A Framework for Including Uncertainty in Robustness Evaluation of Bayesian Neural Network Classifiers

Wasim Essbai
Technische Univ. Wien
Wien, Austria
wasim.essbai@tuwien.ac.at

Andrea Bombarda
University of Bergamo
Bergamo, Italy
andrea.bombarda@unibg.it

Silvia Bonfanti
University of Bergamo
Bergamo, Italy
silvia.bonfanti@unibg.it

Angelo Gargantini
University of Bergamo
Bergamo, Italy
angelo.gargantini@unibg.it

ABSTRACT

Neural networks (NNs) play a crucial role in safety-critical fields, requiring robustness assurance. Bayesian Neural Networks (BNNs) address data uncertainty, providing probabilistic outputs. However, the literature on BNN robustness assessment is still limited, mainly focusing on adversarial examples, which are often impractical in real-world applications. This paper introduces a fresh perspective on BNN classifier robustness, considering natural input variations while accounting for prediction uncertainties. Our approach excludes predictions labeled as "unknown", enabling practitioners to define alteration probabilities, penalize errors beyond a specified threshold, and tolerate varying error levels below it. We present a systematic approach for evaluating the robustness of BNNs, introducing new evaluation metrics that account for prediction uncertainty. We conduct a comparative study using two NNs – standard MLP and Bayesian MLP – on the MNIST dataset. Our results show that by leveraging estimated uncertainty, it is possible to enhance the system's robustness.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Software and its engineering → Software verification and validation**.

KEYWORDS

Robustness, Bayesian Neural Networks, Alterations, Uncertainty

ACM Reference Format:
Wasim Essbai, Andrea Bombarda, Silvia Bonfanti, and Angelo Gargantini. 2024. A Framework for Including Uncertainty in Robustness Evaluation of Bayesian Neural Network Classifiers. In *2024 IEEE/ACM International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest '24), April 20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3643786.3648026

1 INTRODUCTION

The robustness of artificial neural networks (ANN) refers to their ability to provide reliable and stable predictions in the face of uncertainty, noise, or alterations in the input data. Robustness is a desirable characteristic for ANNs, especially when used in critical scenarios [7]: since real-world data are often imperfect, noisy, and subject to unpredicted variations, ANNs may give wrong predictions causing harm to people. To address some shortcomings of ANNs, Bayesian NNs (BNNs) have been introduced, offering several advantages and proving valuable in various applications owing to their unique characteristics. A primary advantage lies in the fact that BNNs provide a probabilistic framework for modeling *uncertainty* through the use of Bayesian inference [9]. Instead of producing a single point estimation, they generate probability distributions over predictions, allowing for a better understanding and quantification of uncertainty associated with each prediction. This is particularly important in applications where awareness of uncertainty levels is crucial, such as in medical diagnoses or financial predictions [8, 26]. In these scenarios, BNNs offer the capability to take into account uncertainty in predictions, despite the potentially higher computational cost associated with their usage compared to conventional ANNs [16].

For this reason, BNNs started to be widely studied, although limited attention has been given to their robustness assessment. Specifically, existing works in the literature mainly analyze the robustness of BNNs in a probabilistic manner, focusing solely on adversarial examples, which do not represent all possible forms of input perturbations. Indeed, with adversarial examples, attackers exploit the internal structure of the NN to create artificial inputs that, in most cases, cannot be disclosed from the original ones but are designed in a way that they are misclassified or misinterpreted [20]. Moreover, several research studies have shown that adversarial examples are uncommon and unlikely to occur in many fields [19], such as in the medical one. Hence, it is preferable to give more attention to natural input perturbations, which are more probable to occur and are domain-specific. In particular, inputs subject to natural alterations are different from the original ones, due to input acquisition errors or unexpected environmental conditions. The desired property of the NNs is to keep the correct output even when inputs are altered.

In this paper, we propose a novel approach for evaluating the robustness of BNNs, incorporating uncertainty into the classification process by introducing the concept of *unknown predictions*. In this way, uncertainty can be exploited by the system to prevent taking decisions when the BNN is not certain about its predictions and, for example, it may require human users to intervene and take the burden of the decision (e.g., *collaborative AI*). Our approach evaluates the robustness of BNNs against natural input alterations and builds upon the framework introduced in [2], extending its application from deterministic ANNs to BNNs. With the proposed approach, in this paper, we aim to determine whether BNNs are more robust than standard NNs and if the utilization of uncertainty

DeepTest '24, April 20, 2024, Lisbon,Portugal

ACM ISBN 979-8-4007-0574-8/24/04.
https://doi.org/10.1145/3643786.3648026

can enhance their robustness. Additionally, we propose a comprehensive evaluation framework for BNNs that takes into account both predictive correctness and uncertainty.

In this study, we apply the proposed framework to a Bayesian Multilayer Perceptron (MLP) specifically designed for the classification task on the MNIST dataset. We then compare its performance with that of a standard deterministic MLP with the same structure. Our experiments show that the two networks demonstrate similar robustness when assessed through a deterministic robustness evaluation process. However, when the uncertainty is considered during classification, the BNN exhibits higher robustness. This underscores that adopting BNNs and considering their uncertainty can enhance both the performance and safety of the systems when the presented approach is employed in safety-critical domains.

The remainder of this paper is structured as follows. In Sect. 2, we provide an overview of BNNs, the uncertainty they model, and how they can be used to perform classification tasks. Moreover, we introduce the notion of input alterations used in the robustness analysis process. Following that, Sect. 3 and Sect. 4 detail the contributions made in this work. In the former, we delve into the classification using uncertainty, introducing the concept of indecision and extending the definition of accuracy by including the number of unknown samples. In the latter, we extend the definition of robustness proposed in [4], by adapting it to BNNs and including the number of unknown predictions for a more comprehensive assessment. Moving forward, in Sect. 5, we investigate whether the approach in Sect. 3 can enhance the robustness of the system embedding the BNN, using the robustness measures detailed in Sect. 4. Lastly, Sect. 6 provides a review of related works and Sect. 7 concludes the paper.

