Neural Computing and Applications (2021) 33:10259–10275
https://doi.org/10.1007/s00521-021-05789-y

ORIGINAL ARTICLE

Leveraging the Bhattacharyya coefficient for uncertainty quantification in deep neural networks

Pieter Van Molle[1] · Tim Verbelen[1] · Bert Vankeirsbilck[1] · Jonas De Vylder[2] · Bart Diricx[2] · Tom Kimpe[2] · Pieter Simoens[1] · Bart Dhoedt[1]

Received: 23 June 2020 / Accepted: 5 February 2021 / Published online: 1 March 2021

Abstract
Modern deep learning models achieve state-of-the-art results for many tasks in computer vision, such as image classification and segmentation. However, its adoption into high-risk applications, e.g. automated medical diagnosis systems, happens at a slow pace. One of the main reasons for this is that regular neural networks do not capture uncertainty. To assess uncertainty in classification, several techniques have been proposed casting neural network approaches in a Bayesian setting. Amongst these techniques, Monte Carlo dropout is by far the most popular. This particular technique estimates the moments of the output distribution through sampling with different dropout masks. The output uncertainty of a neural network is then approximated as the sample variance. In this paper, we highlight the limitations of such a variance-based uncertainty metric and propose an novel approach. Our approach is based on the overlap between output distributions of different classes. We show that our technique leads to a better approximation of the inter-class output confusion. We illustrate the advantages of our method using benchmark datasets. In addition, we apply our metric to skin lesion classification—a real-world use case—and show that this yields promising results.

Keywords Deep learning · Bayesian networks · Uncertainty · Dropout Monte Carlo

1 Introduction

In the field of computer vision, deep learning has time and again set new state-of-the-art benchmarks for a variety of tasks, such as large-scale image classification [17, 43, 45], object localization [13, 14, 38], and semantic segmentation [29, 39]. Quantifying and handling uncertainty in decision taking is well known in other domains, for example using fuzzy sets in user behaviour tracking [1] or database retrieval [49]. In deep learning, however, uncertainty is often disregarded. It is known that neural networks only output a point estimate of the true underlying predictive distribution. On top of this, the softmax output layer that is typically used to get a probability score, is in general "over-confident" for one class [10]. Therefore, its output must not be interpreted as model confidence.

In many real-world scenarios, the ability to capture uncertainty is indispensable. Without this notion of uncertainty, both the certain and uncertain inputs, as well as special cases, would be treated equally. This behaviour is undesired, especially in high-risk applications, such as

✉ Pieter Van Molle
pieter.vanmolle@ugent.be

Tim Verbelen
tim.verbelen@ugent.be

Bert Vankeirsbilck
bert.vankeirsbilck@ugent.be

Jonas De Vylder
jonas.devylder@ugent.be

Bart Diricx
bart.diricx@barco.com

Tom Kimpe
tom.kimpe@barco.com

Pieter Simoens
pieter.simoens@ugent.be

Bart Dhoedt
bart.dhoedt@ugent.be

[1] IDLab, Department of Information Technology, Ghent University - imec, Ghent, Belgium

[2] Barco N.V., Kortrijk, Belgium

computer-aided medical diagnosis systems. If such a system had the capability to reason about uncertainty, it could refer a patient to a trained professional when it encounters a difficult case. Otherwise, the system could create a false sense of security, having potentially lethal consequences.

Bayesian probability theory offers a sound mathematical framework to design machine learning models with an inherent and explicit notion of uncertainty. Instead of resulting in a single per-class probability, such models are able to estimate the moments of the output distribution for every class, including mean and variance. Despite recent advances in fitting deep learning in a Bayesian framework [4, 15], Bayesian neural networks have not seen a high adoption rate. This is largely due to difficulties in implementation and excessive training times, as well as higher resource and memory requirements at inference time. As a low-cost alternative, [10] prove that training a neural network with dropout is equivalent to variational inference in Bayesian neural networks. Obtaining an estimate of the predictive log-likelihood distribution then boils down to selecting Monte Carlo samples of the neural network output using different dropout masks.

The most commonly used metric to quantify output uncertainty is the variance of the output samples, i.e. the predictive variance. However, we argue that this metric is cumbersome to use. First, variance-based metrics yield values that are typically very small in absolute value and are therefore hard to interpret. Second, such metrics do not take into account any overlap between the output distributions for different classes: a significant overlap could indicate a high level of doubt between these classes. In this paper, we propose a novel uncertainty metric that has a range bounded between 0 (very little uncertainty) and 1 (high uncertainty), and is therefore easy to interpret. In addition, our metric encompasses the distributional overlap between the output classes. We compare our metric to the predictive variance using benchmark datasets. Additionally, we apply our metric to the real-world use case of skin lesion classification. Of all cancers, skin cancer is the most diagnosed in the USA. However, when discovered early, the survival rate for skin cancer exceeds 98% [2]. Since early detection happens visually, this is an excellent case for deep learning. In controlled settings, neural networks achieve similar classification performance to a team of trained dermatologists [6, 8, 32, 33]. However, as previously mentioned, especially in the field of medicine where decisions entail life or death, additional measurements of safety are required before deep learning can be applied in general practices.

Our contributions are the following. First, we highlight the limitations of variance-based uncertainty metrics through a motivating three-way classification problem. We continue by presenting a novel uncertainty metric, that is based on the overlap between output distributions, rather than their variance. We compare both metrics using benchmark datasets, namely MNIST, CIFAR-10 and ImageNet. Finally, we apply the proposed metric to the real-world use case of skin lesion classification.

The remainder of this paper is structured as follows. In Sect. 2, we start by giving background information regarding Bayesian neural networks and uncertainty metrics. Next, we highlight issues with existing metrics for output uncertainty and describe our proposed metric in Sect. 3. In Sect. 4, we give an overview of the algorithmic complexity of each step in calculating uncertainty using our metric. We compare our metric with the predictive variance in an empirical setting, evaluating output uncertainty for a series of benchmark datasets in Sect. 5. We continue this section by introducing suboptimal training conditions and evaluate their effect on the output uncertainty. We apply our metric to the problem of skin lesion classification in Sect. 6. In Sect. 7, we discuss related work. Section 8 recapitulates the main findings of our research.

2 Background

2.1 Bayesian neural networks and variational inference

Consider a neural network as a probabilistic model $p(y|x, w)$. For an input $x \in \mathbb{R}^d$, the network calculates a probability for each of the K possible outputs $y \in \mathcal{Y}$, using the weights w. For classification problems, $\mathcal{Y}$ is the set of classes, and $p(y|x, w)$ is a categorical distribution. Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$, the optimal weights $w^\star$ for the network can be learned by maximum likelihood estimation (MLE):

$$w^\star = \arg\max_w \log p(\mathcal{D}|w) \tag{1}$$

$$= \arg\max_w \sum_{i=1}^{n} \log p(y_i|x_i, w). \tag{2}$$

In a Bayesian neural network [31, 35], the weights of a neural network are no longer fixed values, but rather randomly drawn from a prior distribution $p(w)$. Instead of updating the weights directly during training, the parameters of the weight distributions are updated, by observing data $\mathcal{D}$. As such, the posterior distribution $p(w|\mathcal{D})$ is calculated. Using Bayes theorem, this gives

$$p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} \tag{3}$$

$$= \frac{p(\mathcal{D}|w)p(w)}{\int p(\mathcal{D}|w)p(w)\mathrm{d}w}. \tag{4}$$

Given the posterior, a predictive distribution on the output classes y^* can be calculated for an unseen input x^* as

$$p(y^*|x^*, \mathcal{D}) = \int p(y^*|x^*, w)p(w|\mathcal{D})\mathrm{d}w. \tag{5}$$

Unfortunately, the posterior cannot be calculated analytically, due to the intractable integral in the denominator. Variational inference [20] sidesteps this issue, by approximating the true posterior $p(w|\mathcal{D})$ with a variational distribution $q_\theta(w)$, parameterized by θ, such that it is most similar to $p(w|\mathcal{D})$. This is done by minimizing the Kullback–Leibler (KL) divergence between the true posterior distribution $p(w|\mathcal{D})$ and the variational distribution $q_\theta(w)$. Therefore, the optimal parameters $\theta^\star$ for the variational distribution are defined as

$$\theta^\star = \arg\min_\theta \mathrm{KL}[q_\theta(w)||p(w|\mathcal{D})] \tag{6}$$

$$= \arg\min_\theta \mathrm{KL}[q_\theta(w)||p(w)] - \mathbb{E}_q[\log p(\mathcal{D}|w)] + \log p(\mathcal{D}), \tag{7}$$

where

$$\mathrm{KL}[q_\theta(w)||p(w)] = \int q_\theta(w) \log \frac{q_\theta(w)}{p(w)}\mathrm{d}w. \tag{8}$$

The resulting cost function is widely known as the variational free energy or the negative evidence lower bound (ELBO):

$$\mathcal{F}(\mathcal{D}, \theta) = \mathrm{KL}[q_\theta(w)||p(w)] - \mathbb{E}_q[\log p(\mathcal{D}|w)]. \tag{9}$$

2.2 Monte Carlo dropout

Dropout [44] is a regularization technique, commonly used to reduce overfitting in neural networks. With dropout the units in a neural network are randomly set to 0 with a probability d. By dropping a different set of units at each training step, dropout training can be seen as the equivalent of training a large ensemble of neural networks. At test time, no units are dropped. They are rather multiplied by $1 - d$. The expected output magnitude at training time is thereby ensured to be the same as the output magnitude at test time.

