Statistical Guarantees for the Robustness of Bayesian Neural Networks

Luca Cardelli[1] , **Marta Kwiatkowska**[1] , **Luca Laurenti**[1*] , **Nicola Paoletti**[2] ,
Andrea Patane[1*] and **Matthew Wicker**[1*]
[1] University of Oxford
[2]Royal Holloway University of London
{luca.cardelli, marta.kwiatkowska, luca.laurenti, andrea.patane, matthew.wicker}@cs.ox.ac.uk,
nicola.paoletti@rhul.ac.uk

Abstract

We introduce a probabilistic robustness measure for Bayesian Neural Networks (BNNs), defined as the probability that, given a test point, there exists a point within a bounded set such that the BNN prediction differs between the two. Such a measure can be used, for instance, to quantify the probability of the existence of adversarial examples. Building on statistical verification techniques for probabilistic models, we develop a framework that allows us to estimate probabilistic robustness for a BNN with statistical guarantees, i.e., with *a priori* error and confidence bounds. We provide experimental comparison for several approximate BNN inference techniques on image classification tasks associated to MNIST and a two-class subset of the GTSRB dataset. Our results enable quantification of uncertainty of BNN predictions in adversarial settings.

1 Introduction

Bayesian Neural Networks (BNNs), i.e. neural networks with distributions over their weights, are gaining momentum for their ability to capture the uncertainty within the learning model, while retaining the main advantages intrinsic to deep neural networks [MacKay, 1992; Gal, 2016]. A wide array of attacks and formal verification techniques have been developed for deterministic (i.e. non-Bayesian) neural networks [Biggio and Roli, 2018]. However, to date, only methods based on pointwise uncertainty computation have been proposed for BNNs [Feinman *et al.*, 2017]. To the best of our knowledge, there are no methods directed at providing guarantees on BNNs that fully take into account their probabilistic nature. This is particularly important in safety-critical applications, where uncertainty estimates can be propagated through the decision pipeline to enable safe decision making [McAllister *et al.*, 2017].

In this work, we present a statistical framework to evaluate the *probabilistic robustness* of a BNN. The method comes with statistical guarantees, i.e., the estimated robustness meets *a priori* error and confidence bounds. In particular, given an input point $x^* \in \mathbb{R}^m$ and a (potentially uncountable) bounded set of input points $T \subset \mathbb{R}^m$, we aim to compute the probability (induced by the distribution over the BNN weights) that there exists $x \in T$ such that the BNN prediction on x differs from that of x^*. Note that this is a probabilistic generalisation of the usual statement of (deterministic) robustness to adversarial examples [Goodfellow *et al.*, 2014].

We formulate two variants of probabilistic robustness. The first variant describes the probability that the deviation of the network's output (i.e., of the class likelihoods) between x^* and any point in T is bounded. This variant accounts for the so-called *model uncertainty* of the BNN, i.e., the uncertainty that derives from partial knowledge about model parameters. The second variant quantifies the probability that the predicted class label for x^* is invariant for all points in T. This accounts for both model uncertainty and *data uncertainty*, which is related to intrinsic uncertainty in the labels. These properties allow one to estimate, for instance, the probability of the existence of adversarial examples.

The exact computation of such robustness probabilities is, unfortunately, infeasible, as the posterior distribution of a BNN is analytically intractable in general. Hence, we develop a statistical approach, based on the observation that each sample taken from the (possibly approximate) posterior weight distribution of the BNN induces a deterministic neural network. The latter can thus be analysed using existing verification techniques for deterministic networks (e.g. [Huang *et al.*, 2017; Katz *et al.*, 2017; Ruan *et al.*, 2018]). Thus, we can see the robustness of a BNN as a Bernoulli random variable whose mean is the probability that we seek to estimate (see Section 5). In order to do so, we develop a sequential scheme based on Jegourel *et al.* [2018], a statistical approach for the formal verification of stochastic systems. Namely, we iteratively sample the BNN posterior and check the robustness of the resulting deterministic network with respect to the input subset T. After each iteration, we apply the Massart bounds [Massart, 1990] to check if the current sample set satisfies the *a priori* statistical guarantees. Thus, we reduce the number of samples only to those needed in order to meet the statistical guarantees required. This is essential for the computational feasibility of the method, as each sample entails solving a computationally expensive verification sub-problem. Moreover, our method is generally applicable in that the estimation scheme is independent of the choice of the deterministic verification technique.

We evaluate our method on fully connected and convolutional neural networks, trained on the MNIST handwritten digits dataset [LeCun and Cortes, 2010] and a two-class subset of the the German Traffic Sign Recognition Benchmark (GTSRB) [Stallkamp *et al.*, 2012] respectively. We compare the robustness profiles of three different BNN inference methods (Monte Carlo

[*]equal contribution

dropout [Gal and Ghahramani, 2016], variational inference [Blundell *et al.*, 2015], and Hamiltonian Monte Carlo [Neal, 2012]), showing that our notion of probabilistic robustness results in an effective model selection criterion and provides insights into the benefits of BNN stochasticity in mitigating attacks[1].

