DeDUCE: Generating Counterfactual Explanations Efficiently

Benedikt Höltgen* **Lisa Schut** **Jan M. Brauner** **Yarin Gal**
OATML Group
Department of Computer Science
University of Oxford
Oxford, United Kingdom
benedikt.hoeltgen@mailbox.org

Abstract

When an image classifier outputs a wrong class label, it can be helpful to see what changes in the image would lead to a correct classification. This is the aim of algorithms generating counterfactual explanations. However, there is no easily scalable method to generate such counterfactuals. We develop a new algorithm providing counterfactual explanations for large image classifiers trained with spectral normalisation at low computational cost. We empirically compare this algorithm against baselines from the literature; our novel algorithm consistently finds counterfactuals that are much closer to the original inputs. At the same time, the realism of these counterfactuals is comparable to the baselines. The code for all experiments is available at `https://github.com/benedikthoeltgen/DeDUCE`.

1 Introduction

Much of the recent staggering success of machine learning is due to large and complex models whose inner workings are in many cases elusive. The prime examples of such 'black box' models are deep neural networks. Despite the drawbacks that come with a poor understanding of their inner workings, such models are already widely deployed in practice due to their state-of-the-art performance in many areas. But this poor understanding can make it difficult to interpret the decision making process of such models, and thus to identify the problem when a model makes an error.

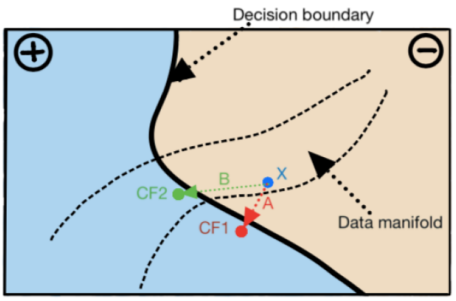

Figure 1: In many use cases, counterfactuals should lie just across the decision boundary as seen from the original x and lie on the data manifold, which is here satisfied by CF2 but not by CF1 (taken from [30]).

To explain erroneous classifications, it can therefore be helpful to investigate the relevant decision boundary. An intuitive way to do so is to look for alternative inputs that are similar but classified differently; this is the realm of counterfactual explanations. For debugging, we usually want to generate realistic counterfactuals that are just across the model's classification decision boundary (fig. 1). We want a counterfactual that stays as close to the original input as possible but results in a correct classification, in order to understand the system's decision making. At the same time, the counterfactual should stay on the manifold of realistic images (CF2 in fig. 1) since we are interested in inputs that can be interpreted. Being a nascent area of research, there

*External collaborator to OATML Group, work done while a master student there.

1st Workshop on eXplainable AI approaches for debugging and diagnosis (XAI4Debugging@NeurIPS2021).

is so far no algorithm providing such counterfactual explanations for large models at low cost. In this work, we introduce a fast novel algorithm using specific properties of residual networks (ResNets), a widely used model, to find counterfactuals which are very close to the original input.

The contributions presented in this work are the following:

- We introduce and implement a new, scalable algorithm for generating counterfactual explanations in residual networks trained with spectral normalisation. In comparison to previous approaches, this algorithm has a low engineering overhead as well as low computational requirements; it does not rely on auxiliary generative models and can run on single forward pass residual networks rather than on ensembles of models.
- We propose and implement a way to assess realism in counterfactual explanations, drawing on the literature on anomaly detection.
- We evaluate the novel algorithm on the MNIST dataset and compare its performance against three baselines from the literature.

2 Background

2.1 Counterfactual explanations

Counterfactual explanations for machine learning models have been introduced in [31].[2] They are used as post-hoc explanations for individual decisions. In general, a counterfactual explanation is understood as the presentation of an alternative input $\mathbf{x}'$, the counterfactual, which is in some way similar to the original $\mathbf{x}$ yet leads to a different prediction. Due to their similarity (both conceptually and w.r.t. the algorithms that generate them), counterfactuals are often introduced in juxtaposition with adversarial examples (AEs).[3] Freiesleben [5] provide an illuminating discussion of various definitions in the literature and come to the conclusion that while AEs are necessarily misclassified, counterfactuals need not be (and often should not be). Furthermore, in agreement with [21], they define counterfactuals as the *closest* alternative input (on some suitable metric) that changes the prediction (to a pre-defined target, if applicable). This seems to be too restrictive in general as there are applications that do not require such a strong focus on proximity. Furthermore, the definition then depends too much on the choice of the similarity metric. Consequently, we will generally call an alternative input $\mathbf{x}'$ a counterfactual to an input $\mathbf{x}$ under model f if it is similar to $\mathbf{x}$ on a suitable similarity relation but changes the prediction of the model.[4]

Counterfactual explanations can be useful for debugging: They can be used to answer questions like 'Why did the self-driving car misidentify the fire hydrant as a stop sign?' [8]. More precisely, they can answer the questions 'What would the image need to look like in order to be classified as a fire hydrant?' and 'What changes would need to happen in other fire hydrant (stop sign) images in order to be classified as a stop sign (fire hydrant)?'. Depending on the model and the application, we might only be interested in counterfactuals that look realistic. In particular, we learn more about the decision boundary when we understand the changes made to the image. If, on the other hand, the classification is changed by adding unrealistic noise, then we only learn that the classifier is not robust to this. In terms of Lipton [18], such counterfactuals are less informative. Counterfactual explanations are often required to be realistic [26], likely [21], or plausible [11]. Perhaps the most useful way to formulate it is to require counterfactuals to be 'likely states in the (empirical) distribution of features' which are 'close to the data manifold' [11]. Another often-mentioned desideratum for counterfactual explanations is sparsity: in general, the less features are changed in input space, the better. Sparse perturbations are usually more interpretable as the change in classification can be attributed to a smaller part of the input which is easier to grasp both in terms of the model behaviour and in terms

[2] A similar notion of counterfactuals in ML has also been explored by [15] in the context of algorithmic fairness rather than explainability.

