Reproduction and critique of Getting a CLUE

Guilherme Dinis Junior **Maria Movin** **Zhendong Wang**

Abstract

Understanding the confidence of model prediction is particularly critical in life safety scenarios, such as medical diagnosis and autonomous driving. Among different interpretable machine learning approaches, Counterfactual explanations provide a powerful means to interpret example-based machine learning predictions. This project focuses on reproducing the evaluation procedure with repetitive experiments to validate the results reported in the paper concerning counterfactual explanations for uncertainty predictions, titled *Getting a CLUE*. Our contributions are summarized as follows: (1) reviewing and reproducing the evaluation results reported in the paper; (2) extending the results with additional metrics of counterfactual methods; (3) implementing CLUE in a new framework; and (4) providing a critique of CLUE's method and source code.

1 Introduction

Knowing when a model can confidently make a prediction in order to determine if one can trust the outcome is one of the main drives behind research in the area of uncertainty estimation [Carneiro et al., 2020, Tanno et al., 2021]. This capability is particulary critical in life safety scenarios, such as medical diagnosis [Carneiro et al., 2020, Tanno et al., 2021], weather forecasting [Wang et al., 2019, Glitschenski et al., 2020], and autonomous driving [Hoel et al., 2020].

Generally though, uncertainty can be an indicator of issues in machine learning systems, such as the lack of data for a particular cohort. Most work in the area of uncertainty estimation focuses on measuring uncertainty, by computing metrics such as Negative Log Likelihood (NLL) and Brier Score [Brier, 1950]. Despite being reliable, under certain conditions such as having calibrated models [Ashukha et al., 2020], these metrics aren't directly related to the input, and thus provide little introspection as to their interpretation. Counterfactual explanations refer to an example-based approach that can provide actionability in order to change the minimum set of specific features to achieve a desired outcome [Molnar, 2019]. For example, when there is a negative prediction of a loan application, it is important for the applicant or the practitioner to understand what feature changes can be done to transition to a positive outcome.

For the paper under review, titled *Getting a CLUE*, Antoran et al. [2021] sought to combine techniques from counterfactual explanation in order to generate actionable insights into uncertain model predictions. The authors make use of generative models to create counterfactuals that both lie in the data manifold and decrease the level of uncertainty for any distributional model. In addition to this technique to explain away uncertainties, they also provide a framework for evaluating counterfactual explanations of uncertainty. For this project, we have focused on reproducing the evaluation procedure, with multiple runs to validate the results reported in the paper, and implementing CLUE in a new framework (Tensorflow) to generate counterfactuls for MNIST images. For the evaluation, we sought to use the authors' published code to reproduce their results. In addition to this, we also endeavored to report other metrics common to counterfactual method evaluation.

Our contributions are thus: (1) reviewing and reproducing the evaluation results reported in the paper; (2) extending the results with additional metrics of counterfactual explanations; (3) implementing CLUE in a new framework; and (4) providing a critique of CLUE's method and source code.

Project report for DD2412 or FDD3412

We found that, while the counterfactual technique is well motivated, CLUE's evaluation framework is overly extensive and computationally demanding; the results published in the paper are a product of intricately engineered transformations, the instructions for which were limited. Furthermore, the source code proved difficult to reason about, with a tight coupling of steps, and the results rendered in the notebooks lacked a direct mapping to the published results. The source code for our work is available at `https://gits-15.sys.kth.se/mmovin/clue`. It contains the notebooks and scripts we used to run evaluations, and our implementation of CLUE Tensorflow.

2 Related Work

Deep learning models (and some other ML models) are usually considered black-boxes. The underlying structures (i.e. internal weights) are not comprehensible for humans due to their non-linearity and complexity [Vilone and Longo, 2020]. Many practitioners often do not trust the prediction results, especially in specific critical application domains, e.g. business areas and healthcare [Tonekaboni et al., 2019]. However, there has been little work focused on explainability with probabilistic machine learning models.

In order to address this gap, Depeweg et al. [2017] combined sensitivity analysis of neural networks with uncertainty prediction to inspect how much the impact of the gradients is with respect to the input features. They decomposed prediction uncertainty into two components to examine epistemic and aleatoric sensitivity - high epistemic sensitivity suggests better data monitoring to keep the data in the manifold; in contrast, high aleatoric sensitivity shows missing dependencies with unobserved variables. In the experiments, they tested Bayesian neural networks (BNNs) with two hidden layers with samples from 8 UCI datasets for sensitivity analysis. Their empirical evaluation showed that their approach suggested aleatoric and epistemic sensitivity results aligned with human intuitions.

Additionally, Chang et al. [2019] presented a counterfactual solution FIDO by adopting deep generative models in image classification. They applied adaptive Bernoulli dropout and a robust generative in-filling approach to produce FIDO saliency maps compared to previous work. By optimizing two objectives, smallest deletion region (SDR) and smallest supporting region (SSR), the authors adopted three reference heuristics: mean, blur and random; together with local generative models, variantial auto-encoder (VAE) and contextual attention GAN (CA) for generating the reference point in saliency computation.

In Antoran et al. [2021]'s original work, their focus was on investigating counterfactual explanations for uncertainty estimation, specifically for Bayesian neural networks (BNN). They introduced counterfactual latent uncertainty explanations (CLUE), and conducted an intensive experiment with a comparison with previous work from Depeweg et al. [2017] and Chang et al. [2019]. According to their proposed evaluation framework, CLUE outperformed both sensitivity analysis and FIDO; and their user study results showed that CLUE is better at predicting probabilistic models' behaviour.

3 Methods

CLUE [Antoran et al., 2021] can generate a counterfactual which remains in the same data manifold with the original sample, such that the model becomes less uncertain about the prediction. The problem is defined as below: given a deep generative model (DGM): $p_\theta = \int p_\theta(x|z)p(z)dz$ with the latent variable z; and a prediction model $p(y|x, \mathcal{D}) = \int p(y|x, w)p(w|\mathcal{D})dw$, where w denotes the prediction model's weights. The goal of CLUE [Antoran et al., 2021] is to find a counterfactual sample x_{clue} that has lower prediction uncertainty compared to the original sample x_0. For measuring the uncertainty, the loss function is illustrated below:

$$L(z) = \mathcal{H}(y|\mu_\theta(x|z)) + d(\mu_\theta(x|z), x_0), \tag{1}$$

where $\mu_\theta(x|z)$ is the prediction mean, $d(\cdot)$ is the pairwise distance metric and $\mathcal{H}(\cdot)$ represents the differentiable estimate of uncertainty. We would like to find the counterfactual x_{clue} with the following formula:

$$z_{clue} = \arg\min{}_z L(z), \tag{2}$$

$$x_{clue} = \mu_0(x|z_{clue}) \tag{3}$$

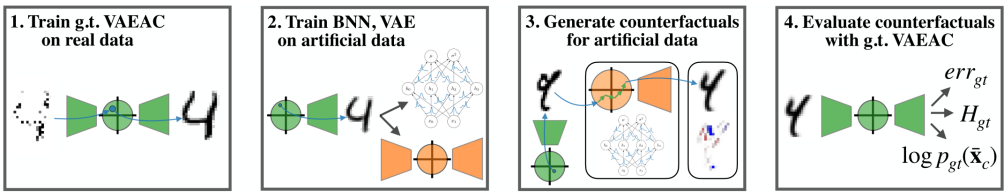


Figure 1: Evaluation pipeline. Figure 6 from Antoran et al. [2021]

.