In summary, the paper makes the following main contributions:

- We define two variants of probabilistic robustness for BNNs, which generalise safety and reachability defined for deterministic networks. These can be used to quantify robustness against adversarial examples.
- Building on analysis techniques for deterministic neural networks, we design a statistical framework for the estimation of the probabilistic robustness of a BNN, which ensures *a priori* statistical guarantees.
- We evaluate our methods on state-of-the-art approximate inference approaches for BNNs on MNIST and GTSRB, for a range of properties. We quantify the uncertainty of BNN predictions in adversarial settings.

2 Related Work

Most existing methods for the analysis and verification of neural networks are designed for deterministic models. These can be roughly divided into heuristic search techniques and formal verification techniques. While the focus of the former is usually on finding an adversarial example [Goodfellow *et al.*, 2014; Wicker *et al.*, 2018; Wu *et al.*, 2018], verification techniques strive to formally prove guarantees about the robustness of the network with respect to input perturbations [Huang *et al.*, 2017; Katz *et al.*, 2017; Ruan *et al.*, 2018]. Alternatively, statistical techniques posit a specific distribution in the input space in order to derive a quantitative measure of robustness for deterministic networks [Webb *et al.*, 2018; Cohen *et al.*, 2019]. However, this approach may not be appropriate for safety-critical applications, because these typically require a worst-case analysis and adversarial examples often occupy a negligibly small portion of the input space. Dvijotham *et al.* [2018] consider a similar problem, i.e., that of verifying (deterministic) deep learning models over probabilistic inputs. Even though they provide stronger probability bounds than the above statistical approaches, their method is not applicable to BNNs.

Bayesian uncertainty estimation approaches have been investigated as a way to flag possible adversarial examples on deterministic neural networks [Feinman *et al.*, 2017], though recent results suggest that such strategies might be fooled by adversarial attacks designed to generate examples with small uncertainty [Grosse *et al.*, 2018]. However, as these methods build adversarial examples on deterministic network and use uncertainty only at prediction time, their results do not capture the actual probabilistic behaviour of the BNN in adversarial settings. In contrast, our approach allows for the quantitative analysis of probabilistic robustness of BNNs, yielding probabilistic guarantees for the absence of adversarial examples.

A Bayesian perspective to adversarial attacks is taken by Rawat *et al.* [2017], where experimental evaluation of the relationship between model uncertainty and adversarial examples is given. Similarly, Kendall *et al.* [2015] study the correlation between uncertainty and per-class prediction accuracy in a semantic segmentation problem. These approaches are, however, pointwise, in that the uncertainty information is estimated for one input at a time. Instead, by applying formal verification techniques on the deterministic NNs sampled from the BNN, our method supports worst-case scenario analysis on possibly uncountable regions of the input space. This allows us to obtain statistical guarantees on probabilistic robustness in the form of *a priori* error and confidence bounds.

Cardelli *et al.* [2018] present a method for computing probabilistic guarantees for Gaussian processes in a Bayesian inference settings, which applies to fully connected BNNs in the limit of infinite width. However, the method, while exact for Gaussian processes, is only approximate for BNNs and with an error that cannot be computed.

3 Bayesian Neural Networks

In this section we provide background for learning with BNNs, and briefly review the approximate inference methods employed in the remainder of the paper. We use $f^{\mathbf{W}}(x) = [f_1^{\mathbf{W}}(x),...,f_C^{\mathbf{W}}(x)]$ to denote a BNN with C output units and an unspecified number (and kind) of hidden layers, where $\mathbf{W}$ is the weight vector random variable. Given a distribution over $\mathbf{W}$ and $w \in \mathbb{R}^W$, a weight vector sampled from the distribution of $\mathbf{W}$, we denote with $f^w(x)$ the corresponding deterministic neural network with weights fixed to w. Let $\mathcal{D} = \{(x,c)|x \in \mathbb{R}^m, c \in \{c_1,...,c_C\}\}$ be the training set. We consider classification with a softmax likelihood model, that is, assuming that the likelihood function for observing class c_h, for an input $x \in \mathbb{R}^m$ and a given $w \in \mathbb{R}^W$, is given by $\sigma_h(f^w(x)) = \frac{e^{f_h^w(x)}}{\sum_{j=1}^{C} e^{f_j^w(x)}}$. We define $\sigma(f^w(x)) = [\sigma_1(f^w(x)),...,\sigma_C(f^w(x))]$, the combined vector of class likelihoods, and similarly we denote with $\sigma(f^{\mathbf{W}}(x))$ the associated random variable induced by the distribution over $\mathbf{W}$. In Bayesian settings, we assume a prior distribution over the weights, i.e. $\mathbf{W} \sim p(w)$[2], so that learning for the BNN amounts to computing the posterior distribution over the weights, $p(w|\mathcal{D})$, via the application of the Bayes rule. Unfortunately, because of the non-linearity generally introduced by the neural network architecture, the computation of the posterior cannot be done analytically [MacKay, 1992]. Hence, various approximation methods have been investigated to perform inference with BNNs in practice. Among these methods, in this work we consider Hamiltonian Monte Carlo [Neal, 2012], variational inference through Bayes by Backprop [Blundell *et al.*, 2015], and Monte Carlo Dropout [Gal, 2016]. We stress, however, that the method we present is independent of the specific inference technique used, as long as this provides a practical way of sampling weights w from the posterior distribution of $\mathbf{W}$ (or an approximation thereof).

