Counterfactual Latent Uncertainty Explanations

Tanvi Shetty
tanvi.shetty@tu-dortmund.de
Technical University Dortmund
Germany

ABSTRACT

Uncertainty plays a pivotal role in machine learning as it tells us if we can trust the predictions made by the model. So, estimating uncertainty and being able to interpret it is crucial to avoid overconfident wrong predictions. The method discussed in this report is Counterfactual Latent Uncertainty Explanations (CLUE). It indicates how to alter the input while keeping it in the same distribution, such that confident predictions are obtained. CLUE interprets the uncertainty estimates from differentiable probabilistic models, like Bayesian Neural Networks (BNNs).

KEYWORDS

uncertainity, interpretability, estimation, bayesian neural network

ACM Reference Format:
Tanvi Shetty. 2022. Counterfactual Latent Uncertainty Explanations. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/.XXXXXXX

1 INTRODUCTION

Science and technology have made significant advances due to machine learning. Whether humans are directly using machine learning classifiers as tools or are deploying models within other products, one of the vital concerns is overconfident predictions which are incorrect. Models make very confident wrong predictions for inputs that are difficult to classify. So, in these situations, it would be preferred if the model said it was not sure about the prediction rather than providing an incorrect prediction.

Capturing uncertainty can help in avoiding such overconfident wrong predictions. For this, probabilistic machine learning models, such as Bayesian neural networks (BNNs), can be used since they provide reliable uncertainty estimates [6]. BNNs can capture uncertainty and translate it to their predictions. If the user is aware that a model is uncertain about certain decisions, the incorrect predictions can be avoided by throwing away the uncertain predictions.

Instead of only estimating the uncertainty, trying to understand why the predictions were uncertain can also help the practitioner greatly. Machine learning practitioners can learn more about their datasets by understanding what makes a model uncertain. When a model's parameters in some regions of the input space are unknown, it indicates little data in those areas. More data from these areas can help constrain the model's parameters more effectively, resulting in more reliable predictions. When there is irreducible uncertainty in a dataset, more input features must be measured to produce reliable predictions.

Moreover, it helps identify unreliable measuring techniques. Understanding what makes complex models unpredictable has received very little research. The work discussed in this report explores how uncertainty can be used in ML to improve interpretability and a new method to explain BNN uncertainty estimates. The proposed method is called CLUE. It is used to explain uncertainty as show in the Figure 1.

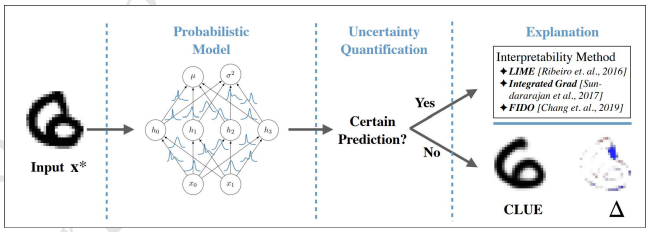

Figure 1: Why is CLUE required?

This report begins by explaining Bayesian Methods in Deep Learning in Section 2. Some related work on which the proposed method is built will be discussed in Section 3. Section 4 explains the method to estimate uncertainty proposed in the paper being discussed, and section 4 briefs about the experimental validations performed. In section 5, the advantages and disadvantages of the proposed model are discussed.

2 BAYESIAN METHODS

This section explains the fundamental concepts from the ML literature on which the proposed method is based. The uncertainty estimation in BNNs is briefly described in section 2.1. The following section, 2.2, discusses recent developments in Deep Generative Models (DGMs), which can be applied to feature imputation.

2.1 Uncertainty Estimation in BNNs

Probabilistic models known as Bayesian neural networks (BNN) incorporate neural networks' adaptability within a Bayesian framework. Making predictions with Bayesian neural networks requires marginalizing over parameter distributions, as shown in Figure 2. In particular, latent input variables have been added to BNNs to estimate functions with complex stochasticity, including bimodality or heteroscedasticity. This model class can account for model uncertainty via a distribution over weights (epistemic uncertainty) while also describing complicated stochastic patterns via a distribution

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/.XXXXXXX

over the latent input variables (aleatoric uncertainty). The heteroscedastic Gaussian likelihood functions are employed for regression, and their standard deviation is used to express uncertainty. We use the categorical distribution's entropy to measure uncertainty in classification. It is possible to divide predictive uncertainty into Aleatoric uncertainty and Epistemic entropy, and each explains information to practitioners in a distinct way [3]. The data's generative process contains inherent noise that causes irreducible or aleatoric uncertainty, which typically appears as class overlap. The lack of knowledge regarding the weights is represented by a model or epistemic doubt. Epistemic uncertainty emerges when we query points beyond the training manifold because the model is underspecified by the data. The posterior distribution over the predictor's

