BNN-DP: Robustness Certification of Bayesian Neural Networks via Dynamic Programming

Steven Adams [1] **Andrea Patanè** [2] **Morteza Lahijanian** [3] **Luca Laurenti** [1]

Abstract

In this paper, we introduce BNN-DP, an efficient algorithmic framework for analysis of adversarial robustness of Bayesian Neural Networks (BNNs). Given a compact set of input points $T \subset \mathbb{R}^n$, BNN-DP computes lower and upper bounds on the BNN's predictions for all the points in T. The framework is based on an interpretation of BNNs as stochastic dynamical systems, which enables the use of Dynamic Programming (DP) algorithms to bound the prediction range along the layers of the network. Specifically, the method uses bound propagation techniques and convex relaxations to derive a backward recursion procedure to over-approximate the prediction range of the BNN with piecewise affine functions. The algorithm is general and can handle both regression and classification tasks. On a set of experiments on various regression and classification tasks and BNN architectures, we show that BNN-DP outperforms state-of-the-art methods by up to four orders of magnitude in both tightness of the bounds and computational efficiency.

1. Introduction

Adversarial attacks (small and often imperceptible perturbations to input points that can trigger incorrect decisions) have raised serious concerns about the robustness of models learned from data (Biggio & Roli, 2018; Goodfellow et al., 2015). Bayesian Neural Networks (BNNs), i.e., neural networks with distributions placed over their parameters, have been proposed as a potentially more robust machine learning paradigm compared to their deterministic counterpart (Carbone et al., 2020; McAllister et al., 2017). While retaining the advantages intrinsic to deep learning (e.g., representation learning), BNNs enable principled evaluation of model uncertainty, which can be used for flagging out-of-distribution samples and robust decision making (Kahn et al., 2017). However, existing methods that formally (i.e., with certified bounds) evaluate the robustness of BNNs (Berrada et al., 2021; Wicker et al., 2020; Lechner et al., 2021) are limited to posterior distributions with bounded support, thus not supporting the majority of the algorithms commonly employed to train BNNs (Blundell et al., 2015; Zhang et al., 2018a; Osawa et al., 2019) and lack scalability to BNNs with non-negligible posterior variance estimates.

In this paper, we present BNN-DP, a novel algorithmic framework that quantifies the adversarial robustness of BNNs with formal (strong) guarantees. BNN-DP is scalable and supports posterior distributions of unbounded support, as commonly used in BNNs, e.g., Gaussian distributions. We consider both regression and classification settings. For a compact set of input points $T \subset \mathbb{R}^{n_0}$, we study the robustness of the BNN's decision, i.e., argmax of the expectation of the softmax in case of classification and expectation of the output of the BNN for regression, for all the points in T. As exact computation of these quantities is infeasible (Berrada et al., 2021), we focus on computing piecewise affine (PWA) upper and lower bounds. Inspired by Marchi et al. (2021), we take a unique view of BNNs as stochastic dynamical systems that evolve over the layers of the neural network and show that the computation of the BNN robustness can be formulated as the solution of a Dynamic Program (DP). This allows us to break the computation of adversarial robustness into a set of simpler sub-problems (one for each layer of the BNN). Critically, while each of these problems is still possibly non-convex, we show that accurate and efficient PWA relaxations can be derived for each by relying on tools from Gaussian processes and convex optimizations.

We validate our framework on several regression and classification tasks, including the Kin8nm, MNIST, Fashion MNIST, and CIFAR-10 datasets, and a range of BNN architectures. For all tasks, the results show that our method outperforms state-of-the-art competitive approaches in both

[1]Delft Center for Systems and Control, Technical University of Delft, Delft, 2628 CD, The Netherlands [2]School of Computer Science and Statistics, Trinity College Dublin, Dublin 2, Ireland [3]Departement of Aerospace Engineering Sciences and Computer Science, University of Colorado Boulder, Boulder, CO 80303, USA. Correspondence to: Steven Adams <s.j.l.adams@tudelft.nl>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

precision and computational time. For instance, on the Fashion MNIST dataset, our approach achieves an average 93% improvement in certified lower bound compared to Berrada et al. (2021), while being around 3 orders of magnitude faster. In summary, this paper makes the following main contributions:

- we introduce a framework based on stochastic dynamic programming and convex relaxation for the analysis of adversarial robustness of BNNs,
- we implement an efficient algorithmic procedure of our framework for BNNs trained with Gaussian variational inference (VI) in both regression and classification settings,[1] and
- we benchmark the robustness of a variety of BNN models on five datasets, empirically demonstrating how our method outperforms state-of-the art approaches by orders of magnitude in both tightness and efficiency.

Related Works Many algorithms have been developed for certification of deterministic (i.e., non Bayesian) neural networks (NNs) (Katz et al., 2017; Weng et al., 2018; Wong & Kolter, 2018; Bunel et al., 2020). However, these methods cannot be employed to BNNs because they all assume the weights of the network have a fixed value, whereas in the Bayesian setting they are distributed according to the BNN posterior. Methods for certification of BNNs have recently presented in (Wicker et al., 2020; Berrada et al., 2021; Lechner et al., 2021). Wicker et al. (2020) consider a different notion of robustness than the one in this paper, not directly related to adversarial attacks on the BNN decision. Furthermore, that work considers a partitioning procedure in weight space that makes it applicable only to small networks and/or with small variance. The method proposed in (Berrada et al., 2021) is based on dual optimization. Hence, it is restricted to distributions with bounded support and needs to solve non-convex problems at large computational costs for classification tasks. Separately, Lechner et al. (2021) aims to build an intrinsic safe BNN by truncating the posterior in the weight space. Cardelli et al. (2019a); Wicker et al. (2021) introduced statistical approaches to quantify the robustness of a BNN, which however, does not return formal guarantees, which are necessary in safety-critical settings. Empirical methods that use the uncertainty of BNNs to flag adversarial examples are introduced in (Rawat et al., 2017; Smith & Gal, 2018). These, however, consider only point-wise uncertainty estimates, specific to a particular test point and do not account for worst-case adversarial perturbations.

Various recent works have proposed formal methods to compute adversarial robustness for Gaussian Processes (GPs) (Cardelli et al., 2019b; Smith & Gal, 2018; Patanè et al., 2022; Smith et al., 2022). In BNNs, however, due to the non-linearity of activation functions, the distribution over the space of functions induced by a BNN is generally non-Gaussian, even if a Gaussian distribution in weight space is assumed. Hence, the techniques that are developed for GPs cannot be directly applied to BNNs.

[1] Our code is available at https://github.com/sjladams/BNN_DP.

2. Robust Certification of BNNs Problem

2.1. Bayesian Neural Networks (BNNs)

For an input vector $x \in \mathbb{R}^{n_0}$, we consider fully connected neural networks $f^w : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{K+1}}$ of the following form for $k = 0, \dots, K$:[2]

$$\begin{aligned} z_0 &= x, & \zeta_{k+1} &= W_k(z_k^T, 1)^T, \\ z_k &= \phi_k(\zeta_k), & f^w(x) &= \zeta_{K+1}, \end{aligned} \tag{1}$$

where K is the number of hidden layers, n_k is the number of neurons of layer k, $\phi_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ is a vector of continuous activation functions (one for each neuron) in layer k, and $W_k \in \mathbb{R}^{n_k \times n_{k+1}}$ is the matrix of weights and biases that correspond to the kth layer of the network. We denote the vector of parameters by $w = (W_0^T, \dots, W_K^T)^T$ and the mapping from ζ_{k_1} to ζ_{k_2} by $f^w_{k_1:k_2} : \mathbb{R}^{n_{k_1}} \to \mathbb{R}^{n_{k_2}}$ for $k_1, k_2 \in \{0, ..., K\}$. ζ_{K+1} is the final output of the network (or the logit in the case of classification problems).

In the Bayesian setting, one starts by assuming a prior distribution $p(w)$ over the parameters w and a likelihood function $p(y|x, w)$. We adopt bold notation to denote random variables and write $f^{\boldsymbol{w}}$ to denote a BNN defined according to Eqns. (1). The likelihood is generally assumed to be Gaussian in case of regression and categorical for classification, where the probability for each class is given as the softmax of the neural network final logits (MacKay, 1992). Then, given a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N_\mathcal{D}}$, learning amounts to computing the posterior distribution $p(w|\mathcal{D})$ via the Bayes rule (MacKay, 1992). The posterior predictive distribution over an input x^* is finally obtained by marginalising the posterior over the likelihood, i.e., $p(y^*|x^*, \mathcal{D}) = \int p(y^*|x^*, w)p(w|\mathcal{D})dw$. The final output (decision) of the BNN, $\hat{y}(x^*)$, is then computed by minimising a loss function $\mathcal{L}$ averaged over the predictive distribution, i.e.,

$$\hat{y}(x^*) = \arg\min_y \int \mathcal{L}(y, y^*)p(y^*|x^*, \mathcal{D})dy^*.$$

[2] Note that the formulation of neural networks considered in Eqn (1) also includes convolutional neural networks (CNNs). In fact, the convolutional operation can be interpreted as a linear transformation into a larger space; see, e.g., Chapter 3.4.1 in (Gal & Ghahramani, 2016). This allows us to represent convolutional layers equivalently as fully connected layers, and do verification for CNNs as we show in Section 7.

In this paper, we focus on both regression and classification problems. In regression, an l_2 loss is generally used, which leads to an optimal decision $\hat{y}(x^*)$ given by the mean of the predictive posterior distribution (Neal, 2012), i.e., $\hat{y}(x^*) = \mathbb{E}_{\mathbf{y}\sim p(y|x^*,\mathcal{D})}[\mathbf{y}]$.[3] For classification, ℓ_{0-1} loss is typically employed, which results in

$$\hat{y}(x^*) = \underset{i\in\{1,\dots,n_{K+1}\}}{\arg\max} \; \mathbb{E}_{\boldsymbol{w}\sim p(w|\mathcal{D})}\left[\text{softmax}^{(i)}(f^{\boldsymbol{w}}(x^*))\right],$$

where $\text{softmax}^{(i)}$ is the ith component of the n_{K+1}-dimensional softmax function.[4] Unfortunately, because of the non-linearity introduced by the neural network architecture, the computation of the posterior distribution and consequently of $\hat{y}(x^*)$ cannot be done analytically. Therefore, approximate inference methods are required. In what follows, we focus on mean-field Gaussian Variational Inference (VI) approximations (Blundell et al., 2015). Specifically, we fit an approximating multivariate Gaussian $q(w) = \mathcal{N}(w \mid \mu_w; \Sigma_w) \approx p(w \mid \mathcal{D})$ with mean μ_w and block diagonal covariance matrix Σ_w such that for $k \in \{0,\dots,K\}$ and $i \in \{1,\dots,n_k\}$, the approximating distribution of the parameters corresponding to the ith node of the kth layer is

$$q(W_k^{(i,:)}) = \mathcal{N}\left(W_k^{(i,:)} \mid \mu_{w,k,i};\, \Sigma_{w,k,i}\right) \tag{2}$$

with mean $\mu_{w,k,i}$ and covariance matrix $\Sigma_{w,k,i}$.[5]

Remark 1. *While our primary focus is on VI, the techniques presented in this paper can be applied to other approximate inference methods, such as HMC (Neal, 2012) and Dropout (Gal & Ghahramani, 2016). In these cases, the prediction of the BNN is obtained by averaging over a finite ensemble of NNs. For this setting, the dynamic programming problem in Theorem 1 reduces to computing piecewise linear relaxations for each layer of a weighted sum, i.e., an average, of deterministic neural networks, and propagating the resulting relaxations backward.*

2.2. Problem Statement

Given a BNN $f^{\boldsymbol{w}}$ trained on a dataset $\mathcal{D}$, as common in the literature (Madry et al., 2017), for a generic test point x^*, we represent the possible adversarial perturbations by defining a compact neighbourhood T around x^* and measure the changes in the BNN output caused by limiting the perturbations to lie within T.

[3] In the remainder, we may omit the probability measure of an expectation or probability when it is clear from the context.

[4] Analogous formulas can be obtained for the weighted classification loss by factoring in misclassification weights in the argmax.

[5] BNN-DP can be extended to Gaussian approximation distributions with inter-node or inter-layer correlations. In that case, to solve the backward iteration scheme of Theorem 1, the value functions need to be marginalized over partitions in weight space.

Definition 1 (Adversarial Robustness)**.** *Consider a BNN $f^{\boldsymbol{w}}$, a compact set $T \subset \mathbb{R}^{n_0}$, and input point $x^* \in T$. For a given threshold $\gamma > 0$, $f^{\boldsymbol{w}}$ is adversarially robust in x^* iff*

$$\forall x \in T, \quad \|\hat{y}(x) - \hat{y}(x^*)\|_p \le \gamma,$$

where $\|\cdot\|_p$ is an ℓ_p norm.

