Two-step counterfactual generation for OOD examples

NAWID KESHTMAND, University of Bristol, England

RAUL SANTOS-RODRIGUEZ, University of Bristol, England

JONATHAN LAWRY, University of Bristol, England

Two fundamental requirements for the deployment of machine learning models in safety-critical systems are to be able to detect out-of-distribution (OOD) data correctly and to be able to explain the prediction of the model. Although significant effort has gone into both OOD detection and explainable AI, there has been little work on explaining why a model predicts a certain data point is OOD. In this paper, we address this question by introducing the concept of an OOD counterfactual, which is a perturbed data point that iteratively moves between different OOD categories. We propose a method for generating such counterfactuals, investigate its application on synthetic and benchmark data, and compare it to several benchmark methods using a range of metrics.

Additional Key Words and Phrases: Counterfactual explanations, Explainable OOD detection

1 INTRODUCTION

In recent years, machine learning (ML) models have been increasingly deployed for prediction tasks. However, they still make erroneous predictions when exposed to inputs from an unfamiliar distribution. This poses a significant obstacle to the deployment of ML models in safety-critical applications such as healthcare and autonomous vehicles. Consequently, for applications in these domains, two fundamental requirements for the deployment of ML models are; 1) being able to identify data that is from a different distribution from the data on which the model was trained, which is referred to as out-of-distribution (OOD) detection, outlier detection, or anomaly detection [30]; 2) being able to explain the prediction of the model [24]. There has been significant work on improving the accuracy of OOD detectors although, there has not been much work on explaining why a data point is OOD [20]. As OOD detection algorithms are increasingly used in safety-critical domains, providing explanations for high-stakes decisions has become an ethical and regulatory requirement [26]. Therefore, it is important to develop methods that provide both accurate OOD scores and also provide an explanation of why specific data points are detected as OOD.

OOD detection can be considered a binary classification problem, where a data point can belong either to the in-distribution (ID) class or to the OOD class [4]. Additionally, there are different versions of the OOD detection problem, which are referred to as near-OOD and far-OOD detection [23, 29]. OOD data points that have neither non-discriminative (class-irrelevant) nor discriminative (class-relevant) features are referred to as far-OOD data and are therefore very dissimilar to the ID data. Whilst data points that contain similar non-discriminative features but do not contain discriminative features in common with the ID data are referred to as near-OOD data. As near OOD data points are more similar than far OOD data points to the ID data, near-OOD detection is generally considered to be

more difficult than far-OOD detection [29]. An example of non-discriminative and discriminative features is in images, where there are backgrounds (non-discriminative) plus objects (discriminative). OOD images may have both similar objects and a similar background to ID images.

Results suggest that for near-OOD detection, standard methods of OOD detection such as the Mahalanobis Distance do not perform well [15]. Moreover, the proposed extension of the Mahalanobis Distance which takes into account class labels, and also the distribution fitted to the entire training data, does not work well in the far-OOD setting [23]. Huang et Al recently proposed a solution for OOD detection for both near and far-OOD detection which involves identifying two types of features in the latent space: discriminative and non-discriminative latent features. In their work, discriminative latent features are sufficient for distinguishing near-OOD data, while non-discriminative latent features are used for far-OOD detection [9]. Their approach involves decomposing the learned latent features into the two proposed types of latent features and then summing the Mahalanobis Distance estimated from each type of latent feature. This enabled their approach to be effective at both near and far-OOD detection. This showed the utility of non-discriminative and discriminative latent features for OOD detection. We argue that the categories of near and far-OOD intuitively capture different extents to which OOD data differs from ID data and hence they can play a useful role in the development of explainable OOD detection. To explain the classification of a data point, it can be helpful to investigate the relevant decision boundary between classes. To get insight into why a data point is OOD, we believe it would be beneficial to look at investigating the decision boundary between a near-OOD, far-OOD, and an ID data point. An intuitive way to do this is to look for alternative inputs which are similar but are classified differently, These are referred to as counterfactuals [19, 21]. The usefulness of the counterfactual generated is related to how interpretable is this change in inputs. One way of perturbing the data is by changing several, if not all, latent features simultaneously. However, in this situation, 'information overload' is possible, i.e., too much evidence may cause confusion, reducing trust [14]. We argue that changes in inputs of the following form are more interpretable:

- Changes to individual latent features of the data.
- Changes to individual areas of interest (such as background or foreground object).