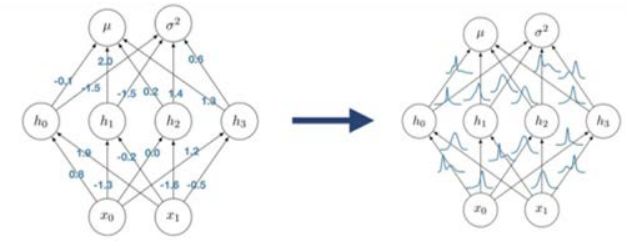

Figure 2: Bayesian neural network (BNN) is formed by combining a neural network with Bayesian inference.

parameters, given a dataset, a prior on our model's weights, and a likelihood function, captures our uncertainty about the value the weights should take. This parameter uncertainty is converted into predictive uncertainty through marginalization, producing trustworthy error boundaries and avoiding overfitting.

The posterior over parameters and the prediction distribution for BNNs are both intractable. The paper uses scale-adapted Stochastic Gradient Hamiltonian Monte Carlo (SG-HMC) [11] to approximate these objects. The main advantage of BNNs over conventional NNs is the ability to capture model uncertainty. 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.

2.2 Deep Latent Variable Models

Neural networks are used by Deep Generative Models (DGM) to model complex and high-dimensional data. Deep Latent Variable Models (DLVM), a class that assumes that data can be created from a set of latent variables, are the focus of the work under discussion.

2.2.1 Variational Autoencoders. An autoencoder whose training is regulated to prevent overfitting and guarantee that the latent space has good properties that enable generative processes is referred to as a variational autoencoder. A variational autoencoder is a two-part architecture consisting of an encoder and a decoder. It is trained to reduce the amount of reconstruction error between the encoded-decoded data and the initial data. It performs similar to a typical autoencoder. To introduce some regularization into the latent space, we encode an input as a distribution over the latent space rather than as a single point. It modifies the encoding-decoding process slightly.

A distribution over the latent space is encoded as the input during training. After that, a point is sampled from that distribution from the latent space. Further, the sampled point is decoded, and the reconstruction error can be computed. As a final step, backpropagation of reconstruction error occurs.

2.2.2 Variational Autoencoders with Arbitrary Conditioning. High-quality artificial samples can be generated using VAEs. Unfortunately, unlike other DGMs [5, 7, 9], the stock VAE framework provides no straightforward way to perform conditional generation. So, the Variational Autoencoder with Arbitrary Conditioning (VAEAC) proposed [10], a VAE-based solution to this problem. In this model, an arbitrary subset of observed features is conditionally encoded by a variational autoencoder, and the remaining features are sampled in a single shot. Both real-valued and categorical features may be used. Stochastic variational Bayes performs training of the model.

3 RELATED WORK

The literature on machine learning interpretability is discussed in this section. Since so little has been done to use Bayesian methods to improve the interpretability of machine learning models, there is little overlap between the two areas in the publications currently available. However, given that many of the fundamental principles that have influenced the development of these methodologies are also applicable to uncertainty interpretability, it is crucial to assess current interpretability approaches. Subsection 3.1 discusses counterfactual explanations, and Subsection 3.2 discusses uncertainty sensitivity analysis [4], an existing approach for understanding uncertainty estimates in BNNs.

3.1 Counterfactual Explanations

The machine learning community has emerged as a multidisciplinary community of researchers and industry practitioners interested in creating techniques to identify bias in machine learning models, creating algorithms to combat bias, producing explanations for machine decisions that are understandable to humans, holding companies accountable for unfair decisions, etc. The developers of machine learning models can use explanations to find, isolate, and resolve faults and other performance problems. In various instances, human-readable explanations for machine-generated decisions are helpful. It aids the user in understanding which of their attributes served as powerful motivators when making a decision. Checking their algorithms for bias can also be helpful. In some situations, an explanation gives the user feedback they can use to get the desired result later [12]. "Counterfactual explanations" are a particular category of explanation that connects what would have occurred if a model's input had been altered in a particular way. Contrary to other explainability strategies, counterfactual explanations offer recommendations on how to get the desired result rather than directly responding to the "why" aspect of a decision.