2 BACKGROUND

In this section, we first start by presenting the concepts of BNNs. Then, we delve into the uncertainty modeling, both aleatoric and epistemic. Finally, we introduce the classification problem, together with the definition of accuracy, and we give an intuition of what an input alteration is.

2.1 Bayesian Neural Networks

This section gives a brief description of BNNs in order to understand the main differences with standard NNs. A BNN is the result of the Bayesian approach applied to a standard NN, which introduces stochasticity [13]. In general, the goal of a standard NN is to find a function $y = f^{\mathbf{W}}(x)$ that maps input $\mathbf{x}$ and output $\mathbf{y}$, where $\mathbf{W}$ are the parameters of the network. In a standard NN, the parameters $\mathbf{W}$ are found during the training and are fixed point estimations. In a BNN, instead, the parameters $\mathbf{W}$ are considered to be random variables, and the training is done through Bayesian inference. This modeling strategy introduces stochasticity, meaning that at each prediction a sampling on the weights' distribution is performed. This allows multiple models $f^{\mathbf{W}}$, depending on the weights sampled, with a probability distribution $p(\mathbf{W})$. The output can thus be modeled as $\mathbf{y} = f^{\mathbf{W}}(x) + \epsilon$, where ϵ is a random noise indicating that $f^{\mathbf{W}}$ is only an approximation of the real mapping function.

Therefore, a BNN can be defined as any standard NN where the weights are assumed to be random variables and trained using Bayesian inference. Consequently, the output of the network also becomes a random variable. This allows for modeling the output using a probabilistic distribution.

During the training process, given a prior distribution on the weights and a training set D, the posterior is computed using the Bayes' theorem:

$$p(\mathbf{w}|D) = \frac{p(D|\mathbf{w}) \cdot p(\mathbf{w})}{p(D)} \tag{1}$$

Through the use of a prior distribution over the weights, it is possible to incorporate prior knowledge into the model. However, calculating the posterior may be challenging, as the term $p(D)$ could result in complex integrals that are computationally intractable.

2.1.1 *Inference algorithms.*

To overcome the complexity of integral computation, two main approaches are commonly used: Markov Chain Monte Carlo (MCMC), which is a sampling technique, and *variational* inference, which aims to provide an approximation of the posterior.

Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a popular family of algorithms that allows sampling from complex probability distributions [15]. While MCMC algorithms are powerful with exact posterior distributions, they face limitations in terms of scalability. As the size of the model grows, applying MCMC becomes increasingly computationally expensive and, in some cases, infeasible.

Variational inference

This set of techniques consists in finding the optimal approximation for the desired distribution [5]. This method introduces a distribution, referred to as the variational distribution, which is parameterized by a set of parameters. The goal is to learn the parameter values in a way that the variational distribution approximates the exact posterior. Essentially, the inference is reformulated as an optimization problem, aiming to minimize the disparity between the variational distribution and the true posterior distribution. This optimization process enhances the scalability and computational efficiency of variational inference compared to traditional MCMC methods. In the realm of ML, a common choice employed is the *mean-field variational* family [10]. This last inference algorithm is selected for this study due to its adequacy for our purposes and computational efficiency.

2.1.2 *Uncertainty modeling.*

The main advantage of using BNNs lies in the possibility to model the uncertainty, which can come both from the data or from the model itself.

Aleatoric uncertainty

Aleatoric uncertainty, also known as data uncertainty, refers to the inherent stochasticity present in the process being modeled with a BNN. This kind of uncertainty emerges from noisy and ambiguous data points. It represents an error that cannot be eliminated even with the addition of more data [12]. This type of uncertainty can be interpreted as the network's knowledge about a specific input: when a data point is very different from the ones used during training, the network would provide a prediction with higher uncertainty.

Epistemic uncertainty

Epistemic uncertainty, also referred to as model uncertainty, represents the uncertainty in the model parameters. Within the context

of BNNs, epistemic uncertainty captures the uncertainty in the model structure and architecture. This type of uncertainty can be reduced by acquiring more data or by employing a different model architecture. It can also be used for assessing the model quality.

2.1.3 Classification with BNNs.

In this work, we analyze the scenario of classification with BNNs. The BNN classifier's predicted output undergoes the same process as a standard NN. The network generates as output an N-dimensional vector p, where N represents the number of classes. The predicted class is determined as $C(d) = \arg\max(p)$. Notably, due to the probabilistic nature of BNNs, the vector p is derived by averaging multiple predictions for the same input, expressed as:

$$p = \frac{1}{T}\sum_{t=1}^{T} f^{\mathbf{w}_t}(x)$$

where $\mathbf{w}_t$ epresents the weight samples at the t^{th} prediction, and T is the number of predictions performed for the same input x. This averaging approach ensures a more stable output.

As for more classical deterministic NN classifiers, the correctness of a BNN classifier can be assessed through the *accuracy*.

Definition 1 (Accuracy)**.** Let C be a classifier and D a set of inputs. The *accuracy* of C on D is defined as:

$$acc(C, D) = \frac{\|d \in D | C(d) = correctLabel(d)\|}{\|D\|}$$

where $C(d)$ represents the classification output for the input d, and $correctLabel(d)$ gives the correct label for the input d.

Intuitively, the *accuracy* measures how many samples in D are correctly classified w.r.t. the total number of samples. In this study, the accuracy will serve as a quality metric for evaluating the classifier. Specifically, we will introduce a framework that enables the selection of a target accuracy value against which the robustness is to be assessed. The specified value can be seen as the minimum acceptable accuracy, and its choice is based on the user's requirements and the criticality of the task.

