Counterfactual Explanation via Search in Gaussian Mixture Distributed Latent Space

Anonymous authors
Paper under double-blind review

Abstract

Counterfactual Explanations (CEs) are an important tool in Algorithmic Recourse for addressing two questions: 1. What are the crucial factors that led to an automated prediction/decision? 2. How can these factors be changed to achieve a more favorable outcome from a user's perspective? Thus, guiding the user's interaction with AI systems by proposing easy-to-understand explanations and easy-to-attain actionable changes is essential for the trustworthy adoption and long-term acceptance of AI systems. In the literature, various methods have been proposed to generate CEs, and different quality measures have been suggested to evaluate these methods. However, the generation of CEs is usually computationally expensive, and the resulting suggestions are unrealistic and thus non-actionable. In this paper, we introduce a new method to generate CEs for a pre-trained binary classifier by first shaping the latent space of an autoencoder to be a mixture of Gaussian distributions. CEs are then generated in latent space by linear interpolation between the query sample and the centroid of the target class. We show that our method maintains the characteristics of the input sample during the counterfactual search. In various experiments, we show that the proposed method is competitive based on different quality measures on image and tabular datasets – efficiently returns results that are closer to the original data manifold compared to three state-of-the-art methods, which are essential for realistic high-dimensional machine learning applications.

1 Introduction

Machine learning models have recently become ubiquitous in our society, and the demand for explainability is rising, especially in high-stake applications like healthcare, finance, and employment. Explainability of the model's behavior is the prerequisite for establishing trust in machine learning-based decision systems. Researchers have developed various techniques to explain the relationships between the input and output of a model. While models with relatively simple design (e.g., logistic regression, decision trees, rule fit algorithm) can be interpreted straightforwardly, more complex models are analyzed based on simpler surrogate models. These surrogate models emulate the more complex model in a post-hoc fashion locally and provide explanations in terms of local feature importance (i.e., for a given input). For example, LIME (Ribeiro et al., 2016) perturbs the input features and measures the effects of the perturbation through a local linear approximation of the complex model to assess the importance of the input features. On the other hand, SHAP (Lundberg & Lee, 2017) applies a game-theoretic approach to evaluate the contribution of input features to the output. While LIME and SHAP have their specific advantages and disadvantages, they have a shared objective: estimate the effect of each input feature on a given prediction.

Another important line of research aims to answer the following question in a binary classification setting – "How can the input be changed to achieve a prediction representing the favored class instead of the unfavored one?". The exploration of outcomes in alternative similar yet non-occurring worlds is called counterfactual analysis (Menzies & Beebee, 2020). With a pre-trained and fixed model, the only way for the model to produce a different output is by altering the input. To this end, counterfactual explanations (CEs) provide prescriptive suggestions on the features of the query sample (i.e., input) that have to change (and also by how much they need to change) to achieve the desired outcome. We require this change to be minimal (i.e., associated with low costs) and actionable (i.e., realistic and feasible). With a clear semantic implication and a common logical grounding, CEs are generally easy to understand by end-users (Fernandez et al., 2020). In fact, in literature, the most direct possible usage of CEs is the guidance to the end-users.

However, CEs have their downsides. High-dimensional input spaces lead to high-dimensional CEs with limited utility for the potential users of the explanation (less intuitive and less feasible) In addition, searching through the high-dimensional space for CEs is computationally expensive. Other than the problem caused by the dimensionality, without proper constraints, it is possible to generate out-of-sample CEs close to the original data distribution. Out-of-sample CEs can result in explanations/suggestions that are not actionable (i.e., unlikely to be achieved since they do not correspond to the training data distribution). Adversarial samples (Goodfellow et al., 2015) which resemble the original sample but have changed imperceptibly to fool the classifier, might be a good example to illustrate this situation even though they are designed for a different purpose(deceive human perception and pre-trained classifiers). We show in Section 4 that a search in the original space for CEs might cause this situation. An actionable CE should stay close to the data manifold and suggest meaningful (i.e., realistic and easily achievable) changes to a query sample. Various requirements are proposed in the literature for generating useful CEs: actionable, sparse, valid, proximate, and computationally efficient to generate (Verma et al., 2020). It is also clear that there might be a trade-off between these requirements.

We introduce a method for generating CEs by using interpolation within the latent representations of the input data to achieve the requirements mentioned above. We perform experiments with an image dataset – MNIST and two tabular datasets – the Adult income and Lending Club loan default. Our main contributions are: (1) a model agnostic framework for finding feasible and actionable CEs that are prominent in scalability in data with a low computational expense, (2) a novel strategy for manipulating the latent space for the counterfactual search, (3) a comparison of methods across the image and tabular datasets.