In the original work, the pairwise distance combines both distance between the original feature and prediction space:

$$d(x, x_0) = \lambda_x d_x(x, x_0) + \lambda_y d_y(f(x), f(x_0)), \tag{4}$$

where $d_x(x, x_0)$ is the L1 distance between x and x_0, and $d_y(f(x), f(x_0))$ can be mean squared errors (for regression) or cross-entropy (for classification). In their original formulation, DGM is represented as a variational auto-encoder (VAE) and Bayesian neural networks (BNN) serves as the prediction model.

Algorithm 1: CLUE algorithm

input : Original sample x_0, distance metric $d(\cdot)$, uncertainty estimate $\mathcal{H}$, DGM decoder $\mu_\theta(x|z)$, DGM encoder $\mu_\phi(z|x_0)$

output : CLUE counterfactual x_{clue}

1 Calculate initial $z \leftarrow \mu_\phi(z|x_0)$

2 **while** *Loss L is not converged* **do**

3 Decode: $x \leftarrow \mu_\theta(x|z)$

4 Calculate $\mathcal{H}(y|x)$ using the prediction model

5 $L = \mathcal{H}(y|x) + d(x, x_0)$

6 Update z with $\bigtriangledown_z L$

7 Decode: $x_{clue} \leftarrow \mu_\theta(x|z)$

8 **return** x_{clue}

Algorithm 1 demonstrates the iterative optimization process of the CLUE method. First, the initial latent state z was calculated using the DGM encoder $\mu_\phi(z|x_0)$. After that, CLUE applies Adam optimization to update the loss metric L iteratively (as shown between Line 3 and Line 6 in Algorithm 1), until it is converged. Finally, we decode using the DGM decoder $\mu_\theta(x|z)$ to generate the counterfactual sample x_{clue}. The evaluation pipeline of CLUE is shown in Figure 1. For adjustments of uncertainty sensitivity analysis and U-FIDO for BNN prediction models, we refer to their original work [Antoran et al., 2021] for more details.

4 Data

We conduct the reproducibility experiment on a total of three different datasets: *LSAT*, *Wine* datasets from the UCI archive, and the *MNIST* dataset. Among these datasets, *LSAT* and *Wine* are tabular data with continuous target class; while *MNIST* is image data that has categorical target. The amounts of training samples vary from 1,438 and 55,000, and for testing they range from 160 and 5,000. More detailed information is shown in Table 1.

5 Experiments and Findings

In our experiments, we first ran the notebooks published by the authors as they are. These made use of several pre-trained models that were also published, including the models required for the evaluation framework - i.e. Arbitrarily-Conditioned Variational Auto-Encoder (VAEAC), Bayesian Neural Network (BNN), and auxiliary Deep Generative Model (DGM). Thus, the only code that ran

Table 1: Description of the datasets used in the reproducible experiment.

Dataset	# training	# testing	Data type	Target
LSAT	17,432	4,358	tabular	continuous
Wine	1,438	160	tabular	continuous
MNIST	50,000	5,000	image	categorical

here was the search for counterfactuals with different hyper-parameters. Second, we added two new metrics to the evaluation process that we used to compare the three methods in question. Third, we re-ran the notebooks without using the pre-trained models, essentially training every model from scratch. To reproduce the results of the paper, we had to interpret and implement the logic described to standardize the metrics. Finally, we implemented the CLUE algorithm from scratch in Tensorflow and used it to search for MNIST CLUEs. In the next subsections, we described the steps in our experiments in detail.

5.1 Procedure

To reproduce the results in the paper, we ran the code provided by the authors. The authors published an archive with approximately 4GB in size, containing notebooks and pre-trained models. There are two notebooks we focused on: one containing the code to run CLUE on tabular data and another specifically for MNIST. Both contain fairly similar logic, with the main distinctions being around how the input shapes are handled and metrics computed. We ran both notebooks on the same machine - a 128 core instance, with 250GB of RAM, and an NVIDIA TITAN RTX GPU with 4608 cores and 24GB of RAM. We used jupyter in a python 2.7 environment, along with other python packages specified as dependencies by the authors on their published repository. To implement CLUE, we used Tensorflow instead of pytorch, which was the framework of choice for the authors, and ran training on the same machine.

5.1.1 Using pre-trained models and re-training

The authors set up the experiments such that each counterfactual search was exclusively focused on reducing either one of the uncertainties: epistemic or aleatoric. They did this by using a weight variable, which they set to zero or one, depending on which uncertainty they want to explain away. Hence, all results have two variants - epistemic and aleatoric. Having executed the notebooks, we inspected the graphs generated, which were of two types. One comparing both uncertainty changes against the changes in X, and another comparing them against changes in $logP(x)$. These first runs gave us insights into aspects such as the actual magnitude of the metrics computed, since these are scaled in the paper, the loss, and duration of execution. For some datasets, such as MNIST, the runtime was relatively high, with each run taking upwards of 5-7 hours on our computational environment, so we modified the code to export metrics to have them in hand for manipulation without having to re-run the notebooks. In order to compare them to the reference paper, we scaled the metrics. To train new models, we disabled a flag to load pre-trained models and followed the same procedure, exporting metrics.

5.1.2 Adding new Evaluation Metrics

The authors published the following metrics: change in X ($||\Delta x||$), the uncertainty (ΔH_{gt} and Δerr), and relevance ($\Delta logp_{gt} = min(0, logp_{gt}(\bar{x_c}) - logp_{gt}(\bar{x_0}))$.

We added two new metrics to the code: *sparsity* and *changes in y*. Sparsity is simply a measure of how many features in the input x have changed from the given instance x_0 to the counterfactual x_c. While $||\Delta x||$ gives a magnitude of changes, *sparsity* only tells us how many changes were done, relatively speaking. The second metric, *changes in y*, measures the magnitude of change in the case of regression, $|y(x_c) - y(x_0)|$, and the changed class for classification, $I_{y(x)_c = y(x_0)}$. Both of these are commonly used metrics in the evaluation of counterfactual explanations.