[10] relates dropout training to variational inference by approximating $p(w|\mathcal{D})$ with a variational distribution $q_\theta(w)$. As such, arriving from Eq. (5), the predictive output distribution given an unseen input x^* is approximated as:

$$p(y^*|x^*, \mathcal{D}) \approx \int p(y^*|x^*, w)q_\theta(w)\mathrm{d}w. \tag{10}$$

Gal and Ghahramani further show that sampling weights from $q(w)$ are mathematically equivalent to applying dropout on the neural network weights. Therefore, the integral in Eq. (10) can further be approximated by taking Monte Carlo samples with different dropout masks:

$$\int p(y^*|x^*, w)q_\theta(w)\mathrm{d}w \approx \frac{1}{T}\sum_{t=1}^{T} p(y^*|x^*, \hat{w}_t) \tag{11}$$

in which $\hat{w}_t \sim q_\theta(w)$. To summarize, calculating the predictive output distribution for a given input boils down to performing T stochastic forward passes through the network at inference time. At each forward pass, a different dropout mask is applied on the network weights. The outputs of every forward pass are then averaged to arrive at the final predictive distribution.

2.3 Bayes by backprop

The KL divergence in Eq. (9) also contains an intractable integral. We therefore apply a variance reduction technique known as random numbers. By sampling weights $\hat{w}_t$ from the variational distribution $q_\theta(w)$ we then arrive at the following approximation:

$$\mathcal{F}(\mathcal{D}, \theta) \approx \frac{1}{T}\sum_{i=1}^{T} \log q_\theta(\hat{w}_t) - \log p(\hat{w}_t) - \log p(\mathcal{D} \mid \hat{w}_t) \tag{12}$$

where $\hat{w}_t$ denotes the Monte Carlo weight sample drawn from $q_\theta(w)$. The result is a tractable cost function that can be optimized w.r.t. the variational parameters θ.

2.4 Uncertainty

While there can be many sources of uncertainty, in the context of modelling it is convenient to categorize the uncertainty type as either aleatoric or epistemic [7]. The first type, aleatoric uncertainty, captures the random noise intrinsic to the observations. This type of uncertainty cannot be reduced by collecting additional data. The second type, epistemic uncertainty, accounts for the uncertainty in the model parameters. Gathering more data can improve upon the epistemic uncertainty.

In a Bayesian neural network, both types of uncertainty can be calculated. For an unseen input x^*, the output distribution over the classes y^* is given by Eq. (5). Using variational inference, this distribution is approximated by replacing the true posterior $p(w|\mathcal{D})$ with the variational distribution $q_\theta(w)$ as explained in Sect. 2.1:

$$p(y^*|x^*, \mathcal{D}) \approx \int p(y^*|x^*, w) q_\theta(w) \mathrm{d}w = \mathbb{E}_{q_\theta(w)}[p(y^*|x^*, \mathcal{D})]. \tag{13}$$

An unbiased estimator of the output distribution is then constructed by sampling weights $\hat{w}_t$ from $q_\theta(w)$:

$$\mathbb{E}_{q_\theta(w)}[p(y^*|x^*, \mathcal{D})] \approx \frac{1}{T} \sum_{t=1}^{T} p(y^*|x^*, \hat{w}_t). \tag{14}$$

Based on the estimator in Eq. (14), the uncertainty on a specific output class y_k^* ($k \in \{1, 2, \ldots, K\}$) can be calculated using the definition of variance [42] as

$$\begin{aligned} \mathrm{Var}_{q_\theta(w)}[p(y_k^*|x^*, \mathcal{D})] = \mathbb{E}_{q_\theta(w)}[p(y_k^*|x^*, \mathcal{D})^2] \\ - \mathbb{E}_{q_\theta(w)}[p(y_k^*|x^*, \mathcal{D})]^2. \end{aligned} \tag{15}$$

This is the variance on the sampled output probabilities for output class k. The overall output uncertainty is then averaged across all individual variance measurements for the K output classes [21]. We call this quantity the *predictive variance*.

3 A metric for output uncertainty

Consider the following intuitive example. A Bayesian neural network is trained on a three-way classification problem ($K = 3$). For 2 different data examples, Fig. 1 (top) shows the distribution of the Monte Carlo samples for each of the three output classes. While both examples would be classified as "green", it is important to notice the difference in uncertainty between these examples. In the first example, on the left, there is almost no overlap between the top-2 classes, respectively, "green" and "orange". Hence, we would argue that, for this example, the "green" class is, in fact, the correct class. In the second example, on the right, we see the opposite. The distributions for the top-2 classes are highly overlapping. Consequently, we cannot say for sure that the assigned "green" class is correct.

Contrary to this intuition, a variance-based metric such as in Eq. (15), would assign a high uncertainty to the first example and a low uncertainty to the second example. Therefore, we aim for a different approach to quantify the output uncertainty.

A naive approach would be to look at the difference between the class output probabilities, i.e. the softmax output. A metric based on these differences assigns an uncertainty value that is inversely proportional to the difference between the class probabilities. This has the advantage that it is applicable to all neural network architectures, including traditional networks with deterministic outputs. However, different network weights can result in very different network outputs. Figure 1 (bottom) shows two sampled softmax outputs for each of the examples above. We see that, especially for the example on the left, the class probabilities can be very close, as well as far apart. This is unwanted behaviour for a robust uncertainty metric. We therefore propose a metric that is based on the overlap between the Monte Carlo distributions for each class.

In a first step, we draw inspiration for the estimator in Eq. (14) to calculate sample distributions for each of the K classes. More specifically, we draw T output samples for each class which together constitute the sample distribution. We identify these distributions as $d_1, \ldots, d_K$, where d_k represents the sample distribution with the k'th highest mean. Next, we estimate the overlap between the distribution with the highest mean and all others, using the normalized Bhattacharyya coefficient [3]. This is done by constructing a histogram h_1 from d_1, and a histogram h_k

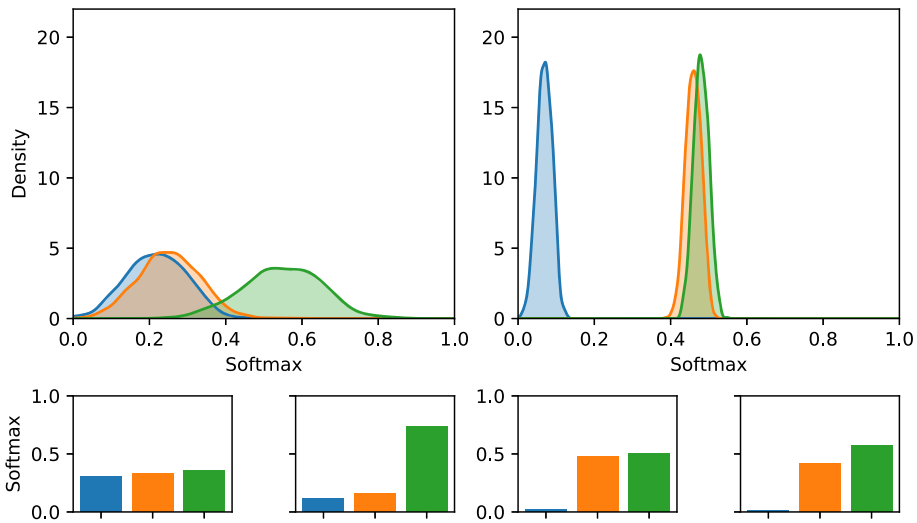

Fig. 1 **Top**: two examples of the predictive output distribution for each class in a three-way classification problem. **Bottom**: for each example, two softmax outputs are sampled from the respective predictive distributions

from d_k, where $k > 1$, both with n bins. The normalized Bhattacharyya coefficient is then given by:

$$BC(h_1, h_k) = \frac{1}{T} \sum_{\ell=1}^{n} \sqrt{h_{1\ell} h_{k\ell}} \tag{16}$$

in which $h_{1\ell}$ and $h_{k\ell}$ are the number of samples in the ℓ-th bin of, respectively, histograms h_1 and h_k. The result is a value between 0 and 1, where a 0 indicates no overlap between the histograms, while a 1 indicates a perfect overlap (i.e. the histograms are identical). Finally, we use these values to define the output uncertainty as

$$U_{\text{output}} = \max_{k>1} BC(h_1, h_k), \tag{17}$$

where a high value indicates a high uncertainty, and a low value indicates a low uncertainty. In practice, we find that only considering the overlap between the output distributions for the top-2 classes yields a tight lower bound for the uncertainty quantity:

$$U_{\text{output}} \approx BC(h_1, h_2), \tag{18}$$

as we show in our experimental section. We refer to $BC(h_1, h_2)$ as the *BC uncertainty*. An example of how to calculate the BC uncertainty, given a dropout neural network, is shown in Fig. 2.

The BC uncertainty has two hyperparameters, namely the number of output samples T that are drawn for a given class (i.e. the number of times an input image is passed through the neural network), and the number of bins n used when constructing the histograms. These hyperparameters are not independent: the T output samples for a given class are divided over the n bins. Values for T and n should be chosen with this in mind. Enough samples T should be drawn, so that the sample distributions properly approximate the true distributions over the output classes. The probability scores outputted by the softmax function are continuous values in the interval of [0, 1]. This interval will be divided into n equally large subintervals when constructing the histograms. Therefore, the range for the BC uncertainty directly relates to n. In case $n = 1$, all BC uncertainties will be equal to 1, since there is perfect overlap between the two histograms. As n grows bigger, in the limit, $\lim_{n\to\infty} BC(h_1, h_2) = 0$. In this case, assuming no output samples are exactly the same, the subintervals are too small for there to be any overlap. In our experiments, we have chosen the value of 100 for both T and n.