The remainder of this paper is structured as follows: First, we briefly review the existing methods of counterfactual explanation (Section 2). Then, in Section 3, we propose a CE method via autoencoder-aided search in Gaussian Mixture (GM) distributed latent space. In Section 4, we present the evaluation results of our method against three state-of-the-art methods. Finally, Section 5 concludes the paper.

2 Background and Related Work

2.1 Generating Counterfactuals by Perturbing the Original Input Space

One large branch of the literature generates counterfactuals by perturbing the input feature space. Inverse classification (Lash et al., 2017) maintains sparsity by partitioning the features into immutable and mutable features and imposing budgetary constraints on the mutable features. A sampling approach is proposed by Laugel et al. (2017) with the growing spheres method to traverse the input space for CEs. Gradient descent is utilized in input space to find contrastive explanations (Dhurandhar et al., 2018), which are separated into pertinent positives and pertinent negatives. They included an autoencoder loss to keep the explanations in sample. Gradient descent methods are improved with the introduction of prototypes, guiding the gradient descent towards the average value of the target class by averaging the training sets representation in the latent space (Van Looveren & Klaise, 2020). GRACE (Le et al., 2020) is designed for neural networks on tabular data that combines the concepts of contrastive explanations with interventions by performing constrained gradient descent adding an additional loss that is a measure of information gain to keep the resulting explanations sparse. Several methods mentioned above already include generative models to maintain in sample. The criteria of the XAI for CEs are included in the methods either as constraints or backbone of the loss design, which might lead to difficult optimization especially when the input is high dimensional.

2.2 Generating Counterfactuals by Perturbing the Latent Space

Input feature perturbation methods without proper regularization can generate counterfactuals unconvincing (Goyal et al., 2019) and infeasible, resembling adversarial samples. To address this problem, latent space perturbations methods could be a solution since they utilize generative and probabilistic models in the algorithm design to ensure counterfactuals have a high probability under

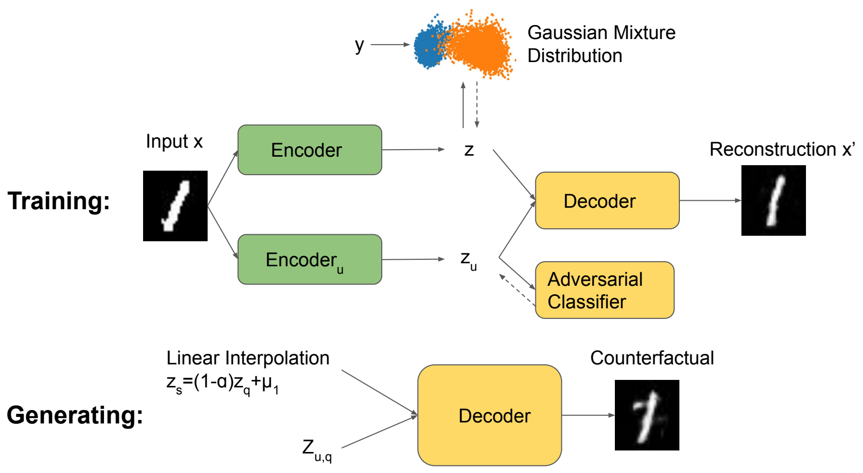

Figure 1: Diagram of the data flow in the algorithm. During Training: an input sample x from the training dataset is embedded in a latent space with a Gaussian mixture distribution by the Encoder. x is also embedded into a latent space to get z_u learned by the Encoder$_\mathrm{u}$ with an Adversarial Classifier that helps remove the information related to the classification label $\hat{y}$ of x under the pre-trained classifier f. z and z_u are then combined to get reconstruction x'. During Generating: x_q is a certain query sample we need to explain with CE. z_q is Encoder(x_q) and μ_1 is the mean of target class in latent space z. A linear search is performed in the latent space of z to get a searched potential z_s. Then z_s and $z_{u,q}$ are projected back to the original space by the Decoder until the decision boundary of the classifier is crossed.

the data distribution $p(X)$. One example is StylEx (Lang et al., 2021), which includes the classifier in the generative model and manipulates the latent space to visualize the counterfactual search. Moreover, ExplainGAN (Samangouei et al., 2018) is a method for finding counterfactual explanations for images by training multiple autoencoders and using the signal from the classifier and discriminators to inform the learned representations. Growing sphere search is performed in the latent space of a conditional variational autoencoder to generate counterfactuals (Pawelczyk et al., 2020). Gradient descent in the latent space of a variational autoencoder is applied with regularization terms (Balasubramanian et al., 2021). Sharpshooter (Barr et al., 2021) searchs counterfactuals by linear interpolation in the latent space with aid from two separate autoencoders.