Definition 1 is analogous to the standard notion of adversarial robustness employed for deterministic neural networks (Katz et al., 2017) and Bayesian models (Patanè et al., 2022). As discussed in Section 2.1, the particular form of a BNN's output depends on the specific application considered. Below, we focus on regression and classification problems.

Problem 1. *Let $T \subset \mathbb{R}^{n_0}$ be a compact subset. Define functions $I(y) = y$ and $softmax(y) = [softmax^{(1)}(y), ..., softmax^{(n_{K+1})}(y)]$. Then, for a BNN $f^{\boldsymbol{w}}$, $h \in \{I, softmax\}$, and $i \in \{1, ..., n_{K+1}\}$, compute:*

$$\begin{aligned} \pi^{(i)}_{\min}(T) &= \min_{x\in T} \mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h^{(i)}(f^{\boldsymbol{w}}(x))\right], \\ \pi^{(i)}_{\max}(T) &= \max_{x\in T} \mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h^{(i)}(f^{\boldsymbol{w}}(x))\right]. \end{aligned} \tag{3}$$

In the regression case ($h = I$), Problem 1 seeks to compute the ranges of the expectation of the BNN for all $x \in T$. Similarly, in the classification case ($h = \text{softmax}$), Eqns. (3) define the ranges of the expectation of the softmax of each class for $x \in T$. It is straightforward to see that these quantities are sufficient to check whether $f^{\boldsymbol{w}}$ is adversarially robust for $x \in T$; that is, if $\sup_{x\in T} ||\hat{y}(x) - \hat{y}(x^*)||_p \le \gamma$.

Remark 2. *Our method can be extended to other losses, i.e., other forms of h in Eqns. (3), as long as affine relaxations of h can be computed.*

Approach Outline Due to the non-convex nature of $f^{\boldsymbol{w}}$ and possibly h, the computation of $\mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}[h(f^{\boldsymbol{w}}(x))]$ is analytically infeasible. To solve this problem, in Section 4, we view BNNs as stochastic dynamical systems evolving over the layers of the neural network. Through this, we show that adversarial robustness can be characterized as the solution of a dynamic programming (DP) problem. This allows us to break its computation into K simpler optimization problems, one for each layer. Each problem essentially queries a back-propagation of the uncertainty of the BNN through h and from one layer of the neural network to the next. Due to the non-convex nature of the layers of the BNN, these problems still cannot be solved exactly. We overcome this problem by using convex relaxations. Specifically, in Section 5, we show that efficient PWA relaxations can be obtained by recursively bounding the DP problem. In Section 6, we combine the theoretical results into a general algorithm called BNN-DP that solves Problem 1 efficiently.

3. Preliminaries on Relaxations of Functions

To propagate the uncertainty of the BNN from one layer to the other, we rely on upper and lower approximations of the corresponding Neural Network (NN), also known as *relaxations*. For vectors $\check{x}, \hat{x} \in \mathbb{R}^n$, we denote by $[\check{x}, \hat{x}]$ the n-dimensional hyper-rectangle defined by $\check{x}$ and $\hat{x}$, i.e., $[\check{x}, \hat{x}] = [\check{x}^{(1)}, \hat{x}^{(1)}] \times [\check{x}^{(2)}, \hat{x}^{(2)}] \times \ldots \times [\check{x}^{(n)}, \hat{x}^{(n)}]$. We consider two types of relaxations, interval and affine.

Definition 2 (Interval Relaxation)**.** *An interval relaxation of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ over a set $T \subseteq \mathbb{R}^n$ are two vectors $\check{b}, \hat{b} \in \mathbb{R}^m$ such that $f(x) \in [\check{b}, \hat{b}]$ for all $x \in T$.*

Definition 3 (Affine Relaxation)**.** *An affine relaxation of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ over a set $T \subseteq \mathbb{R}^n$ are two affine functions $\check{A}x + \check{b}$ and $\hat{A}x + \hat{b}$ with $\check{A}, \hat{A} \in \mathbb{R}^{m \times n}$ and $\check{b}, \hat{b} \in \mathbb{R}^m$ such that $f(x) \in [\check{A}x + \check{b}, \hat{A}x + \hat{b}]$ for all $x \in T$.*

Interval and symbolic arithmetic can be used to propagate relaxations through the layers of a NN. Let, $[\alpha]_+ := \max\{\alpha, 0\}$ and $[\alpha]_- := \min\{\alpha, 0\}$ represent the saturation operators on α. For a vector or matrix, $[\cdot]_+$ and $[\cdot]_-$ represent element-wise max and min, respectively. We adopt the notation of Liu et al. (2021) and write interval arithmetic w.r.t. a linear mapping M compactly as $\otimes$ where $M \otimes [\check{b}, \hat{b}] := \left[[M]_+\check{b} + [M]_-\hat{b},\ [M]_+\hat{b} + [M]_-\check{b}\right]$, and use the similar notation for symbolic arithmetic.

4. BNN Certification via Dynamic Program

As observed in Marchi et al. (2021), NNs and consequently BNNs can be viewed as dynamical systems evolving over the layers of the network. In particular, for $k \in \{0, ..., K\}$, Eqn. (1) can be rewritten as:

$$z_{k+1} = \phi_{k+1}(\boldsymbol{W}_k(z_k^T, 1)^T) \tag{4}$$

with initial condition $z_0 = x$. Since, in a BNN, weights and biases are random variables sampled from the approximate posterior $q(\cdot)$, Eqn. (4) describes a non-linear stochastic process evolving over the layers of the NN. This observation leads to the following theorem, which shows that $\mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h(f^{\boldsymbol{w}}(x))\right]$ can be characterized as the solution to a backward recursion DP problem.

Theorem 1. *Let $f^{\boldsymbol{w}}(x)$ be a fully connected BNN with K hidden layers and $h : \mathbb{R}^{n_{K+1}} \to \mathbb{R}^l$ be an integrable function. For $k = 0, ..., K$, define functions $V_k : \mathbb{R}^{n_k} \to \mathbb{R}^l$ backwards-recursively as:*

$$V_K(z) = \mathbb{E}_{\boldsymbol{W}_K\sim q(\cdot)}\left[h(\boldsymbol{W}_K(z^T, 1)^T)\right], \tag{5a}$$

$$V_{k-1}(z) = \mathbb{E}_{\boldsymbol{W}_{k-1}\sim q(\cdot)}\left[V_k(\phi_k(\boldsymbol{W}_{k-1}(z^T, 1)^T))\right]. \tag{5b}$$

Then, it holds that $\mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h(f^{\boldsymbol{w}}(x))\right] = V_0(x)$.

The proof of Theorem 1 is reported in Appendix A.1 and obtained by induction over the layers of the NN by relying on the law of total expectation and independence of the parameters distribution at different layers.[6] Figure 1 illustrates the backward-iteration scheme of Theorem 1 for a two-hidden-layer BNN. Starting from the last layer, value functions V_k are constructed according to Eqns. (5a) and (5b) describing how the output of layer k is transformed in the previous layers. Theorem 1 is a central piece of our framework as it allows one to break the computation of $\mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h(f^{\boldsymbol{w}}(x))\right]$ into $K+1$ (simpler) sub-problems, one for each layer of the BNN. In fact, note that V_k is a deterministic function. Hence, all the uncertainty in V_{k-1} depends only on the weights of layer $k-1$. This is a key point that we use to derive efficient methods to solve Problem 1. Nevertheless, we stress that since $V_k(z)$ is obtained by propagating z over $K-k$ layers of the BNN, this is still generally a non-convex function, whose exact optimisation is infeasible in practice. Consequently, we employ the following corollary, which guarantees that, to solve Problem 1, it suffices to recursively bound V_k following Eqns. (5a) and (5b).

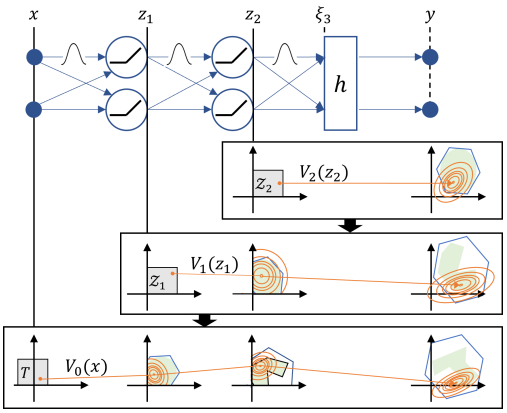


Figure 1. Illustration of the DP algorithm in Theorem 1 for a BNN with two hidden layers. Value functions V_k are mappings from the latent and input spaces of the BNN to the mean of the output distribution. For each mapping, the distribution and mapping of a single point is displayed in orange. Starting from the last hidden layer, we recursively compute PWA approximations of the mappings. The true mean of the BNN for all $z_2 \in \mathcal{Z}_2$ is in the green oval, which we over-approximate by the blue hexagon.

Corollary 1.1. *For $k \in \{1, \ldots, K\}$, let functions $\check{V}_k, \hat{V}_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_l}$ be relaxations of $V_k(z_k)$, i.e, $\forall z_k \in \mathbb{R}^{n_k}, \check{V}_k(z_k) \leq V_k(z_k) \leq \hat{V}_k(z_k)$. Then*

$$\mathbb{E}_{\boldsymbol{W}_{k-1}\sim q(\cdot)}\left[\check{V}_k(\phi_k(\boldsymbol{W}_{k-1}(z^T, 1)^T))\right] \leq V_{k-1}(z_k) \leq \mathbb{E}_{\boldsymbol{W}_{k-1}\sim q(\cdot)}\left[\hat{V}_k(\phi_k(\boldsymbol{W}_{k-1}(z^T, 1)^T))\right].$$

[6] While the vast majority of VI algorithms make this assumption, Theorem 1 can be generalized to the case where there is inter-layer correlation by marginalizing Eqn. (5a) and (5b) over partitions in correlated weight-spaces.

Further, for $i \in \{1, \dots, l\}$, it holds that $\pi^{(i)}_{\min}(T) \geq \min_{x \in T} \check{V}^{(i)}_0(x)$ and $\pi^{(i)}_{\max}(T) \leq \max_{x \in T} \hat{V}^{(i)}_0(x)$.

Corollary 1.1 allows us to recursively find relaxations of V_k via Theorem 1. In what follows, we focus on finding PWA relaxations $\check{V}_k$ and $\hat{V}_k$. To achieve that, there are two basic steps: (i) initialization of $\check{V}_k, \hat{V}_k$ via Eqn. (5a), and (ii) backward propagation of $\check{V}_k, \hat{V}_k$ through a hidden layer of the BNN via Eqn. (5b). In Section 5, we first show an efficient method to perform step (ii) and then focus on (i).

5. PWA Relaxation for Dynamic Programming

Our goal in this section is to find PWA relaxations of V_k. To do that, we first show how to propagate affine relaxations of the value function backwards through a single hidden layer of the BNN via Eqn. (5b) and then generalize this result to PWA relaxations. Note that, because the support of a BNN is generally unbounded, affine relaxations of Eqn. (5b) and (5a) lead to overly conservative results (an affine function should over-approximate a non-linear function over an unbounded set). Thus, PWA relaxations are necessary to obtain tight approximations. Finally, in Subsection 5.3 we show how to compute relaxations for Eqn. (5a).

5.1. Affine Value Functions

For the sake of presentation, we focus on upper bound $\hat{V}_{k-1}$; the lower bound case follows similarly. Let $\hat{V}_k : \mathbb{R}^{n_k} \to \mathbb{R}^l$ be an affine upper bound on V_k. Then, by Corollary 1.1 and the linearity of expectation, it holds that

$$\hat{V}_{k-1}(z) = \hat{V}_k(\mathbb{E}_{\boldsymbol{W}_{k-1} \sim q(\cdot)} \left[\phi_k(\boldsymbol{W}_{k-1}(z^T, 1)^T)\right]). \quad (6)$$

Recall that here q is a Gaussian distribution (see Section 2.1). Hence, due to the closure of Gaussian random variables w.r.t. linear transformations, we can rewrite Eqn. (6) as:

$$\hat{V}_{k-1}(z) = \hat{V}_k(\mathbb{E}_{\boldsymbol{\zeta} \sim \mathcal{N}(m_k(z);\, \mathrm{diag}(s_k(z)))} \left[\phi_k(\boldsymbol{\zeta})\right]), \quad (7)$$

where $m_k : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}$ and $s_k : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}_{\geq 0}$ are defined component-wise as

$$\begin{aligned} m^{(i)}_k(z) &= \mu_{w,k-1,i}(z^T, 1)^T, \\ s^{(i)}_k(z) &= (z^T, 1)\Sigma_{w,k-1,i}(z^T, 1)^T, \end{aligned} \quad (8)$$

for all $i \in \{1, \dots, n_k\}$ with $\mu_{w,k-1,i}, \Sigma_{w,k-1,i}$ being mean and covariance of the ith node of the kth layer. $\mathrm{diag}(s)$ is a diagonal matrix with the elements of s on the main diagonal. Note that Eqn (7) reduces the propagation of the value function to the propagation of a Gaussian random variable ($\boldsymbol{\zeta}$) through an activation function (ϕ_k). In Proposition 2, we show how this propogation can be achieved analytically for ReLU activiation functions. Generalization to other activation functions is discussed in Remark 3.