This is more likely to be true when the individual latent features are typically selected to be understandable. Therefore, we aim to limit the change in the counterfactual to only changing one area of interest at a time, where the areas we consider in this work are the discriminative and non-discriminative latent features. Thus in this paper, we study the problem of explaining OOD data points using the framework of counterfactual explanations. We introduce a definition of an OOD counterfactual as a perturbed data point that is generated by means of a two-step procedure that iteratively maps the point to move between far-OOD, near-OOD, and the ID categorization of data. We put forth a general framework for generating the counterfactuals and focus on a particular instantiation of the framework. The general framework involves changing the non-discriminative and discriminative latent features separately so that they are more similar to those of ID data. By having two distinct steps we can:

- Generate counterfactuals across the different OOD categories.
- Examine how the (latent and/or input) features of a data point change as it moves from far-OOD to near-OOD and finally to ID.

Understanding how the latent features are transformed by the two steps can help to differentiate between non-discriminative and discriminative latent features as well as indicate the extent to which certain latent features are responsible for the classification of a data point as OOD. The contributions of this paper are therefore as follows:

- Introducing a new method for explainable OOD detection by generating counterfactual data points (Section 3).

- Providing experimental results on tabular datasets showing that the proposed method generates counterfactuals that are more realistic and less likely to be identified as OOD compared to other baselines (Section 5).

2 BACKGROUND

Throughout this paper, scalar mathematical quantities will be represented in lowercase, vectors will be in lowercase and bold, whilst matrix quantities will be capitalized and bold. The term 'class labels' will refer to the labels for the different classes of the ID dataset. The term 'OOD labels' will refer to whether the data point is ID or OOD.

2.1 Counterfactual explanations

There are several different methods for explaining the predictions of an ML model. This includes local vs global approaches, anchors, SHAP, etc [18]. One popular approach is counterfactual explanations. Counterfactuals are used as post-hoc local explanations for individual decisions. For an initial input vector x we look at asking the question why x is classified as class c (or in our case why x is classified as being OOD). The framework of counterfactuals reformulates this "why" question so that it asks "what is the smallest change to x that would result in it belonging to a class that is different from c". A counterfactual explanation is an alternative input vector x', $x' = x + \delta$ which involves x being perturbed in a particular way, where δ is referred to as a perturbation. This generally involves perturbing the input such that we can see the changes to the input features of x which would result in x crossing the decision boundary and being classified as a different class. This enables us to see which input features are important for the classification of the data. 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?' and 'What would the image need to look like in order to be classified as a fire hydrant?' [6, 8].

Depending on the model and the application, the requirements for the generated counterfactual may be different. One desired property of a counterfactual explanation is for the data to be realistic, i.e., a likely scenario for the user in question [25]. Another commonly discussed requirement is sparsity: in general, the fewer features that are changed, the better. Sparse perturbations are usually more interpretable as the change in classification can be attributed to a smaller number of input features. This can make it easier to understand how the perturbation affects the model's prediction. If all the input features change a little, it can be harder to understand how each individual feature affects the model prediction. However, it is also possible to change a subset of the input features that are similar to one another, whilst keeping other input features the same. This could be viewed as imposing block sparsity.