3 Our Method

3.1 Problem Setting

Our method requires access to the dataset and a previously trained classifier that we want to explain with counterfactual explanations. We only consider a binary classification in this paper. In the problem setting of counterfactual explanation, there is usually a base class where instances belong that class wants to seek actionable counterfactual explanation toward the target class. The dataset is composed $\mathcal{D} = (x_i, y_i)_{i=1}^{K}$ where we can split x into two classes where x_i belongs to the base class has $\hat{y}_i = 0$ and x_i belongs to the target class has $\hat{y}_i = 1$ under the classifier $\hat{y} = f(x)$. x_q is a query sample with an output 0 from the classifier f. 1 is the desired target output. Hence, CEs are needed for the query sample x_q under the classifier f.

3.2 Desiderata for CE Search

Our approach is driven by the following main desiderata:

Desideratum 1 Intrinsic and label-irrelevant characteristics of the input query should be retained during the search process, i.e., the target label should be reached based on intrinsic properties of the input and with low effort.

Desideratum 2 The generating step, should return highly realistic, i.e., feasible/actionable CEs.

Desideratum 3 The CEs should be retrieved efficiently in high-dimensional settings to ensure practical application to real-life settings.

3.3 Learning Components

We design a CE generation method based on two main steps: a Training Step in which we use the same training dataset used in training the classifier to learn an autoencoder with its latent space shaped by enforcing a Gaussian-mixture distribution on the embeddings and a Generating Step in which we use interpolation in the latent space to find a relatively less computational expensive counterfactual explanation. The algorithm flow is shown in Figure 1. It only requires access to the training dataset and prediction of the pre-trained classifier f that we aim to explain through appropriate CEs. For our Training Step, we form a new training dataset $\mathcal{D}_t = (x_i, \hat{y}_i)_{i=1}^{K}$ by replacing original y in $\mathcal{D}$ with $\hat{y}$.

In the following part of this subsection, we will describe each component of the Training Step in Figure 1. The details of Generating Step are described in Section 3.4.

Label Relevant Branch: Gaussian Mixture Distribution Gaussian Mixture models are usually used in a supervised way, but in our usage, it is supervised by the classification label produced by the pre-trained classifier. Our goal for generating a CE is to cross the decision boundary of the pre-trained classifier by taking advantage of Gaussian mixture distribution in the latent space, in which case, we could generate a CE without having a process of optimization for each query sample. Intuitively, in the latent space, data points with the same class label should cluster closer to each other. Inspired by GM loss (Wan et al., 2018), we could use a classification constraint and likelihood constraints to 'push' the latent space to a Gaussian mixture distribution for further manipulation. After proper training, we could 'force' the extracted embedding z on the training set following a Gaussian mixture distribution expressed in Equation 1, in whichμ_c and σ_c are the mean and covariance of class c in the latent space, and $p(c)$ is the prior probability of class c. In the binary classification setting, $c \in [0, 1]$.

$$p(z) = \sum_c p(z \mid c)p(c) = \sum_c \mathcal{N}(z; \mu_c, \Sigma_c)p(c) \tag{1}$$

If the latent space follows a Gaussian mixture distribution, the conditional probability distribution of a latent embedding z given its class label c can be expressed in Equation 2. The corresponding posterior probability distribution can be expressed in Equation 3.

$$p(z \mid c) = \mathcal{N}(z; \mu_c, \Sigma_c)p(c) \tag{2}$$

$$p(c \mid z) = \frac{\mathcal{N}(z; \mu_c, \Sigma_c)p(\mu_c)}{\sum_{c=1}^{C} \mathcal{N}(z; \mu_c, \Sigma_c)} \tag{3}$$

A **classification loss** $\mathcal{L}_{cls}$ is then calculated as the cross-entropy between the posterior probability distribution and the class label as is shown in Equation 4.

$$\mathcal{L}_{cls} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\mathcal{N}(z_i; \mu_{\hat{y}}, \Sigma_{\hat{y}})p(\mu_{\hat{y}})}{\sum_{c=1}^{C} \mathcal{N}(z_i; \mu_c, \Sigma_c)} \tag{4}$$

Applying the classification loss only cannot reach our goal of forcing the latent space to be a Gaussian mixture distribution. There will be situations where a z_i can be far away from the corresponding target class centroid μ_c and still be correctly classified since it is relatively closer to μ_c than to the means of the other classes in multiple classifications–which could be an outlier. To fix this problem, we then use a likelihood to measure the extent to of the training data fits the Gaussian mixture distribution. The **likelihood** for $\{z, c\}$ is expressed in Equation 5. The likelihood could serve as a constraint to the original classification loss.

$$\mathcal{L}_{lkd} = -\sum_{i=1}^{N} \log \mathcal{N}(z_i; \mu_{z_i}, \Sigma_{z_i}) \tag{5}$$