Proposition 2. *For $k \in \{1, \dots, K\}$, let $\hat{V}_k$ be an affine function and $Z \subset \mathbb{R}^{n_{k-1}}$ be a compact set. Define function $r_k : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}_{\geq 0}$ as $r_k(z) = \sqrt{s_k(z)}$, and let $\check{r}_k, \hat{r}_k : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}_{\geq 0}$ be an affine-relaxation of r_k w.r.t. Z. Further, define $g : \mathbb{R}^2 \to \mathbb{R}$ as*

$$g(\mu, \sigma) = \frac{\mu}{2}\left[1 - erf\left(\frac{-\mu}{\sigma\sqrt{2}}\right)\right] + \frac{\sigma}{\sqrt{2\pi}} e^{-(\mu/\sigma\sqrt{2})^2},$$

and, let $\check{g}_i, \hat{g}_i : \mathbb{R}^2 \to \mathbb{R}$ be an affine-relaxation of g w.r.t. $\{(m^{(i)}_k(z), r^{(i)}_k(z)) \mid \forall z \in Z\}$, Then, for $\check{A}, \hat{A} \in \mathbb{R}^{n_{k-1} \times n_k}$ and $\check{b}, \hat{b} \in \mathbb{R}^{n_k}$ defined as, $\forall i \in \{1, \dots, n_k\}$,

$$\begin{aligned} \check{A}^{(i,:)} &= [\nabla_z g(m^{(i)}_k(z), r^{(i)}_k(z))]_{z=z^*}, \\ \check{b}^{(i)} &= g(m^{(i)}_k(z^*), r^{(i)}_k(z^*)) - \check{A}^{(i,:)} z^*, \\ [\cdot, \hat{A}^{(i,:)} z + \hat{b}^{(i)}] &= (m^{(i)}_k, \hat{r}^{(i)}_k)^T \otimes [\check{g}, \hat{g}], \end{aligned}$$

with $z^* \in Z$ and ∇_z being the gradient w.r.t. z, it holds that $\forall z \in Z$, $\mathbb{E}_{\boldsymbol{\zeta} \sim \mathcal{N}(m_k(z);\, s_k(z))}\left[\hat{V}_k(ReLU(\boldsymbol{\zeta}))\right] \in \hat{V}_k \otimes [\check{A}z + \check{b}, \hat{A}z + \hat{b}]$.

The proof of Proposition 2 is based on the convexity of the expected value of a rectified Gaussian w.r.t. its mean and variance. The proof and detailed procedures for obtaining affine relaxations of g and r are reported in Appendix B.1. Next, we show how the result of Proposition 2 can be extended to PWA relaxations of the value functions.

Remark 3. *The results of Proposition 2 (as well as Propositions 4 and 5 below) extend to any continuous activation function ϕ_k. That is, as shown in (Benussi et al., 2022), every continuous activation function can be under and over-approximated by PWA functions $\check{\phi}_k, \hat{\phi}_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ such that $\check{\phi}_k \leq \phi_k \leq \hat{\phi}_k$. Consequently, $\mathbb{E}\left[\check{\phi}_k(\boldsymbol{\zeta})\right] \leq \mathbb{E}\left[\phi_k(\boldsymbol{\zeta})\right] \leq \mathbb{E}\left[\hat{\phi}_k(\boldsymbol{\zeta})\right]$, which allows the extension of Proposition 2 from ReLU to general continuous ϕ_k.*

5.2. Piecewise Affine Value Functions

For $N \in \mathbb{N}$, let $\mathcal{Z}_k = \{Z_{k,1}, \dots, Z_{k,N}\} \subseteq \mathbb{R}^{n_k}$ be a partition of the support of $f^{\boldsymbol{w}}_{0:k}$, and let $\check{V}_{k,j}, \hat{V}_{k,j} : \mathbb{R}^{n_k} \to \mathbb{R}^l$ be an affine relaxation of $V_k(z_k)$ w.r.t. $Z_{k,j}$ for all $j \in \{1, \dots, N\}$, i.e., $\forall z_k \in Z_{k,j}$ $V_k(z_k) \leq \hat{V}_{k,j}$ with $\hat{V}_{k,j} := \hat{A}_{k,j} z_k + \hat{b}_{k,j}$. Then, by Eqn. (5b) and the law of total expectation, we obtain an upper bound on V_{k-1}:

$$V_{k-1}(z) \leq \sum_{j=1}^{N} \hat{b}_{k,j} \underbrace{\mathbb{P}_{\boldsymbol{\zeta} \sim \mathcal{N}(m_k(z);\, \mathrm{diag}(s_k(z)))}\left[\boldsymbol{\zeta} \in Z_{k,j}\right]}_{\text{9a}} + \hat{A}_{k,j} \underbrace{\tilde{\mathbb{E}}_{\boldsymbol{\zeta} \sim \mathcal{N}(m_k(z);\, \mathrm{diag}(s_k(z)))}\left[\phi_k(\boldsymbol{\zeta}) \mid \boldsymbol{\zeta} \in Z_{k,j}\right]}_{\text{9b}}, \quad (9)$$

where $\tilde{\mathbb{E}}_{\boldsymbol{\zeta} \sim p}\left[\boldsymbol{\zeta} \mid \boldsymbol{\zeta} \in Z\right] := \mathbb{E}_{\boldsymbol{\zeta} \sim p}\left[\boldsymbol{\zeta} \mid \boldsymbol{\zeta} \in Z\right] \mathbb{P}_{\boldsymbol{\zeta} \sim p}\left[\boldsymbol{\zeta} \in Z\right]$. The lower bound on V_{k-1} follows similarly. Term 9a is

simply the probability that a Gaussian random variable ($\boldsymbol{\zeta}$) is in a given set (partition $Z_{k,j}$). If the partition is hyper-rectangular, in Lemma 3 we express Term 9a in closed-form.

Lemma 3. *For $k \in \{1, \ldots, K\}$, $\check{\zeta}, \hat{\zeta} \in \mathbb{R}^{n_k}$, it holds that* [7]

$$\mathbb{P}_{\boldsymbol{\zeta} \sim \mathcal{N}(m_k(z);\, diag(s_k(z)))} \left[\boldsymbol{\zeta} \in [\check{\zeta}, \hat{\zeta}]\right] = \tag{10}$$
$$\frac{1}{2^{n_k}} \prod_{i=1}^{n_k} erf\left(\frac{\hat{\zeta}^{(i)} - m_k^{(i)}(z)}{\sqrt{2 s_k^{(i)}(z)}}\right) - erf\left(\frac{\check{\zeta}^{(i)} - m_k^{(i)}(z)}{\sqrt{2 s_k^{(i)}(z)}}\right)$$

Term 9b is the conditional expectation of the random variable propagated through an activation function. The following shows that we can decompose this term in expectations, which we can bound using the result of Proposition 2, and probabilities for which Lemma 3 can be applied.

Proposition 4. *For $k \in \{1, \ldots, K\}$, vectors $\check{\zeta}, \hat{\zeta} \in \mathbb{R}^{n_k}$, and $\boldsymbol{\zeta} \sim \mathcal{N}\left(m_k(z);\, diag\left(s_k(z)\right)\right)$, it holds that*[8]

$$\tilde{\mathbb{E}}\left[ReLU\left(\boldsymbol{\zeta}\right) \mid \boldsymbol{\zeta} \in [\check{\zeta}, \hat{\zeta}]\right] =$$
$$\mathbb{E}\left[ReLU\left(\boldsymbol{\zeta} + [\check{\zeta}]_+\right)\right] - \mathbb{E}\left[ReLU\left(\boldsymbol{\zeta} + [\check{\zeta}]_+\right)\right] +$$
$$[\check{\zeta}]_+ \mathbb{P}\left[\boldsymbol{\zeta} \in [[\check{\zeta}]_+, \infty)\right] - [\hat{\zeta}]_+ \mathbb{P}\left[\boldsymbol{\zeta} \in [[\hat{\zeta}]_+, \infty)\right].$$

Next, we show how these results can be extended to unbounded sets in partition $\mathcal{Z}_k$, i.e., unbounded support $f^{\boldsymbol{w}}_{0:k}$.

Unbounded Support If $f^{\boldsymbol{w}}_{0:k}$ has an unbounded support, then there must necessarily be at least a region that is unbounded in the partition $\mathcal{Z}_k$. While for this region we can still apply Lemma 3 to compute Term 9a, we cannot use Proposition 4 to compute a bound for Term 9b. Instead, we rely on Proposition 5 (below), which derives relaxations based on the fact that Gaussian distributions decay exponentially fast (thus, faster than a linear function).

Proposition 5. *For $k \in \{1, \ldots, K\}$, $i \in \{1, \ldots, n_k\}$, and vector $\check{\zeta} \in \mathbb{R}^{n_k}$, it holds that*[9]

$$\frac{1}{2}[m_k^{(i)}(z)]_- \leq$$
$$\tilde{\mathbb{E}}_{\boldsymbol{\zeta} \sim \mathcal{N}(m_k(z);\, diag(s_k(z)))}\left[ReLU\left(\boldsymbol{\zeta}\right) \mid \boldsymbol{\zeta} \in [\check{\zeta}, \infty)\right]$$
$$\leq \frac{1}{2}[m_k^{(i)}(z)]_+ + \sqrt{\frac{s_k^{(i)}(z)}{2\pi}}.$$

[7] A similar result holds for unbounded regions defined by vector $\check{z}$, that is, $\boldsymbol{\zeta} \in [\check{z}, \infty)$ or $\boldsymbol{\zeta} \in (\infty, \check{z}]$, as shown in Appendix B.2.

[8] A similar relation can be obtained for ϕ_k being the identity function, as shown in Appendix B.3.

[9] A similar relation can be obtained for ϕ_k being the identity function, as shown in Appendix B.4.

Algorithm 1 Adversarial Robustness for Classification

function Classification(T, $\{N_k\}_{k=1}^{K+1}$)
  for $k \in \{1, \ldots, K+1\}$ **do**
    $Z_{k,main} = [\check{\zeta}_k, \hat{\zeta}_k]$ `(Prop 7)`
    $\{Z_{k,j}\}_{j=1}^{N_k-1} =$ `REFINE`$(Z_{k,main})$
    $\mathcal{Z}_k = \{Z_{k,j}\}_{j=1}^{N_k-1} \bigcup Z^C_{k,main}$
  end for
  $\{[\check{V}_{K+1,j}, \hat{V}_{K+1,j}]\}_{j=1}^{N_{K+1}} =$ `IBPSoftmax`$(\mathcal{Z}_{K+1})$
  for $k \in \{K+1, \ldots, 1\}$, **for** $l \in \{1, \ldots, N_k\}$ **do**
    $[\check{V}_{k-1,l}, \hat{V}_{k-1,l}] =$
      `BP`$(\{[\check{V}_{k,j}, \hat{V}_{k,j}]\}_{j=1}^{N_k}, \mathcal{Z}_k, \mathcal{Z}^{(l)}_{k-1})$
  end for
  Return: $\min_{x \in T} \check{V}_0(x)$, $\max_{x \in T} \hat{V}_0(x)$
end function

5.3. Relaxation of the Last Layer of BNN

We show how to compute interval relaxations of Eqn (5a). For the regression case ($h = I$), the process is simple since Eqn. (5a) becomes an affine function. That is, $V_K(z) = m_K(z)$, where $m_K(z)$ as defined in Eqn. (8), and hence no relaxation is required. For classification, however, further relaxations are needed because $h =$ softmax, i.e., the output distribution of the BNN (the logit) is propagated through the softmax. The following proposition shows that an interval relaxation can be obtained by relaxing the distribution of $\boldsymbol{W}_K(z^T, 1)^T$ by Dirac delta functions on the extremes of h for each set in the partition of the BNN's output.

