9974^{1,000} < 0.08$.

4   EVALUATION

We evaluate the performance of our approach against the previously discussed SoA approach for computing lower bounds for probabilistic robustness (Wicker et al., 2020). As in previous studies, the approach is evaluated on the MNIST dataset (LeCun et al., 2010) and additionally the CIFAR10 dataset (Krizhevsky, 2009). All experiments were run on an AMD Ryzen Threadripper 3970X 32-core with 256GB RAM.

The evaluation is divided into three parts. **(1)** Benchmark against the SoA pure sampling approach in (Wicker et al., 2020). **(2)** Ablation study on the use of gradient-based dynamic scaling factors. **(3)** ablation study on the number of expansion iterations.

For the MNIST baselines, we use 6 fully-connected BNNs from (Wicker et al., 2020). However, we note that these networks have narrow distributions (s.d. of $\approx 10^{-5}$). Therefore, we also use two networks trained for broader posterior distributions (s.d. of $\approx 10^{-3}$) that achieve comparable clean accuracy and
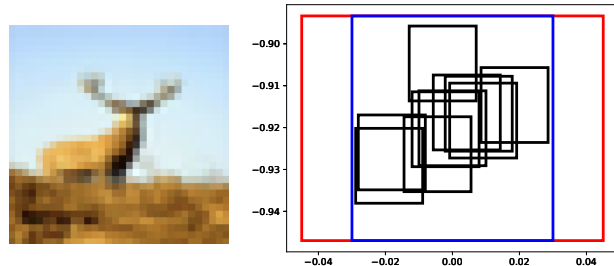
Figure 2: A 2-dimensional cross-section of the safe region $\mathcal{C}$ covered by pure sampling (black), PIE (blue), and GIE (Red) on the same image in CIFAR dataset.

Figure 3: Probabilistic certification as a function of ρ, the gradient-based scaling factor in Algorithm 2.

show greater natural resilience to adversarial attack, a key strength of BNNs (Bortolussi et al., 2022) – we provide the train details in Appendix D. MNIST networks are identified using an ℓxn nomenclature with ℓ, the number of hidden layers, and n, the number of nodes per hidden layer.

We also include a Convolutional Neural Network (CNN) trained on the CIFAR10 dataset. The CNN consists of 5 deterministic convolutional layers followed by 3 fully-connected Bayesian layers with 256, 64, and 10 nodes, respectively. As in (Wicker et al., 2020), all parameter posteriors are Gaussian.

4.1 Comparison of Pure Sampling, PIE, and GIE Approaches

The performances of sampling-based methods are biased by the compute available and the maximum number of iterations. To remove this bias, we enforce an equal limit on the total number of LBP calls per image. The results are presented in Table 1. We note that the GIE approach must also compute network gradients - a step that neither PIE nor the pure sampling approach require. Nevertheless, in practice, we find that the computation of network gradients is much faster than the LBP used in all approaches. We quantify and discuss the impact on runtime in Appendix C. Importantly, we noticed an error in the official implementation in (Wicker et al., 2020) affecting the computation of p_{safe}. We have corrected this error to generate the results presented in Table 1 and provide a detailed breakdown of the mistake in Appendix B.

The results show a significant improvement over the pure sampling method of (Wicker et al., 2020) in all cases with gradient-based scaling providing the tightest guarantees. Moreover, we remark that in obtaining these results the method from (Wicker et al., 2020) depends on careful selection of hyperparameters, namely the orthotope side length, where the use of suboptimal values yields trivially small guarantees – for the results
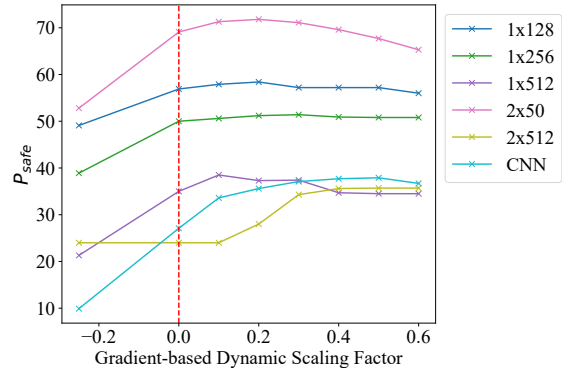
in Table 1 we used grid-search to tune the hyperparameters in the pure sampling approach. Differently, by using a small value for the initial orthotope's side length, our approach is able to adapt to each image and each network with little consideration for tuning hyperparameters.