Gaussian mixture loss $\mathcal{L}_{GM}$ we optimize to update the parameters of Encoder, μ_c and Σ_c, during Step 1, is defined in Equation 6, in which λ is a weighting coefficient.

$$\mathcal{L}_{GM} = \mathcal{L}_{cls} + \lambda_{lkd}\mathcal{L}_{lkd} \tag{6}$$

Label Irrelevant Branch: Encoder$_u$ and Adversarial Classifier We notice that generative models usually try to generate various unseen samples. However, the generation of CEs needs to satisfy different criteria, as mentioned in Section 1. In our problem setting, to satisfy *Desideratum 1*, which requires maintaining the characteristics of the query sample during the search, it is intuitive to adopt disentanglement methods in the latent space of an autoencoder.

Inspired by a Two-Step Disentanglement Method Hadad et al. (2020), we introduce an Adversarial Classifier to ensure that the embedding z_u captured by the Encoder$_{\text{u}}$ is classification label-irrelevant. It is inspired by GANs, where the discriminator gradually loses the capacity to tell the generated data from the real data during the training phase. While GANs are usually used to improve the quality of generated output – telling fake from real, the adversarial component in our design encourages the Encoder$_{\text{u}}$ to dismiss information about the labels, leading to disentanglement. With the Adversarial Classifier, we could guarantee that the z_u and $\hat{y}$ are disentangled and independent, which prepares for the interpolation in the Generating Step. $\hat{y}'$ is the classification label of z_u through the Adversarial Classifier. The **adversarial classification loss** is shown in Equation 7 as binary cross entropy loss.

$$\mathcal{L}_{adv} = -\frac{1}{N}\sum_{i=1}^{N} \hat{y_i}\text{log}\hat{y_i}' + (1-\hat{y_i})\text{log}(1-\hat{y_i}') \tag{7}$$

Autoencoder The reconstruction error of the autoencoder is shown in Equation 8.

$$\mathcal{L}_{rec} = \frac{1}{N}\sum_{i=1}^{N} ||x_i - x_i'||^2 \tag{8}$$

Summary The configuration of the network in the Training Step is composed of three network branches: first, in the **Label Relevant Branch**, the $\mathcal{L}_{GM}$ forces z to be Gaussian Mixture Distribution. Second, in the **Label Irrelevant Branch**, the Adversarial Classifier is trained to minimize the **adversarial classification loss** $\mathcal{L}_{adv}$ in Equation 7 – it is trained to classify z_u to $\hat{y}$. Third, the autoencoder network is trained to minimize the **total loss** $\mathcal{L}$, the sum of three terms as shown in Equation 9: (i) the **reconstruction loss** $\mathcal{L}_{rec}$ as shown in Equation 8, (ii) **likelihood** in Equation 5 and (iii) minus the **adversarial classification loss** $\mathcal{L}_{adv}$ in Equation 7.

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{lkd}\mathcal{L}_{adv} - \lambda_{adv}\mathcal{L}_{adv} \tag{9}$$

3.4 ALGORITHM

Algorithm 1 and 2 show pseudo code to process our method in addition to Figure 1 which is used to generate the counterfactual in Section 4.

For the Training Step, we sample a batch from the training dataset to update the parameters of Adversarial Classifier ω, Encoder ψ and mean and covariance μ_b, μ_t, and σ_b, σ_t – base class and target class combined as training data. Then we sample another batch to update the Encoder, Encoder$_u$, and Decoder parameters together(ψ, π, and ϕ). We iterate this procedure until convergence of the loss functions is reached. During the end of the Training Step: we should get a **label relevant** latent space z following Gaussian Mixture distribution (see Figure **??** for PCA of z in Appendix) and a **label irrelevant** latent space z_u to capture the label irrelevant information of samples. In Generating