We provide an illustrative example, highlighting the strength of our approach. Figure 3 shows an image of a measuring cup that is wrongly classified. Next to this image, the figure shows the corresponding output histograms of the four classes with the highest sample variance. Empirically, we see that each of these histograms still have a small variance, indicating that the overall predictive variance will be low as well, close to zero. But ideally, a wrong classification comes with a high uncertainty. On the other hand, because of the high overlap between the top-2 classes—which are highlighted in bold—the BC uncertainty for this image is 0.88, which is significantly higher than the predictive variance. Indeed, when inspecting the image and the labels, the confusion makes sense.

The given example shows the advantage of our approach over variance-based methods. While the sample variance may be low, there could very well still be a high class overlap. In this case, the BC uncertainty is better suited. When the resulting Monte Carlo distributions have a high variance, as well as a high overlap, both metrics will result in a high uncertainty. In a similar fashion, when there is a low sample variance, and little to no overlap, both metrics will result in a low uncertainty.

Fig. 2 An overview of how to calculate the BC uncertainty for a given example, for the case of a neural network trained with dropout. The example is passed T times through the network. At each pass, a different dropout mask is sampled from the dropout distribution, resulting in T softmax outputs. From these outputs, we construct a histogram for the top-2 classes, having the highest and second to highest mean. Using these histograms, we calculate the BC uncertainty

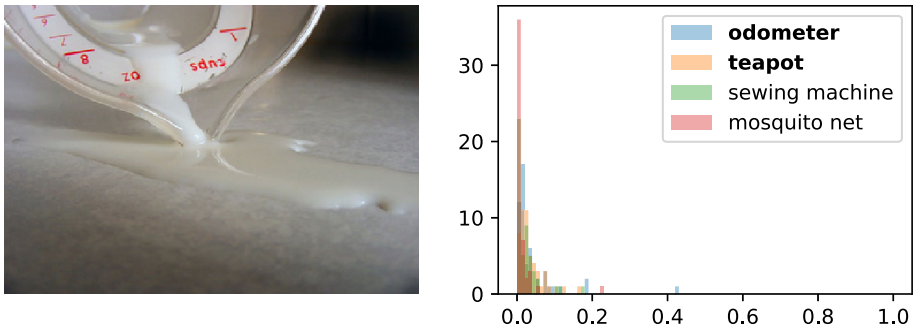

Fig. 3 **Left**: an image of a measuring cup (from the ImageNet test set). **Right**: the output histograms for the image on the left, for the 4 classes with the highest sample variance. A bold label indicates that this class is also in the top-2 classes with the highest mean

4 Algorithmic complexity

In this section, we reiterate on the steps required to calculate the BC uncertainty for a given classification output, listing all the corresponding algorithmic complexities. Note that we do not take the complexity of the forward pass of a neural network into account.

Given a classification problem with K output classes, forwarding T duplicates of an input image through a stochastic neural network will result in K sample distributions (one for each class), each containing T probability scores. Calculating the mean for each distribution has a complexity of $\mathcal{O}(K * T)$. Next, we identify the top-2 distributions, having the highest and second to highest mean, with a complexity of $\mathcal{O}(K)$. Using these two (continuous) distributions, we construct two discrete histograms, by dividing the T probability scores in each distribution over n bins. This process also has a complexity of $\mathcal{O}(T)$. Finally, we use the Bhattacharyya coefficient to calculate the overlap between both histograms, by comparing the number of samples in corresponding bins, with a complexity of $\mathcal{O}(n)$. This results in a total complexity of $\mathcal{O}(K * T + n)$. It is important to note that this complexity pales in comparison with the algorithmic complexity of the forward pass of a neural network.

5 Experiments

We evaluate our proposed uncertainty metric, based on the Bhattacharyya coefficient, using different benchmark datasets. For each of these datasets, we train a deep dropout neural network until convergence. During evaluation, we leave dropout enabled, using the same dropout rate as during training. It is important to note that our goal is not to reach state-of-the-art classification performance on these datasets. We rather aim to train a sufficiently powerful model, enabling us to empirically validate the output uncertainty.

MNIST The MNIST database of handwritten digits [26] consists of 60,000 train samples and 10,000 test samples. Each sample is comprised of a 28 by 28 pixels greyscale image of a single handwritten digit, along with its class label (0 through 9).

The network architecture for this dataset consists of two convolutional layers with 3×3 kernels, having, respectively, 16 and 32 filters, followed by two fully connected layers with, respectively, 128 and 10 units. All hidden layers have rectified linear unit (ReLU) activations[1]. We apply dropout with a probability of 0.6 after all activations.

CIFAR-10 The CIFAR-10 dataset [23] consists of 50,000 train samples and 10,000 test samples. These samples contain a 32 by 32 pixels colour image and are evenly distributed among ten output classes.

For the CIFAR-10 dataset, we train a deep neural network architecture consisting of six convolutional layers with 3×3 kernels (32, 32, 64, 64, 128 and 128 filters), followed by three fully connected layers (128, 64 and 10 units). Again, all hidden layers have in contrast to the MNIST activations. In contrast to the MNIST architecture, only the fully connected hidden layers are followed by dropout, also with a probability of 0.6.

ImageNet The train set used for the ImageNet Large Scale Visual Recognition Challenge 2012 [40] includes over one million high-resolution images, spread among 1,000 classes. The accompanying test set contains 50,000 images. Both the test set and the train set have a uniform class distribution.

For the experiments using the ImageNet dataset, we use a pre-trained Inception v1 model [45], where we set the dropout rate to 0.6.

We report the test set accuracy our networks achieve for each of the datasets in Table 1.

[1] More recent activation functions, such as the scaled polynomial constant unit activation function (SPOCU) [22], could potentially result in a higher classification accuracy. However, for the application of uncertainty quantification, we found that ReLU activation function achieves similar results.

Table 1 Test set accuracy for the different datasets

Dataset	Accuracy
MNIST	0.98
CIFAR-10	0.81
ImageNet	0.71

Predictions are made by taking 100 stochastic forward passes and averaging the result.

5.1 Approximating output uncertainty

In a first experiment, we test whether the approximation of the output uncertainty using only the top-2 classes is justified. We calculate the overlap between the sample histograms for the top-1 class output, and all other outputs, using the Bhattacharyya coefficient defined in Eq. (16), for both the MNIST test set and the CIFAR-10 test set. We limit ourselves to these datasets due to computational constraints. Coefficients are calculated by forwarding each test example 100 times through the network ($T = 100$). We show a violin plot of the difference $U_{\text{output}} - BC(h_1, h_2)$ for both datasets. We limit ourselves to the cases for which $U_{\text{output}} > BC(h_1, h_2)$, i.e. there exists a class $k > 2$ that yields a larger Bhattacharyya coefficient compared to class 1. We find that this is only the case for less than 10% of the test sets. For both datasets we see the largest mass close to 0, with a mean of 0.016 for MNIST and a mean of 0.019 for CIFAR-10. As a result, in most cases U_{output} is only marginally larger than $BC(h_1, h_2)$. We can therefore state that $BC(h_1, h_2)$, i.e. the BC uncertainty, is a justified approximation for the output uncertainty, greatly reducing the number of operations required for calculation. We will therefore use the BC uncertainty in the remainder of our experiments. As an illustration, below the violin plot, we show two examples where $U_{\text{output}} > BC(h_1, h_2)$ for each dataset, accompanied by h_1 (green), h_2 (orange) and $h_{\max}$ (blue) as given by

$$h_{\max} = \arg\max_{h_k} BC(h_1, h_k). \tag{19}$$

For all examples, we see indeed a large overlap between h_2 and $h_{\max}$, resulting in a negligible difference between U_{output} and $BC(h_1, h_2)$ (Fig. 4).

5.2 Uncertainty vs. accuracy

We generate a prediction for each example in the test set, by forwarding the example 100 times through the network ($T = 100$). Additionally, using the obtained sample distribution, we calculate both the predictive variance and the BC uncertainty. Figure 5 shows the histogram of the classification accuracy as a function of the predictive variance (top), and the BC uncertainty (bottom), for each of the datasets. We expect the overall accuracy to be higher, when the network has a lower uncertainty, i.e. the network is confident in its prediction. We see, indeed, that this holds for both metrics. However, we argue that this relation is more pronounced for the BC uncertainty, especially for ImageNet. In addition, for the predictive variance, the histogram boundaries differ for every dataset, rendering the predictive variance incomparable across datasets. Combined with the overall small absolute magnitudes, this makes the predictive variance hard to interpret.

5.3 Uncertainty vs. task difficulty

As is indicated in the previous subsection, the range of the predictive variance is different for each dataset. This has the consequence that absolute numbers cannot be applied to estimate task difficulty.

Figure 6 shows the distribution of the predictive variance (top) and the BC uncertainty (bottom) for the three datasets. As shown in Table 1, classifying MNIST digits is easier than classifying CIFAR-10 images, while ImageNet is the hardest to classify. This increase in difficulty is only partly reflected by the predictive variance, where we see both the mean and the spread of the CIFAR-10 uncertainty distribution increase, compared to MNIST. The distribution for the ImageNet dataset, however, has an average uncertainty close to 0, as well as a very small spread. This can be explained by the number of classes considered in each dataset. ImageNet contains 1000 classes compared to 10 classes for MNIST and CIFAR-10, i.e. two orders of magnitude larger. This means that for a given input example a large number of class probabilities will be close to zero, resulting in small individual class variances. Averaging all class variances indeed results in an overall small predictive variance.

