Understanding Uncertainty in Bayesian Neural Networks

Javier Antorán Cabiscol

Department of Engineering

University of Cambridge

This dissertation is submitted for the degree of

Master of Philosophy in Machine Learning and Machine Intelligence

Darwin College August 2019

I would like to dedicate this thesis to my ever loving parents, who have made my studies in Cambridge possible.

Declaration

I, Javier Antorán Cabiscol of Darwin College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, 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.

All software used in this thesis was written in Python from scratch for the purpose of this thesis. No proprietary tools were used.

Word Count: 14,974

Javier Antorán Cabiscol
August 2019

Acknowledgements

First and foremost, I would like to thank my supervisors Tameem Adel and José Miguel Hernández-Lobato for their support and guidance throughout this project. Their supervision has allowed me the freedom to pursue my interests while providing me with an endless stream of suggestions and ideas to explore. Working with them has been a pleasure.

I would also like to thank my coursemates for sharing my enthusiasm for machine learning and for motivating me to make the most out of the MPhil. Finally, I would like to thank Michael Hutchinson for his suggestions on the writing of this thesis.

Abstract

In applications with strict safety or fairness requirements, such as self-driving cars, drug toxicity prediction or loan default prediction, understanding why a model made a decision can be as crucial as the prediction's accuracy. Critical applications are also behind the surging mainstream interest in Bayesian deep learning. Bayesian neural networks are able to provide reliable uncertainty estimates together with their predictions. These can prevent automated systems from behaving erratically when faced with unforeseen circumstances. Surprisingly, there has been very little work in leveraging the uncertainty estimates provided by probabilistic models to make automated decisions more interpretable.

This work develops CLUE, a new method for interpreting predictive uncertainty estimates using counterfactual generation. By searching for low uncertainty points that are near the input we wish to explain, CLUE answers the question: "How should we change an input such that it makes our model less uncertain?" A generative model is used to ensure that CLUE's explanations consist of plausible input configurations. Unlike traditional interpretability methods, which are often limited to asking questions in terms of class probabilities, an uncertainty-based approach can provide insight into both regression and classification predictions. We also propose a clustering algorithm that groups similar CLUE explanations, allowing for the identification of the main sources of uncertainty within datasets.

A novel functionally-grounded framework for the quantitative evaluation of uncertainty explanations is proposed. It uses a conditional generative model as a ground-truth generative process. This allows for the generation of data with which to train models as well as the evaluation of explanations' uncertainty. CLUE is found to perform favourably compared to baseline methods using the aforementioned framework. Our qualitative assessment matches our numerical results, finding that CLUE generates cleaner and more intuitive saliency maps than the baselines.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ARD	Automatic Relevance Determination
BNN	Bayesian Neural Network
CLUE	Counterfactual Latent Uncertainty Explanation
DGM	Deep Generative Model
DLVM	Deep Latent Variable Model
ELBO	Evidence Lower Bound
EP	Expectation Propagation
FGSM	Fast Gradient Sign Method
FIDO	Fill-In the DropOut region
GDPR	General Data Protection Regulation
GMM	Gaussian Mixture Model
GP	Gaussian processes
HMC	Hybrid Monte Carlo
LIME	Local Interpretable Model-agnostic Explanations
MAP	Maximum a Posteriori
MC	Monte Carlo

MCMC	Markov Chain Monte Carlo
MFVI	Mean Field Variational Inference
ML	Machine Learning
NN	Neural Network
OOD	Out Of Distribution
pSGLD	Preconditioned Stochastic Gradient Langevin Dynamics
rms	Root Mean Square
SGD	Stochastic Gradient Descent
$SG-HMC$	Stochastic Gradient Hybrid Monte Carlo
SGLD	Stochastic Gradient Langevin Dynamics
$SG-MCMC$	Stochastic Gradient Markov Chain Monte Carlo
SGVI	Stochastic Gradient Variational Inference
$U-FIDO$	Uncertainty Fill-In the DropOut region
VAE	Variational Autoencoder
VAEAC	Variational Autoencoder with Arbitrary Conditioning
VI	Variational Inference

Chapter 1

Introduction

Data-driven automated decision making is becoming ubiquitous in modern society. With it, has come the need to understand the reasoning behind decisions made automatically. Being able to interpret a machine learning model's predictions can help practitioners identify and correct models with flawed reasoning such as an image classifier that learns to make decisions based on contextual cues instead of learning to recognise the intended objects (Mudrakarta et al., 2018) or a loan default predictor which places an unduly amount of weight on immutable characteristics such as race or gender (Adel et al., 2019). Interpretability is also important for end-users to build trust in an automated decision making system and to provide them with recourse should they be the subjects of decisions made automatically. Understanding why a person was denied a loan gives them the agency to make changes such that their approval would be guaranteed were they to re-apply.

Neural Networks (NN) are a class of Machine Learning (ML) models which have recently soared in popularity due to their flexibility and scalability to large amounts of data (LeCun et al., 2015). These models present a black-box solution to a wide array of problems in the fields of healthcare, finance, and criminal justice, among others. They allow practitioners to deploy accurate decision making tools without the need for detailed domain knowledge.

Simple models, like logistic regression, are inherently interpretable. Here, the amount of influence each input feature exerts over predictions is given by the corresponding weight's magnitude. The polarity of a feature's contribution matches the sign of its weight. Unfortunately, the nonlinearities and vast weight spaces intrinsic to NNs prevent such intuitive forms of analysis. In response, a rapidly growing subfield of ML has formed around developing methods to interpret complex models' decisions (Montavon et al., 2018).

NNs are most commonly trained in a Maximum a Posteriori (MAP) framework and treated as deterministic functions. As a result, these models produce unreliable uncertainty estimates and can behave erratically when faced with test samples which are very different from the ones seen during training. In contrast, the Bayesian approach substitutes weight point estimates with probability distributions, allowing us to consider all possible parameter values and thus all possible functions that can be expressed by our model, when making predictions. In Bayesian Neural Networks (BNN), uncertainty in weight space is translated into uncertainty in predictions, allowing for an accurate representation of "what we do not know" (Gal, 2016). Uncertainty estimates provide a significant step towards safer and more interpretable automated decision making. Not wanting to stop there, we may also ask the question: "Why is our model uncertain?"

In finance, uncertainty is often treated as an additional cost. Decisions are based on which options are assigned the highest expected reward while being the least uncertain. In this setting, answering the question: "Why is our model uncertain about a prediction?" is equally as important as explaining why it made the prediction. Returning to our previous example, an applicant who has been denied a loan may not just want to know what they can do to get approved but what they can do to get approved with high confidence.

Understanding what causes a model to be uncertain may also help machine learning practitioners learn about their datasets. When a model is uncertain about its parameters in some areas of input space, it suggests that data is sparse those regions. When training a loan default predictor, this may facilitate the identification of sub-groups (by age, gender, race, etc.) that are under-represented in the training data. Collecting more data from these groups can allow us to better constrain the model's parameters, leading to accurate predictions for a broader range of clients. Identifying the sources of irreducible uncertainty (see section 2.2 for an explanation of the different types of uncertainties) in a dataset may reveal the need to measure additional input features in order to make accurate predictions. It can also help detect unreliability in our measurement techniques.

It comes as somewhat of a surprise that, unlike interpreting machine learning predictions, which is a burgeoning field, there has been very little research on understanding what causes complex models to be uncertain. The objective of this work is to explore how uncertainty can be used for interpretability in ML and to develop new methods to explain the uncertainty estimates given by BNNs.

1.1 Thesis Contributions

The main contributions of this work are as follows:

1. A **unifying review of existing machine learning interpretability methods** and evaluation metrics. Special emphasis is placed on uncertainty sensitivity analysis (Depeweg et al., 2017), the only existing method designed for the interpretation of uncertainty estimates.
2. Proposing Counterfactual Latent Uncertainty Explanations or CLUE, a **novel approach to interpreting uncertainty estimates** in BNNs that is applicable to both tabular data and image data in both regression and classification tasks. Unlike uncertainty sensitivity analysis, CLUE provides explanations for individual datapoints and is able to consider mutual interactions among input variables in its explanations.
3. Proposing the use of a clustering algorithm to group similar CLUE explanations, thus **providing an overview of the sources of uncertainty within datasets**.
4. Proposing a **novel framework for the evaluation of uncertainty explanations** that uses a conditional generative model as the ground-truth generative process of the data. This allows to both generate data with which to train models and evaluate explanations provided by interpretability methods.
5. An **evaluation of CLUE** using the previously mentioned framework. The proposed method is found to compare favourably to baseline approaches. This is followed by a discussion on how our results can be leveraged to take further steps towards interpreting complex models' uncertainty estimates.

1.2 Thesis Outline

This work concerns itself with the study of how the different types of uncertainty estimates obtained from complex models, such as BNNs, can be leveraged to better understand these models' decisions. Chapter 2 provides the theoretical background necessary to understand the tools and ideas presented in the rest of this thesis. Despite the lack of previous research into the interpretation of uncertainty estimates, many of the principles that have guided the work done in interpreting neural network predictions can be applied to uncertainty. We dedicate chapter 3 to reviewing the different types of existing ML interpretability methods, presenting what we call "a taxonomy of explanations." We follow this by introducing the different ways in which interpretability methods can be evaluated and presenting uncertainty sensitivity analysis (Depeweg et al., 2017), to the best of our knowledge, the only existing method dedicated to interpreting uncertainty estimates.

In chapter 4, we propose CLUE, a method that explains why a BNN is uncertain about specific inputs by providing alternative or "counterfactual" feature configurations which are similar to the original inputs but for which the BNN is not uncertain. Comparing the original and the counterfactual input configurations provides the user with a way to understand which features are responsible for the BNN being uncertain. We also propose the use of a clustering algorithm to group data points for which the uncertainty can be explained in similar ways, generating summaries about the global causes of uncertainty in datasets.

In chapter 5, we introduce a novel framework for the evaluation of uncertainty interpretability methods. It is based on using a conditional generative model as the ground-truth generative process of the data. This allows us to obtain the true uncertainty values associated with each interpretability method's explanations. In addition, evaluating explanations' likelihoods under the ground-truth generative model allows for the detection of out-of-distribution explanations. We show that CLUE compares favourably to baseline approaches using the proposed evaluation scheme for two regression datasets and two classification datasets. We also perform a qualitative evaluation of uncertainty interpretability methods on MNIST, where we find CLUE to produce the clearest and most intuitive explanations. Finally, in chapter 6, we discuss the results obtained with the aforementioned techniques and, in light of these, outline promising avenues for future work.

Chapter 2

Background: Bayesian Methods in Deep Learning

In this chapter, we introduce the fundamental ideas and tools from the ML literature which this thesis draws upon in latter chapters. In section 2.1, we give a brief overview of existing approximate inference methods for BNNs. In section 2.2, we discuss the different types of uncertainties that these models can express. We follow this by reviewing recent developments in Deep Generative Models (DGM) and how these can be used for feature imputation in section 2.3. Finally, in section 2.4, we discuss a variational approach for fitting Gaussian Mixture Models (GMM).

2.1 Approximate Inference in BNNs

Given a dataset $\mathscr{D} = \{\mathbf{X}, \mathbf{Y}\}$ composed of inputs $\mathbf{X} = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$ and targets $\mathbf{Y} = \{\mathbf{y}_1 \dots \mathbf{y}_n\}$, and a neural network parametrised by $\mathbf{w}$, the maximum likelihood criterion:

$$\mathbf{w}_{ml} = \arg\max_{\mathbf{w}} p(\mathscr{D}|\mathbf{w}) = \arg\max_{\mathbf{w}} \log p(\mathscr{D}|\mathbf{w}) \tag{2.1}$$

aims to find the single setting of the weights $\mathbf{w}_{ml}$ which is most likely to have generated our data. This is often not desirable however, as it does not guarantee that our model will explain unseen data well. It is indeed the case that maximum likelihood often leads to generalisation issues, overconfidence in predictions, and erratic behaviour on Out Of Distribution (OOD) samples.

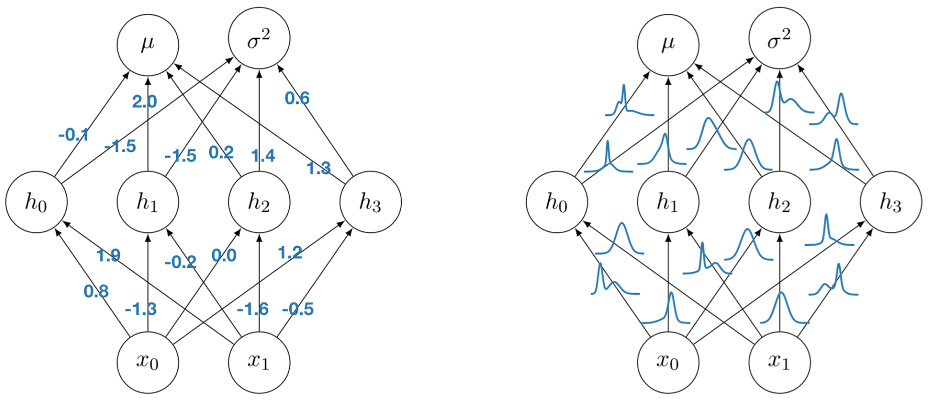

Fig. 2.1 Left: In a NN, each weight has a single value. Right: BNNs define probability distributions over weights.

The probabilistic, or Bayesian, approach provides a more principled path towards making inferences from data. In this framework, a prior distribution $p(\mathbf{w})$ is introduced to reflect our initial beliefs about the parameters. An appropriate choice of prior can allow us to balance model complexity and wellness of fit. Our prior beliefs are updated when we observe data through Bayes' rule:

$$p(\mathbf{w}|\mathscr{D}) = \frac{p(\mathscr{D}|\mathbf{w})p(\mathbf{w})}{p(\mathscr{D})} \tag{2.2}$$

This operation returns a probability distribution over all possible configurations of our parameters, placing more weight on settings which are more likely to have generated the data, as illustrated in fig. 2.1. Consequently, this distribution encodes our uncertainty about what values these parameters should take. Predictions for new datapoints $\mathbf{X}^* = \{\mathbf{x}_1^* \ldots \mathbf{x}_n^*\}$ are made through marginalisation:

$$p(\mathbf{Y}^*|\mathbf{X}^*, \mathscr{D}) = \int p(\mathbf{Y}^*|\mathbf{X}^*, \mathbf{w})p(\mathbf{w}|\mathscr{D})\, d\mathbf{w} \tag{2.3}$$

In this way, uncertainty in model parameters is translated into uncertainty in predictions, yielding reliable error bounds and preventing overfitting. Bayesian methods enable NNs to be used for tasks which require model uncertainty, such as continual learning (Nguyen et al., 2018), active learning (Depeweg et al., 2018), and Bayesian optimisation (Springenberg et al., 2016).

Unfortunately, the non-linearities in NNs make computing the integral in (2.3) intractable. Analogously, obtaining the model evidence:

$$p(\mathscr{D}) = \int p(\mathscr{D}|\mathbf{w})p(\mathbf{w})\,d\mathbf{w} \tag{2.4}$$

is also intractable. Thus, if we are to capture model uncertainty in neural networks, we must resort to approximations. The rest of this section is dedicated to the introduction of various approximate inference strategies for BNNs. In section 5.2, we compare the fidelity of the uncertainty estimates provided by each of these methods, ultimately finding SG-HMC to perform best.

2.1.1 Variational Inference

Variational Inference (VI) (Hinton and van Camp, 1993) approximates the posterior distribution over model parameters $p(\mathbf{w}|\mathscr{D})$ with the approximate or variational distribution $q_\phi(\mathbf{w})$. This function is parametrised by ϕ, which are known as the variational parameters. We would like $q_\phi(\mathbf{w})$ to be as close to $p(\mathbf{w}|\mathscr{D})$ as possible. VI achieves this by searching for the configuration of ϕ which minimises the KL divergence between the two distributions:

$$\mathrm{KL}(q_\phi(\mathbf{w})\,||\,p(\mathbf{w}|\mathscr{D})) = \int q_\phi(\mathbf{w}) \log \frac{q_\phi(\mathbf{w})}{p(\mathbf{w}|\mathscr{D})}\,d\mathbf{w} \tag{2.5}$$

$$= \log p(\mathscr{D}) + \underbrace{\int q_\phi(\mathbf{w}) \log \frac{q_\phi(\mathbf{w})}{p(\mathbf{w})} d\mathbf{w}}_{\mathrm{KL}(q_\phi(\mathbf{w})\,||\,p(\mathbf{w}))} - \underbrace{\int q_\phi(\mathbf{w}) \log p(\mathscr{D}|\mathbf{w}) d\mathbf{w}}_{\mathbb{E}_{q_\phi(\mathbf{w})}[\log p(\mathscr{D}|\mathbf{w})]} \tag{2.6}$$

Unfortunately, (2.6) still contains the model evidence, an intractable term. Nonetheless, leveraging the non-negativity of the KL divergence[1], we can use (2.6) to define a lower bound on the model evidence:

$$\log p(\mathscr{D}) \geq \mathrm{ELBO} = \mathbb{E}_{q_\phi(\mathbf{w})}[\log p(\mathscr{D}|\mathbf{w})] - \mathrm{KL}(q_\phi(\mathbf{w})\,||\,p(\mathbf{w})) \tag{2.7}$$

VI minimises (2.5) indirectly through the maximisation of the Evidence Lower Bound (ELBO). Simultaneously, the marginal log-likelihood of the data is maximised. The ELBO can be decomposed into two terms: the conditional log-likelihood of the data given the parameters, which measures how well the data is explained, and a negative KL divergence, which penalises the variational posterior for differing from the prior.

[1]This can be shown to be true using Jensen's inequality: $g(\mathbb{E}[x]) \leq \mathbb{E}[g(x)]$, where $g(\cdot)$ is a convex function.

The success of VI is heavily dependent on the chosen family of variational distributions. There is a trade-off between choosing an expressive family, which will better approximate the posterior, or a simple one that will lead to tractable calculations. Since the selected family is rarely rich enough to contain the exact posterior, estimations made with VI are often biased.

Stochastic Gradient Variational Bayes

Independently of the choice of family of approximate posteriors, optimising (2.7) analytically is intractable for neural network models. We must resort to Monte Carlo (MC) estimates of the ELBO and gradient based optimisation. An unbiased MC estimator of (2.7) can be formulated as:

$$\text{ELBO} \approx \sum_{k=1}^{K} \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}_k) - \log q_\phi(\mathbf{w}_k) + \log p(\mathbf{w}_k); \quad \mathbf{w}_k \sim q_\phi(\mathbf{w}) \tag{2.8}$$

Nonetheless, differentiating the weight sampling operation with respect to ϕ remains a critical issue. Building upon earlier work, where biased (Graves, 2011) or high-variance (Paisley et al., 2012) gradient estimators were employed, Kingma and Welling (2014); Rezende et al. (2014) introduce the "reparametrisation trick." This method re-phrases the sampling procedure $\mathbf{w}_k \sim q_\phi(\mathbf{w})$ as the output of a differentiable function of ϕ and an auxiliary noise variable:

$$\mathbf{w}_k = g(\phi, \varepsilon_k); \quad \varepsilon_k \sim p(\varepsilon) \tag{2.9}$$

providing us with a low-variance unbiased gradient estimator. Its use is referred to as Stochastic Gradient Variational Inference (SGVI). For a diagonal Gaussian approximate posterior $\phi = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$, (2.9) takes the form: $\mathbf{w}_k = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}_k$.[2]

Building upon this work, Blundell et al. (2015) propose Bayes by backprop, a method for approximate inference in BNNs which computes an unbiased estimate of the ELBO with minibatches $\widetilde{\mathscr{D}}$:

$$\text{ELBO} \approx \sum_{k=1}^{K} \frac{|\mathscr{D}|}{|\widetilde{\mathscr{D}}|} \log p(\mathbf{Y}_{\widetilde{\mathscr{D}}}|\mathbf{X}_{\widetilde{\mathscr{D}}},\mathbf{w}_k) - \log q_\phi(\mathbf{w}_k) + \log p(\mathbf{w}_k); \quad \mathbf{w}_k \sim q_\phi(\mathbf{w}) \tag{2.10}$$

In following work, Kingma et al. (2015) make the observation that, for Gaussian approximate posteriors, the distribution over a fully connected layer's activations is also Gaussian and can

[2] $\odot$ refers to the element-wise product.

be obtained analytically. The authors propose the "local reparametrisation trick," an unbiased estimator which is based on sampling activations as opposed to weights and has a variance that scales with $1/\tilde{\mathscr{D}}$.

In practise, the expressivity of SGVI posteriors is limited by the need for a distribution which admits the reparametrisation trick. In BNNs, a mean field (fully factorised) Gaussian is the most common choice, as it limits the number of variational parameters. Unfortunately, this provides a crude approximation to the complex true posterior over network weights. This problem is aggravated by the mode-seeking properties of the variational objective. In (2.5), we can see that the KL divergence penalises regions where q/p is large. Therefore, optimising the ELBO for unimodal variational posteriors results in these fixating on a single mode of the true posterior, providing an approximation which is faithful only locally.

An alternative approximate inference method is Expectation Propagation (EP). This approach minimises $\mathrm{KL}(p(\mathbf{w}|\mathscr{D})\,||\,q_{\phi}(\mathbf{w}))$, avoiding the mode-seeking nature of VI. It was first applied to BNNs by Hernández-Lobato and Adams (2015). Hernandez-Lobato et al. (2016) propose training BNNs by minimising an α-divergence between the approximate and true posteriors. This objective generalises both VI and EP, giving us a way to regulate the locality of our posterior approximation through the choice of the parameter α. It is also worth noting that some stochastic regularisation techniques, such as dropout and multiplicative Gaussian noise, can be re-interpreted as VI with multiplicative Bernoulli (Gal and Ghahramani, 2016) and fixed-variance Gaussian (Kingma et al., 2015) approximate posteriors respectively. There has also been work linking regular Stochastic Gradient Descent (SGD) with VI (Mandt et al., 2017).