Counterfactual explanations are typically constructed by addressing an optimization problem similar to:

$$\mathbf{x}_c = \arg\max_{\mathbf{x}} \left(p_I(\mathbf{y} = c \mid \mathbf{x}) - d\left(\mathbf{x}, \mathbf{x}_0\right)\right) \text{ s.t. } \mathbf{y}_0 \neq c$$

Where $\mathbf{y}$ is the desired outcome, $\mathbf{x}_0$ refers to the original input to the predictor and $d()$ is some pairwise distance metric.

A machine learning model may use multiple inputs to get the same outcome; as a result, many counterfactual explanations may apply to a given input. The counterfactual set of input variables that comes the closest to the original input configuration is often what we are most interested in. To do this, CLUE looks for counterfactuals in an auxiliary DGM's lower-dimensional latent space. Given that the DGM limits CLUE's search space to the data manifold, this decision is ideally suited for uncertain situations.

3.2 Uncertainty Sensitivity Analysis

Uncertainty Sensitivity Analysis [4] is an existing technique for interpreting uncertainty estimations. This approach seeks to quantify the significance of each input dimension to a selected uncertainty metric. In order to achieve this, all test data points are averaged to obtain the gradients of the uncertainty measure for the selected input dimension:

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

Predictive entropy H is the chosen uncertainty metric in this expression. However, any alternative measure, such as aleatoric entropy, epistemic entropy, or standard deviations, can take its place. The equation can be viewed as a sum of linear approximations with centres at each test point that approximates a global explanation of a model's uncertainty. However, this approach can be misleading

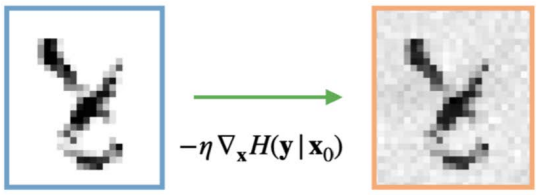

Figure 3: Noisy input configuration generated using Sensitivity Analysis

because predictive uncertainty measures are not linear functions of the input variables. It is especially true for models like NNs that are particularly non-linear., such as NNs. It is plausible that under these models, a change in a low sensitivity input dimension will result in a more significant change in the predicted uncertainty than a similar change in a high sensitivity dimension. There is a high likelihood that $\nabla_{\mathrm{x}} \mathcal{H}$w does not point in the direction of the data manifold in high-dimensional input spaces. It leads to meaningless explanations. This can be seen in the Figure 3.

4 PROPOSED METHOD

This section presents a novel method for uncertainty interpretability called Counterfactual Latent Uncertainty Explanations (CLUE) [1]. The explanations given by the method are referred to as CLUEs. CLUE helps to find the slightest change that would have had to be made to input while maintaining it in distribution for the model to have been more confident in its decision about said input. CLUE can provide explanations for regression and classification tasks on both tabular data and images. In section 4.1, the core CLUE algorithm is explained. 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 Workflow

A counterfactual explanation for uncertainty aims to identify a configuration of the input parameters that makes the uncertainty of a machine learning model drop relative to its uncertainty for reference input. It enables us to produce counterfactual explanations for classification and regression, where standard deviations and predicted entropy can be employed as an uncertainty metric. The model can be analyzed to find the input factors it thinks are crucial by identifying inputs that correspond to similar predictions but with narrower error bars.

Any model that generates an uncertainty measure differentiable with respect to its inputs can use CLUE. It qualifies as a post-hoc white-box interpretability method since it requires the repeated evaluation of these gradients. CLUE is a local method since it explains specific input space points. It does not, however, rely on rough linear approximations. CLUE's explanations have a concrete meaning because it is also a counterfactual approach. To ensure that its explanations correspond to realistic input parameter settings, CLUE employs a DGM. In the discussed study, a VAE is used.

As shown in Figure 4, the main idea of CLUE is to encode our input to a latent space. Then, perform some optimization that aims to minimize uncertainty. Finally, decode into some resulting input for which our model is more certain.