On the other hand, the BC uncertainty has the advantage that it only looks at the overlap between the top-2 classes. It is therefore invariant to the number of classes. Besides this, the BC uncertainty is bounded between 0 and 1. These properties allow us to apply BC uncertainty to estimate the task difficulty in absolute terms. This is clearly illustrated in Fig. 6 in which we see that, indeed, the overall BC uncertainty increases as the task gets more difficult.

5.4 Uncertainty vs. prediction

In the previous subsections we have highlighted the benefits of the BC uncertainty in comparison with the predictive variance. Subsequently, we further evaluate the robustness of our proposed technique.

To begin, we apply the Bhattacharyya coefficient to gain insight into the decision process of a neural network. Figure 7 gives four example images, randomly sampled

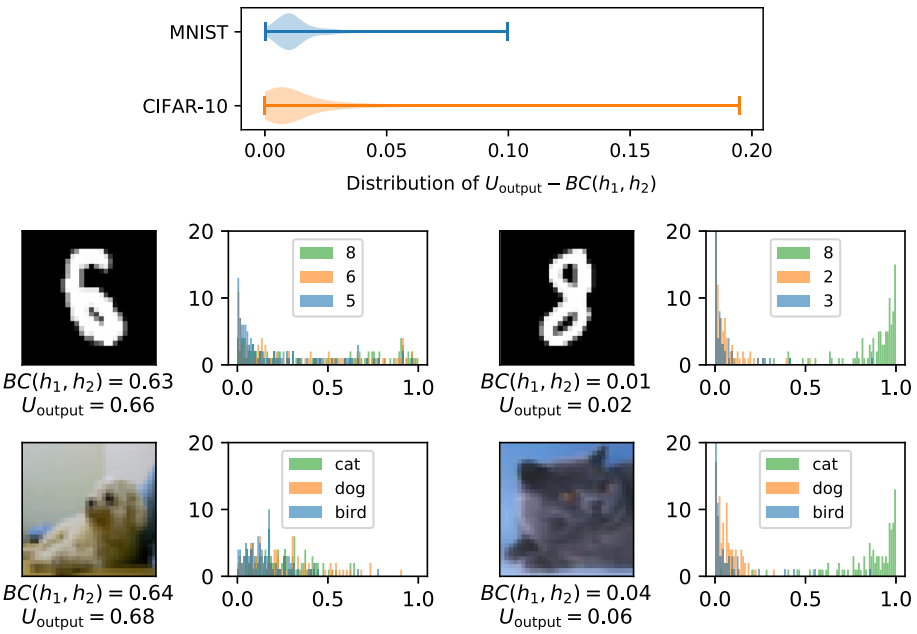

Fig. 4 **Top**: Violin plots for the difference between U_{output} and $BC(h_1, h_2)$, both for MNIST and CIFAR-10. The whiskers specify the minimum and maximum values. For both datasets, the largest mass lies close to 0. **Bottom**: MNIST and CIFAR-10 examples where U_{output} is greater than $BC(h_1, h_2)$, for the case of high uncertainty (left), as well as the case of low uncertainty (right), with their corresponding output histograms (green: h_1, orange: h_2, blue: h_{max}). For all examples, there is a great overlap between h_{max} and h_2, resulting in comparable Bhattacharyya coefficients

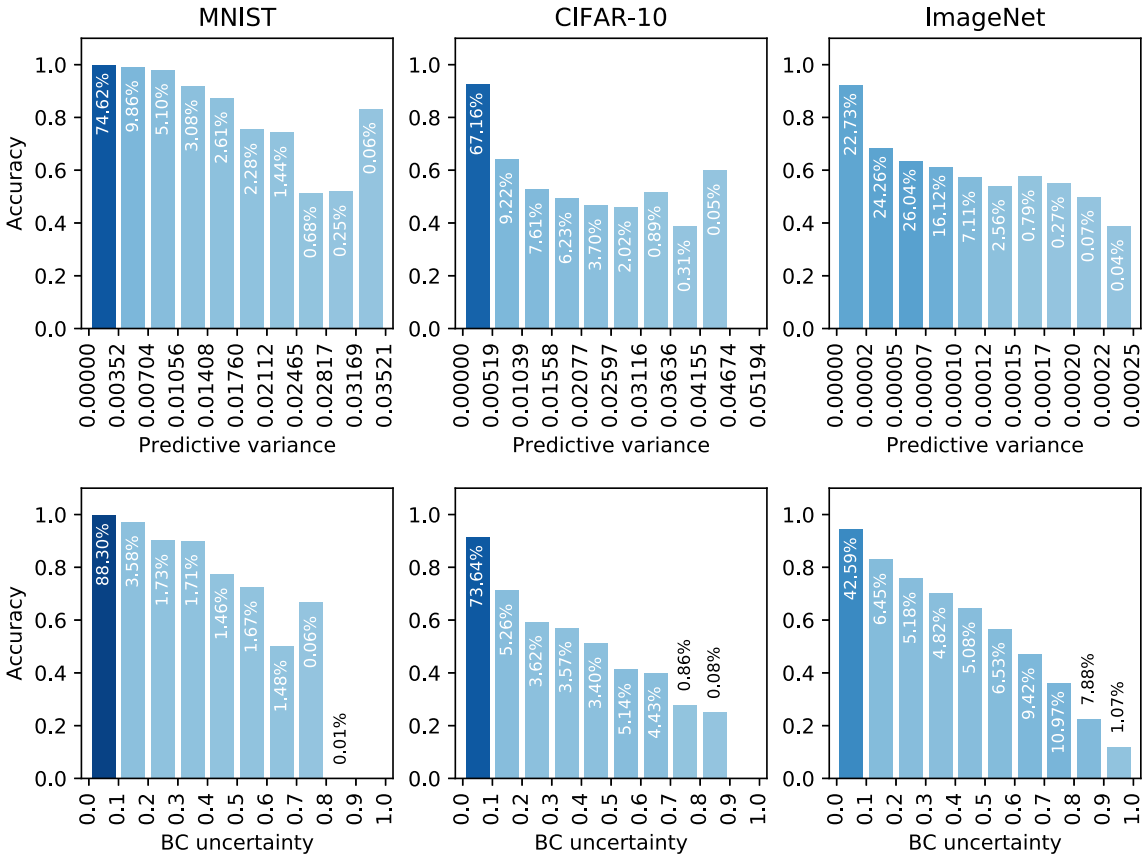

Fig. 5 Test set accuracy as a function of the output uncertainty. For the top row, grouping is done by means of the predictive variance. For the bottom row, BC uncertainty is applied. The labels on the X-axis specify the range of the respective uncertainty metric for each group. For the predictive variance, histogram boundaries were chosen based on the minimum and maximum values. The percentage inside each bar indicates the fraction of the test set contained in the respective group. This is also reflected by the colour of the bar. The darker the bar, the more test set images its respective group contains

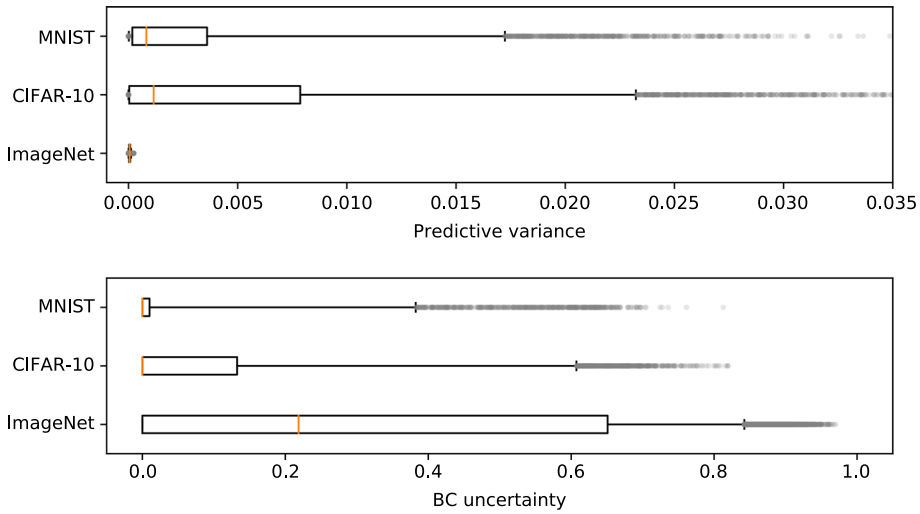

Fig. 6 Distribution of the output uncertainty, for the MNIST, CIFAR-10 and ImageNet test set. **Top**: predictive variance. **Bottom**: BC uncertainty. Whiskers represent the 5% and 95% interval. The transparent dots are uncertainty values outside of this interval

from the three datasets, for each of the following specific criteria. The first set of images at the top are images that are correctly classified and have a BC uncertainty equal to 0. These images all share similar traits: the target is centred in the image, with little to no distraction. Such images can be considered as "easy to classify". The second set of images, in the centre, are images that have a high BC uncertainty. We consider both correctly and incorrectly classified images. In comparison with the previous set, images in this set have a higher degree of noise. In some cases, such as the "3" that is mistaken for a "8", the image indeed closely resembles the predicted class. Therefore, images like these are considered "more difficult to classify". In a real-world setting, images matching this criterion would require additional evaluation. Lastly, at the bottom, we show a set of images that are wrongly classified, but have a BC uncertainty that is also 0. Images like these are the most troublesome, since they have a high chance of going unnoticed, while they should, in fact, require further assessment. We highlight two specific cases among these images. In the first case, the target closely resembles the predicted class. Examples are the "3", that is classified as a "8", and the "airplane", that is mistaken for a "bird". This case is the most common. For some exceptions, it is actually the ground truth label which is wrong, as is the case for the "gong".

