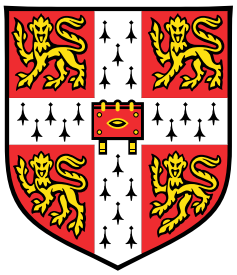
Uncertainty Estimation in Bayesian Neural Networks And Links to Interpretability



Lucy R. Chai

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Churchill College

August 2018

I would like to dedicate this thesis to my family. Thanks for the endless opportunities ...

Declaration

I, Lucy R. Chai of Churchill College, being a candidate for the MPhil in Machine Learning, Speech and Language Technology, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. This dissertation contains fewer than 14,950 words and has fewer than 20 figures.

Lucy R. Chai
August 2018

Acknowledgements

I would like to acknowledge my supervisor, Dr. Tameem Adel, for his guidance throughout this project. I have learned a lot from him during the past few months. I would also like to thank Stefan Depeweg and Dr. José Miguel Hernández-Lobato for their invaluable discussions and insights on my project. Thanks to Daniel Rothchild, Michael Zhao, and Charlie He for feedback on earlier versions of this work and the MLSALT staff for their energy and patience in teaching us the newest topics in this field.

Secondly, I would like to acknowledge my peers in the MLSALT and Churchill Scholars cohorts. I have learned so much from all of you, and I look forward to seeing the amazing things you accomplish.

Finally, I would like to thank the Winston Churchill Foundation of the United States for providing me with the opportunity to study at Cambridge. I am so incredibly grateful for this opportunity.

Abstract

Bayesian neural networks, a hybrid of deep neural networks and probabilistic models, combine the flexibility of deep learning with estimates of uncertainty in predictions. However, like deep neural networks, they are often difficult to interpret – we do not know how correct predictions are made and what makes the prediction uncertain.

Numerous approaches to interpreting neural network predictions have been studied, but there is limited work on interpreting uncertainty in model predictions. Here, we propose a method to visualise the contribution of individual features to predictive uncertainty, epistemic uncertainty (from the model weights), and aleatoric uncertainty (inherent in the input). Our approach measures the change in uncertainty when a given feature of the input is known, compared to when it is unknown.

Applying our approach to the CIFAR10 and ISIC2018 skin lesion diagnosis datasets, we generate smooth visualisations highlighting pixels in the input image that most impact each type of uncertainty. On inputs that are difficult to classify, different areas of the image contribute to epistemic and aleatoric uncertainties. This suggests that some areas of the test image determine its classification (aleatoric uncertainty), other areas distinguish it from the training distribution (epistemic uncertainty), and both contribute to overall predictive uncertainty.

Table of contents

List of figures **xiii**

Nomenclature **xv**

1 Introduction **1**
1.1 Motivation 1
1.2 Research Aims 2
1.3 Thesis Outline 2

2 Background **3**
2.1 Deep Neural Networks 3
2.1.1 Convolutional Neural Networks 4
2.1.2 Error Backpropagation 5
2.2 Bayesian Neural Networks 6
2.2.1 Posterior Weight Distribution 7
2.2.2 Variational Inference 8
2.2.3 Bayesian Predictive Distribution 11
2.3 Uncertainty Estimation 12
2.3.1 Why Should We Decompose Predictive Uncertainty? 12
2.3.2 Decomposing Predictive Uncertainty in Classification 13
2.3.3 Uncertainty Intuition 15

3 Related Work **17**
3.1 What is Interpretability and Why is it Hard? 17
3.2 How Do We Make Models Interpretable? 18
3.3 Additional Challenges with Interpretability in Images 19
3.4 Interpretations of Uncertainty 20

4 Methodology **23**
4.1 Approach . . . 23
4.1.1 Review of Predictive Difference . . . 23
4.1.2 Extension to Uncertainty . . . 24
4.1.3 Implementation Details . . . 28
4.2 Bayesian Neural Network Implementation . . . 29
4.3 Datasets . . . 31

5 Results **33**
5.1 Uncertainty in Bayesian Neural Networks . . . 33
5.2 Uncertainty Decomposition . . . 35
5.2.1 Uncertainty Decomposition with Synthetic Data . . . 36
5.2.2 Uncertainty Decomposition in CIFAR10 . . . 38
5.3 Mapping Uncertainty onto Input Pixels . . . 38
5.3.1 Epistemic and Aleatoric Uncertainty . . . 39
5.3.2 Comparison to Predictive Difference . . . 41
5.3.3 Adding More Training Data . . . 43
5.3.4 Skin Lesion Diagnosis . . . 44

6 Conclusions **47**
6.1 Limitations of our approach . . . 48
6.2 Future work . . . 48

References **51**

Appendix A Sensitivity Analysis in Images **57**

Appendix B Additional Salience Visualisations **59**
B.1 Robustness Across BNN replicates . . . 59
B.2 Absolute Magnitudes of Uncertainty Change . . . 60
B.3 Additional Visualisations . . . 62

List of figures

2.1 Deep Neural Networks 4
2.2 Convolutional Neural Networks 5
2.3 Bayesian Neural Networks 7

4.1 Conditional Sampling and Multivariate Analysis 29
4.2 Early Stopping 30

5.1 Bayesian Predictive Distribution 34
5.2 Comparing Bayesian and Deep Network Decisions 35
5.3 Synthetic Data Distributions 36
5.4 Trends in Epistemic and Aleatoric Uncertainty 37
5.5 Uncertainty Decomposition in CIFAR10 38
5.6 Visualising Uncertainty on Images with High Predictive Uncertainty 40
5.7 Visualising Uncertainty on Images with Low Predictive Uncertainty 41
5.8 Uncertainty and Predictive Difference 42
5.9 Increasing Training Examples 43
5.10 Visualising Uncertainty on Skin Cancer Images 45

A.1 Sensitivity Analysis in Images 58

B.1 Robustness Across Runs 59
B.2 Magnitude of Uncertainty Change 61
B.3 Additional Visualisations 62

Nomenclature

$x^{(n)}$	The n-th training input
$y^{(n)}$	The n-th training label
$\hat{y}^{(n)}$	The model prediction on n-th training input
x^*	Test input
y^*	Test label
$\hat{y}^*$	The model prediction on the test input
x_i	The i-th feature of input x
w	Weights of a neural network
D	Training data, tuples of (x^n, y^n)
$E_z[f(z)]$	Expected value of $f(z)$ with respect to z
$\mathbb{H}[z]$	Entropy of z
$\mathbb{I}[z_1, z_2]$	Information Gain of z_1 and z_2
$\mathcal{N}(z; \mu, \sigma)$	Normal distribution for variable z
$z^k \sim p(z)$	Sample z from distribution $p(z)$

Acronyms / Abbreviations

BNN	Bayesian Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
MC	Monte Carlo (Samples)

Chapter 1

Introduction

1.1 Motivation

Deep neural networks (DNNs) have achieved widespread popularity in various machine learning applications [40, 32, 57]. However, these models can exhibit unintuitive behaviour [52, 59], and how the model arrives at the correct answer often remains unknown.

With the success of DNNs and their potential to aid in making critical decisions, there is increasing need for models that are both accurate and interpretable. For example, a DNN can predict if patient has a medical condition [69], but because current models are imperfect, interpretability helps doctors verify model correctness. Similar ideas apply when using machine learning in public policy [19] or autonomous driving [35], in which interpretability allows us check model decisions and build trust in the system [64].

Interpretable models can also alert us to potential failure cases, such as failure to generalise on unseen data. For instance, a model may attain high accuracy in distinguishing images of wolves from huskies but rely on the presence of snow or grass in the background for the classification [52]. An interpretation showing influential pixels in the decision allows us to conclude that the model will fail on pictures of huskies in snow. Knowing this, we can debug and improve the model by providing such images in training.

Interpretations of the pixels responsible for a decision are known as salience maps, and have been widely investigated in computer vision. These visualisations have been generated by tweaking existing neural network architectures [14, 67, 66] or by measuring model response to slight perturbations [55, 21] or masking of the input [69, 66].

Previous methods of computing salience only use point estimates of the model weights and predictions. However, we often want to know not only what the model prediction is, but also how *uncertain* the prediction is. Uncertainty tells us if the prediction is potentially noisy. Bayesian neural nets (BNNs) – a hybrid of DNNs and probabilistic models – address this

predictive uncertainty and provide a theoretical framework for decomposing uncertainty into uncertainty about the model weights (epistemic) and uncertainty in the input that the model does not capture (aleatoric) [16]. Using these uncertainty estimates allows us to formulate a new approach to interpreting deep Bayesian models.

1.2 Research Aims

In this work we aim to interpret BNN uncertainties by generating salience maps for uncertainty. Depeweg et al. [17] have previously investigated the sensitivity of uncertainty to input features using a gradient-based approach. However, here we focus on interpreting images whose input pixels are highly correlated; using a gradient-based approach directly cannot capture this structure. Our approach extends on the work of Zintgraf et al. [69] to generate smooth salience maps for uncertainty. This project involves training BNN models, deriving equations for uncertainty dependence on input pixels, and demonstrating the method on image datasets.

1.3 Thesis Outline

We start with an overview of deep learning, BNNs, and uncertainty estimation (Chapter 2). Then, we survey interpretability and related work (Chapter 3). We introduce our approach in Chapter 4. Chapter 5 contains the experiments, starting with validation studies on BNNs and uncertainty (§5.1-5.2), and then showing visualisations of uncertainty on the CIFAR10 and ISIC2018 datasets (§5.3). We provide concluding remarks and avenues for future work in Chapter 6.

Chapter 2

Background

This chapter covers background for the remainder of this work. We first discuss traditional deep learning techniques (§2.1), then introduce Bayesian Neural Networks (BNNs) which combine deep learning with probabilistic methods (§2.2). BNNs provide uncertainty estimates on model predictions, which consist of epistemic uncertainty from model weights and aleatoric uncertainty inherent in the input; these types of uncertainties should be considered separately because they are reduced in different ways (§2.3).

2.1 Deep Neural Networks

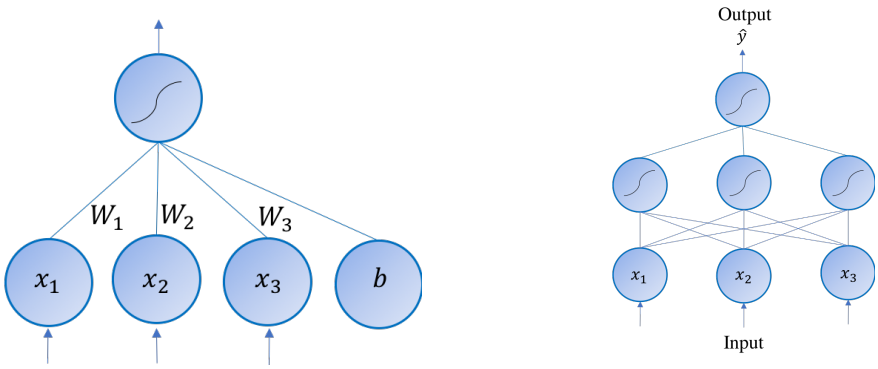
In recent years, deep neural networks (DNNs) have emerged as top-performing models [41] that serve as flexible function approximators [45] or automatic feature generators [50, 37]. The core unit of these models is the *neuron* – loosely based on the biological neuron – that calculates a non-linear transformation of a weighted sum of its inputs. On a d-dimensional input vector $x = [x_1, x_2, \ldots, x_d]$, the operation performed by each neuron is represented by:

$$act(\sum_{i=1}^{d} W_i x_i + b) \tag{2.1}$$

where each W_i is a scalar weight that multiplies an element of the input vector, b is a scalar bias, and $act(\cdot)$ is any activation function such as sigmoid, tanh, or ReLU (Fig. 2.1a). Neurons can be concatenated to form layers, and layers can be further hierarchically organised into networks (Fig. 2.1b).

A forward pass through the network applies subsequent weighted summations and activations through each layer, yielding $\hat{y} = f^w(x)$ where $\hat{y}$ is the network output, x is the input, w aggregates all weights over L layers in the network ($w = \{w_l\}_{l=1}^{L}$), and f encapsulates all operations of the DNN. The weights w are adjusted via an optimisation algorithm to find a

function f^w that correctly identifies important features of the input while ignoring extraneous information.



(a) Neuron (b) Deep Neural Network

Fig. 2.1 (A) Individual neurons perform a weighted summation and a nonlinear activation, shown on a 3-dimensional input. (B) Neurons stacked into a DNN with two hidden layers. Bias terms are omitted in this illustration.

2.1.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are variants of neural networks that are widely applied to inputs in image format. Unlike neurons in *fully-connected* layers that sum over all dimensions of the input as mentioned above, neurons in convolutional layers perform a weighted summation over a local patch of the input (Fig. 2.2). This same local summation is applied to all patches of the input. In essence, this performs a discrete convolution of the weights with the input.

An advantage of the convolutional layer is a reduction in the number of parameters. Unlike fully-connected layers which require a weight parameter for each dimension of the input, convolutional layers only require a weight for each element of a local patch, and weights are shared across all patches in the input. This reduction in the number of parameters is key for high-dimensional image inputs. Furthermore, local connections and shared weights allow for location invariance, so relevant objects can be detected regardless of where they appear in the input image.

CNNs are hierarchically organised, reminiscent of the primate visual ventral cortex [20]. They combine convolutional layers with pooling layers that subsample the input and allow the network to be robust to small shifts in the input. Subsequent dense layers on the reduced feature spaces capture holistic information about the processed input. This architecture is common across CNNs such as AlexNet [40], VGG Net [55], GoogLeNet [58], and ResNet [30], which have achieved wide success in computer vision applications.

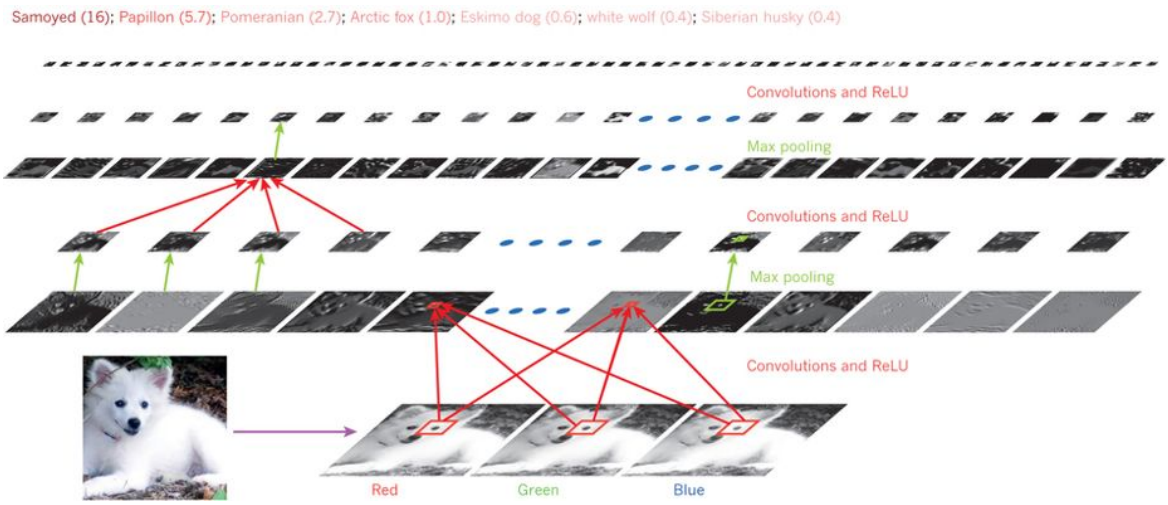


Fig. 2.2 Illustration of a CNN showing convolutional operations over patches of the input (in red) and max-pooling (in green). Image from [32].

2.1.2 Error Backpropagation

The performance of DNNs relies on adjusting weight parameters to learn a desired input-output mapping with respect to an appropriate loss function. We start with N input-output pairs of labeled data: $\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\ldots,(x^{(N)},y^{(N)})\}$, where each $x^{(n)}$ is a d-dimensional vector. A typical loss for regression tasks is the squared error:

$$E_{mse} = \frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)} - \hat{y}^{(n)}\right)^2 \tag{2.2}$$

in which $\hat{y}^{(n)} = f^w(x^{(n)})$ is the model output of the n-th training input. A common loss for classification is cross entropy:

$$E_{ce} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y^{(n),c}\log\hat{y}^{(n),c} \tag{2.3}$$

in which $\hat{y}^{(n),c}$ is the predicted softmax of the c-th class, and the label $y^{(n),c}$ indicates whether the n-th example truly belongs to the c-th class. In softmax, the predictions $\hat{y}^{(n),c}$ are normalised to be greater than zero and sum to one across all c.