4.2 Algorithm

The CLUE algorithm is shown in the Figure 5. CLUE takes a code in latent space and decodes it using the generator from a variational autoencoder (VAE) into some point in a potentially complex high-dimensional input space. The initial input for which we are looking for a counterfactual explanation is denoted by $\mathbf{x}_0$. The VAE characterises the latent space as a feature landscape instead of just attempting to embed the data there, which makes the latent space more suitable for the generation of data. The VAE's encoder is denoted as $q_\phi(\mathbf{z} \mid \mathbf{x})$, and the decoder is denoted as $p_\theta(\mathbf{x} \mid \mathbf{z})$. Then a probabilistic predictor like BNN is used to estimate the uncertainty associated with predictions for this point. The differentiable uncertainty metric is referred to by the abbreviation H. The predicted means of these models are denoted as $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\mathbf{z}] = \mu_\phi(\mathbf{z} \mid \mathbf{x})$ and $\mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[\mathbf{x}] = \mu_\theta(\mathbf{x} \mid \mathbf{z})$, respectively. A gradient optimiser is used to minimise the CLUE objective given below.

$$\mathscr{L}(\mathbf{z}) = H\left(\mathbf{y} \mid \mu_\theta(\mathbf{x} \mid \mathbf{z})\right) + \lambda \left\|\mu_\theta(\mathbf{x} \mid \mathbf{z}) - \mathbf{x}_0\right\|_1$$

The predictive uncertainty of our generations and the distance measure between our original points and generations are used to optimise the objective. The distance metric should be decided depending on the task. The initial value of $\mathbf{z}$ is set to $\mathbf{z}_0 = \mu_\phi\left(\mathbf{z} \mid \mathbf{x}_0\right)$ to aid optimisation. The trade-off between generating outputs with

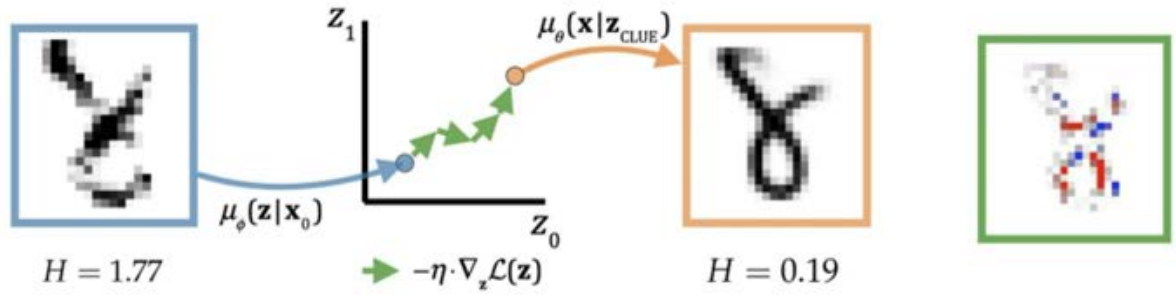

Figure 4: Main Idea of CLUE

Algorithm 1: CLUE

Inputs: original datapoint $\mathbf{x}_0$, distance function $d(\cdot,\cdot)$, Uncertainty estimator $\mathcal{H}$, DGM decoder $\mu_\theta(\cdot)$, DGM encoder $\mu_\phi(\cdot)$

Set initial value of $\mathbf{z} = \mu_\phi(\mathbf{z}|\mathbf{x}_0)$;
while *loss* $\mathcal{L}$ *is not converged* **do**
  Decode: $\mathbf{x} = \mu_\theta(\mathbf{x}|\mathbf{z})$;
  Use predictor to obtain $\mathcal{H}(\mathbf{y}|\mathbf{x})$;
  $\mathcal{L} = \mathcal{H}(\mathbf{y}|\mathbf{x}) + d(\mathbf{x}, \mathbf{x}_0)$;
  Update $\mathbf{z}$ with $\nabla_\mathbf{z}\, \mathcal{L}$;
end
Decode explanation: $\mathbf{x}_{\text{CLUE}} = \mu_\theta(\mathbf{x}|\mathbf{z})$;

Output: Uncertainty counterfactual $\mathbf{x}_{\text{CLUE}}$

Figure 5: CLUE algorithm

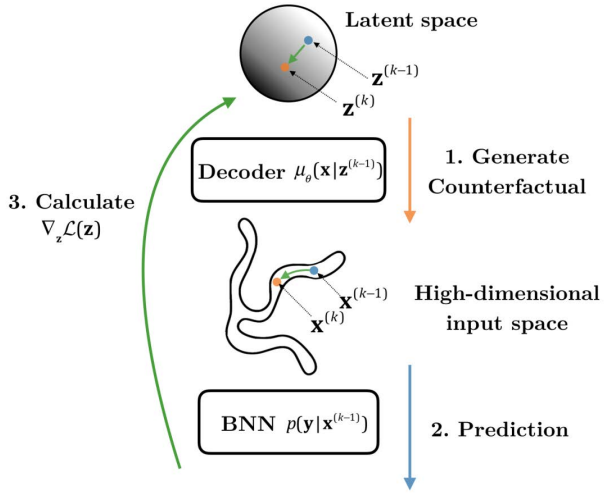

Figure 6: Optimization process of CLUE

minimal uncertainty and results similar to the initial input is controlled by the hyperparameter λ. In order to make this hyperparameter independent of the input dimensionality, we often set $\lambda = \lambda_0/n(\mathbf{x})$, where $n(\cdot)$ is a function that returns the number of elements in a vector. Figure 6 depicts the optimisation process. The gradient-based optimiser can be used for optimisation to update our latent code with an estimate of uncertainty by differentiating through both our probabilistic predictor and our VAE decoder. Taking free-form updates in this latent space ensures that our updates result in points in manifold in our high dimensional input space. CLUEs are obtained as:

$$\mathbf{x}_{\text{CLUE}} = \mu_\theta\left(\mathbf{x} \mid \mathbf{z}_{\text{CLUE}}\right); \quad \mathbf{z}_{\text{CLUE}} = \arg\min_{\mathbf{z}} \mathscr{L}(\mathbf{z})$$

4.3 Multiplicity

For any given input, there might be a variety of plausible changes that can be made in order to make the model more confident about the input. It is reflected in the fact that the objective of CLUE is non-convex, and it finds only the local optima. Because of this, multiple plausible explanations can be obtained. All explanations, however, mirror the original data points being explained. So, the non-convexity of CLUE's objective is exploited to generate diverse CLUEs, as seen in Figure 7.

Exposure to this diversity may alert practitioners to the similarities between a model's original input and various classes that make it uncertain. As a result of different initializations, various amounts of uncertainty get eliminated by CLUEs. Occasionally, CLUE may also fail to provide a feature configuration with a considerable reduction in uncertainty over the original input.

5 EVALUATING COUNTERFACTUAL EXPLANATIONS OF UNCERTAINTY

Counterfactuals should be relevant, lying close to the original inputs and representing plausible parameter values close to the data manifold. They should also be informative, highlighting elements that affect the uncertainty of our BNN. Access to the data's generative process is necessary for evaluating these criteria.