In summary, the approach here presented considerably outperforms the present SoA in estimating probabilistic robustness leading to bounds that often more than double those in (Wicker et al., 2020).

4.2 Ablation Study on the Use of Gradient-Based Dynamic Scaling Factors

Here we explore the impact of using gradient-based dynamic scaling factors. In Figure 3, we plot the *probabilistic certification*, i.e., the mean p_{safe}, attained for a range of gradient-based scaling factors and networks. Each point is averaged over 50 images from the MNIST test set, same as (Wicker et al., 2020). In each case, we observe that the optimal scaling factor is larger than 0 – which represents no gradient-guided scaling and is marked by the red dashed line. The optimal gradient-based scaling factor varies between networks, as does its effect on the overall p_{safe}. Examining the results for individual images with a gradient-based scaling factor of 0 goes some way to explaining this behaviour.

For the 1x256 network, we find that 89.1% of the correctly classified images have p_{safe} values greater than 99.9% or less than 0.1%. On these networks, most of the images are either trivially verifiable or prohibitively challenging, with only 10.9% in between. In such instances, the use of gradient-based scaling factors has little effect, as p_{safe} is either very close to 1, or adversarial examples are found before p_{safe} can reach substantial values.

To further illustrate this point, we examined the CNN network, which was most sensitive to the gradient-

Dataset	Network	Accuracy (%)	Adversarial Accuracy (%)	Input Perturbation, ϵ	Probabilistic Certification (%)		
					Sampling (Wicker et al.)	Ours (PIE)	Ours (GIE)
MNIST	1x128	96	40		26.0	56.9	**58.4**
	1x256	94	42	0.025	22.0	50.0	**51.2**
	1x512	94	36		10.0	35.0	**37.3**
	2x256	92	16		72.0	74.0	**74.0**
	2x512	82	2	0.001	22.0	22.0	**26.1**
	2x1024	68	0		0.0	0.0	0.0
	2x50	96	70	0.01	58.6	67.3	**71.8**
	2x150	92	54	0.001	49.8	54.9	**56.4**
CIFAR10	CNN	72	2	0	22.0	27.1	**35.6**

Table 1: Probabilistic robustness results for the pure sampling method from (Wicker et al., 2020) and ours. Adversarial accuracies were obtained using an $\epsilon = 0.05$ bounded PGD attack for all networks. Probabilistic certification denotes the mean p_{safe} by each approach. Also, see the Appendix for a comparison of the probabilistic certification results obtained by each approach with the empirical probabilistic robustness values for each network.

based scaling factor in the same way. We found that only 68.6% of the correctly classified images lay outside the 0.1% - 99.9% boundary. To remove this dilution effect, we examined how gradient-based dynamic scaling impacts a single image, for which we use the same example given in Figure 2. Using no gradient-based dynamic scaling on this image, we achieve $p_{\text{safe}} = 36.5$ while using a gradient-based scaling factor of 0.2 we reach $p_{\text{safe}} = 97.8$, highlighting the importance of gradient-based dynamic scaling on certain inputs.

4.3 Ablation Study on the Number of Iterations

To better understand the inner workings of the iterative expansion approaches presented in this paper, we calculated the number of iterations it took Algorithm 1 to reach the maximum probabilistic certification presented in Table 1. The results are available in Figure 4. We observe that Algorithm 1 takes 8 iterations to start computing a meaningful lower bound and at most 13 iterations to reach maximum probabilistic certification for the MNIST networks. The minimum number of iterations to obtain a meaningful lower bound can be reduced, e.g., to a single iteration, by increasing the value of λ in line 8 of Algorithm 1; however, this is disadvantageous for two reasons: (1) one has to manually search for an optimal value for λ similarly to the pure sampling approach, (2) this can reduce the certified robustness obtained in later iterations because the expansion rate of PIE increases with λ.