2.2 Input alterations

As in other previous works investigating the robustness of NN classifiers [2], in this paper we consider only *real-world alterations* that can affect the network's performance. In particular, an *alteration* can be defined as follows.

Definition 2 (Alteration)**.** An *alteration* of type A of an input t is a transformation of t that mimics the possible effect on t when a problem occurs in reality. The range of plausible alterations of type A is identified as $[L_A, U_A]$. Given a dataset D, D^{A_i} denotes the set obtained by altering all the data in D with an alteration of type A with *level* $i \in [L_A, U_A]$. The range of plausible alterations must be defined in a manner guaranteeing the existence of a level u for which $D^{A_u} = D$, signifying the absence of alteration.

Considering real-world scenarios, e.g., when images are given to a classifier, examples of alterations may be blur, zoom, Gaussian noise, etc. In general, alterations can happen with different probabilities that depend on the alteration level. Therefore, we model this phenomenon by defining an *alteration probability*.

Definition 3 (Alteration probability)**.** Let A be an alteration. p_A identifies the probability distribution of the alteration level having as support the interval $[L_A, U_A]$.

In practice, one may not know a priori the probability of each level of alteration. For this reason, the alteration probability can also serve as a tool to assign greater importance to certain alteration levels compared to others. An example of alteration probability is the *uniform* one: this distribution assigns equal importance to all alteration levels. Formally:

$$p_A(i) = \begin{cases} \dfrac{1}{U_A - L_A}, & for\ L_A \le i \le U_A \\ 0, & otherwise \end{cases}$$

3 CLASSIFICATION USING UNCERTAINTY

As previously introduced, BNNs allow for obtaining probability distribution as outputs modeling the uncertainty of the prediction. In this section, we leverage this uncertainty and we consider it during the classification process. The classification using uncertainty estimated by a BNN empowers the classifier from making wrong decisions, since only "certain" predictions will be considered valid.

The idea is to establish a threshold, and whenever the uncertainty surpasses this threshold, the system will output an *unknown* statement. Specifically, aleatoric uncertainty is employed as quantity to be kept under a defined threshold due to its closer association with the data, making it more appropriate. The threshold value should be proportional to the desired confidence in the predictions. In our investigation, we define the threshold using linear interpolation between the maximum and minimum uncertainty levels, formally:

$$threshold = maxUnc * (1 - \alpha)$$

where $maxUnc$ is the maximum uncertainty that can occur (in most cases, it can be considered 100%) in the specific application and $\alpha \in [0, 1]$ is the confidence level. The maximum level of aleatoric uncertainty is reached by considering the most uncertain scenario, wherein all values in the output vector are equal. On the other side, the minimum uncertainty achievable is 0, as it is always a non-negative quantity. This minimum is reached when the output vector contains only zero values, except for one, since this means that the network is sure of its decision. The use of this threshold allows us to introduce the concept of *indecision*, defined as follows.

Definition 4 (Indecision)**.** Let C be a classifier and D a set of inputs. The *indecision* of C on D is defined as:

$$ind(C, D) = \frac{\|d \in D | C(d) = unknownLabel\|}{\|D\|}$$

where $C(d)$ represents the classification output for the input d.

Intuitively, this quantity measures the portion of samples in D for which C is not able to give a "certain" classification w.r.t. to the total number of samples. To take into account the number of unknown predictions of a BNN, the definition of accuracy previously given in Def. 1 is modified to exclude from its computation the samples on which the BNN is indecisive.

Definition 5 (Accuracy in the presence of indecision)**.** Let C be a classifier and D a set of inputs. The *accuracy* of C on D when the

indecision is considered is defined as:

$$acc(C, D) = \frac{\|d \in D | C(d) = correctLabel(d)\|}{\|D\| - \|d \in D | C(d) = unknownLabel\|}$$

where $C(d)$ represents the classification output for the input d, and $correctLabel(d)$ gives the correct label for the input d.

Excluding unknowns aims to emphasize the network accuracy in its actual predictions. This choice can be justified, particularly in critical applications like medical diagnosis. In such cases, receiving an unknown statement and seeking support from a (human) doctor is preferable to making an inaccurate decision. Def. 5 formally captures this concept, evaluating the BNN only on predictions that are not considered as unknown and are thus potentially critical.

4 ROBUSTNESS MEASURE

In Arcaini et al. [2], a novel approach to evaluate the robustness of a CNN classifier has been presented. Our study extends this approach to BNNs to take into account uncertainty. Indeed, exposing a network to altered images typically leads to a reduction in overall accuracy. The concept here is to measure the robustness by evaluating how much the accuracy decreases when the network encounters perturbed data due to some alteration (see Sect. 2.2). Intuitively, a lower reduction in accuracy indicates higher robustness.

To articulate this, we introduce the notion of *tolerance*, which quantifies the *reward* for maintaining an acceptable performance. This performance can be measured through accuracy or any other quality metric, such as precision, recall, or F1-score. Therefore, given a quality metric, we need to define the worst-case accepted value and tolerate differently every value that is "better" than the worst accepted one.

Definition 6 (Tolerance)**.** Within the context of classification, given a quality metric x and the minimum accepted threshold th, the tolerance is a function denoted as $tol(x, th)$, such that:

$$\begin{cases} 0 \leq tol(x, th) \leq 1, \ for\ x \geq th \\ tol(x, th) = 0, \ for\ x < th \end{cases}$$

Depending on the case-specific requirements, different functions may be used. One typical example of tolerance is the *linear* tolerance, that can be expressed as:

$$tol(x, th) = \frac{max(min(x, x_{max}) - th, 0)}{x_{max} - th}$$

This function is suitable when it is necessary to be above the threshold with some margin. The denominator is added for normalization, x_{max} represents the value of x with maximum tolerance. In general, it can be the maximum value that x can assume, but lower values may also make sense. In other words, x_{max} defines the baseline with respect which to evaluate the robustness.