[3] This is only partly helpful since there is also no consensus on the definition of AEs. While scholars generally agree that AEs are necessarily misclassified, there is no consensus on whether they need to be generated by imperceptible perturbations. Another, less operational definition could be that they are not detected to be outside the distribution of realistic datapoints – although the notion of 'misclassification' could already be interpreted as requiring the datapoint to be in-distribution as it implies that there is a correct classification.

[4] One might drop the requirement of changing the prediction for applications beyond explanations, e.g. for assessing fairness [15].

of the input itself. If all parts of the input change a little, it might be harder to understand what the perturbation means and how it affects the model.

2.2 Epistemic uncertainty

Previous work has shown that epistemic uncertainty can be used as a proxy for realism [27, 26]. There are two kinds of uncertainty relevant to machine learning models. Aleatoric uncertainty, on the one hand, is inherent in the data distribution and cannot be reduced. It is high when there is no clear ground-truth label and maximised in the extreme case of random labels. Epistemic uncertainty, on the other hand, is due to a lack of knowledge on the part of the model. It is a quantity that can be reduced for a given input by including it in the training set. Given this characterisation, epistemic uncertainty estimates are used for active learning (selecting samples that are particularly useful to train on) as well as for detecting out-of-distribution inputs.

It has also been observed that neuron activations in late layers of a neural network, which we call *features*, can be used to estimate epistemic uncertainty when two properties called sensitivity and smoothness are satisfied by the mapping into the feature space [28]. Sensitivity can be seen as a lower Lipschitz bound, ensuring that the features remain sensitive to differences between inputs, thereby preventing 'feature collapse': Otherwise, out-of-distribution (OoD) inputs might not be distinguishable from in-distribution (iD) inputs as they could be mapped to the same area in feature space. Conversely, smoothness can be seen as an upper Lipschitz bound: Similar inputs are guaranteed not to be too far from each other in feature space, such that distances in this space remain meaningful. Building on [2], Liu et al. [19] show that applying spectral normalisation (SN, [20]) with a coefficient $c \leq 1$ to ResNets [9] is enough to enforce both sensitivity and smoothness. Using such models, Mukhoti et al. [22] fit a probability distribution to the feature space after the last ResNet block, using the feature representations of the training data. To estimate the epistemic uncertainty of an input, they then calculate the negative log-likelihood of its feature representation under the learned distribution. For the probability distribution, they use a Gaussian mixture model (GMM). We will make use of this approach, called Deep Deterministic Uncertainty (DDU), in the algorithm we propose below.

2.3 Related work

In this section, we provide a (non-exhaustive) survey of algorithms that provide counterfactual explanations. In the initial paper, Wachter et al. [31] propose to simply optimise the objective

$$\arg\min_{\mathbf{x}'} \max_{\lambda} (f(\mathbf{x}') - \mathbf{y}')^2 \cdot \lambda + d(\mathbf{x}, \mathbf{x}') \tag{1}$$

where f denotes the model, $\mathbf{y}'$ the desired model prediction, d a pre-defined distance metric, and λ is a hyper-parameter. They suggest to iterate through increasing values of λ, always solving for $\mathbf{x}'$ for fixed λ, until a counterfactual sufficiently close to the original input is found. This is quite a minimal approach, which comes at the cost of $\mathbf{x}'$ not being constrained to lie on the data manifold (which might be required, depending on the task).

Van Looveren and Klaise [29] build on this (and on [3]), but focus on generating more interpretable counterfactuals by optimising a more complex objective function. Their prototype-guided approach minimises the loss

$$cL_{pred} + \beta L_1 + L_2 + L_{AE} + L_{proto}. \tag{2}$$

Here, $L_{pred} = f(\mathbf{x}')_l - \max_{i \neq l} f(\mathbf{x}')_i$ where $f(\mathbf{x}')_l$ is the softmax output of the classifier f on the counterfactual $\mathbf{x}'$ for the original class l, i.e. its confidence that $\mathbf{x}'$ belongs to class l. L_1 and L_2 refer to the corresponding distances between $\mathbf{x}$ and $\mathbf{x}'$ in input space. $L_{AE} = \gamma \|\mathbf{x}' - \mathsf{AE}(\mathbf{x}')\|_2^2$ where AE is an autoencoder trained on the training data. Lastly, $L_{proto} = \theta \|\mathsf{ENC}(\mathbf{x}') - \mathsf{proto}_t\|_2^2$ where $\mathsf{proto}_t = \frac{1}{K} \sum_{k=1}^{K} \mathsf{ENC}(\mathbf{x}_k^t)$ is the latent prototype defined by the K nearest neighbours $\mathbf{x}_k^t$ of $\mathbf{x}'$ in target class t and ENC is an encoder. For untargeted counterfactuals, t is chosen as $\arg\min_{t \neq l} \|\mathsf{ENC}(\mathbf{x}') - \mathsf{proto}_t\|_2$. While L_{pred} enforces a change of classification, L_{AE} and L_{proto} are included to encourage realism, measured by a low reconstruction loss under the autoencoder and similarity to similar training samples in the encoded latent space. The additional use of an autoencoder and the additional loss terms generate a computational overhead that slows down the approach. Furthermore, the approach comes with many hyperparameters which might require a lot of tuning when applying it to a new task. The interplay of the different loss terms is not straightforward

to analyse either formally or conceptually, so it is hard to predict and improve the performance on new applications.