Hamiltonian Monte Carlo (HMC) proceeds by defining a Markov chain whose invariant distribution is $p(w|\mathcal{D})$, and relies on Hamiltonian dynamics to speed up the exploration of the space. Differently from the two other methods discussed below, HMC does not make any assumptions on the form of the posterior distribution, and is asymptotically correct. The result of HMC is a set of samples w_i that approximates $p(w|\mathcal{D})$.

Variational Inference (VI) proceeds by finding a suitable approximating distribution $q(w) \approx p(w|\mathcal{D})$ in a trade-off between approximation accuracy and scalability. The core idea is that

[1] Code is available at *https://github.com/matthewwicker/StatisticalGuarenteesForBNNs*

[2] Usually depending on hyperparameters, omitted here for simplicity.

$q(w)$ depends on some hyper-parameters that are then iteratively optimized by minimizing a divergence measure between $q(w)$ and $p(w|D)$. Samples can then be efficiently extracted from $q(w)$.

Monte Carlo Dropout (MCD) is an approximate variational inference method based on dropout [Gal and Ghahramani, 2016]. The approximating distribution $q(w)$ takes the form of the product between Bernoulli random variables and the corresponding weights. Hence, sampling from $q(w)$ reduces to sampling Bernoulli variables, and is thus very efficient.

4 Problem Formulation

A BNN defines a stochastic process whose randomness comes from the distribution over the weights of the neural network. Thus, the probabilistic nature of a BNN should be taken into account when studying its robustness.

For this purpose, we formulate two problems, instances of probabilistic reachability and probabilistic safety, properties that are widely employed for the analysis of stochastic processes [Abate et al., 2008; Bortolussi et al., 2016]. At the same time, these problems constitute a probabilistic generalization of the reachability [Ruan et al., 2018] and safety specifications [Huang et al., 2017] typical of deterministic neural networks. In particular, in Problem 1 we consider reachability of the value of the softmax regression, while Problem 2 is concerned with perturbations that affect the classification outcome.

Problem 1 Consider a neural network f^w with training dataset D. Let x be a test point and $T \subseteq \mathbb{R}^m$ a bounded set. For a given $\delta \geq 0$, compute the probability

$$p_1 = P(\omega_1(f^w)|D); \text{ where}$$
$$\omega_1(f^w) = \{\forall x' \in T \text{ s.t. } |f^w(x') - f^w(x)|_p > \delta\}$$

and $|\cdot|_p$ is a given seminorm. For $0 \leq \epsilon < 1$, we say that f^w is robust with probability at least $1-\epsilon$ in x with respect to set T and perturbation δ iff $p_1 \leq \epsilon$.

For a set T and a test point x, Problem 1 seeks to compute the probability that there exists $x' \in T$ such that the output of the softmax layer for x' deviates more than a given threshold δ from the output for x. Note that x' is not necessarily an element of T. If T is a bounded region around x, Problem 1 corresponds to computing the robustness of f^w with respect to local perturbations. Note that we only require that T is bounded, and so T could also be defined, for instance, as a set of vectors derived from a given attack.

The probability value in Problem 1 is relative to the output of the softmax layer, i.e., to the vector of class likelihoods, and not to the classification outcome, which is instead considered in Problem 2. In fact, probabilistic models for classification further account for the uncertainty in the class prediction step by placing a Multinoulli distribution on top of the softmax output [Gal, 2016]. Specifically, the class of an input x is assigned by the stochastic process $m(x)$ with values in $\{1, \ldots, C\}$, where the probability that $m(x) = h$, $h \in \{1, \ldots, C\}$, is given by

$$P(m(x) = h) = \int z P(\sigma_h(f^w(x)) = z|D)\, dz.$$

Taking the classification aspect and Multinoulli distribution into account poses the following problem.

Problem 2 Consider a neural network f^w with training dataset D. Let x be a test point and $T \subseteq \mathbb{R}^m$ a bounded set. We compute the probability

$$p_2 = P(\omega_2(f^w)|D); \text{ where}$$
$$\omega_2(f^w) = \{\exists x' \in T \text{ s.t. } m(x') \neq m(x)\}.$$

For $0 \leq \epsilon < 1$, we say that f^w is safe with probability at least $1-\epsilon$ in x with respect to set T iff $p_2 \leq \epsilon$.