The value of the loss function is high upon random initialisation of the weights. The objective of training is to adjust weights to reduce loss, using the gradient of the loss function with respect to the weights. Gradients can be computed via the error backpropagation algorithm, a recursive application of the chain rule of derivatives.

Let y_L represent the final output of a neural network with weights w. The gradient of the loss with respect to the final layer of weights w_L is:

$$\frac{\partial E}{\partial w_L} = \frac{\partial E}{\partial y_L} \times \frac{\partial y_L}{\partial z_L} \times \frac{\partial z_L}{\partial w_L} \tag{2.4}$$

in which E represents the loss function, $y_L = act(z_L)$ represents the activation function, and $z_L = w_L \times y_{L-1}$ represents the weighted summation over inputs to layer L via matrix multiplication.

For every subsequent layer l in the network, the gradient of loss with respect to weights w_l follows a recursive formula:

$$\frac{\partial E}{\partial w_l} = \frac{\partial E}{\partial y_L} \times \left(\prod_{i=l+1}^{L} \frac{\partial y_i}{\partial z_i} \times \frac{\partial z_i}{\partial y_{i-1}} \right) \times \frac{\partial y_l}{\partial z_l} \times \frac{\partial z_l}{\partial w_l} \tag{2.5}$$

Frameworks such as Tensorflow [2] and PyTorch [48] compute gradients via automatic differentiation. After obtaining gradients, the weights are shifted in the direction of the gradient to reduce the loss toward a local optimum. Modifications such as momentum, which accelerates gradient descent in relevant directions, and step size decay throughout learning further quicken the progression toward a local optimum [4, 49].

2.2 Bayesian Neural Networks

In traditional deep neural networks, the trained weights of a DNN are a point estimate for each parameter, leading to deterministic network outputs for a given input. On the other hand, Bayesian neural networks – or BNNs – specify a distribution over the weight parameters (Fig. 2.3) [46, 43]. Estimating this posterior weight distribution, $p(w|D)$, allows BNNs to capture uncertainty in the predictions.

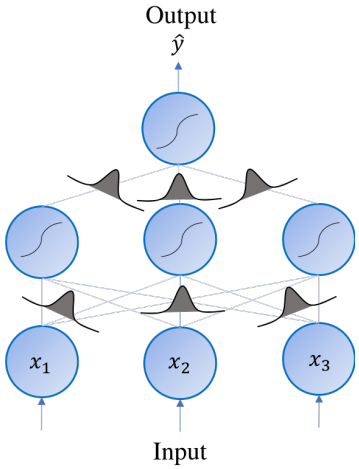


Fig. 2.3 BNNs specify a distribution over the weights rather the point estimate used in DNNs. Bias terms are omitted in the illustration.

2.2.1 Posterior Weight Distribution

Given N training inputs and outputs $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\}$, a BNN learns a *posterior distribution* over the network weights, $p(w|D)$. As before, w aggregates all weights over L layers in the network: $w = \{w_l\}_{l=1}^{L}$. This distribution represents how likely a particular setting of weights are after seeing the training data, instead of the point estimate of weights in DNNs. A direct application of Bayes' rule yields:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)} \tag{2.6}$$

In the numerator, the first term $p(D|w)$ reflects the likelihood of the training data given a particular weight setting w. Assuming that each training data point is independent, this quantity becomes the product of likelihoods for each individual training point:

$$p(D|w) = \prod_{n=1}^{N} p(y^{(n)}|w, x^{(n)}) \tag{2.7}$$

For regression tasks, the likelihood term may refer to Gaussian likelihood, in which $p(y|w,x) = \mathcal{N}(y; \mu, \Sigma)$. Depending on the specific implementation, the quantities μ, Σ, or both can predicted from the input x via a neural network. Classification tasks may use the softmax predictions of the neural network directly as the likelihood: $p(y = c|w,x) = f^w(x)^c$, where c is the true class of the input, and the softmax predictions for each class $f^w(x)^c$ are normalized to be greater than zero and sum to one.

The second term of the numerator, $p(w)$, is the prior distribution over the weights. It reflects our belief in the distribution of the weights without seeing any data. One example prior distribution is the Gaussian distribution.

Both terms in the numerator are tractable to compute for a particular setting of w. However, the problem with attaining the posterior distribution is the denominator, $p(D)$. Computing $p(D)$ involves marginalising over all weight settings:

$$p(D) = \int p(D|w)p(w)dw \tag{2.8}$$

In many models this integral is intractable, motivating the need for approximations of the posterior. Sampling methods, such as Metropolis Hastings or Hamiltonian Monte Carlo, yield unbiased estimates of the true posterior but may be slow to converge. An alternative approach is variational inference, in which the true posterior distribution is approximated with a simpler variational distribution [34, 7]. This approximation is biased, but is often faster than sampling methods. In the remainder of this work, we use a variational distribution to approximate the posterior weight distribution of BNNs.

2.2.2 Variational Inference

Variational inference approximates the intractable posterior distribution $p(w|D)$ with a simpler tractable distribution over the model weights, $q(w)$, with variational parameters v. The variational parameters v are fitted so that $q(w)$ approximates the desired posterior $p(w|D)$. This fitted variational distribution, rather than the true posterior, is then used for model predictions.

One way to measure the distance between two probability distributions $q(x)$ and $p(x)$ is using Kullback-Leibler divergence, or KL-divergence, defined as:

$$\mathrm{KL}(q(x)||p(x)) \equiv \mathbb{E}_{q(x)}\left[\log \frac{q(x)}{p(x)}\right] = \int q(x) \log \frac{q(x)}{p(x)} dx \tag{2.9}$$

To make the variational distribution $q(w)$ close to the posterior $p(w|D)$, we want to minimise the KL-divergence between these two distributions:

$$\begin{aligned}
\mathrm{KL}(q(w)||p(w|D)) &= \mathbb{E}_{q(w)}\left[\log \frac{q(w)}{p(w|D)}\right] \\
&= \int q(w) \log \frac{q(w)}{p(w|D)} dw \\
&= \int q(w) \log \frac{q(w)p(D)}{p(D|w)p(w)} dw \\
&= \int q(w) \log \frac{q(w)}{p(w)} dw + \int q(w) \log p(D) dw - \int q(w) \log p(D|w) dw \\
&= \mathrm{KL}(q(w)||p(w)) + \log p(D) - \mathbb{E}_{q(w)}\left[\log p(D|w)\right] \qquad (2.10)
\end{aligned}$$

Due to the $\log p(D)$ term, the KL-divergence cannot be computed directly. However, by rearranging terms, we obtain:

$$\log p(D) = \mathrm{KL}(q(w)||p(w|D)) + \mathbb{E}_{q(w)}\left[\log p(D|w)\right] - \mathrm{KL}(q(w)||p(w)) \qquad (2.11)$$

Because $\log p(D)$ is a (albeit intractable) constant, minimising $\mathrm{KL}(q(w)||p(w|D))$ is equivalent to maximising $\mathbb{E}_{q(w)}\left[\log p(D|w)\right] - \mathrm{KL}(q(w)||p(w))$. This latter expression is referred to as the evidence lower bound, or ELBO, and can be also derived from the log probability of the data:

$$\begin{aligned}
\log p(D) &= \log \int p(D|w)p(w) dw \\
&= \log \int \frac{q(w)}{q(w)} p(D|w)p(w) dw
\end{aligned}$$

Applying Jenson's inequality:

$$\begin{aligned}
&\geq \int q(w) \log \frac{p(D|w)p(w)}{q(w)} dw \\
&= -\int q(w) \log \frac{q(w)}{p(w)} dw + \int q(w) \log p(D|w) dw \\
&= -\mathrm{KL}(q(w)||p(w)) + \mathbb{E}_{q(w)}\left[\log p(D|w)\right]
\end{aligned}$$

Thus we have that:

$$\log p(D) \geq -\mathrm{KL}(q(w)||p(w)) + \mathbb{E}_{q(w)}\left[\log p(D|w)\right] \qquad (2.12)$$

Unlike (2.11), inequality (2.12) lacks the term $\mathrm{KL}(q(w)||p(w|D))$. However, due to the non-negative properties of KL-divergence, we conclude that the ELBO is less than or equal to the log probability of the data.

Therefore, finding variational parameters v to minimise the KL divergence is the same as maximising the ELBO. In effect, variational inference translates the problem of inference over the weight distribution into the optimisation problem of maximising the ELBO. Once the ELBO objective is defined, we can sample from $q(w)$ and use backpropagation, like in DNNs, to find optimal values of the variational parameters that maximise the ELBO.

Alternative Divergences

The KL-divergence is only one way of capturing the distance between distributions. An alternative metric is the more general α-divergence [3, 68]:

$$D_\alpha(p(x)||q(x)) = \frac{1}{\alpha(1-\alpha)}\left(1 - \int p(x)^\alpha q(x)^{1-\alpha} dx\right) \tag{2.13}$$

Using the same distributions $q(w)$ and $p(w|D)$ as before, as $\lim \alpha \to 0$ we recover $\mathrm{KL}(q(w)||p(w|D))$ used in variational inference, but this KL-divergence encourages $q(w)$ to be zero everywhere $p(w|D)$ is zero. Consequently, a variational distribution fitted with the KL-divergence criteria tends to underestimate uncertainty because it fits a local mode of the posterior [42, 31].

The motivation for using α-divergence is to obtain better uncertainty estimates by relaxing this constraint. A small value of α encourages the variational distribution to fit the highest mode of the posterior, and a large α encourages the variational distribution to cover the entire posterior mass [31]. When $\alpha = 0.5$, the divergence is symmetric such that $D_{0.5}(p||q) = D_{0.5}(q||p)$. The α-divergence objective has been applied to BNNs in both regression and classification contexts, with the intermediate value of $\alpha = 0.5$ observed to produce better predictions than $\alpha = 0$ used in variational inference and $\alpha = 1$ used in expectation propagation [42, 31, 15].

Variational Distribution

The essence of the variational distribution is that it is similar enough to the posterior weight distribution after minimisation of divergence, but easy to sample from. A common form of the variational distribution is the mean-field approximation, in which we assume that the variational distribution factorizes into the product of distributions by treating the weights as independent variables [7]. Furthermore, we can use a Gaussian as the variational distribution,

allowing one to more easily sample from a Normal distribution rather than the exact weight posterior [8]. In this case, the variational distribution becomes [16]:

$$q(w) = \prod_{l=1}^{L} \prod_{i=1}^{V_l} \prod_{j=1}^{V_{l-1}+1} \mathcal{N}(w_{ijl}; \mu_{ijl}^{w}, \sigma_{ijl}^{2,w}) \tag{2.14}$$

There are two variational parameters, μ_{ijl}^{w} and $\sigma_{ijl}^{2,w}$ for each weight between the j-th neuron of layer $l-1$ and the i-th neuron of layer l, effectively doubling the number of parameters.

For large networks, doubling the number of parameters is costly. Another practical distribution is the *dropout approximate variational distribution*, where the distribution for each weight can be interpreted as a mixture of two Gaussians with small spread (essentially delta peaks) and one component is centered at zero [42, 23, 24]. Here, the variational distribution becomes [26]:

$$q(w) = \prod_{l=1}^{L} M_l \cdot \text{diag}\left[\text{Bernoulli}(1-p)^{V_l}\right] \tag{2.15}$$

M_l is a $V_{l-1} \times V_l$ sized matrix of variational parameters, and V_l is the number of neurons in layer l. The variational parameters are multiplied by a diagonal masking matrix which zeroes out columns of M_l with probability $1-p$. The attractiveness of this approach is that previously trained models with dropout (stochastically zeroing out neurons [56]), simply with dropout on at test time, can be used to approximate sampling from the weight posterior [27]. This dropout approximate variational distribution can be implemented simply by adding a dropout layer before each neural network layer [42], thus avoiding a two-factor increase in the number of parameters required by the mean-field Gaussian approximation.

2.2.3 Bayesian Predictive Distribution

With a posterior distribution over the weights, the output y^* of a test input x^* can be predicted by marginalizing over the weights, using the posterior weight distribution:

$$p(y^*|x^*, D) = \int p(y^*|x^*, w) p(w|D) dw \tag{2.16}$$

Due to the intractability of the exact posterior, we replace $p(w|D)$ with its approximate variational distribution $q(w)$ after the variational parameters ν have been fitted:

$$p(y^*|x^*, D) \approx \int p(y^*|x^*, w) q(w) dw \tag{2.17}$$

Rather than direct integration over the weight-space, a common approximation is Monte Carlo (MC) sampling, in which K samples are drawn from the weight distribution, $w^k \sim q(w)$, and each sample is used to calculate the likelihood $p(y^*|x^*, w^k)$. Using this approximation, the distribution over the output becomes:

$$p(y^*|x^*, D) \approx \frac{1}{K} \sum_{k=1}^{K} p(y^*|x^*, w^k) \tag{2.18}$$

2.3 Uncertainty Estimation

While DNNs provide point estimates for predictions, BNNs provide a predictive *distribution*. The spread of this distribution captures the certainty or lack thereof in the model's predictions. But why do we care about uncertainty? Measuring uncertainty tells us when predictions on an test input are potentially noisy because the input falls outside the training distribution, or because there are unobserved variables that the model fails to capture. The point predictions from DNNs cannot tell us about these properties of the input [23].

The total uncertainty in the prediction, i.e. the *predictive uncertainty*, is the sum of epistemic and aleatoric uncertainty [16, 18]. *Epistemic uncertainty* refers to our uncertainty in the model parameters. This can be imagined as the spread of the posterior weight distribution $p(w|D)$, in which a flatter posterior distribution reflects higher epistemic uncertainty, while a peaked posterior distribution reflects lower epistemic uncertainty. On the other hand, *aleatoric uncertainty* refers to uncertainty originating from the input. Given the input instance and fixed weight parameters, high aleatoric uncertainty means that we have a noisy estimate of its output (for regression) or we do not know what class it belongs to (for classification). High aleatoric uncertainty suggests that we do not have enough information to predict the output value for an input with fixed weight settings, due to unobserved or latent variables that the model cannot capture [16, 17].

2.3.1 Why Should We Decompose Predictive Uncertainty?

Decomposing overall predictive uncertainty is important because epistemic and aleatoric uncertainty tell us about different facets of an input. High epistemic uncertainty suggests that the test input is an outlier relative to the training distribution. We can reduce epistemic uncertainty by collecting more training data near the test region, such that in the limit of infinite data, epistemic uncertainty collapses to zero [26, 35] (i.e. if we know all data in the universe, we are confident of the function mapping input to output, and thus the weight distribution becomes a delta peak). More data does not help aleatoric uncertainty; to reduce

it, we need knowledge about unobserved variables via additional features or more refined measurements (e.g. more precise sensors) [17, 26]. In practice, these measurements are often unavailable, so it is not always possible to reduce aleatoric uncertainty.

A second argument for decomposing uncertainty is that depending on our application, we may prioritise one type of uncertainty over another. In reinforcement learning, we aim to efficiently explore the state-action space. Here, we want to collect data in regions of high epistemic uncertainty (places where data is scarce), allowing us to know about previously unseen parts of the space. We would not prioritise collecting data in regions of high aleatoric uncertainty, as we cannot reduce the noise in our estimates with more data. Alternatively, if we want to predict our return from different types of stocks, we care more about aleatoric uncertainty. If we want to make a risk-adverse investment, we prioritise stocks with low aleatoric uncertainty and high return over those with higher aleatoric or epistemic uncertainty. These distinctions cannot be made from predictive uncertainty alone; inputs with high predictive uncertainty can have contributions from epistemic, aleatoric, or both types of uncertainty.

2.3.2 Decomposing Predictive Uncertainty in Classification

In classification problems, we can measure uncertainty as the entropy of the softmax distribution [27]:

$$\mathbb{H}[\hat{y}^*|x^*, D]$$

which decomposes into the sum of two terms:

$$\mathbb{H}[\hat{y}^*|x^*, D] = \mathbb{I}[\hat{y}^*, w|x^*, D] + E_{w\sim p(w|D)}\left[\mathbb{H}[\hat{y}^*|x^*, w]\right] \tag{2.19}$$

where $\mathbb{I}$ represents information gain, E is expected value, and $\hat{y}^*$ represents the model's output softmax on input x^*.

Depeweg et al. [16] interpreted (2.19) as a decomposition of uncertainty into its aleatoric and epistemic components. Namely, $\mathbb{H}[\hat{y}^*|x^*, D]$, the entropy in the output classification, is the predictive uncertainty.

On the right-hand side of (2.19), $E_{w\sim p(w|D)}\left[\mathbb{H}[\hat{y}^*|x^*, w]\right]$ is the average entropy when the weights are fixed, and thus the uncertainty arises from the input x^* rather than the weights. Therefore, $E_{w\sim p(w|D)}\left[\mathbb{H}[\hat{y}^*|x^*, w]\right]$ can be interpreted as the aleatoric uncertainty. This quantity reflects our uncertainty in the predicted class of the input using only the available features and a fixed weight setting.