Following the same idea, it is possible to define a way to penalize instances where the performance declines to unacceptable levels.

Definition 7 (Penalization)**.** Within the context of classification, given a quality metric x and the minimum accepted threshold th, the *penalization* is a function denoted as $dep(x, th)$, such that:

$$\begin{cases} 0 \leq dep(x, th) \leq 1, \ for\ x \leq th \\ dep(x, th) = 0, \ for\ x > th \end{cases}$$

The two previously presented functions are intended to separately formulate strategies for rewarding and penalizing the performance trend. Specifically, the tolerance function operates solely within the range where performance remains acceptable, whereas the penalization function yields positive values when performance falls below acceptable levels.

4.1 Robustness without uncertainty

Given the previous definitions, we now introduce the definition of robustness against an alteration A. In particular, given A, the robustness takes into account the probability of each alteration level, the tolerance of the system to the decrease in accuracy, and the penalization for unacceptable accuracy values.

Definition 8 (Robustness)**.** Let C be a classifier, tol be a tolerance function, dep be a penalization function, and $acc(C, D^{A_i})$ be the accuracy of the classifier C over the input dataset D when the alteration A is applied with level i. The robustness $rob^A(C, D) \in [0, 1]$ of a classifier C w.r.t. alteration of type A in the range $[L_A, U_A]$ on a dataset D is formally defined as:

$$rob^A(C, D) = \frac{\int_{L_A}^{U_A} [tol(acc(C, D^{A_i}), \Theta) - dep(acc(C, D^{A_i}), \Theta)] \cdot p_A(i)\ di}{2} + \frac{1}{2}$$

where Θ is a threshold referring to minimum accuracy accepted, p_A is the probability distribution of the alteration levels, and $\frac{1}{2}$ is a regularization factor added to make the robustness always in $[0, 1]$.

This formula is derived from [4] and has been adjusted to incorporate penalization terms. Intuitively, the robustness is calculated as the sum (i.e., integral) of the difference between the reward (i.e., the tol function) and the penalization (i.e., the dep function) assigned to the results obtained when a defined level of alteration is applied to all input samples, with weighting based on the probability of the considered alteration level.

4.2 Robustness evaluation using uncertainty

The presented approach relies solely on accuracy as the quality metric. However, when dealing with uncertainty, this may result in a non-comprehensive metric. For example, using the definition of accuracy in Def. 5, a network that outputs only unknown values would achieve 100% accuracy. Although the robustness formula in Def. 8 would produce a result of 1, indicating high robustness, the network is essentially ineffective. These observations led to the concept of *robustness against indecision*, defined as follows:

Definition 9 (Robustness against indecision)**.** Let C be a classifier, tol a tolerance function, and dep a penalization function. The robustness against indecision $rob^A_{Ind}(C, D) \in [0, 1]$ of a classifier C w.r.t. alteration of type A in the range $[L_A, U_A]$ on a dataset D is defined formally as:

$$rob^A_{Ind}(C, D) = \frac{\int_{L_A}^{U_A} [tol(1 - ind(C, D^{A_i}), \gamma) - dep(1 - ind(C, D^{A_i}), \gamma)] \cdot p_A(i)\ di}{2} + \frac{1}{2}$$

where γ is a threshold referring to minimum accepted ratio of "certain" predictions, p_A is the probability distribution of the alteration levels and $ind(C, D^{A_i})$ is the unknown ratio of C evaluated on D^{A_i}.

This new definition has a similar structure to the one in Def. 8. Nonetheless, in this instance, the robustness metric assumes a different meaning. It pertains to the network's capacity to provide "certain" predictions even in the presence of altered input data.

The metrics introduced in Def. 8 and Def. 9 can be employed in conjunction to achieve a more comprehensive evaluation, by considering both *accurate* results and only "certain" predictions, when uncertainty is used in the classification process.

4.3 Effectiveness

The inclusion of uncertainty into the robustness evaluation process is crucial. However, sometimes it is more practical to have a single comprehensive metric that incorporates all the necessary information. Following the approach presented in Sect. 3, when uncertainty is used to compute the accuracy, the two primary quality metrics used are accuracy and indecision. In this study, we propose to combine both metrics to derive a complete, singular metric called *effectiveness* and defined as follows.

Definition 10 (Effectiveness)**.** Let C be a classifier, D an input dataset, and $acc(C, D)$ and $ind(C, D)$, respectively, the accuracy and the indecision of the classifier C over D. The *effectiveness* of C on D is:

$$eff(C, D) = \frac{acc(C, D) \cdot (1 - ind(C, D))}{ind(C, D) + 1}$$

Intuitively, this definition implies that a network that outputs exclusively unknown values is equally ineffective as one that solely produces incorrect predictions. Furthermore, it is directly proportional to accuracy and inversely proportional to indecision. As a result, it quantifies how much the network is effective in its predictions. This metric exhibits the following properties:

- When the accuracy is zero, the effectiveness is zero;
- When the indecision is one, the effectiveness is zero;
- When the indecision is zero, the effectiveness reflects the accuracy, i.e., $eff(C, D) = acc(C, D)$;
- For a fixed accuracy value, the lower the indecision value, the higher the effectiveness. Thus, given two networks with the same accuracy, it penalizes the most uncertain one.

With this new quality metric, it becomes possible to derive a comprehensive robustness metric, evaluating indecision and accuracy at the same time. This can be achieved by using the effectiveness as the quality metric in the integral presented in Def. 11. We refer to this new comprehensive metric as *augmented robustness*.