A very different route to provide interpretable counterfactuals is taken by Schut et al. [26], based on the notion of uncertainty [6]. They suggest that counterfactuals are realistic if the model classifies them with low epistemic uncertainty and unambiguous when the model has low aleatoric uncertainty. Consequently, they propose to minimise overall uncertainty in models that provide accurate uncertainty estimates through their softmax output, such as deep ensembles [16]. They show that maximising the ensemble's target class prediction $f(\mathbf{x}')_t$ is sufficient to minimise its predictive entropy and hence both epistemic and aleatoric uncertainty. Rather than using an off-the-shelf optimisation algorithm as the two previously mentioned approaches, Schut and colleagues compute the gradient of the classification loss in input space and identify the most salient pixel for reducing this loss. Similar to JSMA [23], they iteratively change the most salient pixel until the target prediction is above 99%, when the uncertainty is sufficiently low. This works well in practice but using an ensemble of models (for their MNIST experiments they use 50 models) is computationally expensive, which might hinder its deployment in practice.

Another approach that has been suggested both for providing counterfactual explanations and for algorithmic recourse is REVISE [10]. This algorithm requires the availability of a generative model, such as the decoder of a VAE trained on the training data. Similar to [31], the overall idea is to minimise the function

$$\ell(f(G(\mathbf{z}')), t) + \lambda \cdot \|G(\mathbf{z}') - \mathbf{x}\|_1 \tag{3}$$

where f is the classifier, t is the target, ℓ is some loss function, and G is the generative model. To find a $\mathbf{z}'$ that minimises the loss, $\mathbf{z}$ is initialised to the encoding of the original input $\mathbf{x}$; then the gradient of the loss in the latent space is computed and the algorithm iteratively takes small steps in latent space until the prediction changes to the target. Since the resulting counterfactual $\mathbf{x}' = G(\mathbf{z}')$ is produced by the generative model, it can be dissimilar to $\mathbf{x}$: Although the L_1 norm is known to encourage sparsity, the algorithm cannot be expected to provide sparse solutions, as the changes are not taken in the input space. Another disadvantage of using a generative model is, of course, the need to train it beforehand which can pose difficulties of varying degree depending on the data domain. Several recent works utilise GANs for generating counterfactuals, such as [13].

3 Novel algorithm: DeDUCE

For large image datasets, ResNets are often the model of choice. When ResNets are trained with spectral normalisation, we can use DDU (section 2.2) to estimate their epistemic uncertainty. DDU achieves state-of-the-art results in OoD detection such as MNIST vs. FashionMNIST. The authors also demonstrate that it can be used for active learning; this implies that it is particularly useful to train on inputs whose representations have low likelihood under the GMM, i.e. such inputs are substantially different from the previous training data. This suggests that DDU's measure of epistemic uncertainty can be a useful target when aiming to generate counterfactuals that are similar to the original training data. The idea of the novel algorithm presented here is to cross the model's decision boundary while keeping the epistemic uncertainty as low as possible, using DDU. Instead of maximising the whole GMM density for minimising epistemic uncertainty (i.e. maximising feature-space density), we propose to only maximise the target class density (fig. 2). Otherwise, one would also maximise the original label's class density, which could lead to unstable behaviour. As we also want to cross the decision boundary quickly, we suggest to change the pixels that are most salient for the gradient of the loss

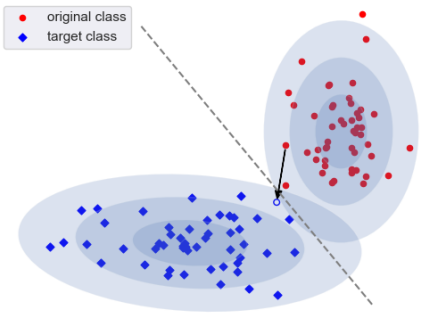

Figure 2: The training data is used to fit a Gaussian distribution for each class in a (here depicted) feature space. DeDUCE then changes a given input in a way that increases the density under the target class Gaussian in the feature space, until the decision boundary (dashed line) is crossed. This allows to generate realistic counterfactuals.

$$\mathbf{g} = \nabla_{\mathbf{x}} \left(\ell_c(\mathbf{x}, t) \cdot \lambda - \log p_t(f_Z(\mathbf{x})) \right) \tag{4}$$

where f is the model, t the target class, ℓ_c the cross-entropy, p_t the target class density, and f_Z the feature extractor, i.e. the part of the model that maps to the feature space after the last ResNet block. This means we are trying to change the input in a way that quickly changes the classification (first term) and makes it more similar to the target class in feature space (second term). The first term typically has values in $[0, 100]$, while the second term can have values as low as 10^{-6}. Instead of changing the pixels that minimise the weighted loss for some λ, we select the pixels which make the largest relative difference to either of the two loss terms. Therefore, we use the alternative gradient

$$\mathbf{g} = \frac{1}{\ell_c(f(\mathbf{x}'), t)} \nabla_{\mathbf{x}'} \ell_c(f(\mathbf{x}'), t) \cdot \mu - \frac{1}{|\log p_t(f_Z(\mathbf{x}'))|} \nabla_{\mathbf{x}'} \log p_t(f_Z(\mathbf{x}')) \tag{5}$$

instead of the gradient of the weighted loss given in equation (4). Note that the cross-entropy loss $\ell_c(f(\mathbf{x}'), t)$ is non-negative whereas $\log p_t(f_Z(\mathbf{x}'))$ can be negative or positive, depending on whether the density is above or below one. Despite their similarity, including both terms in the objective indeed improves the quality of generated counterfactuals; we also find that using the alternative gradient without further weighting ($\mu = 1$) works better than the gradient of the loss for any value of λ (see appendix A).