Algorithm 1 Training Step of our proposed architecture

```
Require: $\psi$, $\pi$, $\phi$ and $\omega$ the initial parameters of Encoder, Encoder$_u$, Decoder and Adversarial
    Classifier; $\mu_c$ and $\Sigma_c$ the initial mean and covariance of the Gaussian distribution of $z$; $n$ the
    number of iterations; $\lambda_{lkd}$ and $\lambda_{adv}$ the weights of regularization terms $\mathcal{L}_{lkd}$ and $\mathcal{L}_{adv}$; $c \in [0,1]$
 1: while not converged do
 2:     for $i = 0$ to $n$ do
 3:         Sample $\{x, y\}$ a batch from dataset $\mathcal{D}_t$.
 4:         $\omega \overset{+}{\leftarrow} -\bigtriangledown_\omega \mathcal{L}_{adv}$
 5:         $\mathcal{L}_{GM} \leftarrow \mathcal{L}_{cls} + \lambda_{lkd}\mathcal{L}_{lkd}$
 6:         $\psi, \mu_c, \Sigma_c \overset{+}{\leftarrow} -\bigtriangledown_{\psi,\mu_c,\sigma_c} (\mathcal{L}_{GM} - \lambda_{adv}\mathcal{L}_{adv})$
 7:         Sample $\{x, y\}$ a batch from dataset $\mathcal{D}_t$.
 8:         $\mathcal{L} \leftarrow \mathcal{L}_{rec} + \lambda_{lkd}\mathcal{L}_{lkd} - \lambda_{adv}\mathcal{L}_{adv}$
 9:         $\psi, \pi, \phi \overset{+}{\leftarrow} -\bigtriangledown_{\psi,\pi,\phi} \mathcal{L}$
10:     end for
11: end while
```

Algorithm 2 Generating Step of our proposed architecture

```
Require: $S$ samples $\alpha = \alpha_{s=1}^{S}$ in $(0,1]$; $x_q$ the query sample; $x_s$ potential CE; $f$ classifier; $tol$
    tolerance; $T$ probability of target counterfactual class (0.5 for decision boundary); $\mu_1$ the mean
    of target class in latent space of the label relevant branch
 1: $z_q \leftarrow Encoder(x_q)$
 2: $z_{u,q} \leftarrow Encoder_u(x_q)$
 3: for $\alpha_s$ in $\alpha$ do
 4:     $z_s = (1-\alpha)z_q + \alpha\mu_1$
 5:     $x_s \leftarrow Decoder(z_s, z_{u,q})$
 6:     if $|f(x_s) - T| < tol$ then
 7:         $x_{cf} = x_s$
 8:     end if
 9: end for
10: return $x_{cf}$
```

Step, where we generate a CE of a query sample x_q, We first pass x_q from the base class through the Encoder and the Encoder$_u$ to obtain the sample's embeddings z_q and $z_{u,q}$. Then we get the searched potential z_s by linear interpolation in latent space: $z_s = (1-\alpha)z_q + \alpha\mu_1$ (α in $(0,1]$), where μ_1 is the mean of target class in latent space of the label relevant branch. z_s and $z_{u,q}$ combined are decoded through the Decoder to get x_s, and the pre-trained classifier f assesses its classification score. The counterfactual search stops if it crosses the decision boundary and is within a user (end-user or developer of the algorithm) specified tolerance tol, which is the desired maximum distance from T for the generated counterfactual's classification score. For the experiments in Section 4, the decision boundary T is set to be 0.5. The search is performed by sampling along the line with a finite number of α.

4 Experiments and Evaluation

We compare our method to three other counterfactual methods introduced in Section 2: Gradient Descent Method improved with Prototypes (Prototype) (Van Looveren & Klaise, 2020), CE Generation with Reinforcement Learning (RL) (Samoilescu et al., 2021) and Gradient Descent in the Latent Space of a VAE (GDL) (Balasubramanian et al., 2021) on three datasets: MNIST (LeCun et al., 1998) (image), Adult Dataset (Kohavi & Becker, 1996) (tabular) and Lending Club (Yash, 2020) (tabular). The reason why we choose Prototype and RL is that they are both designed not only for image datasets but also for tabular datasets. Besides, they both use autoencoders to learn the representation to remain close to the data distribution, similar to our design, while they use other constraints to ensure desiderata like sparsity. We use the package ALIBI Klaise et al. (2021) for Prototype and RL to stay close to the original design. GDL is a relatively simple baseline, but it is