An important consequence of Problem 2 is that, for regions of the input space where the model is unsure which class to assign (i.e., where all classes have similar likelihoods), it is likely that the Multinoulli samples of Eqn. (4) induce a classification different from that of x, thus leading to a low probability of being safe. In contrast, Problem 1 does not capture this aspect as it considers relative variations of the class likelihoods.

Note that the only source of uncertainty contributing to the stochasticity of Problem 1 comes from the distribution of the weights of the BNN, i.e. from $p(w|D)$. This is the so-called model uncertainty, i.e. the uncertainty that accounts for our partial knowledge about the model parameters [Gal, 2016]. On the other hand, Problem 2 accounts both for model uncertainty and data uncertainty i.e. the noise of the modelled process. We stress that both robustness measures introduced in Problem 1 and 2 do not consider any specific decision making procedure, and are as such independent and prior to the particular decision making techniques placed on top of the Bayesian model.

Unfortunately, for BNNs, the distribution of f^w is intractable. Hence, to solve Problem 1 and 2 approximation techniques are required. In what follows, we illustrate a statistically sound method for this purpose.

5 Estimation of BNN Robustness Probability

We present a solution method to Problems 1 and 2, which builds on a sequential scheme to estimate the probability of properties ω_1 and ω_2, (roblems 1 and 2, respectively. This solution comes with statistical guarantees, in that it ensures arbitrarily small estimation error with arbitrarily large confidence.

Our method is based on the observation that a sample of the BNN weights induces a deterministic NN. Hence, we can decide the satisfaction of ω_1 and ω_2 for each sample using existing formal verification techniques for deterministic networks. More precisely, given a BNN f^w, we verify ω_1 and ω_2 over deterministic NNs f^w, where w is a weight vector sampled from $p(w|D)$ (or its approximation $q(w)$). For ω_2, along with w, we also need to sample the class $m(x)$ from the Multinoulli distribution of Eqn. (4). Details about the deterministic verification methods used here are given in Section 5.1.

Therefore, for $j = 1, 2$, we can see the satisfaction of $\omega_j(f^w)$ as a Bernoulli random variable Z_j, which we can effectively sample as described above, i.e., by sampling the BNN weights and formally verifying the resulting deterministic network. Then, solving Problem j amounts to computing the expected value of Z_j. That is, we evaluate the probability that ω_j is true w.r.t. f^w. For this purpose, we derive an estimator $\hat{p}_j$ for p_j such that:

$$\hat{p}_j = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\omega_j}(f^{w_i}) \quad E[Z_j] = p_j; \tag{1}$$

where $\{f^{w_i}\}_{i=1,\dots,n}$ is the collection of sampled deterministic networks.

We want $\hat{p}_j$ to satisfy a priori statistical guarantees. Namely, for arbitrary absolute error bound $0 < \varepsilon < 1$ and confidence $0 < \delta < 1$ (i.e., the probability of producing a false estimate), the following must hold:

$$P(|\hat{p}_j - p_j| > \varepsilon) \leq \delta \quad (2)$$

Chernoff bounds [Chernoff, 1952] are a popular technique to determine the sample size n required to satisfy (2) for a given choice of ε and δ. Specifically, the estimate $\hat{p}_j$ after n samples satisfies (2) if

$$n > \frac{1}{2\varepsilon^2} \log \frac{2}{\delta} \quad (3)$$

These bounds are, however, often overly conservative, leading to unnecessarily large sample size. Tighter bounds were formulated by Massart [1990], where the sample size depends on the unknown probability to estimate p_j. In particular, Massart bounds require only a small fraction of samples when p_j is close to 0 or 1, but are not directly applicable for their dependence on the unknown p_j value. Jegourel et al. [2018] solve this issue by extending Massart bounds to work with confidence intervals for p_j instead of p_j itself. For arbitrary $0 < \gamma < \varepsilon$, let $I_{p_j} = [a, b]$ denote the $1-\delta$ confidence interval for p_j obtained after n samples. Then, (2) holds if n satisfies

$$n > \frac{2}{9\gamma^2} \log \frac{2}{\delta} \cdot \begin{cases} (3b + \varepsilon)(3(1-b) - \varepsilon) & \text{if } b < 1/2 \\ (3(1-a) + \varepsilon)(3a + \varepsilon) & \text{if } a > 1/2 \\ (3/2 + \varepsilon)^2 & \text{otherwise} \end{cases} \quad (4)$$

We employ a sequential probability estimation scheme to solve Problem 1 and 2, which utilizes the above bounds to determine, after each sample, if the current estimate provides the required guarantees (given by parameters ε, δ and γ). By applying a sequential scheme, we crucially avoid unnecessary sampling because the analysis terminates as soon as the statistical guarantees are met. This considerably improves the efficiency of our method given that drawing of each Bernoulli sample entails solving a potentially computationally expensive NN verification problem.