Lastly, if we rearrange the equation and subtract aleatoric uncertainty from predictive uncertainty [33], we have: $\mathbb{I}[\hat{y}^*, w|x^*, D] = \mathbb{H}[\hat{y}^*|x^*, D] - E_{w\sim p(w|D)}[\mathbb{H}[\hat{y}^*|x^*, w]]$. This difference represents epistemic uncertainty because it is the remaining uncertainty from the model weights, not from the input. High epistemic uncertainty means that, on each sample from the weight posterior, the model predicts a different class with high confidence for the same input [27].

Because we cannot tractably integrate over all settings of the weights, we rely on sampling values for the weights from the simpler variational distribution $w^k \sim q(w)$.

Estimating Predictive Uncertainty

To approximate the predictive uncertainty for an input x^*:

$$\mathbb{H}[\hat{y}^*|x^*, D] = -\sum_c p(\hat{y}^* = c|x^*, D) \times \log p(\hat{y}^* = c|x^*, D)$$

We expand $p(\hat{y}^* = c|x^*, D)$ as a marginalisation over weights:

$$= -\sum_c \left(\int p(\hat{y}^* = c|x^*, w) p(w|D) dw \right) \times \log \left(\int p(\hat{y}^* = c|x^*, w) p(w|D) dw \right)$$

and replace the exact posterior with the variational distribution:

$$\approx -\sum_c \left(\int p(\hat{y}^* = c|x^*, w) q(w) dw \right) \times \log \left(\int p(\hat{y}^* = c|x^*, w) q(w) dw \right)$$

Finally, rather than direct integration, we take K MC samples of the weights:

$$\approx -\sum_c \left(\frac{1}{K} \sum_k p(\hat{y}^* = c|x^*, w^k) \right) \log \left(\frac{1}{K} \sum_k p(\hat{y}^* = c|x^*, w^k) \right) \tag{2.20}$$

where $p(\hat{y}^* = c|x^*, w^k)$ is the predicted softmax output for class c using the k-th sample of weights w^k from $q(w)$.

Estimating Aleatoric Uncertainty

The aleatoric uncertainty is the average entropy for a particular setting of the weights:

$$\begin{aligned} E_{w\sim p(w|D)}\left[\mathbb{H}\left[\hat{y}^*|x^*,w\right]\right] &= -\int p(w|D)\left[\sum_c p(\hat{y}^*=c|x^*,w)\log p(\hat{y}^*=c|x^*,w)\right]dw \\ &\approx -\int q(w)\left[\sum_c p(\hat{y}^*=c|x^*,w)\log p(\hat{y}^*=c|x^*,w)\right]dw \\ &\approx -\frac{1}{K}\sum_k\sum_c p(\hat{y}^*=c|x^*,w^k)\log p(\hat{y}^*=c|x^*,w^k) \end{aligned} \tag{2.21}$$

We apply the same steps of replacing the exact posterior with the variational distribution and taking K weight samples from the variational distribution.

Estimating Epistemic Uncertainty

Finally, the epistemic uncertainty is the difference between (2.20) and (2.21):

$$\begin{aligned} \mathbb{I}\left[\hat{y}^*,w|x^*,D\right] &= \mathbb{H}\left[\hat{y}^*|x^*,D\right] - E_{w\sim p(w|D)}\left[\mathbb{H}\left[\hat{y}^*|x^*,w\right]\right] \\ &\approx -\sum_c\left(\frac{1}{K}\sum_k p(\hat{y}^*=c|x^*,w^k)\right)\log\left(\frac{1}{K}\sum_k p(\hat{y}^*=c|x^*,w^k)\right) \\ &\quad + \frac{1}{K}\sum_k\sum_c p(\hat{y}^*=c|x^*,w^k)\log p(\hat{y}^*=c|x^*,w^k) \end{aligned} \tag{2.22}$$

2.3.3 Uncertainty Intuition

Two inputs may have the same predictive uncertainty, but have differing epistemic and aleatoric uncertainties. The following hypothetical example demonstrates this phenomena in a simple 2-class scenario to motivate why we consider the uncertainties separately.

Suppose we want to classify an input between two classes. On the first input $x^{(1)}$, we take $K = 4$ samples from $q(w)$ and obtain softmax output over the four stochastic forward passes:

$$\begin{bmatrix}0.0 & 1.0\\ 0.0 & 1.0\\ 0.0 & 1.0\\ 0.0 & 1.0\end{bmatrix}$$

By applying (2.20) and (2.21), we have that both epistemic and aleatoric uncertainty are zero, resulting in zero predictive uncertainty.

With a stochastic neural network, it is possible for each sample w^k from $q(w)$ to yield very confident predictions, and for the predicted class to change on every sample. For example, if on input $x^{(2)}$ we obtain:

$$\begin{bmatrix}1.0 & 0.0\end{bmatrix}$$
$$\begin{bmatrix}0.0 & 1.0\end{bmatrix}$$
$$\begin{bmatrix}1.0 & 0.0\end{bmatrix}$$
$$\begin{bmatrix}0.0 & 1.0\end{bmatrix}$$

then by a similar approach, epistemic uncertainty will be one (with log base 2), but aleatoric uncertainty will still be zero.

Finally, for a point $x^{(3)}$, we obtain output of:

$$\begin{bmatrix}0.5 & 0.5\end{bmatrix}$$
$$\begin{bmatrix}0.5 & 0.5\end{bmatrix}$$
$$\begin{bmatrix}0.5 & 0.5\end{bmatrix}$$
$$\begin{bmatrix}0.5 & 0.5\end{bmatrix}$$

where epistemic uncertainty is zero, but aleatoric uncertainty is one.

For both $x^{(2)}$ and $x^{(3)}$, the predictive uncertainty is the same. However, the stochastic forward passes illustrate the decomposition of predictive uncertainty into epistemic and aleatoric components – uncertainty is entirely from the model weights for $x^{(2)}$, and entirely from the input for $x^{(3)}$. Intuitively, aleatoric uncertainty measures uncertainty in the softmax classification on individual weight samples, while epistemic uncertainty captures how much the predictions deviate across weight samples.

Chapter 3

Related Work

In this chapter we survey recent work in interpretability in machine learning. Inherent difficulties in interpretability are the lack of concrete definitions and limitations in evaluation (§3.1). Despite these difficulties, several interpretable models have been proposed, either by incorporating interpretability into the model architecture or by explaining a black-box model (§3.2). Interpretability on datasets with highly correlated features presents additional challenges (§3.3), and so far there is only limited work on interpreting uncertainty in model decisions (§3.4).

3.1 What is Interpretability and Why is it Hard?

As data-driven algorithms become more widespread in automating everyday decisions, there is growing pressure on the interpretability of such approaches. One far-reaching example is the European General Data Protection Regulation, implemented in 2018, which states that users have a right to "obtain an explanation of the decision reached" based on automated processing [47, 28]. The right to explanation is not just a recent concept; under the United States Equal Credit Opportunity Act of 1974, creditors must notify applicants of adverse action taken, accompanied by a "statement of reasons for adverse action" [1]. Yet, as more everyday decisions become digitised with machine learning, model interpretability becomes crucial for regulation compliance and client-side satisfaction.

Motivations for interpretable models are numerous. For instance, Weller [64] suggests that explanations can allow developers to debug a system, users to rationalise predictions, experts to audit decisions, or system deployers to guide customer behavior. However, a crucial challenge is that interpretability itself is difficult to define. Unlike related areas of fairness and privacy, rigorous criteria for interpretability do not yet exist [19]. Rather, interpretability is often defined relative to qualitative and human-driven standards: Biran and Cotton [6]

define interpretable models as those whose "operations can be understood by a human," while Ribeiro et al. [52] state that explaining a prediction means "presenting textual and visual artifacts that provide qualitative understanding" of the model. To further complicate matters, interpretability is used interchangeably with other words such as understandability, comprehensibility, and explanability [5], and forms only one subcomponent of overall model transparency [64].

The lack of concrete definitions for interpretability makes it highly subjective; interpretations for a model must center around the requirements of the intended audience. Model explanations for experts are unintelligible for laymen, and vice versa. Overly simplistic explanations are not trusted [22], but explanations must also be concise enough for users' limited "perceptual budgets" [51]. Doshi-Velez and Kim [19] propose a three-class *taxonomy* for evaluating interpretability including 1) using domain expert knowledge on real tasks, 2) conducting experiments with non-experts on simplified tasks, or 3) comparing to models already vetted to be interpretable via prior human experiments, like sparse logistic regression and decision trees [52]; all three modes of evaluation require a human in the loop, either directly or indirectly.

An interpretable model is generally viewed as advantageous, but we must be careful that the additional transparency yielded by the interpretation does not cause harm [64]. One potentially harmful scenario occurs when the audience of an explanation differs from the beneficiaries; such is the case when recommender systems provide explanations to manipulate user actions [64]. A second consideration is the balance between interpretability and model performance. The main goal is to maximise accuracy and reliability – e.g. in aircraft autopilot systems we prefer less interpretable systems resulting in fewer crashes than more interpretable ones causing more accidents [64]. Thus, we may not always desire the most interpretable model but rather a model that is both high-performing and understandable for the desired audience.

3.2 How Do We Make Models Interpretable?

One of the most successful modern models, the deep neural network, is inherently uninterpretable. Szegedy et al. [59] observed that combinations of neurons in DNNs – not just single neurons – are sensitive to meaningful patterns of the input and that barely-visible perturbations of the input drastically change the predicted class. How do we explain model decisions in the face of these challenges?

Approaches to interpretability are broadly categorised as *model-based* or *model-agnostic*. Model-based approaches focus on algorithms that are inherently explainable, including

decision trees [52], falling rule lists [63], sparse linear models [61], and nearest neighbors [22]. The challenge with these fixed classes is balancing interpretability and accuracy with limited model complexity; to avoid overwhelming a human user, a decision tree must not have too many leaves, or a nearest neighbor model too many neighbors. Such restrictions may reduce accuracy due to lesser model complexity, but accuracy is also important to avoid generating trivial explanations which are easy to understand yet yield no connection to the data [54, 5].

Model-based interpretations are not limited to these fixed categories of algorithms. One can construct models parametrised by example instances, which serve as explanations without sacrificing accuracy. Chen et al. [10] construct a deep network that learns prototypical parts of training images, such as characteristics of birds, and identifies prototypes in a test image to classify the instance. A related model is the generative Bayesian Case Model, which infers clusters on data parametrised by a prototypical example and a subspace of relevant features important for that cluster [36].

Alternatively, model agnostic approaches build ad-hoc interpretations of already high-performing models by treating the model as a black box, thus avoiding the accuracy vs. interpretability tradeoff. The model and the explanation are disjoint – one can use the same model with different levels of explanation catered to the user and easily switch models while retaining the same method of explanation [51]. Model-agnostic explanations can either be *local explanations* of a particular instance or *global explanations* of the overall system [19, 52, 64]. Local explanations focus on a provided test input. They may determine the features of the input influencing a prediction [53] or the training instances contributing to the decision on a test input (Koh and Liang [38] take the latter approach using influence functions [12]). Global explanations provide an understanding of the model as a whole. For example, Ribeiro et al. [52] propose SP-LIME, which highlights features important to the classifier and example instances that maximise coverage over all features. Similarly, the DEMUD algorithm selects interesting instances for the model based on singular value decomposition reconstruction error and generates feature-based explanations via the residual vectors of the decomposition [62].

3.3 Additional Challenges with Interpretability in Images

In images, we often aim to determine which input pixels are most responsible for a model decision; such explanations are termed salience maps in computer vision. However, interpreting model decisions on images presents additional challenges, as images are very high dimensional inputs with highly correlated pixels. Due to spatial correlations among pixels,

we desire salience maps that are smooth; because patches of neighboring pixels are often similar, it is unintuitive when one pixel contributes to a decision but similar neighbors do not. While we focus on images in this work, it is important to note that smoothly varying metrics of feature importance are relevant for any dataset with spatial or temporal correlations, not just images in particular.

Image salience maps can be generated by isolating the intermediate activations of a network and mapping them into the input pixel space. For instance, Zeiler and Fergus [66] propose a deconvolutional architecture to identify pixels contributing to individual neuron responses. A related method also focusing on intermediate activations is the Class Activation Map, which rewrites the CNN pooling operation to highlight class-discriminative regions in an image [67]. These approaches of upsampling from intermediate activations yields smooth salience maps.

When we do not have access to intermediate activations of a CNN, we can determine the input pixels that impact the final classification. Simonyan and Zisserman [55] take a gradient-based approach, generating salience maps that roughly localise the pertinent object, but are not necessarily smooth and contain strong responses in pixels irrelevant to the classification [21]. Smoothness has been addressed by adding a total-variation penalty to the loss function in a gradient-based iterative procedure [21] and a real-time masking model approach [14]. Both approaches generate masks which obscure key parts of the image, finding either the smallest region which maximally reduces class score when obscured or the smallest region sufficient to preserve the prediction when everything else is covered. One can also study model saliency by obscuring parts of an image and recording changes in model output [66, 69], in which the smoothness of salience map depends on the size of the obscured patch [69].

3.4 Interpretations of Uncertainty

These previous approaches depend on the predictive output, and in some cases the intermediate activations, of deep neural networks. But often we care about not only the prediction of a model, but also how uncertain the prediction is, to know if the prediction is potentially noisy. Bayesian neural networks address the inherent uncertainty in model predictions, but factors that contribute to uncertainty have not been extensively investigated. Depeweg et al. [17] examine the *sensitivity* of uncertainty to changes in individual features by taking the gradient of uncertainty with respect to the input. Their method works well for datasets in which features measure individual human-understandable values, like temperature and pressure. However, there are limitations when applying this approach on images because it

considers the sensitivity of features in isolation, ignoring the highly correlated structure of image pixels (see Appendix A). Furthermore, each pixel we perceive is the combination of three color channels, so meaningful interpretations should consider channels jointly rather than producing a salience map separately for each channel.

In this work, we follow the approach of Depeweg et al. [17] in understanding how uncertainty changes due to input pixels, but derive the change in uncertainty by extending the prediction difference approach from Robnik-Šikonja and Kononenko [53] and Zintgraf et al. [69]. This approach generates smooth visualisations that highlight parts of an image contributing to uncertainty in the model prediction. Our methodology is further described in the following chapter.

Chapter 4

Methodology

In this chapter we introduce a method to generate salience maps for natural images using Bayesian neural networks (§4.1), extending the ideas of predictive difference (§4.1.1) and uncertainty decomposition (§4.1.2). This model-agnostic approach does not depend on the specific implementation of Bayesian neural network, so we will then describe a BNN implementation used in the following experiments, chosen for improved uncertainty estimates via α-divergences and a reduced parameter space via dropout (§4.2). We also overview the datasets used in experiments (§4.3).

4.1 Approach

Following the approach of Depeweg et al. [17], we seek to determine which input dimensions affect the uncertainty of the Bayesian model and to decompose this uncertainty into its epistemic and aleatoric components. To circumvent the limitations of sensitivity analysis (taking the gradient of uncertainty with respect to the input), we use model-agnostic explanations inspired by predictive difference analysis [53, 69], as explained below.

4.1.1 Review of Predictive Difference

On a test input x^*, the predictive difference approach estimates the relevance of each feature x_i^* to the prediction by comparing the model output when feature x_i^* is known, to the model output when x_i^* is unknown [53]. The original implementation uses point estimates for weights rather than a weight distribution. For models that output a softmax probability distribution, $p(\hat{y}^* = c|x^*)$ represents the softmax output of class c when all features are known, and $p(\hat{y}^* = c|x^*_{-i})$ is the softmax output of class c when all features in x^*, *except the i-th feature*, are known. Thus, we have:

$$\text{pd}_{i,c}(x^*) \equiv p(\hat{y}^* = c|x^*) - p(\hat{y}^* = c|x^*_{-i}) \tag{4.1}$$

In practice, rather than taking the direct difference of probabilities, one can also measure the difference in log probabilities, or the difference in log odds [53]. Regardless of the exact metric of difference, all approaches require knowing $p(\hat{y}^* = c|x^*)$ and $p(\hat{y}^* = c|x^*_{-i})$. The probability of the input x^* belonging to class c, $p(\hat{y}^* = c|x^*)$, is directly obtained via a forward pass of the model. When we do not know feature x^*_i, we cannot take the same approach. Instead, we estimate $p(\hat{y}^* = c|x^*_{-i})$ by marginalisation over x_i:

$$\begin{aligned} p(\hat{y}^* = c|x^*_{-i}) &= \int p(\hat{y}^* = c, x_i|x^*_{-i}) dx_i \\ &= \int p(\hat{y}^* = c|x_i, x^*_{-i}) p(x_i|x^*_{-i}) dx_i \end{aligned} \tag{4.2}$$

In effect, (4.2) simulates the counterfactual of not knowing the i-th feature by marginalising over all values of that feature, whose distribution $p(x_i|x^*_{-i})$ is estimated from the training data. Note that we use x^*_i to refer to the value of the i-th feature of a test input, and x_i to refer to the value of the i-th feature estimated from training inputs (we further address estimating $p(x_i|x^*_{-i})$ in §4.1.3).