Dataset	Method	KP_{ref}	KP_{pret}	$\|\|\Delta x\|\|_1$	ΔH	KP_{train}	$\|\|\Delta x\|\|_1$	ΔH
LSAT	CLUE	0.64	0.45 (0.00)	0.49 (0.04)	0.39 (0.04)	0.49 (0.02)	0.57 (0.01)	0.40 (0.00)
	FIDO	0.51	0.29 (0.00)	0.48 (0.00)	0.23 (0.00)	0.15 (0.01)	0.32 (0.01)	0.21 (0.00)
	Sense	0.67	0.46 (0.00)	0.00 (0.00)	0.68 (0.00)	0.29 (0.04)	0.16 (0.05)	0.43 (0.09)
Wine	CLUE	0.14	0.32 (0.00)	0.54 (0.00)	0.18 (0.00)	0.29 (0.01)	0.43 (0.02)	0.28 (0.00)
	FIDO	0.02	0.45 (0.00)	0.17 (0.00)	0.65 (0.00)	0.23 (0.01)	0.19 (0.01)	0.42 (0.03)
	Sense	0.03	0.26 (0.00)	0.35 (0.00)	0.37 (0.00)	0.32 (0.00)	0.30 (0.00)	0.48 (0.00)
MNIST	CLUE	0.27	0.25 (0.00)	0.22 (0.00)	0.13 (0.00)	0.26 (0.00)	0.24 (0.00)	0.10 (0.00)
	FIDO	0.50	0.51 (0.00)	0.08 (0.00)	0.50 (0.00)	0.50 (0.00)	0.08 (0.00)	0.50 (0.00)
	Sense	0.68	0.68 (0.00)	0.01 (0.00)	0.67 (0.00)	0.66 (0.00)	0.01 (0.00)	0.66 (0.00)

Table 2: Results for counterfactual explanations reducing aleatoric uncertainties. Knee Point values (KP) from original article (KP_{ref}, from experiments with pretrained models (KP_{pret}), and from experiments where everything is retrained (KP_{train}). All values presented are mean (variance) values after three runs

5.1.3 Standardizing Results

In the paper, the authors briefly mention scaling the results according to a heuristic, so that the values would range of $[0, \frac{1}{\sqrt{2}}]$. They explicitly state using the max for each measurement between the results for U-FIDO and CLUE. With interpreted this logic as follows:

- Uncertainty (ΔH_{gt} and Δerr): first, find the maximum value between U-FIDO and CLUE for each dataset across hyper-parameter runs. Then, multiply this value by $\sqrt{2}$, and use the result as a denominator for the values (note that uncertainty here is an amount that has been explained away, and hence assumed to range from zero to it's total value). Third, reverse the range so that zero becomes the best result, by subtracting the metric from the upper bound of $\frac{1}{\sqrt{2}}$. For the resulting metric, lower is better.
- Changes in x (Δx): similar to uncertainty, we take the maximum value between U-FIDO and CLUE for each dataset across hyper-parameter runs, and multiply it by $\sqrt{2}$, and use that as a denominator. The only distinction is that we don't reverse the range. For the resulting metric, lower is better.
- Relevance ($\Delta logp_{gt} = min(0, logp_{gt}(\bar{x_c}) - logp_{gt}(\bar{x_0})$): here we introduced code to compute the value using the log probabilities of the BNN with every hyper-parameter configuration, for both the counterfactual example $\bar{x_c}$ and it's original instance $\bar{x_0}$. Upon subtracting the two, we applied the minimum operating to replace any positive values with zero (i.e. where the log probability had increase, the change is considered neutral. Finally, we simply reverse the sign, so that the lowest value is zero; here too, lower is better.

For the two metrics we introduce, *sparsity* and *changes in y*, we apply the same standardization logic, where we multiply the maximum between U-FIDO and CLUE by the $\sqrt{2}$, and use the value as denominator. No further processing is done, and for both metrics, lower is better. At this point it's worth noting that, with the exception of uncertainty, *lower* here is only a theoretically desirable property, but perhaps incomplete. For instance, fewer changes to the instance x is theoretically desirable, but not every change is equal to another. And, as pointed out by the authors, some changes lead to instances further away from the data manifold.

The results of the pre-trained and newly trained models are reported in tables 1-4. For each result, we report the mean and variance of three runs.

5.1.4 Implementing CLUE and searching for MNIST counterfactuals

To be able to validate if the authors had provided enough information about the method CLUE we decided to re-implement the method. We limited the implementation to only the main CLUE algorithm and not the evaluation framework, due to the complexity of the evaluation framework. To be able to assess the implemented method we decided to run it on MNIST dataset and provide examples of CLUEs for some MNIST images.

The implementation was done in Tensorflow Keras and included training a BNN, a Variational Auto-Encoder (VAE) and the implementation of the CLUE algorithm. Since the BNN is the black-box CLUE wants to explain and thus not really part of the algorithm, we decided to go with an implementation of a BNN that was available online (Kang [2021]). For the VAE and for the CLUE algorithm we decided to implement it as described in the paper. Overall was the implementation details in the paper sufficient to reimplement the algorithm, with some cases that were unclear. One unclear hyperparameter was the weight for the part of the loss that captures reduction of uncertainty, by consolidating the authors code we found that it is set to 1 for either epistemic or aleatoric uncertainty, in our runs in this experiment we added weight of 1 for both parts since we wanted to evaluate the CLUEs on both aleatoric and epistemic uncertainty. We ran two experiments with our own implementation: (1) one to generate CLUEs for the test images that the BNN was *most* uncertain and (2) one for test images the BNN was *least* uncertain about. We also did a small ablation study, where we compared the CLUEs to a reconstructed original image where the original were run through the VAE's encoder and decoder. The idea of this experiment was to understand how much of the uncertainty that was already removed by reconstructing the image with the VAE. Since the VAE will ensure the CLUE is on the data manifold, we hypothesized that some uncertainty already is removed by this step of the CLUE method.

5.2 Results

We have four tables with our results: tables 3 and 2 have results for the knee points computed based on $||\Delta\bar{x}||$ as a reference point, and the epistemic and aleatoric uncertainties, Δerr and ΔH_{gt} respectively; tables 4 and 5 have results for knee points computed based on $\Delta logp_{gt}$ as reference point, and the epistemic and aleatoric uncertainties, Δerr and ΔH_{gt} respectively. We focus on the sample results in table 2, and other tables can be found in the appendix.