2.1.2 Markov Chain Monte Carlo

Under some circumstances, computing the model evidence is not necessary in order to build a Markov chain with $p(\mathbf{w}|\mathscr{D})$ as its stationary distribution. This allows (2.3) to be approximated as:

$$p(\mathbf{Y}^*|\mathbf{X}^*,\mathscr{D}) \approx \sum_{k=1}^{K} p(\mathbf{Y}^*|\mathbf{X}^*,\mathbf{w}_k); \quad \mathbf{w}_k \sim p(\mathbf{w}|\mathscr{D}) \tag{2.11}$$

Markov Chain Monte Carlo (MCMC) approximate inference techniques are often unbiased, converging to the true posterior in the limit. Unfortunately, when dealing with complex high-dimensional weight spaces, they suffer from slow mixing and can take many iterations to converge. Moreover, the cost of each MC step increases with the size of the dataset. These issues cause traditional MCMC techniques to rarely be considered for modern deep learning problems.

Hybrid Monte Carlo

When dealing with complex high-dimensional distributions, such as posteriors over NN weights, a large random step in parameter space will likely lead to a very low probability region. MCMC algorithms, such as Metropolis-Hastings, are forced to use proposal distributions with small standard deviations, generating correlated samples and leading to behaviour that resembles a random walk. In response, Neal (1996) proposed the use of Hamiltonian or Hybrid Monte Carlo (HMC).

HMC draws an analogy to a fictitious physical system in which each possible setting of the parameters $\mathbf{w}$ is represented by a point on a surface. Each point's probability density relates to the system's potential energy function U as:

$$\log p(\mathbf{w}|\mathscr{D}) \propto \log p(\mathscr{D},\mathbf{w}) = -U(\mathbf{w}) \tag{2.12}$$

In this way, a model's parameter space can be envisioned as an uneven surface where each point's height depends inversely on its probability under the posterior. An auxiliary momentum variable $\mathbf{r}$, of the same shape as $\mathbf{w}$, is introduced. This allows us to define the joint distribution $p(\mathbf{w},\mathbf{r})$ and, with it, the Hamiltonian $H(\mathbf{w},\mathbf{r})$ which gives the combined kinetic and potential energy of our system:

$$\log p(\mathbf{w},\mathbf{r}) \propto -U(\mathbf{w}) - \frac{1}{2}\mathbf{r}^{\intercal}M^{-1}\mathbf{r} = -H(\mathbf{w},\mathbf{r}) \tag{2.13}$$

M contains the fictitious mass associated with each parameter. We can simulate an object drifting around in the physical system we have defined by differentiating with respect to a fictitious time variable τ:

$$\begin{cases} d\mathbf{w} = M^{-1}\mathbf{r}\,d\tau \\ d\mathbf{r} = -\nabla U(\mathbf{w})\,d\tau \end{cases} \tag{2.14}$$

Neal (1996) shows that this process leaves the canonical distribution of $\mathbf{r}$ and $\mathbf{w}$ invariant. It also leaves the value of H constant. In order to explore the complete parameter space, H should be changed periodically by re-sampling $\mathbf{r}$. The marginal distribution of $\mathbf{w}$ in (2.13) is the same as that of (2.12). As a result, we can obtain samples from $p(\mathbf{w}|\mathscr{D})$ by sampling from $p(\mathbf{w},\mathbf{r})$ and discarding the values of $\mathbf{r}$. In practise, the continuous time system is approximated with ε-discretisation (substituting $d\tau$ with ε). A new sample is saved every fixed number of steps. Metropolis steps are used to compensate for discretisation error.

Stochastic Gradient Hybrid Monte Carlo (SG-HMC)

HMC is the gold standard for approximate inference in BNNs. However, it is not often used in modern deep learning because of the computational cost of evaluating $\nabla p(\mathscr{D}|\mathbf{w})$ for large datasets. Recently, a sub-field of machine learning has emerged around designing scalable alternatives by combining stochastic gradient optimisation with MCMC methods (Ma et al., 2015).

Given a mini-batch of $\mathscr{D}$, which we refer to as $\widetilde{\mathscr{D}}$, the gradient of U can be estimated as:

$$\nabla U(\mathbf{w}) \approx \nabla \tilde{U}(\mathbf{w}) = -\frac{|\mathscr{D}|}{|\widetilde{\mathscr{D}}|}\left(\nabla \log p(\widetilde{\mathscr{D}}|\mathbf{w}) + \nabla p(\mathbf{w})\right) = \nabla U(\mathbf{w}) + v \tag{2.15}$$

Making the assumption that the error introduced by the stochastic approximation is distributed normally $v \sim \mathscr{N}(0, 2B\varepsilon)$, Chen et al. (2014) show that naively substituting ∇U with $\nabla \tilde{U}$ results in dynamics that do not leave $p(\mathbf{w}, \mathbf{r})$ invariant. They propose the inclusion of a friction term C in the momentum update from (2.14). This addition compensates for the entropy introduced into the system by v and results in the following updates:

$$\begin{cases} d\mathbf{w} = M^{-1}\mathbf{r}\, d\tau \\ d\mathbf{r} = -\nabla \tilde{U}(\mathbf{w})\, d\tau - CM^{-1}\mathbf{r}\, d\tau + v'; \quad v' \sim \mathscr{N}(0, 2(C - \hat{B})\, d\tau) \end{cases} \tag{2.16}$$

where $\hat{B}$ is an estimate of the gradient noise. The authors show that this process leaves $p(\mathbf{w}, \mathbf{r})$ invariant. Somewhat surprisingly, after ε-discretisation, (2.16) yields steps equivalent to SGD with momentum and an additive noise term. Unfortunately, the Metropolis algorithm still requires computations over the whole dataset, defeating the purpose of mini-batch methods. For this reason, SG-MCMC methods omit the metropolis step and therefore present a bias which grows with the value of the step size. There has been further work on improving various aspects of SG-HMC including but not limited to adaptive hyper parameter tuning (Springenberg et al., 2016), step size scheduling (Zhang et al., 2019) and preconditioning (Ma et al., 2015).

Stochastic Gradient Langevin Dynamics

Langevin Monte Carlo is a particularisation of HMC in which the momentum is re-sampled and candidate states are proposed after every single step. Intuitively, the strength of HMC, which lies in its capacity to produce uncorrelated samples, requires multiple steps between proposals. Notwithstanding, the Stochastic Gradient counterpart of Langevin Dynamics (SGLD), proposed by Welling and Teh (2011), allows us to perform approximate inference in BNNs in a shockingly simple manner:

$$d\mathbf{w} = \frac{1}{2}M\left(\nabla \log p(\mathbf{w}) + \frac{|\mathscr{D}|}{|\widetilde{\mathscr{D}}|}\nabla \log p(\widetilde{\mathscr{D}}|\mathbf{w})\right)d\tau + v^{'}; \quad v^{'} \sim \mathscr{N}(0, M\,d\tau) \tag{2.17}$$

As show in (2.17), SGLD's weight update reflects that of regular SGD with an additive noise term. This simplicity can be extended to the choice of weight matrix M. Li et al. (2016) propose the use of the popular optimisation technique RMSprop (Tieleman and Hinton, 2012) as a preconditioner, yielding preconditioned SGLD (pSGLD).

2.2 Decomposing Uncertainty Estimates in BNNs

Let us consider the possible causes for uncertainty in our predictions:

- There may be inherent noise in the generative process of our data or some unaccounted-for factor which creates variability in our targets.
- We are unable to properly constrain our model's parameters. This means that, out of all the possible functions that our model can represent, we are unsure of which ones better explain the data. This could be due to the use of a very complex model relative to the amount of training data or a lack of diversity in the training data.
- Our choice of model structure is wrong and is unable to reflect the generative process of the data.

We refer to the first type as irreducible or **aleatoric uncertainty**. The latter two uncertainties can be grouped under model or **epistemic uncertainty**. This type of uncertainty can be reduced by observing more data. To maintain computational tractability and because BNNs are very flexible models, capable of expressing a wide range of functions, in this work, we only consider model uncertainty caused by uncertainty in the parameters.

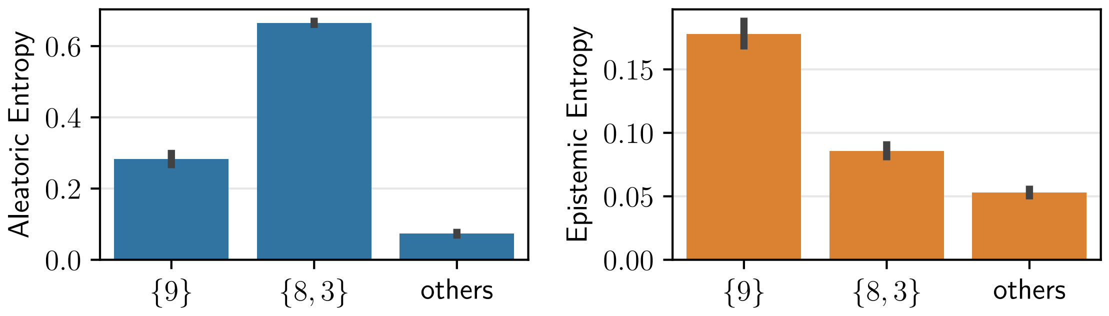

Fig. 2.2 Epistemic and aleatoric entropy values measured with a 2 hidden layer, 1200 hidden unit, fully connected BNN trained with SGVI on a modified version of MNIST (LeCun and Cortes, 2010).

Being able to capture model uncertainty is the main advantage of BNNs over regular NNs. Intuitively, when using an uninformative prior, the Bayesian update (2.2) will not result in overfitting, independently of the complexity of our model. In situations where there is little data, the posterior will resemble the prior, expressing a large amount of uncertainty about which parameters are best. As the amount of data increases, the likelihood term in (2.2) will gain weight, allowing for more complex explanations. Neal (1996) and Gal (2016) argue that, in a Bayesian setting, it is desirable to use large models independently of the amount of data available, as their ability to capture a greater range of functions allows for more accurate uncertainty estimates.

2.2.1 Measuring Uncertainties in BNNs

In this work, we consider BNNs which parametrise two types of distributions over target variables: the categorical for classification problems and the Gaussian for regression. Following Depeweg (2019), we quantify the uncertainty of a categorical distribution with classes $C = \{c_1 \dots c_K\}$, using its predictive entropy:

$$H(y^*|\mathbf{x}^*, \mathscr{D}) = \sum_{c \in C} p(y^*{=}c|\mathbf{x}^*, \mathscr{D}) \log p(y^*{=}c|\mathbf{x}^*, \mathscr{D}) \tag{2.18}$$

This quantity contains aleatoric and epistemic components. The former is estimated as:

$$H_a = \mathbb{E}_{p(\mathbf{w}|\mathscr{D})}[H(y^*|\mathbf{x}^*, \mathbf{w})] \approx \frac{1}{K} \sum_{k}^{K} H(y^*|\mathbf{x}^*, \mathbf{w}_k) \tag{2.19}$$

Given the additive nature of entropy, the epistemic component can be obtained as the difference between the total and aleatoric entropies. This quantity is also known as the

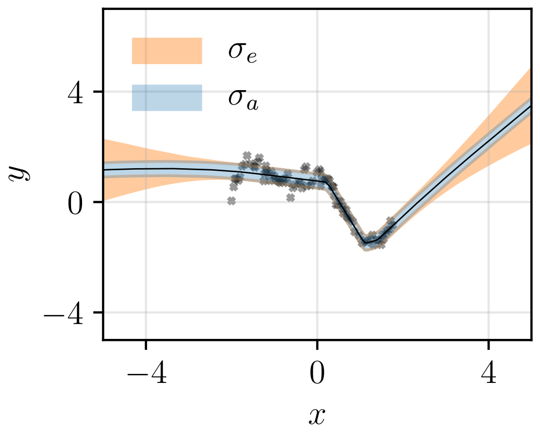

Fig. 2.3 Aleatoric (σ_a) and epistemic (σ_e) standard deviation estimates obtained with a single hidden layer, 100 hidden unit, fully connected BNN fitted with SGVI on a 1-d toy regression dataset.

mutual information between $\mathbf{y}^*$ and $\mathbf{w}$:

$$H_e = I(\mathbf{y}^*, \mathbf{w}|\mathbf{x}^*, \mathscr{D}) = H(y^*|\mathbf{x}^*, \mathscr{D}) - \mathbb{E}_{p(\mathbf{w}|\mathscr{D})}[H(y^*|\mathbf{x}^*, \mathbf{w})] \tag{2.20}$$

We illustrate this decomposition with a small experiment in which a BNN is trained on a modified version of the MNIST dataset; 80% of samples corresponding to the digit 9 have been removed and 30% of samples corresponding to digits 8 and 3 have had their labels permuted. As can be seen in fig. 2.2, the label ambiguity introduced into the data results in increased aleatoric entropy for digits 3 and 8. The lack of data causes increased epistemic entropy for digit 9.

For regression, marginalising model parameters by approximating the integral in (2.3) with Monte Carlo induces a Gaussian mixture distribution over outputs. There is no closed-form expression for the entropy of this distribution. Instead, we use the variance of the GMM as an uncertainty metric:

$$\sigma^2(\mathbf{y}^*|\mathbf{x}^*, \mathscr{D}) \approx \underbrace{\frac{1}{K}\sum_k^K \mu(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w}_k)^2 - (\frac{1}{K}\sum_k^K \mu(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w}_k))^2}_{\sigma^2(\mathbb{E}_{p(\mathbf{y}^*|\mathbf{x}, \mathbf{w}_k)}[\mathbf{y}^*])} + \underbrace{\frac{1}{K}\sum_k^K \sigma^2(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w}_k)}_{\mathbb{E}_{p(\mathbf{w}|\mathscr{D})}[\sigma^2(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w}_k)]} \tag{2.21}$$

This expression decomposes naturally into aleatoric and epistemic components. The aleatoric variance is given by the average variance of the predictive distributions obtained with each sample of $\mathbf{w}$. The variance in the predictive means captures dissimilarity among explanations given by different weight configurations. It is a measure of model uncertainty. In fig. 2.3, we can see that the aleatoric standard deviation matches that of the training data. The epistemic standard deviation is small near the train data and grows as we move further away.

2.3 Deep Latent Variable Models

Deep Generative Models (DGM) leverage neural networks to model complex and high-dimensional data. This field of ML has seen much success recently, with advances such as autoregressive networks (Germain et al., 2015) and GANs (Goodfellow et al., 2014). We focus on Deep Latent Variable Models (DLVM), a class which makes the assumption that our data $\{\mathbf{x}_n\}_{n=1}^N$ can be generated from a set of latent variables $\{\mathbf{z}_n\}_{n=1}^N$.

2.3.1 Variational Autoencoders

The Variational Autoencoder (VAE) (Kingma and Welling, 2014) approximates the intractable posterior over a latent vector $p(\mathbf{z}|\mathbf{x})$ with the approximate distribution $q(\mathbf{z}|\mathbf{x})$. Using a derivation analogous to (2.5)-(2.7), an ELBO can be derived:

$$\log p(\mathbf{x}) \geq \text{ELBO} = -D_{KL}(q(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})) + \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] \tag{2.22}$$

The VAE framework parametrises both $p(\mathbf{x}|\mathbf{z})$ and $q(\mathbf{z}|\mathbf{x})$ with neural networks. Their weights are given by θ and ϕ respectively. Unlike in section 2.1.1, where a different set of parameters had to be learnt for each network weight, the inference network $q_\phi(\mathbf{z}|\mathbf{x})$ allows us to share a fixed-dimension set of variational parameters among all latent variables $\{\mathbf{z}_n\}_{n=1}^N$. The variational posterior is obtained with a forward pass through this network. This is known as amortised variational inference.

Similarly to the methods described in section 2.1.1, (2.22) is maximised using stochastic gradient optimisation after being approximated with Monte Carlo[3]:

$$\text{ELBO} \approx \sum_{k=1}^{K} \underbrace{\log p(\mathbf{z}_k) - \log q_\phi(\mathbf{z}_k|\mathbf{x})}_{\text{KL term}} + \underbrace{\log p_\theta(\mathbf{x}|\mathbf{z}_k)}_{\text{Reconstruction term}} \quad ; \quad \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x}) \tag{2.23}$$

In practise, the variational posterior takes the form of a diagonal Gaussian $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x})^2 \cdot I)$, the prior is chosen to be a unit variance isotropic Gaussian $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, I)$, and K is set to 1. Unless specified otherwise, these choices are be maintained

[3]We assume the KL divergence can not be obtained in closed form for generality.

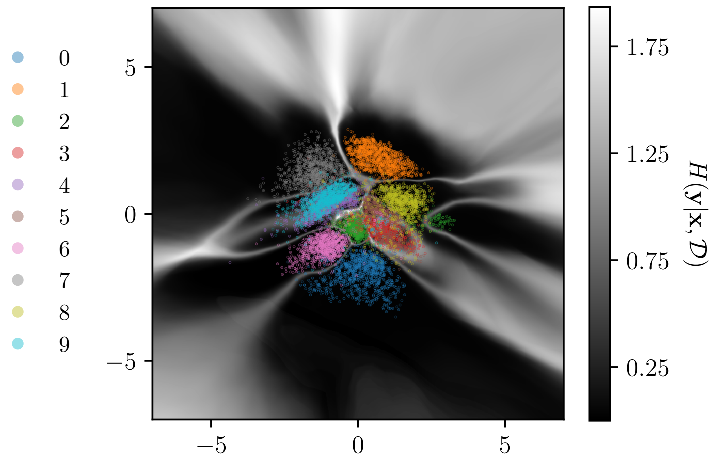

Fig. 2.4 Predictive entropy estimates for artificial MNIST digits generated from a 2-dimensional VAE latent space. The MNIST test set digits have been projected onto the latent space and are displayed with a different colour per class.

throughout this work. The reparameterisation trick (Kingma and Welling, 2014; Rezende et al., 2014) is used with (2.23) to obtain unbiased estimates of the gradient of (2.22). This allows θ and ϕ to be learnt jointly.

The architectural choice of combining an inference network (encoder) with a generative model (decoder) is strongly reminiscent of autoencoder networks (Bengio et al., 2013). This similarity is strengthened by the dimensionality of $\mathbf{z}$ typically being chosen to be substantially smaller than that of $\mathbf{x}$. The key difference between a VAE and a regular autoencoder is the KL term in the ELBO. This term ensures that the approximate posterior does not collapse to a point estimate of $\mathbf{z}$, as is the case in an autoencoder. Intuitively, to minimise the effects of the stochasticity of sampling $\mathbf{z}$ to the reconstruction objective, points nearby in latent space should map to similar points in the input domain. This results in a structured latent space and allows for the generation of new datapoints through ancestral sampling.

Visualising Uncertainty in Latent Space

A well known issue with VAEs is that the region of latent space where the encoder places probability mass, also known as the aggregate posterior,

$$q_\phi(\mathbf{z}) = \int q_\phi(\mathbf{z}|\mathbf{x})p(\mathbf{x})\,d\mathbf{x} \tag{2.24}$$

does not match the prior $p(\mathbf{z})$.

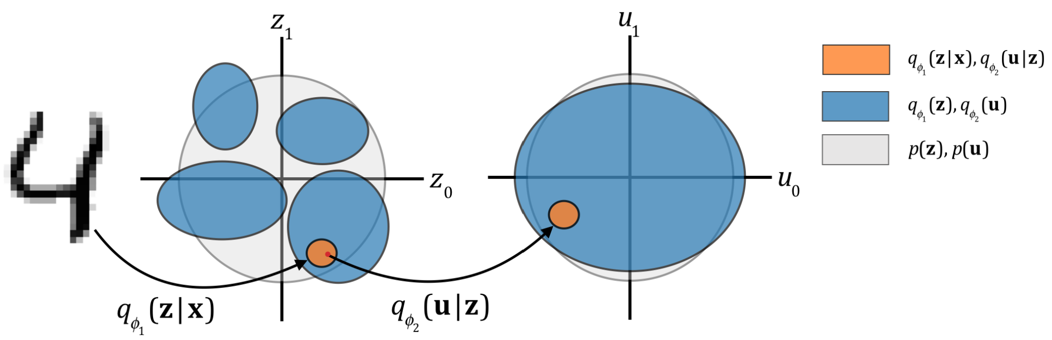
z_1
z_0
u_1
u_0
$q_{\phi_1}(\mathbf{z}|\mathbf{x})$
$q_{\phi_2}(\mathbf{u}|\mathbf{z})$
$q_{\phi_1}(\mathbf{z}|\mathbf{x}), q_{\phi_2}(\mathbf{u}|\mathbf{z})$
$q_{\phi_1}(\mathbf{z}), q_{\phi_2}(\mathbf{u})$
$p(\mathbf{z}), p(\mathbf{u})$

Fig. 2.5 In its first stage, the two-level VAE maps input samples to approximate posteriors in the outer latent space. The aggregate posterior over this latent space need not resemble the isotropic Gaussian prior. The second VAE maps samples from the outer latent space to approximate posteriors in the inner latent space. The aggregate posterior over the inner latent space more closely matches the prior.

To visualise this phenomenon, we train a BNN and a VAE on MNIST. We sample points from the VAE's latent space and evaluate their uncertainty with the BNN. This same methodology is employed by Gal and Smith (2018); Smith and Gal (2018) to analyse the vulnerability of BNNs to adversarial examples. In chapter 4, we use a similar setup as a part of CLUE, our proposed uncertainty interpretability method.

As shown in fig. 2.4, clusters of same-class digits form in latent space. The aggregate posterior presents low density in the spaces between clusters. Digits generated from these areas are of low-quality, causing our BNN to be uncertain. The outer regions of latent space, where the isotropic Gaussian prior has low density, also generate uncertain digits.

2.3.2 Two-Level Variational Autoencoders

The issue of distribution mismatch in VAEs is, in part, responsible for these models producing worse quality samples than other DGMs and has been studied at length (Antoran and Miguel, 2019; Rosca et al., 2018). Recently, Dai and Wipf (2019) have proposed the two-level VAE as a solution to distribution mismatch. After training a standard VAE, a second VAE is trained on samples from the first VAE's latent space. As illustrated in fig. 2.5, the aggregate posterior over the inner latent variables, which we denote by $q(\mathbf{u})$, more closely resembles the prior. The joint distribution over inputs and latent variables factorises as: $p(\mathbf{x},\mathbf{z},\mathbf{u}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}|\mathbf{u})p(\mathbf{u})$. We refer the reader to (Dai and Wipf, 2019) for a detailed analysis.