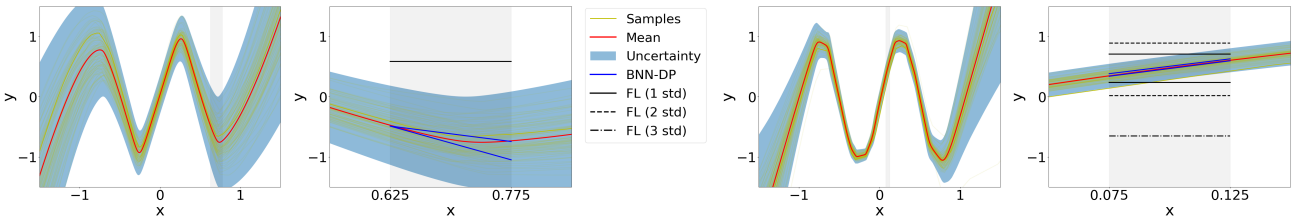
Proposition 6. *For $N \in \mathbb{N}$, let $\{Z_1, \ldots, Z_N\} \subseteq \mathbb{R}^{n_{K+1}}$ be a partition of supp$(f^{\boldsymbol{w}}(x))$. Then, for $i \in \{1, \ldots, n_{K+1}\}$ and $\mathbf{w} \sim q(\cdot)$, it holds that*

$$\sum_{j=1}^{N} [\min_{\zeta \in Z_j} h^{(i)}(\zeta)] \mathbb{P}\left[f^{\boldsymbol{w}}(x) \in Z_j\right] \leq \mathbb{E}\left[h^{(i)}(f^{\boldsymbol{w}}(x)\right] \leq$$
$$\sum_{j=1}^{N} [\max_{\zeta \in Z_j} h^{(i)}(\zeta)] \mathbb{P}\left[f^{\boldsymbol{w}}(x) \in Z_j\right].$$

A particularly simple case is when there are only two sets in the partition of the BNN's output layer. Then, the following corollary of Proposition 6 guarantees that, similarly to deterministic NNs (Zhang et al., 2018b), we can determine adversarial robustness by simply looking at the logit.

Corollary 6.1. *Let $\{[\check{\zeta}, \hat{\zeta}], Z\} \subseteq \mathbb{R}^{n_{K+1}}$ be a partition of supp$(f^{\boldsymbol{w}})$. Then, for $i, j \in \{1, \ldots, n_{K+1}\}$ and $\boldsymbol{w} \sim q(\cdot)$, it holds that*

$$e^{\hat{\zeta}^{(j)}} - e^{\check{\zeta}^{(i)}} + \left(\frac{1}{\mathbb{P}\left[f^{\boldsymbol{w}}(x) \in [\check{\zeta}, \hat{\zeta}]\right]} - 1\right) \sum_{l=1}^{n_{K+1}} e^{\hat{\zeta}^{(l)}} \leq 0$$
$$\implies \mathbb{E}\left[softmax^{(j)}(f^{\boldsymbol{w}}(x)) - softmax^{(i)}(f^{\boldsymbol{w}}(x))\right] \leq 0.$$



(a) BNN with 1 hidden layer and 2048 nodes.

(b) BNN with 2 hidden layers and 64 nodes per layer.

Figure 2. Certified affine bounds on the mean of BNNs trained on the 1D Noisy Sine dataset w.r.t. the grey marked interval of input.

6. BNN-DP Algorithm

We summarize our overall procedure to solve Problem 1 in an algorithm called BNN-DP. Algorithm 1 presents BNN-DP for the classification setting; the procedure for the regression setting follows similarly and is provided in Appendix D. Algorithm 1 consists of a forward pass to partition the latent space of the BNN (Lines 2-4), and a backward pass to recursively approximate the value functions via Eqns. (5a) and (5b) (Lines 7-10). The last layer of the BNN (Eqn. 5a) is handled by the `IBPSoftmax` function in Line 7 using the results of Proposition 6. The `BP` function in Line 9 performs the back-propagation over the hidden layers of the BNN (Eqn. 5b) using the results of Lemma 3 and Proposition 4 and 5. The detailed procedures of `IBPSoftmax` and `BP` can be found in Appendix D. In what follows, we describe how we partition the support of the latent space of the BNN, and discuss the computational complexity of BNN-DP.

Partitioning Recall that our results rely on hyper-rectangular partitions. Hence, for each layer k, we employ the following proposition to find a hyper-rectangular subset of the support of each layer that captures at least $1-\epsilon$ of the probability mass of $\text{supp}(f^{\boldsymbol{w}}_{0:k})$.

Proposition 7. *For $k \in \{1,\dots,K\}$, let $\epsilon \in [0,1]$ be a constant, and $Z \subset \mathbb{R}^{n_{k-1}}$ be a compact set. Then, for vectors $\check{\zeta}_k, \hat{\zeta}_k \in \mathbb{R}^{n_k}$ defined such that $\forall i \in \{1,\dots,n_k\}$,*

$$\check{\zeta}_k^{(i)} = \max_{z\in Z}\left[\mathit{erf}^{-1}(-\eta)\sqrt{2s_k^{(i)}(z)} + m_k^{(i)}(z)\right], \quad (11)$$

$$\hat{\zeta}_k^{(i)} = \min_{z\in Z}\left[\mathit{erf}^{-1}(\eta)\sqrt{2s_k^{(i)}(z)} + m_k^{(i)}(z)\right], \quad (12)$$

where $\eta = (1-\epsilon)^{\frac{1}{n_k}}$, it holds that, $\forall z \in Z$,

$$\mathbb{P}_{\boldsymbol{\zeta}\sim\mathcal{N}(m_k(z);\, \mathit{diag}(s_k(z)))}\left[\boldsymbol{\zeta} \in [\check{\zeta}_k, \hat{\zeta}_k]\right] \geq 1-\epsilon.$$

Here, Eqns (11) and (12) are convex minimization problems, which can be efficiently solved via, e.g., the gradient descent algorithm. We denote the resulting region obtained via Proposition 7 as $Z_{k,main} \subset \text{supp}(f^{\boldsymbol{w}}_{0:k})$. Then, $Z_{k,main}$ can be further refined by interval splitting.

Computational Complexity Similarly as for linear bounding procedures for deterministic neural networks, see e.g. [Zhang et al. 2018], the cost of computing piecewise-affine relaxations of a BNN with K layers and n neurons per layer is polynomial in both K and n. Refinement, which is not part of the main algorithm, has exponential cost in n. In practice, however, in NNs and consequently in BNNs, only a few neurons are generally active, and those are the ones that most influence the posterior (Frankle & Carbin, 2018). Therefore, the refining procedure can focus only on these neurons. Because of this, in almost all the experiments in Section 7, only 2 regions in the partition per hidden layer were required to certify robustness, even in cases where the BNN had large posterior variance and thousands of neurons.

7. Experimental Results

We empirically evaluated BNN-DP on various regression and classification benchmarks. We ran our experiments on an AMD EPYC 7252 8-core CPU and train the BNNs using Noisy Adam (Zhang et al., 2018a) and variational online Gauss-Newton (Khan et al., 2018). We first validate the bounds obtained by BNN-DP for BNNs trained on samples from an 1D sine with additive noise (referred to as the 1D Noisy Sine). We then analyse a set of BNNs with various architectures trained on the 2D dimensional equivalent of 1D Noisy Sine and the Kin8nm dataset.[10] The latter dataset contains state-space readings for the dynamics of an 8 link robot arm, and is commonly used as a regression task to benchmark BNNs (Hernández-Lobato & Adams, 2015; Gal & Ghahramani, 2016). Last, we turn our attention to classification and evaluate BNNs trained on the MNIST, Fashion MNIST and CIFAR-10 datasets. [11]

As a baseline for our experiments, we consider the state-of-the-art approach of Berrada et al. (2021), to which we refer as "FL". In fact, FL is the only existing method that can provide robustness certification for BNNs in similar settings as our BNN-DP. Nevertheless, we must remark that even

[10] Available at `http://www.cs.toronto.edu/~delve`.

[11] Our code is available at `https://github.com/sjladams/BNN_DP`.

Table 1. Comparison between BNN-DP and FL on various fully connected BNN architectures, with K being the number of hidden layers, and n_{hid} the number of neurons per layer. The results are the average over 100 test point, and the computation times are averaged over all architectures. The best values for each comparison are reported in bold.

(a) γ-Robustness Regression Tasks

			2D Noisy Sine			Kin8nm		
K	ϵ	n_{hid}	BNN-DP	FL (5 std)	FL (3 std)	BNN-DP	FL (5 std)	FL (3 std)
1	1e-2	64	**0.041**	1.8	0.8	**0.044**	0.7	0.3
		256	**0.04**	3.0	1.2	**0.040**	45.4	24.7
		512	**0.039**	6.4	2.5	**0.041**	12.6	6.0
2	1e-3	64	**0.109**	718.1	101.1	**0.070**	31.3	10.8
		128	**0.239**	112.2	20.1	**0.240**	1459.8	420.9
		256	**0.376**	599.3	92.1	**0.968**	9420.9	2715.9
3	5e-4	64	**0.477**	699.2	74.8	**0.348**	12638.5	59304.6
		128	**0.629**	11214.4	1142.8	**0.964**	433149.4	232811.6
		256	**14.180**	275408.1	2882.3	**69.488**	3441470.8	21877545.4
Cmp. Time (sec.)			8.0	7.8	**7.7**	13.2	**7.5**	7.6

(b) ϵ-Robustness Classification Tasks

		MNIST			Fashion MNIST		
K	n_{hid}	BNN-DP	FL (5 std)	FL (3 std)	BNN-DP	FL (5 std)	FL (3 std)
1	64	**0.0150**	0.0090	0.0102	**0.0128**	0.0077	0.008
	128	**0.0145**	0.0091	0.0131	**0.0065**	0.0041	0.0045
	256	**0.0137**	0.0082	0.0090	**0.0081**	0.0043	0.0046
	512	**0.0131**	0.0070	0.0073	**0.0092**	0.0044	0.0048
2	64	**0.0073**	0.0041	0.0042	**0.0048**	0.0024	0.0026
	128	**0.0062**	0.0028	0.0035	**0.0021**	0.0018	0.0019
	256	**0.0049**	0.0023	0.0023	**0.0032**	0.0016	0.0016
3	64	**0.0032**	0.0014	0.0016	**0.0015**	0.0006	0.0008
	256	**0.0018**	0.0009	0.0009	**0.0007**	0.0006	**0.0007**
Cmp. Time (sec)		**15.2**	859.2	805.3	**22.1**	767.6	760.2

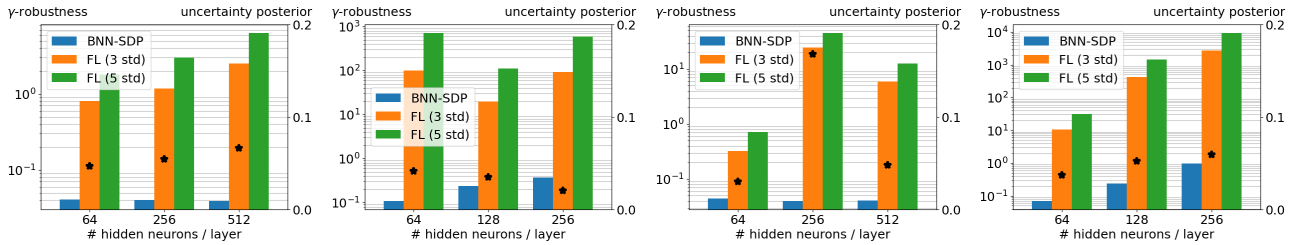
FL is not fully formal; it works by truncating the Gaussian posterior distribution associated to each weight at a given multiple of its standard deviation (std), disregarding a large part of the posterior distribution. Hence, the returned bound is not sound over the full posterior but only a subset of it. More importantly, the disregarded portion of the posterior grows exponentially with the number of weights of the networks. Already for a two hidden layer BNN with 48 neurons per layer, FL verifies only 0.1% of the BNN posterior when truncated at 3 std. Thus, the bounds computed by FL are optimistic and not mathematically guaranteed to hold. In contrast, not only BNN-DP returns formal bounds accounting for the whole posterior, but also the benchmark results show that BNN-DP bounds are much tighter than FL ones.

7.1. Bound Validation

We validate and qualitatively compare the bounds obtained by BNN-DP and FL on BNNs with 1 and 2 hidden layers trained on 1D Noisy Sine. The results of these analyses are reported in Figure 2. Visually, we see that BNN-DP is able to compute tight affine relaxations (blue lines) on the mean of the BNNs over the grey shaded intervals. In contrast, already in this simple scenario, and even when truncating the posterior distribution at just 1 std, FL returns as guaranteed output intervals $[-1.68, 0.59]$ and $[0.23, 0.71]$ for the 1 and 2 hidden layer BNN, respectively. Hence, even though FL disregards most of the BNNs posterior, BNN-DP still produces tighter bounds. When using 3 std, the FL interval bounds become even wider, that is $[-7.07, 9.31]$ and $[-0.65, 1.54]$, for the 1 and 2 hidden layer BNN, respectively. Intuitively, the major improvement of the bounds can be explained by the fact that, while BNN-DP directly averages the uncertainty of each layer by solving the DP in Theorem 1, FL solves an overall optimisation problem that at each layer considers the worst combination of parameters in the support of the (truncated) distribution, leading to conservative bounds. In fact, the bound computed by FL is looser in the one-hidden layer case than in the two-hidden layers one by one order of magnitude, precisely because of the higher variance of the former BNN compared to the second. In what follows, we see that analogous observations apply to more complex deep learning benchmarks.

