Counterfactuals From the Penultimate Layer - Results Report

Conor Finlay

March 12, 2025

1 Introduction

This document contains the results for some experiments which evaluate the use of Bayesian last-layer models for counterfactual explanations, generated from the penultimate layer representations of the model.

The model architecture remained the same for all experiments, only the Bayesian/non-Bayesian nature of the last-layer model was varied, as well as the training objective of the model. Details of the architecture can be found in Appendix 3.3.

2 Uncertainty-reducing counterfactuals

In this section, we evaluate the performance of each type of model on the task of generating uncertainty-reducing counterfactuals. There are 6 models in total, which are:

- **Bayesian models**
  - **Bayesian-C**: The Bayesian last-layer model trained with a classification-dominated objective.
  - **Bayesian-A**: The Bayesian last-layer model trained with an autoencoder-dominated objective.
  - **Bayesian-J**: The Bayesian last-layer model trained with a joint-objective.
- **Non-Bayesian models**
  - **Deterministic-C**: The deterministic model trained with a classification-dominated objective.
  - **Deterministic-A**: The deterministic model trained with an autoencoder-dominated objective.
  - **Deterministic-J**: The deterministic model trained with a joint-objective.

We identify fifty images from the MNIST test set which are classified with high entropy by the Bayesian-J model. A previous experiment involved using the fifty images each model was most uncertain about, but we found it more useful to compare a consistent set of images across all models.

On these fifty images, the models seek to generate counterfactuals which reduce the uncertainty of the model's prediction. This involves an optimisation within the penultimate layer space, inspired by CLUE[1], which seeks to minimise both the entropy of the model's prediction and the distance between the original and counterfactual penultimate layer representations. This is expressed as:

$$\min_{x'} \left(H(y|x') + \lambda \cdot \|x - x'\|_2 \right) \tag{1}$$

where x is the original image, x' is the counterfactual, y is the model's prediction, $H(y|x')$ is the entropy of the model's prediction, and λ is a hyperparameter which balances the two terms.

In all of our experiments, we set $\lambda = 0.001$.

2.1 Qualitative Analysis

2.2 Entropy Reduction

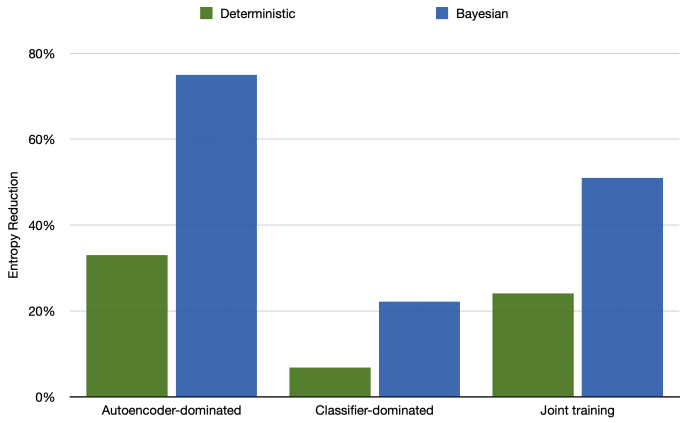


Figure 1: Comparison of the percentage entropy reduction of the models.

2.3 Counterfactual Realism/Latent Space Distance

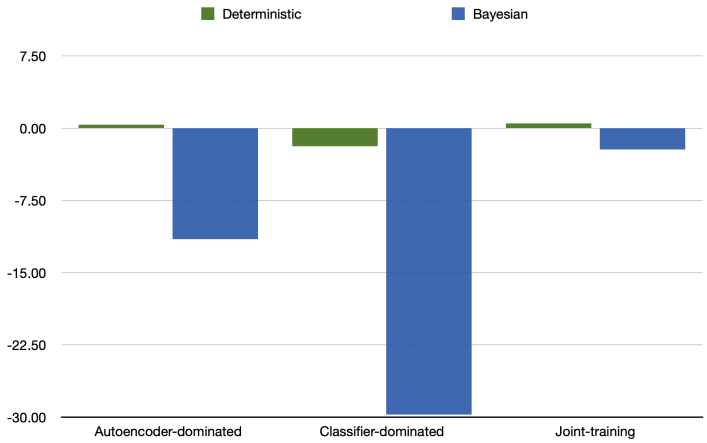


Figure 2: Comparison of the difference in realism between the original and counterfactual images.