Some of the results we got from running the notebooks using the provided pre-trained models are equivalent in ordering to the ones reported in the paper, but very few values matched it. In our runs, we obtained almost the exact same results each time - though there were minor differences, so the variance rounded to 2 decimals points is mostly zero while at 4 or more decimal points, we see a difference. In all three datasets, CLUE had the highest amount of alatoric uncertainty reduced. For epistemic uncertainty, CLUE had the highest reduction for Wine and MNIST, while FIDO had the best result for LSAT. In the case of epistemic uncertainty, for instance (table 3), CLUE tied in with Sensitivity on the knee point for LSAT but had the best knee point for Wine and MNIST. Comparing $||\Delta x||_1$ in tables 2 and 5, CLUE made the most changes to the input in every case. In table 6 (Appendix) we see a similar pattern for both *changes of y* and *sparsity*, with CLUE having the highest or second highest value. These patterns were observed for both the pre-trained and newly trained models; however, the magnitude of the values differed between - note that the metrics are scaled.

When it comes to counterfactual generation using our implementation of CLUE, we focused on MNIST. Some samples of counterfactuals for highly uncertain and certain predictions from a trained BNN are in figures 2 and 3 (Appendix), respectively. Most counterfactuals generated are fairly easily distinguishable digits, from the samples draw. We see that in some cases, two similar images can have different counterfactuals, e.g. first two digits in figure 2. Examples with high uncertainty can also undergo significant changes, as seen in the last image in figure 2. The figures 2 and 3 also includes corresponding reconstructed image. Interestingly we see that the CLUEs are very similar to the reconstructed images and that ΔH_{gt} is sometimes larger for the reconstructed images compared to the CLUEs. We can also see that CLUE generates counterfactual explanations that belong to another class compared to the original.

In figures 4 and 5 (Appendix), we have CLUE, U-FIDO, and Sensitivity's uncertainties plotted against $||\Delta x||$ and $\Delta logp(x)$, respectively. CLUE exhibits an erratic pattern on the Wine dataset. On this same dataset, Sensitivity reduces both uncertainties as both $||\Delta x||$ and $||\Delta logp(x)$ increase. For MNIST, CLUE reduces both uncertainties as $||\Delta x||$ decreases and $\Delta logp(x)$ increases, while U-FIDO is consistent in its reduction of both uncertainties as the two variables increase.

5.3 Discussion

In measuring sparsity for CLUE in MNIST, we observed a high value indicating many pixels undergo a change, but from visual inspection, we noticed little noise in the output - i.e. the changes yield meaningful images for human perception. Curiously, with the pre-trained models, CLUE had relatively high changes in the output value, including class changes. This is possibly due to the fact that the loss function disregarded the *change in y* measure. In future experiments, it would be interesting to consider how this constraint affects CLUE's results.

Another idea for future experiments, inspired by our finding that the reconstructed MNIST images sometimes explain away more uncertainty compared to CLUE, would be to add the VAE reconstruction as a baseline to all the experiments. This would allow us to understand how much of the uncertainty reduction that comes from the sampling in the latent space versus the search with gradient decent.

It is worth discussing the interpretation of the metrics, particularly post standardization. Scaling values to the same range can provide a fair way to compare them. However, another approach could have been to measure changes relative to the baseline, e.g. by what percentage did aleatoric distance reduce? The standardization seems to have been mostly driven by the use of euclidean distance to compute the knee points. Bar the computation, the plots were limited in conveying the best result without the knee points.

The x-axis for the $||\Delta x||$ plots given an interpretation of how much a *change in x* affects uncertainty for all three methods. Surprisingly, it is not the case that a higher degree of freedom in changing the input gives lower uncertainty predictions for any of the three methods across all datasets. CLUE's erratic pattern on wine could have been an artifact of the relatively low number of samples compared to the other datasets, and even for the other two datasets, simplistic patterns are hard to conclude from the plots. Furthermore, the choice of metrics here is limited, since *change in x* have no qualitative measures or constraints, when in practice, some features can be more malleable than others to make an acceptable counterfactual.

6 Challenges

6.1 Conceptual Complexity

There are two problems the paper tries to address: (1) generating counterfactual explanations of uncertainty and (2) evaluating them. The introduction of the evaluation framework adds significant complexity, largely in part, in our perception, due to the number of components involved in its steps. This on its own isn't problematic; however, the limited amount of text dedicated to describing the individual steps in the framework makes it challenging to reflect over the theoretical contribution of each step. This is exacerbated by the various transformations applied to the metrics - the detailing of which is also fairly limited - reported in the tables and graphs. An improvement of the paper would be to have a mode in-depth description of the evaluation components.

6.2 Computational Complexity

According the method proposed authors, evaluating counterfactual explanations of uncertainty requires an additional three networks: a g.t. VAEAC, a BNN and an auxiliary DGM. Granted, each of these is to be trained on the dataset in question - hence, different methods for counterfactual explanation of uncertainty for a dataset would leverage the same evaluation models. While we grasp the described benefits of the method, including the protection against adversarial training of bayesian models under evaluation, the added complexity can translate into significant cost for moderately to large datasets - even if the goal is to only explain predictions that are most uncertain. For MNIST, as an example, running CLUE with 30 different hyperparameters to explain aleatoric and epistemic uncertainties took close to 60min for a total of 1700 samples on a machine with a TITAN RTX GPU (4608 cores, 24GB in memory) on and 128 CPU cores. On a more modest device, one can argue the computation is prohibitively expensive for practical applications.

6.3 Reproducing Results

Even with the authors code, the experiments were difficult to reproduce. The evaluation method suggested in the paper contains several different steps, and it is difficult to know where the difference in our results compared to the authors' results stems from. In addition, the suggested evaluation method does not include all details on how to get the final results, not in the paper and not in the code. The article is written in a ambiguous way, so that it is difficult to understand what the tables actually shows. For example, Table 2 in the original article has the title $\Delta \log p_{gt} vs ||\Delta \bar{x}||$, but when reading in the body text it seems like the results show $\Delta H_{get} vs \Delta \log p_{gt}$ - which is also aligned with plots found in the notebooks.

Another challenge, this one pertaining to the code more than any other aspect, was the tight coupling of components and logic. For instance, the authors set seeds insides functions of classes across several modules. This makes is very difficult to both define one seed for an entire run and to prevent any seeds from pre-determining the outcome. On top of this, most the code lacked any documentation for the functions, variables, and modules. There was heavy use of repetition on, for instance, computing aleatoric and epistemic uncertainty that could easily been abstracted for reuse and readability.