Definition 11 (Augmented robustness)**.** Let C be a classifier, tol a tolerance function, and dep a penalization function. The augmented robustness $rob^A_{Aug}(C, D) \in [0, 1]$ of a classifier C w.r.t. alteration of type A in the range $[L_A, U_A]$ on a dataset D is defined formally as:

$$rob^A_{Aug}(C, D) = \frac{\int_{L_A}^{U_A} [tol(eff(C, D^{A_i}), \beta) - dep(eff(C, D^{A_i}), \beta)] \cdot p_A(i)\ di}{2} + \frac{1}{2}$$

where β is a threshold referring to minimum effectiveness accepted and p_A is the probability distribution of the alteration levels.

In this definition, selecting the β parameter might be challenging as it is less intuitive. One possible approach for its determination is to independently choose the parameters Θ in Def. 8 and γ in Def. 9, and then set $\beta = \frac{\Theta}{\gamma+2} \cdot \gamma$.

5 EXPERIMENTAL EVALUATION

In this section, we present the results of the robustness analysis conducted on our developed models and we draw comparisons between them. Specifically, we define a set of plausible alterations, inspired by the work of Arcaini et al. [2]. This enables us to compute the accuracy on the altered data and subsequently to compute the robustness scores introduced in Sect. 4. Our experimental evaluations aim to answer to the following research questions (RQs):

- RQ1 Are BNNs, used in a deterministic manner, more robust than standard NNs?
- RQ2 Can the robustness of BNNs be improved by incorporating uncertainty into the classification process?
- RQ3 Is relying only on the accuracy adequate for evaluating BNNs when uncertainty is employed?

5.1 Case Study

In the following experiments, we will use the Modified National Institute of Standards and Technology (MNIST) dataset [17], which is a popular benchmark for image classification tasks.

To assess the proposed methods, we compare two networks with identical architecture: a standard NN and a BNN. The performance of the standard NN will serve as a benchmark, providing a reference against which the performance of the BNN can be compared. The architecture chosen in this work is a Multilayer Perceptron (MLP). Although for image processing tasks Convolutional Neural Networks (CNNs) are a better choice [1], the MLP has a more understandable, simple, and intuitive structure. Indeed, during the design phase, CNNs require the specification of more hyperparameters such as the number of filters, their dimensions, stride, and padding. Although these settings enhance CNNs' effectiveness in handling grid-scale data, they simultaneously introduce complexity in model interpretation. Consequently, CNNs are more sensitive to the choice of hyperparameters and the evaluation results may be influenced by this selection. It is important to note that the network architecture does not impact the generality of the applied approaches. Future work studies may explore alternative architectures for further investigations with the same framework. Both networks have three fully connected layers, with 784 neurons in the input layer, 100 in the hidden layer, and 10 in the output layer. However, in terms of trainable parameters, the standard NN has 79, 510 parameters, whereas the BNN has 161, 040 trainable parameters, approximately double the count. This higher number is due to the weight modeling using normal distributions. Consequently, for each connection, two parameters are needed: the mean and the variance of the distribution.

5.2 Alterations

The alterations selected in the experiments are general perturbations that could potentially arise during image acquisition. It is important to note that networks used in this work are designed for research purposes and are not intended for a specific application. Consequently, a precise domain analysis prior to the experiments was not feasible. Nevertheless, we took care to select general and plausible perturbations to assess the validity of our approach.

For each alteration, we define an alteration range $[L_A, U_A]$, in which we uniformly sample 21 alteration levels to evaluate the

Table 1: Alterations used for robustness analysis

Alteration type	Alteration level	$[L_A, U_A]$
Gaussian noise (GN)	Variance noise	$[0, 0.20]$
Blur (B)	Blurring radius	$[0, 2]$
Brightness variation (BV)	Brightness %	$[-0.5, 0.5]$
Horizontal translation (HT)	Pixel number	$[-20, 20]$
Vertical translation (VT)	Pixel number	$[-20, 20]$
JPEG Compression (JC)	Compression level	$[0, 100]$
Zoom (Z)	Zoom level	$[1, 2]$

accuracy. Tab. 1 shows the alterations chosen for the robustness analysis with their respective ranges.

5.3 Experimental setup

Initially, for both models, the robustness computation follows Def. 8 and we do not consider any uncertainty in our evaluation. In these experiments, the tolerance function is linear, with a threshold $\Theta = 0\%$ and *maxAcc* set to the nominal accuracy (i.e., the accuracy of the network when no alteration is applied to the input samples). The choice of the minimum accepted accuracy $\Theta = 0\%$ aims to yield independent results across different application domains, but this does not undermine the generality of our approach. With $\Theta = 0\%$, no penalization is applied, resulting in a metric ranging from 0.5 in the worst-case scenario to 1 in the best-case scenario. This metric is computed for both models as it employs only the accuracy.

For the BNN, we conduct a further analysis. Specifically, for each alteration, we perform the classification without and with aleatoric uncertainty to investigate whether there is a performance improvement. The threshold to generate unknown predictions is calculated with a confidence level of $\alpha = 0.80$, and aleatoric uncertainty is computed using the approach presented in [14]. Subsequently, we compute the robustness against indecision, as defined in Def. 9, by using a linear tolerance and a threshold $\gamma = 0\%$. Finally, we compute the augmented robustness, as defined in Def. 11.

Additionally, to ensure a fair comparison between various types of alterations and robustness metrics, we maintain a uniform probability distribution of alteration levels for all alteration types. Moreover, in the case of BNNs, given their probabilistic nature, we perform ten predictions for each input sample.