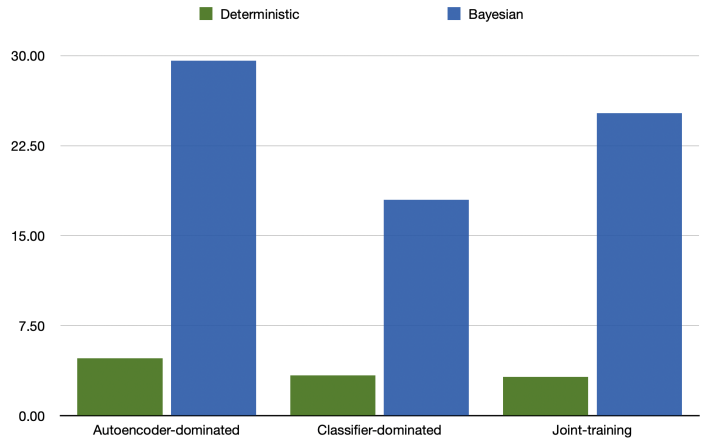


Figure 3: Comparison of the latent space distance between the original and counterfactual images.

2.4 Class Consistency

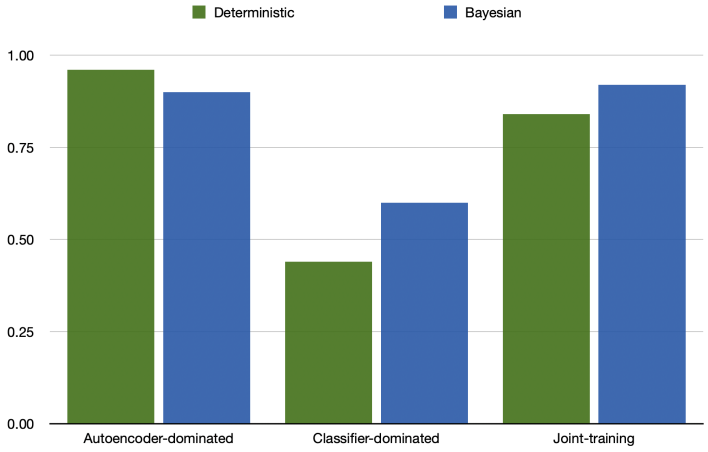


Figure 4: Comparison of the class consistency.

3 Class-changing counterfactuals

In this section, we evaluate the performance of each type of model on the task of generating class-changing counterfactuals. That is, we specify a target class that differs from the original class of an image, and optimisation attempts to find a minimal change (in latent space) which changes the class prediction to the target class.

As a result, the new optimisation problem is:

$$\max_{x'} \left(\hat{y}_{\text{target}}(x') - \lambda \cdot \|x - x'\|_2 \right) \tag{2}$$

where x is the original image, x' is the counterfactual, $\hat{y}_{\text{target}}(x')$ is the softmax prediction confidence of the target class, and λ is a hyperparameter which balances the two terms.