5.5 Suboptimal training conditions

The datasets used in these experiments, MNIST, CIFAR-10 and ImageNet, have a number of properties in common. Not only do they all contain a large amount of images, they are also (almost) perfectly balanced. Unfortunately, in a real-world setting, this is hardly ever the case. Most realistic datasets are usually small in size and/or suffer from class imbalance. We therefore evaluate the performance of the BC uncertainty metric under such conditions.

To simulate a small train set size, we use only a fraction of the full train set to train a dropout neural network. We do this for MNIST and CIFAR-10. The considered fractions are 0.01, 0.05, 0.1, 0.2, 0.4, 0.8 and 1.0 of the original train set. In a similar fashion we introduce an artificial class imbalance in the train set. This time, instead of down-sampling all classes, we take only a fraction of the available images for a specific class. In addition we include a fraction of 0.0, specifying the edge case of unseen data. We do this for the digit "3", and the class "cat", for MNIST and CIFAR-10, respectively. We repeat this experiment using a Bayesian neural network in Appendix A.

Results are presented in Fig. 8. This figure shows the evolution of the BC uncertainty, either as a function of the train set fraction (top) or as a function of the under-sampled class fraction (bottom). In the latter case, only predictions and uncertainties for the specific class are included. For each scenario, the distribution of the BC uncertainty is shown separately for correct and incorrect classifications. The classification accuracy for each scenario is written above the respective distributions. We treat both cases individually.

Train set size As expected, the accuracy increases as the train set size increases. Regarding the BC uncertainty, we make the distinction between the correctly and incorrectly classified images. The distribution for the correctly classified images starts out with a high mean, and a large spread. Both of these decrease significantly as the train set size increases. In contrast, the mean uncertainty for the incorrectly classified images slightly decreases, initially, after which it seems to stabilize. We also observe that for all train set sizes the mean BC uncertainty for the correct cases is always lower than the mean BC uncertainty for the

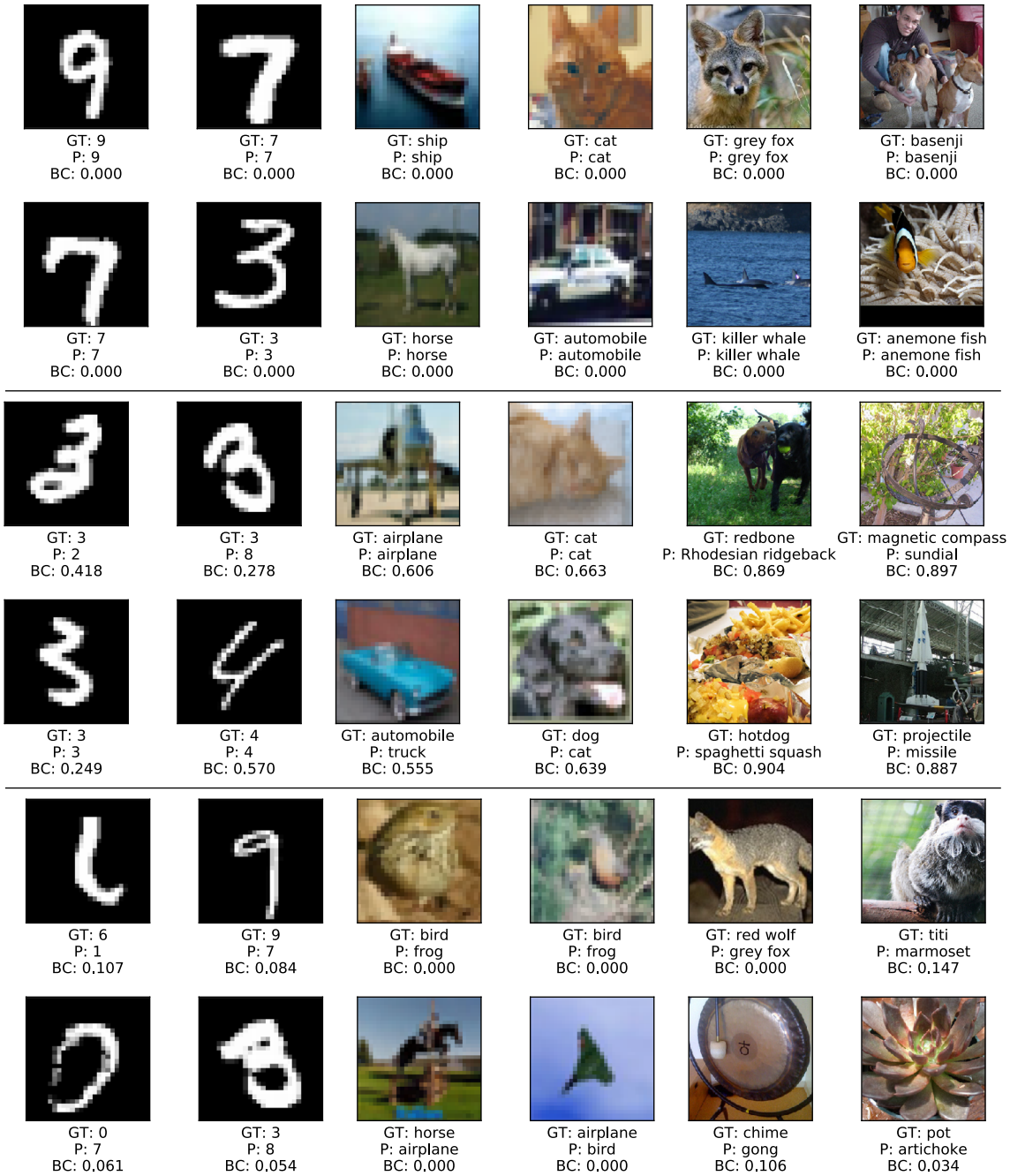

Fig. 7 Example images from the MNIST, CIFAR-10 and ImageNet datasets, sampled according to the following criteria. **Top**: correctly classified images, with a low BC uncertainty. **Centre**: images with a high BC uncertainty. **Bottom**: wrongly classified images that are assigned a low BC uncertainty

incorrect cases. As the train set size increases, this observation becomes more pronounced.

When insufficient training data are available, a neural network fails to learn a mapping from images to their respective class labels. This results in a classification accuracy close to random guessing. This is the case when attempting to train a CIFAR-10 classifier using only 1% or 5% of the train set. In these cases, the corresponding BC uncertainties are very high (close to 1.0), highlighting the low confidence of the network.

Simulated class imbalance Similar observations can be made with respect to a simulated class imbalance, although only when the under-sampled class is sufficiently present in the train set (starting from 5% for MNIST, and from 40% from CIFAR-10). When too little examples of a under-sampled class are present, the neural network maps these

