GETTING A CLUE: A METHOD FOR EXPLAINING UNCERTAINTY ESTIMATES

Javier Antorán
University of Cambridge
ja666@cam.ac.uk

Umang Bhatt
University of Cambridge
usb20@cam.ac.uk

Tameem Adel
University of Cambridge
tah47@cam.ac.uk

Adrian Weller
University of Cambridge &
The Alan Turing Institute
aw665@cam.ac.uk

José Miguel Hernández-Lobato
University of Cambridge &
Microsoft Research &
The Alan Turing Institute
jmh233@cam.ac.uk

ABSTRACT

Uncertainty estimates from machine learning models allow domain experts to asses prediction reliability and can help practitioners identify model failure modes. We introduce Counterfactual Latent Uncertainty Explanations (CLUE), a method that answers: "How should we change an input such that our model produces more certain predictions?" We perform a user study, concluding that CLUE allows users to understand which regions of input space contribute to predictive uncertainty.

1 INTRODUCTION

Machine learning (ML) models that produce uncertainty estimates can be helpful in decision-critical applications. These applications are behind the burgeoning interest in probabilistic ML. Probabilistic models express model uncertainty and provide reliable error bars with their predictions (Gal, 2016). Well-calibrated uncertainty can be equally as important as making accurate predictions. It can prevent automated systems from behaving erratically when faced with unbalanced or biased datasets or with out-of-distribution (OOD) test cases.

In practice, predictive uncertainty conveys skepticism about a model's output; however, its utility need not stop there: we posit predictive uncertainty could be rendered useful and actionable were it to be put in terms of model inputs, answering the question: "Which input features lead my prediction to be uncertain?" This reformulation has two stakeholders: ML practitioners and domain experts working in tandem with ML models. Understanding which input features are responsible for epistemic uncertainty can help practitioners learn in which regions the training data is sparse. For example, when training a loan default predictor, a data scientist (i.e., practitioner) can identify sub-groups (by age, gender, race, etc.) under-represented in the training data. Collecting more data from these groups, and thus further constraining their model's parameters, could lead to accurate predictions for a broader range of clients. In a clinical scenario, a doctor (i.e., domain expert) can use an automated decision-making system to assess whether a patient should receive a treatment. In the case of high uncertainty, the system will suggest the doctor reject its output in order to avoid introducing noise into the doctor's reasoning. Were the uncertainty to be explained in terms of which features the model found anomalous, the doctor could direct their attention appropriately.

To that end, we present CLUE; to the best of our knowledge, the first algorithm for contextualizing the predictive uncertainty of Bayesian Neural Networks (BNN) in terms of specific input features. We proceed as follows: we review uncertainty in probabilistic models, describe our method, and analyze its properties with both functional and human-subject experiments.

2 PRELIMINARIES: UNCERTAINTY IN BAYESIAN MODELS

Given a dataset $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^N$, the posterior distribution over a predictive model's parameters, $p(\mathbf{w}|\mathcal{D})$, encodes our uncertainty about what value $\mathbf{w}$ should take. Through marginalization, this uncertainty is translated into predictive uncertainty, yielding reliable error bounds and preventing overfitting. For NNs, computing the exact posterior is intractable, but there are many approximate algorithms (MacKay, 1992; Hernández-Lobato & Adams, 2015). We consider two types of uncertainties: Irreducible or **aleatoric uncertainty** is caused by class overlap in our dataset. Model or **epistemic uncertainty** arises when points are off the training manifold or belong to sparse regions (Depeweg, 2019). Explaining these uncertainties separately can provide further insights to users.

We consider models which parameterize two types of distributions over target variables: the categorical for classification problems and the Gaussian for regression. In both cases, we approximate NNs' predictive posteriors and uncertainty estimates with Monte Carlo estimators. These are differentiable with respect to model inputs. Following Depeweg (2019), we quantify the uncertainty of categorical distributions using predictive entropy: $H(y^*|\mathbf{x}^*, \mathcal{D})$ and of Gaussians with variance. Leveraging their additive nature, these quantities can be decomposed into aleatoric (H_a, σ_a^2) and epistemic components (H_e, σ_e^2).