The novel approach that we call DeDUCE (Deep Deterministic Uncertainty-based Counterfactual Explanations) is described in algorithm 1. In order to only make small and sparse changes to the original input, DeDUCE iteratively perturbs only few pixels at a time. At each iteration, it determines the most salient pixels for maximising the objective, by computing the gradient in input space, and then perturbs them by a fixed step size δ. This is similar to how the Jacobian-based Saliency Map Attack (JSMA) [23] generates adversarial examples and the approach of [26] generates counterfactuals. Thereby, DeDUCE iteratively perturbs the input in small steps in a way that makes it more and more similar to the target class until it crosses the decision boundary. The algorithm stops when the softmax output for the target class is above 50% as this corresponds to the model choosing 'in target class' over 'not in target class'. Following [26], DeDUCE limits the number of times each pixel can be updated[5] and clips $\mathbf{x}'$ to the input domain bounds. Similar to some work on adversarial examples [4], we also add momentum to the gradient, replacing the expression for $\mathbf{g}$ by

$$\mathbf{g}_k = \frac{\nabla_{\mathbf{x}_k} \ell_c(f(\mathbf{x}_k), t)}{\ell_c(f(\mathbf{x}_k), t)} - \frac{\nabla_{\mathbf{x}_k} \log p_t(f_Z(\mathbf{x}_k))}{|\log p_t(f_Z(\mathbf{x}_k))|} + m \cdot \mathbf{g}_{k-1}, \tag{6}$$

with $\mathbf{x}_k$ referring to the state of the input after $k > 0$ iterations. For the experiments reported below, we change one pixel at a time and use a momentum of 0.6. Adding momentum of this size often does not make a difference; in our experiments, it only affected around 1.3% of the generated counterfactuals.

Algorithm 1 DeDUCE

Inputs: original input $\mathbf{x} \in \mathcal{X}$, target class t, trained model f with feature extractor f_Z, training data $X_{tr} \in \mathcal{X}^N$, coefficient λ, step size δ, max iterations max_iter, max pixel changes p, number of pixels m, target confidence γ.
Output: counterfactual $\mathbf{x}' \in \mathcal{X}$

1: (before deployment) apply f to X_{tr} and fit the GMM $p(\mathbf{z}) = \frac{1}{|C|} \sum_{c \in C} p_c(\mathbf{z})$
2: $\mathbf{x}' \leftarrow \mathbf{x}$
3: $k \leftarrow 0$
4: $\mathbf{P} \leftarrow \mathbf{0}_{dim(\mathcal{X})}$
5: **while** $f(\mathbf{x}')_t \leq \gamma$ **and** $k <$ max_iter **do**
6: compute gradient $\mathbf{g}$ in input space
7: select m most salient pixels: $I =$ `select_q_largest_masked`$(|\mathbf{g}|, \mathbf{P} < p)$
8: update these pixels: $\forall i \in I : \mathbf{x}'[i] \leftarrow \mathbf{x}'[i] +$ `sign`$(\mathbf{g}[i]) \cdot \delta$
9: clip to input domain: $\mathbf{x}' \leftarrow$ `clip`$(\mathbf{x}')$
10: $\forall i \in I : \mathbf{P}[i] \leftarrow \mathbf{P}[i] + 1$
11: $k \leftarrow k + 1$
12: **return** $\mathbf{x}'$

[5] This is achieved by counting the number of updates per pixel in $\mathbf{P}$ and applying a mask to the gradient $\mathbf{g}$ that sets $\mathbf{g}[i]$ to 0 if $\mathbf{P}[i] \geq r$. The expression `select_q_largest_masked`$(|\mathbf{g}|, \mathbf{P} < r)$ in line 7 denotes the procedure of first applying this mask and then selecting the positions of the q largest values of $|\mathbf{g}|$.

4 Experiments

4.1 Dataset and metrics

We perform experiments on the MNIST dataset [17], so far the only widely used image dataset in the literature on counterfactual explanations. We use the L_1 and L_0 distances in input space to assess similarity and sparsity, respectively. We also want to measure how realistic the generated counterfactuals are, as this is usually helpful (cf. section 2.1). There is no consensus in the field on how to measure realism (or 'plausibility' [12]), and for images, this proves to be quite difficult. Therefore, we turn to the literature on anomaly detection and use the approach that performed best on an anomaly detection task for MNIST in a recent study [24]. This approach, called AnoGAN [25], uses a pre-trained generative adversarial network (GAN, [7]) to compare the similarity of a given input $\mathbf{x}$ with the closest image $G(\mathbf{z})$ that the GAN can generate. AnoGAN uses gradient descent in latent space to minimise the loss

$$\|G(\mathbf{z}) - \mathbf{x}\|_1 + \lambda \cdot \|f_D(G(\mathbf{z})) - f_D(\mathbf{x})\|_1, \tag{7}$$

where G is the generator and f_D is a mapping to a later layer of the discriminator. The first term gives the L_1 distance between the generated sample and the input whereas the second term measures how similar their feature representations are in the discriminator model. We use a Wasserstein GAN [1] trained on MNIST. To reduce the dependence on the initial $\mathbf{z}$, we perform gradient descent three times from different, randomly sampled starting points. We use $\lambda = 0.1$, an initial learning rate of 0.03, and bound the number of iterations to 4000. We tested how well the resulting metric allows to distinguish actual MNIST images from EMNIST character as well as Fashion-MNIST images. Our AnoGAN method achieves an AUROC of 0.913 and 0.998, respectively. Note that these comparisons are quite different to the generated counterfactuals and thus only provide general sanity checks rather than actual performance tests.