2.2 Counterfactual explanation approaches

There are several different approaches to generating counterfactual explanations. The Counterfactual Instances (CFI) approach [28] works as follows: x is the original instance vector whilst x' is the counterfactual. Let the model be given by q, and let p_t be the target posterior probability of class t (usually given a value of 1). The model's prediction on class t is given by q_t. Let λ be a hyperparameter. This method constructs counterfactual instances x' from an instance x by minimizing the following loss, $\mathcal{L}$, given in Eqn. 1:

$$\mathcal{L}(x', x) = (q_t(x') - p_t)^2 + \lambda L_1(x', x) \tag{1}$$

where L_1 is given by Eqn. 2:

$$L_1(x', x) = \left\| x' - x \right\|_1 \tag{2}$$

The first term rewards the counterfactual for belonging to the desired class, and the use of the norm encourages sparse solutions. Various adaptations of this approach have also been proposed. Dhurandhar et Al train an auto-encoder to reconstruct instances of the training set [5]. They then include the reconstruction error of the perturbed instance as an additional loss term in the objective function. As a result, the perturbed instance lies close to the training data manifold. The Counterfactuals Guided by Prototypes (Proto) approach by Van Looveren and Klaise includes an additional term in the loss function that optimizes for the distance between the latent features of a counterfactual instance z' and a the latent features of 'prototypical' instance of the target class z_{proto} [17]. In doing this, it adds the additional requirement that interpretability also means that the generated counterfactual is close to a representative member of the target class.

3 METHOD

3.1 Overview

By framing OOD detection as a binary classification problem, where a data point, x, can belong to the ID data class or the OOD data class, we can use the framework of counterfactuals explanations to investigate how the data point changes as it moves from OOD to ID. This can be used to explain why a data point is classified as OOD. An example of this could be using a shape classifier for the purpose of classifying between circles and triangles. In the case where an ID dataset consists of red triangles and blue circles, a purple hexagon would be considered as a far-OOD data point. An initial counterfactual that would move the purple hexagon to a near-OOD data point, which is more similar to the ID data, would be to change the color to obtain a blue hexagon. Yet, despite having a similar color to the circles, the shape classifier would not be able to classify the hexagon as it is near-OOD and does not possess the same shape characteristics (sides, corners, or vertices) as a triangle or circle. Therefore, the next step of the OOD counterfactual generation process would be to change the blue hexagon to a blue circle so that it is an ID data point. In the context of explaining why a data point is classified as OOD, we formulate counterfactual generation in terms of a perturbation δ which satisfies Eqn. 3:

$$D_{ID}(f_z(x + \delta)) < D_{OOD}(f_z(x + \delta)) \tag{3}$$

where D_{ID} and D_{OOD} are metrics that represent how far the input is from being ID and OOD respectively, and f_z is a latent feature extractor. But, as we do not have access to the OOD dataset during training time, we instead aim to find δ which satisfies Eqn. 4:

$$D_{ID}(f_z(x + \delta)) < D_{ID}(f_z(x)) \tag{4}$$

The generation of a counterfactual for an OOD data point may involve perturbing the input so that the non-discriminative and/or discriminative latent features are more similar to those found in ID data. If both non-discriminative and discriminative latent features are changed simultaneously this can make the overall transformation harder to interpret, as it would be difficult to tell which of the perturbed latent features are related to non-discriminative latent features and which are related to the discriminative latent features. Additionally, it would be difficult to tell how much each type of latent feature is responsible for the data point being classified as OOD.

Taking inspiration from Huang et Al, we propose that an OOD counterfactual should be able to provide explanations for both near and far-OOD data points [9]. Therefore, it is desired that the OOD data point is perturbed such that both non-discriminative and discriminative latent features of the counterfactual become more similar to that of the ID data but the non-discriminative and discriminative latent features are perturbed in separate steps. In this case, we formalize an OOD counterfactual as a perturbed data point that is generated in a two-step procedure that involves changing the non-discriminative followed by the discriminative latent features, or discriminative followed by non-discriminative

latent features. More formally, a latent vector z is separated into a non-discriminative part, z_n, and discriminative part z_d i.e., $z = (z_n; z_d)$. Then we can satisfy Eqn. 4 by breaking it down into two steps given by Eqn. 5 and 6:

$$D_{ID}(f_z(x + \delta_n)) < D_{ID}(f_z(x)) \tag{5}$$

$$D_{ID}(f_z(x + \delta_d)) < D_{ID}(f_z(x)) \tag{6}$$

where δ_n and δ_d correspond to the perturbation to the non-discriminative and discriminative feature partitions respectively. Eqn. 4 is satisfied by having $\delta = \delta_n + \delta_d$. Performing the counterfactual generation process in this way, explicitly ensures that the generated counterfactuals are similar to the ID data in both the non-discriminative and discriminative latent features. In addition, the counterfactuals can be made to change data points to different degrees of OOD, such as going from far-OOD to ID, near-OOD to ID or even far-OOD to near-OOD. This can improve interpretability since a user can examine the change in a certain step to see whether there is a large or small change to a particular latent feature partition to become more ID. A large change would indicate that the data point was OOD due to that subset of latent features being far away from the ID data. If there was only a single set of latent features, then it may not be possible to find out whether it is the non-discriminative and/or discriminative latent features causing the data point being classified as OOD.

In order to generate the OOD counterfactual, there are three general parts required:

- Obtain latent features of the OOD data, z.
- Decompose z into two partitions, z_d and z_n, using a classifier.
- Perturb the OOD data, x, in a way that makes each latent partition more similar to the latent feature partitions of the ID data.

Therefore a user can generate an OOD counterfactual using any approach which satisfies these three steps.

3.2 Obtaining and decomposing features

Whilst Huang et Al obtain latent features of the data using a pre-trained neural network and then decompose the latent features using an independent cross-entropy (ICE) classification loss, we take an alternative approach [9]. We decided to use one of the baselines from the work of Huang et Al, which was to use Principal Components Analysis (PCA) to obtain the latent features, z. We used PCA as we wanted to obtain latent features which maximize the variance in the data. We additionally separate the principal components into a non-discriminative partition, z_n, and a discriminative partition z_d.

To separate the data into partitions, we examine the conditional entropy of a subset of the principal components Z_{subset} with the class labels Y, $H[Y|Z_{subset}]$, using Eqn. 7 [27]:

$$H[Y|Z_{subset}] = \mathbb{E}_{z_{subset} \sim Z_{subset}} [h(P(y = 1|z_{subset})] \tag{7}$$

where y corresponds to the label for a single data point and z_{subset} corresponds to the latent feature (principal component) subset for a single data point. Also, $h(p) = -p \log p - (1 - p) \log(1 - p)$ is the entropy of a Bernoulli distribution. A variety of different classifiers could be used to obtain the Bernoulli distribution such as a logistic regression classifier. In our work, we used a Quadratic Discriminant Analysis (QDA) classifier [7]. This involved fitting a Gaussian mixture model (GMM) to the latent features of each partition. Then, the minimal subset z_d is obtained using

Eqn. 8:

$$z_d = \min_{Z_{subset}} \mathcal{L}_{Partition}(Z_{subset}, \hat{Y}_{subset}, \hat{Y}^{\complement}_{subset}) \tag{8}$$

where $\mathcal{L}_{Partition}$ is given by Eqn: 9

$$\mathcal{L}_{Partition}(Z_{subset}, \hat{Y}_{subset}, \hat{Y}^{\complement}_{subset}) = H[\hat{Y}_{subset}|Z_{subset}] - H[\hat{Y}^{\complement}_{subset}|Z^{\complement}_{subset}] \tag{9}$$

where $Z^{\complement}_{subset}$ includes all the latent features not in Z_{subset}, $\hat{Y}_{subset}$ and $\hat{Y}^{\complement}_{subset}$ correspond to predictions of a classifier trained on Z_{subset} and a classifier trained on $Z^{\complement}_{subset}$ respectively.