7.2. Regression Benchmarks

We consider a set of BNNs with various architectures trained on the 2D Noisy Sine and Kin8nm regression datasets. To asses the certification performance of BNN-DP, we compute the difference between the upper and lower bounds on the expectation of the BNNs, referred to as the γ-robustness, for its input in a ℓ_∞-norm ball of radius ϵ centered at a sampled data point. Clearly, a smaller value of γ implies a tighter bound computation. Results, averaged over 100 randomly sampled test points, are reported in Table 1a. For all experiments, BNN-DP greatly improves the value of γ-robustness provided by the FL-baseline by 1 to 4 orders of magnitude with similar computation times. We also note that the larger the BNN is, the larger the improvement in the (tightness of the) bounds are, which empirically demonstrates the superior scalability of BNN-DP. Figure 3 explicitly shows the impact of the model size and variance on the certified γ-robustness. For BNNs with 1 hidden layer, BNN-DP guarantees small γ-robustness (and hence tighter bounds) irrespective of the number of neurons as well as the amount of uncertainty. In contrast, as already observed for the 1D Noisy Sine case, FL is particularly impacted by the variance of the posterior distribution. For BNNs with two hidden layers, BNN-DP requires partitioning the latent space, which leads to a positive correlation with the value of γ-robustness and the number of hidden neurons. A similar, but more extreme, trend is also observed for FL.



(a) 1 hidden layer - 2D NS (b) 2 hidden layers - 2D NS (c) 1 hidden layer Kin8nm (d) 2 hidden layers Kin8nm

Figure 3. Analysis of γ-robustness illustrated as bars (in log-scale), and uncertainty of the posterior distribution (black stars), for BNNs architectures trained on 2D Noisy Sine (NS) and Kin8nm.

7.3. Classification Benchmarks

We now evaluate BNN-DP on the MNIST, Fashion MNIST and CIFAR-10 classification benchmarks. In order to quantitatively measure the robustness of an input point x^*, we consider the maximum radius ϵ for which the decisions on ℓ_∞-norm perturbations of x^* with radii ϵ is invariant. That is, any perturbation of x^* smaller than ϵ does not change the classification output; hence, the larger ϵ in x^*, the more robust the BNN in the specific point. Results are reported in Table 1b and 2. For the fully connected BNN architectures, BNN-DP not only is able to certify a substantially larger ϵ compared to the baseline, but also it does so by orders of magnitude smaller computation time. This is because our approach uses interval relaxations (Proposition 6) to bound the softmax, whereas FL explicitly considers a non-convex optimization problem, which is computationally demanding. For the Bayesian CNN architectures, FL is able to certify a slightly larger ϵ, at the costs of magnitudes of orders increase of computation time. This can be explained by the decreasing support of the BNN posterior certified by FL for increasing network size, whereas, the ϵ certified by BNN-DP holds for the whole posterior.

Table 2. Comparison of the ϵ-robustness obtained with BNN-DP and FL for various BNN architectures, with K_{conv} convolutional layers concatenated to K fully connected hidden layers, and n_{hid} neurons per fully connected layer. The convolutional layers have n_{kern} kernels of size 4×4 with stride 1. Inference on the convolutional and linear layers is performed using Dropout and Bayes by Backprop, respectively. The results are the average over 100 test points, and the computation times are averaged over all architectures. The best values for each comparison are reported in bold.

Dataset	K_{conv}	n_{kern}	K	n_{hid}	BNN-DP	FL (5 std)	FL (3 std)
Fashion MNIST	1	2	1	64	0.00065	0.00112	**0.00122**
	2	2	1	64	0.00061	0.00109	**0.00117**
	Cmp. Time (sec)				**3.7**	545.5	319.3
CIFAR-10	2	4	0	-	0.00007	0.00009	**0.00010**
	3	3	0	-	0.00011	0.00019	**0.00021**
	Cmp. Time (sec)				**1.8**	250.7	201.2

8. Conclusion

We introduced BNN-DP, an algorithmic framework to certify adversarial robustness of BNNs. BNN-DP is based on a reformulation of adversarial robustness for BNNs as a solution of a dynamic program, for which efficient relaxations can be derived. Our experiments on multiple datasets for both regression and classification tasks show that our approach greatly outperforms state-of-the-art competitive methods, thus paving the way for applications of BNNs in safety-critical applications.

Acknowledgements

This work was supported in part by the NSF grant 2039062.

References

Benussi, E., Patanè, A., Wicker, M., Laurenti, L., and Kwiatkowska, M. Individual fairness guarantees for neural networks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 651–658. International Joint Conferences on Artificial Intelligence Organization, 2022. doi: 10.24963/ijcai.2022/92.

Berrada, L., Dathathri, S., Dvijotham, K., Stanforth, R., Bunel, R. R., Uesato, J., Gowal, S., and Kumar, M. P. Make sure you're unsure: A framework for verifying probabilistic specifications. *Advances in Neural Information Processing Systems*, 34:11136–11147, 2021.

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

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. pp. 1613–1622, 2015.

Bunel, R., De Palma, A., Desmaison, A., Dvijotham, K., Kohli, P., Torr, P., and Kumar, M. P. Lagrangian decomposition for neural network verification. In *Conference*

on Uncertainty in Artificial Intelligence, pp. 370–379. PMLR, 2020.

Carbone, G., Wicker, M., Laurenti, L., Patanè, A., Bortolussi, L., and Sanguinetti, G. Robustness of bayesian neural networks to gradient-based attacks. *Advances in Neural Information Processing Systems*, 33:15602–15613, 2020.

Cardelli, L., Kwiatkowska, M., Laurenti, L., Paoletti, N., Patanè, A., and Wicker, M. Statistical guarantees for the robustness of Bayesian neural networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 5693–5700. AAAI Press, 2019a.

Cardelli, L., Kwiatkowska, M., Laurenti, L., and Patanè, A. Robustness guarantees for Bayesian inference with Gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7759–7768, 2019b.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

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

Harva, M. et al. *Hierarchical variance models of image sequences*. PhD thesis, Master's thesis, Helsinki University of Technology, Espoo, 2004.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pp. 1861–1869. PMLR, 2015.

Kahn, G., Villaflor, A., Pong, V., Abbeel, P., and Levine, S. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International Conference on Machine Learning*, pp. 2611–2620. PMLR, 2018.

Lechner, M., Zikelic, D., Chatterjee, K., and Henzinger, T. Infinite time horizon safety of bayesian neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

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

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

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Marchi, M., Gharesifard, B., and Tabuada, P. Training deep residual networks for uniform approximation guarantees. In *Learning for Dynamics and Control*, pp. 677–688. PMLR, 2021.

McAllister, R., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., and Weller, A. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc., 2017.

Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Osawa, K., Swaroop, S., Khan, M. E. E., Jain, A., Eschenhagen, R., Turner, R. E., and Yokota, R. Practical deep learning with bayesian principles. *Advances in neural information processing systems*, 32, 2019.

Patanè, A., Blaas, A., Laurenti, L., Cardelli, L., Roberts, S., and Kwiatkowska, M. Adversarial robustness guarantees for gaussian processes. *Journal of Machine Learning Research*, 23:1–55, 2022.

Rawat, A., Wistuba, M., and Nicolae, M.-I. Adversarial phenomenon in the eyes of bayesian deep learning. *arXiv preprint arXiv:1711.08244*, 2017.

Smith, L. and Gal, Y. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018.

Smith, M. T., Grosse, K., Backes, M., and Alvarez, M. A. Adversarial vulnerability bounds for gaussian process classification. *Machine Learning*, pp. 1–39, 2022.

Socci, N., Lee, D., and Seung, H. S. The rectified gaussian distribution. *Advances in neural information processing systems*, 10, 1997.

Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Daniel, L., and Dhillon, I. Towards fast computation of certified robustness for ReLU networks. In *International Conference on Machine Learning (ICML)*, 2018.

Wicker, M., Laurenti, L., Patanè, A., and Kwiatkowska, M. Probabilistic safety for Bayesian neural networks. *Uncertainty in Artificial Intelligence (UAI)*, 2020.

Wicker, M., Laurenti, L., Patanè, A., Chen, Z., Zhang, Z., and Kwiatkowska, M. Bayesian inference with certifiable adversarial robustness. *AISTATS*, 2021.

Winn, J., Bishop, C. M., and Jaakkola, T. Variational message passing. *Journal of Machine Learning Research*, 6 (4), 2005.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5286–5295. PMLR, 2018.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pp. 5852–5861. PMLR, 2018a.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. *arXiv preprint arXiv:1811.00866*, 2018b.

A. Proofs Section 4

A.1. Proof Theorem 1

By the law of total expectation and because of the independence of the weights distribution at different layers it holds that

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h(f^{\boldsymbol{w}}(x))\right] &= \mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[\mathbb{E}_{\boldsymbol{W}_K\sim q(\cdot)}\left[h(\boldsymbol{W}_K(f^{\boldsymbol{w}}_{0:K}(x)^T,1)^T)\right]\right] \\
&= \mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[V_K(f^{\boldsymbol{w}}_{0:K}(x))\right] \\
&= \mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[\mathbb{E}_{\boldsymbol{W}_{K-1}\sim q(\cdot)}\left[V_K(\phi_K(\boldsymbol{W}_{K-1}(f^{\boldsymbol{w}}_{0:K-1}(x)^T,1)^T))\right]\right] \\
&= \mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[V_{K-1}(f^{\boldsymbol{w}}_{0:K-1}(x))\right]
\end{aligned}$$

Repeating this procedure backwards over the layers of the neural networks, we obtain $\mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h(f^{\boldsymbol{w}}(x))\right] = V_0(x)$.

A.2. Corollary 1.1

If, for $k \in \{1,\dots,K\}$, we have that $\forall z_k \in \mathbb{R}^{n_k}$, $\check{V}_k(z_k) \leq V_k(z_k) \leq \hat{V}_k(z_k)$. Then, for any probability density distribution $p : \mathbb{R}^{n_k} \to \mathbb{R}_{\geq 0}$ it holds that

$$\int_{\text{supp}(p)} \check{V}_k(z)p(z)dz \leq \int_{\text{supp}(p)} V_k(z)p(z)dz \leq \int_{\text{supp}(p)} \hat{V}_k(z)p(z)dz,$$

or rewritten in terms of expectations,

$$\mathbb{E}_{\boldsymbol{z}\sim p(\cdot)}\left[\check{V}_k(\boldsymbol{z})\right] \leq \mathbb{E}_{\boldsymbol{z}\sim p(\cdot)}\left[V_k(\boldsymbol{z})\right] \leq \mathbb{E}_{\boldsymbol{z}\sim p(\cdot)}\left[\hat{V}_k(\boldsymbol{z})\right].$$

Furthermore, by Theorem 1, $\check{V}_0(x) \leq \mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h(f^{\boldsymbol{w}}(x))\right] \leq \hat{V}_0(x)$. Consequently, for $i \in \{1,\dots,l\}$, it holds that

$$\min_{x\in T} \mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h(f^{\boldsymbol{w}}(x))\right] \geq \min_{x\in T} \check{V}_0^{(i)}(x), \qquad \max_{x\in T} \mathbb{E}_{\boldsymbol{w}\sim q(\cdot)}\left[h(f^{\boldsymbol{w}}(x))\right] \leq \max_{x\in T} \hat{V}_0^{(i)}(x).$$

B. Proofs Section 5

For the proof of Proposition 2 we rely on some properties of *rectified Gaussian Distributions* (Harva et al., 2004; Winn et al., 2005; Socci et al., 1997) that we will first introduce below.

In the special case of ϕ_k being the ReLU function, the dynamics of the BNN over hidden layer k can be described by the so-called *rectified Gaussian Distribution*, as illustrated in Figure 4, and formally defined as follows.

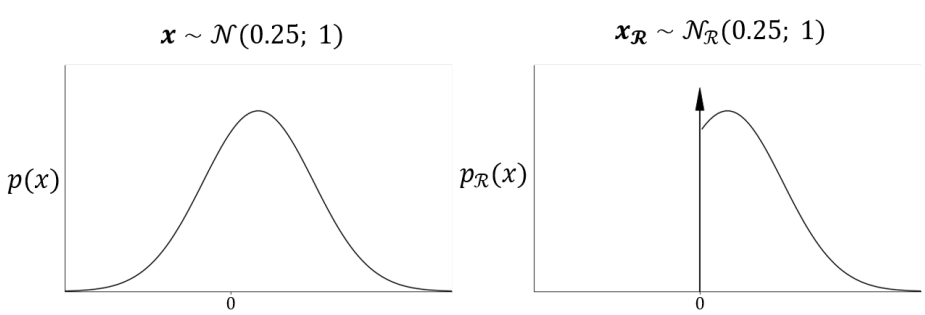


Figure 4. Gaussian probability density function and its related rectified version.