Finally, every model used was trained under specific hyper-parameters, e.g. for one of the MNIST BNN the authors used 15 burn-in steps, 150 snapshots, 45 resample prior iterations, etc. The total number of parameters choices is thus fairly high. It would be interesting to study the same problem using, perhaps, simpler to implement techniques, such as deep ensembles [Lakshminarayanan et al., 2017] instead of a BNN - with the aim of not only reducing the hyper-parameter search space, but making the method more accessible and practical to use.

6.4 Computational Frameworks

One of our first blockers lied in having the environment required to run the code provided by the authors up and running. The authors published code that is compatible python 2.7, which is as of this writing deprecated (and was at the time the code was published on Github). Fortunately, they did provide a list of dependencies needed. The problem lied in the choice of computational framework used, pytorch. Two out of three members of our group are more familar with tensorflow, and this gave us the lens through which we evaluate the benefits and drawbacks of pytorch.

The framework pytorch seems to provide low level APIs which can be very beneficial for research. However, there are some aspects of the framework that make it particularly challenging for research dissemination and reproducibility. For one, in using pytorch, one makes explicit transfers of data buffers between devices, i.e. CPU and GPU. Hence, transferring code from one machine to another (without a GPU) requires code changes or extensive 'if...else' style conditioning, which makes readability harder. Second, a model that was trained on a GPU, using CUDA, can only be loaded on a device with a GPU and CUDA installed. This property severely constrains transfer learning, and made it impossible for us to experiment with CLUE's pre-trained models without having a GPU from a specific vendor in hand. It would be more beneficial for the research community if pytorch models were exported in framework agnostic formats or at least whilst being device independent with pytorch, so that other researchers and practitioners can load and use them in their own experiments without need to replicate the hardware conditions in which the model was trained.

7 Conclusion

In this work we found that, while it was possible to implement the CLUE algorithm given the information in the paper, it was difficult to reproduce the results due to several reasons: overly complex evaluation strategy, an unclear description the standardization of results and code written with deprecated dependencies. The additional metrics show some draw backs of CLUE, namely that it changes many features and thus some CLUEs unfavorably in terms of actionanability, and that it is changing the prediction of the BNN to a greater extent compared to the baseline methods. In addition, our ablation study shed an important light on the VAE used in the CLUE method, and our results suggest that much of the benefit from CLUE comes from sampling in the VAE latent space. With these findings in mind, interesting areas for future work would be to add more constraints to the loss function to see if the additional metrics can be improved, and to investigate the use of only VAE reconstruction as a way to create counterfactual explanations that reduces uncertainty.

8 Ethical Consideration

In a thorough review about the role of artificial intelligence (AI) in achieving the Sustainable Development Goals (SGDs), Vinuesa et al. [2020] states that AI can aid 134 targets (79%) across all SDGs. While AI undoubtedly could benefit the SGDs, there is still a risk in blindly trusting the recommendations, insights, or predictions AI provides in real world applications. The field of explainable AI (XAI) focuses on increasing the transparency of AI, with the overall goal to help humans understand, trust, and effectively manage the results of AI. One risk with XAI methods, CLUE included, is that they provide overly simplified explanations that can provide a false sense of security. Many AI models used today are very complex black box models which are still difficult to understand even with methods like CLUE, and it is important to not forget that the explanations we get only can explain a small fraction of the model.

9 Self Assessment

After following the project grading guideline, we find that our project deserves to be graded as a Very Good Project (C or Pass for us PhD students).

We successfully reran the main experiments of the paper and successfully re-implemented the method in a new framework for which it did not have a open-source version. In addition, we added new validation metrics to illustrate other important aspects of counterfactual explanations as well as a small ablation study that sheds important light on the VAE in the CLUE method. To end we also gave a thorough critique about the reproducibility issues found in both the paper and the original code.

References

J. Antoran, U. Bhatt, T. Adel, A. Weller, and J. M. Hernández-Lobato. Getting a {clue}: A method for explaining uncertainty estimates. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=XSLF1XFq5h`.

A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJxI5gHKDr`.

G. W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78: 1–3, 1950.

G. Carneiro, L. Zorron Cheng Tao Pu, R. Singh, and A. Burt. Deep learning uncertainty and confidence calibration for the five-class polyp classification from colonoscopy. *Medical Image Analysis*, 62:101653, May 2020. ISSN 1361-8423. doi: 10.1016/j.media.2020.101653.

C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud. Explaining image classifiers by counterfactual generation. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1MXz20cYQ`.

S. Depeweg, J. M. Hernández-Lobato, S. Udluft, and T. Runkler. Sensitivity analysis for predictive uncertainty in bayesian neural networks, 2017.

I. Glitschenski, W. Schwarting, R. Sahoo, A. Amini, S. Karaman, and D. Rus. Deep orientation uncertainty learning based on a bingham loss. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=ryloogSKDS`.

C.-J. Hoel, T. Tram, and J. Sjöberg. Reinforcement Learning with Uncertainty Estimation for Tactical Decision-Making in Intersections. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7, Sept. 2020. doi: 10.1109/ITSC45102.2020.9294407.

C. Kang. Bayesian Convolutional Neural Network | Chan's Jupyter, 2021. URL `https://goodboychan.github.io/python/coursera/tensorflow_probability/icl/2021/08/26/01-Bayesian-Convolutional-Neural-Network.html`.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

C. Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. 2019.

R. Tanno, D. E. Worrall, E. Kaden, A. Ghosh, F. Grussu, A. Bizzi, S. N. Sotiropoulos, A. Criminisi, and D. C. Alexander. Uncertainty modelling in deep learning for safer neuroimage enhancement: Demonstration in diffusion MRI. *NeuroImage*, 225:117366, Jan. 2021. ISSN 1095-9572. doi: 10.1016/j.neuroimage.2020.117366.

S. Tonekaboni, S. Joshi, M. D. McCradden, and A. Goldenberg. What Clinicians Want: Contextualizing Explainable Machine Learning for Clinical End Use. In *Proceedings of the 4th Machine Learning for Healthcare Conference*, pages 359–380. PMLR, Oct. 2019. ISSN: 2640-3498.

G. Vilone and L. Longo. Explainable artificial intelligence: a systematic review, 2020.

R. Vinuesa, H. Azizpour, I. Leite, M. Balaam, V. Dignum, S. Domisch, A. Felländer, S. D. Langhans, M. Tegmark, and F. Fuso Nerini. The role of artificial intelligence in achieving the Sustainable Development Goals. *Nature Communications 2020 11:1*, 11(1):1–10, jan 2020. ISSN 2041-1723. doi: 10.1038/s41467-019-14108-y. URL https://www.nature.com/articles/s41467-019-14108-y.