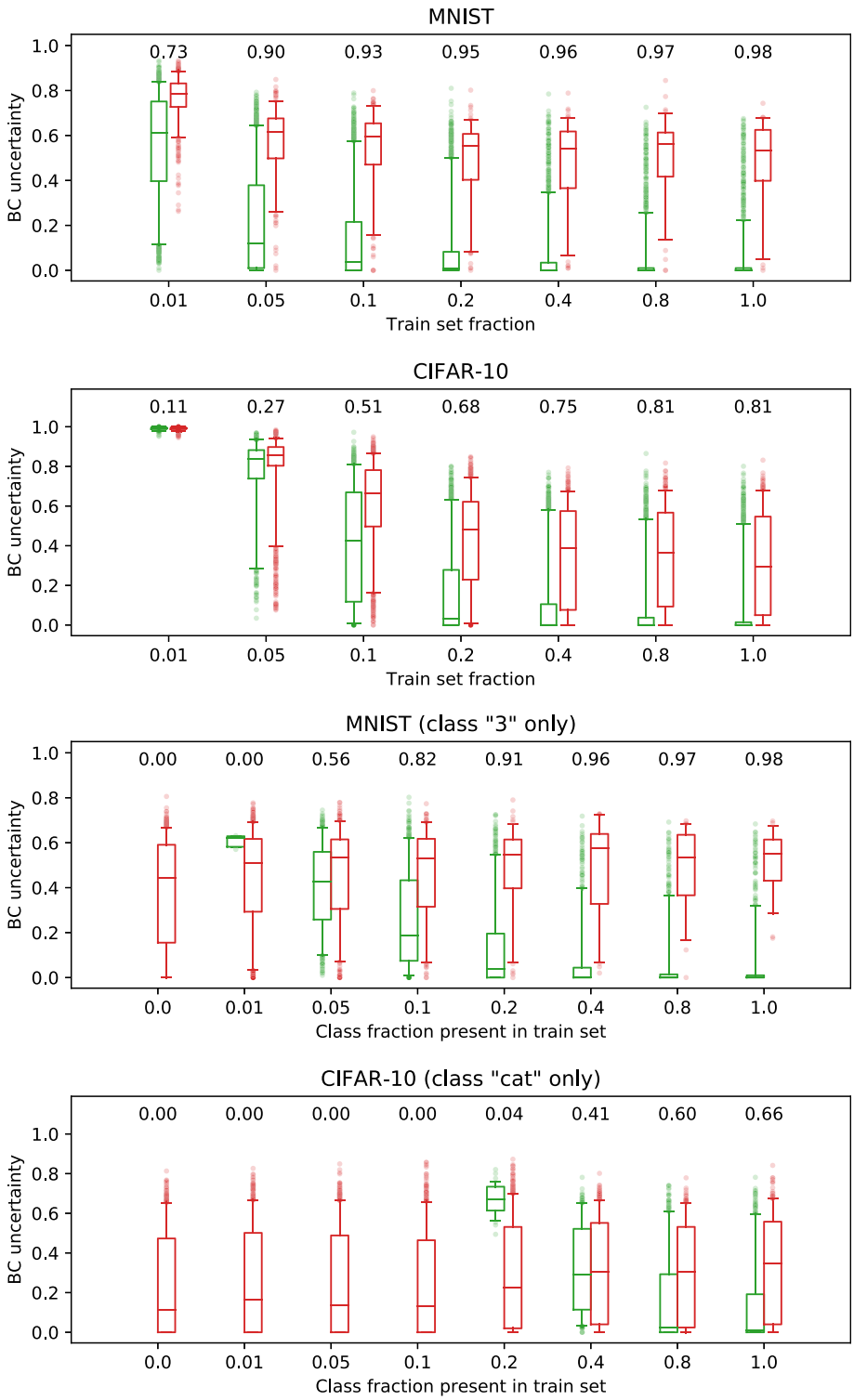

Fig. 8 Distribution of the BC uncertainty for different training scenarios, both for MNIST and CIFAR-10. Per dataset, either only a fraction of the full train set is used (top), or only a fraction of the available examples for a single class are included for training (bottom). For each scenario, the distribution for the correct classifications is shown in the left (green) box plot, and the distribution for the misclassifications is shown in the right (red) box plot. The whiskers represent the 5% and 95% interval. The transparent dots are uncertainty values outside of this interval. The classification accuracy is given by the number above each box plot pair

examples to an other, most similar class. This highlights the importance of constructing a train set in which each class is properly represented. This indicates that further research is required when applying the BC uncertainty as an outlier detector.

6 Use case: skin lesion classification

We apply our BC metric to a real-world problem: skin lesion classification. For this, we use the HAM10000 dataset of common pigmented skin lesions [47]. This dataset is relatively small in size, containing only 10,015

dermoscopic images of skin lesions. The dataset is also heavily imbalanced, a common problem among medical datasets [5, 12, 46]. About 67% of all images belong to "nevi" class, while only 1% of images are instances of "dermatofibroma". Overall, seven types of skin lesions are present in this dataset.

We randomly split the full dataset into a train set, containing 9,013 images (approximately 90% of all images), and a validation and test set, both containing 501 images (approximately 5%). We use the train and validation set to train a deep neural network architecture and to find optimal hyperparameters. Given the limited amount of available training data, we opt for transfer learning. We use a ResNet50 network [17] which was pre-trained on ImageNet as the basis for our classifier. We replace the final layer with a custom head that we train for skin lesion classification. The head is constructed of two fully connected hidden layers, respectively, having 512 and 64 units, followed by a fully connected output layer with 7 units. Both hidden layers have ReLU activations. In order to cast this network as a Bayesian neural network, we apply dropout after every hidden layer, with a probability of 0.6.

We obtain predictions as well as BC uncertainties for the hold-out test set by forwarding each of the test images $T = 100$ times through the network. This way, we achieve a classification accuracy of 0.82, matching the performance of a similar network architecture that is evaluated without dropout. We want to stress again that it is not our intention to achieve the overall best state-of-the-art performance on this dataset, but that we rather use the trained neural networks to evaluate uncertainty metrics.

6.1 Uncertainty vs. accuracy

First, we validate the BC uncertainty metric on the skin lesion classification use case by repeating the experiment from Sect. 5.2. We calculate the BC uncertainty for each image in the test set. In Fig. 9, we plot the histogram of the classification accuracy as a function of the BC uncertainty. About 72% of the images are classified with an uncertainty lower than 0.2. Within this group, most images (ca. 55% of the entire test set) have a very low uncertainty (BC = 0.0), with a resulting accuracy of 0.97. Furthermore, we see again that the accuracy is inversely proportional to the BC uncertainty. A lower BC uncertainty yields a higher accuracy, and vice versa.

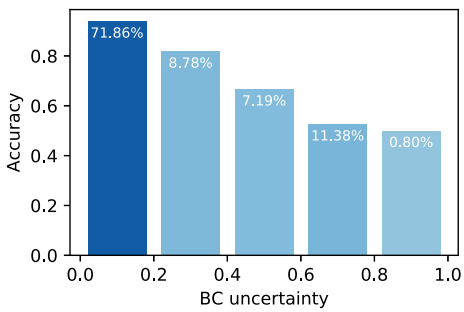

Fig. 9 The accuracy when binning the test set images according to their BC uncertainty. The labels on the X-axis specify the range. The amount of images (as a fraction) for each group is specified inside the bar. This is also reflected by the colour of the bar. The darker the colour, the higher the amount of images in the respective group

6.2 The (un)certain cases

We inspect some of the classified images. In Fig. 10, we show four images with their respective sample histograms for each class. The top-2 histograms, used to calculate the BC uncertainty, are coloured green and orange, respectively. In the examples in the first two rows, we notice that there is a large overlap between the top-2 histograms, resulting in a high BC uncertainty. The bottom two images have a BC uncertainty equal to 0.0. These are clear examples of nevi, which are, overall, fairly easy to classify.

6.3 The (un)certain classes

Finally, Fig. 11 (top) shows the BC uncertainty distributions on a per-class basis. The mean and spread are especially low for the "nevus" class. As mentioned before, on average, most of the nevi are easily classified by a trained professional. Additionally, the nevi are the most abundant type of skin lesion in the dataset. This shows that the classifier is most confident on the class of which it encountered the most examples during training.

Besides a ground truth label for each image, the HAM10000 dataset also contains additional meta-data regarding the confirm type of the diagnosis. All skin lesions were diagnosed in one of four ways. These are, in ascending order of rigorousness: "single image expert consensus", "serial imaging showing no change", "confocal microscopy with consensus dermoscopy", and "histopathology"; the latter type indicates that the lesion had to be surgically removed due to the assessed risk. Images that were labelled in a more rigorous way are overall more difficult cases, especially when the final diagnosis is benign. Therefore, a nevus that has the confirm type "histopathology" is usually more ambiguous, or difficult to classify, than a nevus confirmed by "single image consensus". Figure 11 (bottom) shows the distribution of the BC uncertainty for the images of nevi, for each of the diagnosis confirm types. We see that, indeed, the distribution for the "histopathology" confirm type has a large spread, in comparison with the other confirm types.

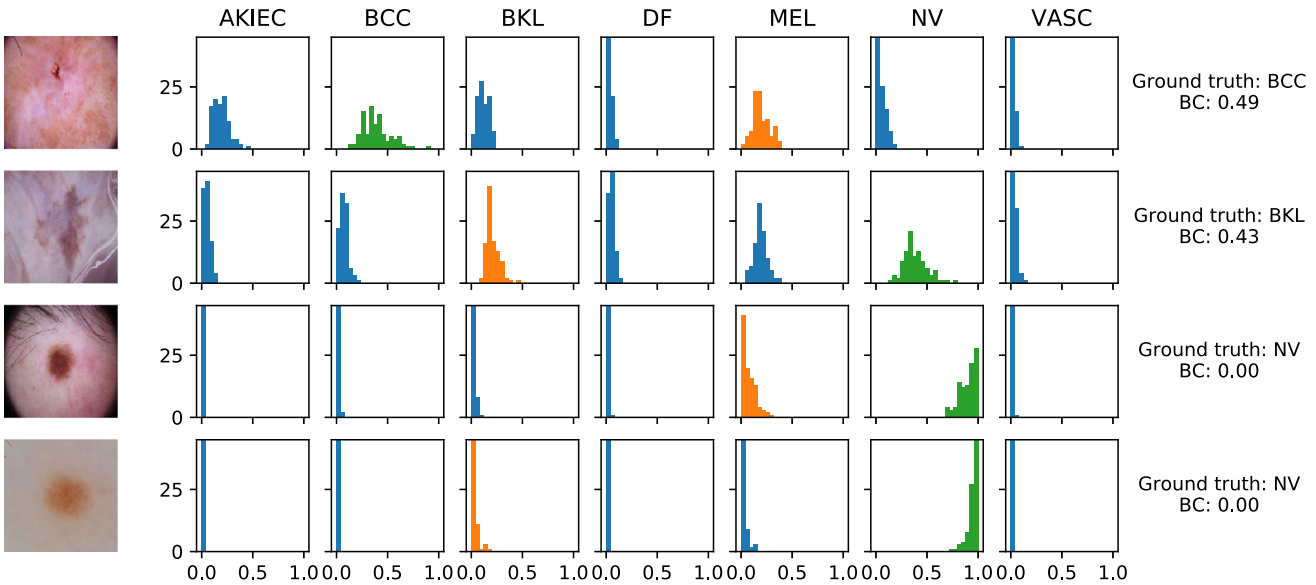

Fig. 10 Examples of skin lesions, and their corresponding output histograms, as well as the BC uncertainty. The two images at the top have a high uncertainty, while those at the bottom have a low uncertainty. The histograms with the highest and second to highest mean are coloured green and orange, respectively