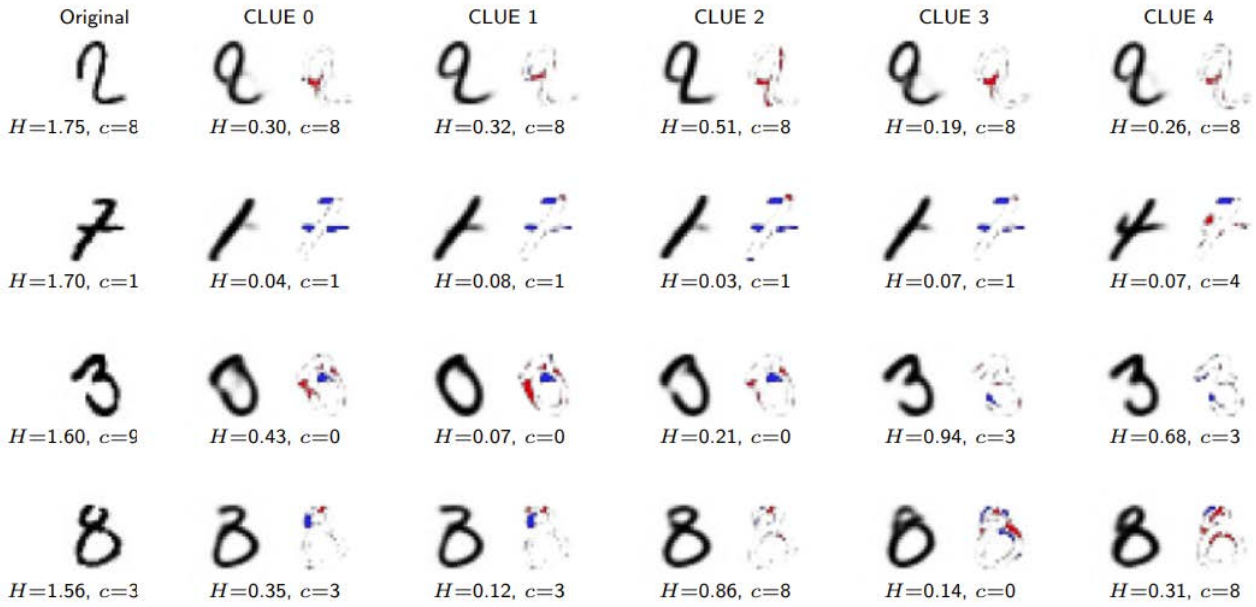

Figure 7: Multiplicity of CLUE

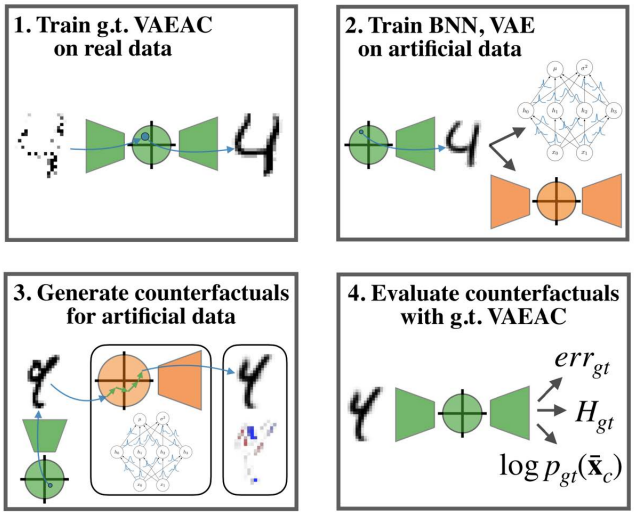

Figure 8: Evaluation of Counterfactual Explanations of Uncertainty Computationally

Figure 8 shows how the evaluation of counterfactual explanations of uncertainty computationally. The VAEAC, which we consider a data generation process, is highlighted in green, and the auxiliary DGM used by the technique under evaluation is highlighted in orange.

First, a g.t. VAEAC is trained on a real dataset to model the inputs and targets jointly. It also allows us to query the conditional distribution over targets given inputs. After that, the BNN and an auxiliary DGM are trained on artificial data, which has the same distribution. In the third step, additional artificial data is sampled, and counterfactual explanations are generated for the uncertain samples. Finally, the g.t. VAEAC is used to obtain the conditional distribution over targets given counterfactual inputs $p_{\text{gt}}(\mathbf{y} \mid \bar{\mathbf{x}}_c)$ and the input's true uncertainty $\mathcal{H}_{\text{gt}}$. It helps evaluate whether counterfactuals are on-manifold through $\log p_{\text{gt}}(\bar{\mathbf{x}}_c)$. Also, $\mathcal{H}_{\text{gt}}$ allows us to evaluate if the generated counterfactuals address the actual sources of uncertainty in the data.

6 EXPERIMENTAL VALIDATION

In this section, we evaluate CLUE's performance relative to baseline uncertainty interpretability methods and discuss the user study performed.

6.1 Computational Evaluation

There are no methods that CLUE can be directly compared to because there hasn't been much work done on interpreting uncertainty estimations. To produce counterfactual uncertainty explanations, two current ML interpretability algorithms are modified. These give us benchmarks against which to evaluate CLUE. Using the evaluation framework presented in Section 5, we compare CLUE, Localized Sensitivity, and U-FIDO.

In local sensitivity analysis, a single datapoint is used to produce a local analogue of uncertainty sensitivity analysis [4]. A counterfactual explanation that reduces the uncertainty of the BNN can be constructed by just moving in the gradient direction. FIDO [2], a counterfactual interpretability method for images, is adopted as the second baseline to explain uncertainty (U-FIDO). To reduce the classification score, FIDO seeks to identify the smallest region

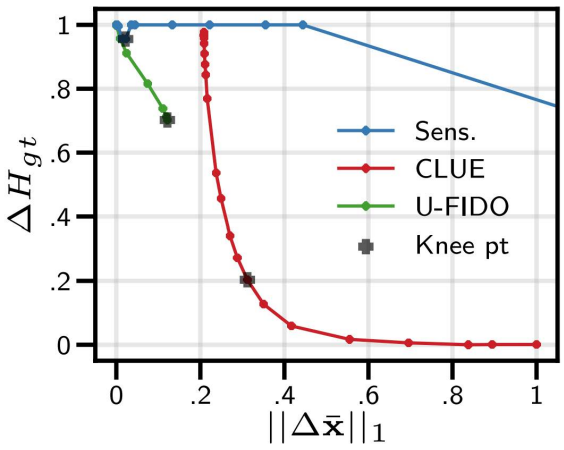

Figure 9: Informativeness to relevance trade-off

that must be removed and replaced with an uninformative input. We accomplish this by substituting an uncertainty metric for the classification score. The counterfactuals should retain as much of the original input as possible while explaining as much uncertainty. With the use of the hyperparameters η, λ_x, and λ_b for Local Sensitivity, CLUE, and U-FIDO, respectively, the informativeness (large $\Delta\mathcal{H}$gt) to relevance (small $\|\Delta\bar{x}\|1$) trade-off is controlled. In order to plot Pareto-like curves, a logarithmic grid search across hyperparameters is carried out. The minimum values for our two measures are 0, but their maximum values depend on the dataset and the approach used. Relevance for Sensitivity increases linearly with η. These measurements saturate for both high and small values of λx (or λ_b) for CLUE and U-FIDO. The values acquired using these methods do not overlap as a result. As seen in Figure 9, CLUE is better equipped to account for uncertainty than U-FIDO, and U-FIDO consistently generates smaller relevance values than CLUE.