B. Wang, J. Lu, Z. Yan, H. Luo, T. Li, Y. Zheng, and G. Zhang. Deep Uncertainty Quantification: A Machine Learning Approach for Weather Forecasting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2087–2095, Anchorage, AK, USA, July 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330704. URL https://doi.org/10.1145/3292500.3330704.

A Appendix

A.1 Couterfactuals and Regenerated Images

The images in figures 3 and 2 were generated by taking an original, building a counterfactual using our implementation of CLUE, and using our trained VAE.

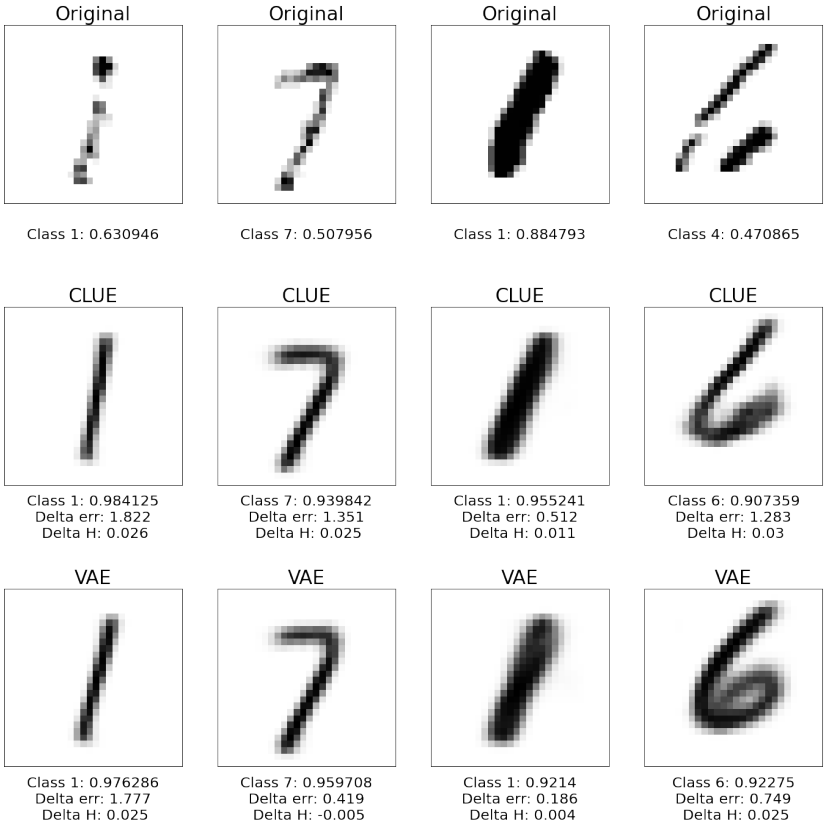


Figure 2: Examples on MNIST CLUEs together with the original image for which the BNN was most uncertain, and with the reconstructed original (VAE)

A.2 Results Tables

In tables 3, 4 and 5, we present metrics generated from multiple runs of CLUE using the authors pre-trained models, newly trained models - i.e. not our implementation.

Table 6, we show our two added metrics for the three methods on all three datasets. These were generated for the pre-trained models only.

A.3 Plots of Results from using Different Hyper-parameters

In figures 5e and 5f, we plot the changes in aleatoric and epistemic uncertainty with respect to two variables: $||\Delta x||$ and $\Delta logp(x)$.

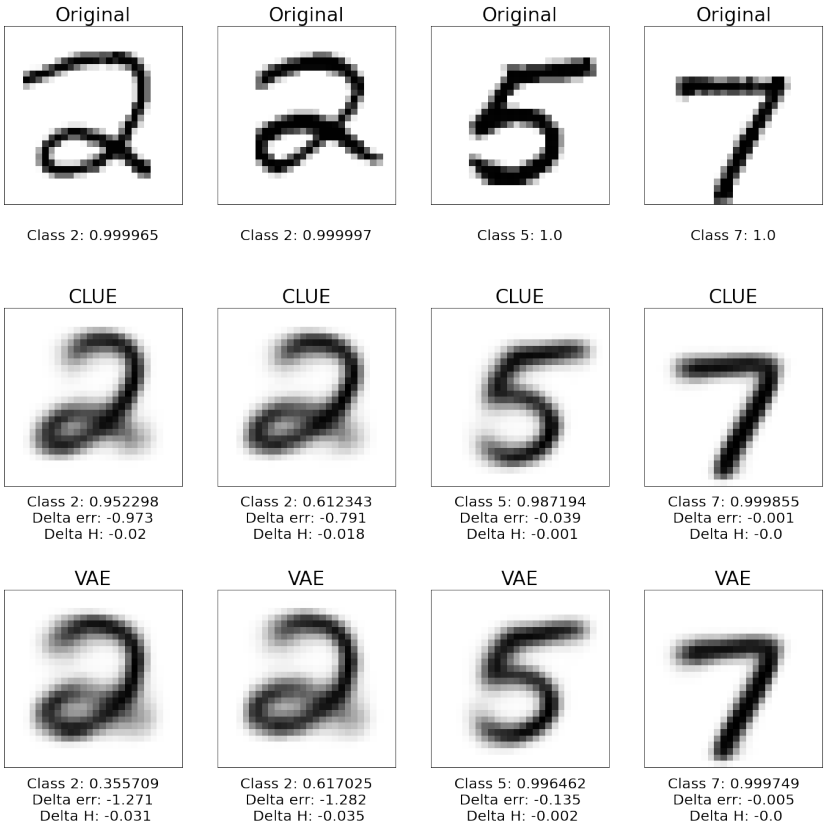


Figure 3: CLUEs explaining the originals for which the BNN was most certain

Dataset	Method	KP_{ref}	KP_{pret}	$\|\|\Delta x\|\|_1$	Δerr	KP_{train}	$\|\|\Delta x\|\|_1$	Δerr
LSAT	CLUE	0.52	0.40 (0.00)	0.56 (0.00)	0.29 (0.00)	0.36 (0.00)	0.51 (0.00)	0.30 (0.00)
	FIDO	0.36	0.49 (0.00)	0.00 (0.00)	0.70 (0.00)	0.49 (0.00)	0.00 (0.00)	0.70 (0.00)
	Sense	0.70	0.40 (0.00)	0.41 (0.00)	0.48 (0.00)	0.37 (0.01)	0.36 (0.02)	0.44 (0.04)
Wine	CLUE	0.01	0.32 (0.00)	0.54 (0.00)	0.18 (0.00)	0.32 (0.00)	0.51 (0.01)	0.19 (0.01)
	FIDO	0.22	0.45 (0.00)	0.17 (0.00)	0.65 (0.00)	0.48 (0.00)	0.01 (0.00)	0.69 (0.00)
	Sense	0.69	0.26 (0.00)	0.35 (0.00)	0.37 (0.00)	0.39 (0.02)	0.15 (0.07)	0.51 (0.11)
MNIST	CLUE	0.26	0.47 (0.00)	0.23 (0.00)	0.40 (0.00)	0.45 (0.00)	0.22 (0.00)	0.39 (0.00)
	FIDO	0.38	0.55 (0.00)	0.08 (0.00)	0.55 (0.00)	0.54 (0.00)	0.09 (0.00)	0.55 (0.00)
	Sense	0.66	0.69 (0.00)	0.01 (0.00)	0.69 (0.00)	0.67 (0.00)	0.01 (0.00)	0.68 (0.00)