3.1 Qualitative Analysis

3.1.1 Example 1: 9 → 8

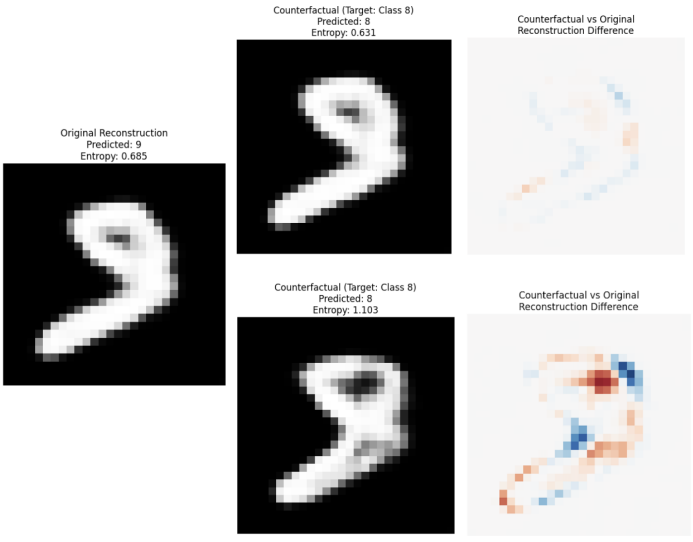
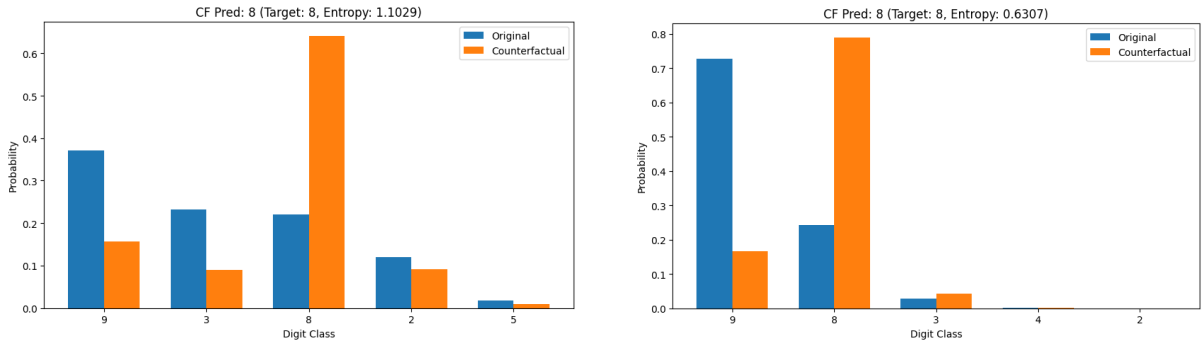


Figure 5: Comparison of the class-changing counterfactuals from 9 to 8 (top: deterministic, bottom: Bayesian).



(a) Bayesian model.

(b) Deterministic model.

Figure 6: Softmax outputs before and after counterfactual optimisation ($9 \rightarrow 8$).

3.2 Target Class Confidence

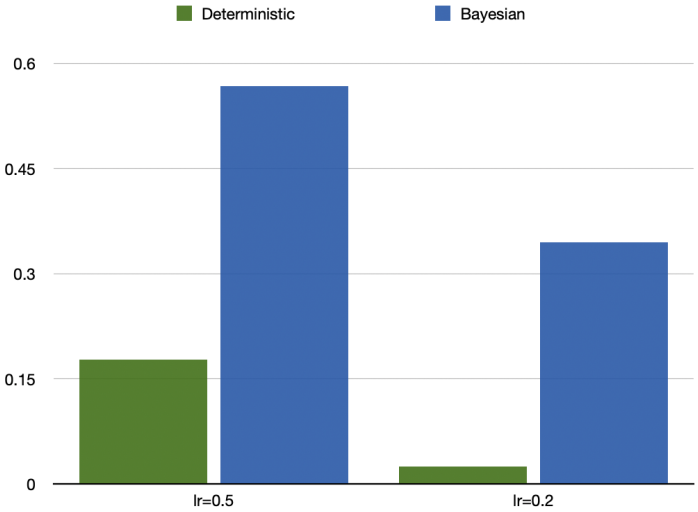


Figure 7: Target class confidence increase for class-changing counterfactuals (reconstructions).

	Learning rate	Distance weight	Steps
Params 1 (aggressive learning rate)	0.5	0.001	200
Params 2 (lower learning rate)	0.2	0.001	200

Table 1: Parameter settings with different learning rates.

	Deterministic	Bayesian
Params 2	0.580	0.974

Table 2: Target class confidence increase in latent space (pre-reconstruction).

Looking at Table 2, we see that the average target class confidence increase during optimisation (before reconstruction) is only 0.58 for deterministic model. In practice, this is not because in most instances the figure is around that value. Instead, in each instance, the optimisation either succeeds, in which case the target class confidence is 1.0, or it fails, in which case the target class confidence is 0.0. As such, the 0.58 figure indicates that the optimisation fairs 42% of the time for a learning rate of 0.2.

Although the learning rate has an effect on the Bayesian model's optimisation, it is more continuous and never results in failed optimisation. We explore this in the following section.

3.3 Optimisation

In Figure 8, we plot the target class confidence during optimisation for the deterministic and Bayesian models. This shows the stark contrast between the counterfactual optimisation between the two models.