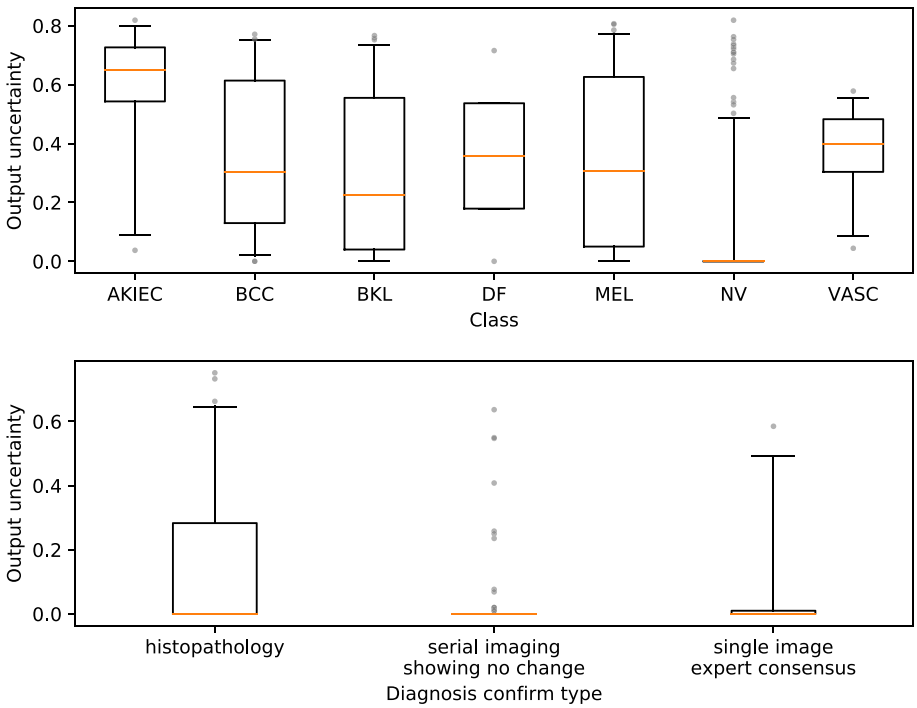

Fig. 11 **Top**: distribution of the BC uncertainty for each of the seven classes in the test set. Notably, the most present class (nevus, NV) has the lowest overall uncertainty. **Bottom**: distribution of the BC uncertainty for the "nevus" class, as a function of the diagnosis confirm type. The whiskers represent the 5% and 95% interval. The transparent dots are values outside of this interval

7 Related work

Bayesian neural networks The last decade saw a resurgence of Bayesian methods for neural networks. It started with [15] who introduced Monte Carlo variational inference (MCVI), a practical and scalable variational inference scheme for neural networks [4] expanded on this work by introducing Bayes by Backprop, an efficient backpropagation-compatible algorithm for learning a probability distribution over the weights of a neural network. They applied their algorithm to successfully train a Bayesian feed-forward neural network on the MNIST dataset [42] further extended this approach and presented how Bayes by Backprop can be applied to convolutional neural networks (CNNs). Their Bayesian CNNs attain similar performance to their frequentist counterparts on the

MNIST dataset, as well as the CIFAR-10 and CIFAR-100 datasets. Around the same time, [37] proposed a similar approach [30] introduced a variational distribution that, in contrast to [4], does not treat each of the weights of the neural network independently. Instead, they used a matrix variate Gaussian distribution [16], treating the weight matrix as a whole.

Alternatively, [10] provided proof that dropout training can be interpreted as an approximation of variational inference. In follow-up work, the authors evaluated their technique on both the MNIST dataset and the CIFAR-10 dataset [9]. Because of its ease in implementation, this method has seen a widespread use.

Quantifying uncertainty Variance-based uncertainty quantification using dropout networks is applied to both detection tasks [28, 36, 50] and segmentation tasks [21, 24, 34, 41, 48]. While this metric is most common, alternatives to quantify uncertainty exist, such as the predictive entropy or mutual information [11, 18]. These metrics are, however, unconstrained to a certain interval, making it more challenging to interpret the uncertainty values.

Although we have only described our BC uncertainty metric in relation to MCVI (or the approximation through MC dropout), it is applicable to all approaches, regardless of how the output distribution is obtained. A popular, albeit resource-intensive, alternative is ensembling [25]. Here, multiple duplicates of a deep neural network are initialized with different random weights and are trained on (a subset of the) train set. At test time, an example is forwarded through all duplicates, and the result is averaged.

In addition, [27] do not use an (approximate) Bayesian neural network to quantify uncertainty. Rather, the authors compare the probability density of a test example in the feature space of the neural network to the class-conditional distributions derived from the train set. This approach, however, focuses on aleatoric uncertainty and not on the combination with epistemic uncertainty. Furthermore, the focus of this approach lies on classification models trained using a softmax loss. It is unclear to what extent the assumptions, and by extension the method, are valid for other losses or other AI applications.

In a similar fashion, [19] present the *trust score* as a simple and effective way to define whether or not one should trust the output of a classifier. In a first step, an empirically dense subset of examples (the α-high-density set) is constructed for each of the possible output classes during training. The trust score for a test example is then given as the ratio of the distance from that example to the α-high-density set of the nearest class that differs from the predicted class, to the distance from the test example to the α-high-density set of the predicted class.

Although both of the above approaches impose no requirements on the classifier, they come with the added cost of an additional step, both at training time and at inference time. In contrast, our method only requires dropout in the network architecture.

8 Conclusion

In this work, we presented a novel metric for quantifying output uncertainty in stochastic neural networks, based on the Bhattacharyya coefficient, aptly named BC uncertainty. Unlike previously studied metrics—which are based on the total output variance—our metric uses the Bhattacharyya coefficient to estimate the inter-class uncertainty in the prediction of a classification neural network. We provided an intuitive example as to why this is preferable. Additionally, the BC uncertainty is bounded between 0 and 1. This makes the metric easier to interpret and allows us to compare the output uncertainty across datasets.

We empirically validated our metric using benchmark datasets. These are MNIST, CIFAR-10 and ImageNet. We illustrated that, indeed, the model achieves a much higher classification accuracy when the uncertainty is low. In addition, we saw that the uncertainty increases along with the task difficulty when quantified using the Bhattacharyya coefficient. This is not always the case for variance-based metrics. We took a closer look at some examples, randomly sampled according to their corresponding BC uncertainty. Here, we confirmed that images with a low uncertainty are overall easier to classify than those with a high uncertainty. In a final experiment, we introduced suboptimal training conditions and evaluated their influence on the uncertainty. We demonstrated that the model outputs a high uncertainty when it is trained with insufficient data. However, it is important to note that this is not the case when a single class is missing completely. Therefore, further research is required to apply our metric as an anomaly detector.

We used our metric in the real-world use case of skin lesion classification with the HAM10000 dataset. We explained that this dataset is heavily imbalanced, which will have implications on the uncertainty for certain under-represented classes. For example, we showed that the model shows low uncertainty for the class it encountered most during training. We also illustrated that the model has a high classification accuracy for lesions with a low associated uncertainty.

Although the results are positive, our experiments, in particular those regarding skin lesion classification, were conducted without the input from dermatology experts. Therefore, further research is required into the application of deep learning and uncertainty in general practices.

A Additional experiments

We repeat the experiments in Sect. 5.5 using a Bayesian neural network. Due to computational limitations, we restrict ourselves to the MNIST dataset.

The Bayesian neural network is implemented following [42]. The network architecture is similar to the architecture used in Sect. 5.5, consisting of two convolutional layers with 3×3 kernels. These have 16 and 32 filters, respectively. The convolutional layers are followed by a fully connected hidden layer with 128 units and a fully connected output layer with 10 units. The hidden layers have Softplus activations. This function is a smooth approximation of the ReLU activation function that has the analytically important advantage that it never becomes zero [42]. We train the network using Bayes by Backprop [4].

When optimal training conditions apply (using the entire train set), the network reaches a classification accuracy of 0.96 on the test set. This is similar to the 0.98 accuracy achieved by the dropout neural network. Now we introduce suboptimal training conditions in two ways. First, we reduce the train set size by taking only a fraction (0.01, 0.05, 0.1, 0.2, 0.4, 0.8 and 1.0) of the available data. Second, we introduce a simulated class imbalance by downsampling a single class (the digit "3"). For this class, we include only a fraction of the available examples. Additionally, we use a fraction of 0.0 to evaluate the edge case of unseen data.

Figure 12 shows the evolution of the BC uncertainty under different training scenarios for the case of limited training data (top) and for the case of imbalanced training data (bottom). Per scenario, the distribution for the BC uncertainty is given, both for the correct and incorrect classifications. The classification accuracy is shown above these distributions. The results for the Bayesian neural network are comparable to those for the dropout neural network (given in Fig. 8).

Limited train set size When the train set is too small (up until a fraction of 0.2), the network fails to discover patterns in the data, resulting in a low classification accuracy. These scenarios come with a high BC uncertainty. Once enough training data become available (fraction 0.2 and above), the accuracy starts to rise. For the correct classifications, the distribution starts with a high spread that is quickly diminished once more data become available. On the other hand, the distribution for the misclassifications has a mean that is always higher than the distributional mean for the correct classifications. It only slightly decreases and maintains a large spread.

Simulated class imbalance The neural network needs sufficient training data in order to distinguish the digit "3". When there are not enough examples (fractions 0.0, 0.01 and 0.05), the network tries to map examples for this digit onto another digit.