6.2 User Study

The Machine Learning interpretability method's ultimate purpose is to enable humans to use it to help with real-world tasks. Human-based evaluation is a crucial step in confirming the effectiveness of ML explainability tools [8]. It is essential to evaluate how much CLUEs, as opposed to human intuition or local sensitivity, assist machine learning practitioners in identifying the sources of uncertainty in ML models. To achieve this, a forward-simulation task that focuses on a suitable local test to assess CLUEs has been suggested. The user study makes use of the LSAT and COMPAS datasets. Each variation of the main survey has ten distinct participants. Graduate ML students who act as stand-ins for professionals in the industry are the participants. There are 18 questions in the main survey, 9 for each dataset. Figure 10 displays a sample question.

The practitioners are shown two context points: the certain context point, which is below the rejection threshold, and the uncertain context point, which is above the threshold. The certain context point serves as a local counterfactual explanation for the uncertain context point. Using both context points as references, practitioners are asked to predict whether a new test point will be above or

	Uncertain
Age	Less than 25
Race	Caucasian
Sex	Male
Current Charge	Misdemeanour
Reoffended Before	Yes
Prior Convictions	1
Days Served	0

	Certain
Age	Less than 25
Race	African-American
Sex	Male
Current Charge	Misdemeanour
Reoffended Before	No
Prior Convictions	0
Days Served	0

	?
Age	Less than 25
Race	Hispanic
Sex	Male
Current Charge	Misdemeanour
Reoffended Before	No
Prior Convictions	0
Days Served	0

Figure 10: Example of data user study

Table 1: Results of the user study

	Combined	LSAT	COMPAS
CLUE	82.22	83.33	81.11
Human CLUE	62.22	61.11	63.33
Random	61.67	62.22	61.11
Local Sensitivity	52.78	56.67	48.89

below the set threshold. In general, this will indicate whether the uncertainty of the BNN for the new point will be high or low. The utility of particular context points created by CLUE is compared in the survey to four distinct approaches, which differ in how certain context points are selected. The four methods are random selection of a certain point from the test set as a control, generation of a counterfactual certain point with Local Sensitivity, CLUE, or Human CLUE.

Participants who will not take the main survey are requested to pair uncertain context points with related certain points in order to produce a Human CLUE. As seen in Figure 11, a pilot procedure was used to choose the points used in our main survey. It avoids biases being added to the point selection process and guarantees that context points are relevant to test points. A participant in the pilot procedure sees a pool of randomly chosen certain and uncertain points, and the participant is required to choose test points from this pool of options. The participant is then instructed to map each test point into an equivalent uncertain point without replacement. In this manner, uncertain context points relevant to test points are acquired.

Table 1 shows the results of the user study. The average accuracy of the participant using CLUE was 82.22%, Human CLUE was 62.22%, Random was 61.67%, and Local Sensitivity was 52.78%. CLUE outperformed the other three, and the Local Sensitivity performed worse than even Random.

7 ADVANTAGES AND DISADVANTAGES

The significant advantage of CLUE is that it helps make the uncertainty estimates from BNNs more interpretable. It helps the user understand why specific inputs generate uncertain predictions. There is very little prior work related to this. The features in the inputs can be found to make the model generate more certain predictions using CLUE. Overall, it helps in improving the transparency of models.

However, there are disadvantages to CLUE as well. There can be scenarios where the explanations generated and the alternative suggestions proposed, even though in the same data manifold, are not in the user's control. For instance, suggesting someone change their age is not actionable.

8 CONCLUSION

This report briefly discusses the method of Counterfactual Latent Uncertainty Explanations (CLUE). It is a method that can explain uncertainty estimates provided by BNNs. Despite the rapidly growing body of work in ML interpretability, this topic has been mostly ignored. Uncertainty sensitivity analysis [4], the only method currently in use to determine how interpretable uncertainty is, attempts to model each input feature's contribution to each type of uncertainty using a sum of linear approximations made at each test point. However, this approximation does not hold for complex models like BNNs.