The estimation scheme is outlined in Algorithm 1 and works as follows. At the n-th iteration, we sample a weight vector w from the posterior and, only for Problem 2, the class of the test input $m(x)$ (lines 4-5). The n-th Bernoulli sample (variable SAT in line 6) is obtained by applying a suitable deterministic verification method (see Section 5.1) on f^w. After updating the number of successes k and trials n (line 7), we use these to update the estimator $\hat{p}$ (line 8) and compute a $1-\delta$ confidence interval I_p for the robustness probability (line 9). We use I_p to derive the sample size n^M as per the Massart bounds of Equation (4) (line 10), and update the number of required samples to $n_{max} = \lceil \min(n^M, n^C) \rceil$ (line 11), where n^C is the sample size computed as per (3) in line 1 of the algorithm. In other words, we select the best between Chernoff and Massart bounds, as Massart bounds are tighter than Chernoff bounds when $\hat{p}$ is close to 0 or 1, but Chernoff bounds perform better when $\hat{p}$ is close to 0.5. If $n \geq n_{max}$, we return $\hat{p}$, which is guaranteed to satisfy (2). Otherwise, we iterate over an additional sample. Concrete values of the Chernoff and Massart bounds for our experiments are reported in Appendix A.2.

Algorithm 1 BNN robustness estimation

Input: x – test point, T – search space for adversarial inputs, f – network architecture, $p(w|D)$ – posterior on weights, – property (φ_1 or φ_2), $\varepsilon, \gamma, \delta$ – Massart bound parameters
Output: $\hat{p}$ – robustness probability estimator satisfying (2)

1: $n^C \leftarrow$ number of samples by Chernoff bounds (3)
2: $n_{max} \leftarrow \lceil n^C \rceil$; $n \leftarrow 0$; $k \leftarrow 0$, 0
3: **while** $n < n_{max}$ **do**
4: $\quad w \leftarrow$ sample from $p(w|D)$
5: $\quad m(x) \leftarrow$ sample class from (4) (only for Problem 2)
6: $\quad$ SAT $\leftarrow$ verify φ over f^w as per Sec. 5.1
7: $\quad k \leftarrow k +$ SAT; $n \leftarrow n+1$
8: $\quad \hat{p} \leftarrow k/n$
9: $\quad I_p \leftarrow$ CONFIDENCEINTERVAL$(1-\delta, k, n)$
10: $\quad n^M \leftarrow$ samples num. by Massart bounds (4) using I_p
11: $\quad n_{max} \leftarrow \lceil \min(n^M, n^C) \rceil$
12: **end while**
13: **return** $\hat{p}$

5.1 Verification of Deterministic NNs

Our estimation algorithm is independent of the choice of T and the deterministic verification method used. Below we describe two configurations that are relevant for robustness analysis of real-world BNNs.

Robustness to Bounded Perturbations. In this configuration, T is defined as a ball around the input test point x. We check whether there exists $x' \in T$ such that φ_j holds for the deterministic NN, f^w. The verification is parametrised by the radius of T. We apply the reachability method of Ruan et al. [2018] that computes a safe enclosure of the NN output over T, along with two points in T that respectively minimize and maximize the output of f^w over T. Note that the worst-case input over all points in T is one of these two extremum points: for Problem 1, it is the point with the largest likelihood discrepancy from x; for Problem 2, it is the point that minimizes the likelihood of the nominal class $m(x)$. Thus, we can proceed to verify φ_j by simply checking the property for the corresponding worst-case input.

Robustness to adversarial attacks. We seek to assess the vulnerability of the network against known attack vectors. We use the white-box methods by Goodfellow et al. [2014] and Madry et al. [2017]. These work by building the network gradient and transversing the input space toward regions of reduced classification confidence. Both are parameterised by an attack strength parameter, used to scale the gradient magnitude. Note that the attack is applied to each realisation of the BNN, and as such each attack vector is optimized specifically for f^w, for each $w \sim p(w|D)$.

6 Results

We evaluate our method on different BNN architectures trained with different probabilistic inference techniques (HMC, VI, MCD – see Section 3). In Section 6.2, we analyse robustness to bounded perturbations for a two class subset of the MNIST dataset. In Section 6.3, we report results for adversarial attacks on the full MNIST dataset and a subset of GTSRB.

Figure 1: On the left are two images from the MNIST dataset with the features to be tested outlined in red. The three central columns contain the heatmaps showing the robustness probabilities (as per Problem 1) for different values of (x axis) and (y axis) for BNNs trained with HMC, VI and MCD. On the right are 3D surface plots of the heatmaps with the position of the surfaces projected on the axes so that the heatmaps are easily comparable.

6.1 Experimental Settings

We focus our experiments on BNNs with ReLU activation functions and independent Gaussian priors over the weights.