We implement a two-level VAE and validate its efficacy on the MNIST dataset. Figure 2.6 shows that, while generating digits from samples of $p(\mathbf{z})$ results in a large amount of low-

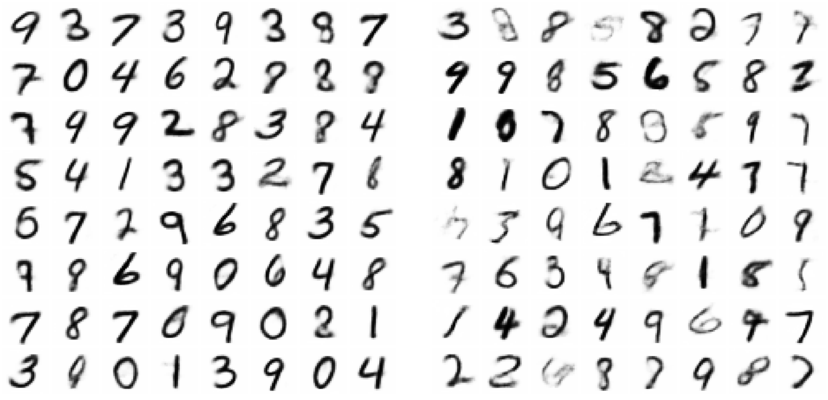

Fig. 2.6 Left: Digits generated from the inner latent space of a two-level VAE trained on MNIST. Right: Digits generated from the latent space of a VAE trained on MNIST. $\mathbf{u}$ and $\mathbf{z}$ are drawn from $\mathcal{N}(\mathbf{0}, I)$.

quality or OOD reconstructions, samples from $p(\mathbf{u})$ map to clean digits. The two-stage mechanism restores the VAE's pivotal ancestral sampling capability. In section 5.3, we use two-level VAEs to generate high-quality samples as part of our proposed framework for measuring the fidelity of counterfactual uncertainty explanations.

2.3.3 Variational Autoencoders with Arbitrary Conditioning

We have seen how high-quality artificial samples can be generated using VAEs. Unfortunately, unlike other DGMs (Douglas et al., 2017; Germain et al., 2015; Iizuka et al., 2017), the stock VAE framework provides no straight forward way to perform conditional generation. Recently, Ivanov et al. (2019) proposed the Variational Autoencoder with Arbitrary Conditioning (VAEAC), a VAE-based solution to this problem.

The objective is to model $p(\mathbf{x}_B|\mathbf{x}_{U\setminus B})$, where U refers to the set of all input dimensions and B is an arbitrary subset. To achieve this, the VAEAC substitutes the VAE's prior distribution with a prior network that parametrises a diagonal Gaussian $p_\psi(\mathbf{z}|\mathbf{x}_{U\setminus B}) = \mathcal{N}(\mathbf{z}; \mu_\psi(\mathbf{x}_{U\setminus B}), \sigma_\psi(\mathbf{x}_{U\setminus B})^2 \cdot I)$. The VAEAC's training objective:

$$\mathcal{F}_{\text{VAEAC}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \,||\, p_\psi(\mathbf{z}|\mathbf{x}_{U\setminus B})) \tag{2.25}$$

makes the prior network learn to predict the inference network's proposal. While the inference network has access to the complete set of features, the prior network can only see $\mathbf{x}_{U\setminus B}$. The KL divergence term acts as the training signal for the prior network, teaching it to implicitly

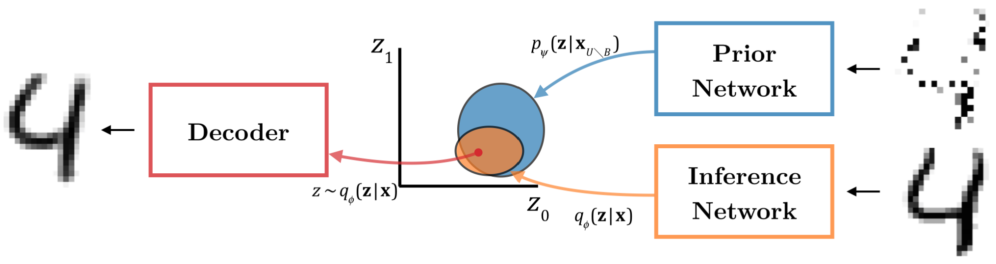
Decoder
z_1
$p_\psi(\mathbf{z}|\mathbf{x}_{U\setminus B})$
Prior Network
Inference Network
$z\sim q_\phi(\mathbf{z}|\mathbf{x})$
z_0
$q_\phi(\mathbf{z}|\mathbf{x})$

Fig. 2.7 Diagram of VAEAC training. The prior network's proposed distribution is typically wider than the inference network's, as the prior network has access to less information. The reconstruction error is estimated with a single sample from the approximate posterior.

infer $\mathbf{x}_B$ from $\mathbf{x}_{U\setminus B}$. Simultaneously, it regularises $q_\phi(\mathbf{z}|\mathbf{x})$, preventing it from collapsing to a point mass. This is illustrated in fig. 2.7. Once trained, we can use a VAEAC to sample possible feature imputation configurations:

$$p_{\theta,\psi}(\mathbf{x}_B|\mathbf{x}_{U\setminus B}) = \mathbb{E}_{p_\psi(\mathbf{z}|\mathbf{x}_{U\setminus B})}[p_\theta(\mathbf{x}_B|\mathbf{z})] \tag{2.26}$$

Intuitively, as B grows larger, there will be more possible settings for the missing features. Thus, the prior network will yield a wider proposal distribution. The VAEAC's critical advantage over alternative inpainting methods is it's capacity to give diverse explanations for $\mathbf{x}_B$ by decoding different samples from $p_\psi(\mathbf{z}|\mathbf{x}_{U\setminus B})$. We use this technique to obtain uncertainty estimates for feature imputation, in section 5.3. Figure 2.8 shows some examples of VAEAC inpaintings on MNIST.

Given that, in a VAEAC, the prior distribution is a function of $\mathbf{x}_{U\setminus B}$, we should be able to generate unconditional samples of $\mathbf{x}$ by decoding samples from $p_\psi(\mathbf{z}|\{\})$, where $\{\}$ refers to the empty set. Be that as it may, in practise, we have found this not to work very well. There is strong distribution mismatch between $p_\psi(\mathbf{z}|\{\})$ and $q_\phi(\mathbf{z})$. A more effective alternative is to adopt a two-level structure by training an auxiliary VAE on samples from $q_\phi(\mathbf{z})$. Drawing samples from the auxiliary latent space and mapping them back to the VAEAC's latent space and then to input space allows for high-quality sample generation. In this way, a single VAEAC can be used for both ancestral sampling and conditional sampling. In addition, it allows us to estimate the log-likelihood of inputs as:

$$\log p(\mathbf{x}) = \log \int p_{\theta_1}(\mathbf{x}|\mathbf{z}) p_{\theta_2}(\mathbf{z}|\mathbf{u}) p(\mathbf{u})\, d\mathbf{z}\, d\mathbf{u} \tag{2.27}$$

where parameter subscripts refer to the outer (1^{st} level) and inner (2^{nd} level) networks. In order to preserve computational tractability, we approximate $p_{\theta_2}(\mathbf{z}|\mathbf{u})$ with a point estimate placed at its mean $p_{\theta_2}(\mathbf{z}|\mathbf{u}) \approx \delta(\mathbf{z} - \mu_{\theta_2}(\mathbf{z}|\mathbf{u}))$. We further approximate (2.27) with importance sampling:

$$\log p(\mathbf{x}) \approx \log \frac{1}{K}\sum_{k=1}^{K} \frac{p_{\theta_1}(\mathbf{x}|\mathbf{z}{=}\mu_{\theta_2}(\mathbf{z}|\mathbf{u}_k))p(\mathbf{u}_k)}{q(\mathbf{u}_k|\mathbf{x})}; \quad \mathbf{u}_k \sim q(\mathbf{u}|\mathbf{x}) \tag{2.28}$$

Our proposed uncertainty explanation evaluation metric, introduced in section 5.3, uses a VAEAC for both artificial data generation and likelihood estimation.

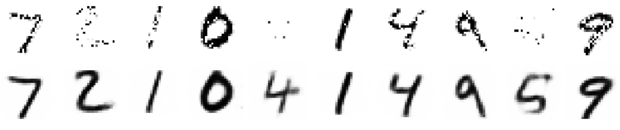

Fig. 2.8 MNIST test-set digits with pixels randomly dropped and corresponding VAEAC inpaintings.

2.4 Unsupervised Clustering with Variational Mixtures of Gaussians

The final tool we use in the development of our uncertainty interpretability methods is the variational mixture of Gaussians (Bishop, 2006). This model provides an example of how a principled Bayesian treatment can help solve the problem of model selection, specifically the selection of the number of mixture components. For an introduction to Gaussian mixture models and the EM algorithm, we refer the reader to chapter 9 of (Bishop, 2006).

As a starting point, we write the conditional distribution of the cluster assignments $\mathbf{Z}{=}\{\mathbf{z}_1 \ldots \mathbf{z}_N\}$ given the mixing proportions $\boldsymbol{\pi}{=}\{\pi_1 \ldots \pi_K\}$ and the conditional distribution of our observa-

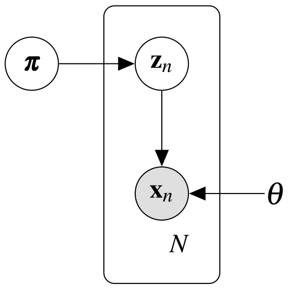

Fig. 2.9 Graphical model for our Bayesian mixture of Gaussians.

tions $\mathbf{X}$ given the cluster assignments $\mathbf{Z}$ and other distribution parameters $\boldsymbol{\theta}=\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$:

$$p(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \pi_k^{z_{nk}}; \quad p_{\boldsymbol{\theta}}(\mathbf{X}|\mathbf{Z}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{nk}} \tag{2.29}$$

We place a conjugate symmetric Dirichlet prior over $\boldsymbol{\pi}$:

$$p(\boldsymbol{\pi}) = Dir(\boldsymbol{\pi}; \boldsymbol{\alpha}_0) = C(\boldsymbol{\alpha}_0) \prod_{k=1}^{K} \pi_k^{\alpha_0 - 1} \tag{2.30}$$

The resulting graphical model is shown in fig. 2.9. We choose an approximate distribution over latent variables that factorises as: $q(\mathbf{Z}, \boldsymbol{\pi}) = q(\mathbf{Z})q(\boldsymbol{\pi})$ and use it to rewrite (2.7), defining a lower bound on the log-likelihood of the data:

$$\log p(\mathbf{X}) \geq \text{ELBO} = \int\int q(\mathbf{Z})q(\boldsymbol{\pi}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}) - \log\left(q(\mathbf{Z})q(\boldsymbol{\pi})\right)\right) d\mathbf{Z} d\boldsymbol{\pi} \tag{2.31}$$

It can be derived from (2.31) that the ELBO is maximised with respect to $q(\mathbf{Z})$ when $\log q(\mathbf{Z}) = \mathbb{E}_{q(\boldsymbol{\pi})}[\log p(\mathbf{Z}|\boldsymbol{\pi})] + \log p_{\theta}(\mathbf{X}|\mathbf{Z}) + k$, with k being a constant. This yields the variational E step update:

$$q^*(\mathbf{Z}) = \prod_{n=1}^{N}\prod_{k=1}^{K} r_{nk}^{z_{nk}}; \quad r_{nk} = \frac{\exp\left(\mathbb{E}_{q(\boldsymbol{\pi})}[\log \pi_k] + \log \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)}{\sum_{j=1}^{K} \exp\left(\mathbb{E}_{q(\boldsymbol{\pi})}[\log \pi_j] + \log \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \tag{2.32}$$

where $r_{nk} = \mathbb{E}_{q(\mathbf{Z})}[z_{nk} = 1]$ are known as responsibilities. Analogously, the ELBO is maximised with respect to $q(\boldsymbol{\pi})$ when $\log q(\boldsymbol{\pi}) = \mathbb{E}_{q(\mathbf{Z})}[\log p(\mathbf{Z}|\boldsymbol{\pi})] + \log p(\boldsymbol{\pi}) + k$, yielding:

$$q^*(\boldsymbol{\pi}) = Dir(\boldsymbol{\pi}|\boldsymbol{\alpha}); \quad \alpha_k = \alpha_0 + \sum_{n=1}^{N} r_{nk} \tag{2.33}$$

Since we are not giving a fully Bayesian treatment to $\boldsymbol{\theta}$, the remaining parameters receive the traditional M step update. In this manner, we optimise (2.31) sequentially with respect to $q(\mathbf{Z})$, $q(\boldsymbol{\pi})$ and $\boldsymbol{\theta}$. This form of procedure is used as a consequence of the expressions for $q(\mathbf{Z})$ and $q(\boldsymbol{\pi})$ depending on the moments of the other random variables.

We can select the number of mixture components by fitting multiple models with different numbers of components and evaluating their ELBO to measure how well the data is explained by each configuration. Nevertheless, this procedure can be computationally expensive. An alternative approach, proposed by Corduneanu and Bishop (2002), is to fit a model with a large number of components and use the learnt mixture proportions as a form of Automatic Relevance Determination (ARD). Components which play small roles in explaining the data will have $r_{nk} \approx 0$ and thus $N_k = \sum_{n=1}^{N} r_{nk} \approx 0$. The posterior's α_k parameter will converge to the prior value α_0. The expected mixing proportions can be obtained as:

$$\mathbb{E}_{q(\boldsymbol{\pi})}[\pi_k] = \frac{\alpha_0 + N_k}{K\alpha_0 + N} \tag{2.34}$$

When a broad prior is chosen $\alpha_0 \rightarrow 0$, the mixing proportions for unnecessary clusters go to 0: $\mathbb{E}_{q(\boldsymbol{\pi})}[\pi_k] \rightarrow 0$. We can set a threshold γ and discard all clusters with $\pi_k < \gamma$. It is worth noting that this approach is made possible by the use of a variance floor which prevents non-relevant components from fixating on individual datapoints for which they take complete responsibility.

Chapter 3

Related Work: The Language of Interpretability

While in the last chapter we explored the theory behind uncertainty and the tools we use in later chapters to construct our proposed methods, in this chapter we cover existing work in the field of machine learning interpretability. There is little overlap between the two topics in the existing literature, as there has been barely any work done in leveraging Bayesian methods to make ML models more interpretable. In spite of this, we believe it is important to review existing interpretability approaches as many of the underlying principles that have guided the development of these techniques are also applicable to uncertainty interpretability.

In section 3.1, we discuss the different dimensions across which interpretability methods can be categorised, building what we refer to as a "taxonomy of explanations." In section 3.2, we introduce the different types of approaches to evaluating and comparing uncertainty interpretability methods. Finally, in section 3.3, we discuss uncertainty sensitivity analysis (Depeweg et al., 2017) which, to the best of our knowledge, is the only existing method for interpreting uncertainty estimates in BNNs.

3.1 A Taxonomy of Explanations

With the passing of the European Union's General Data Protection Regulation (GDPR) (Council of European Union, 2016) which contains articles such as the "right to an explanation"[1] and the creation of governmental organisms such as the UK's Centre for Data

[1]It is worth noting that the text is not legally binding in its current form.

Ethics and Innovation, the already fast-growing subfield of machine learning interpretability has started receiving mainstream attention. In spite of this, the relative youth of this area of research is made evident by the lack of an agreed-upon taxonomy of machine learning interpretability (Doshi-Velez and Kim, 2017). Methodological contributions to the field are often paired with the introduction of an entirely new dimension of categorisation in which it is convenient to explain the method being proposed. A common language of interpretability is critical to the comparison of related techniques and, more broadly, to the maturing of the field. For this reason, we dedicate this section to reviewing some of the existing categories with which interpretability algorithms are described. This lays the groundwork to frame CLUE, our proposed interpretability method, in the context of the existing literature. We give special attention to counterfactual explanations because CLUE is most similar to existing methods within this class of interpretability techniques.

3.1.1 How Interpretability is Achieved

Explaining a machine learning model's predictions requires making use of an inherently interpretable model or creating a secondary, more easily understandable, post-hoc model to explain the original model.

After-the-Fact Explanations

We distinguish between two possible scenarios, explaining black-box and white-box models (Guo et al., 2018). In the former, we only have access to a model's inputs and outputs. LIME, proposed by Ribeiro et al. (2016), is the most popular example of a black-box method. It measures a model's responses to perturbed versions of a test sample and uses them to build a linear approximation to the model's behaviour around the sample in question. This approach has been extended by the introduction of kernels designed to make the weights of linear approximations more faithful to the original model (Lundberg and Lee, 2017).

Black-box methods have the advantage of being model-agnostic. However, access to a model's internals can allow for less crude approximations to its reasoning. The majority of neural network interpretability techniques fall within the white-box category. Dabkowski and Gal (2017) use a network's activations to train a secondary network to output saliency maps. Kim et al. (2018) fit hyperplanes in activation space to probe for human-defined concepts in the representations learnt by a network. Procedures such as Integrated Gradients

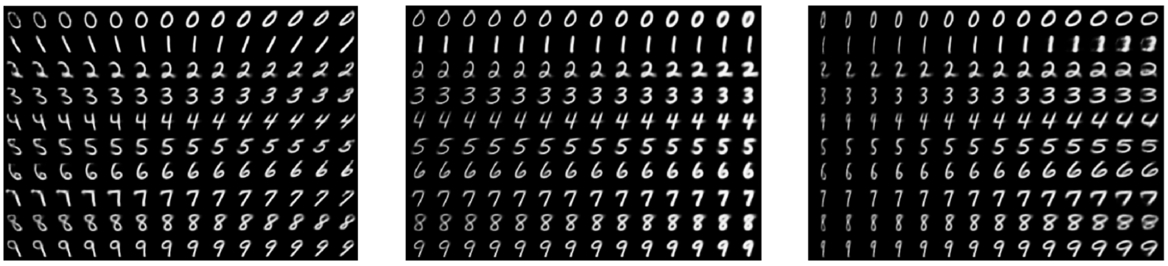
Fig. 3.1 Left: Latent traversals for the azimuth dimension. Centre: Latent traversals for stroke thickness. Right: Latent traversals for digit width. Figure taken from (Antoran and Miguel, 2019).

(Sundararajan et al., 2017) or DeepLift (Shrikumar et al., 2017) extract information from the loss function's gradients with respect to the input features.

Interpretable Models

A common criticism levelled at post-hoc interpretability techniques is that their explanations can not have perfect fidelity with respect to the original model; otherwise, the explanations would have to be as complicated as the model itself. Rudin (2019) advocates for the use of inherently interpretable models, like logistic regression or decision trees, paired with input features which are meaningful to humans. Interestingly, Gaussian processes (GP) can also fall within the category of inherently interpretable models. Lloyd et al. (2014) show that the compositional properties of some kernels allow for the explanation of GP regression models in terms of the addition and product of simple functions.

Complex models, like neural networks, can become inherently interpretable if they provide the user with both predictions and explanations. This is often achieved through the use of a multi-objective optimisation scheme where one of the tasks consists of generating an explanation. This approach is used by Chen et al. (2018a), who train an image classifier with an auxiliary objective of finding image segments that act as prototypes for individual classes. For instance, for the class "dog," a prototype might contain a set of floppy ears. At test time, the model provides predictions together with the most relevant prototypes. A similar approach is taken by Alvarez Melis and Jaakkola (2018), where a prototype generator is trained together with a relevance detector. Classifications are made based on the relevance of each prototype. This same information is provided to the user as an explanation.

A tangential field of work is the development of models capable of learning interpretable representations in a latent space without the use of labels. The subfield of Disentangling (Higgins et al., 2017) aims to do this in a way such that each generative factor of the data is explained by a single latent dimension. These factors can be altered by traversing the

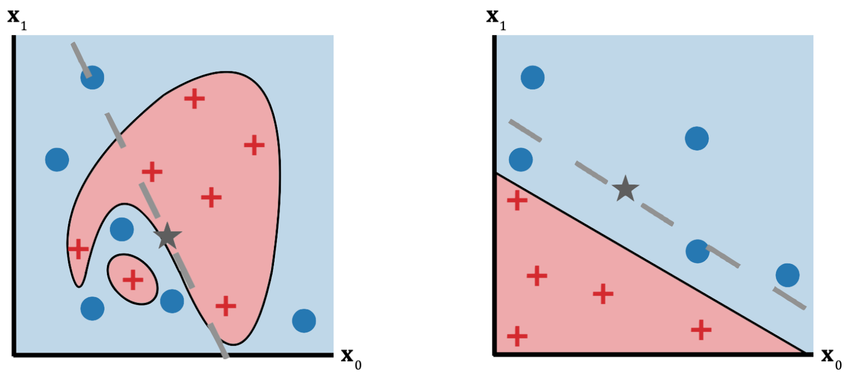

Fig. 3.2 Toy examples of linear approximations to classification models at specific testpoints (marked with a star). Right: A complex non-linear model. Left: Logistic regression.

latent space, as shown in fig. 3.1. Adel et al. (2018) extend this concept by guiding the learnt representations with side-channel human supervision.

3.1.2 Local and Global Explanations

As exemplified in fig. 3.2, linear approximations to non-linear models, such as the ones made by LIME (Ribeiro et al., 2016), are often only faithful in small regions of input space. Owing to this, they can lead us to draw incorrect conclusions. An explanation for a text sentiment classifier may highlight the word "bad" in the expression "not bad" as contributing to a positive classification. An explanation for loan default predictor might tell a client that they have been denied a loan because they have too many credit cards open while approving a different client that has a larger number of cards. Explanations which are only guaranteed to apply to a specific test point are known as local explanations.

On the other hand, fig. 3.2 shows how, for logistic regression, linear approximations taken at any point of input space are parallel to the decision boundary. Any explanations that we build for this model apply to all possible inputs to the model. We refer to these as global explanations.