Definition 4 (Rectified Gaussian Distribution)**.** *Let $\boldsymbol{z}$ be a random variable with Gaussian distribution $\mathcal{N}\left(\mu;\ \sigma^2\right)$ with mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}_{\geq 0}$. Then, $\max(\boldsymbol{z}, 0)$ is a rectified Gaussian random variable.*

The probability density function (pdf) of rectified Gaussian distributions is obtained by a mixture of a discrete distribution at $z = 0$ and a truncated Gaussian distribution with interval $(0, \infty)$, that is the distribution of $\max(\boldsymbol{z}, 0)$ is:

$$\mathcal{N}_{\mathcal{R}}\left(z \mid \mu;\ \sigma^2\right) := \Phi\left(0 \mid \mu; \sigma^2\right)\delta(z) + \mathcal{N}\left(z \mid \mu;\ \sigma^2\right)U(x),$$

where $\Phi : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is the cdf of the standard normal distribution, δ is the Dirac delta function, and U is the unit step function.

There are various important properties of a rectified Gaussian random variable that we will rely on in this proof. First of all, there exists a closed-form expression for the expected value of rectified Gaussian distributions (Harva et al., 2004). Hence, in the case of ϕ_k being the ReLU function, we have a closed form-expression for Equation (7). Second, as a consequence of the convexity properties of the expectation of rectified Gaussian variables presented in the following two Lemmas, we can employ convexity to efficiently find an affine relaxation of Equation (7).

Lemma 8. *For $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}_{\geq 0}$, it holds that $\mathbb{E}_{\boldsymbol{z} \sim \mathcal{N}_{\mathcal{R}}(\mu;\, \sigma^2)}[\boldsymbol{z}]$ is convex w.r.t. $(\mu, \sigma)^T$.*

Proof. Let us denote the closed-form expression for the expected value of rectified Gaussian distributions as derived in (Harva et al., 2004) by function $g : \mathbb{R}^2 \to \mathbb{R}$, that is, we have that,

$$\mathbb{E}_{\boldsymbol{z} \sim \mathcal{N}_{\mathcal{R}}(\mu;\, \sigma^2)}[\boldsymbol{z}] \coloneqq g(\mu, \sigma) = \frac{\mu}{2}\left[1 - \operatorname{erf}\left(\frac{-\mu}{\sigma\sqrt{2}}\right)\right] + \frac{\sigma}{\sqrt{2\pi}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right). \tag{13}$$

Then,

$$\begin{aligned}
\frac{\partial^2 g}{\partial \mu^2} &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) \\
\frac{\partial^2 g}{\partial \sigma^2} &= \frac{\mu^2}{\sigma^3\sqrt{2\pi}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) \\
\frac{\partial^2 g}{\partial \sigma \partial \mu} &= -\frac{\mu}{\sigma^2\sqrt{2\pi}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right)
\end{aligned}$$

Therefore, the Hessian of g w.r.t. $(\mu, \sigma)^T$, that is,

$$H_{(\mu,\sigma)} = \begin{pmatrix} \frac{\partial^2 g}{\partial \mu^2} & \frac{\partial^2 g}{\partial \sigma \partial \mu} \\ \frac{\partial^2 g}{\partial \sigma \partial \mu} & \frac{\partial^2 g}{\partial \sigma^2} \end{pmatrix}$$

can be written as

$$H_{(\mu,\sigma)} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) cc^T$$

where $c = (1, \frac{1}{\sigma})^T \in \mathbb{R}^2$. To determine whether $H_{(\mu,\sigma)}$ is semipositive definite (spd), that is, whether $\forall z \in \mathbb{R}^2, z^T H_{(\mu,\sigma)} z \geq 0$, first observe that $\forall (\mu, \sigma) \in \mathbb{R} \times \mathbb{R}_{\geq 0}$, we have that $\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) \geq 0$. Next, we observe that cc^T is spd, because cc^T is a symmetric matrix with all (leading) principal minors equal to zero. Hence, $H_{(\mu,\sigma)}$ is spd and, consequently, g is convex w.r.t. (μ, σ). □

Lemma 9. *Let $m : \mathbb{R}^n \to \mathbb{R}$ be a linear function defined as $m(x) \coloneqq \mu^T x$, with $\mu \in \mathbb{R}^n$ and $s : \mathbb{R}^n \to \mathbb{R}$ be a quadratic function $s(x) \coloneqq x^T \Sigma x$ with $\Sigma \in \mathbb{R}^{n \times n}$ a positive definite matrix, then it holds that $\mathbb{E}_{\boldsymbol{z} \sim \mathcal{N}_{\mathcal{R}}(m(x);\, s(x))}[\boldsymbol{z}]$ is convex w.r.t. $x \in \mathbb{R}^n$.*

Proof. Define $g : \mathbb{R}^n \to \mathbb{R}$ as in Eqn. (13). Then, the Hessian of g w.r.t. x, that is, $H_x = \frac{\partial^2 g}{\partial x^2}$, can be written as

$$H_x = \frac{1}{\sqrt{2\pi s(x)}}(cc^T + \Sigma - dd^T),$$

where $c = \mu - \frac{m(x)}{s(x)}\Sigma x$ and $d = \frac{1}{\sqrt{s(x)}}\Sigma$. To prove the convexity of g w.r.t. x, it suffices to show that matrix H_x is spd for all $x \in \mathbb{R}^n$. We can conclude directly that $\frac{1}{\sqrt{2\pi s(x)}} > 0$ and cc^T is a spd matrix, since for any $c \in \mathbb{R}^n$, cc^T is a symmetric matrix with all (leading) principal minors equal to zero. Next, we observe that

$$xx^T(\Sigma - dd^T) = xx^T\left(\Sigma - \frac{\Sigma x x^T \Sigma}{x^T \Sigma x}\right) = 0_{n,n}$$

where $0_{n,n} \in \mathbb{R}^{n \times n}$ is the zero matrix. Notice that both matrices xx^T and $0_{n,n}$ are spd. Then, since that the product of two symmetric psd matrices is psd iff its product is symmetric, we conclude that matrix $\Sigma - dd^T$ is psd. Then, since the sum of spd matrices is spd, H_x is spd and consequently g is convex w.r.t. x. □

B.1. Proof Proposition 2

Without any loss of generality, we assume that $m : \mathbb{R}^n \to \mathbb{R}$ is a linear function defined as $m(z) = \mu^T z$, with $\mu \in \mathbb{R}^n$ and $s : \mathbb{R}^n \to \mathbb{R}$ is a quadratic function $s(z) = z^T \Sigma z$, with $\Sigma \in \mathbb{R}^{n \times n}$ being a positive definite matrix, and define the function $r : \mathbb{R}^n \to \mathbb{R}$ as $r(z) = \sqrt{s(z)}$.

The expectation of a rectified Gaussian distributed variable is given by function $g : \mathbb{R}^2 \to \mathbb{R}^1$ as defined in Eqn. (13), such that

$$\mathbb{E}_{\boldsymbol{\zeta} \sim \mathcal{N}_{\mathcal{R}}(m(z);\, s(z))}\left[\boldsymbol{\zeta}\right] = g(m(z), r(z)).$$

Since g is convex w.r.t. z as shown in Lemma 9, for $z \in Z$, g can be lower bounded by its tangent at some point $z^* \in Z$, that is, for $\check{a} \in \mathbb{R}^n$ and $\check{\beta} \in \mathbb{R}$ defined as

$$\check{a} = [\nabla_z g(m(z^*), r(z))]_{x=z^*}, \qquad \check{\beta} = g(m(z^*), r(z^*)) - \check{a}^T z^*,$$

where ∇_z denotes the gradient of g w.r.t.z, it holds that $g(m(z), r(z)) \geq \check{a}^T x + \check{\beta}$. Here, the gradient of g w.r.t. z in terms of $m(z)$ and $r(z)$ is given by

$$\nabla_z g(m(z), r(z)) = \frac{1}{2}\left[1 - \operatorname{erf}\left(\frac{-m(z)}{r(z)\sqrt{2}}\right)\right]\mu + \frac{\Sigma z}{r(z)\sqrt{2\pi}} \exp\left(-\left(\frac{m(z)}{r(z)\sqrt{2}}\right)^2\right).$$

By the convexity of g w.r.t. z, we could upper bound g by finding its maximum, located at on the boundary of Z, and fitting a hyperplane through the maximum and $n-2$ other points on the edge of Z. However, due to the potentially high dimensionality of the Z, this direct procedure is infeasible in practice.

Instead, we first find affine relaxations $\check{g}, \hat{g}$ of g w.r.t. $(m(z), r(z)) \in \{(m(z), r(z)) \mid \forall z \in Z\}$. After that, we use symbolic arithmetic to propagate affine relaxations $(m(z), \check{r}), (m(z), \hat{r})$ of $(m(z), r(z))$ w.r.t. $z \in Z$, through the symbolic interval $[\check{g}, \hat{g}]$ to obtain an affine relaxation of g w.r.t. $z \in Z$, that is

$$[\cdot, \hat{a}x + \hat{\beta}] = (m^{(i)}, \hat{s}^{(i)})^T \otimes [\check{g}, \hat{g}].$$

This completes the proof of the proposition. In the remainder, we explain how in practice $\check{g}$, $\hat{g}$ and $\hat{r}$ can be found.

We denote the set of possible $(m(z), r(z))$ as $P \subset \mathbb{R} \times \mathbb{R}_{\geq}$, that is $P := \{(m(z), r(z)) \mid \forall z \in Z\}$. To find an affine relaxation of g w.r.t. $(m(z), r(z)) \in P$, we use the result of Lemma 9, that states that g is convex w.r.t. (μ, σ), and for $\check{g}$ take the tangent of g at some point in P and compute $\hat{g}$ by fitting a hyper-plane through the largest 3 points on the edge of a convex over-approximation of P.

To find $\hat{r}$ such that $r(z) \leq \hat{r}(z)$, $\forall z \in Z$, we use symbolic arithmetic to propagate an affine relaxation $\check{s}, \hat{s}$ of s w.r.t. z through an affine relaxation $\check{r}_s, \hat{r}_s$ of r w.r.t. s. Notice that, since the square root is a strictly increasing function, such that $\hat{r}(z) = \hat{r}_s(\hat{s}(z))$, we only require $\hat{s}$ and $\hat{r}_s$ to find $\hat{r}_s$. As r is concave w.r.t. s, we take $\hat{r}_s$ the tangent of r at $s(z^*)$ with $z^* \in X$, that is $\hat{r}_s(z) \leq \frac{s(z)}{2\sqrt{s(z^*)}} + \frac{1}{2}\sqrt{s(z^*)}$. In the case that Σ is a diagonal matrix, finding $\hat{s}$ boils down to bounding n one-dimensional quadratic functions. In the case that Σ has non-diagonal terms, we first find a transformation matrix $T \in \mathbb{R}^{n \times n}$, such that $T^T \Sigma T$ becomes a diagonal matrix. Then, we bound s in the transformed space induced by T, and transform the bounds back to the original space using the inverse transformation to obtain $\hat{s}$.