To the best of our knowledge, the only existing method for the interpretation of uncertainty estimates is Uncertainty Sensitivity Analysis (Depeweg et al., 2017). This method quantifies the global relevance of individual input dimensions to a chosen metric of uncertainty H as a sum of linear approximations centered at each test point:

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

As discussed by Rudin (2019), linear explanations of non-linear models, such as this one, can be misleading. High-dimensional input spaces limit the actionability of these explanations, as $\nabla_{\mathbf{x}} H$ likely will not point in the direction of the data manifold. Our method, CLUE, leverages the latent space of a deep generative model (DGM) to avoid having to work with high-dimensional input spaces and to ensure that its explanations are in-distribution. It does not rely on crude linear approximations.

3 OUR APPROACH

We propose Counterfactual Latent Uncertainty Explanations (CLUE). We refer to the explanations given by our method as CLUEs. They answer the question: "What is the smallest change that would have had to be made to an input, while staying in-distribution, for our model to have been more confident in its decision about said input?" We use $\mathbf{x}_0$ to refer to the original input for which we want to find a CLUE. We introduce a latent variable DGM: $p_\theta(\mathbf{x}) = \int (p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})\, d\mathbf{z}$, which, in the rest of this paper, will be the decoder from a variational autoencoder (VAE). The VAE's encoder is denoted as $q_\phi(\mathbf{z}|\mathbf{x})$. The predictive mean of the decoder is: $\mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[\mathbf{x}] = \mu_\theta(\mathbf{x}|\mathbf{z})$ and of the encoder is: $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\mathbf{z}] = \mu_\phi(\mathbf{z}|\mathbf{x})$. CLUE tries to find points in latent space that generate inputs similar to $\mathbf{x}_0$ while being assigned low uncertainty. This is achieved by minimizing:

$$\mathcal{L}(\mathbf{z}) = H(\mathbf{y}|\mu_\theta(\mathbf{x}|\mathbf{z})) + d(\mu_\theta(\mathbf{x}|\mathbf{z}), \mathbf{x}_0) \tag{2}$$

$$\mathbf{x}_{\text{CLUE}} = \mu_\theta(\mathbf{x}|\mathbf{z}_{\text{CLUE}}); \quad \mathbf{z}_{\text{CLUE}} = \arg\min_{\mathbf{z}} \mathcal{L}(\mathbf{z}) \tag{3}$$

We choose $d(\mathbf{x}, \mathbf{x}_0) = \lambda_x d_x(\mathbf{x}, \mathbf{x}_0) + \lambda_y d_y(f(\mathbf{x}), f(\mathbf{x}_0))$ in order to encourage similarity between uncertain points and CLUEs in both input and prediction space. The hyperparameters (λ_x, λ_y) control the trade-off between producing low uncertainty CLUEs and CLUEs which are close to the original inputs. We select $d_x(\mathbf{x}, \mathbf{x}_0) = \|\mathbf{x} - \mathbf{x}_0\|_1$. For regression, $d_y(f(\mathbf{x}), f(\mathbf{x}_0))$ is mean squared error. For classification, we use cross-entropy. A diagram of the minimization of (2) is shown in Figure 1. To facilitate optimization, 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. Our hyperparameters are described in appendix A.

Algorithm 1: CLUE

Inputs: original datapoint $\mathbf{x}_0$, distance function $d(\mathbf{x}, \mathbf{x}_0)$, BNN uncertainty estimator 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 BNN to obtain $H(\mathbf{y}|\mathbf{x})$;
  $\mathcal{L} = H(\mathbf{y}|\mathbf{x}) + d(\mathbf{x}, \mathbf{x}_0)$;
  Update $\mathbf{z}$ with $\nabla_\mathbf{z}\, \mathcal{L}$;
end
Decode explanation: $\mathbf{x}_{\text{CLUE}} = \mu_\theta(\mathbf{x}|\mathbf{z})$;

Output: Counterfactual example $\mathbf{x}_{\text{CLUE}}$

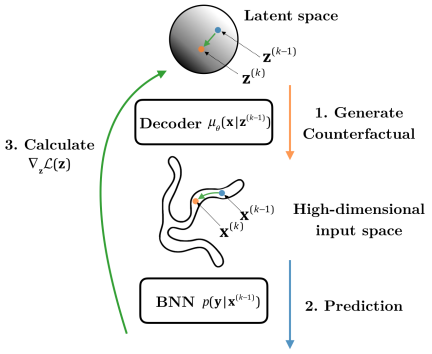


Figure 1: Latent codes are decoded into inputs for which a BNN generates uncertainty estimates; its gradients are backpropagated to latent space.