Rather than summing over all values of x_i, we take an MC sampling approach. We draw M samples $x^m_i \sim p(x_i|x^*_{-i})$ and perform forward passes replacing x^*_i with x^m_i:

$$p(\hat{y}^* = c|x^*_{-i}) \approx \frac{1}{M} \sum_{m=1}^{M} p(\hat{y}^* = c|x^m_i, x^*_{-i}) \tag{4.3}$$

4.1.2 Extension to Uncertainty

Applying this method to uncertainty of BNNs in classification, we quantify the change in uncertainty due to knowing feature x^*_i by:

$$\Delta U_i(x^*) \equiv U\left[\hat{y}^*|x^*, D\right] - U\left[\hat{y}^*|x^*_{-i}, D\right] \tag{4.4}$$

in which U is a placeholder for epistemic, aleatoric, or predictive uncertainty. Similar to before, $U\left[c|x^*\right]$ can be evaluated via forward passes through the BNN, while the evaluation of $U\left[c|x^*_{-i}\right]$ requires marginalisation over x_i. With BNNs there is also an additional marginalisation over weights.

Predictive Uncertainty

Recall that the predictive uncertainty on an input image x^* is the entropy of the categorical softmax after marginalising over the weight settings. Thus, the change in predictive uncertainty due to feature x_i^* is:

$$\Delta U_{i,\text{predictive}}(x^*) = \mathbb{H}\left[\hat{y}^*|x^*, D\right] - \mathbb{H}\left[\hat{y}^*|x^*_{-i}, D\right] \tag{4.5}$$

First, from (2.20), we evaluate $\mathbb{H}\left[\hat{y}^*|x^*, D\right]$ when all features are known by MC sampling $w^k \sim q(w)$:

$$\mathbb{H}\left[\hat{y}^*|x^*, D\right] \approx -\sum_c \left(\frac{1}{K}\sum_k p(\hat{y}^* = c|x^*, w^k)\right) \log\left(\frac{1}{K}\sum_k p(\hat{y}^* = c|x^*, w^k)\right) \tag{4.6}$$

Second, when the feature x_i^* is unknown, the derivation for predictive uncertainty is:

$$\mathbb{H}\left[\hat{y}^*|x^*_{-i}, D\right] = -\sum_c p(\hat{y}^* = c|x^*_{-i}, D) \log p(\hat{y}^* = c|x^*_{-i}, D)$$

Expanding out $p(\hat{y}^* = c|x^*_{-i}, D)$ as a marginalisation over weights w and the feature x_i:

$$\begin{aligned} &= -\sum_c \left[\int\int p(\hat{y}^* = c, x_i, w|x^*_{-i}, D) dx_i dw\right] \\ &\qquad \times \log\left[\int\int p(\hat{y}^* = c, x_i, w|x^*_{-i}, D) dx_i dw\right] \end{aligned}$$

Decomposing the joint distribution:

$$\begin{aligned} &= -\sum_c \left[\int\int p(\hat{y}^* = c|x^*_{-i}, x_i, w) p(x_i|x^*_{-i}) p(w|D) dx_i dw\right] \\ &\qquad \times \log\left[\int\int p(\hat{y}^* = c|x^*_{-i}, x_i, w) p(x_i|x^*_{-i}) p(w|D) dx_i dw\right] \end{aligned}$$

Replacing the weight posterior with the variational distribution:

$$\approx -\sum_c \left[\int \int p(\hat{y}^* = c|x^*_{-i}, x_i, w) p(x_i|x^*_{-i}) q(w) dx_i dw \right] \times \log \left[\int \int p(\hat{y}^* = c|x^*_{-i}, x_i, w) p(x_i|x^*_{-i}) q(w) dx_i dw \right]$$

Finally, replacing the integration with K MC samples of the weights and M samples of x_i from $w^k \sim q(w)$ and $x_i^m \sim p(x_i|x^*_{-i})$, we have that:

$$\mathbb{H}\left[\hat{y}^*|x^*_{-i}, D\right] \approx -\sum_c \left[\frac{1}{MK} \sum_{m=1}^{M} \sum_{k=1}^{K} p(\hat{y}^* = c|x_{-i}, x_i^m, w^k) \right] \times \log \left[\frac{1}{MK} \sum_{m=1}^{M} \sum_{k=1}^{K} p(\hat{y}^* = c|x_{-i}, x_i^m, w^k) \right] \tag{4.7}$$

Aleatoric Uncertainty

Aleatoric uncertainty is uncertainty from the input when the weights are held fixed. In classification, aleatoric uncertainty comes from the softmax; it measures the uncertainty of the input in belonging to each class given a particular weight setting. The change in aleatoric uncertainty due to feature x_i^* is:

$$\Delta U_{i,\text{aleatoric}}(x^*) = E_{w\sim p(w|D)}\left[\mathbb{H}\left[\hat{y}^*|x^*, w\right]\right] - E_{w\sim p(w|D)}\left[\mathbb{H}\left[\hat{y}^*|x^*_{-i}, w\right]\right] \tag{4.8}$$

From (2.21), when we know x_i^*, we draw samples $w^k \sim q(w)$ to compute the aleatoric uncertainty:

$$E_{w\sim p(w|D)}\left[\mathbb{H}\left[\hat{y}^*|x^*, w\right]\right] \approx -\frac{1}{K}\sum_k \left[\sum_c p(\hat{y}^* = c|x^*, w^k) \log p(\hat{y}^* = c|x^*, w^k) \right] \tag{4.9}$$

When we do not know x_i^*, the derivation for aleatoric uncertainty is:

$$E_{w\sim p(w|D)}\left[\mathbb{H}\left[\hat{y}^*|x^*_{-i}, w\right]\right] \approx E_{w\sim q(w)}\left[-\sum_c p(\hat{y}^* = c|x^*_{-i}, w) \log p(\hat{y}^* = |x^*_{-i}, w) \right]$$

Expanding the $p(\hat{y}^* = c|x^*_{-i}, w)$ marginal distribution:

$$= E_{w\sim q(w)}\left[-\sum_c \left(\int p(c|x_i, x^*_{-i}, w)p(x_i|x^*_{-i})dx_i\right) \times \log\left(\int p(c|x_i, x^*_{-i}, w)p(x_i|x^*_{-i})dx_i\right)\right]$$

Replacing the integration over x_i with M samples $x_i^m \sim p(x_i|x^*_{-i})$:

$$\approx E_{w\sim q(w)}\left[-\sum_c \left(\frac{1}{M}\sum_m p(c|x_i^m, x^*_{-i}, w)\right) \times \log\left(\frac{1}{M}\sum_m p(c|x_i^m, x^*_{-i}, w)\right)\right]$$

Finally, replacing the integration over w with K samples $w^k \sim q(w)$:

$$E_{w\sim p(w|D)}\left[\mathbb{H}\left[\hat{y}^*|x^*_{-i}, w\right]\right] \approx -\frac{1}{K}\sum_k \left[\sum_c \left(\frac{1}{M}\sum_m p(c|x_i^m, x^*_{-i}, w^k)\right) \times \log\left(\frac{1}{M}\sum_m p(c|x_i^m, x^*_{-i}, w^k)\right)\right] \tag{4.10}$$

Note that computing (4.10) involves drawing K samples of weights and M samples for x_i^m, but all M samples for x_i^m are evaluated with the *same* weight sample w^k. This process is repeated K times, once for each weight sample w^k.

Epistemic Uncertainty

Taking the epistemic uncertainty as the difference between predictive and aleatoric uncertainties, the change in epistemic uncertainty due to feature x_i^* is:

$$\begin{aligned}\Delta U_{i,\text{epistemic}}(x^*) = &\left(\mathbb{H}(\hat{y}^* = c|x^*) - E_{w\sim p(w|D)}\left[\mathbb{H}\left[\hat{y}^* = c|x^*, w\right]\right]\right) \\ &- \left(\mathbb{H}(\hat{y}^* = c|x^*_{-i}) - E_{w\sim p(w|D)}\left[\mathbb{H}\left[\hat{y}^* = c|x^*_{-i}, w\right]\right]\right)\end{aligned} \tag{4.11}$$

Expressions for the four terms in (4.11) can be estimated by applying (4.6) and (4.7) for predictive uncertainty, and (4.9) and (4.10) for aleatoric uncertainty.

4.1.3 Implementation Details

We follow the improvements demonstrated by Zintgraf et al. [69] for evaluating predictive difference on images, and apply these ideas to compute $\Delta U_i(x^*)$ due to individual pixels of a test image.

The first heuristic by Zintgraf et al. [69] involves the estimation of $p(x_i|x^*_{-i})$. Rather than estimating pixel x_i given the values of all other pixels x^*_{-i} in the image, we restrict the dependencies of pixel x_i to only a local neighbourhood around x_i. Thus, we approximate $p(x_i|x^*_{-i}) \approx p(x_i|\hat{x}^*_{-i})$ in which $\hat{x}^*_{-i}$ refers to those pixels in an $l \times l$ patch around the i-th pixel, rather than all remaining pixels. The extent of l is a hyperparameter. This treats the pixels near x_i as a Markov blanket, and the remaining image pixels are conditionally independent of x_i given the local neighbours. This approximation is sensible for images because the intensities of pixels in an image are highly correlated with nearby pixels, but do not depend on their position within the overall image (which would be the case if we took x^*_{-i} as all remaining pixels). We approximate the joint distribution of x_i and its local neighbourhood patch $\hat{x}_{-i}$ as a multivariate Gaussian:

$$p(x_i, \hat{x}_{-i}) = \mathcal{N}\left(\begin{bmatrix} x_i \\ \hat{x}_{-i} \end{bmatrix}; \mu, \Sigma\right) = \mathcal{N}\left(\begin{bmatrix} x_i \\ \hat{x}_{-i} \end{bmatrix}; \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right) \tag{4.12}$$

in which the parameters μ and Σ are estimated from patches of the training images, with μ a vector of length $l \times l$ for an $l \times l$-sized patch, and Σ matrix with $(l \times l)^2$ elements. Then, given a neighbouring pixels of a test image, $\hat{x}^*_{-i}$, we estimate the conditional distribution of feature x_i as:

$$p(x_i|\hat{x}^*_{-i}) = \mathcal{N}\left(x_i; \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\hat{x}^*_{-i} - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right) \tag{4.13}$$

The second heuristic introduced by Zintgraf et al. [69] is multivariate analysis, rather than univariate analysis in [53]. Instead of marginalising over a single pixel x_i, we marginalise over a patch of $k \times k$ adjacent pixels. This is motivated by the observation that neural network outputs are fairly robust to perturbations in a single pixel, and that individual pixels are often surrounded locally by pixels of similar colour. Thus, it is expected that obscuring a larger patch of $k \times k$ pixels via marginalisation affects resulting uncertainty more than obscuring a single pixel. Then, $\Delta U_i(x^*)$ is taken as the average contribution of all patches containing the i-th pixel. With this approach, the user determines the size of the feature patch; however if k is small the salience maps capture finer textures in the image, and if k is too large the salience maps are blurred [69]. Thus, the size of the feature patch can be selected to

roughly approximate the size of key patterns in the image. Figure 4.1 illustrates the relative positioning of the $k \times k$ feature patch and the $l \times l$ neighbourhood patch in the input image.

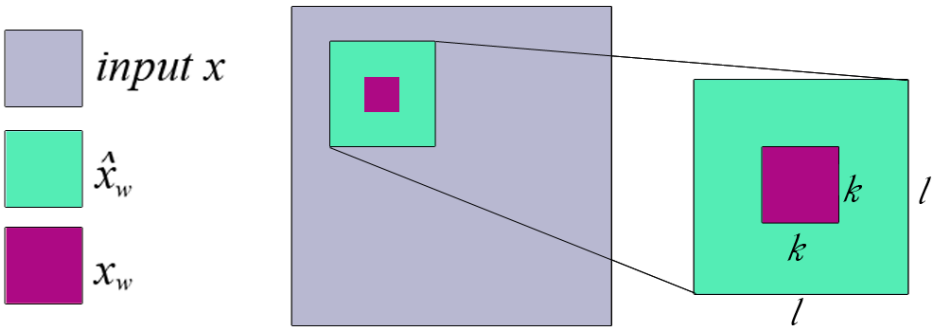


Fig. 4.1 Conditional sampling approximates $p(x_i|x^*_{-i})$ using a local $l \times l$ patch (in green) rather than the full image (in grey). Multivariate analysis marginalises over a $k \times k$ feature patch (in magenta) rather than an individual pixel x_i, such that $k < l$. Image from [69].

4.2 Bayesian Neural Network Implementation

The above approach in computing $\Delta U_i(x^*)$ does not rely on a specific implementation of BNN; it only requires weight samples from a variational distribution and stochastic feed-forward passes that yield softmax activations.

In this work we use the BNN implementation from Li and Gal [42], which reparametrises the α-divergence objective – rather than KL-divergence typically used in variational inference – to be compatible with the dropout approximate variational distribution. Combining these two ideas with the assumption of sufficient data such that $\alpha << N$, the relevant α-divergence objective with cross entropy loss becomes [42]:

$$\mathscr{L}_\alpha(q) \approx \frac{1}{\alpha}\sum_n y^{(n)T} * \log\left(\frac{1}{K}\sum_k p(\hat{y}^{(n)}|x^{(n)}, w^k)^\alpha\right) + L_2(m) \tag{4.14}$$

in which $y^{(n)T}$ refers to a one-hot vector of labels for the n-th training input, $p(\hat{y}^{(n)}|x^{(n)}, w^k)$ refers to the $C \times 1$ vector of predicted softmax activation over C classes using the n-th training input and the k-th sample of the weights, and $L_2(m)$ is an L_2 regularisation term on the weights without dropout, a result of approximating the divergence between the variational and prior distributions [24]. For a full derivation of the objective function refer to [42]. We select this implementation of BNN due to the practical properties of the dropout approximate variational distribution in the number of variational parameters, and the improved uncertainty

estimates offered by α-divergence; however, the same approach of calculating $\Delta U_i(x^*)$ can be performed with any BNN implementation.

We minimise the objective function by performing $K = 10$ stochastic forward passes at training time and using automatic differentiation to update the variational parameters. Specifically, we use a Stochastic Gradient Descent optimiser with a learning rate of 0.01 and Nesterov momentum parameter of 0.9. We use $\alpha = 0.5$, which is shown to yield improved predictions over $\alpha = 0$ (standard variational inference objective) and $\alpha = 1$ (expectation propagation objective) [31, 15]. Our model architecture is identical to that of [42], containing two 2D-convolutional layers of 32 filters and a 3×3 kernel with ReLU activation, followed by a 2×2 max-pooling layer, and two Dense layers with dropout.

Training is terminated when the model reaches a minimum loss on a validation set, taken as a reserved 10% of the training images (Fig. 4.2). We replicate the training procedure three times, each yielding similar results.

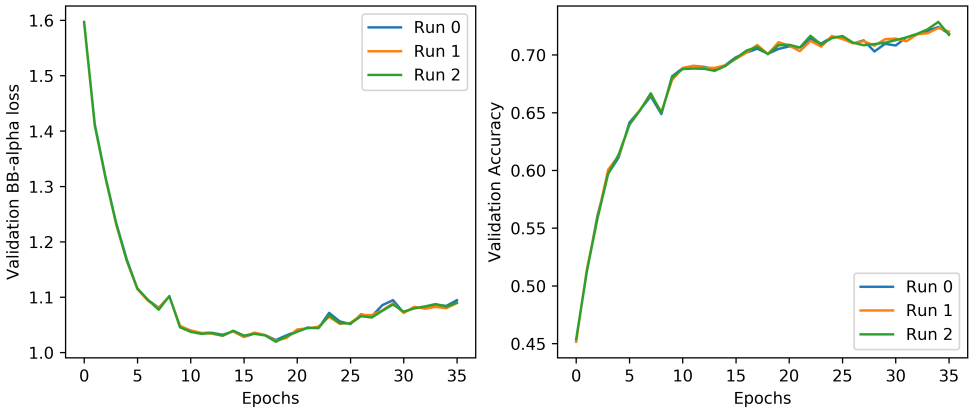


Fig. 4.2 Validation loss (left) and validation accuracy (right) over training epochs. BNN models with the dropout approximate variational distribution and the α-divergence objective are trained on the CIFAR10 dataset. The minimum validation loss is reached at 18 epochs. Accuracy on the test dataset is 71.5±7.4e-4%, but we focus on demonstrating interpretability rather than optimising accuracy.