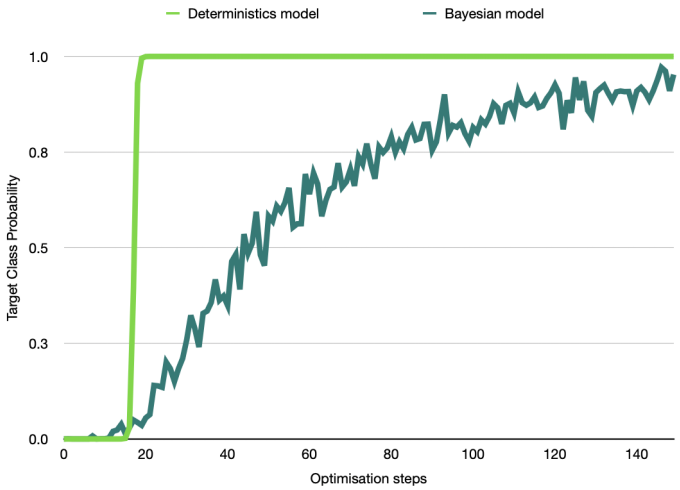


Figure 8: Target class confidence during optimisation.

The optimisation for the Bayesian model makes near instant progress, and proceeds gradually to maximise the target class confidence. On the other hand, the deterministic model's optimisation makes no progress for about 15 steps, before suddenly reaching full target class confidence.

This sheds light on why the counterfactual optimisation for the deterministic model is so unstable. Loosely speaking, the optimisation stumbles around 'blindly' until it comes across a region in latent space for which the target class is predicted, and stays there for the rest of the optimisation. If it never comes across such a region, the optimisation fails. In particular, if there is any significant weight placed on the distance term in the objective function, the optimisation frequently fails as any exploration only serves to increase the value of the loss function.

3.4 Model Architecture

The base architecture follows the ReGene approach ("Classify and Generate"), which consists of two main components:

- **Classifier**: A convolutional neural network with the following structure:
  - Three convolutional blocks, each containing:
    - Convolutional layer
    - Batch normalization
    - ReLU activation
    - Max pooling
  - Flattening layer
  - Linear projection to latent space (dimension 128)
  - Final classification layer
- **Decoder**: A network that reconstructs images from the latent representations:
  - Linear projection from latent space to match flattened dimensions
  - Three transposed convolutional blocks for upsampling
  - Final sigmoid activation for pixel values

The architecture allows for different training objectives:

- **Classification-dominated training**: The classifier portion of the model (and therefore the encoder) is trained to maximise classification accuracy only. The decoder is then trained to reconstruct the training image from the penultimate layer representations of the frozen classifier.
- **Autoencoder-dominated training**: The encoder (classification backbone) and decoder are trained jointly to minimise the reconstruction loss between the input and output images. A single classification head is then trained to maximise classification accuracy on the training set, while taking the embeddings from the frozen encoder.
- **Joint-objective training**: The encoder, decoder and classification head are trained jointly to minimise a loss function which balances the classification accuracy and the reconstruction loss. The encoder balances the classification and reconstruction loss by weighting the two loss terms, while the decoder naturally only minimises the reconstruction loss, and the classification head only the classification loss.

3.5 Bayesian Last Layer (BLL-VI)

For the Bayesian variant, we replace the final classification layer with a Bayesian linear layer:

- **Backbone**: The pretrained convolutional network up to the penultimate layer
- **Bayesian Last Layer**: A variational inference-based Bayesian linear layer with:
    - Gaussian prior on weights (configurable mean and standard deviation)
    - Learnable posterior distributions for weights
    - KL divergence regularization during training

The BLL-VI model provides:

- Uncertainty quantification through multiple forward passes (Monte Carlo sampling)
- Decomposition of uncertainty into aleatoric (data) and epistemic (model) components
- Ability to generate predictions directly from latent representations

During training, the backbone remains frozen while only the Bayesian last layer is optimized, using a loss function that combines cross-entropy and KL divergence terms.

References

[1] Javier Antor'an, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates. *ArXiv*, abs/2006.06848, 2020. URL `https://api.semanticscholar.org/CorpusID:219636236`.