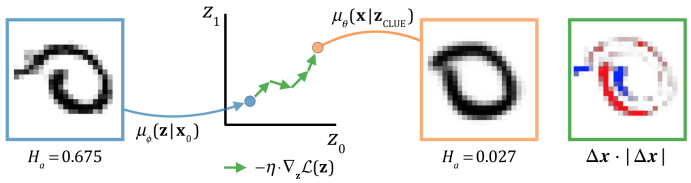
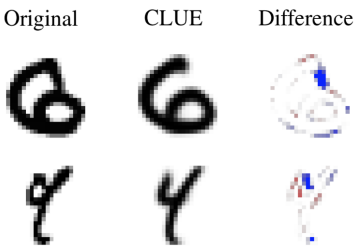


Figure 2: Illustration of how a CLUE is generated for a high aleatoric entropy MNIST digit. The mask, placed to the right, highlights pixel amplitude increases in red and decreases in blue.

We now describe how to show CLUEs to stakeholders. We do not want noise from DGM reconstruction to affect our CLUEs. To right this for tabular data, we use the change in the percentile of each input feature with respect to the training set distribution as a measure of relevance. We only highlight continuous variables for which the CLUE is separated by 15 percentile points or more from the original input. All changes to discrete variables are shown to users. For images, we generate masks by applying a sign-preserving quadratic function to the difference between CLUEs and original samples: $|\Delta\mathbf{x}| \cdot \Delta\mathbf{x}$ with $\Delta\mathbf{x} = \mathbf{x}_{\text{CLUE}} - \mathbf{x}_0$. In Figure 3, we showcase CLUEs for an uncertain sample from the COMPAS dataset and two high uncertainty MNIST digits.



	Person 0		CLUE
AI is uncertain	True	->	False
age	Greater than 45	->	25 - 45
race	African-American		-
sex	Female	->	Male
charge degree	Felony		-
recid before	not recid		-
priors count	1.0		-
days served before	0.0		-

Figure 3: Left: CLUEs for two MNIST digits. Right: a CLUE from the COMPAS dataset.

4 Experiments

We now describe empirical results for CLUE, leveraging the following datasets: MNIST, COMPAS, and LSAT. We train BNNs and VAEs on these datasets. For the former, we use scale adapted SG-HMC (Springenberg et al., 2016) as the approximate inference technique. We define rejected samples as those with uncertainty estimates above their dataset's 80$^{\text{th}}$ percentile. We only generate explanations for rejected samples. Implementation details can be found in appendix A.

4.1 Quantitative Analysis of CLUE

We introduce a localized version of sensitivity analysis that is directly comparable to CLUE; Explanations are generated for individual datapoints by taking a step in the direction of their uncertainty's

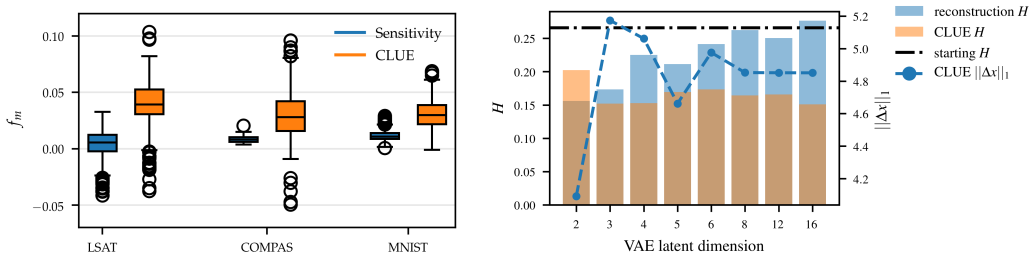


Figure 4: On the left, we compare the values of f_m obtained on the datasets under consideration. On the right, we compare the uncertainty assigned by our BNN to VAE reconstructions with the uncertainty assigned to CLUEs for different capacity VAEs on the COMPAS dataset. An 8 latent dimension VAE seems to be flexible enough to completely retain the original dataset's uncertainty.