B.2. Proof Lemma 3

Due to random variable ($\boldsymbol{\zeta}$) being independent for each dimension, the computation of the probability that $\boldsymbol{\zeta}$ is in hyper-rectangle $[\check{\zeta}_k, \hat{\zeta}_k]$ can be split over the dimension of $\boldsymbol{\zeta}$:

$$\mathbb{P}_{\boldsymbol{\zeta} \sim \mathcal{N}(m_k(z);\, \operatorname{diag}(s_k(z)))}\left[\boldsymbol{\zeta} \in [\check{\zeta}_k, \hat{\zeta}_k]\right] = \prod_{i \in \{1,\ldots,n\}} \mathbb{P}_{\boldsymbol{\zeta}^{(i)} \sim \mathcal{N}\left(m_k^{(i)}(z);\, s_k^{(i)}(z)\right)}\left[\boldsymbol{\zeta}^{(i)} \in [\check{\zeta}_k^{(i)}, \hat{\zeta}_k^{(i)}]\right]$$

where $\forall i \in \{1, \ldots, n\}$

$$\mathbb{P}_{\boldsymbol{\zeta}^{(i)} \sim \mathcal{N}\left(m_k^{(i)}(z);\, s_k^{(i)}(z)\right)} \left[\boldsymbol{\zeta}^{(i)} \in [\check{\zeta}_k^{(i)}, \hat{\zeta}_k^{(i)}]\right] = \frac{1}{2}\left[\text{erf}\left(\frac{\hat{\zeta}_k^{(i)} - m_k^{(i)}(z)}{\sqrt{2 s_k^{(i)}(z)}}\right) - \text{erf}\left(\frac{\check{\zeta}_k^{(i)} - m_k^{(i)}(z)}{\sqrt{2 s_k^{(i)}(z)}}\right)\right].$$

Using $\lim_{x\to\infty} \text{erf}\,(x) = 1$ and $\lim_{x\to\infty} \text{erf}\,(-x) = -1$, the above result can be extended to unbounded intervals as follows

$$\mathbb{P}_{\boldsymbol{\zeta}^{(i)} \sim \mathcal{N}\left(m_k^{(i)}(z);\, s_k^{(i)}(z)\right)} \left[\boldsymbol{\zeta}^{(i)} \in [\check{\zeta}_k^{(i)}, \infty)\right] = \frac{1}{2}\left[1 - \text{erf}\left(\frac{\check{\zeta}_k^{(i)} - m_k^{(i)}(z)}{\sqrt{2 s_k^{(i)}(z)}}\right)\right].$$

B.3. Proof Proposition 4

Since the distribution of $\boldsymbol{\zeta}$ has a diagonal covariance matrix and $[\check{\zeta}_k, \hat{\zeta}_k]$ is a hyper-rectangle, we have that

$$\begin{aligned}
\tilde{\mathbb{E}}\left[\phi_k(\boldsymbol{\zeta}) \mid \boldsymbol{\zeta} \in [\check{\zeta}_k, \hat{\zeta}_k]\right] &= \mathbb{E}\left[\phi_k(\boldsymbol{\zeta}) \mid \boldsymbol{\zeta} \in [\check{\zeta}_k, \hat{\zeta}_k]\right] \mathbb{P}\left[\boldsymbol{\zeta} \in [\check{\zeta}_k, \hat{\zeta}_k]\right] \\
&= \int_{\check{\zeta}_k}^{\hat{\zeta}_k} \frac{\phi_k(z)\mathcal{N}\left(z \mid m_k;\ \text{diag}\,(s_k)\right)}{\mathbb{P}\left[\boldsymbol{\zeta} \in [\check{\zeta}_k, \hat{\zeta}_k]\right]} dz \mathbb{P}\left[\boldsymbol{\zeta} \in [\check{\zeta}_k, \hat{\zeta}_k]\right] \\
&= \int_{\check{\zeta}_k}^{\hat{\zeta}_k} \phi_k(z)\mathcal{N}\left(z \mid m_k;\ \text{diag}\,(s_k)\right) dz \\
&= \left(\int_{\check{\zeta}^{(1)}}^{\hat{\zeta}^{(1)}} \phi_k(z)\mathcal{N}\left(z \mid m_k^{(1)};\ s_k^{(1)}\right) dz, \ldots, \int_{\check{\zeta}^{(n_k)}}^{\hat{\zeta}^{(n_k)}} \phi_k(z)\mathcal{N}\left(z \mid m_k^{(n_k)};\ s_k^{(n_k)}\right) dz\right)^T
\end{aligned}$$

where we ignore the dependence of m_k and s_k on z to simplify the notation. For $i \in \{1, \ldots, n_k\}$, the integral can be split in two parts:

$$\int_{\check{\zeta}^{(i)}}^{\hat{\zeta}^{(i)}} \phi_k(z)\mathcal{N}\left(z \mid m_k^{(i)};\ s_k^{(i)}\right) dz = \int_{\check{\zeta}^{(i)}}^{\infty} \phi_k(z)\mathcal{N}\left(z \mid m_k^{(i)};\ s_k^{(i)}\right) dz - \int_{\hat{\zeta}^{(i)}}^{\infty} \phi_k(z)\mathcal{N}\left(z \mid m_k^{(i)};\ s_k^{(i)}\right) dz.$$

For $\phi_k = I$, using the substitution rule for integration, the former relation can be written as

$$\int_{\check{\zeta}^{(i)}}^{\hat{\zeta}^{(i)}} z\mathcal{N}\left(z \mid m_k^{(i)};\ s_k^{(i)}\right) dz = \int_0^{\infty} (z + \check{\zeta}^{(i)})\mathcal{N}\left(z + \check{\zeta}^{(i)} \mid m_k^{(i)};\ s_k^{(i)}\right) dz - \int_0^{\infty} (z + \hat{\zeta}^{(i)})\mathcal{N}\left(z + \hat{\zeta}^{(i)} \mid m_k^{(i)};\ s_k^{(i)}\right) dz,$$

which can be rewritten in terms of expectations and probabilities to obtain the final relation for $\phi_k = I$:

$$\begin{aligned}
\mathbb{E}\left[\boldsymbol{\zeta}^{(i)} \mid \boldsymbol{\zeta}^{(i)} \in [\check{\zeta}^{(i)}, \hat{\zeta}^{(i)}]\right] \mathbb{P}\left[\boldsymbol{\zeta}^{(i)} \in [\check{\zeta}^{(i)}, \hat{\zeta}^{(i)}]\right] = \\
\mathbb{E}_{\boldsymbol{z} \sim \mathcal{N}\left(m_k^{(i)};\, s_k^{(i)}\right)} \left[\text{ReLU}\left(\boldsymbol{z} + \check{\zeta}^{(i)}\right)\right] - \mathbb{E}_{\boldsymbol{z} \sim \mathcal{N}\left(m_k^{(i)};\, s_k^{(i)}\right)} \left[\text{ReLU}\left(\boldsymbol{z} + \hat{\zeta}^{(i)}\right)\right] + \\
\check{\zeta}^{(i)} \mathbb{P}_{\boldsymbol{z} \sim \mathcal{N}\left(m_k^{(i)};\, s_k^{(i)}\right)} \left[\boldsymbol{z} \in [\check{\zeta}^{(i)}, \infty]\right] - \hat{\zeta}^{(i)} \mathbb{P}_{\boldsymbol{z} \sim \mathcal{N}\left(m_k^{(i)};\, s_k^{(i)}\right)} \left[\boldsymbol{z} \in [\hat{\zeta}^{(i)}, \infty]\right].
\end{aligned}$$

The relation for $\phi_k = \text{ReLU}$ follows directly from the result for $\phi_k = I$, since $\forall i \in \{1, \ldots, n_k\}$ it hods that

$$\int_{\check{\zeta}^{(i)}}^{\hat{\zeta}^{(i)}} \text{ReLU}\,(z)\,\mathcal{N}\left(z \mid m_k^{(i)};\ s_k^{(i)}\right) dz = \int_{[\check{\zeta}]_+^{(i)}}^{[\hat{\zeta}]_+^{(i)}} z\mathcal{N}\left(z \mid m_k^{(i)};\ s_k^{(i)}\right) dz.$$

B.4. Proof Proposition 5

Recall from the proof of Proposition 4 that, $\forall i \in \{1, \dots, n_k\}$, we can write the product of the conditional expectation and probability as

$$\mathbb{E}\left[\boldsymbol{\zeta}^{(i)} \mid \boldsymbol{\zeta}^{(i)} \in [\check{\zeta}^{(i)}, \infty)\right] \mathbb{P}\left[\boldsymbol{\zeta}^{(i)} \in [\check{\zeta}^{(i)}, \infty)\right] = \int_{\check{\zeta}^{(i)}}^{\infty} \zeta \mathcal{N}\left(\zeta \mid m_k^{(i)}(z);\ s_k^{(i)}(z)\right) d\zeta,$$

We can apply substitution $q = \frac{\zeta - m_k^{(i)}(z)}{\sqrt{2 s_k^{(i)}(z)}}$ to rewrite the integral as

$$\int_{\check{\zeta}^{(i)}}^{\infty} \zeta \mathcal{N}\left(\zeta \mid m_k^{(i)}(z);\ s_k^{(i)}(z)\right) d\zeta = \frac{1}{\sqrt{\pi}} \int_{\tilde{\zeta}^{(i)}}^{\infty} (q\sqrt{2 s_k^{(i)}(z)} + m_k^{(i)}(z)) \exp\left(-q^2\right) dq,$$

where $\tilde{\zeta}^{(i)}(z) = \frac{\check{\zeta}^{(i)} - m_k^{(i)}(z)}{\sqrt{2 s_k^{(i)}(z)}}$. We then solve the integral to obtain

$$\int_{\check{\zeta}^{(i)}}^{\infty} \zeta \mathcal{N}\left(\zeta \mid m_k^{(i)}(z);\ s_k^{(i)}(z)\right) d\zeta = \frac{m_k^{(i)}(z)}{2}(1 - \text{erf}\left(\tilde{\zeta}^{(i)}(z)\right)) + \sqrt{\frac{s_k^{(i)}(z)}{2\pi}} \exp\left(-(\tilde{\zeta}^{(i)}(z))^2\right). \tag{14}$$

The above result naturally extends to the case of relu-activation functions, which results in

$$\int_{\check{\zeta}^{(i)}}^{\infty} \text{ReLU}\left(\zeta\right) \mathcal{N}\left(\zeta \mid m_k^{(i)}(z);\ s_k^{(i)}(z)\right) d\zeta = \frac{m_k^{(i)}(z)}{2}(1 - \text{erf}\left([\tilde{\zeta}^{(i)}(z)]_+\right)) + \sqrt{\frac{s_k^{(i)}(z)}{2\pi}} \exp\left(-[\tilde{\zeta}^{(i)}(z)]_+^2\right). \tag{15}$$

Since, $\forall x \in \mathbb{R}, 0 \leq \text{erf}\left([x]_+\right) \leq 1$ and $0 \leq \exp\left(-x^2\right) \leq 1$, the closed form solution for the integral can be bounded as follows

$$\frac{1}{2}[m_k^{(i)}(z)]_- \leq \int_{\check{\zeta}^{(i)}}^{\infty} \text{ReLU}\left(\zeta\right) \mathcal{N}\left(\zeta \mid m_k^{(i)}(z);\ s_k^{(i)}(z)\right) d\zeta \leq \frac{1}{2}[m_k^{(i)}(z)]_+ + \sqrt{\frac{s_k^{(i)}(z)}{2\pi}}, \qquad \forall z \in \mathbb{R}^n,$$

where, $\sqrt{s_k^{(i)}(z)}$ is convex w.r.t. z. Hence, the above upper bound can easily be transformed into a piece-wise affine upper bound.

B.5. Proof Proposition 6 and Corollary 6.1

Let us define function $c : \mathbb{R}^{n_{K+1}} \to \mathbb{R}$ to simplify notation. By the law of total expectation it holds that

$$\mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1})\right] = \sum_{j \in \{1, \dots, N\}} \mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1}) \mid \boldsymbol{\zeta}_{K+1} \in Z_j\right] \mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_j\right]$$

where $\boldsymbol{\zeta}_{K+1} = f^{\boldsymbol{w}}(x)$ and $\boldsymbol{w} \sim q(\cdot)$. Here, the conditional expectations can be lower- and upper-bounded by substituting the distribution of random variable $\boldsymbol{\zeta}_{K+1}$ by a dirac-delta function placed at the min- and max value of $h^{(i)}$ over Z_j, respectively, that is,

$$\mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1})\right] \geq \sum_{j \in \{1, \dots, N\}} [\min_{\zeta \in Z_j} c(\zeta)] \mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_j\right]$$

and

$$\mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1})\right] \leq \sum_{j \in \{1, \dots, N\}} [\max_{\zeta \in Z_j} c(\zeta)] \mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_j\right].$$

Hence, if we take $c(\zeta) = \text{softmax}^{(j)}(\zeta) - \text{softmax}^{(i)}(\zeta)$, for which holds that

$$-1 \leq \max_{\zeta \in \mathbb{R}^m} (\text{softmax}^{(j)}(\zeta) - \text{softmax}^{(i)}(\zeta)) \leq 1,$$

we have that

$$\mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1})\right] \geq -\mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_1\right] + \sum_{j\in\{2,\ldots,N\}} [\min_{\zeta\in Z_l} c(\zeta)]\mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_j\right]$$

and

$$\mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1})\right] \leq \mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_1\right] + \sum_{l\in\{2,\ldots,N\}} [\max_{\zeta\in Z_j} c(\zeta)]\mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_j\right] \tag{16}$$