CONCLUSIONS

As we discussed in the introduction, BNNs have been proposed as an inherently robust architecture. This is
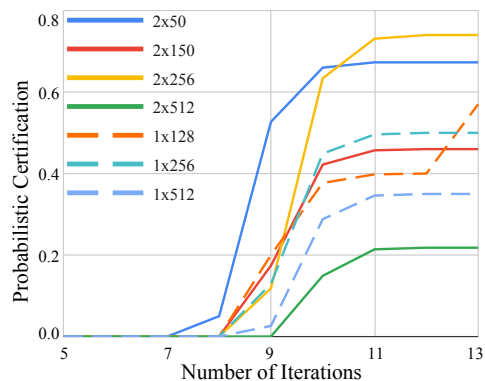

Figure 4: The number of expansion iterations before PIE reaches max probabilistic certifications in Table 1.

particularly important in safety-critical applications, where robust solutions are essential, and models need to be shown to be highly robust before deployment.

Given this, it is essential to ascertain the robustness of BNNs in production. Differently from plain neural networks, where robustness verification can be recast into a deterministic optimisation problem, establishing the robustness of BNNs is a probabilistic problem. Finding an exact solution to this problem is infeasible even for small models. As reported, proposals have been put forward to approximate the probabilistic safety of BNNs by sampling regions in the weight space; however, such approximations are loose and underestimate the actual robustness of the model, thereby providing relatively little information. This limits the possibility of using BNNs in tasks where robustness needs to be verified before deployment.

We have put forward an alternative approach to search the weight space, and therefore, approximate the lower

bounds of probabilistic safety in BNNs. The algorithms proposed provably produce tighter bounds than the SoA sampling-based approaches. This conclusion was validated in all experiments conducted. The approaches were evaluated on all the previously proposed benchmarks, as well as new benchmarks introduced in this paper, reporting a marked improvement on the bounds of all benchmarks. In some cases, we obtain bounds that are 50% higher than the previously known bound of just over 20%, opening the way to better evaluate the robustness of BNNs before deployment.

Acknowledgements

References

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938.

Adams, S., Patane, A., Lahijanian, M., and Laurenti, L. (2023). BNN-DP: robustness certification of bayesian neural networks via dynamic programming. In *International conference on machine learning, ICML*, pages 1613–1622. PMLR.

Anwar, S. M., Majid, M., Qayyum, A., Awais, M., Alnowami, M., and Khan, M. K. (2018). Medical image analysis using convolutional neural networks: a review. *Journal of medical systems*, 42(11):1–13.

Berrada, L., Dathathri, S., Dvijotham, K., Stanforth, R., Bunel, R., Uesato, J., Gowal, S., and Kumar, M. P. (2021). Make sure you're unsure: A framework for verifying probabilistic specifications. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 11136–11147.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *International conference on machine learning, ICML*, pages 1613–1622. PMLR.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars.

Bortolussi, L., Carbone, G., Laurenti, L., Patane, A., Sanguinetti, G., and Wicker, M. (2022). On the robustness of bayesian neural networks to adversarial attacks.

Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., and Misener, R. (2020). Efficient verification of relu-based neural networks via dependency analysis. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*.

Carbone, G., Wicker, M., Laurenti, L., Patané, A., Bortolussi, L., and Sanguinetti, G. (2020). Robustness of bayesian neural networks to gradient-based attacks. In *Advances in Neural Information Processing Systems, NeurIPS*.

Cardelli, L., Kwiatkowska, M., Laurenti, L., Paoletti, N., Patane, A., and Wicker, M. (2019a). Statistical guarantees for the robustness of bayesian neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*.

Cardelli, L., Kwiatkowska, M., Laurenti, L., and Patane, A. (2019b). Robustness guarantees for bayesian inference with gaussian processes. In *The Thirty-Third Conference on Artificial Intelligence, AAAI*.

Dalvi, N. N., Domingos, P. M., Mausam, Sanghai, S. K., and Verma, D. (2004). Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD*, pages 99–108. ACM.