After finding z_d, the non-discriminative partition z_n is then given by the remaining latent features which are not in the subset z_d. The motivation of this loss is that we want to find a Z_{subset} which has a low conditional entropy with Y, as this would suggest that the latent features in Z_{subset} are strongly correlated with Y and would correspond to discriminative latent features. Furthermore, we want to find a $Z^{\complement}_{subset}$ with high conditional entropy as this would mean that the latent features in $Z^{\complement}_{subset}$ are not strongly correlated with Y and therefore likely to correspond to non-discriminative latent features. But, an issue with the loss is that it will tend to separate most of the latent features into z_d as the entropy of z_d will tend to decrease as more latent features are added into z_d and the entropy of z_n will increase as fewer latent features are in z_n. Moreover, it seems that adding additional latent features into Z_{subset} does not lead to $\mathcal{L}_{Partition}$ decreasing by a significant amount. To take into account the bias that more latent features are placed in Z_{subset} generally lowers the entropy, we also add a threshold such that if the difference between two subsets with different cardinality is small, then we choose z_d to be the subset which has the smaller cardinality. We did this as we believe that we should maximize the number of latent features that are present in the z_n as in a realistic classification task, a large number of the features will tend to be unrelated to a classification task. The procedure for generating to separate the features is described further in Appendix .A.

3.3 Perturbing OOD data

After obtaining the latent features and decomposing them into z_n and z_d, counterfactuals can be generated by perturbing OOD data such that the z_n and z_d of the counterfactuals are similar to values of z_n and z_d of the ID data. As Gaussian-based approaches such as the Mahalanobis Distance and maximising the likelihood of a data point with respect to a gaussian performs well at OOD detection, we decided that a counterfactual with a low negative log-likelihood (NLL) under a gaussian mixture model (GMM) obtained from the ID data would mean that the counterfactual is similar to the ID data points. Therefore we believe that minimizing the NLL of data point under a GMM would be an effective loss function to optimize for. Furthermore, as we generated the partitions of the data by using QDA, which is a Gaussian-based classification approach, it would also be beneficial to optimize for a Gaussian-based OOD detection approach. By doing this, there is a better match between the way the partitions were generated and the generation of the counterfactuals. This involved modeling the data distribution of the PCA feature space by a separate Gaussian distribution for z_n and z_d. Counterfactuals are perturbed to minimize the NLL under the GMM defined by non-discriminative partition in the first step of the process and then perturbed to minimize the NLL under the GMM defined by the discriminative partition in the second step of the process. Although, it is possible to optimize for the discriminative partition first and then the non-discriminative partition. The loss for the non-discriminative, $\mathcal{L}_{CF,n}$ and discriminative partitions $\mathcal{L}_{CF,d}$ are given by 10 and 11 respectively:

$$\mathcal{L}_{CF,n} = -\log p_n(f_z(\boldsymbol{x})) \tag{10}$$

$$\mathcal{L}_{CF,d} = -\log p_d(f_z(\boldsymbol{x})) \qquad (11)$$

where p_n and p_d is the density under the non-discriminative and discriminative GMM respectively. Also, f_z is the feature extractor which in this case is the PCA algorithm. We refer to the loss for the non-discriminative and discriminative partitions as the Non-discriminative (Non-dis) and Discriminative (Dis) loss respectively. The process of generating the counterfactuals can be seen in Algorithms 1 and 2 in Appendix B.