Table 3: Results for counterfactual explanations reducing epistemic uncertainties. Knee Point values (KP) from original article (KP_{ref}, from experiments with pretrained models (KP_{pret}), and from experiments where everything is retrained (KP_{train}). All values presented are mean (variance) values after three runs

Dataset	Method	KP_{ref}	KP_{pret}	$\Delta logp_{gt}$	Δerr	KP_{train}	$\Delta logp_{gt}$	Δerr
LSAT	CLUE	0.42	0.25 (0.00)	0.41 (0.00)	0.29 (0.00)	0.22 (0.00)	0.35 (0.01)	0.30 (0.01)
	FIDO	0.00	0.49 (0.00)	0.01 (0.00)	0.70 (0.00)	0.49 (0.00)	0.01 (0.00)	0.70 (0.00)
	Sense	0.70	0.22 (0.00)	0.12 (0.00)	0.45 (0.00)	0.13 (0.00)	0.08 (0.00)	0.34 (0.01)
Wine	CLUE	0.00	0.08 (0.00)	0.28 (0.00)	0.00 (0.00)	0.18 (0.00)	0.38 (0.00)	0.15 (0.01)
	FIDO	0.22	0.42 (0.00)	0.01 (0.00)	0.65 (0.00)	0.49 (0.00)	0.00 (0.00)	0.70 (0.00)
	Sense	0.69	0.32 (0.00)	0.43 (0.00)	0.37 (0.00)	0.19 (0.07)	0.04 (0.00)	0.36 (0.09)
MNIST	CLUE	0.27	0.45 (0.00)	0.24 (0.00)	0.38 (0.00)	0.44 (0.00)	0.26 (0.00)	0.36 (0.00)
	FIDO	0.45	0.60 (0.00)	0.25 (0.00)	0.55 (0.00)	0.58 (0.00)	0.22 (0.00)	0.55 (0.00)
	Sense	0.70	0.70 (0.00)	0.07 (0.00)	0.70 (0.00)	0.70 (0.00)	0.23 (0.00)	0.68 (0.00)

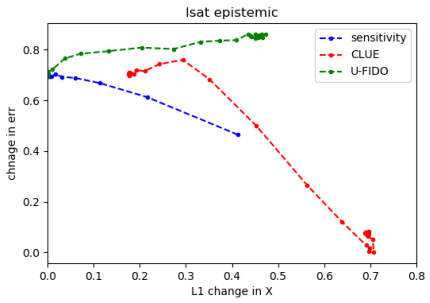
Table 4: $\Delta logp_{get}$ vs Δerr Knee Point (KP) results for counterfactual explanations reducing epistemic uncertainties. Knee Point values (KP) from original article (KP_{ref}, from experiments with pretrained models (KP_{pret}), and from experiments where everything is retrained (KP_{train}). All values presented are mean (variance) values after three runs.

Dataset	Method	KP_{ref}	KP_{pret}	$\Delta logp_{gt}$	ΔH	KP_{train}	$\Delta logp_{gt}$	ΔH
LSAT	CLUE	0.07	0.02 (0.00)	0.00 (0.00)	0.14 (0.00)	0.12 (0.00)	0.13 (0.03)	0.29 (0.00)
	FIDO	0.00	0.02 (0.00)	0.12 (0.00)	0.02 (0.00)	0.01 (0.00)	0.10 (0.00)	0.02 (0.00)
	Sense	0.67	0.63 (0.00)	0.41 (0.00)	0.68 (0.00)	0.33 (0.07)	0.32 (0.00)	0.43 (0.09)
Wine	CLUE	0.00	0.00 (0.00)	0.03 (0.00)	0.00 (0.00)	0.15 (0.00)	0.33 (0.01)	0.18 (0.00)
	FIDO	0.13	0.01 (0.00)	0.01 (0.00)	0.12 (0.00)	0.12 (0.02)	0.03 (0.00)	0.26 (0.08)
	Sense	0.00	0.21 (0.00)	0.28 (0.00)	0.36 (0.00)	0.44 (0.01)	0.20 (0.01)	0.62 (0.00)
MNIST	CLUE	0.15	0.22 (0.0)	0.22 (0.0)	0.0 (0.0)	0.24 (0.01)	0.26 (0.00)	0.05 (0.00)
	FIDO	0.52	0.55 (0.0)	0.24 (0.0)	0.50 (0.0)	0.53 (0.00)	0.20 (0.00)	0.50 (0.00)
	Sense	0.70	0.70 (0.0)	0.04 (0.0)	0.69 (0.0)	0.68 (0.00)	0.14 (0.00)	0.67 (0.00)

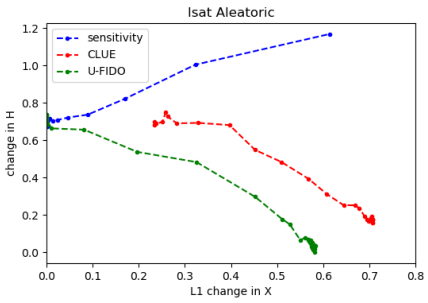
Table 5: $\Delta logp_{get}$ vs ΔH Knee Point (KP) results for counterfactual explanations reducing aleatoric uncertainties. Knee Point values (KP) from original article (KP_{ref}, from experiments with pretrained models (KP_{pret}), and from experiments where everything is retrained (KP_{train}). All values presented are mean (variance) values after three runs