Dreossi, T., Fremont, D. J., Ghosh, S., Kim, E., Ravanbakhsh, H., Vazquez-Chanlatte, M., and Seshia, S. A. (2019). VerifAI: A Toolkit for the Formal Design and Analysis of Artificial Intelligence-Based Systems. In *Computer Aided Verification CAV*, volume 11561 of *Lecture Notes in Computer Science*, pages 432–442. Springer.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. S. (2017). Domain-adversarial training of neural networks. In *Domain Adaptation in Computer Vision Applications*, Advances in Computer Vision and Pattern Recognition. JMLR. org.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations, ICLR*. OpenReview.net.

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

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109.

Henriksen, P. and Lomuscio, A. R. (2020). Efficient neural network verification via adaptive refinement and adversarial search. In *24th European Conference on Artificial Intelligence, ECAI*.

Hosseini, M. and Lomuscio, A. (2023). Bounded and unbounded verification of rnn-based agents in non-deterministic environments. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 2382–2384. IFAAMAS.

Ilyas, A., Santurkar, S., Tsipras, D., logan Engstrom, Tran, B., and Madry, A. (2019). Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems, NeurIPS*.

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., and Bennamoun, M. (2022). Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48.

Katz, G., Barrett, C. W., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification - 29th International Conference, CAV*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*.

Kouvaros, P. and Lomuscio, A. (2021). Towards scalable complete verification of relu neural networks via dependency-based branching. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Lechner, M., Žikelić, D. o. e., Chatterjee, K., and Henzinger, T. (2021). Infinite time horizon safety of bayesian neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 10171–10185. Curran Associates, Inc.

LeCun, Y., Cortes, C., and Burges, C. J. (2010). MNIST handwritten digit database.

Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., Kochenderfer, M. J., et al. (2021a). Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4):244–404.

Liu, Z., Cai, Y., Wang, H., Chen, L., Gao, H., Jia, Y., and Li, Y. (2021b). Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions. *IEEE Transactions on Intelligent Transportation Systems*.

Lomuscio, A. and Maganti, L. (2017). An approach to reachability analysis for feed-forward relu neural networks.

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

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations, ICLR*. OpenReview.net.

Pang, Y., Cheng, S., Hu, J., and Liu, Y. (2021). Evaluating the robustness of bayesian neural networks against different types of attacks.

Pulina, L. and Tacchella, A. (2010). An abstraction-refinement approach to verification of artificial neural networks. In *Computer Aided Verification, 22nd International Conference, CAV*, volume 6174 of *Lecture Notes in Computer Science*, pages 243–257. Springer.

Salimans, T., Kingma, D., and Welling, M. (2015). Markov chain monte carlo and variational inference: Bridging the gap. In *International conference on machine learning, ICML*, pages 1218–1226. PMLR.

Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J. P., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. (2019). Adversarial training for free! In *Advances in Neural Information Processing Systems, NeurIPS*.

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. (2016). Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. T. (2018). Fast and effective robustness certification. In *Advances in Neural Information Processing Systems, NeurIPS*.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. T. (2019). An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL):41:1–41:30.

Sinha, A., Namkoong, H., and Duchi, J. C. (2017). Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations, ICLR*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR*.

Tjeng, V., Xiao, K. Y., and Tedrake, R. (2019). Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations, ICLR*. OpenReview.net.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I. J., Boneh, D., and McDaniel, P. D. (2018). Ensemble adversarial training: Attacks and defenses. In *6th International Conference on Learning Representations, ICLR*. OpenReview.net.

Uchendu, A., Campoy, D., Menart, C., and Hildenbrandt, A. (2021). Robustness of bayesian neural networks to white-box adversarial attacks. In *Fourth IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE*.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. (2018). Formal security analysis of neural networks using symbolic intervals. In *27th USENIX Security Symposium, USENIX*.

Wicker, M., Laurenti, L., Patane, A., and Kwiatkowska, M. (2020). Probabilistic safety for bayesian neural networks. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI*.

Wicker, M., Patane, A., Laurenti, L., and Kwiatkowska, M. (2023). Adversarial robustness certification for bayesian neural networks.