gradient $\mathbf{x}_s = \mathbf{x}_0 - \eta \nabla_{\mathbf{x}} H(\mathbf{y}|\mathbf{x}_0)$. Averaging $|\mathbf{x}_s - \mathbf{x}_0|$ across the test set, we recover (1). We use $f_m = {}^{\Delta H}/_{\|\Delta \mathbf{x}\|_1}$ as a figure of merit, with $\Delta H = H(\mathbf{y}|\mathbf{x}_0) - H(\mathbf{y}|\mathbf{x}_{\text{CLUE}})$. Because BNNs are non-linear, local sensitivity can increase H if the step size is too large. Thus, a very small η is required, rendering the approach ineffective. In Figure 4, we see how CLUE obtains f_m scores an order of magnitude larger than local sensitivity analysis.

To capture our predictive model's reasoning, CLUE's DGM must be flexible enough to preserve atypical features in the inputs. As shown in Figure 4, COMPAS reconstructions from low-capacity VAEs are assigned low predictive uncertainty. The CLUEs generated from these DGMs either leave the inputs unchanged or present large values of $\Delta \mathbf{x}$ while barely reducing H: these degenerate CLUEs simply emphasize regions of large reconstruction error. As our DGM's capacity increases, so does the predictive uncertainty explained by CLUEs. For 5 or more VAE latent dimensions, reconstructions preserve the BNN's uncertainty about inputs, allowing it to be explained by CLUE.

4.2 Qualitative Utility of CLUE: User Study

We conduct a human subject experiment to assess how well CLUEs help users identify whether a model will be uncertain on new datapoints. Specifically, we run two experiments: one with 20 graduate students and another with 30 individuals on the Prolific crowd-sourcing platform. Each experiment has two variants. The first variant involves showing users a set of context points labeled with if their uncertainty surpasses our predefined threshold. We then ask them to predict if new test points will be certain or uncertain to the model. The second variant contains the same labeled context points and test datapoints. However, together with uncertain context points, users are shown CLUEs of how the input features can be changed such that the model's uncertainty falls below the rejection threshold. The users are then asked to decide if unseen points' predictions will be certain or not. If CLUE works as intended, users who take the second test variant should be able to identify new points on which the model will be uncertain more precisely.

Table 1: The "Surveyed" column denotes who participated: either Prolific users from the US, graduate students with machine learning expertise, or Prolific users holding at least a Bachelor's. "Variant" denotes if the surveyed group was just shown uncertainty labels or labels and CLUEs. "Sample Size" denotes how many people received this variant. The next two columns contain the proportion of correct answers when identifying epistemic or aleatoric uncertainty for LSAT, respectively. The following two columns report the same but for COMPAS. The number in the column heading denotes the number of questions asked. The final column denotes total proportion of correct answers.

Surveyed	**Variant**	**Sample Size**	**LSAT Ep. (6)**	**LSAT Al. (7)**	**COMPAS Ep. (6)**	**COMPAS Al. (5)**	**Total (24)**
Prolific	Unc.	10	0.50	0.40	0.53	0.67	0.54
Students	Unc.	8	0.65	0.58	0.56	0.66	0.61
Prolific	CLUE	10	0.60	0.70	0.60	0.40	0.59
Prolific (BS+)	CLUE	9	0.61	0.68	0.54	0.69	0.63
Students	CLUE	7	0.50	0.8	0.67	0.71	0.67

Humans struggle to make decisions that require considering more than 3-5 entities jointly (Cowan, 2010). The LSAT and COMPAS datasets have 4 and 7 input features respectively. Additionally, in our test, users have to identify trends based on between 4 and 6 context points. These characteristics make our test very difficult. In Table 1, we report the results obtained by subgroup of users and task. Users making predictions without CLUEs tend to perform similarly to random guessing. When providing CLUEs to users, especially to those with machine learning knowledge, we find an increase in their ability to identify uncertain samples. In appendix B, we discuss a similar study on MNIST.