At test time, we perform $K = 100$ stochastic forward passes using dropout as samples from the variational distribution, similar to [42]. Note that in the computation of aleatoric uncertainty when feature x_i is unknown, we require using the same weight sample over multiple samples of the missing feature x_i. In Tensorflow, this can be accomplished by setting the `noise_shape` argument in dropout to use the same dropout mask over images containing the x_i samples [2].

4.3 Datasets

A commonly used dataset for machine learning classification is the CIFAR10 dataset [39], which is a collection of 60,000 colour images belonging to ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. All images are 32×32 pixels in size. There are 10,000 test images, with 1,000 examples from each class, and the remaining 50,000 images are used for training. Of the training images, 10% were reserved as a validation set during training of the BNN model.

A related dataset is the CIFAR100 dataset [39], also containing 60,000 32×32 colour images. Each images belongs to one of 100 "fine" classes, and one of 20 "coarse" classes, which are disjoint from the classifications of CIFAR10. Examples from the CIFAR100 dataset are used only when testing the model, after the model is trained on images from CIFAR10, to demonstrate uncertainty in classifying out-of-distribution images.

A third dataset used in this work is the ISIC 2018 Lesion Diagnosis Challenge training dataset for skin cancer [11, 60]. This publicly-available dataset contains 10015 images of skin lesions spanning seven cancerous and benign categories: Nevus (NV), Dermatofibroma (DF), Melanoma (MEL), Actinic keratosis (AKIEC), Benign keratosis (BKL), Basal cell carcinoma (BCC), and Vascular lesion (VASC). All images are 600×450 JPEG images, and are resized to 150×113 pixels for faster computation in experiments. Note that not all skin conditions are equally represented in this dataset. For example, there are 6705 examples of the Nevus category, and 115 examples of the Benign keratosis category. For experiments here, we selected the NV, MEL, BKL, and BCC categories, the four groups with the most examples (6705, 1113, 1099, and 514 respectively), and performed naive random subsampling to equalise the number of examples in each category. We leave experimentation with more sophisticated data resampling techniques, such as SMOTE [9] or ADASYN [29], and additional disease categories for future work.

All images were 3-channel 8-bit RGB images, thus having intensity values of each pixel ranging from zero to 255. Prior to model training and evaluation, all intensity values were divided by 255, such that the range for each pixel in each colour channel was between zero and one.

Chapter 5

Results

This chapter begins with two sections of validation experiments to motivate a Bayesian approach. First, unlike DNNs, BNNs integrate over weights to yield more uncertain predictions when both models are incorrect (§5.1). Second, BNNs provide uncertainty estimates in their decisions, which can be decomposed into epistemic and aleatoric uncertainty – each type of uncertainty is reduced differently, but this cannot be addressed when the uncertainties are combined (§5.2). In §5.3, we show results on changes in uncertainty due to individual pixels on the CIFAR10 and ISIC2018 datasets. We also compare the uncertainty-based approach to the method of predictive difference [69].

5.1 Uncertainty in Bayesian Neural Networks

Softmax output probabilities are one way of measuring uncertainty in classification tasks. A softmax output that places all probability mass on a single class can be interpreted as "confident" in its decision, and one that divides probability mass equally among all classes can be seen as "unconfident." The entropy of the softmax output distribution describes this confidence, with the former having low entropy, and the latter having high entropy. In BNNs after we integrate over the weight distribution, entropy in the softmax output is precisely the predictive uncertainty.

Here, we compare the softmax output of DNNs and BNNs when evaluated on a test instances outside the training distribution. We train DNNs and BNNs with identical model architecture on the CIFAR10 dataset, and evaluate them on images from CIFAR100. These two datasets do not contain overlapping classes.

In Fig. 5.1, we show DNN and BNN predictions on a "large natural outdoor scene" from CIFAR100. The DNN predicts the image as a "bird," placing 80% of the softmax output on that class. Because the DNN does not have dropout at test time, its predictions are

deterministic. The BNN also predicts erroneously because none of the CIFAR10 classes fit the input. However, the predictive softmax, $p(\hat{y}^* = c|x^*)$, is the average output over multiple samples from the weight distribution (error bars show one standard deviation over 100 weight samples). Because the BNN aggregates the softmax outputs over multiple draws from the variational weight distribution, its predictive softmax output is less peaked than that of the DNN.

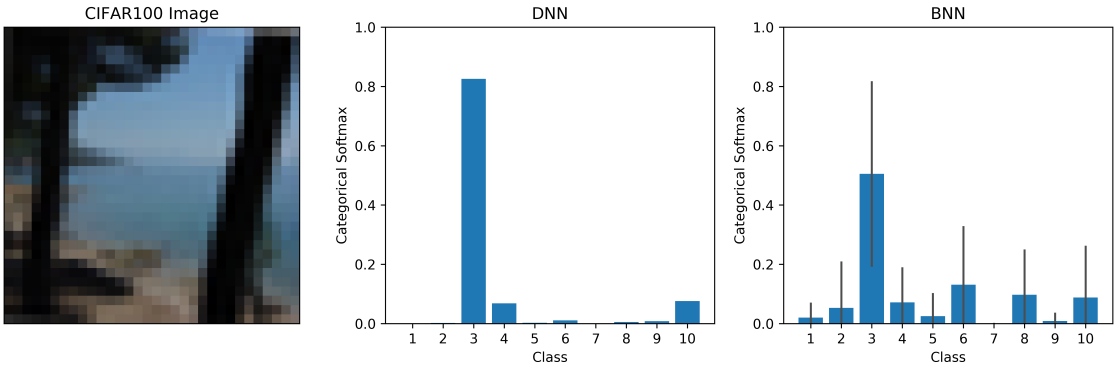


Fig. 5.1 DNN and BNN predictions on an out-of-distribution image. (Left) The out-of-distribution CIFAR100 image. (Middle) Softmax output of a DNN trained on CIFAR10. (Right) A BNN (also trained on CIFAR10) provides a distribution over softmax predictions by taking samples from a weight distribution. The predictive softmax (in blue) involves MC integration over the weights. Errorbars represent one standard deviation across weight samples.

Next, we took the first 100 images of the CIFAR100 test set and performed a similar experiment. We compared the softmax probability mass assigned to the predicted class (taken as the maximum softmax activation over all classes) and the entropy of the predictive softmax across both models (Fig. 5.2). DNNs tend to place more probability mass on the incorrectly predicted class compared to BNNs, resulting in higher maximal softmax activations (most points fall below the diagonal). On the other hand, the predictions of BNNs marginalise over a weight distribution, leading to more entropic softmax distributions than DNNs (most points fall above the diagonal).

These results suggest that, when both models are forced to make an erroneous decision on out-of-distribution image, the predictive softmax distribution captures higher entropy and more uncertainty in BNNs than DNNs, because BNNs consider predictions over the weight distribution. The predictions of DNNs are more confident because the model uses a point estimate of weights.

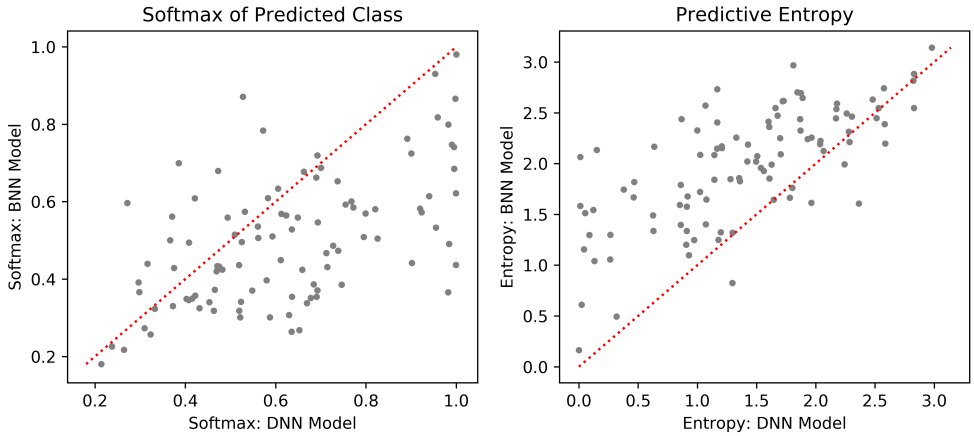


Fig. 5.2 Comparing DNN and BNN predictions. (Left) Softmax activation of the predicted class (taken as the maximal softmax activation over all classes) for a BNN and a DNN on the first 100 test images in CIFAR100. (Right) Entropy of the softmax distribution for a BNN and DNN on the same test images. Both models are trained on CIFAR10; there is no overlap with classes in CIFAR100.

5.2 Uncertainty Decomposition

The entropy of the softmax predictive distribution alone is insufficient to fully characterise uncertainty in a BNN because there is uncertainty from the model weights and from the classification of the input. Using different weight settings could produce very confident, yet disagreeing decisions [27], but after performing MC integration over the weights, we cannot distinguish the uncertainty in the weights from the intrinsic uncertainty in the input. Uncertainty in the weights is exclusive to BNNs; in DNNs there is no concept of a weight distribution.

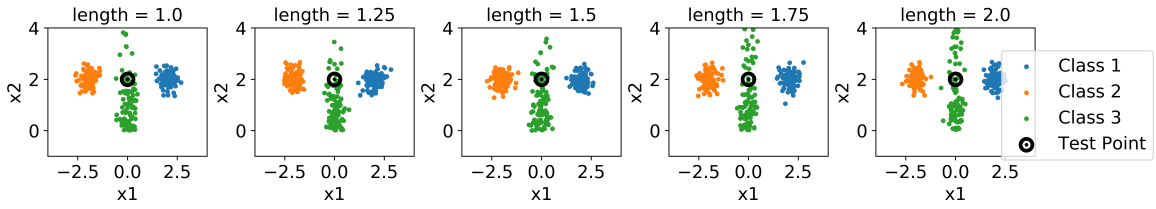
Recall from §2.3 that epistemic uncertainty, measured by the the information gain between the weights and the model output, captures the uncertainty in the weights. Intuitively, epistemic uncertainty measures the deviation in the output using different weight settings. Aleatoric uncertainty captures our uncertainty in the classification of the input with a fixed weight setting, i.e. if there exists enough information in the features to fully classify the output. Empirically, epistemic uncertainty is high for test points far from the training distribution, while aleatoric uncertainty is high for test points on the decision boundaries [44, 26].

In the following experiments with synthetic data, we motivate why epistemic and aleatoric uncertainties should be addressed separately. Epistemic uncertainty is reduced with more

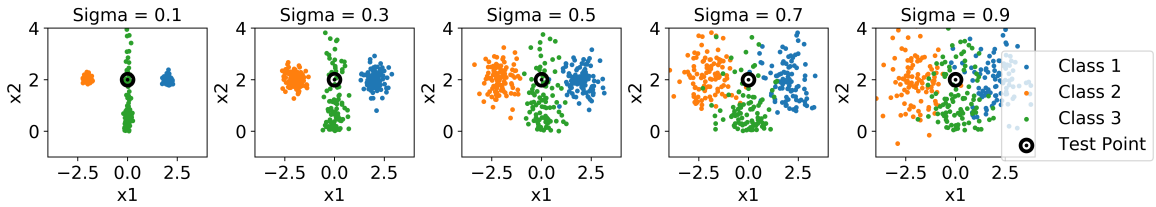
data, but aleatoric uncertainty is reduced with more knowledge about latent variables. We cannot make these distinctions from predictive uncertainty alone.

5.2.1 Uncertainty Decomposition with Synthetic Data

We generate 2-dimensional synthetic data belonging to three classes. In the first experiment, we isolate epistemic uncertainty. Here, the first and second classes are isotropic Gaussians with $\mu_1 = (2,2)$, $\mu_2 = (-2,2)$, and identical $\sigma = 0.25$. The third class is Normally distributed in the first feature with $\mu = 0$ and $\sigma = 0.25$, and exponentially distributed in the second feature. When we increase the lengthscale of the exponential distribution from $l = 1$ to $l = 2$, there is greater probability mass on points further from zero (Fig. 5.3a). In the second experiment, we isolate aleatoric uncertainty. We fix the lengthscale of the exponential distribution at $l = 1.5$ and vary the spread of the classes between $\sigma = 0.1$ and $\sigma = 0.9$ (Fig. 5.3b). The means of all normally distributed features remain the same as before.



(a) Increasing Lengthscale



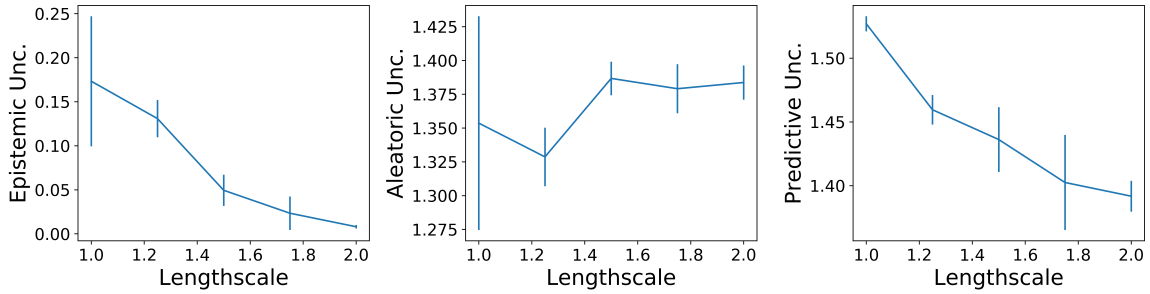
(b) Increasing Spread

Fig. 5.3 (A) To isolate epistemic uncertainty, we increase the lengthscale of the exponential distribution for the third class. (B) To isolate aleatoric uncertainty, we increase the spread of all classes.

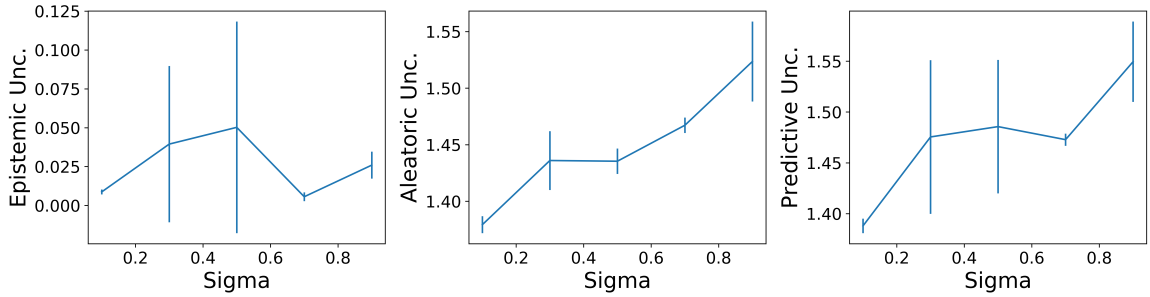
For both experiments, we train networks with two fully-connected layers with the α-divergence objective and dropout at training and test time to simulate samples from the weight distribution. For every value of the lengthscale l and the spread σ of the training distribution, we train three model replicates, and evaluate the epistemic and aleatoric uncertainty of a test point at $(0,2)$.

As the lengthscale of the training distribution increases, the epistemic uncertainty decreases, but aleatoric uncertainty remains similar (Fig. 5.4a). This is consistent with the

notion of higher epistemic uncertainty capturing uncertainty in regions where data is scarce. Therefore, epistemic uncertainty at a test point is reduced when there is more data in the vicinity of that point. On the other hand, aleatoric uncertainty measures latent interactions intrinsic in the data that are not captured by the model. For example, when the spread of the class distributions is large, the distributions overlap at the test point so there is insufficient information to classify it. Aleatoric uncertainty is reduced by more informative measurements, such as reducing the spread of the class distributions (Fig. 5.4b). Note that in this synthetic dataset we can reduce aleatoric uncertainty by reducing the noise in our measurements, but in real-life datasets or complex processes, aleatoric uncertainty is irreducible when we do not know the latent variables involved in the inputs. In both cases, predictive uncertainty decreases as we increase l or reduce σ, so from predictive uncertainty alone we cannot distinguish uncertainty due to model weights from uncertainty inherent in the input.