Wijk, H., Wang, B., and Kwiatkowska, M. (2022). Robustness guarantees for credal bayesian networks via constraint relaxation over probabilistic circuits. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 4885–4892.

Zhang, Y., Wei, Z., Zhang, X., and Sun, M. (2023). Using Z3 for formal modeling and verification of FNN global robustness. In *The 35th International Conference on Software Engineering and Knowledge Engineering, SEKE*, pages 110–113. KSI Research Inc.

Checklist

1. For all models and algorithms presented, check if you include:

   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]

   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]

   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:

   (a) Statements of the full set of assumptions of all theoretical results. [Yes]

   (b) Complete proofs of all theoretical results. [Yes]

   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:

   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]

   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]

   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]

   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [Yes]

   (b) The license information of the assets, if applicable. [Not Applicable]

   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]

   (d) Information about consent from data providers/curators. [Not Applicable]

   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A   Limitations

Methods for verifying the probabilistic robustness of BNNs are extremely sensitive to parameter count - this remains true for both algorithms here proposed. Therefore scaling to larger networks is a challenge due to two primary effects: Firstly, as network size increases so does the parameter count and respective dimensionality of the probability space. The higher this dimensionality, the more important sampling *large* orthotopes becomes, and the less effective sampling *multiple* orthotopes becomes. This reliance on being able to verify single large orthotopes makes these algorithms vulnerable to loose verification techniques. Secondly, as network size increases it is well known that the tightness of bound propagation techniques suffers, thus impacting the ability of our algorithms to verify any orthotopes. Both of these limitations can be alleviated by using tighter verification procedures in place of bound propagation.

B   Correction to (Wicker et al., 2020) implementation

We remarked in Section 4 that while generating the experimental results we identified an error in the implementation in (Wicker et al., 2020). In the following, we present more details of the issue identified. The results presented in the paper refer to the corrected version of their implementation.

Given a series of s potentially-overlapping orthotopes, $\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_s$, we can calculate the probability captured in the series by computing the cumulative probability for each orthotope $\hat{\mathcal{C}}_i$ and summing over the series, while taking into account that we need to subtract the cumulative probabilities of the intersections and so on according to the inclusion-exclusion principle. Computing the cumulative probability of a single orthotope is trivial as the distributions in each dimension are independent; accordingly, we have

$$P(\hat{\mathcal{C}}_i) = \prod_{d=1}^{n_{\boldsymbol{w}}} \frac{1}{2} \left(\mathrm{erf}(\frac{\boldsymbol{\mu}_d - \boldsymbol{\ell}_d^i}{\boldsymbol{\sigma}_d \sqrt{2}}) - \mathrm{erf}(\frac{\boldsymbol{\mu}_d - \boldsymbol{u}_d^i}{\boldsymbol{\sigma}_d \sqrt{2}}) \right), \tag{3}$$

where $\boldsymbol{\mu}_d$ and $\boldsymbol{\sigma}_d$ are the mean and standard deviation of the d-th parameter and $\boldsymbol{\ell}_d^i$ and $\boldsymbol{u}_d^i$ are the lower and upper bounds in the d-th dimension for the i-th orthotope. We can then obtain the probability of the union of all s orthotopes according to the inclusion-exclusion principle. For example, if there are no overlaps, we can simply sum $P(\hat{\mathcal{C}}_i)$'s for all $\hat{\mathcal{C}}_i$ and obtain

$$\hat{p}_{\mathrm{safe}} = \sum_{i=1}^{s} P(\hat{\mathcal{C}}_i). \tag{4}$$

This is correctly presented in Algorithm 1 from (Wicker et al., 2020).

However, the code[1] accompanying the paper implements this differently leading to large overestimates (i.e., incorrect values) of the p_{safe} values as presented in (Wicker et al., 2020).