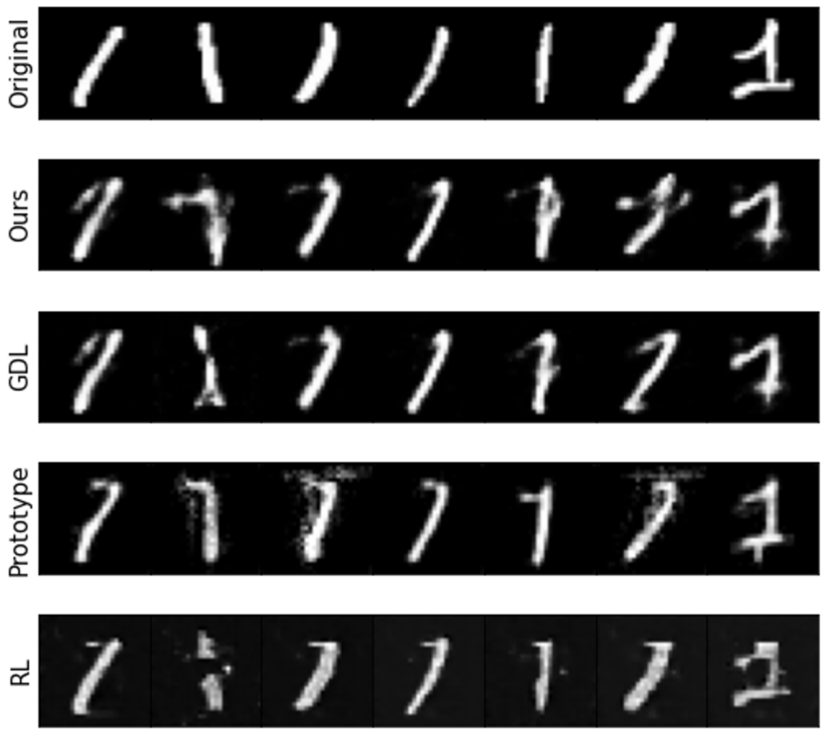

Figure 2: Original and counterfactual samples for MNIST showing from top to bottom: original query images and CEs found via Our Method, GDL, Prototype and RL. Note Our Method exhibits more combination of the characteristics of the original query samples (e.g., tilt and thickness of the strokes) and the target label than the other methods. Our Method and GDL remain (perturbation in latent space) more realistic than the Prototype and RL (perturbation in original space).

similar to our method because both operate in latent space. We implement GDL from scratch since the source code is not available. Our networks are trained on an Intel(r) Core(TM) i7-8700 CPU.

4.1 Measures for Comparison

We evaluate the quality of CEs by the measures taken from literature (Verma et al., 2020; Barr et al., 2021; Wachter et al., 2018): (i) **counterfactual generation time**, time required to find a CE for a given query sample (ii) **validity**, the percentage success in generating CEs that the target labels requested by the users are reached (iii) **proximity**, distance(L_2 norm) from query samples to CEs in original space (iv) **sparsity**, L_1 norm of the change vector in original space (v) **reconstruction loss** – a measure of the CE being in sample. We pass a CE through the autoencoder and measure its loss. A smaller loss indicates closer to the original data distribution because the autoencoder is trained on the same training dataset as the pre-trained classifier. Further details about the metrics used in the experiments can be found in the appendix.

4.2 Datasets, Training and Evaluation

MNIST The MNIST database is a large database of handwritten digits. For MNIST, we adjust the problem as a binary classification problem of predicting ones (our base class) and sevens (our target class) while the original problem setting is multi-calcification. MNIST provides a naturally intuitive visualization of the result of a gradual change from the base class to the target class crossing the decision boundary given a query sample. The counterfactual generated shown in Figure 3 exhibits a combination of characteristics from the query sample (e.g., the tilt of long-stroke) and from the target class (e.g., longer leveled stroke in sevens). Our *Desideratum 1* – generating a CE while keeping the characteristics of the query sample which are not related to the classification prediction is reached. In our problem setting with MNIST, the tilt and length of long strokes of query sample ones are kept during the interpolation process. Table 1 (mean±SD) shows the quality of counterfactual explanations as measured by the above metrics. Our method outperforms the other methods in time, reconstruction, and validity but not in proximity and sparsity (very close to GDL in proximity, though). Our method outperforms Prototype and RL in the time dimension since it searches through the latent space with lower dimensions. It is also more substantial than GDL because it performs interpolation instead of optimization for a single query sample. It indicates that our method is suitable for high-dimensional applications which require intensive computation.

For MNIST, we use a CNN-based network, and we train the model for 20 epochs by stochastic gradient descent using the Adam optimizer and a batch size of 100 on the training dataset. Our model uses two hyperparameters, λ_{adv} and λ_{lkd}, for regularizing the loss functions for the network. We varied λ_{adv} and λ_{lkd} between 0.01 and 1 during training. During the training process, we want the three loss terms $\mathcal{L}_{adv}$, $\mathcal{L}_{lkd}$ and $\mathcal{L}_{cls}$ to remain approximately on the same scale. We found $\lambda_{adv} = 0.05$ and $\lambda_{lkd} = 0.1$ to give us a numerical balance among the three loss terms by checking the testing dataset. We use these values to report our results. The details of the hyper-parameters of the Discriminator are shown in the Appendix.