4.2 Baselines

To assess the performance of the novel algorithm, we compare it with three baselines applied to ResNets without spectral normalisation. The prototype-guided approach that we shall call 'VLK' [29] as well as REVISE [10] were discussed in section 2.3. Their selection is largely based on a general scarcity of algorithms that provide counterfactual explanations, were demonstrated on image data, and are applicable to ResNets. Although also demonstrated on MNIST, the mentioned approach of [26] is not included since it cannot be applied to single ResNets. In order to get the required calibrated uncertainty outputs, one would need an ensemble of around ten models [16], which would make the results much less comparable. In addition to VLK and REVISE, we include JSMA [23] as a third baseline.

It should be noted that **JSMA** was introduced to generate (perceptible) adversarial examples rather than counterfactual explanations, so the comparison with regard to realism is not a fair one. However, since DeDUCE is loosely based on JSMA, the comparison is interesting as it shows how the modifications affect the results. While the original paper recommends changing two pixels at a time, we change one as this makes it perform better in our setting and more comparable to the used DeDUCE algorithm.

To generate counterfactuals with **VLK**, we use the authors' implementation in the `alibi` package [14] in order to be as faithful as possible. As the algorithm is already tuned to MNIST, we only make one change to the default setting, namely setting k for k-nearest (encoded) neighbours in the L_{proto} term to 5. This is recommended in the paper and generally improves the quality of the generated counterfactuals. Note that VLK uses an optimisation algorithm that includes the model's target confidence in the L_{pred} loss term. This means that, contrary to the other three algorithms, we cannot prescribe the generated counterfactuals to have a target confidence just above 50%. In fact, they have a mean confidence of 0.42 and standard deviation of 0.44, with many values being close to 0 or 1.

The third baseline **REVISE** requires more tuning, as it has not been demonstrated on MNIST before. REVISE is designed to be applicable to image data, with the authors providing a demonstration on the CelebA dataset. Sample reconstructions of the used VAE are shown in appendix B. Note that a more powerful generative model than the one employed here could improve the quality of the generated counterfactuals; this might, however, come with even higher engineering efforts and computational costs. Appendix B provides more details on the implementation of REVISE.

4.3 Results

With the four algorithms tuned to the dataset and base model, we can look at their performance. We use five sets of 100 original MNIST images from the test set and try to find counterfactuals for all 9 potential target classes, resulting in 5×900 runs overall for each algorithm. The quantitative metrics are the L_0 and L_1 distance to the original, and the rate at which the algorithms fails to output a candidate counterfactual. For the evaluation of how realistic/anomalous the images are, we use the anomaly detection metric 'AGAN' using AnoGAN. The results are reported in table 1. In order to ensure a fair comparison, only image-target pairs are included for which all algorithms found a counterfactual.

Table 1: Overall means from 5×900 image-target pairs, with standard deviations over the five means in brackets. *AGAN* denotes the AnoGAN score. Original images achieve an average *AGAN* score of 15.44. Lower is better everywhere.

algorithm	*AGAN*	L_0	L_1	*failure in %*
DeDUCE	22.51 (0.72)	**21.16** (0.46)	**10.72** (0.10)	**0.00** (0.00)
JSMA	23.90 (0.41)	25.65 (0.65)	12.63 (0.27)	3.09 (0.28)
VLK	22.95 (0.93)	155.43 (4.15)	38.95 (1.19)	0.09 (0.20)
REVISE	**20.09** (0.88)	752.46 (4.98)	53.86 (0.58)	26.80 (1.03)

REVISE achieves the best results on the AnoGAN metric, but their scores are not as good as for the original images, which get a score of 15.44 ± 2.04. VLK and DeDUCE do not show significant differences on this metric. We note that the directly reconstructed images under the REVISE VAE (without applying REVISE) are judged to be more realistic than the original images, achieving a score of 14.64 ± 0.23. This shows that the metric has a bias towards VAE-generated images, which benefits REVISE and perhaps also VLK, as the latter minimises an autoencoder loss. DeDUCE achieves better results than JSMA on all four metrics, with these differences are all being highly significant ($p < 10^{-7}$ on paired t-tests). Both VLK and REVISE perform much worse than the other two on the sparsity as well as the similarity metric. This is also expected since both DeDUCE and JSMA take pixel-wise steps in the input space. The standard deviations on all metrics are generally fairly low, especially for DeDUCE and JSMA.

There are great differences with respect to the required computation times, not only because the provided implementation of VLK does not support the generation of counterfactuals in batches. Even when all approaches are used to generate counterfactuals individually, VLK is the slowest approach (table 2). JSMA is slightly faster than DeDUCE, which is still significantly faster than REVISE. REVISE has a large standard deviation because some runs take particularly long. Note that REVISE could work with a slightly larger step size and with a lower number of iterations, at the expense of quality. All computations were performed on NVIDIA Tesla T4 GPUs.

Table 2: Average running times (STDs in brackets) for generating counterfactuals individually.

algorithm	*time in sec*
DeDUCE	2.99 (1.69)
JSMA	**1.01** (0.52)
VLK	109.86 (1.54)
REVISE	46.66 (84.33)

As pointed out above, the GAN-based realism metric is only partly reliable. This can already be seen from the observation that VAE reconstructions are generally judged to be more realistic than the original image. Therefore, a qualitative examination of the generated counterfactuals is also warranted, although a conclusive verdict would require a comprehensive human evaluation study. Individual examples generated by the different algorithms for randomly drawn images and targets are presented in figure 3.[6] Note that REVISE failed to output a counterfactual in the fourth column.