5 Conclusion

Uncertainty estimates can help stakeholders know when to place trust in a model's output. In this work, we have introduced CLUE, an approach to give insight into which features are responsible for a model's uncertainty. Experimentally, we find that CLUE produces in-distribution explanations, which effectively trade-off the changes made to inputs and the amount of uncertainty explained away. Our human study reveals that, even after only having been shown a very limited number of samples, CLUEs help users understand the sources of a model's uncertainty.

Acknowledgements

JA acknowledges support from Microsoft Research through its PhD Scholarship Programme. AW acknowledges support from the David MacKay Newton research fellowship at Darwin College, The Alan Turing Institute under EPSRC grant EP/N510129/1 & TU/B/000074, and the Leverhulme Trust via the Leverhulme Centre for the Future of Intelligence (CFI). UB acknowledges support from DeepMind and the Leverhulme Trust via CFI and from the Partnership on AI.

References

Yoshua Bengio. Estimating or propagating gradients through stochastic neurons. *CoRR*, abs/1305.2982, 2013. URL `http://arxiv.org/abs/1305.2982`.

Nelson Cowan. The magical mystery four: How is working memory capacity limited, and why? *Current directions in psychological science*, 19(1):51–57, Feb 2010. ISSN 0963-7214. doi: 10.1177/0963721409359277.

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

Stefan Depeweg, José Miguel Hernández-Lobato, Steffen Udluft, and Thomas A. Runkler. Sensitivity analysis for predictive uncertainty. In *ESANN*, 2017.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

José Miguel Hernández-Lobato and Ryan P. Adams. 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, pp. 1861–1869. JMLR.org, 2015.

David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472, May 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.3.448.

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

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4134–4142. Curran Associates, Inc., 2016.

A Implementation Details

We minimize (2) with the Adam optimizer, running it for a minimum of 5 iterations and a maximum of 30 iterations with a learning rate of 0.1. If the decrease in $\mathcal{L}(\mathbf{z})$ is smaller than $\mathcal{L}(\mathbf{z}_0)/100$ for 3 consecutive iterations, we apply early stopping. Since the prior on $\mathbf{z}$ constrains VAE latent spaces to the same range of values for all datasets and we normalize inputs to be zero mean and unit variance, we find that these hyperparameter settings work well across tasks. In all of the experiments shown in the main text, λ_y is set to 0. Because the scale of H varies with the dimensionality of the targets and number of train samples, and the scale of d varies with the input dimensionality, λ_x is selected individually for each dataset. Our choices are $25/nd$ for MNIST, $2/nd$ for COMPAS and $1.5/nd$ for LSAT. Here, nd refers to the number of input features of each dataset.

We model continuous targets with Gaussian distributions. We use BNNs to parametrize their means and standard deviations. Discrete targets are modeled with categorical distributions, which are also parametrized 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 LSAT and COMPAS datasets, 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 optimization 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 practice. For MNIST, we model pixels as the probabilities of Bernoulli distributions. We feed these probabilities directly into our BNNs and DGMs. We use 20-dimensional latent spaces for MNIST and 8-dimensional latent spaces for COMPAS and LSAT.

We normalize 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 under CLUE's distance term in (2). For MNIST, this normalization is applied to whole images instead of individual pixels. Categorical variables are not normalized. 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.

We use fixed-width fully connected architectures for all BNNs and tabular data VAEs. We use skip connections and batch normalization at every hidden layer. The VAE used for MNIST uses convolutional ResNets (He et al., 2016) for both its encoder and decoder. Network architecture parameters are given in Table 2.

Table 2: Hyperparameters of the NN architectures used with each of the datasets under consideration. The VAE encoder and decoder used for MNIST are formed of bottleneck residual blocks, as described by He et al. (2016). Their depth refers to the number of blocks.