The implementation first computes the dimension-wise probability,

$$P(\boldsymbol{w}_d) = \sum_{i=1}^{s} \frac{1}{2} \left(\mathrm{erf}(\frac{\boldsymbol{\mu}_d - \boldsymbol{\ell}_d^i}{\boldsymbol{\sigma}_d \sqrt{2}}) - \mathrm{erf}(\frac{\boldsymbol{\mu}_d - \boldsymbol{u}_d^i}{\boldsymbol{\sigma}_d \sqrt{2}}) \right), \tag{5}$$

where $\boldsymbol{w}_d$ is the d-th Bayesian parameter. The implementation then computes the product of $P(\boldsymbol{w}_d)$ across the $n_{\boldsymbol{w}}$ dimensions,

$$\hat{p}_{\mathrm{safe}} = \prod_{d=1}^{n_{\boldsymbol{w}}} P(\boldsymbol{w}_d), \tag{6}$$

where $\hat{p}_{\mathrm{safe}}$ is marked with a hat to represent an invalid p_{safe} value. The error comes from the fact that the operations in Equations (3) and (4) are noncommutative.

Below we show two code snippets taken from the (Wicker et al., 2020) implementation. The first shows the orthotopes being passed to the '`compute_interval_prob_weight`' function (lines 1-4). Note that the parameters are separated into weights and biases and by layer at this point. The second snippet shows the definition of the '`compute_interval_prob_weight`' function, where the probability is calculated for each dimension over all

---

[1]At the time of writing, this is available at `https://github.com/matthewwicker/ProbabilisticSafetyforBNNs`.

```
1   pW_0 = compute_interval_probs_weight(np.asarray(vW_0), marg=w_margin, mean=mW_0, std=dW_0)
2   pb_0 = compute_interval_probs_bias(np.asarray(vb_0), marg=w_margin, mean=mb_0, std=db_0)
3   pW_1 = compute_interval_probs_weight(np.asarray(vW_1), marg=w_margin, mean=mW_1, std=dW_1)
4   pb_1 = compute_interval_probs_bias(np.asarray(vb_1), marg=w_margin, mean=mb_1, std=db_1)
5
6   # Now that we have all of the probabilities we just need to multiply them out to get
7   # the final lower bound on the probability of the condition holding.
8   # Work with these probabilities in log space
9   p = 0.0
10  for i in pW_0.flatten():
11      p+=math.log(i)
12  for i in pb_0.flatten():
13      p+=math.log(i)
14  for i in pW_1.flatten():
15      p+=math.log(i)
16  for i in pb_1.flatten():
17      p+=math.log(i)
18  #print math.exp(p)
19  return math.exp(p)
```

```
1   def compute_interval_probs_weight(vector_intervals, marg, mean, std):
2       means = mean; stds = std
3       prob_vec = np.zeros(vector_intervals[0].shape)
4       for i in range(len(vector_intervals[0])):
5           for j in range(len(vector_intervals[0][0])):
6               intervals = []
7               for num_found in range(len(vector_intervals)):
8                   interval = [vector_intervals[num_found][i][j]-(stds[i][j]*marg),
                          vector_intervals[num_found][i][j]+(stds[i][j]*marg)]
9                   intervals.append(interval)
10              p = compute_erf_prob(merge_intervals(intervals), means[i][j], stds[i][j])
11              prob_vec[i][j] = p
12      return np.asarray(prob_vec)
```

orthotopes, as in Equation (5). The returned parameter-wise probabilities are then combined by multiplication as in Equation (6) producing $\hat{p}_{\text{safe}}$, aliased 'p' in the code (lines 9-19).

We are confident this is an error in the implementation. The method itself is correct, but the results produced by the implementation are incorrect (overestimates). To validate this is indeed an error in the code, we asked two expert ML authors to check this portion of the code by making no suggestion it may have an error and both independently identified the same error we report above.

C Evaluation of timing for PIE algorithm

In Table 2 we examine the impact on runtime of generating the gradients required for the PIE algorithm. We observe that in each case there is an increase in runtime, peaking at 3.3% for the 1x128 network. The relative cost decreases as network size increases as a result of the gradient computation scaling better than the LBP computation with network size.