In Section 6.2 we train a fully conneted network (FCN) with 512 hidden nodes on a two-class subset of MNIST (classes one and seven) and then in Section 6.3 we use the entire dataset. Also in Section 6.3, we analyse a two-layer convolutional BNN on a two-class subproblem of GTSRB (examples of the two classes can be seen in the left column of Figure 2). Namely, the convolutional layer contains 25 filters (kernel size: 3 by 3) followed by a fully-connected layer of 256 hidden nodes. Overall, this BNN is characterised by over four million trainable weights. Unfortunately, applying HMC to larger networks is challenging [Neal, 2012].

For the statistical estimation of robustness probabilities, we used the following Massart bounds parameters $= 0.075$, $= 0.075$ and $= 0.05$. Details on training procedures and hyperparameters are included in the Supplementary Materials.

6.2 Robustness to Bounded Perturbations

Figure 1 depicts the results obtained for Problem 1 on two input images randomly selected from the subset of the MNIST dataset, when relying on Ruan *et al.* [2018] for the deterministic verification sub-routine. The input region T is defined as a hyper-rectangle with edge length around the reference test image x. In view of scalability limitations of the underlying deterministic method, we restrict the evaluation to the feature highlighted in red in the left column of Figure 1. We investigate how the robustness probability is affected by variations of and .

First of all, note that the estimated robustness decreases as increases and/or as decreases, as these respectively imply larger regions T and/or tighter constraints on the BNN output values. Interestingly, even in this simple network, the robustness profile strongly depends on the approximation method used for computation of the BNN posterior. In fact, for HMC and VI (respectively second and third column of the figure) we observe smooth changes in the robustness probability w.r.t. and , where these changes are quantitatively more prominent for HMC than for VI. As with HMC no assumption is made on the form of the posterior distribution, this quantitative robustness difference might suggest that the normality assumption made by VI during training is not sufficient in adversarial settings – i.e. as the model is pushed toward corner-case scenarios. In turn, this could make the BNN vulnerable to low-variance adversarial examples [Grosse *et al.*, 2018]. On the other hand, MCD (fourth column of Figure 1) is characterised by an almost deterministic behaviour with respect to Problem 1, with estimated robustness probabilities sharply moving from 1 to 0. This is especially visible when compared with the other two inference methods (fifth column of Figure 1). As the accuracy scores obtained by the three methods are similar, our results seem to suggest that the BNNs trained by MCD behave almost deterministically with respect to probabilistic robustness. Again, this may be due to the fact that, in adversarial settings, the MCD approximation could lead to an underestimation of model uncertainty. Underestimation of the uncertainty for MCD has also been observed in non-adversarial setting by Myshkov *et al.* [2016].

6.3 Robustness to Adversarial Attacks

We analyse the resilience of Bayesian CNNs against adversarial attacks. As robustness of convolutional neural networks is generally defined in terms of misclassification, we provide results for Problem 2.

MNIST

In Figure 2 (central column) we inspect how the BNN behaves under gradient-based attacks with respect to varying attack strength parameterized by. The empirical distribution of robustness values shown in the violin plots was estimated by performing statistical verification on 50 images randomly selected from the MNIST test dataset (the empirical average and standard deviation are respectively depicted by a dot and a line centred around it). Results serve to butress the observations made in the previous section.

Figure 2: On the left we show a two samples from the MNIST training data set. In the center, each violin plot comes from the estimated robustness of 50 different input samples from the network (the same for each violin). On the right, we plot robustness to inform model selection and can observe that MCD robustness peaks are centred at 0 and 1, whereas VI and HMC are more centred.

Figure 3: In the same spirit as Figure 2, we explore the effect of attack strength on the probabilistic robustness of BNNs trained on GTSRB. On the left we show examples from the two classes tested. Unsurprisingly, convolutional neural networks are less robust to attacks, as previously observed in Carlini and Wagner, 2016. On the right, we plot the accuracy of samples from network posteriors. This information may be used to reason about model selection.

Again, we see that robustness values for VI and, especially, MCD are more stretched toward 0-1 values, compared to those obtained for HMC. Interestingly, for strong attacks (i.e. high values of), this leads to a relatively higher robustness for MCMC, while small strength attacks consistently fail for MCD. Thus, it appears that MCD could be a valuable alternative to MCMC for relatively weak attacks, but may quickly lose its advantage for strong attacks. Notice that HMC is the only method consistently showing high probability density around the mean value, suggesting that the uncertainty estimation obtained from the posterior could be used in this cases to flag potential adversarial inputs.

Figure 2 right column shows how knowledge of network robustness can be used to select models according to the desired levels of prediction robustness and accuracy. For example, if one is not concerned with corner-case scenarios, then standard accuracy maximization would have us choose MCD; however, in a case where we are making risk-sensitive decisions, then we might prefer a model that captures a more complete approximation of uncertainty under an adversarial framework - hence trading accuracy for robustness.

Finally, notice that the computational time required for the statistical verification of an image averages 120 seconds in the experiments here provided.