Dataset	Uncertainty type	Method	Validity	$d(f(x) - f(x_{cf}))$	Sparsity ($\|\|\Delta x\|\|_0$)
LSAT	Epistemic	CLUE	-	0.34 (0.00)	0.29 (0.00)
		FIDO	-	0.00 (0.00)	0.00 (0.00)
		Sense	-	0.20 (0.00)	1.00 (0.00)
	Aleatoric	CLUE	-	0.12 (0.00)	0.23 (0.00)
		FIDO	-	0.19 (0.00)	0.09 (0.00)
		Sense	-	0.00 (0.00)	1.00 (0.00)
Wine	Epistemic	CLUE	-	1.04 (0.00)	1.00 (0.00)
		FIDO	-	0.52 (0.00)	0.48 (0.00)
		Sense	-	0.59 (0.00)	1.00 (0.00)
	Aleatoric	CLUE	-	0.35 (0.00)	1.0 (0.00)
		FIDO	-	0.35 (0.00)	0.53 (0.00)
		Sense	-	0.30 (0.00)	0.59 (0.00)
MNIST	Epistemic	CLUE	-	0.41 (0.0)	0.71 (0.0)
		FIDO	-	0.24 (0.0)	0.09 (0.0)
		Sense	-	0.04 (0.0)	0.71 (0.0)
	Aleatoric	CLUE	-	0.38 (0.0)	0.71 (0.0)
		FIDO	-	0.24 (0.0)	0.13 (0.0)
		Sense	-	0.05 (0.0)	0.71 (0.0)

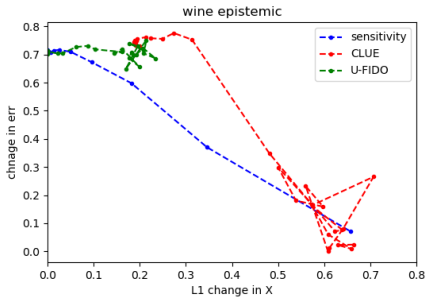
Table 6: Characteristics of counterfactuals for the best set of hyperparameters (decided by knee point). Run on the pretrained models only



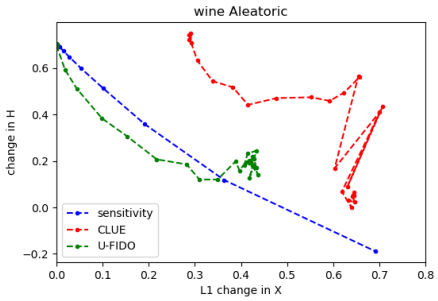
(a) LSAT Epistemic



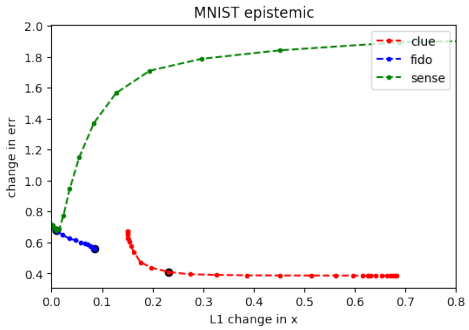
(b) LSAT Aleatoric



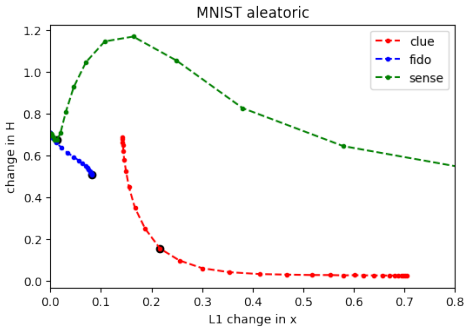
(c) Wine Epistemic



(d) Wine Aleatoric

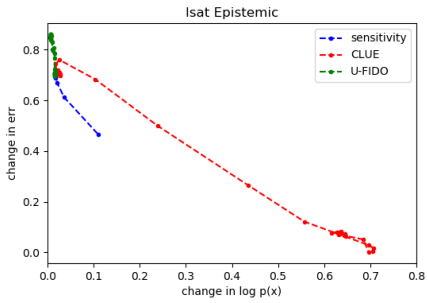


(e) MNIST Epistemic

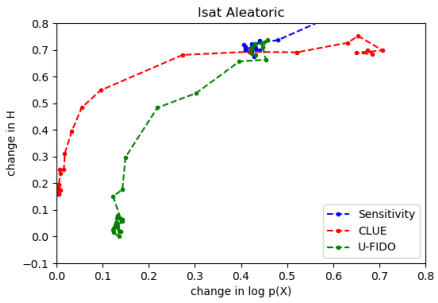


(f) MNIST Aleatoric

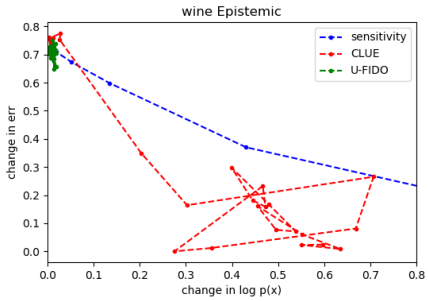
Figure 4: Knee plots for epistemic ($||\Delta x||_1$ vs Δerr) and aleatoric ($||\Delta x||_1$ vs ΔH) counterfactuals for pretrained models.



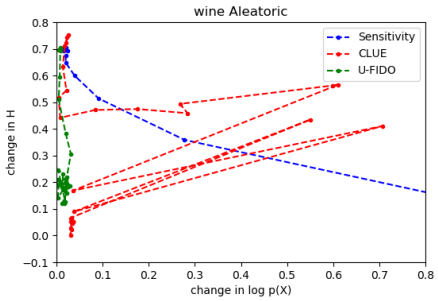
(a) LSAT Epistemic



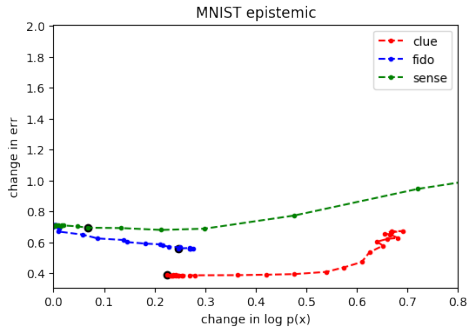
(b) LSAT Aleatoric



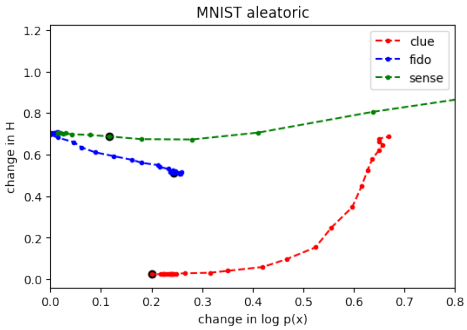
(c) Wine Epistemic



(d) Wine Aleatoric



(e) MNIST Epistemic



(f) MNIST Aleatoric

Figure 5: Knee plots for epistemic ($logp_{gt}$ vs Δerr) and aleatoric ($logp_{gt}$ vs ΔH) counterfactuals for pretrained models