Although global explanations are generally preferable to local ones, they can be impossible to obtain for complex models and may not always be required. A person who has just been denied a loan might not need a full understanding of the model that decided to deny them the loan. Instead, they will simply be interested in learning about what changes they can make in order to flip the classifier's predictions. Ustun et al. (2019) propose the creation of "flipsets," exemplified in table 3.1, for critical decisions made automatically. Alternatively, local explanations can be grouped into summaries or trends (Chen et al., 2018b). These can

Table 3.1 Example of a flipset for a person who is denied a loan by an automatic classifier. The proposed changes guarantee that the model's prediction flips from -1 to 1, all other inputs staying the same. Example taken from (Ustun et al., 2019).

Feature	Current Value		Required Value
n_credit_cards	5	$\rightarrow$	3
current_debt	£3250	$\rightarrow$	£1000
has_savings_account	False	$\rightarrow$	True
has_retirement_account	False	$\rightarrow$	True

give a general overview of a model's reasoning and will not be misleading as long as it is clearly specified to which regions of input space each trend applies. In this vein, Ribeiro et al. (2018) propose an algorithm that finds "anchors": sets of features that, if present, guarantee a certain prediction. Prototype images (Nguyen et al., 2017) can also be seen as summaries.

3.1.3 The Evidence Space for Explanations

Most post-hoc interpretability methods explain models' decisions in terms of which input features alter a prediction the most when replaced with a reference background. The reference is usually chosen to be uninformative with respect to the task at hand. For MNIST digit classification, a completely black image is usually selected. However, for more complex datasets, like ones composed of real-world images or natural language, there may not be a good single choice of reference. In these cases, the most common choice is an OOD background, like a black image or an all-zero embedding vector (Lundberg and Lee, 2017; Shrikumar et al., 2017; Sundararajan et al., 2017).

Adversarial examples (Goodfellow et al., 2015) are able to change a classifier's predictions by imperceptibly changing its input in a way that pushes it off the manifold of the data. Analogously, the choice of an OOD reference can cause meaningless explanations to be constructed. An example of this pathological behaviour is illustrated in fig. 3.3.

In response to this issue, Zintgraf et al. (2017) propose to substitute pixels with their expected values, which are computed from the surrounding pixels, rather than using reference fill-ins. Pixels which cause a substantial change in the classifier's output when replaced with their expectations are assigned high saliency. Further work by Chang et al. (2019) approximates expectations using a conditional generative model. In this way, the explanations that the method can provide are restricted to the input manifold captured by the generative model.

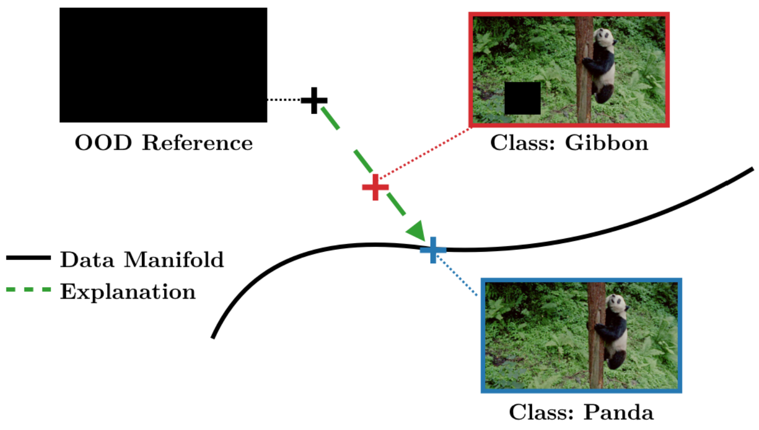

Fig. 3.3 An interpretability method is used to explain why an image (blue cross) is classified as a panda. An adversarial example, formed by the original input with an overlayed black box (red cross), lies between the original image and the reference image. An explanation is built by finding which parts of the original image contribute the most to the "panda" prediction when replaced by the reference image. Since the adversarial example is classified as "gibbon" and its difference with the original image is the black square, the square will be assigned high saliency.

The authors provide experimental validation showing that stronger generative models are able to change the classifier's predictions by replacing fewer pixels.

There has also been research into the use of generative models for the creation of prototypes for image classification (Nguyen et al., 2017; Nguyen et al., 2016). This is most commonly done by solving the following optimisation problem:

$$\arg\max_{\mathbf{x}} \left(p_I(c{=}y|\mathbf{x}) + p_G(\mathbf{x})\right) \tag{3.1}$$

where I refers to an image classifier, y is the class of interest and G is a generative model. Failure to include the $p_G(\mathbf{x})$ term results in the creation of an OOD sample to which classifier assigns high probability.

3.1.4 The Cognitive Chunks of an Explanation

The most common form of machine learning model explanation is the saliency map (Chang et al., 2019; Dabkowski and Gal, 2017; Dhurandhar et al., 2018; Lundberg and Lee, 2017; Shrikumar et al., 2017; Sundararajan et al., 2017; Zintgraf et al., 2017). This type of explanation assigns a different weight to each input feature to indicate the magnitude and polarity of its contribution to a prediction. It is worth noting, however, that this type of explanation is often criticised for only tell us "where the model is looking" and not "why the model made a decision" (Rudin, 2019).

While input domains are often high dimensional, humans can only consider 7 ± 2 cognitive entities at once (Cowan, 2010). Consequently, sparsity is key in making explanations human-friendly. Sparse explanations allow us to consider variables jointly, as opposed to individually. The simplest way to achieve sparsity is to reduce the dimensionality of our explanations by grouping input variables with similar meanings, like adjacent pixels. A different approach is to try to maximise the independence between the blocks that form an explanation. Explanations with large amounts of input variables can become understandable if we can consider each variable on its own. Alternatively, mutual interactions between input features can be made understandable by expressing them in terms of high-level concepts. Examples of such concepts are the disentangled generative factors from MNIST shown in fig. 3.1. Although rotating a digit involves changing many of the image's input pixels, an explanation given in terms of digit rotation is easily understandable. This approach is adopted by Adel et al. (2018); Kim et al. (2018); Olah et al. (2018).

3.1.5 Counterfactual Explanations

Using the term somewhat suggestively, the machine learning interpretability community refers to counterfactual explanations (Wachter et al., 2017) as those that answer questions of the form:

"*With all else being the same, what would a user's income have to have been for them to have been approved for a loan given that their monthly salary was* £$1,000$ *and they were denied the loan?*"

Unlike saliency maps or prototypes, which aim to shed light on the internal reasoning of automated decision making systems, counterfactuals focus on conveying how their input variables need to be modified in order to change the systems' outputs desirably. Many counterfactual explanations can exist for a given input, as there may be multiple inputs that provide the same output for a machine learning model. Naturally, a correct answer to the question above might be: "*the person would have been approved if their monthly income had been* £$1,000,000$." However, this response is less useful than: "*the person would have been approved if their monthly income had been* £$2,000$." We are generally concerned with the counterfactual set of input variables that is closest to the original input configuration.

Counterfactual examples are usually built by solving an optimisation problem that resembles:

$$\arg\max_{\mathbf{x}} \left(p_I(c{=}y|\mathbf{x}) - d(\mathbf{x}, \mathbf{x}_0)\right); \quad \arg\max_{c} p_I(c|\mathbf{x}_0) \neq y \tag{3.2}$$

where y is the desired outcome, $\mathbf{x}_0$ refers to the original input to the predictor and $d(\cdot)$ is some pairwise distance metric. In their work, Dhurandhar et al. (2018) use a combination of L1 and L2 norms as distance metrics. Chang et al. (2019) attempt to find counterfactuals where the number of input pixels that change with respect to the original image is minimum. It is also important for the proposed counterfactuals to represent realistic scenarios and not exploit weaknesses in the predictive model. Telling a person that they would have been approved for a loan had their age been -10 is of very little use. Both works mentioned above use generative models to constrain their explanations such that they are in-distribution.

Ustun et al. (2019) observe that not all input features to a predictive model are equally mutable. Informing someone that they would have been approved for a loan had they attended a different university does not give them a direct path towards changing their situation. The authors introduce the concept of actionable recourse. They attempt to find counterfactuals which minimise a user-defined actionability cost.

FIDO: State of The Art Counterfactual Explanations

Fill-In the DropOut region (FIDO) is a state-of-the-art counterfactual interpretability method for images which was recently introduced by Chang et al. (2019). We give special attention to this method as, in section 5.1.2, we propose a simple adaption to FIDO that allows it to explain uncertainty estimates in BNNs and use it as a baseline with which to compare CLUE, our proposed method.

FIDO aims to find the smallest region B that needs to be deleted and substituted with an uninformative input in order to minimise the classification score: $s_I(c|\mathbf{x}) = \log p_I(c|\mathbf{x}) - \log(1 - p_I(c|\mathbf{x}))$. A binary mask $\mathbf{z}$ is used to express which pixels are be substituted, $z{=}0 \forall z \in B$. Each masking pixel is modelled with a Bernoulli distribution: $p_{\boldsymbol{\rho}}(\mathbf{z}) = \prod_{u \in U} Bern(z_u; \rho_u)$. A conditional generative model G is used to substitute masked pixels with their expectation $\mathbb{E}_{p_G(\mathbf{x}_B|\mathbf{x}_{U\setminus B})}[\mathbf{x}_B]$. The resulting image takes the form:

$$\phi(\mathbf{x}, \mathbf{z}) = \mathbf{z} \odot \mathbf{x} + (\mathbf{1} - \mathbf{z}) \odot \mathbb{E}_{p_G(\mathbf{x}_B|\mathbf{x}_{U\setminus B})}[\mathbf{x}_B] \tag{3.3}$$

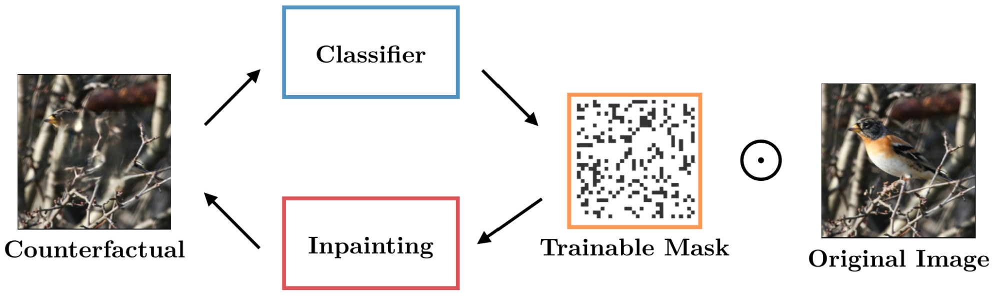

Fig. 3.4 Diagram of FIDO's optimisation procedure. The original image is classified as a brambling with 99.9% confidence. After the bird has been replaced by a suitable background, the probability of the image being a brambling drops to 25.9%, as reported by Chang et al. (2019).

The following objective is minimised with respect to $\boldsymbol{\rho}$ using gradient optimisation:

$$\mathscr{L}(\boldsymbol{\rho}) = \mathbb{E}_{p_{\boldsymbol{\rho}}(\mathbf{z})}[s_I(c|\phi(\mathbf{x},\mathbf{z})) + \lambda ||\mathbf{z}||_1] \tag{3.4}$$

This procedure is depicted in fig. 3.4. The expectation in (3.4) is approximated using Monte Carlo. A biased estimate of gradients with respect to $\boldsymbol{\rho}$ is obtained using the Gumbel-softmax categorical reparametrisation (Jang et al., 2016). The use of a generative model for inpainting ensures that FIDO's explanations are in-distribution. The L1 constraint in (3.4) ensures that FIDO's explanations are sparse. Apart from generating counterfactuals, FIDO's masks can be used as a saliency maps.

3.2 Evaluating Explanations

In the machine learning community, authors are expected to prove the merits of their proposed methods by showing their success on tasks that the community has deemed appropriate. Examples of this are surpassing human performance on a strategy game with a reinforcement learning agent or achieving a large test-set log-likelihood with a generative model. Analogously, this section is dedicated to laying out a hierarchy of evaluation methods for ML interpretability. We adopt the taxonomy proposed by Doshi-Velez and Kim (2017).

3.2.1 Domain Experts and Real Tasks

The end goal of a ML interpretability method is for it to be able to assist humans on a real task. It follows that the gold standard of evaluation methods is having a human domain-expert asses the explanations given by an interpretability method on the task of interest. Be that as it may, this type of evaluation is rarely performed in practise. Access to domain experts is usually limited and a separate evaluation would need to be performed for every task on which we wish to use our interpretability method. Dhurandhar et al. (2018) validate their saliency maps on a task where fMRI images are used to determine if a subject has autism. Neuroscientists are asked to confirm that the provided explanations align with their criteria. Williams et al. (2016) validate a homework hint system based on if students using it achieve better performance.

3.2.2 Human Subjects and Proxy Tasks

A slightly more accessible approach to interpretability method evaluation is to remove the need for domain experts and use lay humans. In some cases, due to lack of expertise in the field, the test subjects are not able to judge the performance of the interpretability method on the task of interest. Instead, a proxy task is used. Ribeiro et al. (2018) evaluate their method using university students as test subjects. The students are asked to predict a classifier's decision for a series of inputs without having seen any explanations and later with a set of reference explanations given by their proposed interpretability method. In this way, they show that their explanations are effective and apply globally. They also ask the students to subjectively judge which explanations they find most helpful between the ones generated with their proposed method and a baseline.

3.2.3 Functionally-Grounded Evaluation

The most common type of evaluation is the one that requires no human interaction. Instead, an automatically computed metric of explanation quality is employed. Although this class of evaluation is the least predictive of real-world performance, it is often the only one available to machine learning researchers with limited resources. It is also the only one available to practitioners who aim to tackle problems in fields in which they are non-experts. For this reason, it is important to develop reliable functionally-grounded metrics. In section 5.3, we propose a metric of this kind for uncertainty interpretability methods.

Dabkowski and Gal (2017) propose to evaluate image saliency maps by finding the smallest rectangular crop that contains all the salient pixels, scaling it to the size of the original image, and evaluating its class probabilities under the model of interest. Small crops and large class probabilities contribute to a better score on their metric. They also propose a metric based on the overlap between the minimum bounding box that contains all salient pixels and a ground-truth bounding box. Another popular metric for evaluating saliency maps is selectivity (Bach et al., 2015; Chang et al., 2019). This quantity is measured as the number of most salient pixels which need to be replaced by a reference value such that the classifier's confidence drops below a threshold. Although this last method could potentially be adapted to uncertainty explanations, all of these metrics are limited by only being applicable to image saliency maps. In this work, we also concern ourselves with tabular data. Here, having our explanations be close to the original datapoints in input space is equally as important as them being sparse. Our proposed figure of merit for uncertainty explanations, presented in section 5.3, reflects this by considering the L1 distance between the original inputs and explanations.

Ustun et al. (2019) propose to evaluate counterfactual explanations based on some user defined metric of actionability. Chen et al. (2018b) evaluate summary explanations or trends based on their sparsity and to what proportion of test points they are applicable. For the case of unsupervised interpretable representation learning, Higgins et al. (2017) propose a metric that measures the independence and axis alignment of latent representations. Adel et al. (2018) propose a metric which probes for a linear relationship between points in a latent space and manually provided side information.

3.3 Explaining Uncertainty in Neural Networks

In chapter 2, we saw how probabilistic modelling can be used to build ML models which provide uncertainty estimates together with their predictions. This is an important step towards building more reliable and interpretable systems. It also opens the door to asking new questions about our models' reasoning.

Consider an active learning scenario where a BNN is trained to predict faults in an industrial process. New parameter settings for the process are proposed by the model such that their estimated mutual information with the model's parameters is maximum. Useful insights into the industrial process might be gained by understanding why our model believes that the selected point is going to be the most informative. Additionally, finding which regions of input space the model believes to be the most aleatoric can help us detect the influence of external factors on the industrial process.

Unfortunately, despite surging mainstream interest in Bayesian deep learning, there has been very little research into interpreting uncertainty estimates for complex models.

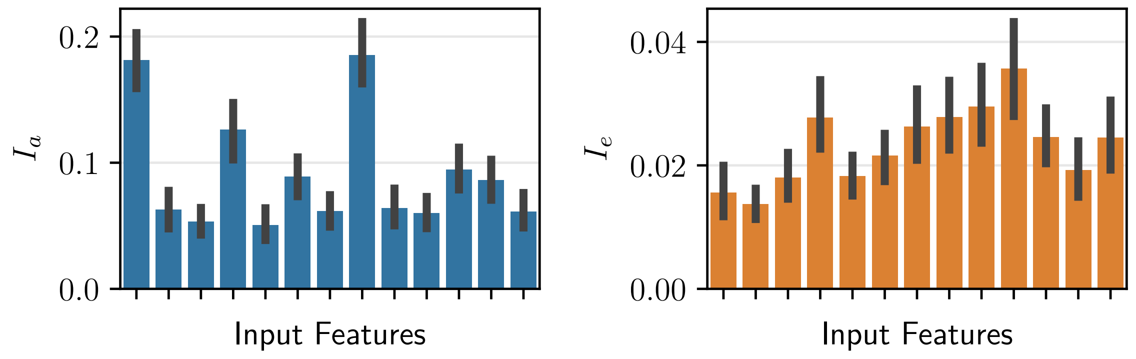

Fig. 3.5 Left: Aleatoric standard deviation sensitivity analysis for the Boston housing dataset (Dua and Graff, 2017). Right: Epistemic standard deviation sensitivity analysis for the same dataset.

3.3.1 Uncertainty Sensitivity Analysis

To the best of our knowledge, the only existing method for the interpretation of uncertainty estimates is Uncertainty Sensitivity Analysis (Depeweg et al., 2017). Analogously to regular sensitivity analysis (Shu and Zhu, 2019), this method aims to quantify the relevance of individual input dimensions to a chosen metric of uncertainty. This is done by averaging the gradients of the uncertainty metric with respect to the input dimension of choice over all test datapoints:

$$I_i = \frac{1}{|\mathscr{D}_{\text{test}}|} \sum_{n=1}^{|\mathscr{D}_{\text{test}}|} \left| \frac{\partial H(\mathbf{y}_n^* | \mathbf{x}_n^*)}{\partial x_{n,i}^*} \right| \tag{3.5}$$

In this expression, predictive entropy H is used as the uncertainty metric of choice. However, it can be replaced by any other metric like aleatoric entropy, epistemic entropy or standard deviations. An example of this type of analysis is shown in fig. 3.5.

Equation (3.5) can be seen as an approximation to a global explanation of a model's uncertainty built as a sum of linear approximations centred at each testpoint. As discussed in section 3.1, this approach can be misleading because predictive uncertainty metrics are not linear functions of the input variables. This is especially true for strongly non-linear models like NNs. In these models, a shift in a low sensitivity input dimension could plausibly cause a larger change in predictive uncertainty than the same shift applied to a high sensitivity dimension.

In high-dimensional input spaces, there is a very strong chance that $\nabla_{\mathbf{x}} H$ does not point in the direction of the data manifold. This results in meaningless explanations. In fig. 3.6,

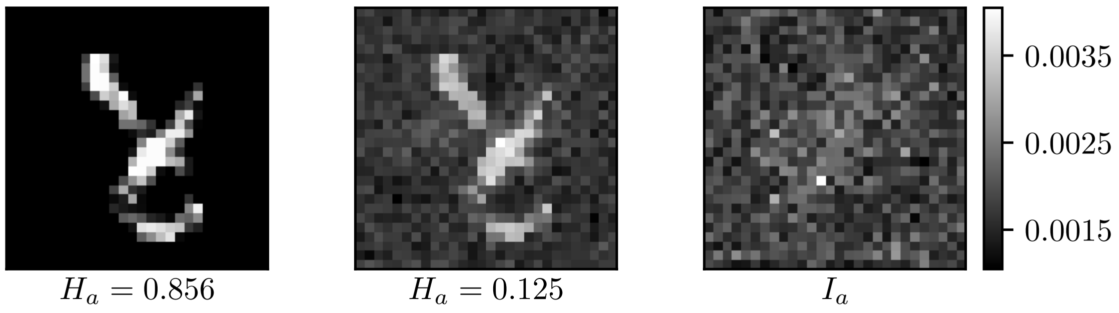

Fig. 3.6 Left: A digit from the MNIST test set with large aleatoric entropy. Centre: The same digit after a step is taken in the direction of $-\nabla_{\mathbf{x}} H_a$. Right: Aleatoric sensitivity analysis for the MNIST test set.

we show an example where a step in the direction of $-\nabla_{\mathbf{x}} H_a$ leads to a seemingly noisy input configuration for which the estimated aleatoric entropy is low. Aggregating these steps for every point in the test set leads to an uncertainty sensitivity analysis explanation that resembles white noise.

Chapter 4

Getting a CLUE: A Novel Method for Explaining Uncertainty

In this chapter, we propose a novel method for uncertainty interpretability, which we call Counterfactual Latent Uncertainty Explanations (CLUE). We refer to explanations given by our method as CLUEs. A CLUE provides an answer to the question:

"*What is the smallest change that would have had to be made to an input, while maintaining it in-distribution, for our model to have been more confident in its decision about said input?*"

CLUE is a versatile method, being able to provide explanations for both regression and classification tasks on both tabular data and images.

In section 4.1, we explain the core CLUE algorithm, framing it in the context of the taxonomy of interpretability methods introduced in section 3.1. We describe the application of CLUE to both image data and tabular data. In section 4.2, we propose an approach to clustering CLUEs which allows us to generate summaries of the sources of uncertainty in a dataset.

4.1 Generating Counterfactual Explanations for Uncertainty

A counterfactual explanation for uncertainty aims to find a setting of the input parameters **x** that causes a ML model's uncertainty to decrease with respect to its uncertainty for a reference input $\mathbf{x}_0$. This objective allows us to generate counterfactual explanations for regression, where standard deviations can be used as uncertainty metrics, and classification, where predictive entropy is used.

As seen in section 3.1.5, existing work on counterfactual machine learning interpretability has mostly focused on classification tasks. Explanations are typically generated by finding ways to flip models' predictions. In contrast, there is no clear way to build meaningful counterfactual explanations for regression. Making use of predictive uncertainty allows us to arrive at an intuitive solution to this problem. By finding inputs which correspond to similar predictions but with smaller error bars, we can interrogate our model about which input factors it deems important.