GTSRB

For GTSRB, it is unsurprising that the CNN models are much less robust than the FCNs used for MNIST. It has been shown previously that for networks trained on ImageNet, the needed to reduce the test set accuracy to 0% was 0.03 [Carlini and Wagner, 2016]. Despite this, it appears that the architecture trained with HMC maintains some robustness to such perturbations. While it displays the best probabilistic safety across all tests, it has less accuracy in some cases than MCD and VI. This clearly brings up the same trade-offs that were discussed for MNIST.

7 Conclusion

We introduced probabilistic robustness for BNNs that takes into account both model and data uncertainty, and can be used to capture, among other properties, the probability of the existence of adversarial examples. We developed a sequential scheme to estimate such a probability with *a priori* statistical guarantees on the estimation error and confidence and evaluated it on fully connected and convolutional networks. Our methods allows one to quantify the tradeoff between accuracy and robustness for different inference procedures for BNNs. We believe our work represents an important step towards the application of neural networks in safety-critical applications.

Acknowledgements

This work has been partially supported by a Royal Society Professorship, by the EU's Horizon 2020 program under the Marie Skłodowska-Curie grant No 722022 and by the EPSRC Programme Grant on Mobile Autonomy (EP/M019918/1).

References

[Abate *et al.*, 2008] Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

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

[Blundell *et al.*, 2015] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[Bortolussi *et al.*, 2016] Luca Bortolussi, Luca Cardelli, Marta Kwiatkowska, and Luca Laurenti. Approximation of probabilistic reachability for chemical reaction networks using the linear noise approximation. In *QEST*, pages 72–88, 2016.

[Cardelli *et al.*, 2018] Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, and Andrea Patane. Robustness guarantees for Bayesian inference with Gaussian processes. *arXiv preprint arXiv:1809.06452*, 2018.

[Carlini and Wagner, 2016] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv e-prints*, page arXiv:1608.04644, Aug 2016.

[Chernoff, 1952] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[Cohen *et al.*, 2019] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[Dvijotham *et al.*, 2018] Krishnamurthy Dvijotham, Marta Garnelo, Alhussein Fawzi, and Pushmeet Kohli. Verification of deep probabilistic models. *arXiv:1812.02795*, 2018.

[Feinman *et al.*, 2017] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv:1703.00410*, 2017.

[Gal and Ghahramani, 2016] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059, 2016.

[Gal, 2016] Yarin Gal. Uncertainty in deep learning. 2016.

[Goodfellow *et al.*, 2014] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[Grosse *et al.*, 2018] Kathrin Grosse, David Pfaff, Michael T Smith, and Michael Backes. The limitations of model uncertainty in adversarial settings. *arXiv:1812.02606*, 2018.

[Huang *et al.*, 2017] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *CAV*, pages 3–29. Springer, 2017.

[Jegourel *et al.*, 2018] Cyrille Jegourel, Jun Sun, and Jin Song Dong. On the sequential Massart algorithm for statistical model checking. In *ISoLA*, pages 287–304. Springer, 2018.

[Katz *et al.*, 2017] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *CAV*, pages 97–117. Springer, 2017.

[Kendall *et al.*, 2015] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[LeCun and Cortes, 2010] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

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

[Madry *et al.*, 2017] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv e-prints*, June 2017.

[Massart, 1990] Pascal Massart. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *The annals of Probability*, pages 1269–1283, 1990.

[McAllister *et al.*, 2017] Rowan McAllister, Yarin Gal, Alex Kendall, Mark Van Der Wilk, Amar Shah, Roberto Cipolla, and Adrian Vivian Weller. Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc., 2017.

[Myshkov and Julier, 2016] Pavel Myshkov and Simon Julier. Posterior distribution analysis for Bayesian inference in neural networks. In *Workshop on Bayesian Deep Learning*, 2016.

[Neal, 2012] Radford M. Neal. MCMC using Hamiltonian dynamics. *arXiv e-prints*, page arXiv:1206.1901, June 2012.

[Rawat *et al.*, 2017] Ambrish Rawat, Martin Wistuba, and Maria-Irina Nicolae. Adversarial phenomenon in the eyes of Bayesian deep learning. *arXiv preprint arXiv:1711.08244*, 2017.

[Ruan *et al.*, 2018] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. In *IJCAI*, pages 2651–2659. AAAI Press, 2018.

[Stallkamp *et al.*, 2012] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[Tran *et al.*, 2016] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

[Webb *et al.*, 2018] Stefan Webb, Tom Rainforth, Yee Whye Teh, and M Pawan Kumar. Statistical verification of neural networks. *arXiv preprint arXiv:1811.07209*, 2018.

[Wicker *et al.*, 2018] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. Feature-guided black-box safety testing of deep neural networks. In *TACAS*, pages 408–426. Springer, 2018.

[Wu *et al.*, 2018] Min Wu, Matthew Wicker, Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. A game-based approximate verification of deep neural networks with provable guarantees. *CoRR*, abs/1807.03571, 2018.