Then, in the special case of $N = 2$, Equation (16) provides a sufficient condition to conclude on adversarial robustness. In particular, we obtain that

$$\max_{\zeta\in Z_1}\left(\frac{\exp\zeta^{(j)} - \exp\zeta^{(i)}}{\sum_{l\in\{1,\ldots,n_{K+1}\}}\exp(\zeta^{(l)})}\right) \leq -\frac{\mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_0\right]}{\mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_1\right]} \implies \mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1})\right] \leq 0.$$

where we used the definition of the softmax functions. We can rewrite the former as follows

$$\max_{\zeta\in Z_1}\left(\frac{\exp\zeta^{(j)} - \exp\zeta^{(i)}}{\sum_{l\in\{1,\ldots,n_{K+1}\}}\exp(\zeta^{(l)})}\right) \leq -\eta \implies \mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1})\right] \leq 0. \tag{17}$$

where $\eta \in \mathbb{R}_{\geq 0}$ is defined as $\eta := \frac{\mathbb{P}[\boldsymbol{\zeta}_{K+1}\in Z_0]}{\mathbb{P}[\boldsymbol{\zeta}_{K+1}\in Z_1]} = \frac{1-\mathbb{P}[\boldsymbol{\zeta}_{K+1}\in Z_1]}{\mathbb{P}[\boldsymbol{\zeta}_{K+1}\in Z_1]}$. Notice that since, $\text{supp}(f^{\boldsymbol{w}}(x)) = \mathbb{R}^{n_k}$ and hence $\text{supp}(\boldsymbol{\zeta}_{K+1}) = \mathbb{R}^{n_k}$, it is guaranteed that $\mathbb{P}\left[\boldsymbol{\zeta}_{K+1} \in Z_1\right] \neq 0$. Since the maximization problem in Eqn. (17) is highly non-convex it can only be solved via exhaustive enumeration as discussed in (Berrada et al., 2021). Hence, to improve computational efficiency, we use that

$$\max_{\zeta\in Z_1}\left(\frac{\exp\zeta^{(j)} - \exp\zeta^{(i)}}{\sum_{l\in\{1,\ldots,n_{K+1}\}}\exp(\zeta^{(l)})}\right) \leq \frac{\max_{\zeta\in Z_1}(\exp\zeta^{(j)} - \exp\zeta^{(i)})}{\max_{\zeta\in Z_1}(\sum_{l\in\{1,\ldots,m\}}\exp(\zeta^{(l)}))},$$

to obtain the following condition

$$\max_{\zeta\in Z_1}(\exp\zeta^{(j)} - \exp\zeta^{(i)}) \leq -\eta\max_{\zeta\in Z_1}\left(\sum_{l\in\{1,\ldots,n_{K+1}\}}\exp(\zeta^{(l)})\right) \implies \mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1})\right] \leq 0,$$

which, in the case that Z_1 is a hyper-rectangle defined by vectors $\check{\zeta}, \hat{\zeta} \in \mathbb{R}^{n_{K+1}}$, reduces to

$$\exp\hat{\zeta}^{(j)} - \exp\check{\zeta}^{(i)} + \eta\sum_{l\in\{1,\ldots,n_{K+1}\}}\exp(\hat{\zeta}^{(l)}) \leq 0 \implies \mathbb{E}\left[c(\boldsymbol{\zeta}_{K+1})\right] \leq 0.$$

C. Proofs Section 6

C.1. Proof Proposition 7

Recall that for $[\check{\zeta}, \hat{\zeta}]$ a hyper-rectangle, according to Lemma 3, computing the probability that $\boldsymbol{\zeta}$ is in Z_k reduces to computing the product of Gaussian CDFs (error functions), that is

$$\mathbb{P}_{\boldsymbol{\zeta}\sim\mathcal{N}(m_k(x);\ \text{diag}(s_k(x)))}\left[\boldsymbol{\zeta} \in [\check{\zeta}_k, \hat{\zeta}_k]\right] = \prod_{i\in\{1,\ldots,n_k\}}\frac{1}{2}\left[\text{erf}\left(\frac{\hat{\zeta}^{(i)} - m_k^{(i)}(x)}{\sqrt{2s_k^{(i)}(x)}}\right) - \text{erf}\left(\frac{\check{\zeta}^{(i)} - m_k^{(i)}(x)}{\sqrt{2s_k^{(i)}(x)}}\right)\right].$$

To ensure that $\mathbb{P}_{\boldsymbol{\zeta}\sim\mathcal{N}(m(x);\ \text{diag}(s(x)))}\left[\boldsymbol{\zeta} \in Z_k\right] = 1 - \epsilon$, we enforce that for each $i \in \{1, \ldots, l\}$, it holds that

$$\frac{1}{2}\left[\text{erf}\left(\frac{\hat{\zeta}^{(i)} - m_k^{(i)}(x)}{\sqrt{2s_k^{(i)}(x)}}\right) - \text{erf}\left(\frac{\check{\zeta}^{(i)} - m_k^{(i)}(x)}{\sqrt{2s_k^{(i)}(x)}}\right)\right] = (1-\epsilon)^{\frac{1}{n_k}}.$$

Next, we choose to place $\hat{\zeta}$ and $\check{\zeta}$ symmetrical around $m(x)$, that is, $\forall i \in \{1, \ldots, n_k\}$, we choose $\hat{\zeta}^{(i)}$ and $\check{\zeta}^{(i)}$ such that

$$\text{erf}\left(\frac{\hat{\zeta}^{(i)} - m_k^{(i)}(x)}{\sqrt{2s_k^{(i)}(x)}}\right) = (1-\epsilon)^{\frac{1}{n_k}}, \quad \text{and} \quad \text{erf}\left(\frac{\check{\zeta}^{(i)} - m_k^{(i)}(x)}{\sqrt{2s_k^{(i)}(x)}}\right) = -(1-\epsilon)^{\frac{1}{n_k}}.$$

Then, by taking the inverse of the error function for both qualities, we obtain the following expressions for $\check{\zeta}$ and $\hat{\zeta}$:

$$\hat{\zeta}^{(i)} = \text{erf}^{-1}\left((1-\epsilon)^{\frac{1}{n}}\right)\sqrt{2s_k^{(i)}(x)} + m_k^{(i)}(x),$$
$$\check{\zeta}^{(i)} = \text{erf}^{-1}\left(-(1-\epsilon)^{\frac{1}{n}}\right)\sqrt{2s_k^{(i)}(x)} + m_k^{(i)}(x).$$

Notice that, these expressions depend via $s(x)$ and $m(x)$ on x which can take values in X. Hence, to ensure that $\forall x \in T$, $\mathbb{P}_{\boldsymbol{\zeta} \sim \mathcal{N}(m(x);\, \text{diag}(s(x)))}\left[\boldsymbol{\zeta} \in Z_k\right] \geq 1 - \epsilon$, it should hold that

$$\hat{\zeta}^{(i)} \geq \min_{x \in X}\left[\text{erf}^{-1}\left((1-\epsilon)^{\frac{1}{n}}\right)\sqrt{2s_k^{(i)}(x)} + m_k^{(i)}(x)\right],$$
$$\check{\zeta}^{(i)} \leq \max_{x \in X}\left[\text{erf}^{-1}\left(-(1-\epsilon)^{\frac{1}{n}}\right)\sqrt{2s_k^{(i)}(x)} + m_k^{(i)}(x)\right].$$

Hence, the optimal choice for $\check{\zeta}^{(i)}$ and $\check{\zeta}^{(i)}$ is such that above constraints hold by equality. Notice that for $\epsilon \in [0, 1]$, both optimization problems reduce to convex minimization problems.

D. Algorithms

The interval relaxation procedure of the softmax layer is summarized in Algorithm 2. The algorithm employs the result of Proposition 6. The back-propagation of a PWA relaxation of the value function is summarized in Algorithm 3, to which we refer as `BP`. For partition $\{Z_{k,j}\}_{j=1}^N$ of supp$(f_{0:k}^{\boldsymbol{w}})$ and a compact subset Z_{k-1} of supp$(f_{0:k-1}^{\boldsymbol{w}})$, `BP` computes the relaxations of Terms 9a in Line 4 and 9b in Line 6 w.r.t. $z_{k-1} \in Z_{k-1}$. It employs Lemma 3 for Term 9a, and Proposition 4 or 5 for Term 9b. Finally, the relaxations of Terms 9a and 9b are combined with PWA relaxation of V_k w.r.t. partition $\{Z_{k,j}\}_{j=1}^N$, denoted as $\{\check{V}_{k,j}, \hat{V}_{k,j}\}_{j=1}^N$, following Eqn. (9) in Line 8-9. Lastly, Algorithm 4 presents BNN-DP for the regression setting.

Algorithm 2 Interval relaxation procedure for Eqn. 5a with $h = \text{softmax}$

1: **function** IBPSoftmax($\{Z_j\}_{j=1}^N$)
2: **for** $j \in \{1, \ldots, N\}$ **do**
3: **if** Z_j is a hyperrectangle **then**
4: Initialize $\check{z}, \hat{z}$ such that $Z_j = [\check{z}^{(0)}, \hat{z}^{(0)}] \times \ldots \times [\check{z}^{(n)}, \hat{z}^{(n)}]$
5: **for** $i \in \{1, \ldots, n\}$ **do**
6: $\check{\beta}_j^{(i)} = \frac{\exp \check{z}^{(i)}}{\exp \check{z}^{(i)} + \sum_{l=1, l\neq i}^N \exp \hat{z}^{(l)}}$,
7: $\hat{\beta}_j^{(i)} = \frac{\exp \hat{z}^{(i)}}{\exp \hat{z}^{(i)} + \sum_{l=1, l\neq j}^N \exp \check{z}^{(l)}}$
8: **end for**
9: **else**
10: $\check{\beta}_j = 0, \hat{\beta}_j = 1$
11: **end if**
12: **end for**
13: **Return:** $\{[\check{\beta}_j, \hat{\beta}_j]\}_{j=1}^N$
14: **end function**

Algorithm 3 Back-Propagation of PWA Relaxations.

function BP($\{[\check{V}_{k,j}, \hat{V}_{k,j}]\}_{j=1}^{N}$, $\{Z_{k,j}\}_{j=1}^{N}$, Z_{k-1})
  for $j \in \{1, \dots, N\}$ **do**
    For $z \in Z_{k,j}$, compute $\check{\mathbb{P}}_j, \hat{\mathbb{P}}_j \in [0, 1]$ s.t.
    $\mathbb{P}_{\boldsymbol{\zeta} \sim \mathcal{N}(m_k(z);\, s_k(z))} [\boldsymbol{\zeta} \in Z_{k,j}] \in [\check{\mathbb{P}}_j, \hat{\mathbb{P}}_j]$,
    compute $\check{\mathbb{E}}_j, \hat{\mathbb{E}}_j : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^l$ s.t.
    $\tilde{\mathbb{E}}_{\boldsymbol{\zeta} \sim \mathcal{N}(m_k(z);\, s_k(z))} [\phi_k(\boldsymbol{\zeta}) \mid \boldsymbol{\zeta} \in Z_{k,j}] \in [\check{\mathbb{E}}_j, \hat{\mathbb{E}}_j]$
  end for
  $[\check{V}_{k-1}, \cdot] = \sum_{j=1}^{N} \check{A}_{k,j} \otimes [\check{\mathbb{E}}_j, \hat{\mathbb{E}}_j] + \check{b}_{k,j} \otimes [\check{\mathbb{P}}_j, \hat{\mathbb{P}}_j]$
  $[\cdot, \hat{V}_{k-1}] = \sum_{j=1}^{N} \hat{A}_{k,j} \otimes [\check{\mathbb{E}}_j, \hat{\mathbb{E}}_j] + \hat{b}_{k,j} \otimes [\check{\mathbb{P}}_j, \hat{\mathbb{P}}_j]$
end function

Algorithm 4 Adversarial Robustness for Regression

function Regression(T, $\{N_k\}_{k=1}^{K+1}$)
  for $k \in \{1, \dots, K-1\}$ **do**
    $Z_{k,main} = [\check{\zeta}_k, \hat{\zeta}_k]$ (Prop 7)
    $\{Z_{k,j}\}_{j=1}^{N_k-1} = \texttt{REFINE}(Z_{k,main})$
    $\mathcal{Z}_k = \{Z_{k,j}\}_{j=1}^{N_k-1} \bigcup Z_{k,main}^{C}$
  end for
  $\check{V}_K(z), \hat{V}_K(z) = m_K(z)$,
  $\mathcal{Z}_K = \{\mathbb{R}^{n_K}\}$
  for $k \in \{K, \dots, 1\}$, **for** $l \in \{1, \dots, N_k\}$ **do**
    $[\check{V}_{k-1,l}, \hat{V}_{k-1,l}] = \texttt{BP}(\{[\check{V}_{k,j}, \hat{V}_{k,j}]\}_{j=1}^{N_k}, \mathcal{Z}_k, \mathcal{Z}_{k-1}^{(l)})$
  end for
  Return: $\min_{x \in T} \check{V}_0(x)$, $\max_{x \in T} \hat{V}_0(x)$
end function