5.4 Evaluation results

Initially, in this section, we compare the robustness of the BNN with the one of the standard NN without employing uncertainty in the classification process. Subsequently, we integrate the uncertainty into the classification to assess whether this technique could improve the BNN's robustness compared to the deterministic scenario.

RQ1: Robustness analysis without uncertainty.
Alg. 1 reports the process we have followed to perform the robustness analysis without uncertainty. Initially, the process starts by sampling all alteration steps within the alteration interval (line 1). Subsequently, for each alteration level, the algorithm computes the accuracy over the modified input data set. During this procedure, the input data set is altered by applying the chosen alteration at the alteration level at that step (line 3). Subsequently, by using an

Algorithm 1 Robustness evaluation algorithm without uncertainty

Require: *image_data*, the test set of images used to evaluate the network
Require: *labels*, the true labels for the *image_data*
Require: *model*, the model trained to classify *image_data*
Require: *alteration*, the applied alteration
Require: *alt_range*, the identified alteration range $[L_A, U_A]$
Require: *n*, the number of levels to uniformly sample from the alteration range
Require: *alt_prob*, the probability distribution of the alteration to be applied
Require: *tol*, the tolerance function to use
Require: *dep*, the penalization function to use
Ensure: *rob*, the computed robustness value
1: $alt\_steps \leftarrow sampleRange(L_A, U_A, n)$
2: **for all** $l \in alt\_steps$ **do** ▷ For each level of alteration
▷ Apply the alteration level to all the *image_data* with level l
3: $altered\_image\_data \leftarrow alteration.applyAlteration(image\_data, l)$
4: $outputs \leftarrow model.predict(altered\_image\_data)$
5: $predictions \leftarrow classify(outputs)$
6: $accuracy \leftarrow computeAccuracy(predictions, labels)$
7: $step\_list.add(l)$
8: $acc\_list.add(accuracy)$
9: **end for**
10: *rob* ← Robustness(*acc_list, step_list, tol, dep, alt_steps*)
11: **return** *rob*

Table 2: Robustness of the two NNs w.r.t. natural alterations

Alteration	GN	B	BV	HT	VT	JC	Z	Avg.
rob^A std. NN [%]	97.40	96.01	99.88	81.03	84.12	62.62	74.67	85.10
rob^A BNN [%]	96.81	93.31	99.88	85.63	86.25	62.08	77.31	85.90

altered input data set, the NN is tasked with predicting the classes to which all inputs belong (from line 4 to line 5). The accuracy over the whole altered input data set is calculated at line 6. Lastly, using all accuracy values computed for each alteration level, the algorithm determines the robustness of the NN (line 10) by applying the formula as in Def. 8.

We have executed this algorithm for all the alterations specified in Tab. 1, and the obtained results are summarized in Tab. 2. Notably, both networks exhibit similar performance across all alterations. Interestingly, the lack of robustness to translation is consistent with expectations, considering one of the inherent limitations of MLPs, namely their inefficiency in handling translation and grid data [1]. Consequently, the Bayesian approach does not appear to overcome the structural limitations of the network's architecture.

Answer to RQ1: *Our experiments have shown that, in general, using a BNN deterministically does not lead to any significant robustness enhancement. Moreover, we have shown that the BNN exhibit the same architectural limitations of the standard NN.*

RQ2 and RQ3: Robustness analysis using uncertainty.
To include the uncertainty in the classification process, we use Alg. 2 which has the same structure as Alg. 1, except that the classification can output unknown statements. In particular, the main differences between the two procedures lie in line 5, where the output probabilities are compared with the $threshold$ to define which samples are classified as "unknown", and in line 7 and 8, where the indecision and effectiveness, used to compute rob^A_{Ind} and rob^A_{Aug}, are calculated.

The evaluation results are summarized in Table 3. In addition to the robustness metrics, we also report the increase in robustness Δrob w.r.t. the deterministic case reported in Table 2.

Algorithm 2 Robustness evaluation algorithm using uncertainty

Require: *image_data*, the test set of images used to evaluate the network
Require: *labels*, the true labels for the *image_data*
Require: *model*, the model trained to classify *image_data*
Require: *alteration*, the applied alteration
Require: *alt_range*, the identified alteration range $[L_A, U_A]$
Require: *n*, the number of levels to uniformly sample from the alteration range
Require: *alt_prob*, the probability distribution of the alteration to be applied
Require: *tol*, the tolerance function to use
Require: *dep*, the penalization function to use
Require: $threshold$, the desired threshold for "certain" predictions
Ensure: *rob*, the computed robustness value
Ensure: *rob_ind*, the computed robustness against indecision value
Ensure: *rob_aug*, the computed augmented robustness value

```
alt_steps ← sampleRange(L_A, U_A, n)
for all l ∈ alt_steps do                          ▷ For each level of alteration
        ▷ Apply the alteration level to all the image_data with level l
    altered_image_data ← alteration.applyAlteration(image_data, l)
    outputs ← model.predict(altered_image_data)
    predictions ← classifyUsingUncertainty(outputs, threshold)
    accuracy ← computeAccuracy(predictions, labels)
    indecision ← computeIndecision(predictions, labels)
    effectiveness ← computeEffectiveness(accuracy, indecision)
    step_list.add(l)
    acc_list.add(accuracy)
    ind_list.add(indecision)
    eff_list.add(effectiveness)
end for
rob ← Robustness(acc_list, step_list, tol, dep,alt_steps)
rob_ind ← RobustnessIndecision(ind_list, step_list, tol, dep,alt_steps)
rob_aug ← RobustnessAugmented(eff_list, step_list, tol, dep,alt_steps)
return rob, rob_ind, rob_aug
```

It is noteworthy that the increase is always positive, meaning that leveraging uncertainty enables the attainment of more robust models. However, this enhanced robustness does not come without trade-offs, as there is always a cost associated with the number of unknown predictions. For instance, in the case of GN, the increase in robustness ($\Delta rob^A = 1.02\%$ in Tab. 3) comes with good robustness against indecision (96.24%). Depending on specific user requirements, this trade-off may or may not be acceptable as someone may prefer to have a higher increase in robustness at the expense of a lower robustness against indecision. The same situation does not apply uniformly to translation, where although there is a significant increase in robustness (8.82% for HT and 5.82% for VT) but at the expense of lower robustness against indecision (around 90%). Therefore, we can infer that, in general, the developed BNN exhibits greater robustness to GN compared to translation.