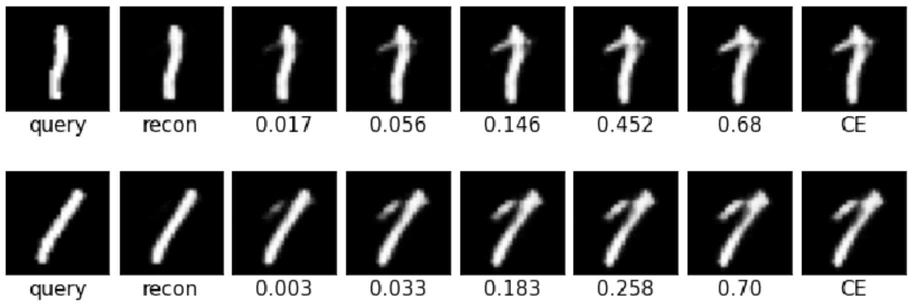

Figure 3: the gradual changes and classification scores for query samples along the interpolation path. Along the paths, the label irrelevant characteristics (e.g., tilt and thickness of the strokes) stay.

Table 1: Summary of metrics for MNIST

Method	time(s)	reconstruction	sparsity	validity(%)	proximity
Our Method	**0.007±0.001**	**0.012±0.003**	2.119 ±0.857	**89.5**	0.193 ±0.075
GDL	1.568 ±0.047	0.247 ±0.048	2.121 ±0.085	70.9	**0.172±0.068**
Prototype	1.345 ±0.095	0.926 ±0.175	**0.014±0.005**	58.7	1.075±0.055
RL	0.085 ±0.012	0.857±0.058	0.015±0.026	60.5	0.834±0.047

ADULT The Adult dataset was drawn from the 1994 United States Census Bureau data. It used personal information such as education level and working hours per week to predict whether an individual earns more or less than \$50,000 per year (Kohavi, 1996). We train a classifier with age, years of education, capital gain, capital loss, hours-per-week as continuous features, and education level as a categorical feature for simplicity (considering mutable and immutable features are not the focus of this paper). The dataset is imbalanced – the instances made less than \$50,000 constitute 25% of the dataset, and the instances made more than \$50,000 constitute 75% of the dataset. To avoid the situation that the accuracy of the classifier for an imbalanced dataset only reflects the distribution of the training dataset, we train a classifier with re-weighting according to the proportion of base and target class. For tabular datasets, more prepossessing is performed compared to image datasets. We normalize the continuous features and use one-hot encoding to deal with the categorical features for the input of the autoencoder. We train the model for 100 epochs by stochastic gradient descent using the Adam optimizer and a batch size of 100. λ_{adv} and λ_{lkd} are set at 0.05 and 0.5. For tabular datasets, we use a multi-layer perceptron-based network. The details of the architectures of the networks for the Adult income dataset are shown in the appendix. Based on the comparison results shown in Table 2, our method outperforms in time and reconstruction dimensions, while

RL is stronger in sparsity and proximity. Intuitively, perturbation in original space without much guidance could quickly end up as adversarial samples (Goodfellow et al., 2015), which provide non-actionable CEs. In contrast, operations in latent space (our method and GDL) could be closer to the original datasets.

Table 2: Summary of metrics for Adult income

Method	time(s)	reconstruction	sparsity	validity(%)	proximity
Our Method	**0.008±0.002**	**0.012±0.005**	0.343±0.127	**90.3**	**0.193±0.045**
GDL	1.568±0.224	0.520±0.135	0.090±0.002	84.2	2.170±0.835
Prototype	7.345±0.784	4.463±0.563	0.014±0.005	75.3	1.075±0.235
RL	1.804±0.112	5.453±0.673	**0.012±0.008**	65.3	0.875±0.132

Lending Club Lending Club is a peer-to-peer lending company that allows individuals to lend to other individuals (Lending Club). This tabular dataset includes whether a borrower defaulted on their loan size, annual income, debt-to-income ratio, FICO score, loan length, etc. We train a classifier to predict default using five continuous and one categorical feature. Since this dataset is also imbalanced – the default class constitutes around 15% of the population, we adjust the loss functions of the classifier and the autoencoder to re-weighted versions based on the proportion of each class. From Table 3, we find that similar to the Adult income dataset, our method is much faster at generating CEs and excels at realism which means it could provide more actionable CEs as suggestions. The training details of this dataset are similar to ADULT. Furthermore, more details are included in the Appendix.

Table 3: Summary of metrics for Lendidng Club default loan

Method	time(s)	reconstruction	sparsity	validity(%)	proximity
Our Method	**0.007±0.001**	**0.132±0.081**	1.713±0.627	**92.1**	3.532±0.258
GDL	3.569 ±0.087	0.142 ±0.046	1.731 ±0.068	66.3	1.652±0.668
Prototype	3.137 ±0.915	0.546 ±0.235	**1.016±0.035**	75.7	1.975±1.045
RL	0.035 ±0.042	0.673±0.124	1.017±0.076	67.5	**1.034±0.127**

5 Conclusion

This paper presents a novel model-agnostic algorithm for finding CEs via linear interpolation in latent space. Our method implements a framework that first disentangles the label relevant and label irrelevant dimensions and then searches in a Gaussian mixture distributed latent space of the label relevant latent dimensions for CEs given a query sample. We demonstrated our method's advantages and disadvantages by comparing it to three similar methods (GDL, Prototype and RL) on three different datasets (MNIST, ADULT income, and Lending Club default loan). We show that our method is faster and provides more valid CEs which are closer to the original dataset (in the dimensions of time, validity and reconstruction). Based on the presented comparison, we suggest that our work could evolve around improving latent representation.