Dataset	BNN Depth	BNN Width	VAE Depth	VAE Width
MNIST	2	1200	6	-
LSAT	2	200	3	300
COMPAS	2	200	3	300

B Further Details on our User Studies and MNIST Study

In order to validate CLUE on image data, we create a modified MNIST dataset with clear failure modes for our users to identify. We first discard all classes except four, seven and nine. We then manually identify 40 sevens from the training set which have dashes crossing their stems. Using K nearest neighbors, we identify the 12 sevens closest to each of the ones manually selected. We delete these 520 sevens from our dataset. We repeat the same procedure for fours which have a closed, triangle-shaped top. We do not delete any digits from the test set. We train a BNN on this new dataset. Our BNN presents high epistemic uncertainty when tested on dashed sevens and closed fours as a consequence of the sparsity of these features in the train set.

We evaluate the test set of fours, sevens, and nines with our BNN. Datapoints that surpass our uncertainty threshold are selected as candidates to be shown in our user study as uncertain context examples or questions. We show CLUEs for a four and a seven that display the characteristics of interest in Figure 5.

Leveraging the modified MNIST dataset, we ran another human-subject experiment with 10 questions and two variants. The first variant was shown to 5 graduate students with machine learning expertise who only received context points and rejection labels (uncertain or not). This group was able to correctly classify 67% of the unseen test points as high or low uncertainty. The second variant was shown to 5 other graduate students with machine learning expertise who received context points together with CLUEs in cases of high uncertainty. This group was able to reach an accuracy of 88% on unseen test points.

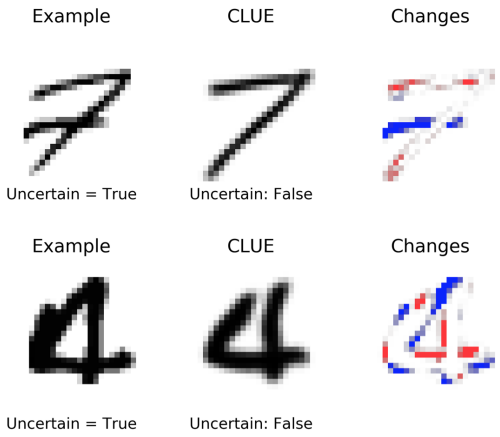


Figure 5: Examples of high uncertainty digits containing characteristics that are uncommon in our modified MNIST dataset. Their corresponding CLUEs and masks are displayed beside them.

In Figure 6, we show an example of the interface presented to users.