While we expect our model to be certain about a limited range of input values, there are likely an infinite amount of input configurations which result in high uncertainty. For this reason, we limit ourselves to asking questions about how we can increase model confidence rather than the opposite.

CLUE can be applied to any model which produces an uncertainty metric that is differentiable with respect to its inputs. It requires the repeated evaluation of these gradients, making it a post-hoc white-box interpretability method. CLUE produces explanations for individual points in input space, making it a local method. However, it does not rely on crude linear approximations. Because it is also a counterfactual method, CLUE's explanations have a tangible meaning. CLUE uses a DGM to ensure that its explanations represent plausible settings of the input parameters. In this work, we use a VAE. It is worth noting that this is not the only possible choice. Other differentiable DGMs, such as GANs, could potentially also be employed. It is important to choose an expressive model, as our method's explanations are limited to the input parameter configurations that the DGM can express.

4.1.1 The CLUE Algorithm

Without loss of generality, we use H to refer to a differentiable uncertainty metric. We use $\mathbf{x}_0$ to refer to the original input for which we want to find a counterfactual explanation. Following the notation introduced in section 2.3.1, our VAE's encoder is denoted as $q_\phi(\mathbf{z}|\mathbf{x})$ and the decoder is $p_\theta(\mathbf{x}|\mathbf{z})$. We write these models' predictive means as $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\mathbf{z}]=\mu_\phi(\mathbf{z}|\mathbf{x})$ and $\mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[\mathbf{x}]=\mu_\theta(\mathbf{x}|\mathbf{z})$ respectively.

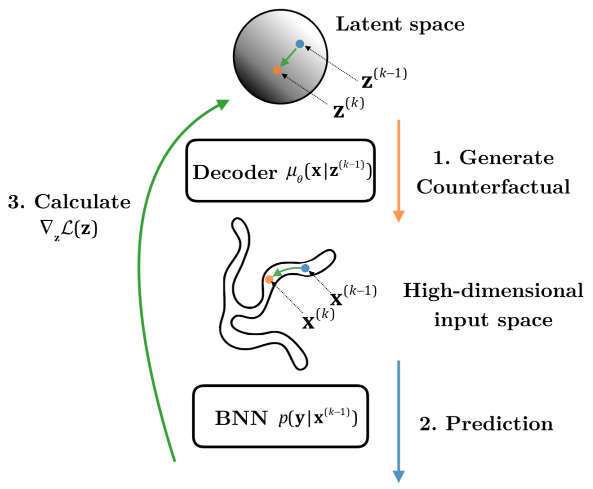
Latent space
$z^{(k-1)}$
$z^{(k)}$
Decoder $\mu_\theta(x|z^{(k-1)})$
1. Generate Counterfactual
3. Calculate $\nabla_z \mathcal{L}(z)$
$x^{(k-1)}$
$x^{(k)}$
High-dimensional input space
BNN $p(y|x^{(k-1)})$
2. Prediction

Fig. 4.1 Illustration of CLUE's optimisation loop. Latent codes are decoded into input samples for which a BNN generates uncertainty estimates. Gradients of these estimates are backpropagated through the BNN and the decoder back to the latent space. Simple changes in latent space correspond to complex changes in input space.

CLUE tries to find points in latent space which generate an input close to $\mathbf{x}_0$ while being assigned low uncertainty. This is achieved by minimising the following objective:

$$\mathscr{L}(\mathbf{z}) = H(\mathbf{y}|\mu_\theta(\mathbf{x}|\mathbf{z})) + \lambda \, ||\mu_\theta(\mathbf{x}|\mathbf{z}) - \mathbf{x}_0||_1 \tag{4.1}$$

The L1 norm is chosen as a distance metric in order to encourage sparse explanations. The hyperparameter λ controls the trade-off between producing low uncertainty outputs and outputs which are close to the original input. We usually set $\lambda = \lambda_0/n(\mathbf{x})$, where $n(\cdot)$ is a function that returns the number of elements in a vector, in order to make this hyperparameter independent of the input dimensionality. CLUEs are obtained as:

$$\mathbf{x}_{\text{CLUE}} = \mu_\theta(\mathbf{x}|\mathbf{z}_{\text{CLUE}}); \quad \mathbf{z}_{\text{CLUE}} = \arg\min_{\mathbf{z}} \mathscr{L}(\mathbf{z}) \tag{4.2}$$

One advantage of generating explanations with a DGM is that CLUE behaves well when explaining OOD samples. In these cases, CLUE finds the code in latent space which is closest to generating the input, returning an in-distribution analogue for our OOD point.

The CLUE objective (4.1) is minimised with a gradient optimiser. A diagram of this procedure is shown in fig. 4.1. It can be applied to batches of inputs simultaneously, allowing us to leverage GPU accelerated matrix computation. To facilitate optimisation, the initial value of $\mathbf{z}$ is chosen to be $\mathbf{z}_0 = \mu_\phi(\mathbf{z}|\mathbf{x}_0)$. The resulting algorithm is given in algorithm 1.

```
Algorithm 1: Counterfactual Latent Uncertainty Explanations (CLUE) Method
  Inputs: original datapoint $\mathbf{x}_0$, regularisation parameter $\lambda$, BNN capable of producing
          uncertainty estimates $H$, DGM decoder $\mu_\theta(\cdot)$, (optional) DGM encoder $\mu_\phi(\cdot)$
1 Set initial vale of $\mathbf{z} = \mu_\phi(\mathbf{z}|\mathbf{x}_0)$;
2 while loss $\mathscr{L}$ is not converged do
3     Decode an input-space sample from the latent vector: $\mathbf{x} = \mu_\theta(\mathbf{x}|\mathbf{z})$;
4     Use BNN to obtain $H(\mathbf{y}|\mathbf{x})$ with any of the expressions presented in section 2.2.1;
5     Compute objective: $\mathscr{L} = H(\mathbf{y}|\mathbf{x}) + \lambda||\mathbf{x} - \mathbf{x}_0||_1$;
6     Update $\mathbf{z}$ with $\nabla_\mathbf{z} \mathscr{L}$;
7 end
8 Decode explanation from latent vector: $\mathbf{x}_{\text{CLUE}} = \mu_\theta(\mathbf{x}|\mathbf{z})$;
  Outputs: counterfactual example $\mathbf{x}_{\text{CLUE}}$, saliency map $(\mathbf{x}_{\text{CLUE}} - \mathbf{x}_0)$
```

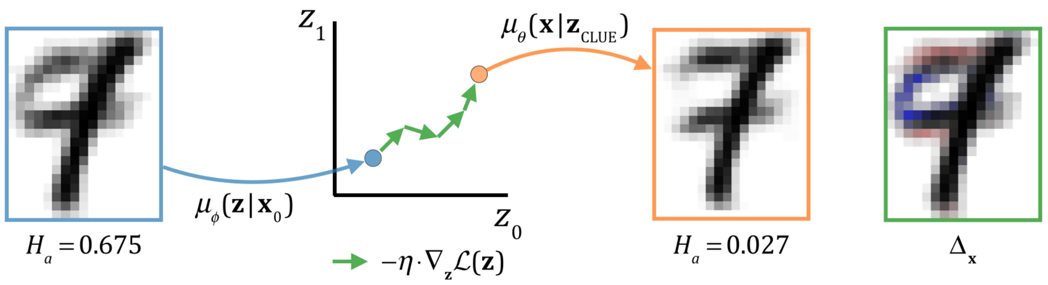
z_1
$\mu_\theta(\mathbf{x}|\mathbf{z}_{\text{CLUE}})$
$\mu_\phi(\mathbf{z}|\mathbf{x}_0)$
z_0
$-\eta \cdot \nabla_z \mathcal{L}(\mathbf{z})$
$H_a = 0.675$
$H_a = 0.027$
$\Delta_\mathbf{x}$

Fig. 4.2 CLUE generation for an MNIST digit which presents high aleatoric entropy. Our BNN assigns large probabilities to classes 7 and 9. After optimisation, we decode a similar looking digit for which the aleatoric entropy is low. The new digit is confidently classified as a 7. The saliency map, placed to the right, highlights increases in pixel amplitude in red and decreases in blue. Although we represent a 2-dimensional toy latent space, the VAE used for this experiment produces 10-dimensional latent codes.

We make use of SGD optimisation with a Nesterov momentum coefficient of 0.5 (Sutskever et al., 2013). We run the optimiser for a minimum of 5 iterations and a maximum of 30 iterations with a step size of 0.1. If the decrease in $\mathscr{L}(\mathbf{z})$ is smaller than $\mathscr{L}(\mathbf{z}_0)/100$ for 2 consecutive iterations, we apply early stopping. We set λ_0 to a value of 5. Since the prior on $\mathbf{z}$ constrains VAE latent spaces to have approximately similar dimensions for all datasets and we normalise inputs to be zero mean and unit variance, we find that the proposed hyperparameter settings work well across tasks.

4.1.2 CLUE for Images

Saliency maps can be extracted using most image-based ML interpretability techniques. CLUE saliency maps can be obtained as the difference between the counterfactual and

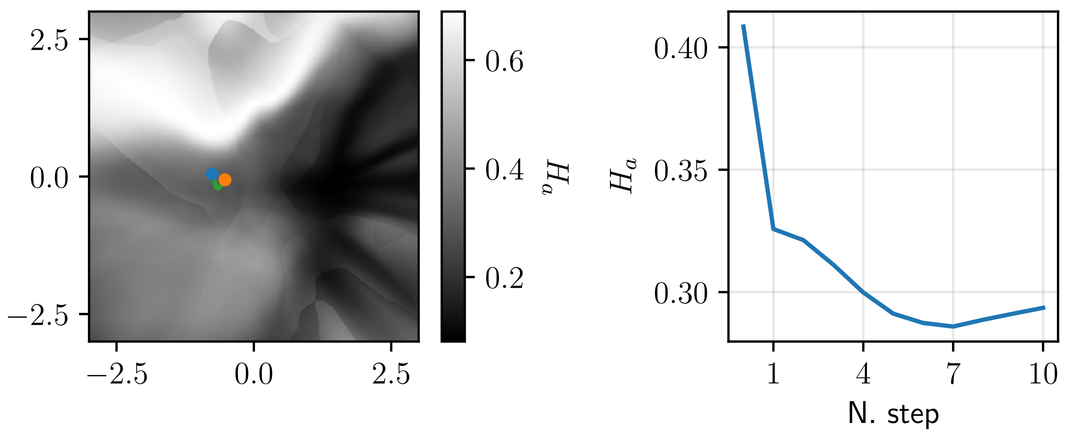

Fig. 4.3 Left: CLUE latent trajectory for a test point from the Credit dataset in a two-dimensional latent space. The blue dot marks the start of the trajectory and the orange one marks the end. Right: Changes in aleatoric entropy for inputs regenerated from latent codes along the trajectory.

original samples: $\Delta\mathbf{x} = (\mathbf{x}_{\text{CLUE}} - \mathbf{x}_0)$. However, unlike regular saliency maps, CLUE's do not just tell us "where the model is looking." They have a tangible meaning. Salient regions are the ones that would have to be changed to decrease uncertainty. The counterfactual explanation gives information about how these regions would have to be changed. We illustrate this with an example in fig. 4.2.

4.1.3 CLUE for Tabular Data

As discussed in section 3.1.4, a barrier to ML interpretability methods that use input features as cognitive chunks is that input spaces can be high dimensional. CLUE explanations often consist of large changes to a small subset of dimensions and small, almost indistinguishable, changes to the rest. We hypothesise that this is caused by the limited expressivity of our VAE, preventing the input features of interest from changing independently from the rest.

To remedy this, we suggest using expressive VAEs with large latent spaces and only displaying changes to variables which are large enough to be significant. We use the change in the percentile of each variable with respect to the training set distribution as a measure of relevance. We only display variables for which the CLUE explanation is separated 15 percentile points or more from the original input. We demonstrate this approach on a sample from the UCI default of credit card clients dataset (Dua and Graff, 2017). This dataset is composed of 23 input variables, of which 3 are categorical and the rest continuous. Figure 4.3 shows the corresponding CLUE optimisation trajectory in a 2-dimensional latent space. The

Table 4.1 Human-readable CLUE explanation for the same test point from the Credit dataset as shown in fig. 4.3. Only significant variable changes (percentile change > 15) are displayed.

	Current Value (Percentile)		Resulting Value (Percentile)
Probability of Default	0.14 (46.37)	$\rightarrow$	0.06 (16.84)
Feature	Current Value (Percentile)		Required Value (Percentile)
Gender	M (-)	$\rightarrow$	F (-)
Payment 4 Delay (Months)	1.00 (62.12)	$\rightarrow$	0.92 (35.81)
Payment 5 Delay (Months)	1.00 (62.12)	$\rightarrow$	0.82 (34.89)
Payment 6 Delay (Months)	1.00 (62.12)	$\rightarrow$	0.77 (34.62)
Payment 2 ($)	1065 (30.23)	$\rightarrow$	2666 (58.27)

result of this procedure is a human-readable CLUE explanation, presented in table 4.1. The predictor's confidence would have been 30 percentile points larger had previous payments been delayed slightly less and the payment from two months ago been about $1500 larger. It is worth noting that the described approach is somewhat tentative and that CLUE would likely benefit from more elaborate forms of statistical significance testing.

4.2 Summarising by Clustering CLUEs

In this section, we propose an approach for clustering similar CLUE explanations, allowing us to identify trends in the sources of uncertainty of tabular datasets. We adopt the variational mixture of Gaussians clustering algorithm, presented in section 2.4, as it provides a straight-forward way to select the number of mixture components automatically. We use $\Delta\mathbf{x}{=}(\mathbf{x}_{\text{CLUE}} - \mathbf{x}_0)$ as the features to be clustered. Intuitively, this results in points for which the provided explanations are similar being grouped.

We initialise the GMM with a large number of clusters, usually $K{=}40$, and set the prior over mixing proportions to be $\alpha_0 = K^{-1} = 0.025$. We discard all clusters for which $\mathbb{E}_{q(\boldsymbol{\pi})}[\pi_k] < 2\alpha_0$. A description for each of the detected clusters is produced based on the predictions made for their points, their uncertainty, and their input feature value range. We then find relevant input variables for each cluster using the same percentile-based approach described in section 4.1.3. This allows us to produce a human-readable summary of the changes necessary to reduce the uncertainty of the points in the cluster. We accompany this summary with box plots comparing the distribution of relevant features across the entire dataset with their distribution for points within the cluster and the cluster's counterfactual explanations. In section 5.5.2, we display excerpts drawn from a summary generated with this method.

Chapter 5

Experimental Validation

In this chapter, we provide key implementation details for CLUE, we demonstrate its application to datasets of different characteristics, and we evaluate its performance relative to baseline uncertainty interpretability methods.

In section 5.1, we introduce the datasets that we use in our experiments and provide our hyperparameter settings for each dataset. We also introduce two baseline methods to which we compare CLUE. In section 5.2, we compare approximate inference methods for BNNs in order to find which is the most appropriate for use with CLUE. In section 5.3, we propose a quantitative approach to evaluating the fidelity of uncertainty interpretability methods. We compare CLUE to the proposed baselines using this method. In section 5.4, we perform a qualitative comparison between CLUE image explanations and the ones provided by our baselines for the MNIST dataset. Finally, in section 5.5, we provide human-readable examples of CLUE single point explanations and summaries for tabular data.

5.1 Experimental Setup

In all of our experiments, we use CLUE with the hyperparameters described in section 4.1. We provide the model architectures used in our experiments, which are dataset dependent, in section 5.1.1. Our baseline methods and their hyperparameters are discussed in section 5.1.2. We implement all models and experiments in Python using the Pytorch[1] open-source deep learning framework.

[1]https://pytorch.org

Table 5.1 Characteristics of the datasets under consideration.

Name	Targets	Input Type	N. Inputs	N. Train	N. Test
Boston	Continuous	Continuous	13	455	51
Wine	Continuous	Continuous	11	4408	490
Credit	Binary	Continuous & Categorical	24	27000	3000
MNIST	Categorical	Image (greyscale)	28×28	60000	10000

5.1.1 Datasets and BNN Architectures

We consider four datasets for our experiments: MNIST handwritten digits (LeCun and Cortes, 2010), Wine quality red (aka Wine), Boston housing (aka Boston) and Default of credit card clients (aka Credit). The last three belong to the UCI dataset repository (Dua and Graff, 2017). The characteristics of these datasets are displayed in table 5.1.

We model continuous targets with Gaussian distributions. We use BNNs to parametrise their means and standard deviations. Discrete targets are modelled with categorical distributions, which are also parametrised by BNNs. When building DGMs, we model continuous inputs with diagonal, unit variance Gaussian distributions. This choice makes these models weigh all input dimensions equally, a desirable trait for explanation generation. We place categorical distributions over discrete inputs, expressing them as one-hot vectors. For the Credit dataset, where there are both continuous and discrete features, data likelihood values are obtained as the product of Gaussian likelihoods and categorical likelihoods. During the CLUE optimisation procedure, we approximate gradients through one-hot vectors with the gradients through softmax functions. This is known as the softmax straight-through estimator (Bengio, 2013). It is biased but works well in practise. For MNIST, we model pixels as the probabilities of a Bernoulli distribution. We feed these probabilities directly into our BNNs and DGMs. We use 10-dimensional latent spaces for MNIST and 4-dimensional latent spaces with UCI datasets.

We normalise all continuously distributed features such that they have 0 mean and unit variance. This facilitates model training and also ensures that all features are weighed equally in CLUE's L1 regularisation term (4.1). For MNIST, this normalisation is applied to whole images instead of individual pixels. Categorical variables are not normalised. Changing a categorical variable implies changing two bits in the corresponding one-hot vector. This creates the same L1 regularisation penalty as shifting a continuously distributed variable two standard deviations.

Table 5.2 Parameters for the NN architectures used with all datasets under consideration.

Dataset	BNN Depth	BNN Width	VAE Depth	VAE Width	VAEAC Depth	VAEAC Width
MNIST	2	1200	3	600	3	600
Boston	2	50	2	200	2	250
Wine	2	50	2	300	2	350
Credit	2	100	3	300	3	350

All of our models are built using fully connected layers. For the VAE and VAEAC models, we use skip-connections and batch normalisation at every layer. Network architecture parameters are given in table 5.2.

5.1.2 Uncertainty Interpretability Baselines

Due to a lack of work in interpreting uncertainty estimates, there are no methods to which CLUE is directly comparable. For this reason, we adapt two existing ML interpretability methods to generate counterfactual uncertainty explanations. These provide us with baselines with which to compare CLUE.

Local Sensitivity Analysis

We create a local analogue of uncertainty sensitivity analysis (Depeweg et al., 2017) by applying it to a single datapoint. A counterfactual example that reduces our BNN's uncertainty can be generated by simply taking a step of size δ in the direction of the gradient:

$$\mathbf{x} = \mathbf{x}_0 - \delta \cdot \frac{\nabla_{\mathbf{x}_0} H(\mathbf{y}|\mathbf{x}_0)}{||\nabla_{\mathbf{x}_0} H(\mathbf{y}|\mathbf{x}_0)||_1} \tag{5.1}$$

We set $\delta = \delta_0/n(\mathbf{x})$. As a result, the choice of δ_0 lets us select the mean per-dimension step size. When performing experiments, we choose δ_0 by using a grid search to find the value that minimises the BNNs uncertainty. This method can be understood as a single-step version of CLUE without a generative model.

U-FIDO

As our second baseline, we adapt FIDO (Chang et al., 2019), a state-of-the-art counterfactual interpretability method presented in section 3.1.5, to explain uncertainty (U-FIDO). We do this by replacing the classification score $s_I(\cdot)$ with an uncertainty metric $H(\cdot)$. U-FIDO finds the masking parameters $\boldsymbol{\rho}$ which minimise:

$$\mathscr{L}(\boldsymbol{\rho}) = \mathbb{E}_{p_{\boldsymbol{\rho}}(\mathbf{z})}[H(\mathbf{y}|\phi(\mathbf{x},\mathbf{z})) + \lambda \|\mathbf{z}\|_1] \tag{5.2}$$

where all undefined terms denote the same objects as in section 3.1.5. We re-frame the regularisation parameter as $\lambda = \lambda_0/n(\mathbf{x})$ and choose $\lambda_0 = 0.5$. A VAEAC is used as U-FIDO's conditional generative model in our experiments. We optimise U-FIDO masks for 30 iterations, approximating the expectation in (5.2) with 20 MC samples per iteration.

5.2 Choosing an Approximate Inference Method

The results presented in this section for all approximate inference methods on the MNIST dataset, except SG-HMC, were obtained as a part of the MLMI4 module's coursework.

We are interested in using an approximate inference technique that allows our BNNs to both fit the data well and produce reliable model uncertainty estimates. This will ensure that our uncertainty interpretability experiments faithfully represent the methods being evaluated and are not biased by weakly performing predictive models or unreliable uncertainty estimates. We compare four approximate inference approaches, all of which were introduced in section 2.1: Bayes By Backprop with the local reparametrisation trick (SGVI), MC dropout, preconditioned stochastic gradient Langevin dynamics and stochastic gradient HMC.

As shown in table 5.3 and table 5.4, SG-HMC performs best on all UCI datasets in terms of rms and classification error. It is a close second to MC dropout on MNIST. In order to test each approximate inference method's capacity to capture model uncertainty, we train BNNs

Table 5.3 Mean and standard deviation for test log likelihood values obtained with each approximate inference method on the datasets under consideration. Each experiment is run 10 times. For the UCI datasets, a random 10% of the data is selected for testing in each run. For MNIST, the default splits are used.