(a) Increasing Lengthscale



(b) Increasing Spread

Fig. 5.4 (A) Epistemic uncertainty decreases as the lengthscale increases, but aleatoric uncertainty remains similar. (B) Aleatoric uncertainty decreases as the spread decreases; epistemic uncertainty remains similar. Errorbars show one standard deviation over model replicates.

5.2.2 Uncertainty Decomposition in CIFAR10

Next we show the decomposition of uncertainty on CIFAR10 test images. Similar to before, we decompose the predictive uncertainty into its epistemic and aleatoric components following the derivations in §2.3. We plot these uncertainties against the softmax output of the predicted class, which is the maximum softmax activation across all classes (Fig. 5.5). The predictive softmax of a BNN, $p(\hat{y}^* = c|x^*)$, involves averaging over weight samples, so attaining a maximum $p(\hat{y}^* = c|x^*) = 1$ means that all weights samples place softmax probability mass entirely on one class, and all weight samples agree on the same class. Therefore, to attain $p(\hat{y}^* = c|x^*) = 1$, all three types of uncertainty are zero.

On the other hand, when the softmax activation of the predicted class is low, the predictive uncertainty contains contributions from both epistemic and aleatoric sources. For example, on test images when the softmax activation of the predicted class is near 0.2 (on the y-axis), we observe a range of values for epistemic and aleatoric uncertainties (see Fig. 5.5). This suggests that when the predictions are uncertain, this may be due to seeing an example unlike previously seen images (epistemic), or seeing an example which is difficult to categorise into a single class with the given features (aleatoric). Again, these differences are obscured when solely looking at predictive uncertainty.

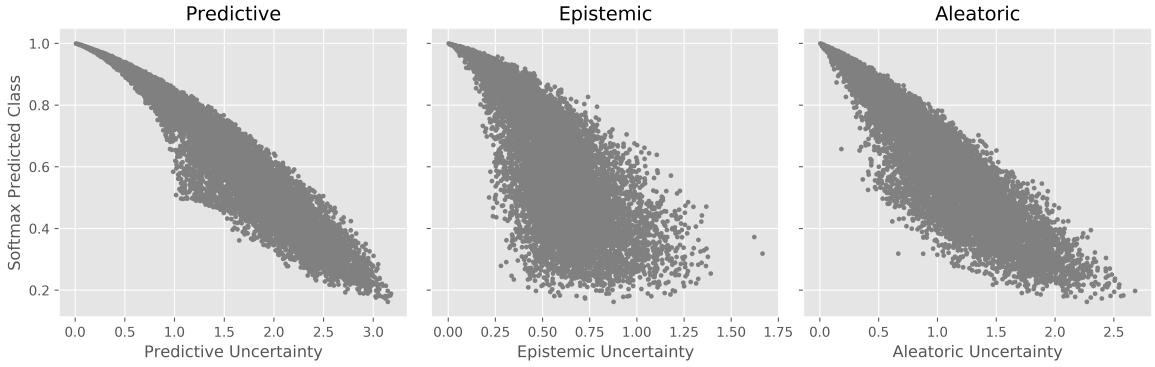


Fig. 5.5 Softmax output of the predicted class) against predictive (left), epistemic (middle), and aleatoric (right) uncertainties on CIFAR10 test images. Each point represents a test image.

5.3 Mapping Uncertainty onto Input Pixels

Following the methodology in Chapter 4, we visualise the contributions of individual pixels to epistemic, aleatoric, and predictive uncertainties. Because we gain additional insights by decomposing overall predictive uncertainty into its epistemic and aleatoric components, we

generate salience maps for each type of uncertainty separately. Our approach computes the change in uncertainty due to individual pixels via:

$$\Delta U_i(x^*) \equiv U\left[\hat{y}^*|x^*, D\right] - U\left[\hat{y}^*|x^*_{-i}, D\right] \tag{5.1}$$

A positive change (shown in red) means that knowing the pixel value makes the decision *more uncertain* and uncertainty increases, while a negative change (in blue) means that knowing the pixel value makes the decision *more confident* and uncertainty decreases.

5.3.1 Epistemic and Aleatoric Uncertainty

We start by investigating $\Delta U_{i,epistemic}(x^*)$ and $\Delta U_{i,aleatoric}(x^*)$ using parameters $k = 8$ and $l = 10$ for patch size and padding, respectively (see §4.1.3).

Images that are difficult to classify

In Fig. 5.6 we show the results for CIFAR10 test images with high predictive uncertainty. In these images, the softmax output after integrating over weights has high entropy, so the softmax probability assigned to the predicted class is low. Examining these salience maps shows us the pixels contributing to uncertainty, despite a correct or incorrect classification.

Aleatoric uncertainty is reduced over key identifying parts of the image, while epistemic uncertainty is increased over the background. This suggests that both the background and foreground influence uncertainty. The reduction in aleatoric uncertainty over key objects is sensible because when these objects are unknown, the remaining pixels do not provide enough information to fully classify the image regardless of the weight setting. The increased epistemic uncertainty over the background suggests that the background in the image is unlike those seen in training, so epistemic uncertainty is higher when these pixels are known. Thus, the predictions on these images may be uncertain due to unconventional backgrounds. We further discuss these trends in the following paragraphs.

In example (A) of 5.6, the ship's bow reduces aleatoric uncertainty, suggesting that knowing these pixels helps to predict the class (on individual weight settings). The surrounding water increases epistemic uncertainty. Because many ship images in the CIFAR10 dataset appear on blue backgrounds [65], it is plausible that a grey background is not seen often in training, thus increasing epistemic uncertainty in this region.

Examples (D) and (E) are instances of incorrect predictions, and the salience maps offers insight on pixels contributing to uncertainty. In (D) the true label is "cat", and the prediction is "automobile," but the image contains a large mechanical object. Aleatoric uncertainty

is reduced over the mechanical object, suggesting that these pixels provide information to classify the image. The cat increases epistemic uncertainty, suggesting that knowledge of these pixels, together with the foreground object, makes the image unlike training examples. In example (E) the true label is "horse" and the prediction is "deer". The salience map for aleatoric uncertainty shows us that the horse head and the human's legs (erroneously) help to classify the image on individual weight samples, but epistemic uncertainty increases over the fence, suggesting that fences are infrequent in the training data.

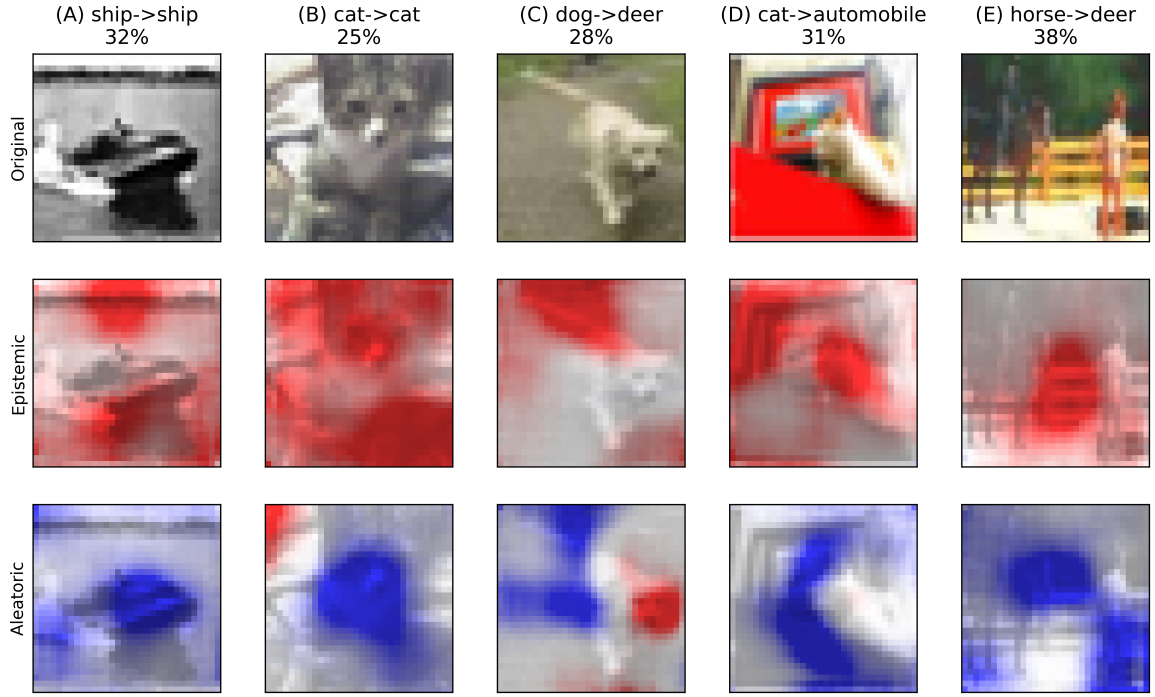


Fig. 5.6 Visualisation of epistemic and aleatoric uncertainties for selected CIFAR10 test images. The original images, with the true class, predicted class, and softmax activation of the predicted class, are in the top row. $\Delta U_{i,epistemic}(x^*)$ and $\Delta U_{i,aleatoric}(x^*)$ are in the middle and bottom rows, respectively. These examples are images with high predictive uncertainty. Colorscale contrast is increased in the salience maps for improved visibility. These overlaid images highlight areas of greatest uncertainty change; for raw magnitudes of $\Delta U_{i,epistemic}(x^*)$ and $\Delta U_{i,aleatoric}(x^*)$ see Fig. B.2.

Images that are easily classified

Next we turn to examples with low predictive uncertainty (Fig. 5.7). These images are classified correctly, and the softmax activation of the predicted class is near 100%. The salience maps for both uncertainties are similar, and uncertainty is reduced over key identifying regions such as the horse's legs or the automobile's wheel. This suggests that for images

that are confidently predicted, the same pixels that provide information for the classification also make the test instance more similar to the training distribution, and when we do not know these pixels, the image is more unlike those seen in training. Obscuring background pixels does not change uncertainty when the main object is known.

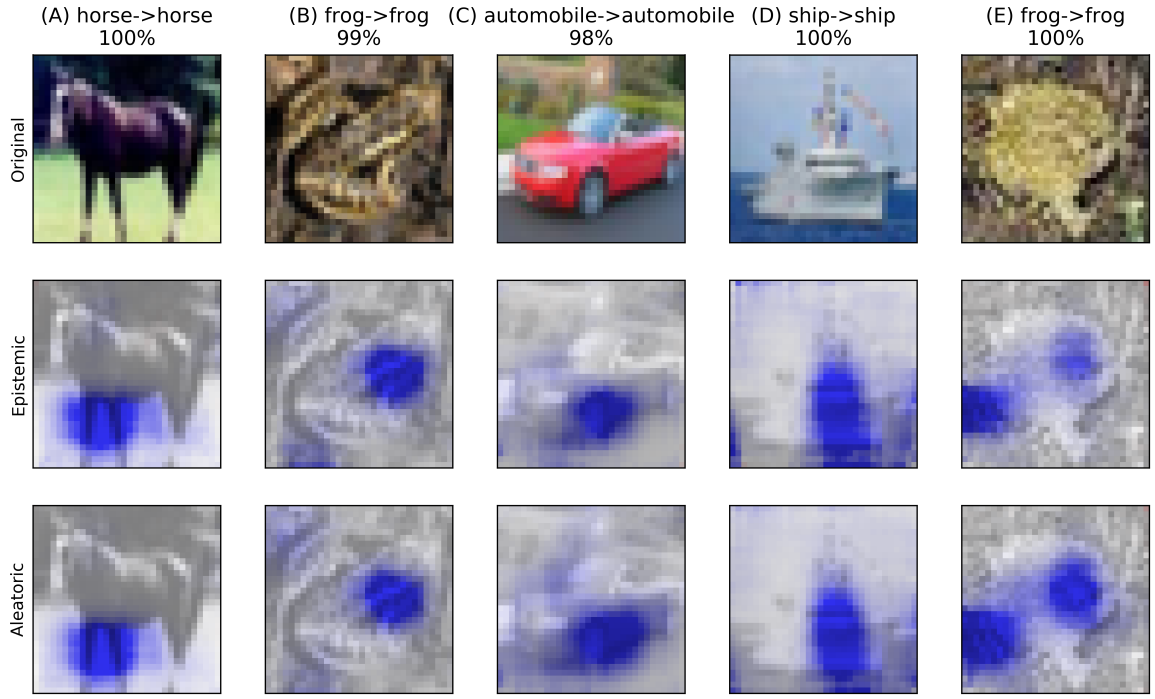


Fig. 5.7 Visualisation of epistemic and aleatoric uncertainties for selected CIFAR10 test images with low predictive uncertainty. The original images with the true class, predicted class, and categorical softmax activation of the predicted class, are in the top row. $\Delta U_{i,epistemic}(x^*)$ and $\Delta U_{i,aleatoric}(x^*)$ are in the middle and bottom rows, respectively. Colorscale contrast is increased in the salience maps for improved visibility. See also Fig. B.2.

From these examples, it may be most informative to consider the salience maps of examples for which the predictive uncertainty is high, i.e. borderline cases, which show a separation of the uncertainty into epistemic and aleatoric components. $\Delta U_{i,epistemic}(x^*)$ highlights regions that differentiate an image from training instances, and $\Delta U_{i,aleatoric}(x^*)$ highlights regions contributing to or detracting from the classification of the image with individual weight settings.

5.3.2 Comparison to Predictive Difference

For the same images in Fig. 5.6, we generated salience maps for $\Delta U_{i,predictive}(x^*)$. Predictive uncertainty is the sum of epistemic and aleatoric uncertainties, and the predictive uncertainty

salience map is the composite of the previous epistemic and aleatoric salience maps (Fig. 5.8). $\Delta U_{i,predictive}(x^*)$ captures both the regions that reduce aleatoric uncertainty, and regions that increase epistemic uncertainty.

We compare the predictive uncertainty salience maps to visualisations of predictive difference [69]. In predictive difference, red indicates *evidence for* a class, and blue indicates *evidence against*. Note that predictive difference requires specification of a target class, and evidence for/against is relative to the target class. In Figure 5.8 the predictive difference is computed relative to the predicted class (the class with the maximal softmax activation).

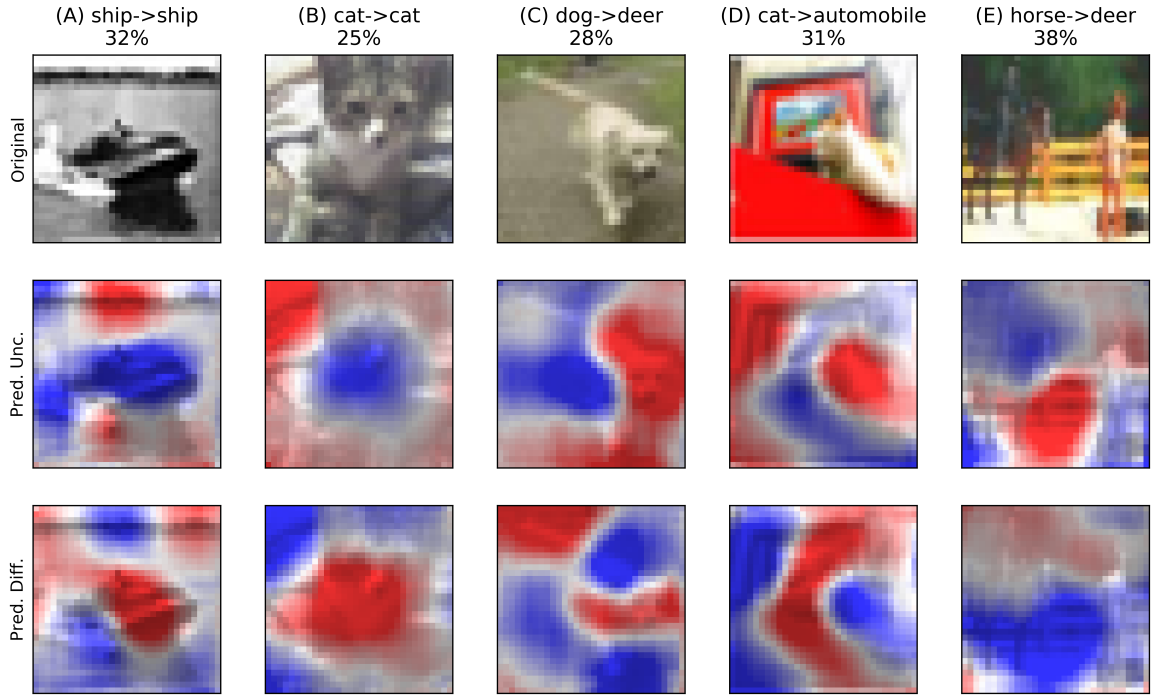


Fig. 5.8 Visualisation of predictive uncertainty (middle) and predictive difference (bottom) as implemented in [69]) of the same CIFAR10 images in Fig. 5.6.