All these observations are encapsulated in our novel metric rob^A_{Aug}, which essentially combines robustness concerning both accuracy and indecision. While the *rob* metric alone may suggest that the network is robust at a similar level to GN, B, and BV, rob^A_{Aug} values indicate that the best performance is achieved for BV, where a high robustness is accomplished along with an equally high level of rob^A_{Ind}. rob^A_{Aug} is particularly valuable when comparing networks with similar robustness and can serve as a selection criterion. Furthermore, it proves beneficial in applications where it is desired that robustness comes with a low degree of unknown predictions.

Answer to RQ2: *We have shown that leveraging uncertainty in the classification process can successfully identify ambiguous situations, thereby preventing wrong decisions. This inherently contributes to making BNNs more robust compared to standard NNs.*

Table 3: Robustness using uncertainty w.r.t. the analyzed alterations

Alteration	GN	B	BV	HT	VT	JC	Z	Avg.
rob^A [%]	98.42	96.16	99.97	89.85	89.94	75.33	79.50	89.88
rob^A_{Ind} [%]	96.24	91.60	98.90	89.93	90.65	76.55	89.01	90.41
rob^A_{Aug} [%]	92.17	85.90	98.13	77.91	78.63	58.54	72.09	80.48
Δrob^A [%]	+1.02	+0.15	+0.09	+8.82	+5.82	+12.71	+4.83	+3.98

Answer to RQ3: *Using accuracy alone is insufficient for a comprehensive evaluation of BNNs when uncertainty is used during classification. It is crucial to employ metrics that describe the network's performance in terms of uncertainty. In this study, we introduced the concepts of indecision in Def. 4 and effectiveness in Def. 10.*

6 RELATED WORK

ANNs are nowadays commonly used in safety-critical tasks, and thus, in the literature, a lot of effort has been spent on their validation. However, unlike classical software, in which the behavior of the artifacts is known a priori and classical white-box testing approaches can be applied, with NNs mainly empirical strategies can be used [23]. In the last years, several attempts at testing NNs have been carried on, especially by focusing on evaluating the *robustness* of such systems. Most of the works are focused on assessing the robustness of NN-based systems against adversarial inputs [20, 24, 25, 27, 28], i.e., inputs designed to fool NN systems by taking into account their internal structure. However, various research studies have demonstrated that such adversarial input perturbations are not common in real-world scenarios [19]. Therefore, there is a growing consensus on the importance of directing attention towards other more plausible scenarios to comprehensively evaluate the actual robustness of an ANN [11, 22]. For this reason, in [2], the authors introduced the concept of robustness against natural alterations, which considers the domain-specific perturbations more likely to occur in practice. This concept was initially applied to CNNs, used for classification, and is further supported by a dedicated Python library [3]. Following that, this definition of robustness has been extended in [4] for NNs used as estimators.

In the context of BNNs, the problem of assessing the robustness is conventionally approached in a probabilistic way. For example, in [6], robustness is defined as the probability that, given a test point, there exists a point within a bounded set such that the prediction of the BNN differs between the two points. Instead, our paper presents a distinctive approach by introducing the novel concept of *unknown predictions*, a dimension that has not been explored in the literature analyzing the robustness of BNNs so far. In contrast to previous works, our focus is to investigate the implications of unknown predictions on BNN robustness assessment. Finally, as for more classical NNs, many researchers have focused on analyzing the performance (e.g., the robustness) of BNNs when adversarial attacks are performed [18, 21]. Nevertheless, as previously stated, adversarial inputs are often unlikely in numerous critical application domains, such as in the medical field. For this reason, our paper focuses on more plausible and natural input alterations, reflecting the practical constraints and requirements of critical applications.

Our work builds upon the concept of incorporating uncertainty into robustness evaluation thanks to the use of BNNs. Although recent works, such as [14], frequently delve into exploring the uncertainty outputted by BNNs, a notable gap in the literature exists. Indeed, the integration of uncertainty into the robustness assessment has not been widely studied. This is a key aspect of our research, aiming to explore and leverage uncertainty estimates provided by BNNs to obtain and assess more robust models.

7 CONCLUSIONS

This paper introduces the first approach for evaluating robustness considering both prediction accuracy and uncertainty through the novel *indecision* concept. Furthermore, we combine precision and indecision in a new metric referred to as *effectiveness*. Our proposed robustness definition, to be applied when inputs are subject to natural perturbations, enables independent expression of tolerance and penalization of performance w.r.t. a specified threshold. Furthermore, our definitions allow practitioners to obtain continuous metrics (instead of using a yes/no metric), making it possible to finely characterize the network's performance. The experimental assessment reveals that BNNs show a robustness level comparable to that of standard NNs with the same architecture when they are utilized conventionally. However, incorporating uncertainty into the classification process has the potential to yield more robust models. We believe that this approach could serve as a foundational starting point toward the development of more robust and autonomous artificial intelligence systems. It is noteworthy that while this framework was developed using a BNN, its applicability extends beyond, as it can be generalized for any machine learning model that provides uncertainty estimations.

