On Generating Plausible Counterfactual and Semi-Factual Explanations for Deep Learning (Supplementary Material)

Anonymous Author(s)
Affiliation
Address
email

S1 Model details

Here we detail all model architectures, their accuracy when appropriate, and the data pre-processing steps. The Adam optimizer was used throughout. All URLs were verified to be available on the 3rd of June 2020. For the reader's convenience, all code and pre-trained models can be found at our repository `https://github.com/after_anon_review`.

S1.1 MNIST CNN

The CNN architecture used to train a single model, which in turn was used for both Expt. 1 and Expt. 2, is detailed in Table 1. This model achieved an accuracy of 99.59% on the 10,000 test images for MNIST. The model was trained for 10 epochs at learning rate (lr) `lr=1e-1`, 10 epochs at `lr=1e-2`, and 10 more epochs at `lr=1e-3`. The data was normalized between -1 and 1, and the batch size was 8.

S1.2 CIFAR-10 CNN

The CNN used in Expt. 1 for CIFAR-10 was a modification of the popular ResNet18 architecture. The code was found at `https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py`. The fourth block of the architecture was removed from our model. In addition, dropout (with p=0.1) was added after every convolutional layer to enable Monte Carlo Dropout, echoing [3]. The accuracy of the network on the 10,000 test images was 92.2%. The model was trained for 30 epochs at `lr=1e-1`, 30 epochs at `lr=1e-2`, and 30 more epochs at `lr=1e-3`. The data was normalized between -1 and 1, and the batch size was 32. Random crops (padding=4), and random horizontal flips were also used.

S1.3 Autoencoders for IM1 and IM2

To train autoencoders (AEs) for these evaluation metrics (IM1 and IM2) to use on CIFAR-10, the architecture found at `https://github.com/jellycsc/PyTorch-CIFAR-10-autoencoder/blob/master/main.py` was used, alongside all its hyperparameter and training choices. To train the AEs for MNIST, the design in Table 2 was used; they were trained for 50 epochs at `lr=1e-2`, and batch size 16.

S1.4 Autoencoder for CEM and Proto-CF

To train the AEs necessary for these techniques, the examples at the documentation `https://docs.seldon.io/projects/alibi/en/stable/` were used.

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

S1.5 GANs

The GANs used for MNIST and CIFAR-10 were found pre-trained at `https://github.com/csinva/gan-vae-pretrained-pytorch`, the reader is referred to this repository for the architecture and training details.

Table 1: The CNN architecture used to train a model on MNIST, this trained model was then used in both experiments.

MNIST CNN	
Layer	Layer Parameters
Conv2d	8 filters, (5x5), (1x1), padding=2
Dropout2d	p=0.1
BatchNorm2d	
ReLU	
Conv2d	8 filters, (5x5), (1x1), padding=2
Dropout2d	p=0.1
BatchNorm2d	
ReLU	
Conv2d	16 filters, (5x5), (2x2), padding=2
Dropout2d	p=0.1
BatchNorm2d	
ReLU	
Conv2d	32 filters, (5x5), (1x1), padding=2
Dropout2d	p=0.1
BatchNorm2d	
ReLU	
Conv2d	64 filters, (5x5), (2x2), padding=2
Dropout2d	p=0.2
BatchNorm2d	
ReLU	
Conv2d	128 filters, (3x3), (1x1), padding=1
GAP	
Linear	128, 10
SoftMax	

S2 Method hyperparameters

Here the details of all hyperparameter choices for the different methods are detailed for both experiments. Namely, the first experiment involving counterfactual explanations, and the second experiment involving semi-factual explanations.

S2.1 Counterfactual experiment

In this experiment five different methods (including our own) were compared.

PIECE The number of epochs for each explanation was 300 in both datasets. The learning rate used was `lr=1e-2` in both datasets. The alpha threshold α was 0.05 in all tests. The Adam optimizer was used.

Min-Edit On MNIST `lr=1e-3` was used for incorrect and close-correct instances, and `lr=1e-2` for correct instances. On CIFAR-10 `lr=1e-1` was used on correct instances, and `lr=1e-2` on incorrect instances. The Adam optimizer was used.

Table 2: The autoencoder architecture used to train the models for IM1 and IM2 on MNIST, subsequently used in Expt. 1 and Expt. 2.

MNIST IM1 and IM2 autoencoder	
Layer	Layer Parameteres
	Encoder
Conv2d	16 filters, (3x3), (1x1), padding=1
ReLU	
Conv2d	16 filters, (2x2), (2x2), padding=0
ReLU	
Conv2d	16 filters, (3x3), (1x1), padding=1
ReLU	
Conv2d	16 filters, (2x2), (2x2), padding=0
	Decoder
Upsample	Scale Factor 2 - Bilinear
ReLU	
Conv2d	16 filters, (3x3), (1x1), padding=1
ReLU	
Upsample	Scale Factor 2 - Bilinear
ReLU	
Conv2d	1 filters, (3x3), (1x1), padding=1
Sigmoid	

C-Min-Edit In MNIST `lr=1e-3` was utilized for incorrect and close-correct instances, and `lr=1e-2` for correct instances. For CIFAR-10 `lr=1e-1` was chosen on correct instances, and `lr=1e-2` on incorrect ones. The lambda parameter λ was set to 0.1 and incrementally increased by 0.1 each epoch in all tests. The distance function $d(.)$ used the L_2 norm, and the Adam optimizer was used.