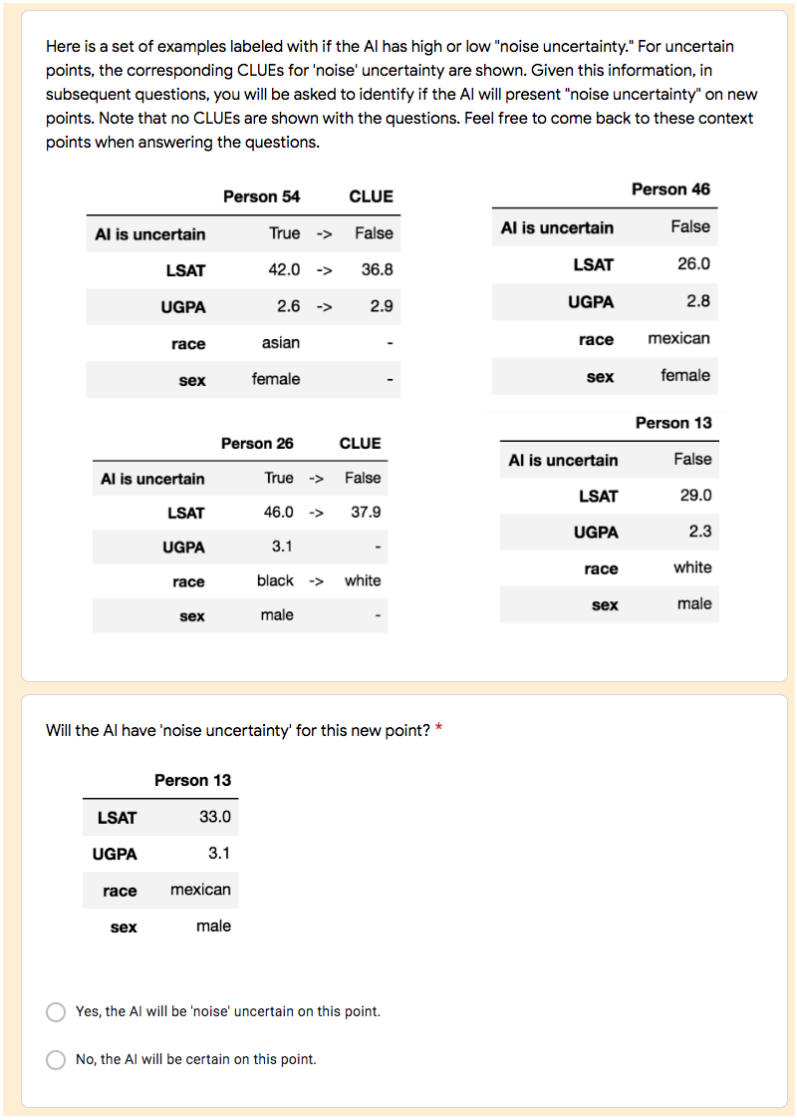


Figure 6: A screenshot of a section from the second test variant for LSAT. The top box shows context examples, with CLUEs. The bottom box shows a question asked to the user.

C MULTIPLICITY OF CLUES

We can exploit the fact that CLUE involves solving a non-convex optimization problem in order to address the existence of multiple plausible counterfactuals. We set our initialisation of $\mathbf{z}$ to $\mathbf{z} = \mu_\phi(\mathbf{z}|\mathbf{x}_0) + \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \sigma_0 \cdot \mathbf{I})$ and perform algorithm 1 multiple times to obtain different CLUEs. Note that this process is parallelizable with batches. We find $\sigma_0 = 0.15$ to give a good trade-off between optimization speed and sample diversity.

In Figure 7, we showcase how different CLUEs for the same original input converge to digits of different classes. Despite this, all explanations resemble the original datapoint being explained. In this way, being exposed to multiple counterfactuals could potentially inform users about similarities of the original input to multiple classes that confuse our model. In Figure 8, we show multiple CLUEs for a single individual from the COMPAS dataset. In this case, uncertainty can be reduced by changing the individual's prior and charge degree or by changing their sex and age range. Making both sets of changes simultaneously also reduces uncertainty.

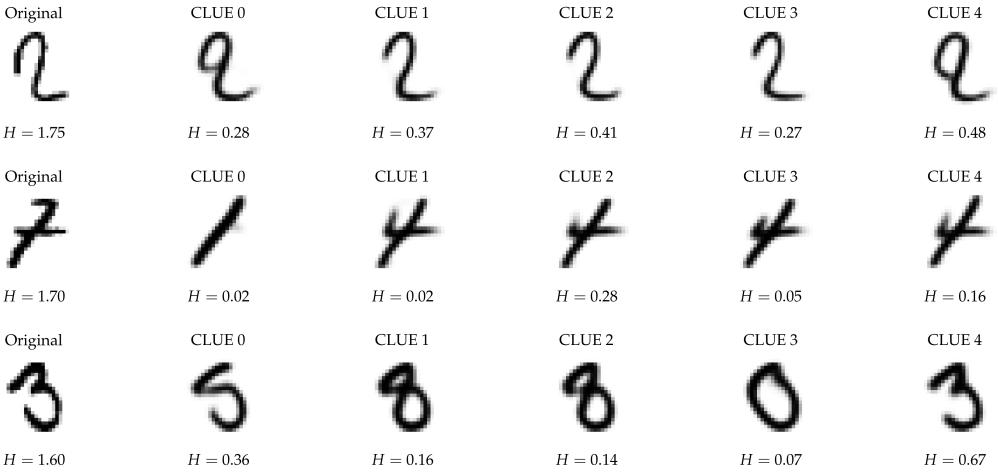


Figure 7: The leftmost column contains three high uncertainty digits from the MNIST test-set. The other five columns contain CLUEs obtained by sampling different $\mathbf{z}$ initialisations.