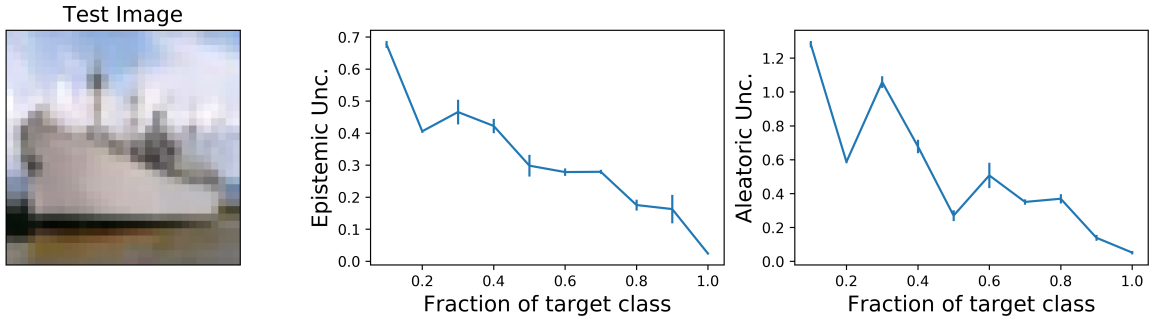
Here, regions that provide evidence for a particular class reduce the predictive uncertainty, and regions that are evidence against increase the predictive uncertainty. This suggests that pixels that reduce the softmax activation of a particular class also increase entropy in the overall softmax distribution. One interesting exception is example (C), in which the dog's face contributes evidence for the deer classification, but it also increases the predictive uncertainty. This suggests that, despite using the dog's face incorrectly to classify the image as a deer, there is also greater uncertainty in this region.

Because predictive difference is computed relative to a target class, when the classifier is incorrect, it may be showing evidence for/against the wrong class. This itself is still

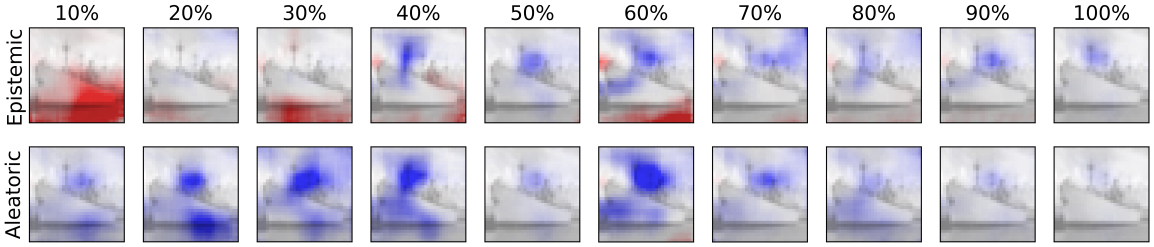
informative in understanding why the classifier is incorrect. However, if predictive uncertainty is high and the softmax output is near-uniform, the predictive difference approach omits crucial information about the remaining classes. $\Delta U_{i,predictive}(x^*)$ summarises the effect of individual pixels on the entire softmax *distribution* rather than a single class, and avoids the alternative of visualising all classes separately.

5.3.3 Adding More Training Data

We seek to visualise how $\Delta U_{i,epistemic}(x^*)$ and $\Delta U_{i,aleatoric}(x^*)$ change as more training data is added in the CIFAR10 dataset. For this experiment, we subsample the training examples for the "ship" class in increments of 10% while using all examples for the remaining classes to simulate adding more data similar to the "ship" test images.



(a) Epistemic and Aleatoric Uncertainties



(b) Salience Maps

Fig. 5.9 (A) The test image (left) and epistemic (middle) and aleatoric (right) uncertainties as more training data in the "ship" class is added. Errorbars show one standard deviation over three model replicates. (B) Visualisations of $\Delta U_{i,epistemic}(x^*)$ and $\Delta U_{i,aleatoric}(x^*)$ with more training data. Colorscale limits are the same across all plots each row.

On a correctly-predicted test image, the epistemic uncertainty decreases as more "ship" examples are seen in training (Fig. 5.9a). Interestingly, there is also a decrease in aleatoric uncertainty. This may happen if there is not enough information to classify the instance

inherent in the model prior, so as we add more data, we shift further away from the prior and the model better captures the information inherent in the input.

With just 10% of the ship examples in training, $\Delta U_{i,epistemic}(x^*)$ is positive in many pixels of the image. Because there are fewer examples of ships in training, slight perturbations in any direction may result in the test example moving farther training distribution due to the limited data in that class. As we add data, $\Delta U_{i,epistemic}(x^*)$ decreases to zero in these pixels, suggesting that perturbing these pixels no longer moves it out of the training distribution. Also, $\Delta U_{i,epistemic}(x^*)$ becomes negative in other pixels that help classify the image. Positive $\Delta U_{i,epistemic}(x^*)$ highlights regions of an image that make it unlike training examples, and Fig. 5.9b suggests that we can eliminate these regions by adding more training examples.

In $\Delta U_{i,aleatoric}(x^*)$, as we add data the predictions become more robust to perturbations – the magnitude of $\Delta U_{i,aleatoric}(x^*)$ decreases from left to right, which means that obscuring image patches has a smaller impact on the classification (on individual weight samples). This suggests that as data is added, the model better captures information in the remaining pixels, so when patches are obscured, the prediction is less dependent on specific local image regions.

5.3.4 Skin Lesion Diagnosis

In medicine, interpretable models are necessary for patients to trust diagnoses and doctors to verify the decisions. Visualising regions of uncertainty medical images is one way of increasing the interpretability of these models. For this experiment we focus on classifying skin lesions from the ISIC2018 dataset in four classes: NV, MEL, BKL, and BCC [60, 11]. We use the same convolutional BNN architecture as in the CIFAR10 experiments, reserve 50 images in each class for validation and testing, and use a balanced subset of 414 images per class for training. This model attains an accuracy of 58.5%. We then compute $\Delta U_{i,epistemic}(x^*)$ and $\Delta U_{i,aleatoric}(x^*)$ using parameters $k = 20$ and $l = 22$. We focus on demonstrating interpretability rather than true medical diagnosis or optimal accuracy.

Similar to the CIFAR10 images, we observe that $\Delta U_{i,epistemic}(x^*)$ and $\Delta U_{i,aleatoric}(x^*)$ highlight different regions of the image when predictive uncertainty is high (Fig. 5.10). Uncertainty does not change due to background pixels, suggesting that surrounding skin does not impact uncertainty, as desired. Pixels that increase epistemic uncertainty may highlight parts of the lesion that are unlike the training examples, and can potentially be reduced by providing more training instances similar to these highlighted areas. On the other hand, aleatoric uncertainty is irreducible unless we obtain more refined measurements; these are regions where doctor input is needed. Our $\Delta U_i(x^*)$ approach eliminates the need to consider each class individually; instead, it considers pixel contributions over the entire

softmax distribution and separates the uncertainty in prediction into reducible and irreducible components.

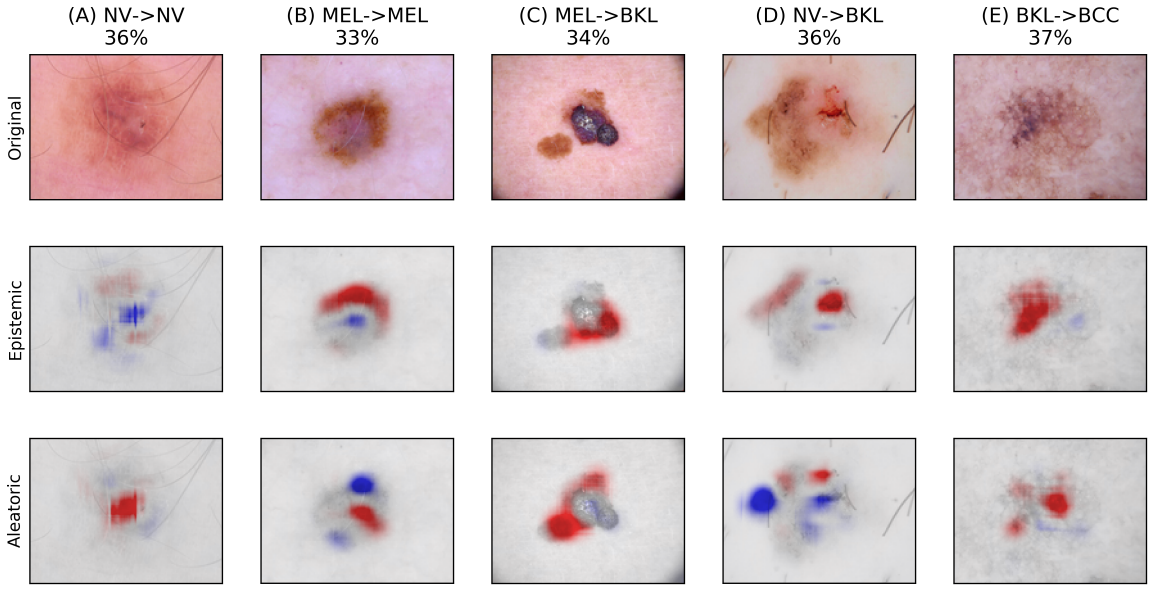


Fig. 5.10 Visualisation of epistemic and aleatoric uncertainties for selected ISIC skin lesion test images, for examples with high predictive uncertainty. The original images, labeled as true class $\rightarrow$ predicted class with softmax activation of the predicted class, are in the top row. The first two examples show correct model predictions, and the remaining are incorrect predictions. $\Delta U_{i,epistemic}(x^*)$ and $\Delta U_{i,aleatoric}(x^*)$ are in the middle and bottom rows, respectively. Colorscale contrast was increased in the salience maps for improved visibility.

Chapter 6

Conclusions

In this project we provide a method for visualising the contribution of individual pixels to uncertainty in image classification. Our approach relies on BNNs to provide uncertainty estimates in the prediction, but is otherwise *model-agnostic*, independent of the specific implementation of the BNN. We focus on *local explanations* – providing explanations for specific test instances.

Our approach quantifies the change in uncertainty due to individual features. We estimate the difference in epistemic, aleatoric, and predictive uncertainty in classifying a test instance when we know the value of all pixels, compared to when we do not know the values of a patch of pixels. While a feature-based change in uncertainty can also be estimated via a gradient [17], our approach yields smooth saliency maps consistent with the spatially correlated structure of images. In this work we focus on classifying images, but this approach is also applicable to any inputs with correlated features, such as those with temporal correlations in speech-processing domains.

We apply our methodology on the CIFAR10 and ISIC2018 lesion diagnosis datasets. In both datasets, we observe that when predictive uncertainty in the classification high, the salience maps for epistemic and aleatoric uncertainty capture different aspects of the image. In CIFAR10, aleatoric uncertainty is reduced over the central object of the image, while epistemic uncertainty increases in the background, suggesting that unconventional backgrounds are a source of uncertainty because they are unlike the examples seen in training. In the skin lesion dataset, we observe that aleatoric and epistemic uncertainty highlight different areas of the lesion, but uncertainty does not change when obscuring parts of the background skin, suggesting that the background does not contribute to uncertainty. Separating aleatoric and epistemic uncertainty, particularly on borderline cases when the output softmax distribution is near-uniform, shows us regions that make the test instance an

outlier (increased epistemic uncertainty), and regions that help the classification (reduced aleatoric uncertainty) or require expert annotation (increased aleatoric uncertainty).

6.1 Limitations of our approach

We acknowledge some limitations of our current methodology. Our approach relies on obscuring $k \times k$ square patches of adjacent pixels to simulate unknown values for a patch, and computes the change in uncertainty based on these patches. However this approach is limited for images with multiple discriminative regions. For example, in an image with two cars, a single patch may not cover both cars, so we cannot measure the change in uncertainty when both cars are unknown. The number of configurations with multiple obscuring patches in multiple locations grows exponentially, and thus is too costly to compute directly. Secondly, BNNs can produce uncertainty estimates of varying quality depending on the divergence objective [31] and an approximation gap (the inability of the variational family to truly approximate the posterior distribution) [13]. Because our approach is model-agnostic, the resulting salience maps heavily depend on the quality of uncertainty estimates from the BNN.

6.2 Future work

Our approach leaves several avenues for further investigation:

Given that our approach is model-agnostic, it would be fascinating to compare salience maps using different variational distribution families. If the resulting salience maps are similar across different variational families, we can use more efficient variational approximations in place of ones that better approximate the weight posterior, allowing us to reduce the computational load of generating salience maps.

Our current approach is entirely qualitative, and there are limited quantitative metrics of interpretability. Given two different explanations for uncertainty in a classifier, such as using different variational distributions or divergence objectives, we would like to determine which interpretation is more understandable. The most appropriate way of comparing interpretability is via user studies, but these experiments are time-intensive and limited in scope. However, we could use properties like total variation and information concentration as proposed in [14] as proxies for interpretability, in which we desire salience maps that are smooth and sparse in the input pixel space. This allows us to quantitatively compare different interpretations of uncertainty.

Finally, while we focus entirely on image data in this work, it would also be interesting to investigate this approach on data with temporal correlations, like speech data. Audio data

often relies on recurrent models, and Bayesian variants of recurrent architectures have been investigated [25]. With our approach, one could investigate the spectral features contributing to model decisions on an audio signal, with interpretations that have smooth variations in feature importance over time.

References

[1] (1974). United states equal credit opportunity act, title 12, part 1002.9.

[2] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

[3] Amari, S.-i. (2012). *Differential-geometrical methods in statistics*, volume 28. Springer Science & Business Media.

[4] Bengio, Y., Boulanger-Lewandowski, N., and Pascanu, R. (2012). Advances in optimizing recurrent networks. *arXiv preprint arXiv:1212.0901*.

[5] Bibal, A. and Frénay, B. (2016). Interpretability of machine learning models and representations: an introduction. In *Proceedings on ESANN*, pages 77–82.

[6] Biran, O. and Cotton, C. (2017). Explanation and justification in machine learning: A survey. In *IJCAI-17 Workshop on Explainable AI (XAI)*, page 8.

[7] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

[8] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

[9] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[10] Chen, C., Li, O., Barnett, A., Su, J., and Rudin, C. (2018). This looks like that: deep learning for interpretable image recognition. *arXiv preprint arXiv:1806.10574*.

[11] Codella, N. C., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., et al. (2018). Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pages 168–172. IEEE.

[12] Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18.

[13] Cremer, C., Li, X., and Duvenaud, D. (2018). Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*.

[14] Dabkowski, P. and Gal, Y. (2017). Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, pages 6970–6979.

[15] Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. (2016). Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*.

[16] Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. (2017a). Uncertainty decomposition in bayesian neural networks with latent variables. *arXiv preprint arXiv:1706.08495*.

[17] Depeweg, S., Hernández-Lobato, J. M., Udluft, S., and Runkler, T. (2017b). Sensitivity analysis for predictive uncertainty in bayesian neural networks. *arXiv preprint arXiv:1712.03605*.

[18] Der Kiureghian, A. and Ditlevsen, O. (2009). Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112.

[19] Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning.

[20] Felleman, D. J. and Van, D. E. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 1(1):1–47.

[21] Fong, R. C. and Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. *arXiv preprint arXiv:1704.03296*.

[22] Freitas, A. A. (2014). Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10.

[23] Gal, Y. (2016). Uncertainty in deep learning. *University of Cambridge*.

[24] Gal, Y. and Ghahramani, Z. (2016a). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

[25] Gal, Y. and Ghahramani, Z. (2016b). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

[26] Gal, Y., Hron, J., and Kendall, A. (2017a). Concrete dropout. In *Advances in Neural Information Processing Systems*, pages 3584–3593.

[27] Gal, Y., Islam, R., and Ghahramani, Z. (2017b). Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*.

[28] Goodman, B. and Flaxman, S. (2016). European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*.

[29] He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE.

[30] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[31] Hernández-Lobato, J. M., Li, Y., Rowland, M., Hernández-Lobato, D., Bui, T., and Turner, R. E. (2016). Black-box α-divergence minimization.

[32] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.

[33] Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

[34] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

[35] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5580–5590.

[36] Kim, B., Rudin, C., and Shah, J. A. (2014). The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems*, pages 1952–1960.

[37] Kim, Y., Lee, H., and Provost, E. M. (2013). Deep learning for robust feature generation in audiovisual emotion recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3687–3691. IEEE.

[38] Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*.

[39] Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.

[40] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[41] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

[42] Li, Y. and Gal, Y. (2017). Dropout inference in bayesian neural networks with alpha-divergences. *arXiv preprint arXiv:1703.02914*.

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

[44] Malinin, A. and Gales, M. (2018). Predictive uncertainty estimation via prior networks. *arXiv preprint arXiv:1802.10501*.