Counterfactual interpretability methods attempt to explain a complex model's decisions for specific inputs by generating alternative inputs similar to the original ones but for which the model's decisions would have been different. This concept was extended to uncertainty, and alternative input configurations to which the model allocates low uncertainty make up counterfactual uncertainty explanations.

A VAE's latent space is searched by CLUE for latent vectors that produce input configurations similar to the input yet have low uncertainty. Images and tabular data can both be explained using CLUE. More intriguingly, CLUE may produce explanations for classification and regression models, in contrast to other ML interpretability techniques, which are restricted to explaining classification decisions. Additionally, a functionally-grounded framework for assessing counterfactual uncertainty interpretability methodologies was put forward.

This process requires employing a conditional generative model to generate ground truth data. It is used to create training samples for the model and to provide accurate estimates of the uncertainty and likelihood of an explanation. After that, we conduct a user study. It discovers that after being exposed to CLUEs, users can better forecast their model's behaviour.

9 ACKNOWLEDGMENTS

I want to express my special gratitude and thanks to the supervisors for allowing me to work on this topic and for their guidance.

REFERENCES

[1] Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. 2020. Getting a clue: A method for explaining uncertainty estimates. *arXiv preprint arXiv:2006.06848* (2020).
[2] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. 2018. Explaining image classifiers by counterfactual generation. *arXiv preprint arXiv:1807.08024* (2018).
[3] Stefan Depeweg. 2019. *Modeling epistemic and aleatoric uncertainty with bayesian neural networks and latent variables.* Ph. D. Dissertation. Technische Universität München.
[4] Stefan Depeweg, José Miguel Hernández-Lobato, Steffen Udluft, and Thomas Runkler. 2017. Sensitivity analysis for predictive uncertainty in Bayesian neural networks. *arXiv preprint arXiv:1712.03605* (2017).
[5] Laura Douglas, Iliyan Zarov, Konstantinos Gourgoulias, Chris Lucas, Chris Hart, Adam Baker, Maneesh Sahani, Yura Perov, and Saurabh Johri. 2017. A universal marginalizer for amortized inference in generative models. *arXiv preprint arXiv:1711.00695* (2017).
[6] Yarin Gal et al. 2016. Uncertainty in deep learning. (2016).
[7] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. 2015. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*. PMLR, 881–889.
[8] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. 2018. Metrics for explainable AI: Challenges and prospects. *arXiv preprint arXiv:1812.04608* (2018).
[9] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1–14.
[10] Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. 2018. Variational autoencoder with arbitrary conditioning. *arXiv preprint arXiv:1806.02382* (2018).
[11] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. 2016. Bayesian optimization with robust Bayesian neural networks. *Advances in neural information processing systems* 29 (2016).
[12] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.

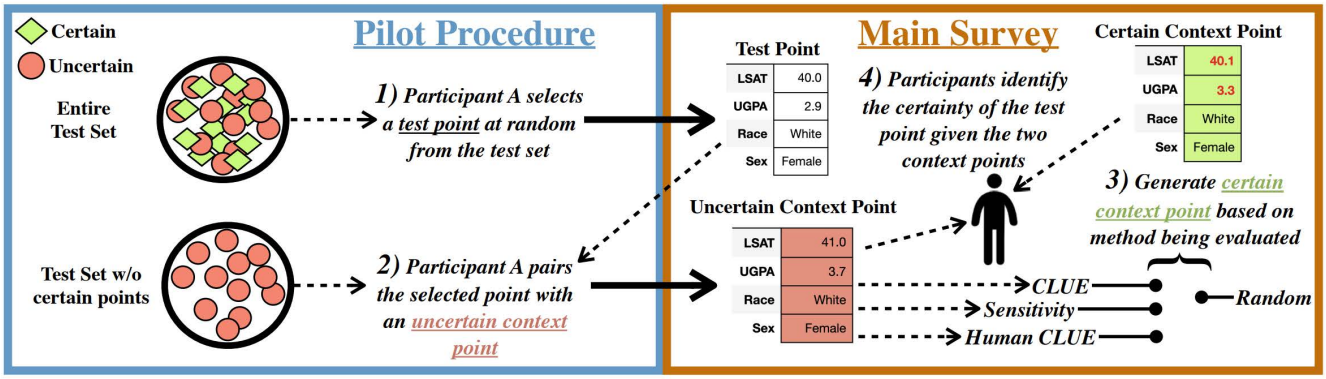

Figure 11: Workflow of User Study