CEM The default parameters suggested in the official documentation at `https://docs.seldon.io/projects/alibi/en/stable/` were used for both MNIST and CIFAR-10. However, `c_init` was set to 0, and `c_steps` set to 1 on CIFAR-10.

Proto-CF The default parameters suggested in the official documentation at `https://docs.seldon.io/projects/alibi/en/stable/` were used for both MNIST and CIFAR-10. However, `c_init` was again set to 0, and `c_steps` set to 1 on CIFAR-10.

S2.2 Semi-factual experiment

In this experiment three different methods (including our own method PIECE) were compared.

PIECE The number of epochs for each explanation was 300. The learning rate was `lr=1e-2`. The alpha threshold α was 0.05, and the Adam optimizer was used.

Min-Edit The learning rate used was `lr=1e-2` alongside the Adam optimizer.

C-Min-Edit The learning rate used was `lr=1e-2`. The lambda parameter λ was set to 0.1 and incrementally increased by 0.1 each epoch. The distance function $d(.)$ used the L_2 norm, and lastly the Adam optimizer was used.

S3 Machine specifications

Both experiments used the same machine to generate all explanations. The machine was a: MacBook Pro; Processor 2.9 GHz Intel Core i5; Memory 16 GB 2133 MHz LPDDR3; 500GB storage. (n.b., no GPU was used, only a CPU).

However, to train the CNNs and AE models, Google Colab `https://colab.research.google.com/` was utilized alongside its built-in GPU capability.

S4 Additional plots

To quote Section 3 of the main paper:

> 'Interestingly, for all results on MNIST, a plot of the NN-Dist measure against the MC-Mean/MC-STD scores show a significant linear relationship r = -0.8/0.82. So, the more a generated counterfactual is grounded in the training data, the more likely it is to be plausible...'

Here, the plots for these correlations are shown, which demonstrate that there is a strong correlation between plausibility (measured with MC Dropout), and the distance of an explanation's latent representation from the training data, as many have previously eluded to (e.g., see [4, 2, 9, 10]). This discovery lends much credibility to NN-Dist as a valid evaluation metric (when generating explanations with GANs, as is becoming popular [8, 5, 1, 6, 7]) to measure the plausibility of generated counterfactual explanations.

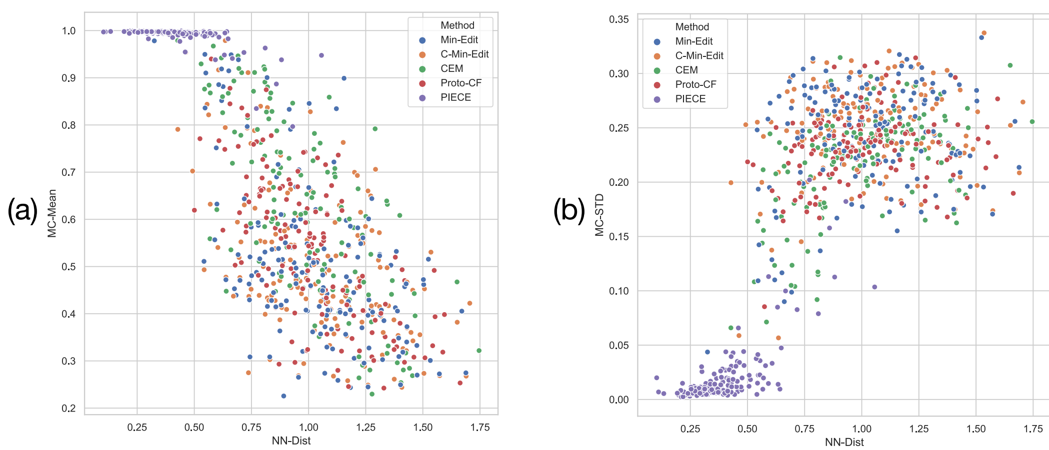

Figure 1: The latent representation x of every explanation image for every method in MNIST is shown; specifically, its NN-Dist plotted against its MC-Mean and MC-STD scores. The results show strong linear correlations between how close an explanation (n.b., explanations generated with a GAN) is to a training instance, and how plausible it is.

References

[1] A. Feghahati, C. R. Shelton, M. J. Pazzani, and K. Tang. Cdeepex: Contrastive deep explanations. 2018.

[2] A.-H. Karimi, G. Barthe, B. Belle, and I. Valera. Model-agnostic counterfactual explanations for consequential decisions. *arXiv preprint arXiv:1905.11190*, 2019.

[3] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.

[4] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *arXiv preprint arXiv:1907.09294*, 2019.

[5] S. Liu, B. Kailkhura, D. Loveland, and H. Yong. Generative counterfactual introspection forexplainable deep learning. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2019.

[6] P. Samangouei, A. Saeedi, L. Nakagawa, and N. Silberman. Explaingan: Model explanation via decision boundary crossing transformations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–681, 2018.

[7] J. Seah, J. Tang, A. Kitchen, and J. Seah. Thinking like a machine—generating visual rationales through latent space optimization. 2018.

[8] S. Singla, B. Pollack, J. Chen, and K. Batmanghelich. Explanation by progressive exaggeration. In *International Conference on Learning Representations*, 2020.

[9] A. Van Looveren and J. Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

[10] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.