Dataset	MNIST loglike	Boston loglike	Wine loglike	Credit loglike
SGVI	-0.089 ± 0.011	-2.650 ± 0.074	-0.921 ± 0.071	-0.439 ± 0.005
MC Dropout	$\mathbf{-0.044 \pm 0.002}$	-2.471 ± 0.129	$\mathbf{-0.909 \pm 0.078}$	-0.428 ± 0.007
pSGLD	-0.066 ± 0.008	-2.585 ± 0.117	-0.917 ± 0.072	-0.433 ± 0.007
SG-HMC	-0.057 ± 0.007	$\mathbf{-2.359 \pm 0.177}$	-0.921 ± 0.065	$\mathbf{-0.427 \pm 0.007}$

Table 5.4 Mean and standard deviation for test rms and error % values obtained with each approximate inference method on the datasets under consideration. Each experiment is run 10 times. For the UCI datasets, a random 10% of the data is selected for testing in each run. For MNIST, the default splits are used.

Dataset	MNIST err %	Boston rms	Wine rms	Credit err %
SGVI	2.082 ± 0.015	3.371 ± 0.709	0.618 ± 0.048	0.180 ± 0.004
MC Dropout	$\mathbf{1.377 \pm 0.005}$	3.060 ± 0.699	0.616 ± 0.047	0.178 ± 0.003
pSGLD	1.761 ± 0.010	3.612 ± 0.807	0.618 ± 0.046	0.179 ± 0.004
SG-HMC	1.480 ± 0.008	$\mathbf{3.042 \pm 0.649}$	$\mathbf{0.613 \pm 0.051}$	$\mathbf{0.177 \pm 0.004}$

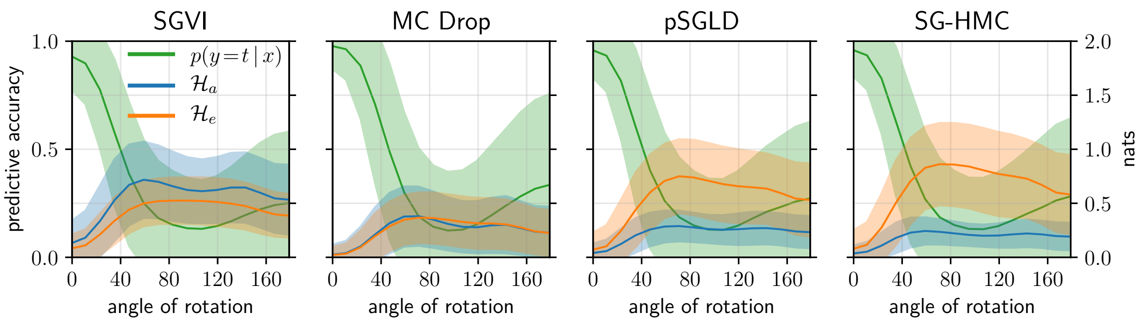

Fig. 5.1 Predictive accuracy, aleatoric and epistemic entropy values obtained for progressive rotations of MNIST digits. The error bars correspond to the standard deviations obtained over the whole testset.

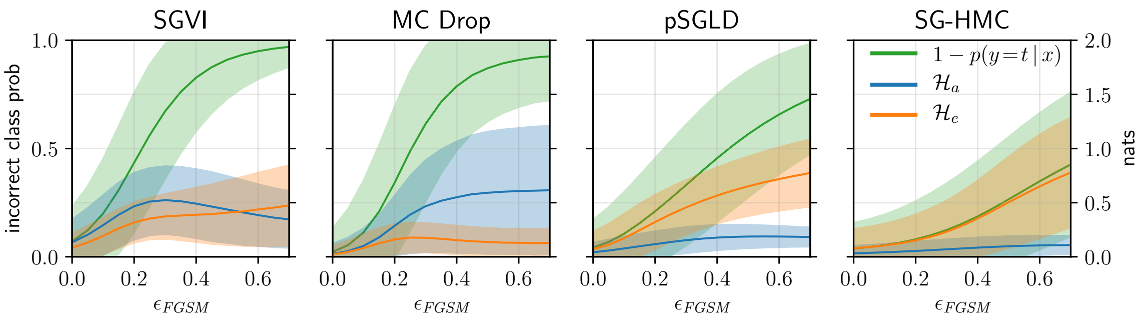

Fig. 5.2 Error rate, aleatoric and epistemic entropy values obtained when applying progressively larger adversarial perturbations to MNIST images. The error bars correspond to the standard deviations obtained over the whole testset.

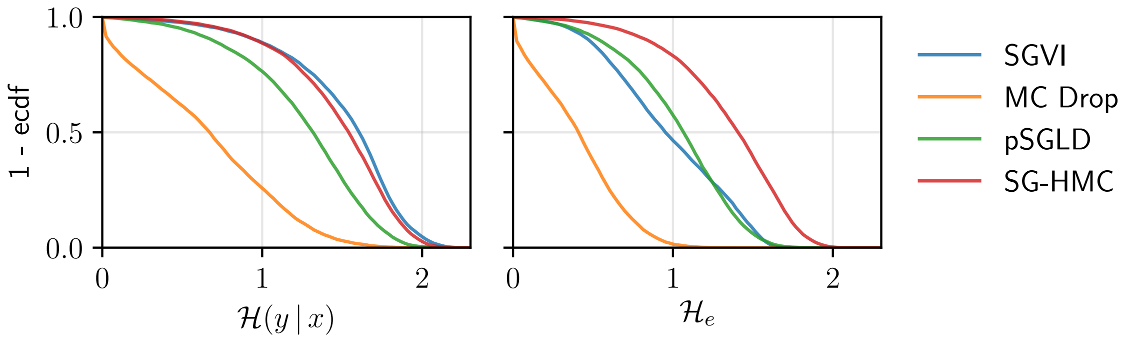

Fig. 5.3 Expected cumulative density functions for predictive entropy (left) and epistemic entropy (right) obtained on the KMNIST test set with models trained on MNIST.

on MNIST and evaluate them on OOD samples: rotated MNIST digits, Japanese letters from the KMNIST dataset (Clanuwat et al., 2018) and adversarial examples generated with the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015). In all of these scenarios, we would like the epistemic entropy to be both large and the dominant source of uncertainty. In figs. 5.1 and 5.2, we observe that this is most true for SG-HMC. Interestingly, this method provides some robustness to adversarial perturbations. SG-HMC also leads to our BNN displaying the most model uncertainty when evaluated on samples from a different dataset than it was trained on, as shown in fig. 5.3.

We hypothesise that the success of SG-HMC stems from it not being limited to the crude localised approximations to the posterior employed by MC-Dropout and SGVI. The use of momentum allows SG-HMC to avoid random-walk behaviour and explore parameter-space more efficiently than pSGLD. We employ this approximate inference method for the rest of the experiments in this thesis.

We make use of scale adapted SG-HMC, an approach to automatic hyperparameter discovery proposed by Springenberg et al. (2016). This technique estimates the mass matrix and the noise introduced by stochasticity in the gradients using exponentially decaying moving average filters during the burn-in phase. We use a fixed step size of $\varepsilon = 0.01$ and batch sizes of 512. We set a diagonal 0 mean Gaussian prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \sigma_w^2 \cdot I)$ over each layer of weights. We place a conjugate Gamma hyperprior over σ_w^2 with parameters $\alpha = \beta = 10$. We periodically update σ_w^2 for each layer using Gibbs sampling.

On MNIST, we burn in our chain for 30 epochs, using the first 15 to estimate SG-HMC parameters. We re-sample momentum parameters every 10 steps and perform a Gibbs sweep over the prior variances every 50 steps. We save parameter settings every 4 epochs until a total of 100 sets of weights are stored. For UCI experiments, we perform a burn-in of 200

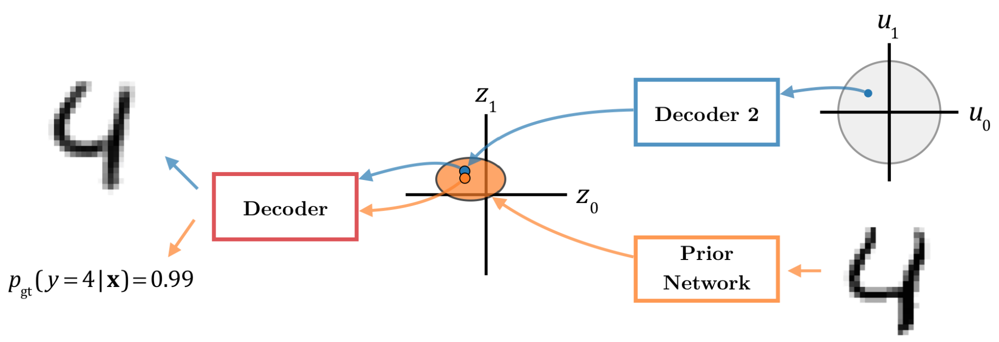

Fig. 5.4 A two-level VAEAC can be used as a ground-truth generative model and a classifier simultaneously. The blue path illustrates how the model is used for data generation and likelihood estimation. The orange path shows how it is used for obtaining a distribution over targets, given a set of input features.

epochs, using the first 100 to estimate SG-HMC parameters. We save weight configurations every 10 epochs. All other parameters are kept the same as for MNIST.

5.3 Quantitative Comparison of Uncertainty Interpretability Methods

In this section, we introduce a novel approach for the evaluation of counterfactual uncertainty explanations. We use this metric to compare CLUE with our baseline methods.

5.3.1 Functionally-Grounded Evaluation of Uncertainty Interpretability Methods

We would like for a way to measure the fidelity of our counterfactual uncertainty interpretability methods without having to resort to expensive human-based tests. This metric should measure the capacity of a method to give meaningful explanations: counterfactuals should be inputs for which our BNN is less uncertain but do not exploit weaknesses to adversarial perturbations in our network. Explanations must also represent plausible parameter settings, laying close to the manifold of the data.

We propose the use of a two-level VAEAC model, introduced in section 2.3.3, as a ground-truth data generator. We refer to this model's decoder as p_{d} and to its prior network as p_{p}. We train ground-truth VAEACs to capture the joint distributions of inputs and targets of the datasets presented in section 5.1.1. We then generate high-quality artificial datasets, which we denote as $\bar{\mathscr{D}}$, through ancestral sampling from the ground-truth VAEACs' inner latent spaces:

$$\bar{\mathscr{D}} = \{\bar{\mathbf{x}}_n, \bar{\mathbf{y}}_n\}_{n=1}^{N}; \quad (\bar{\mathbf{x}}_n, \bar{\mathbf{y}}_n) \sim p_{\mathrm{d}}(\mathbf{x}, \mathbf{y}|\mathbf{u}_n); \quad \mathbf{u}_n \sim \mathscr{N}(\mathbf{u}; \mathbf{0}, I) \tag{5.3}$$

We generate both artificial train and test sets. We train our BNNs and auxiliary DGMs using the artificial train sets and use our uncertainty interpretability methods to generate explanations for the artificial test set points.

Since a ground-truth VAEAC is the true generative process of our artificial data, we can obtain the true distribution over targets for any given input using the VAEAC's conditioning mechanism:

$$p_{\mathrm{gt}}(\bar{\mathbf{y}}|\bar{\mathbf{x}}) = \mathbb{E}_{p_{\mathrm{p}}(\mathbf{z}|\bar{\mathbf{x}})}[p_{\mathrm{d}}(\bar{\mathbf{y}}|\mathbf{z})] \tag{5.4}$$

Ground-truth entropy and standard deviation estimates can then be obtained from this distribution using (2.18) and (2.21) respectively. They represent the aleatoric uncertainty values that would be given by an idealised BNN. Given a point from the artificially generated test set $\bar{\mathbf{x}}_0$ and its corresponding counterfactual uncertainty explanation $\bar{\mathbf{x}}_c$, we can evaluate how much aleatoric entropy is explained by $\bar{\mathbf{x}}_c$ as:

$$\Delta_{H_{\mathrm{gt}}} = H_{\mathrm{gt}}(\bar{\mathbf{y}}|\bar{\mathbf{x}}_0) - H_{\mathrm{gt}}(\bar{\mathbf{y}}|\bar{\mathbf{x}}_c) \tag{5.5}$$

An analogous procedure is used when working with standard deviations. How well epistemic entropy is explained by $\mathbf{x}_c$ can be evaluated through the decrease in the difference between our BNN's predictions and the ground-truth model's predictions. For classification tasks, we measure this difference across whole test sets in terms of classification error:

$$\begin{aligned} \Delta_{err_{\mathrm{gt}}} &= err_{\mathrm{gt}}(\bar{\mathbf{X}}_0) - err_{\mathrm{gt}}(\bar{\mathbf{X}}_c); \\ err_{\mathrm{gt}}(\mathbf{X}) &= \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(\mathbb{E}_{p(\mathbf{y}_n|\mathbf{x}_n,\bar{\mathscr{D}})}[\mathbf{y}_n] = \mathbb{E}_{p_{\mathrm{gt}}(\mathbf{y}_n|\mathbf{x}_n)}[\mathbf{y}_n]) \end{aligned} \tag{5.6}$$

For regression, we use root mean square (rms) error:

$$\begin{aligned} \Delta_{rms_{\mathrm{gt}}} &= rms_{\mathrm{gt}}(\bar{\mathbf{X}}_0) - rms_{\mathrm{gt}}(\bar{\mathbf{X}}_c); \\ rms_{\mathrm{gt}}(\mathbf{X}) &= \sqrt{\frac{1}{N}\sum_{n=1}^{N} (\mathbb{E}_{p(\mathbf{y}_n|\mathbf{x}_n,\bar{\mathscr{D}})}[\mathbf{y}_n] - \mathbb{E}_{p_{\mathrm{gt}}(\mathbf{y}_n|\mathbf{x}_n)}[\mathbf{y}_n])^2} \end{aligned} \tag{5.7}$$

Aleatoric and epistemic uncertainty decreases are not enough to validate the usefulness of explanations. These also need to be similar to the datapoints for which we wish to explain the uncertainty. For this reason, we pair the described metrics with the per input-dimension mean difference between datapoints and their explanations: $|\Delta_x| = {\|\mathbf{x}_0-\mathbf{x}_c\|_1}/{n(\mathbf{x}_0)}$. We propose the use of ${\Delta_{H_{gt}}}/{|\Delta_x|}$ and ${\Delta_{\sigma_{gt}}}/{|\Delta_x|}$ as figures of merit for aleatoric uncertainty interpretability explanations. Analogously, ${\Delta_{err_{gt}}}/{|\Delta_x|}$ and ${\Delta_{rms_{gt}}}/{|\Delta_x|}$ can be used for epistemic uncertainty. Although we acknowledge that comparing methods with fixed values of $|\Delta_x|$ would provide a more informative analysis, U-FIDO and CLUE provide no easy way to control this variable. This is especially true in the case of CLUE, where the VAE's reconstruction error sets a lower bound on $|\Delta_x|$.

Finally, we can measure how close $\mathbf{x}_c$ lies to the manifold of the data by evaluating its log-likelihood under the ground-truth VAEAC: $\log p_{\text{d}}(\mathbf{x}_c)$. This value is approximated using (2.28) and dropping the likelihood contributions corresponding to target variables.

5.3.2 CLUE's L1 Regularisation Parameter

Using aleatoric uncertainty estimates from the artificial MNIST and Wine datasets, we investigate the effects of CLUE's regularisation parameter λ_0, showing our results in fig. 5.5. As expected, a larger λ_0 results in explanations which are closer to the original inputs. However, this effect yields diminishing returns for larger regularisation parameters. The differences between the maximum values of $|\Delta_x|$ and its lower asymptotes are not very large. We find that there is a minimum value of $|\Delta_x|$ which is determined by the reconstruction error of the VAE. In this way, the performance of CLUE is strongly dependent on the expressivity of its generative model.

The uncertainty assigned by the ground-truth VAEAC to our explanations grows linearly with the choice of λ_0 for MNIST. As a result of there being a large number of low uncertainty samples in each dataset, the mean value for this parameter lies close to the lower 5th percentile. The top errorbar, which denotes the 95th percentile, is more representative of our method's performance on uncertain samples. For wine, the ground-truth model's conditional standard deviation presents very small fluctuations with the choice of λ_0.

The likelihoods of the explanations provided by CLUE are also relatively unaffected by the regularisation parameter. Our experiments show that CLUE is insensitive to the choice of λ_0. This behaviour can be seen as an advantage because it simplifies hyperparameter selection. However, it comes at the cost of making the method less flexible.

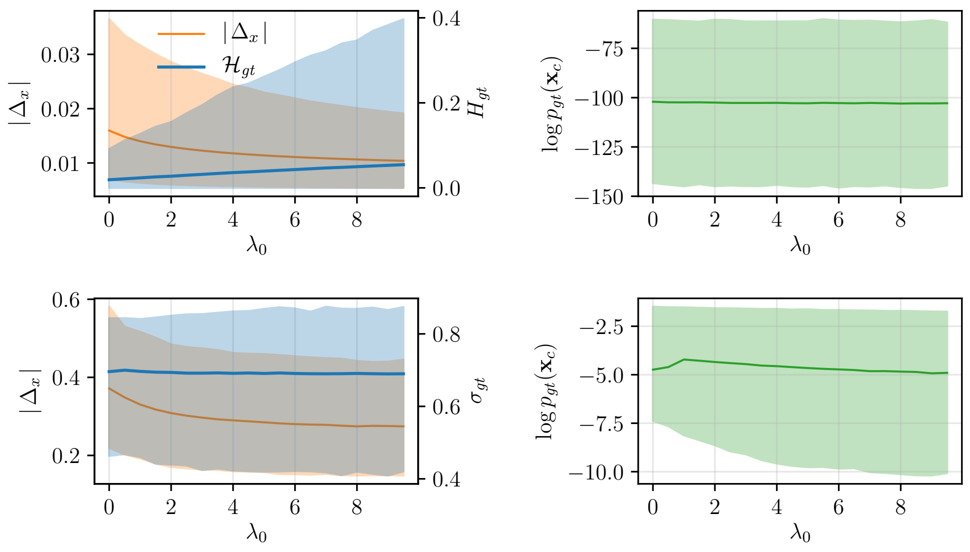

Fig. 5.5 The plots on the left show the evolution of the ground-truth generative models' uncertainty and $|\Delta_x|$ with the regularisation parameter λ_0. The right side plots show the change in the explanations' likelihood under the ground-truth models with λ_0. The top row shows the results obtained on artificial MNIST and the the bottom row refers to artificial Wine. Solid lines represent the mean values computed across all samples in the testsets. The errorbars denote 5th and 95th percentiles.

5.3.3 Evaluating Uncertainty Explanations

We train ground-truth VAEAC models on each of the datasets presented in section 5.1.1. We generate the same number of artificial train samples as train samples there are in the original train sets. We generate 3000 artificial test points per dataset.

In tables 5.5 to 5.8, we show the results obtained when attempting to explain the aleatoric uncertainty in our artificial testsets. We provide mean and standard deviation values computed across all testpoints. We show results for both the unmodified artificial test points and uncertainty explanations. In all tables, the first two rows show the aleatoric uncertainty values produced by our BNNs and the ground-truth VAEACs. Methods that manage to decrease the former value with respect to the artificial test set but not the latter reduce uncertainty by exploiting flaws in our BNNs instead of providing useful explanations. The following two rows correspond to the feature variation per input dimension and the figure of merit. The last row contains the explanations' log-likelihoods under the ground-truth generative models, estimated with 10,000 samples.

We consider that explanations are not OOD as long as their log-likelihood is at least as large as that of the artificial testset's samples. Methods which provide in-distribution explanations are compared based on their figure of merit. It is worth noting that all of these values are strongly dataset dependent and serve only to compare interpretability methods, not to draw comparisons across datasets.

For all datasets except Credit, local sensitivity analysis decreases our BNNs' aleatoric uncertainty the most. However, only a small fraction of this decrease translates onto the ground-truth VAEAC. MNIST presents an exception to this rule, as the VAEAC's entropy is reduced by half. A possible explanation for this behaviour is that, in some circumstances, adversarial examples can transfer across models (Petrov and Hospedales, 2019). Local sensitivity obtains the worse figure of merit in all experiments. Furthermore, the likelihood values corresponding to this method's explanations are smaller than those of the artificial test points for all datasets. Local sensitivity produces OOD explanations which manage to "trick" BNNs but are of little value otherwise. Since local sensitivity can be seen as CLUE without the use of a generative model, the large performance difference between these two methods validates the utility of CLUE's DGM.

Both U-FIDO and CLUE produce in-distribution explanations for all datasets. U-FIDO is able to produce explanations which are closer to the original samples. Its masking mechanic favours strong performance on this metric, as it allows a large proportion of input features to stay the same. Nevertheless, U-FIDO is limited to optimising the parameters of a mask. It depends on the expectation over a generative model to fill-in the masked inputs. CLUE is more flexible, being able to search a latent space for the point which produces the least uncertain input feature configuration. Our proposed method explains away more uncertainty than U-FIDO but does so at the cost of deviating further from the original samples due to the VAE's imperfect reconstructions. CLUE obtains figures of merit that are two times larger than U-FIDO's on MNIST and Wine. Both methods perform similarly on Credit, although CLUE does slightly better. U-FIDO obtains a larger figure of merit on Boston.

We observe large standard deviations across all of our results. These are often similar to or larger than the mean values reported and are caused by the variability in uncertainty among our datasets' samples. For U-FIDO and CLUE, this variability is translated onto the values of $|\Delta_x|$. Points with small initial uncertainty values are assigned explanations which barely deviate from the original inputs. The opposite holds true for very uncertain datapoints. There is no variance in $|\Delta_x|$ for explanations obtained with local sensitivity as this parameter is fixed with the choice of δ_0.

In tables 5.9 to 5.12, we show the results obtained with epistemic uncertainty explanations. Here, the first row refers to the epistemic uncertainty values given by our BNNs. Since our VAEAC is the ground-truth generative process of the data, it can not express epistemic uncertainty. Instead, we use rms and classification error as proxies, providing values for these in the second column. Since these metrics are computed with whole datasets, as opposed to sample-wise, we do not provide standard deviation values for them. The rest of the values provided are analogous to those from our aleatoric uncertainty experiments.