[6]For both the image ids and the target classes, eight integers from 0 to 9 were drawn at random. One pair was drawn twice while for another pair, the target was equal to the label, resulting in six id-target pairs.

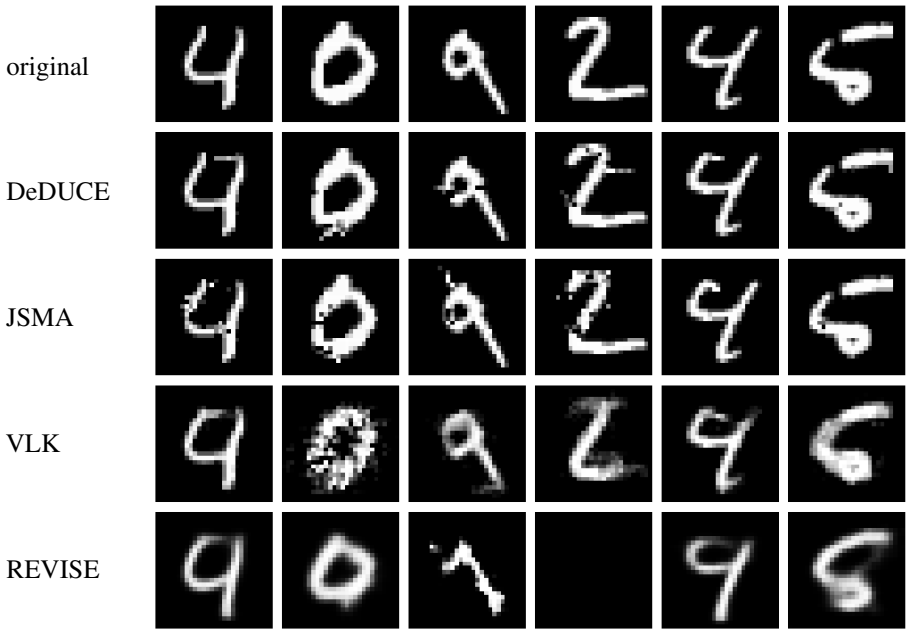

Figure 3: Random examples of generated counterfactuals. The first row shows the original image, then the results of applying DeDUCE, JSMA, VLK, and REVISE. The target classes are 9, 9, 3, 6, 9, and 6, respectively. The individual metric scores are reported in table 3.

Table 3: Metric scores for each of the generated counterfactuals presented in fig. 3.

	(original, target)						
algorithm	(4,9)	(0,9)	(9,3)	(2,6)	(4,9)	(5,6)	metric
DeDUCE	14.01	21.52	25.55	22.82	12.23	17.71	*AGAN*
	5	18	12	20	3	10	L_0
	3.20	7.96	6.75	9.19	2.59	2.96	L_1
JSMA	23.31	22.32	19.86	25.77	16.66	23.01	*AGAN*
	29	15	19	35	10	6	L_0
	11.19	10.39	8.85	15.69	4.73	2.68	L_1
VLK	11.77	56.81	18.64	23.06	18.41	22.70	*AGAN*
	60	276	141	177	93	160	L_0
	10.35	81.19	27.28	48.84	17.37	36.72	L_1
REVISE	15.13	21.47	27.41	N/A	5.49	25.80	*AGAN*
	784	784	692	N/A	784	784	L_0
	31.90	66.08	57.63	N/A	17.60	62.13	L_1

Column 2 and 4 seem generally difficult, while all approaches arguably find good counterfactuals for column 1 and 5, with the exception of JSMA on the former. Column 3 and 6 are met with varying success. Overall, DeDUCE and VLK might be seen to provide the best counterfactuals. The individual metric scores are presented in table 3. Despite the lack of a user study, it seems clear that the NLL and AGAN scores do not always reflect human judgement. Because of this and the much higher L_0 and L_1 scores, the low realism scores of counterfactuals generated by VLK and REVISE clearly do not imply that they are more interpretable than the ones generated by DeDUCE.

5 Discussion

As demonstrated in the experiments, DeDUCE is an efficient algorithm performing small and sparse perturbations that often result in realistic counterfactuals. In particular, it provides counterfactuals that are much more similar to the original image than the other considered approaches. This allows to give more precise explanations for the model's decision making. As discussed, DeDUCE is only applicable to classifiers that satisfy sensitivity and smoothness assumptions. It is sufficient to have a ResNets with some loose spectral normalisation, which might be desired anyway. Still, this clearly restricts the applicability of DeDUCE. Overall, DeDUCE could prove to be a viable technique for a considerable number of use cases, namely image classification tasks that require the deployment of large neural networks.

One limitation of this work is the lack of human evaluation studies. In order to conclusively assess how interpretable the generated counterfactuals are and how helpful the explanations are for debugging, such studies will eventually be necessary. Another limitation is the lack of demonstrations on more complex datasets. While MNIST allows a first proof of concept, the algorithm is designed to be scalable to larger datasets and should also be assessed there.

Lastly, future work could also try to improve the algorithm itself. In particular, it might be possible to use a different latent density model instead of the one used here. For example, the GMM could be fitted to ambiguous data (using multiple labels and a probabilistic fit) to improve density estimation on such inputs. This could make it necessary to also train the classification model on ambiguous data. Another option could be to use confidence-weighting of the datapoints for fitting the class-wise Gaussians. We leave these ideas as avenues for future research.