[45] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[46] Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

[47] Parliament and of the European Union, C. (2016). General data protection regulation, recital 71.

[48] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

[49] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

[50] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[51] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016a). Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.

[52] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016b). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

[53] Robnik-Šikonja, M. and Kononenko, I. (2008). Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):589–600.

[54] Rüping, S. et al. (2006). Learning interpretable models.

[55] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[56] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[57] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[58] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[59] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

[60] Tschandl, P., Rosendahl, C., and Kittler, H. (2018). The ham10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions. *arXiv preprint arXiv:1803.10417*.

[61] Ustun, B. and Rudin, C. (2016). Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391.

[62] Wagstaff, K. L. and Lee, J. (2018). Interpretable discovery in large image data sets. *arXiv preprint arXiv:1806.08340*.

[63] Wang, F. and Rudin, C. (2015). Falling rule lists. In *Artificial Intelligence and Statistics*, pages 1013–1022.

[64] Weller, A. (2017). Challenges for transparency. *arXiv preprint arXiv:1708.01870*.

[65] Wexler, J. (2017). Facets: An open source visualization tool for machine learning training data.

[66] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

[67] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929.

[68] Zhu, H. and Rohwer, R. (1995). Information geometric measurements of generalisation.

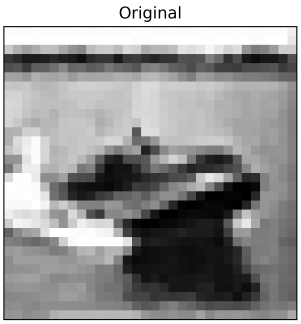
[69] Zintgraf, L. M., Cohen, T. S., Adel, T., and Welling, M. (2017). Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*.

Appendix A

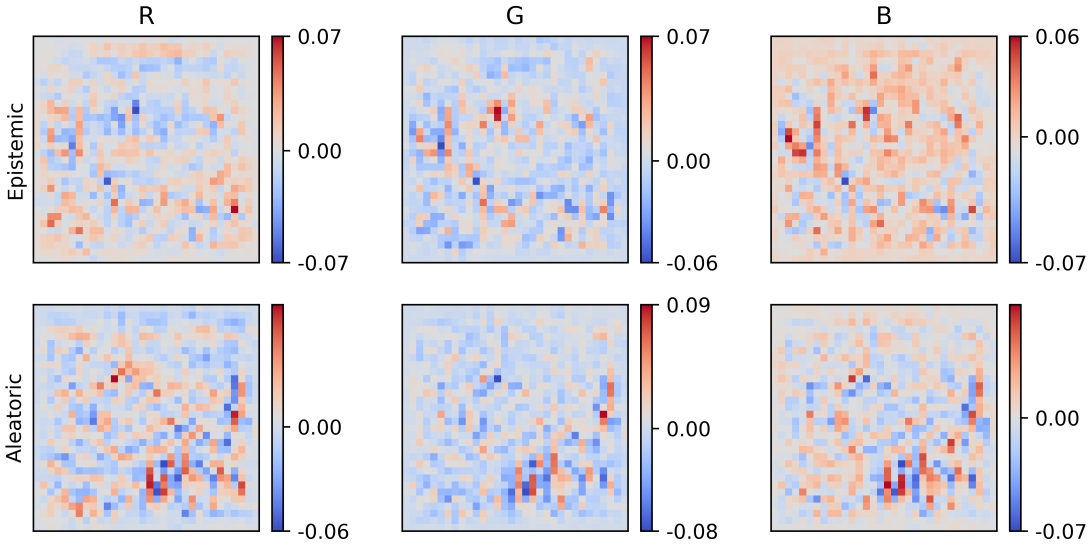
Sensitivity Analysis in Images

Depeweg et al. [17] determine the features affecting uncertainty in BNNs using a gradient-based approach, by taking the derivative of epistemic and aleatoric uncertainties with respect to each input feature. Features with high aleatoric sensitivity suggest a dependence on unobserved variables, while features with high epistemic sensitivity suggest that these features should be monitored to remain in regions where the model is confident. This approach works well when features can vary independently.

In Fig. A.1 we show the results of directly applying the gradient-based sensitivity approach to images. In images, patches of neighboring pixels are often similar, but with the gradient-based approach the sensitivity of a pixel is not similar to that of surrounding pixels. Furthermore, images are a composite of three colour channels, and the sensitivity approach considers each channel separately, thus generating three sensitivity maps. These two limitations motivate an alternative approach to visualising uncertainty that is smooth in the input features, which is the focus of our work.



(a) Original Image



(b) Sensitivity maps

Fig. A.1 (A) Original image from the CIFAR10 test set. (B) Sensitivity of epistemic uncertainty (top row) and aleatoric uncertainty (bottom row) with respect to each pixel of the red, green, and blue color channels.

Appendix B

Additional Salience Visualisations

B.1 Robustness Across BNN replicates

The visualisations shown previously compute $\Delta U_i(x^*)$ from a single BNN. However, we trained three replicate BNNs on the same dataset, which only differ in random seeds. In Fig. B.1 we show the salience maps generated from these model replicates, suggesting that convergence of variational parameters and the visualisations are robust to random seeds.

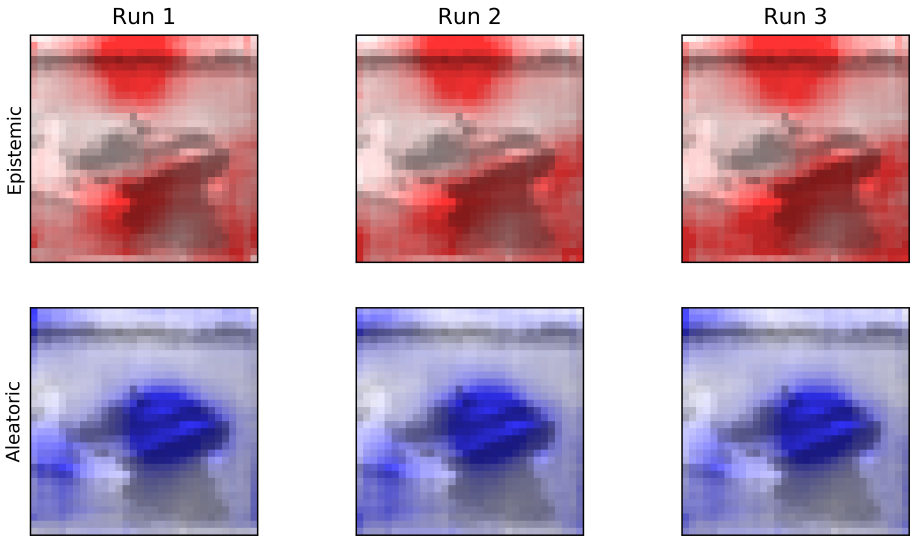
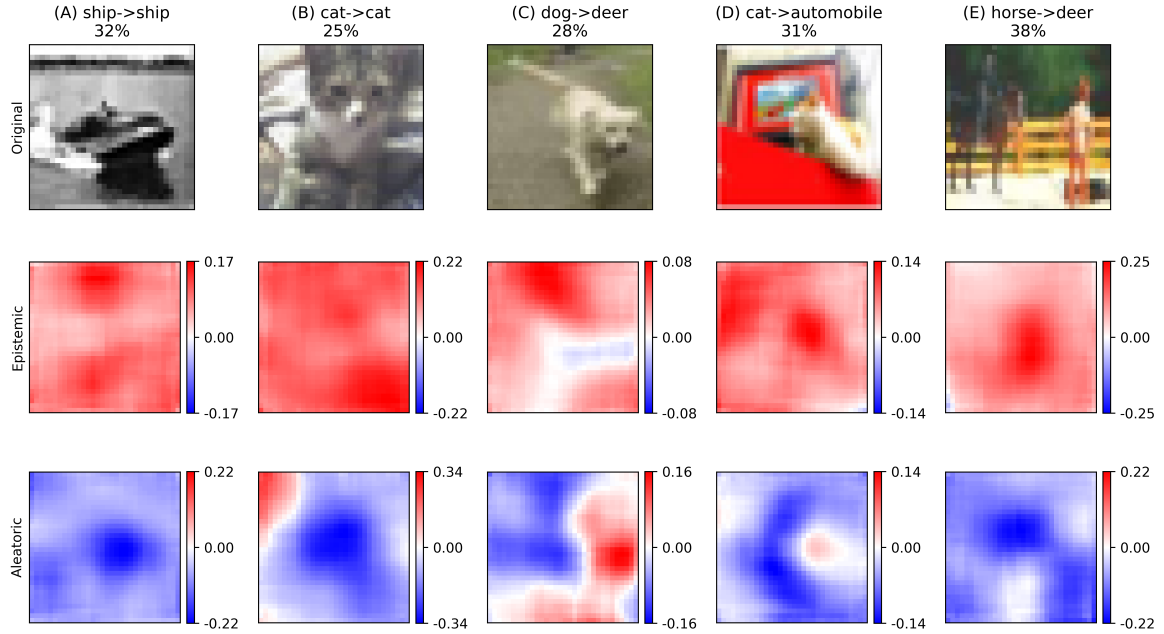


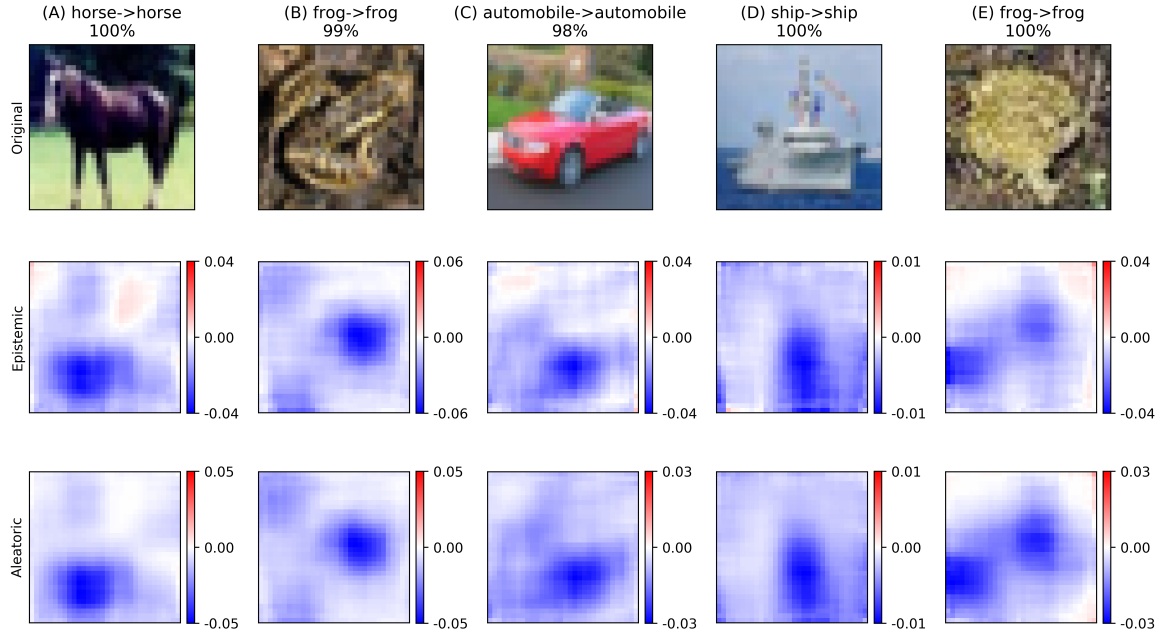
Fig. B.1 Epistemic and aleatoric uncertainty salience maps generated three replicates of the BNN on a CIFAR10 test image.

B.2 Absolute Magnitudes of Uncertainty Change

In Figs. 5.6 and 5.7 we overlay $\Delta U_i(x^*)$ on the input image. Here we show the magnitude of $\Delta U_i(x^*)$ for the same images (Fig. B.2). The magnitude of $\Delta U_i(x^*)$ tends to be higher for images with high predictive uncertainty (when the classification is uncertain) than those with low predictive uncertainty. This suggests that for images that are confidently predicted, uncertainty is robust to the obscuration of the object. Perhaps in these cases, the background pixels provide enough information to classify the object, and are consistent with those seen in training, unlike images with unconventional backgrounds that have higher predictive uncertainty



(a) Examples with high predictive uncertainty



(b) Examples with low predictive uncertainty

Fig. B.2 Magnitudes of $\Delta U_i(x^*)$, rather than overlays, for when a feature is known compared to when it is not.

B.3 Additional Visualisations

We show visualisations of uncertainty for random test images, rather than selecting for high and low predictive uncertainty (Fig. B.3). The salience maps for epistemic and aleatoric uncertainty are similar when the classification is certain (high maximal softmax and low predictive uncertainty), but diverge in uncertain classifications.

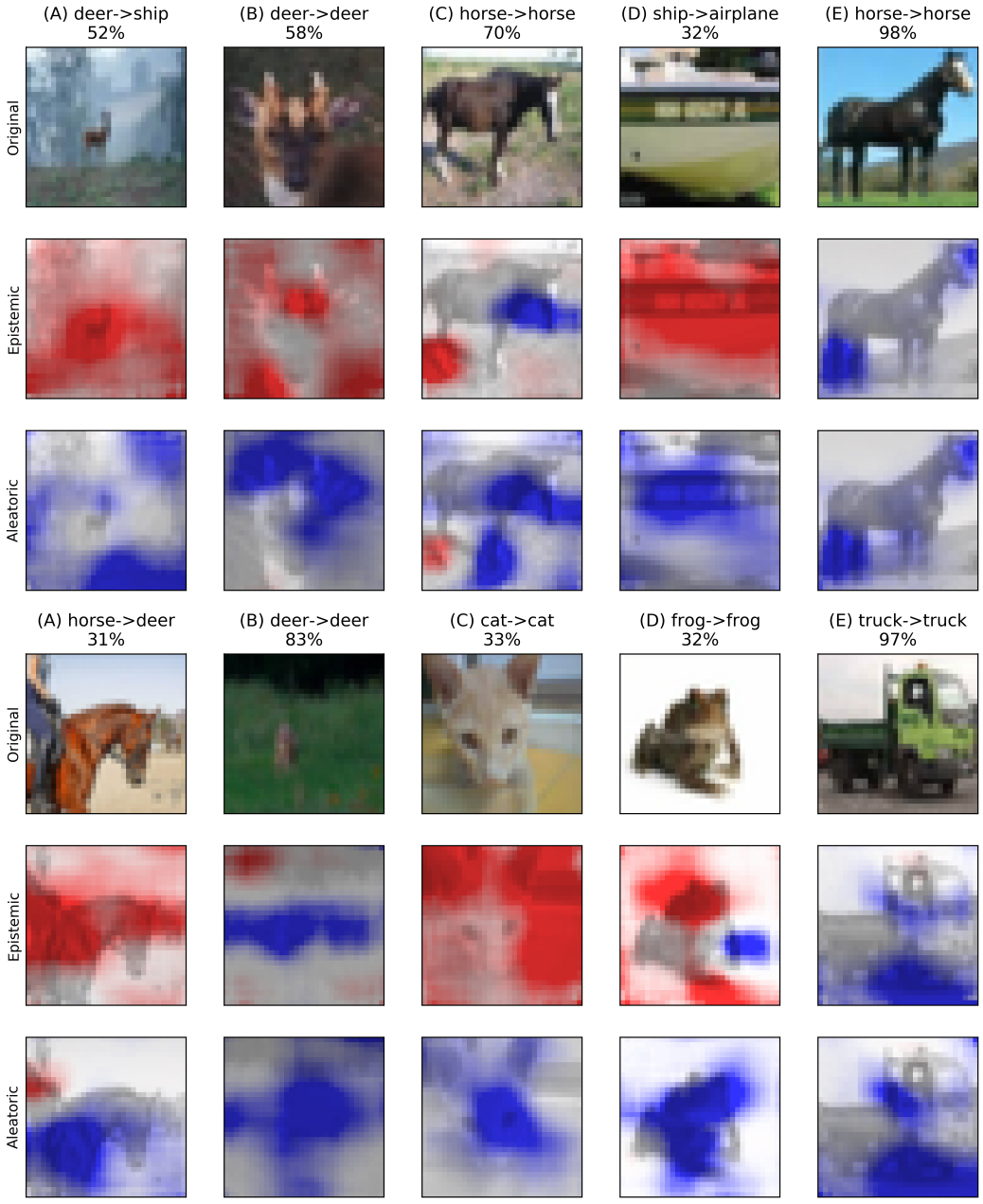


Fig. B.3 Visualisations of epistemic and aleatoric uncertainty salience maps for 10 randomly selected test images. Colorscale contrast increased for visualisation.