The results obtained in this second set of experiments are similar to those obtained with aleatoric uncertainty. Local sensitivity produces OOD explanations which do not explain the ground-truth VAEACs' uncertainty. This is especially true for the Wine dataset, where local sensitivity causes the rms error to increase. U-FIDO and CLUE produce plausible parameter configurations. CLUE outperforms FIDO on Wine and MNIST. Generally, CLUE is able to explain away more uncertainty and U-FIDO produces explanations closer to the original datapoints. MNIST and Boston are exceptions to this rule. On these datasets, CLUE and U-FIDO perform best on all metrics, respectively.

Table 5.5 Aleatoric uncertainty explanation experiment results obtained on the Boston dataset. We provide mean and standard deviation values computed across all testpoints.

Boston, Aleatoric	Artificial Test Set	Local Sensitivity	U-FIDO	CLUE
$\sigma_{a,\text{BNN}}$	0.437 ± 0.245	0.253 ± 0.121	0.314 ± 0.083	0.323 ± 0.087
σ_{gt}	0.374 ± 0.142	0.356 ± 0.105	0.333 ± 0.079	0.336 ± 0.080
$\lvert\Delta_x\rvert$	-	0.500 ± 0.000	0.181 ± 0.387	0.290 ± 0.320
$\frac{\Delta_{\sigma_{\text{gt}}}}{\lvert\Delta_x\rvert}$	-	0.053 ± 0.210	$\mathbf{0.274 \pm 0.569}$	0.171 ± 0.309
$\log p_{\text{d}}(\bar{\mathbf{x}}_c)$	-275.93 ± 868.80	-1036.71 ± 1389.33	-206.74 ± 1034.22	-135.69 ± 842.13

Table 5.6 Aleatoric uncertainty explanation experiment results obtained on the Wine dataset. We provide mean and standard deviation values computed across all testpoints.

Wine, Aleatoric	Artificial Test Set	Local Sensitivity	U-FIDO	CLUE
$\sigma_{a,\text{BNN}}$	0.734 ± 0.246	0.626 ± 0.255	0.669 ± 0.224	0.658 ± 0.220
σ_{gt}	0.694 ± 0.182	0.682 ± 0.214	0.691 ± 0.183	0.662 ± 0.154
$\lvert\Delta_x\rvert$	-	0.250 ± 0.000	0.197 ± 0.344	0.450 ± 0.297
$\frac{\Delta_{\sigma_{\text{gt}}}}{\lvert\Delta_x\rvert}$	-	0.045 ± 0.856	0.034 ± 0.512	$\mathbf{0.070 \pm 0.308}$
$\log p_{\text{d}}(\bar{\mathbf{x}}_c)$	-8.33 ± 3.79	-14.29 ± 4.64	-6.45 ± 2.88	-4.01 ± 2.65

Table 5.7 Aleatoric uncertainty explanation experiment results obtained on the MNIST dataset. We provide mean and standard deviation values computed across all testpoints. $\log p_{\text{d}}(\mathbf{x})$ values are obtained on dynamically binarised images.

MNIST, Aleatoric	Artificial Test Set	Local Sensitivity	U-FIDO	CLUE
$H_{a,\text{BNN}}$	0.963 ± 0.430	0.744 ± 0.402	0.924 ± 0.415	0.873 ± 0.374
H_{gt}	0.119 ± 0.256	0.062 ± 0.179	0.090 ± 0.225	0.020 ± 0.096
$\lvert\Delta_x\rvert$	-	0.050 ± 0.000	0.008 ± 0.016	0.016 ± 0.010
$\frac{\Delta_{H_{\text{gt}}}}{\lvert\Delta_x\rvert}$	-	1.152 ± 3.581	3.058 ± 2.738	$\mathbf{6.260 \pm 3.034}$
$\log p_{\text{d}}(\bar{\mathbf{x}}_c)$	-97.26 ± 25.36	-286.84 ± 63.08	-98.59 ± 25.73	-102.45 ± 25.79

Table 5.8 Aleatoric uncertainty explanation experiment results obtained on the Credit dataset. We provide mean and standard deviation values computed across all testpoints.

Credit, Aleatoric	Artificial Test Set	Local Sensitivity	U-FIDO	CLUE
$H_{a,\text{BNN}}$	0.350 ± 0.140	0.236 ± 0.093	0.237 ± 0.099	0.222 ± 0.062
H_{gt}	0.475 ± 0.168	0.459 ± 0.183	0.381 ± 0.183	0.293 ± 0.160
$\lvert\Delta_x\rvert$	-	0.300 ± 0.000	0.387 ± 0.642	0.674 ± 0.679
$\frac{\Delta_{H_{\text{gt}}}}{\lvert\Delta_x\rvert}$	-	0.053 ± 0.803	0.246 ± 0.610	$\mathbf{0.270 \pm 0.327}$
$\log p_{\text{d}}(\bar{\mathbf{x}}_c)$	-57.92 ± 74.31	-69.84 ± 75.46	-39.94 ± 64.71	-23.25 ± 24.86

Table 5.9 Epistemic uncertainty explanation experiment results obtained on the Boston dataset. We provide mean and standard deviation values computed across all testpoints.

Boston, Epistemic	Artificial Test Set	Local Sensitivity	U-FIDO	CLUE
$\sigma_{e,\text{BNN}}$	0.166 ± 0.086	0.160 ± 0.083	0.123 ± 0.057	0.093 ± 0.049
rms_{gt}	0.229	0.228	0.161	0.187
$\lvert\Delta_x\rvert$	-	0.050 ± 0.000	0.122 ± 0.370	0.281 ± 0.315
$\frac{\Delta_{rms_{\text{gt}}}}{\lvert\Delta_x\rvert}$	-	0.019	**0.522**	0.173
$\log p_{\text{d}}(\bar{\mathbf{x}}_c)$	-275.93 ± 868.80	-345.15 ± 1801.16	-240.23 ± 1003.92	-205.16 ± 878.46

Table 5.10 Epistemic uncertainty explanation experiment results obtained on the Wine dataset. We provide mean and standard deviation values computed across all testpoints.

Wine, Epistemic	Artificial Test Set	Local Sensitivity	U-FIDO	CLUE
$\sigma_{e,\text{BNN}}$	0.247 ± 0.105	0.241 ± 0.102	0.190 ± 0.081	0.148 ± 0.070
rms_{gt}	0.223	0.227	0.164	0.117
$\lvert\Delta_x\rvert$	-	0.100 ± 0.000	0.236 ± 0.367	0.369 ± 0.307
$\frac{\Delta_{rms_{\text{gt}}}}{\lvert\Delta_x\rvert}$	-	-0.047	0.247	**0.322**
$\log p_{\text{d}}(\bar{\mathbf{x}}_c)$	-8.33 ± 3.79	-9.01 ± 4.04	-5.96 ± 2.95	-4.39 ± 2.07

Table 5.11 Epistemic uncertainty explanation experiment results obtained on the MNIST dataset. We provide mean and standard deviation values computed across all testpoints. $\log p_{\text{d}}(\mathbf{x})$ values are obtained on dynamically binarised images.

MNIST, Epistemic	Artificial Test Set	Local Sensitivity	U-FIDO	CLUE
$H_{e,\text{BNN}}$	0.135 ± 0.054	0.110 ± 0.059	0.129 ± 0.051	0.119 ± 0.044
err_{gt}	0.053	0.039	0.036	0.031
$\lvert\Delta_x\rvert$	-	0.100 ± 0.000	0.030 ± 0.021	0.016 ± 0.009
$\frac{\Delta_{err_{\text{gt}}}}{\lvert\Delta_x\rvert}$	-	0.135	0.482	**3.162**
$\log p_{\text{d}}(\bar{\mathbf{x}}_c)$	-97.26 ± 25.36	-486.77 ± 96.56	-98.73 ± 31.85	-102.567 ± 25.65

Table 5.12 Epistemic uncertainty explanation experiment results obtained on the Credit dataset. We provide mean and standard deviation values computed across all testpoints.

Credit, Epistemic	Artificial Test Set	Local Sensitivity	U-FIDO	CLUE
$H_{e,\text{BNN}}$	0.008 ± 0.007	0.008 ± 0.006	0.004 ± 0.004	0.004 ± 0.005
err_{gt}	0.417	0.394	0.180	0.176
$\lvert\Delta_x\rvert$	-	0.150 ± 0.000	0.376 ± 0.643	0.546 ± 0.586
$\frac{\Delta_{err_{\text{gt}}}}{\lvert\Delta_x\rvert}$	-	0.170	**0.631**	0.442
$\log p_{\text{d}}(\bar{\mathbf{x}}_c)$	-57.92 ± 74.31	-59.35 ± 73.84	-38.94 ± 60.57	-27.04 ± 30.56

Table 5.13 Time taken to run each uncertainty interpretability method for a batch of 2000 MNIST images on an NVIDIA K40 GPU. Mean and standard deviation values are obtained over 10 batches.

	Local Sensitivity	U-FIDO	CLUE
Time (s)	1.70 ± 0.01	529.63 ± 0.10	53.47 ± 0.33

5.3.4 Comparison of Computational Costs

Table 5.13 shows a comparison of the time taken to run each uncertainty interpretability method. Local sensitivity is the computationally cheapest method, as it does not involve optimisation. Both U-FIDO and CLUE use more costly iterative procedures. Having to use Monte Carlo to estimate gradients makes U-FIDO the most expensive approach. Thanks to our initialisation and early stopping strategies, both described in section 4.1, CLUEs are an order of magnitude cheaper to compute than U-FIDO explanations.

5.4 Qualitative Comparison of MNIST Explanations

In fig. 5.6, we compare explanations generated by CLUE and U-FIDO for four samples chosen from the top ten most uncertain MNIST test digits and four high uncertainty artificial samples generated by the ground-truth VAEAC from section 5.3.1. CLUE consistently produces good quality counterfactuals. Its saliency maps clearly indicate sections that should be removed, like extra vertical lines that have been added to digits 7 and 5 or blurry sections in digits 1 and 3. They also correctly identify sections that should be added, successfully closing the circles for all three 9s and the digit 0. U-FIDO produces readable counterfactuals for the first two digits despite converting the 9 into a 4. It struggles with the second pair of digits, introducing noise artefacts into its explanations. U-FIDO produces sensible counterfactuals for the artificially generated digits but learns noisy masks which do not always direct the user's attention to the regions of interest.

Compared to CLUE, U-FIDO lacks flexibility. When a generative model is conditioned on pixels from a digit that presents high uncertainty, it will likely also be uncertain about which inpainting configuration to propose. U-FIDO chooses the expected inpainting configuration which, in some cases, results in an amalgamation of possible explanations. This matches the results from table 5.7, where U-FIDO's MNIST explanations present over double the variability in their uncertainty than CLUE's. On the other hand, CLUE is able to find the single point in latent space that best explains the input, always generating clean outputs.

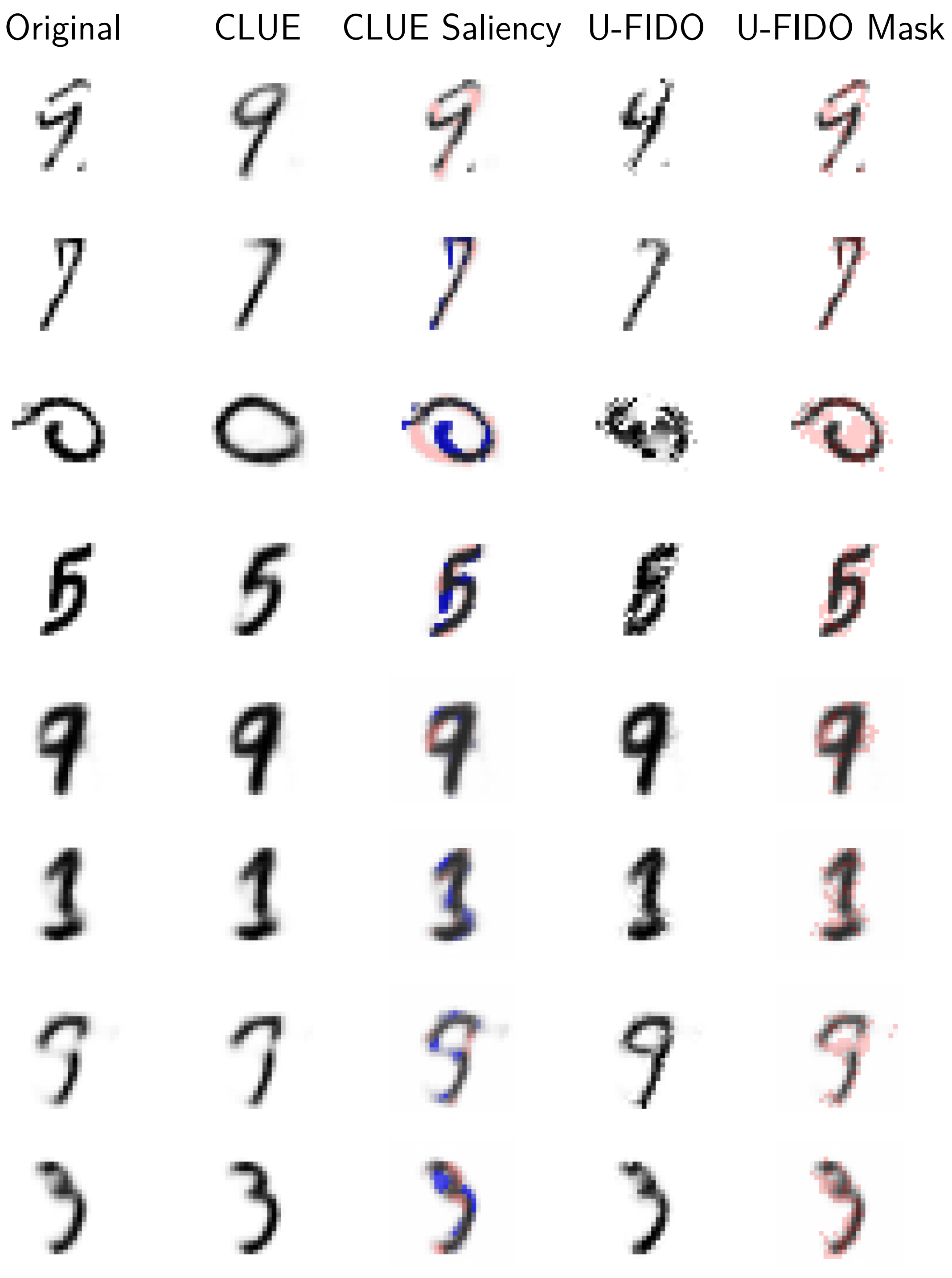

Fig. 5.6 The left column shows four of the most uncertain digits in the MNIST test set followed by four of the most uncertain digits generated by the ground-truth VAEAC from section 5.3.1. The following two columns show CLUE's explanations for these digits' predictive entropy and the corresponding saliency maps. Sections highlighted in blue mark removals and red sections denote additions. The final two columns show U-FIDO's explanations and masks. Pixels which are substituted by their expectation under a generative model are shown in red.

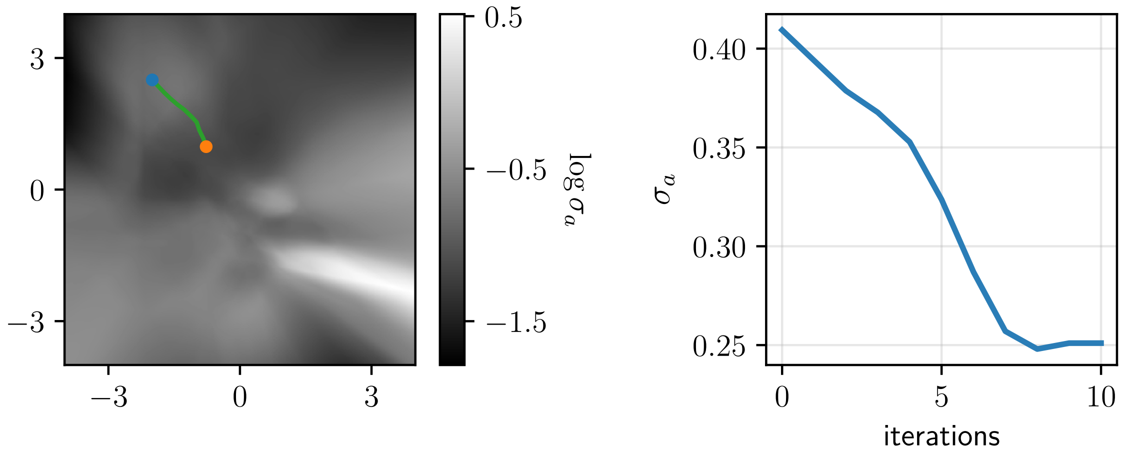

Fig. 5.7 Left: CLUE latent trajectory in a two-dimensional latent space for a test point from the Boston testset. The blue dot marks the start of the trajectory, and the orange one marks the end. Right: Changes in aleatoric std for inputs regenerated from latent codes along the trajectory.

5.5 Human-Readable Uncertainty Explanations for Tabular Data

In this section, we provide an example of a CLUE explanation for a test point from the Boston regression dataset and a set of excerpts from the CLUE summary generated for the Credit classification dataset.

5.5.1 An Explanation for an Individual Point from the Boston Dataset

We choose a testpoint from the Boston dataset to which our BNN has assigned high aleatoric uncertainty and use it to generate a CLUE explanation. Figures 5.7 and 5.8 show the corresponding latent-space and input-space trajectories. As discussed in section 4.1.3, CLUE benefits from the use of large latent spaces. Nevertheless, as Boston is our smallest and simplest dataset, CLUE's performance deteriorates least here when generating explanations from a 2-dimensional latent space. The only input variables which present a shift greater than 15 percentile points are the pupil-teacher ratio and the proportion of the population which is of low socioeconomic status. The resulting human-readable CLUE is displayed in table 5.14. The predicted target for the counterfactual explanation is only 2 percentile points separated from the original sample's prediction. However, the aleatoric standard deviation has been reduced by 75 percentile points.

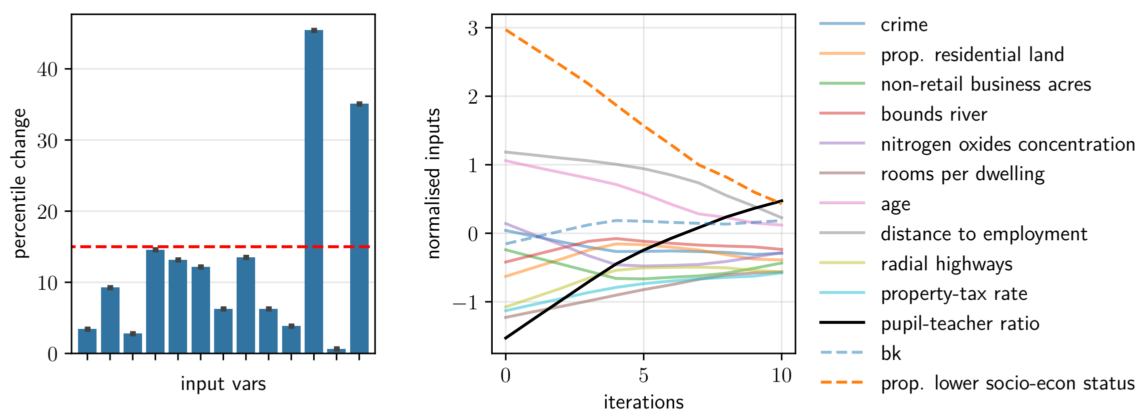

Fig. 5.8 Left: Absolute percentile difference between a sample from the Boston testset and its CLUE explanation for each input feature. The dashed red line marks the established threshold for a change to be considered significant: 15 points. Right: Normalised input feature configurations regenerated from latent codes placed along the CLUE trajectory. Variables which change significantly are highlighted.

Table 5.14 Human-readable CLUE explanation for the same test point from the Boston dataset as shown in fig. 5.8. Only significant variable changes (percentile shift > 15) are displayed.

	Current Value (Percentile)		Resulting Value (Percentile)
Predicted Price (in \$1000's)	15.05 (18.72)	$\rightarrow$	16.20 (20.79)
Aleatoric std. (in \$1000's)	3.74 (100.00)	$\rightarrow$	2.29 (24.33)
Feature	Current Value (Percentile)		Required Value (Percentile)
Pupil-teacher ratio	15.11 (11.89)	$\rightarrow$	19.48 (57.21)
Prop. lower economic status	33.62 (98.67)	$\rightarrow$	15.61 (63.04)

CLUE finds that a low pupil-teacher ratio combined with a large proportion of low socioeconomic status residents are the cause of the increased aleatoric uncertainty. A practitioner could pair these results with the knowledge that family sizes generally decline as their socioeconomic status grows (Dribe et al., 2014) and decide to add birth rates to the list of input variables.

5.5.2 A Summary for the Credit Dataset

We employ the clustering procedure described in section 4.2 to generate a summary of the main sources of aleatoric uncertainty for the Credit dataset. A VAE with a 4-dimensional latent space is used with CLUE. Our ARD strategy marks 20 mixture components as active out of 40. For brevity, in table 5.15, we only display an overview of the characteristics of the first 10 components, 8 of which are active.

Table 5.15 Number of points assigned to each cluster and aleatoric entropy decrease for the first 10/40 mixture components. Components 4 and 7 are not active.

N. Cluster	N. Points	H_a Decrease (Percentile decrease)
0	1407/27000	0.06 (11.38)
1	599/27000	0.08 (11.56)
2	507/27000	0.06 (13.58)
3	1228/27000	0.10 (24.67)
5	370/27000	0.15 (36.41)
6	2819/27000	0.04 (10.46)
8	23/27000	0.42 (77.95)
9	815/27000	0.24 (47.89)

Our summaries consist of clustered local explanations. For this reason, it is important to provide the range of input space in which each cluster's datapoints lie. This ensures that the end-user is not misled about the applicability of the summary explanations. Here, we omit this information for brevity, only displaying the variables which are responsible for the aleatoric uncertainty in each cluster together with their counterfactual counterparts. For each cluster, our human-readable explanation table is composed of the mean values, computed over the full cluster, of every continuous feature deemed significant. These are accompanied by boxplots which allow for the comparison of complete distributions.

Components 0 and 6 are responsible for sizeable amounts of points, but present small entropy decreases. These clusters contain low uncertainty samples, as shown in table 5.16. The input-space changes proposed for their points are similar to each other in that they are of low magnitude. The only feature that sees significant variation is the level of education of the clients.