A Supplementary Material

In what follows we report the supplementary material of the paper. We first give specific details on the parameter settings for the inference procedures used for BNN posterior approximation. We then provide additional results using heuristic adversarial attacks.

A.1 Experimental Settings

We report details on the training procedure for the three inference methods analysed in the main text.

HMC. We utilised the implementation of HMC provided in the Edward Python package [Tran *et al.*, 2016]. We used an update step size of 0.01 and the numerical integrator was given 5 update steps per sample.

The Gaussian priors on the convolutional filters were set to have mean 1 and variance 0.01 and the Gaussian priors of the fully connected layer were centred at 0 with variance 1.

The fully connected network for MNIST obtained roughly 87% accuracy on the test set. The CNN trained on the GTSRB dataset comprises about 4.3 million parameters, and reaches around 92% accuracy on the test set.

VI. For our implementation of this method we again relied on the Edward Python package [Tran *et al.*, 2016], using minimisation of the KL divergence. For the network on GTSRB we train using a batch size of 128, and we use the Adam optimizer with a 0.001 learning rate over 15000 iterations. These parameters are identical to the training parameters used for MNIST with the exception of training for 5000 iterations with a higher learning rate of 0.01. The priors were set accordingly to the ones used for HMC.

MCD. In order to choose a dropout value, we grid search parameter of the Bernoulli that governs the dropout method and select the value that gives highest test set accuracy. This resulted in the 0:5 drop-out rate in the BNN used for MNIST, 0:25 and 0:5 respectively for the two layers that make up the CNN used for GTSRB.

A.2 Number of Posterior Samples for Robustness Probability Estimation

Figure 4 shows the number of BNN posterior samples required such that the estimated robustness probability satisfies the following statistical guarantees: error bound = 0:075 and confidence = 0:075. These are the parameters used in all our experiments.

Recall from Section 5 that in our estimation scheme the number of samples n depends on the true probability value p, and that we employ a more conservative version of the Massart bounds that use the (1) confidence interval for p, $I_p = [a;b]$, instead of p itself, which is obviously unknown. The x axis of the plot indeed does not represent the true value of p, but it represents one of the two extrema of I_p (a when $a > 0:5$, b when $b < 0:5$, see Equation 4). Indeed we observe that, for probabilities approximately between 0.23 and 0.79, the probability-independent Chernoff bounds are tighter than the (conservative) Massart bounds. Nevertheless, for probability values close to 0 or 1, the Massart bounds are considerably less conservative than the Chernoff bounds, which require 292 samples. For instance, with the above parameters and = 0:05, when $b <= 0:1$, we need at most 171 samples; when $a >= 0:9$, at most 181 samples.

A.3 Quantitative Difference between Problem 1 & 2

In Figure 5, we extend the tests on robustness to bounded perturbations that appear in Figure 1. It is clear to see that solving Problem 2 is, in some sense, similar to solving Problem 1 with an implicit value of , where that value is approximately the value needed to yield a misclassification.

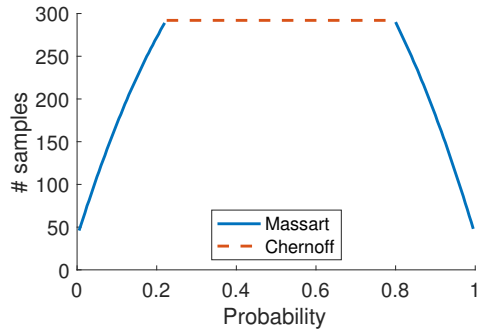

Figure 4: Number of posterior samples (y axis) required in our estimation scheme as a function of the robustness probability. We consider the best of Massart (blue) and Chernoff (orange) bounds. Parameters are = 0:075, = 0:075 and = 0:05.

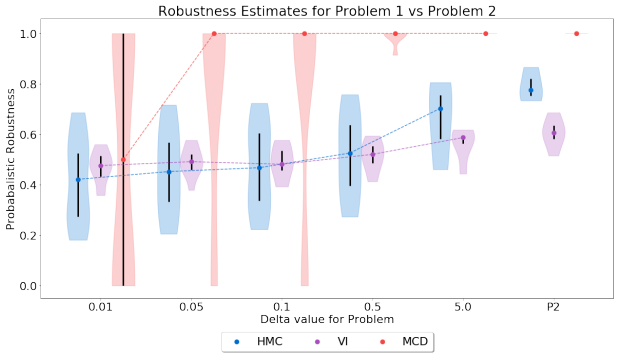

Figure 5: In this figure we demonstrate the difference between Problems 1 and 2. To generate the statistics in this plot we extended the reachability analysis shown in Figure 1.

Figure 6: We supplement the findings presented in figures 2 and 3 by running the exact same experiment but using the gradient based attack proposed in [Goodfellow *et al.*, 2014]. We note that though there is a slight quantitative variation between the figures (which is explained by the relative weakness of FGSM when compared to PGD), the qualitative behaviour in the robustness-accuracy trade-off is identical between the two tests.