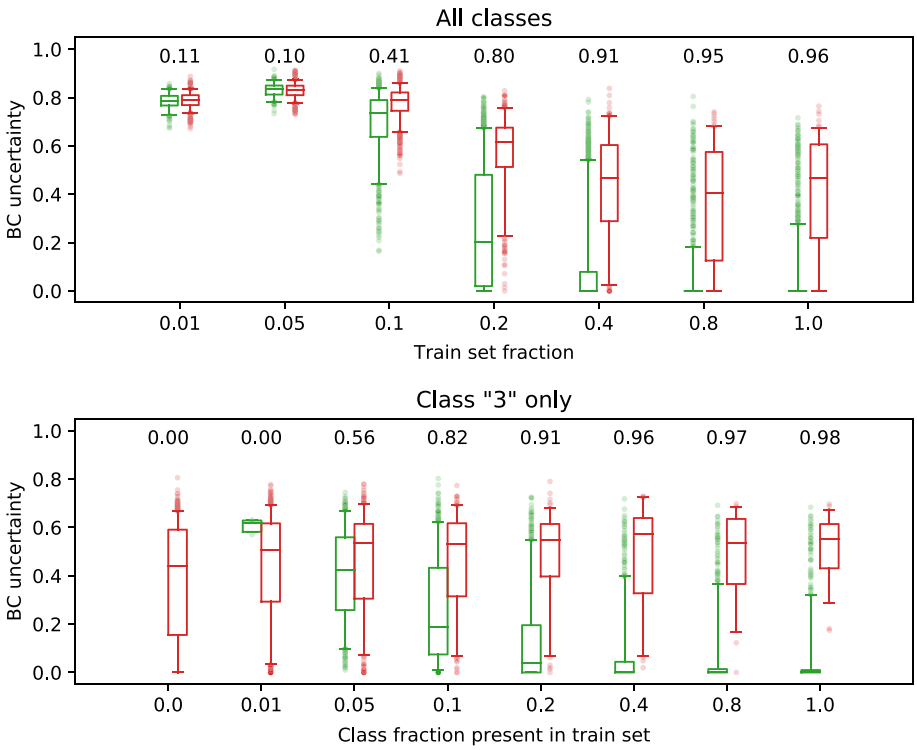

Fig. 12 Distribution of the BC uncertainty for a Bayesian neural network trained on the MNIST dataset under different training scenarios. Either only a fraction of the full train set is used (top), or only a fraction of the available examples for the digit "3" is included for training (bottom). For each scenario, the distribution for the correct classifications is shown in the left (green) box plot, and the distribution for the incorrect classifications is shown in the right (red) box plot. The whiskers represent the 5% and 95% interval. The transparent dots are uncertainty values outside of this interval. The classification accuracy is given by the number above each box plot pair

Funding Part of this work has been supported by Flanders Innovation & Entrepreneurship, by way of grant agreement HBC.2016.0436/HBC.2018.2028 (DermScan).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Acharjya DP et al (2020) Behavioural intention of customers towards smartwatches in an ambient environment using soft computing: an integrated sem-pls and fuzzy rough set approach. Int J Ambient Comput Intell (IJACI) 11(2):80–111
2. American Cancer Society (2020) Cancer facts & figures 2020
3. Bhattacharyya A (1943) On a measure of divergence between two statistical populations defined by their probability distributions. Bull Calcutta Math Soc 35:99–109
4. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural networks. arXiv preprint arXiv:150505424
5. Bria A, Marrocco C, Tortorella F (2020) Addressing class imbalance in deep learning for small lesion detection on medical images. Comput Biol Med 120:103735
6. Celebi ME, Codella N, Halpern A (2019) Dermoscopy image analysis: overview and future directions. IEEE J Biomed Health Inform 23(2):474–478
7. Der Kiureghian A, Ditlevsen O (2009) Aleatory or epistemic? does it matter? Struct Saf 31(2):105–112
8. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S (2017) Dermatologist-level classification of skin cancer with deep neural networks. Nature 542(7639):115–118
9. Gal Y, Ghahramani Z (2016) Bayesian convolutional neural networks with Bernoulli approximate variational inference. arXiv preprint arXiv:150602158
10. Gal Y, Ghahramani Z (2016) Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: International conference on machine learning, pp 1050–1059
11. Gal Y, Islam R, Ghahramani Z (2017) Deep Bayesian active learning with image data. In: Proceedings of the 34th international conference on machine learning, Vol 70, JMLR. org, pp 1183–1192
12. Gan D, Shen J, An B, Xu M, Liu N (2020) Integrating tanbn with cost sensitive classification algorithm for imbalanced data in medical diagnosis. Comput Ind Eng 140:106266
13. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
14. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
15. Graves A (2011) Practical variational inference for neural networks. In: Advances in neural information processing systems, pp 2348–2356
16. Gupta AK, Nagar DK (2018) Matrix variate distributions. Chapman and Hall/CRC
17. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
18. Houlsby N, Huszár F, Ghahramani Z, Lengyel M (2011) Bayesian active learning for classification and preference learning. arXiv preprint arXiv:11125745
19. Jiang H, Kim B, Guan M, Gupta M (2018) To trust or not to trust a classifier. In: Advances in neural information processing systems, pp 5541–5552
20. Jordan MI, Ghahramani Z, Jaakkola TS, Saul LK (1999) An introduction to variational methods for graphical models. Mach Learn 37(2):183–233
21. Kendall A, Badrinarayanan V, Cipolla R (2015) Bayesian segnet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. arXiv preprint arXiv:151102680
22. Kisel'ák J, Lu Y, Švihra J, Szépe P, Stehlík M (2020) "spocu": scaled polynomial constant unit activation function. Neural Comput Appl 1–17
23. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Tech. rep, Citeseer
24. Kwon Y, Won JH, Kim BJ, Paik MC (2020) Uncertainty quantification using Bayesian neural networks in classification: application to biomedical image segmentation. Comput Stat Data Anal 142:106816
25. Lakshminarayanan B, Pritzel A, Blundell C (2017) Simple and scalable predictive uncertainty estimation using deep ensembles. In: Advances in neural information processing systems, pp 6402–6413
26. LeCun Y, Bottou L, Bengio Y, Haffner P et al (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
27. Lee K, Lee K, Lee H, Shin J (2018) A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In: Advances in neural information processing systems, pp 7167–7177
28. Leibig C, Allken V, Ayhan MS, Berens P, Wahl S (2017) Leveraging uncertainty information from deep neural networks for disease detection. Sci Rep 7(1):17816
29. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3431–3440
30. Louizos C, Welling M (2016) Structured and efficient variational deep learning with matrix gaussian posteriors. In: International conference on machine learning, pp 1708–1716
31. MacKay DJ (1992) A practical Bayesian framework for backpropagation networks. Neural Comput 4(3):448–472
32. Marchetti MA, Codella NC, Dusza SW, Gutman DA, Helba B, Kalloo A, Mishra N, Carrera C, Celebi ME, DeFazio JL et al (2018) Results of the 2016 international skin imaging collaboration international symposium on biomedical imaging challenge: Comparison of the accuracy of computer algorithms to dermatologists for the diagnosis of melanoma from dermoscopic images. J Am Acad Dermatol 78(2):270–277
33. Marchetti MA, Liopyris K, Dusza SW, Codella NC, Gutman DA, Helba B, Kalloo A, Halpern AC, Soyer HP, Curiel-Lewandrowski C et al (2020) Computer algorithms show potential for improving

dermatologists' accuracy to diagnose cutaneous melanoma: Results of the international skin imaging collaboration 2017. J Am Acad Dermatol 82(3):622–627
34. Nair T, Precup D, Arnold DL, Arbel T (2020) Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation. Med Image Anal 59:101557
35. Neal R (1995) Bayesian learning for neural networks. Toronto, Ontario, Canada: Department of Computer Science, University of Toronto
36. Ozdemir O, Woodward B, Berlin AA (2017) Propagating uncertainty in multi-stage bayesian convolutional neural networks with application to pulmonary nodule detection. arXiv preprint arXiv:171200497
37. Posch K, Steinbrener J, Pilz J (2019) Variational inference to measure model uncertainty in deep neural networks. arXiv preprint arXiv:190210189
38. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
39. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention, Springer, pp 234–241
40. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
41. Seebock P, Orlando JI, Schlegl T, Waldstein SM, Bogunovic H, Klimscha S, Langs G, Schmidt-Erfurth U (2019) Exploiting epistemic uncertainty of anatomy segmentation for anomaly detection in retinal OCT. IEEE Trans Med Imaging 39(1):87–98
42. Shridhar K, Laumann F, Llopart Maurin A, Liwicki M (2018) Bayesian convolutional neural networks. arXiv preprint arXiv:180605978
43. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556
44. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
45. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
46. Thabtah F, Hammoud S, Kamalov F, Gonsalves A (2020) Data imbalance in classification: experimental evaluation. Inf Sci 513:429–441
47. Tschandl P, Rosendahl C, Kittler H (2018) The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci Data 5:180161
48. Wang G, Li W, Aertsen M, Deprest J, Ourselin S, Vercauteren T (2019) Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. Neurocomputing 338:34–45
49. Xue Y, Deng Y, Garg H (2021) Uncertain database retrieval with measure-based belief function attribute values under intuitionistic fuzzy set. Inf Sci 546:436–447
50. Yildirim MY, Ozer M, Davulcu H (2019) Leveraging uncertainty in deep learning for selective classification. arXiv preprint arXiv:190509509