For components 3, 5, and 9, a significant portion of the uncertainty can be explained by CLUE. As shown in table 5.17 and fig. 5.9, our BNN would have assigned a lower probability of default to individuals in cluster 3 had they been given slightly more credit and their previous payments been slightly larger. Points in cluster 9 are assigned a 60% probability of default on average. Table 5.18 and fig. 5.10 show that this large value can be explained by the presence of two atypical features; the 815 clients in this cluster delayed their second to last payment by 4 months and completely missed their last payment. Our method suggests that, had the distribution of these inputs for points in the cluster been more similar to their distribution over the whole dataset, the probability of default would have been reduced to 35%. The explanation for cluster 5's uncertainty is given in table 5.19 and fig. 5.11. Here, our BNN believes there is a 25% chance that users will default on their credit. Payments from 5

Table 5.16 CLUE explanation for cluster 0.

Cluster 0	Current Value (Percentile)		Resulting Value (Percentile)
Probability of Default	0.13 (28.93)	→	0.09 (17.55)
Feature	Current Value		Required Value
Education: undergrad	0.55%	→	99.57%

and 6 months ago were delayed by an average of 3 and 2 months respectively. The amounts paid in recent months display large fluctuations. Our model would have been more certain about the clients not defaulting had older payments been delayed less and the amounts paid in recent months been more consistent.

Although the most substantial uncertainty decrease is achieved for cluster 8, this component only has 23 points assigned to it. Explanations sourced from these points will likely not be very widely applicable.

Table 5.17 CLUE summary explanation for cluster 3.

Cluster 3	Current Value (Percentile)		Resulting Value (Percentile)
Probability of Default	0.18 (53.30)	→	0.13 (28.63)
Feature	Current Value (Percentile)		Required Value (Percentile)
Given credit ($)	117475 (43.56)	→	150609 (54.66)
Previous payment 1 ($)	2177 (50.63)	→	3790 (66.11)
Previous payment 2 ($)	2144 (51.78)	→	3718 (67.10)
Previous payment 3 ($)	1720 (49.03)	→	2436 (59.70)
Previous payment 6 ($)	1626 (51.99)	→	2780 (64.69)

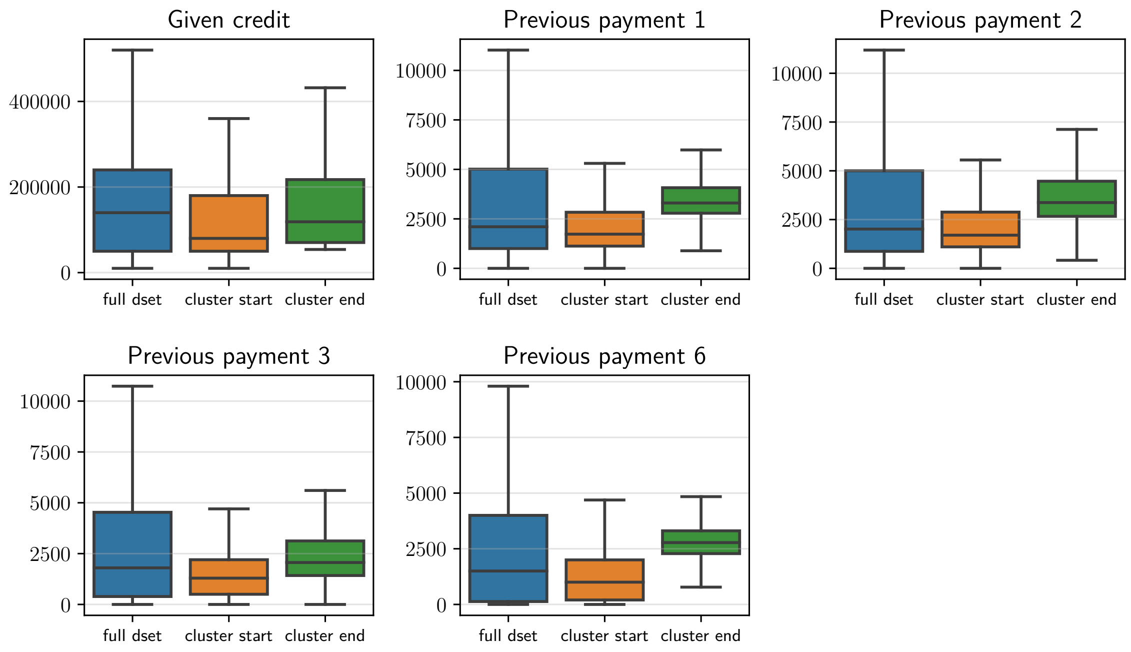

Fig. 5.9 CLUE summary boxplots for cluster 3.

Table 5.18 CLUE summary explanation for cluster 9.

Cluster 9	Current Value (Percentile)		Resulting Value (Percentile)
Probability of Default	0.60 (82.51)	→	0.35 (34.66)
Feature	**Current Value (Percentile)**		**Required Value (Percentile)**
Education: high school	46.01%	→	64.17%
Education: other	19.88%	→	4.91%
Payment delay 6 (months)	3.95 (85.30)	→	1.75 (32.76)
Previous payment 4 ($)	0 (0.00)	→	3724 (65.86)

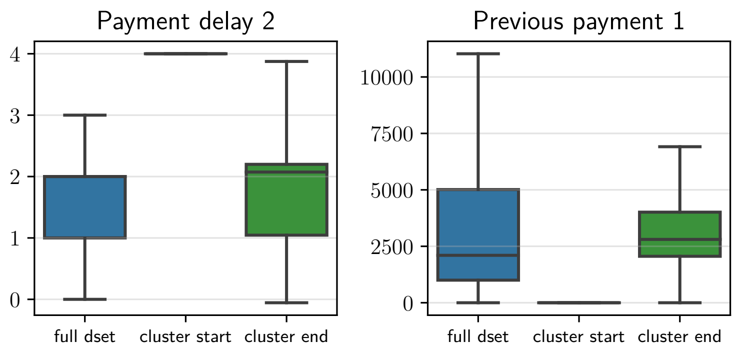

Fig. 5.10 CLUE summary boxplots for cluster 9.

Table 5.19 CLUE summary explanation for cluster 5.

Cluster 5	Current Value (Percentile)		Resulting Value (Percentile)
Probability of Default	0.25 (68.73)	→	0.12 (32.32)
Feature	**Current Value (Percentile)**		**Required Value (Percentile)**
Education: high school	39.46%	→	59.46%
Payment delay 5 (months)	3.28 (90.11)	→	1.63 (33.49)
Payment delay 6 (months)	2.29 (89.74)	→	1.62 (35.41)
Previous payment 1 ($)	2721 (55.96)	→	4126 (69.15)
Previous payment 3 ($)	5650 (80.96)	→	3203 (67.87)
Previous payment 4 ($)	1029 (43.39)	→	2850 (59.58)

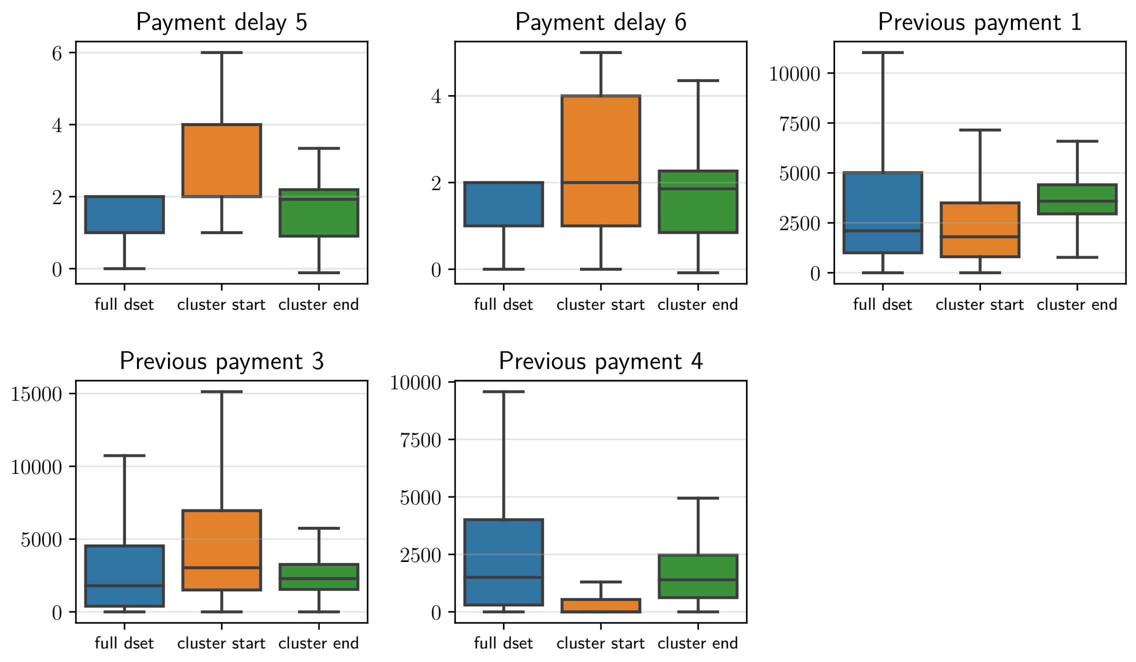

Fig. 5.11 CLUE summary boxplots for cluster 5.

Chapter 6

Conclusions and Future Work

This last chapter is dedicated to discussing the results obtained in this thesis and promising avenues for future work.

6.1 Discussion

In this work, we explore the application of neural network interpretability methods to uncertainty estimates provided by BNNs. We aim to develop methods that help us better understand why our models predict certain inputs as being more aleatoric or epistemic. Despite the rapidly growing body of work in ML interpretability, this topic has been mostly ignored. The only existing approach to uncertainty interpretability, uncertainty sensitivity analysis (Depeweg et al., 2017), attempts to model each input feature's contribution to each type of uncertainty using a sum of linear approximations taken at each test point. Our experiments show that this produces a crude approximation that does not hold for complex models such as BNNs.

Counterfactual interpretability methods attempt to explain a complex model's decisions for specific inputs by generating alternative inputs which are similar to the original ones but for which the model's decisions would have been different. We extend this concept to uncertainty: Counterfactual uncertainty explanations consist of alternative input configurations to which the model of interest assigns less uncertainty. In this vein, we adapt FIDO, a state-of-the-art method from the counterfactual interpretability literature, to generate uncertainty explanations, naming it U-FIDO.

We propose a novel counterfactual uncertainty interpretability technique: Counterfactual Latent Uncertainty Explanations or CLUE. This method searches the latent space of a VAE for latent vectors that generate input configurations which have low uncertainty while being similar to the input we wish to explain. CLUE can generate explanations for both images and tabular data. More interestingly, unlike most ML interpretability methods which are limited to explaining classification decisions, targeting the uncertainty in predictions allows CLUE to generate explanations for both classification and regression models. We also propose a summarisation technique which uses a Gaussian mixture model to group similar CLUE explanations, detecting trends in our datasets. Finally, we propose a novel functionally-grounded framework for the evaluation of counterfactual uncertainty interpretability techniques. This approach consists of using a conditional generative model as a ground-truth generative process. We use it to both generate samples with which to train our models and obtain ground-truth estimates of our explanations' uncertainty and log-likelihood.

We first use the aforementioned framework to investigate CLUE's regularisation parameter, finding that it has limited influence over the generated counterfactual explanations. As a consequence, our method is somewhat inflexible in terms of adjusting the trade-off between proposing explanations that are similar to the original datapoints and proposing low uncertainty explanations. We then compare CLUE with U-FIDO and a localised version of uncertainty sensitivity analysis. The latter performs weakly. Although it is able to produce samples which reduce a BNN's uncertainty, this reduction does not translate onto the ground-truth generative model's uncertainty. Furthermore, the log-likelihoods assigned to local sensitivity explanations reveal them to be OOD. These results match our expectations, suggesting the validity of our evaluation framework. Both U-FIDO and CLUE produce in-distribution explanations. While CLUE manages to explain away more of the uncertainty in the datapoints it is provided, U-FIDO's masking mechanism allows it to generate explanations that differ less from the original datapoints. CLUE outperforms U-FIDO in 5 out of our 8 quantitative experiments. We also perform a qualitative assessment of CLUE's and U-FIDO's explanations for high uncertainty MNIST digits, finding that CLUE produces cleaner, more easily understandable explanations. This matches our quantitative results for this dataset. Lastly, we provide examples of how our summarisation technique can be used to find intuitive counterfactuals that are applicable to vast sets of datapoints.

6.2 Future Work

The results obtained in this thesis suggest multiple promising avenues for future research into making the uncertainty estimates from BNNs more interpretable.

Although we have shown that counterfactual inputs can successfully be used to explain uncertainty, we have not delved into how uncertainty explanations differ from explanations for regular predictions. We expect this line of research to be most fruitful for regression problems, which have been mostly ignored by the ML interpretability community. Here, minimising uncertainty while maintaining the predicted values fixed provides us with a straight forward way to interrogate our models about their reasoning. For classification tasks, we may find that both approaches provide similar results, as minimising a categorical distribution's entropy is equal to maximising the probability of the chosen class.

In this work, we have not performed extensive analysis on how our explanations of predictive uncertainty alter models' predictive means. Intuitively, a counterfactual uncertainty explanation is only useful if it leaves the expectation of the predictive distribution unchanged. Although neither U-FIDO nor CLUE place any constraints on this value, modifying them to do so is straight forward. These methods' DGMs could be substituted by conditional generative models which capture joint distributions over inputs and targets, allowing explanations to be conditioned on particular output values.

Our comparisons of CLUE and U-FIDO reveal that there are advantages to both approaches and suggest that better results might be obtained by incorporating a masking mechanic into CLUE. This would reduce the effects of the reconstruction error from CLUE's generative model, allowing the method to produce sparser explanations while maintaining its flexibility. Furthermore, by choosing the number of masked pixels, we could obtain more control over the distance between original samples and explanations. This would allow for more informative comparisons to be drawn between methods by measuring the amount of uncertainty explained at successively increasing values of $|\Delta_x|$. Mask probabilities could also be used to quantify feature significance, thus replacing the current percentile based heuristic.

Despite our encouraging results, many questions remain open with regards to our proposed functionally-grounded evaluation scheme. It is unclear how much of the variability in the original datasets is lost when training our ground-truth VAEAC. A weak ground-truth model could lead to strong results on our evaluation metrics not translating to real-world tasks. Ideally, our evaluation methods should be cross-validated with human-based evaluation on a variety of tasks. This would ensure that methods that perform well within our framework also do so in the real world.

References

Adel, T., Ghahramani, Z., and Weller, A. (2018). Discovering interpretable representations for both deep generative and discriminative models. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 50–59, Stockholmsmässan, Stockholm Sweden. PMLR.

Adel, T., Valera, I., Ghahramani, Z., and Weller, A. (2019). One-network adversarial fairness. In *AAAI 2019*.

Alvarez Melis, D. and Jaakkola, T. (2018). Towards robust interpretability with self-explaining neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 7775–7784. Curran Associates, Inc.

Antoran, J. and Miguel, A. (2019). Disentangling in Variational Autoencoders with Natural Clustering. *arXiv e-prints*, page arXiv:1901.09415.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46.

Bengio, Y. (2013). Estimating or propagating gradients through stochastic neurons. *CoRR*, abs/1305.2982.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622. JMLR.org.

Chang, C.-H., Creager, E., Goldenberg, A., and Duvenaud, D. (2019). Explaining image classifiers by counterfactual generation. In *International Conference on Learning Representations*.

Chen, C., Li, O., Barnett, A., Su, J., and Rudin, C. (2018a). This looks like that: deep learning for interpretable image recognition. *CoRR*, abs/1806.10574.

Chen, C., Lin, K., Rudin, C., Shaposhnik, Y., Wang, S., and Wang, T. (2018b). An interpretable model with globally consistent explanations for credit risk. *CoRR*, abs/1811.12615.

Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient hamiltonian monte carlo. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1683–1691, Bejing, China. PMLR.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical japanese literature.

Corduneanu, A. and Bishop, C. M. (2002). Variational bayesian model selection for mixture distributions.

Council of European Union (2016). General data protection regulation. https://gdpr-info.eu.

Cowan, N. (2010). The magical mystery four: How is working memory capacity limited, and why? *Current directions in psychological science*, 19(1):51–57.

Dabkowski, P. and Gal, Y. (2017). Real time image saliency for black box classifiers. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6967–6976. Curran Associates, Inc.

Dai, B. and Wipf, D. (2019). Diagnosing and enhancing VAE models. In *International Conference on Learning Representations*.

Depeweg, S. (2019). *Modeling Epistemic and Aleatoric Uncertainty with Bayesian Neural Networks and Latent Variables*. PhD thesis, Technical University of Munich.

Depeweg, S., Hernandez-Lobato, J.-M., Doshi-Velez, F., and Udluft, S. (2018). Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1184–1193, Stockholmsmässan, Stockholm Sweden. PMLR.

Depeweg, S., Hernández-Lobato, J. M., Udluft, S., and Runkler, T. A. (2017). Sensitivity analysis for predictive uncertainty. In *ESANN*.

Dhurandhar, A., Chen, P.-Y., Luss, R., Tu, C.-C., Ting, P., Shanmugam, K., and Das, P. (2018). Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 592–603. Curran Associates, Inc.

Doshi-Velez, F. and Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. *arXiv e-prints*, page arXiv:1702.08608.

Douglas, L., Zarov, I., Gourgoulias, K., Lucas, C., Hart, C., Baker, A., Sahani, M., Perov, Y., and Johri, S. (2017). A universal marginalizer for amortized inference in generative models. *CoRR*, abs/1711.00695.

Dribe, M., Hacker, J. D., and Scalone, F. (2014). The impact of socio-economic status on net fertility during the historical fertility decline: a comparative analysis of canada, iceland, sweden, norway, and the usa. *Population studies*, 68(2):135–149. 24684711[pmid].

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org.

Gal, Y. and Smith, L. (2018). Sufficient Conditions for Idealised Models to Have No Adversarial Examples: a Theoretical and Empirical Study with Bayesian Neural Networks. *ArXiv e-prints*.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 881–889, Lille, France. PMLR.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

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

Graves, A. (2011). Practical variational inference for neural networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc.

Guo, W., Huang, S., Tao, Y., Xing, X., and Lin, L. (2018). Explaining deep learning models – a bayesian non-parametric approach. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4519–4529, USA. Curran Associates Inc.

Hernandez-Lobato, J., Li, Y., Rowland, M., Bui, T., Hernandez-Lobato, D., and Turner, R. (2016). Black-box alpha divergence minimization. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1511–1520, New York, New York, USA. PMLR.

Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1861–1869. JMLR.org.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

Hinton, G. E. and van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, pages 5–13, New York, NY, USA. ACM.

Iizuka, S., Simo-Serra, E., and Ishikawa, H. (2017). Globally and locally consistent image completion. *ACM Trans. Graph.*, 36(4):107:1–107:14.

Ivanov, O., Figurnov, M., and Vetrov, D. (2019). Variational autoencoder with arbitrary conditioning. In *International Conference on Learning Representations*.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., and sayres, R. (2018). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677, Stockholmsmässan, Stockholm Sweden. PMLR.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436 EP –.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

Li, C., Chen, C., Carlson, D., and Carin, L. (2016). Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1788–1794. AAAI Press.

Lloyd, J. R., Duvenaud, D., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. (2014). Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1242–1250. AAAI Press.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Ma, Y.-A., Chen, T., and Fox, E. B. (2015). A complete recipe for stochastic gradient mcmc. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2917–2925, Cambridge, MA, USA. MIT Press.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *J. Mach. Learn. Res.*, 18(1):4873–4907.

Montavon, G., Samek, W., and Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1 – 15.

Mudrakarta, P. K., Taly, A., Sundararajan, M., and Dhamdhere, K. (2018). Did the model understand the question? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1896–1906, Melbourne, Australia. Association for Computational Linguistics.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto.

Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., and Yosinski, J. (2017). Plug play generative networks: Conditional iterative generation of images in latent space. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3510–3520.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. (2016). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 3395–3403, USA. Curran Associates Inc.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. (2018). The building blocks of interpretability. *Distill*.

Paisley, J., Blei, D. M., and Jordan, M. I. (2012). Variational bayesian inference with stochastic search. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1363–1370, USA. Omnipress.

Petrov, D. and Hospedales, T. M. (2019). Measuring the transferability of adversarial examples. *CoRR*, abs/1907.06291.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China. PMLR.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "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*, KDD '16, pages 1135–1144, New York, NY, USA. ACM.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*.

Rosca, M., Lakshminarayanan, B., and Mohamed, S. (2018). Distribution Matching in Variational Inference. *arXiv e-prints*, page arXiv:1802.06847.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153, International Convention Centre, Sydney, Australia. PMLR.

Shu, H. and Zhu, H. (2019). Sensitivity Analysis of Deep Neural Networks. *arXiv e-prints*, page arXiv:1901.07152.

Smith, L. and Gal, Y. (2018). Understanding Measures of Uncertainty for Adversarial Example Detection. In *UAI*.

Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. (2016). Bayesian optimization with robust bayesian neural networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 4134–4142. Curran Associates, Inc.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 3319–3328. JMLR.org.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1139–III–1147. JMLR.org.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Ustun, B., Spangher, A., and Liu, Y. (2019). Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 10–19, New York, NY, USA. ACM.

Wachter, S., Mittelstadt, B. D., and Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *CoRR*, abs/1711.00399.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 681–688, USA. Omnipress.

Williams, J. J., Kim, J., Rafferty, A., Maldonado, S., Gajos, K. Z., Lasecki, W. S., and Heffernan, N. (2016). Axis: Generating explanations at scale with learnersourcing and machine learning. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, L@S '16, pages 379–388, New York, NY, USA. ACM.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. (2019). Cyclical stochastic gradient MCMC for bayesian deep learning. *CoRR*, abs/1902.03932.

Zintgraf, L. M., Cohen, T. S., Adel, T., and Welling, M. (2017). Visualizing deep neural network decisions: Prediction difference analysis. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.