	Person 0		CLUE
AI is uncertain	True	->	False
age	Greater than 45		-
race	African-American		-
sex	Female		-
charge degree	Felony	->	misdemeanour
recid before	not recid		-
priors count	1.0	->	0.0
days served before	0.0		-

	Person 0		CLUE
AI is uncertain	True	->	False
age	Greater than 45	->	25 - 45
race	African-American		-
sex	Female	->	Male
charge degree	Felony		-
recid before	not recid		-
priors count	1.0		-
days served before	0.0		-

	Person 0		CLUE
AI is uncertain	True	->	False
age	Greater than 45	->	25 - 45
race	African-American		-
sex	Female	->	Male
charge degree	Felony	->	misdemeanour
recid before	not recid		-
priors count	1.0	->	0.0
days served before	0.0		-

	Person 0		CLUE
AI is uncertain	True	->	True
age	Greater than 45	->	25 - 45
race	African-American	->	Asian
sex	Female		-
charge degree	Felony		-
recid before	not recid		-
priors count	1.0		-
days served before	0.0		-

Figure 8: Four possible explanation for a high uncertainty individual from the COMPAS dataset. The "AI is uncertain" row indicates if the configuration displayed would have been rejected by our model due to high uncertainty. Notice how, for the rightmost example, CLUE fails to converge to an input configuration bellow the uncertainty threshold.

D SENSITIVITY ANALYSIS IN HIGH DIMENSIONAL SPACES

In high-dimensional input spaces, often $\nabla_{\mathbf{x}} H$ will not point in the direction of the data manifold. This can result in meaningless explanations. In Figure 9, we show an example where a step in the direction of $-\nabla_{\mathbf{x}} H$ leads to a seemingly noisy input configuration for which the predictive entropy is low. Aggregating these steps for every point in the test set leads to an uncertainty sensitivity analysis explanation that resembles white noise. In Figure 10, we display a CLUE obtained from the same starting digit. The counterfactual explanation corrects the lower portion of the 8.

Despite being able to generate "adversarial examples for uncertainty" in large input spaces, local uncertainty sensitivity analysis performs poorly on our figure of merit f_m as shown in Figure 4. We find that, for each MNIST sample, a different η is required to minimize uncertainty with this approach. A single choice of step size overshoots for some datapoints, increasing their uncertainty, while barely changing it for others.

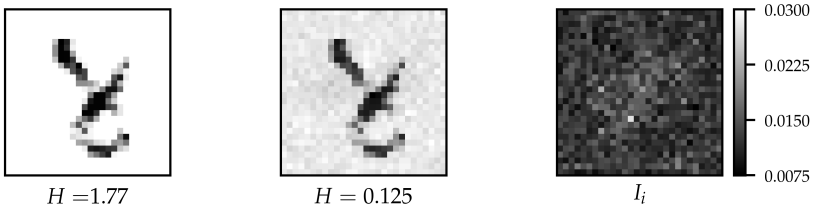


Figure 9: Left: A digit from the MNIST test set with large predictive entropy. Centre: The same digit after a step is taken in the direction of $-\nabla_{\mathbf{x}} H$. Non-zero weight is assigned to pixels which are always zero valued. Right: Uncertainty sensitivity analysis for the MNIST test set.

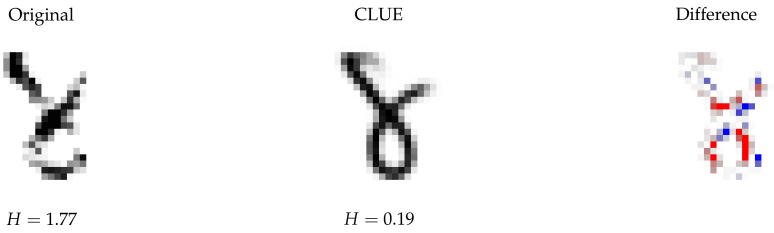


Figure 10: CLUE and difference mask for a high uncertainty test point from the MNIST test-set.