References

Rachana Balasubramanian, Samuel Sharpe, Brian Barr, Jason Wittenbach, and C. Bayan Bruss. Latent-CF: A Simple Baseline for Reverse Counterfactual Explanations. *arXiv:2012.09301 [cs]*, June 2021.

Brian Barr, Matthew R. Harrington, Samuel Sharpe, and C. Bayan Bruss. Counterfactual Explanations via Latent Space Projection and Interpolation. *arXiv:2112.00890 [cs]*, December 2021.

Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 590–601, Red Hook, NY, USA, December 2018. Curran Associates Inc.

Carlos Fernandez, Foster Provost, and Xintian Han. Full Text: Explaining Data-Driven Decisions made by AI Systems: The Counterfactual Approach. https://onikle.com/articles/48299, January 2020.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [cs, stat]*, March 2015.

Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual Visual Explanations. *arXiv:1904.07451 [cs, stat]*, June 2019.

Naama Hadad, Lior Wolf, and Moni Shahar. A Two-Step Disentanglement Method. *arXiv:1709.00199 [cs, stat]*, January 2020.

Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi Explain: Algorithms for Explaining Machine Learning Models, June 2021.

Ron Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pp. 202–207, Portland, Oregon, August 1996. AAAI Press.

Ronny Kohavi and Barry Becker. UCI Machine Learning Repository: Adult Data Set. https://archive.ics.uci.edu/ml/datasets/adult, 1996.

Oran Lang, Yossi Gandelsman, Michal Yarom, Yoav Itzhak Wald, Gal Elidan, Avinatan Hassidim, Bill Freeman, Phillip Isola, Amir Globerson, Michal Irani, and Inbar Mosseri. Explaining in Style: Training a GAN to explain a classifier in StyleSpace. In *Proc. ICCV 2021*, 2021.

Michael T. Lash, Qihang Lin, W. Nick Street, and Jennifer G. Robinson. A budget-constrained inverse classification framework for smooth classifiers. *arXiv:1605.09068 [cs, stat]*, June 2017.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse Classification for Comparison-based Interpretability in Machine Learning. *arXiv:1712.08443 [cs, stat]*, December 2017.

Thai Le, Suhang Wang, and Dongwon Lee. GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model's Prediction. *arXiv:1911.02042 [cs, stat]*, October 2020.

Yann LeCun, Corinna Cortes, and Christopher Burges, J.C. Handwritten Digit Database. `http://yann.lecun.com/exdb/mnist/`, 1998.

Lending Club. Lending Club 2007-2020Q3. `https://www.kaggle.com/datasets/ethon0426/lending-club-20072020q1`, 2020.

Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, December 2017.

Peter Menzies and Helen Beebee. Counterfactual Theories of Causation. In Edward N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2020 edition, 2020.

Martin Pawelczyk, Johannes Haug, Klaus Broelemann, and Gjergji Kasneci. Learning Model-Agnostic Counterfactual Explanations for Tabular Data. *Proceedings of The Web Conference 2020*, pp. 3126–3132, April 2020. doi: 10.1145/3366423.3380087.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv:1602.04938 [cs, stat]*, August 2016.

Pouya Samangouei, Ardavan Saeedi, Liam Nakagawa, and Nathan Silberman. ExplainGAN: Model Explanation via Decision Boundary Crossing Transformations. In *ECCV (10)*, January 2018.

Robert-Florian Samoilescu, Arnaud Van Looveren, and Janis Klaise. Model-agnostic and Scalable Counterfactual Explanations via Reinforcement Learning, June 2021.

Arnaud Van Looveren and Janis Klaise. Interpretable Counterfactual Explanations Guided by Prototypes. *arXiv:1907.02584 [cs, stat]*, February 2020.

Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual Explanations for Machine Learning: A Review. *arXiv:2010.10596 [cs, stat]*, October 2020.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR, March 2018.

Weitao Wan, Yuanyi Zhong, Tianpeng Li, and Jiansheng Chen. Rethinking Feature Distribution for Loss Functions in Image Classification. *arXiv:1803.02988 [cs]*, March 2018.

Yash. Lending Club 2007-2020Q3. https://www.kaggle.com/datasets/ethon0426/lending-club-20072020q1, 2020.