Limitations and Future Work. In our experimental evaluation, we have used the simple MNIST case study in order to verify the feasibility of our approach and to introduce the problem of robustness estimation for BNNs. Indeed, this paper presents a preliminary attempt to introduce uncertainty in robustness evaluation. However, we are working on applying the same framework to a more complex case study, to verify the scalability of the proposed approach and its generalizability to other application scenarios and domains.

Incorporating uncertainty enhances classifier robustness is somewhat expected. Our definition of robustness is able to capture this. However, we would like to work on assessing the usefulness of our approach by discussing it with experts and by applying it to industrial case studies. Additionally, we aim to investigate the possible enhancement of BNN robustness through the adoption of existing methods and techniques of robust training. In particular, we seek to analyze changes in indecision when those methods are used. Another interesting aspect is to study whether a correlation between "unknown" and incorrect predictions exists. This would provide valuable insights into the network's efficiency in making accurate predictions.

ACKNOWLEDGMENTS

This work has been partially funded by PNRR - ANTHEM (AdvaNced Technologies for Human-centrEd Medicine) - PNC0000003 – CUP: B53C22006700001 - Spoke 1 - Pilot 1.4.

REFERENCES

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi. 2017. Understanding of a convolutional neural network. In *2017 Int. Conf. on Engineering and Technology (ICET)*. 1–6.
[2] P. Arcaini, A. Bombarda, S. Bonfanti, and A. Gargantini. 2020. Dealing with Robustness of Convolutional Neural Networks for Image Classification. In *2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*. 7–14.
[3] P. Arcaini, A. Bombarda, S. Bonfanti, and A. Gargantini. 2021. ROBY: a Tool for Robustness Analysis of Neural Network Classifiers. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. 442–447.
[4] P. Arcaini, A. Bombarda, S. Bonfanti, A. Gargantini, D. Gamba, and R. Pedercini. 2022. Robustness assessment and improvement of a neural network for blood oxygen pressure estimation. In *2022 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE Computer Society.
[5] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2017. Variational Inference: A Review for Statisticians. *J. Amer. Statist. Assoc.* 112, 518 (2017), 859–877.
[6] L. Cardelli, M. Kwiatkowska, et al. 2019. Statistical Guarantees for the Robustness of Bayesian Neural Networks. arXiv:1903.01980 [cs.LG]
[7] N. Carlini and D. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. 39–57.
[8] J. Gawlikowski, C. R. N. Tassi, et al. 2023. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review* 56, Suppl 1 (2023), 1513–1589.
[9] E. Goan and C. Fookes. 2020. Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018* (2020), 45–87.
[10] Wei Han and Yun Yang. 2019. Statistical Inference in Mean-Field Variational Bayes. *arXiv: Statistics Theory* (2019).
[11] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. arXiv:1903.12261 [cs.LG]
[12] E. Hüllermeier and W. Waegeman. 2021. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning* 110 (03 2021).
[13] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun. 2022. Hands-On Bayesian Neural Networks—A Tutorial for Deep Learning Users. *IEEE Computational Intelligence Magazine* 17, 2 (2022), 29–48.
[14] Yongchan K., Joong-Ho W., Beom J. K., and Myunghee C. P. 2020. Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics and Data Analysis* 142 (2020), 106816.
[15] F. Korner-Nievergelt, T. Roth, S. von Felten, J. Guélat, B. Almasi, and P. Korner-Nievergelt. 2015. *Markov Chain Monte Carlo Simulation.* Elsevier, 197–212.
[16] J. Lampinen and A. Vehtari. 2001. Bayesian approach for neural networks—review and case studies. *Neural networks* 14, 3 (2001), 257–274.
[17] Y. LeCun and C. Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010).
[18] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. 2019. Adv-BNN: Improved Adversarial Defense through Robust Bayesian Neural Network. arXiv:1810.01279 [cs.LG]
[19] R. Mangal, A. V. Nori, and A. Orso. 2019. Robustness of Neural Networks: A Probabilistic and Practical Approach. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE Computer Society, Los Alamitos, CA, USA.
[20] M. H. Meng, G. Bai, S. G. Teo, Z. Hou, Y. Xiao, Y. Lin, and J. S. Dong. 2022. Adversarial Robustness of Deep Neural Networks: A Survey from a Formal Verification Perspective. *IEEE Trans. on Dependable and Secure Computing* (2022).
[21] Y. Pang, S. Cheng, J. Hu, and Y. Liu. 2021. Evaluating the Robustness of Bayesian Neural Networks Against Different Types of Attacks. arXiv:2106.09223 [cs.LG]
[22] E. Rusak, L. Schott, R. S. Zimmermann, J. Bitterwolf, O. Bringmann, M. Bethge, and W. Brendel. 2020. A simple way to make neural networks robust against diverse image corruptions. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 53–69.
[23] J. Schumann, P. Gupta, and S. Nelson. 2003. On verification & validation of neural network based controllers. (2003).
[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. 2014. Intriguing properties of neural networks. 2nd International Conference on Learning Representations, ICLR 2014.
[25] Beilun Wang, Ji Gao, and Yanjun Qi. 2016. A Theoretical Framework for Robustness of (Deep) Classifiers against Adversarial Examples. *arXiv: Learning* (2016). https://api.semanticscholar.org/CorpusID:35742668
[26] G. Wang, W. Li, M. Aertsen, et al. 2019. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing* 338 (2019), 34–45.
[27] T. W. Weng, H. Zhang, et al. 2018. Evaluating the robustness of neural networks: An extreme value theory approach. In *Int. Conf. on Learning Representations*.
[28] A. W. Wijayanto, C. Jun Jin, K. Madhawa, and T. Murata. 2018. Robustness of Compressed Convolutional Neural Networks. In *2018 IEEE International Conference on Big Data (Big Data)*. 4829–4836.