D Training Details

In addition to evaluating our approaches on the SoA benchmarks, we evaluated our approach on two fully connected BNNs and a convolutional BNN (only the fully connected layers are Bayesian). Similarly to the SoA, we used normal distributions for the prior (with mean 0 and standard deviation 1) and the posterior of the Bayesian layers. Using a wider standard deviation than the SoA, as expected, has resulted in networks that are more adversarially robust (cf. Table 1). We trained the networks using variation inference (Blundell et al.,

Network	Relative Runtime ($\pm\%$)
1x128	+3.3
1x256	+2.5
1x512	+1.4
2x256	+0.8
2x512	+0.6
2x1024	+0.2
2x50	+2.3
2x150	+0.4
CNN	+0.5

Table 2: Impact of computing gradients for PIE algorithm on runtime. The data are presented as a percentage increase or decrease in runtime as compared to the (Wicker et al., 2020) approach on the same networks.

2015) using a learning rate of 0.001 with Adam optimiser (Kingma and Ba, 2015). All the hidden layers in the networks use ReLU activations.

The fully connected networks are trained on the MNIST dataset and each have 2 hidden layers with 50 and 150 nodes, respectively. The convolutional network is trained on the CIFAR10 dataset and has 5 convolutional layers with filter size 3 and stride 2, followed by 3 fully connected Bayesian hidden layers of sizes x and y.

E   Comparison against Empirical Probabilistic Robustness

In Figure 5, we compare the pure sampling approach of (Wicker et al., 2020), as well as the PIE and GIE approaches, presented in this paper, against the empirical values we obtained for the probabilistic robustness of the networks in Table 1. We calculated the values for the empirical probabilistic robustness by sampling from the BNN posterior and using IBP to check whether classified correctly. Each value for the empirical accuracy shows the percentage of samples that resulted in correct classification.

We observe that PIE and GIE always perform favourably compared to pure sampling, and in some instances, such as for 1-layer networks, PIE and GIE significantly outperform pure sampling. Nevertheless, for most networks, there is still a gap between the certified robustness obtained by each of the approaches and the empirical probabilistic robustness.
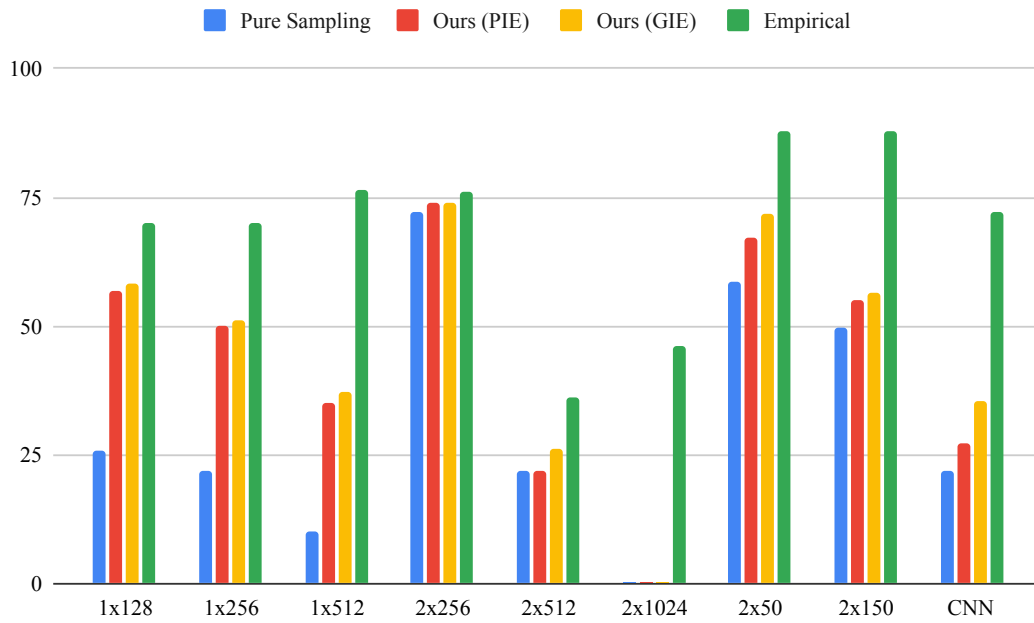
Figure 5: Comparison of the pure sampling, PIE, and GIE approaches against the empirical probabilistic robustness, obtained by sampling parameters from the BNN posterior and using IBP to verify whether they result in correct classification. 50 samples were used for calculating each empirical accuracy.