Acknowledgements

We would like to thank Andreas Kirsch for helpful discussions. B.H. was supported through a DAAD scholarship. L.S. and J.M.B. were supported by DeepMind and Cancer Research UK, respectively. Both L.S. and J.M.B. were also supported by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems (EP/S024050/1).

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017.

[2] P. L. Bartlett, S. N. Evans, and P. M. Long. Representing smooth functions as compositions of near-identity functions with implications for deep network optimization. *arXiv preprint arXiv:1804.05012*, 2018.

[3] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *arXiv preprint arXiv:1802.07623*, 2018.

[4] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *CVPR*, pages 9185–9193, 2018.

[5] T. Freiesleben. The intriguing relation between counterfactual explanations and adversarial examples. *Minds & Machines*, pages 1–33, 2021.

[6] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *NIPS*, 27, 2014.

[8] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee. Counterfactual visual explanations. In *International Conference on Machine Learning*, pages 2376–2384. PMLR, 2019.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[10] S. Joshi, O. Koyejo, W. Vijitbenjaronk, B. Kim, and J. Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615*, 2019.

[11] A.-H. Karimi, G. Barthe, B. Schölkopf, and I. Valera. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*, 2021.

[12] M. T. Keane, E. M. Kenny, E. Delaney, and B. Smyth. If only we had better counterfactual explanations: Five key deficits to rectify in the evaluation of counterfactual xai techniques. *arXiv preprint arXiv:2103.01035*, 2021.

[13] E. M. Kenny and M. T. Keane. On generating plausible counterfactual and semi-factual explanations for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11575–11585, 2021.

[14] J. Klaise, A. Van Looveren, G. Vacanti, A. Coca, and R.-F. Samoilescu. Alibi: Algorithms for explaining machine learning models, 2019. URL `https://github.com/SeldonIO/alibi`.

[15] M. J. Kusner, J. R. Loftus, C. Russell, and R. Silva. Counterfactual fairness. *arXiv preprint arXiv:1703.06856*, 2017.

[16] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] Z. C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[19] J. Z. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax-Weiss, and B. Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint arXiv:2006.10108*, 2020.

[20] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[21] C. Molnar. *Interpretable machine learning*. Lulu.com, 2020.

[22] J. Mukhoti, A. Kirsch, J. van Amersfoort, P. H. Torr, and Y. Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*, 2021.

[23] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[24] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 2021.

[25] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157, 2017.

[26] L. Schut, O. Key, R. McGrath, L. Costabello, B. Sacaleanu, M. Corcoran, and Y. Gal. Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. *AISTATS*, 2021.

[27] L. Smith and Y. Gal. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018.

[28] J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal. Uncertainty estimation using a single deep deterministic neural network. In *ICML*, pages 9690–9700, 2020.

[29] A. Van Looveren and J. Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2020.

[30] S. Verma, J. Dickerson, and K. Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[31] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31:841, 2017.

A Effect of different gradient expressions

Table 4: Effect of different objective functions with different hyperparameter settings. Settings with λ use a loss function as in equation (4), whereas settings with μ use the alternative gradient as in equation (5). The alternative gradient with no further weighting ($\mu = 1$) works best on all metrics. In particular, not including the classification loss at all ($\lambda = 0$) performs much worse.

setting	L_0	L_1	*failure*
$\lambda = 0$	27.36	13.47	0.1%
$\lambda = 1$	27.29	13.45	0.1%
$\lambda = 10$	27.16	13.40	0.1%
$\lambda = 10^2$	25.88	12.92	0%
$\lambda = 10^3$	24.58	12.38	0%
$\lambda = 10^4$	25.10	12.53	0%
$\lambda = 10^5$	25.38	12.65	0%
$\mu = \frac{1}{5}$	25.45	12.75	0%
$\mu = 1$	**24.47**	**12.32**	0%
$\mu = 5$	24.92	12.46	0.1%

B REVISE implementation

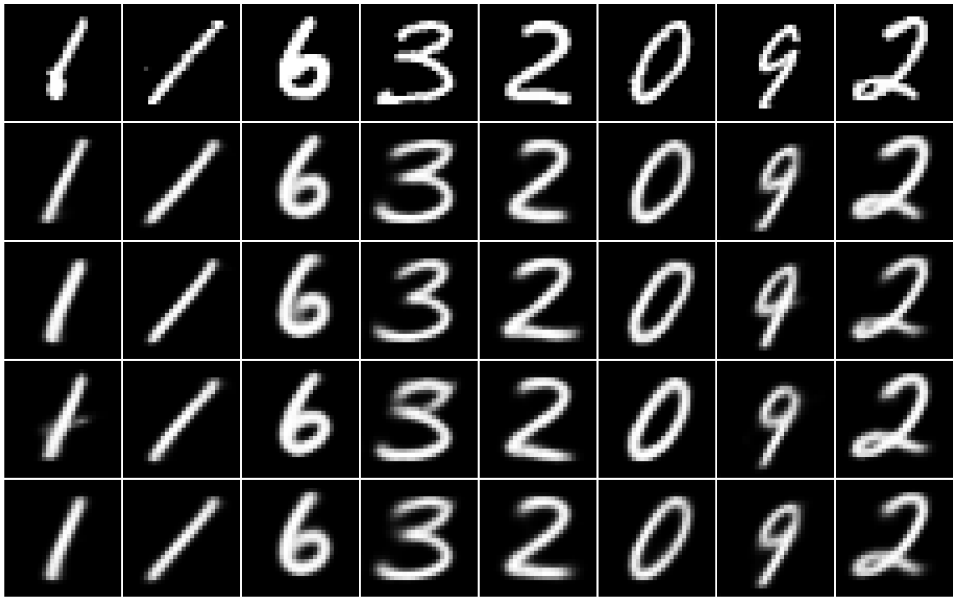

Figure 4: Some random reconstructions from the VAE used for REVISE. The first row shows randomly selected original MNIST images, followed by three rows of sample reconstructions and the average of 50 reconstructions in the last row.

Recall that REVISE takes small steps in the latent space of the VAE, guided by the gradient of the loss function $\ell(f(G(\mathbf{z}')), t) + \lambda \cdot \|G(\mathbf{z}') - \mathbf{x}\|_1$ where f is the classifier, t is the target, G is the generative model, and $\mathbf{x}$ is the original image. As in the original paper, we use the cross-entropy loss function for ℓ. In addition to the VAE, it is therefore necessary to tune λ as well as the gradient step size that we denote by δ. I perform a grid search on $\lambda \in \{0.1, 1, 10\}$ and $\delta \in \{10^{-3}, 10^{-4}, 10^{-5}\}$, considering both qualitative and quantitative performance. For the three setting of λ, we limit the number of iterations to 50,000, 10,000, and 5,000, respectively.[7] For $\lambda = 10$, the algorithm hardly ever terminates: in all settings for δ, it has a failure rate of over 75%. Comparing all settings on the

[7] In all three cases, less than 5% of the runs terminated in the last 60% of the iterations, i.e. after step 20,000, 4,000, and 2,000, respectively. This shows that to significantly decrease the failure rate, the iteration limits would need to be raised by more than an order of magnitude, if that helps at all. This is taken as a justification

same image-target pairs would then mean to leave out the vast majority of counterfactuals, so I report quantitative evaluations only for the other six settings in table 5. Most notably, for $\lambda = 0.1$, the L_1 values are very high; a look at the generated images confirms that these are too far from the original inputs to be useful. Given $\lambda = 1$, the setting with $\delta = 10^{-5}$ performs clearly the best overall, so we adopt this for the evaluation on the testset.

Table 5: Performance of REVISE in different settings of λ and δ. For $\lambda = 0.1$, the counterfactuals are very dissimilar to the original input (see L_1) and among settings with $\lambda = 1$, the one with $\delta = 10^{-5}$ performs best overall. Settings with $\lambda = 10$ are not included as the failure rate is above 75%.

λ	δ	L_0	L_1	*failure*
0.1	10^{-3}	754.34	88.79	16.1%
0.1	10^{-4}	758.12	91.43	12%
0.1	10^{-5}	764.93	84.92	11.8%
1	10^{-3}	767.75	58.66	23%
1	10^{-4}	771.78	54.72	26.3%
1	10^{-5}	771.89	54.11	22.2%

C Additional DeDUCE outputs

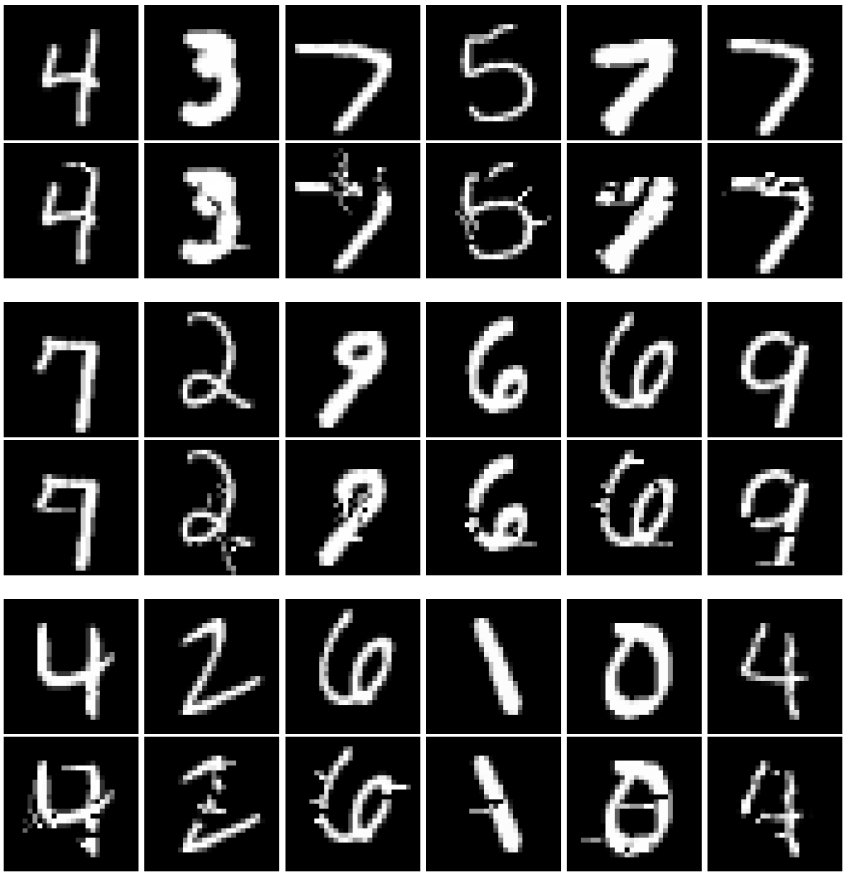

Figure 5: Additional randomly selected examples produced by DeDUCE. The targets are: **in row two** 9, 1, 4, 4, 1, 9; **in row four** 9, 9, 0, 1, 2, 3; **in row six** 0, 5, 5, 2, 9, 1. In general, the algorithm seems to have the largest difficulties with generating 0s and 1s.

for keeping them at their present values. As a comparison, recall that we limit DeDUCE (and JSMA) to 700 iterations.