4 2D TOY EXPERIMENT

We consider a low dimensional (2D) simulation where we generate two Gaussian distributions with a mean (3,0) and (-3,0) with a tied covariance matrix $\Sigma = 0.5I_2$. There is also an additional OOD distribution which has a mean (0,2) and covariance $\Sigma = 0.3I_2$. PCA with two principal components is then used to extract the latent features which capture the most variance in the ID data. The setup of the toy dataset and the principal components can be seen in Fig. 1. Each latent feature is evaluated for how effective it is at capturing the discriminative information in the ID dataset using the conditional entropy $H[Y|X]$, where $pc1$ is shown to have a $H[Y|pc1] = 1.92e^{-3}$ whilst $pc2$ has a $H[Y|pc2] = 1.00$.
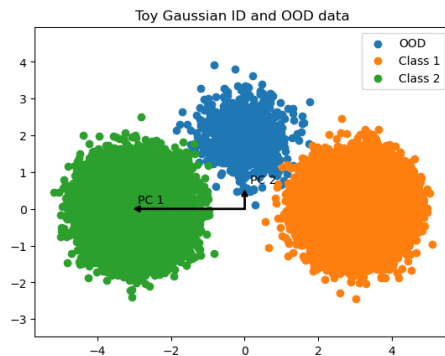

Fig. 1. Toy dataset showing the ID data, OOD data, and Principal components of ID data.

After obtaining the principal components, an OOD counterfactual can be produced using Eqn. 10 and 11. The counterfactual produced depends on whether z_n or z_d is optimized first in the two-step optimization process. Fig. 2 shows the generation of the counterfactuals by first optimizing for z_n then z_d, whilst Fig. 3 shows the generation of the counterfactuals by first optimizing for z_d then z_n. In Fig. 2, it can clearly be seen that the counterfactuals initially move in the vertical direction, which does not aid in the classification of the data points. In the second step, the data points are moved in the horizontal direction which enables them to be easily classified. Having both steps present clearly shows that non-discriminative and discriminative latent features are important for deciding whether a data point is OOD.

For additional clarity of how the counterfactual changes during the optimization process, we have also shown the optimization trajectory taken by a single data point in Fig . 4. In Fig. 4a, an initial point at position (0,2) moves towards the centroid of Class 1 by first moving along the vertical (non-discriminative) direction then along the horizontal (discriminative) direction. Whilst in Fig 4b, the data point moves along the horizontal (discriminative) and then vertical (non-discriminative) directions.

To get an indication of whether optimizing for Eqn. 10 and 11 would generate counterfactuals that are indistinguishable from the ID data, we examined the OOD detection performance when using $\mathcal{L}_{Total} = \mathcal{L}_{CF,n} + \mathcal{L}_{CF,d}$ as a metric for
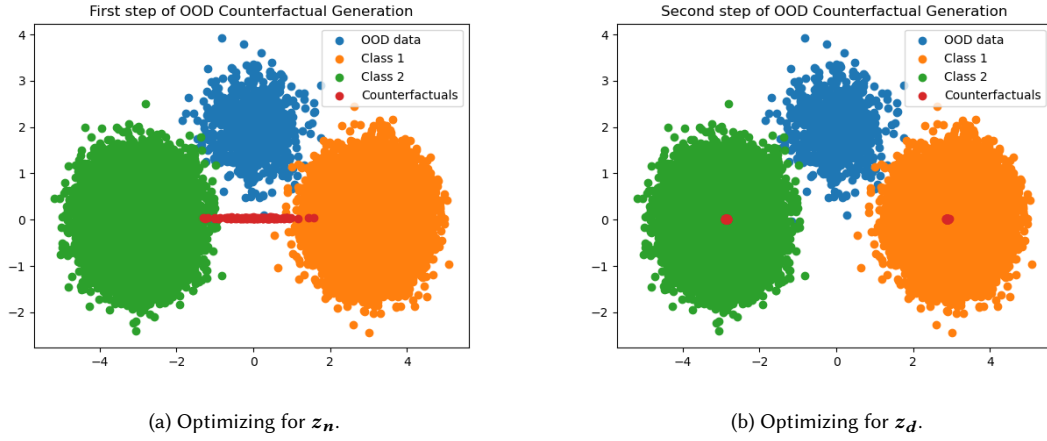
(a) Optimizing for z_n.

(b) Optimizing for z_d.

Fig. 2. OOD counterfactual generation when optimizing for z_n then z_d.

(a) Optimizing for z_d.
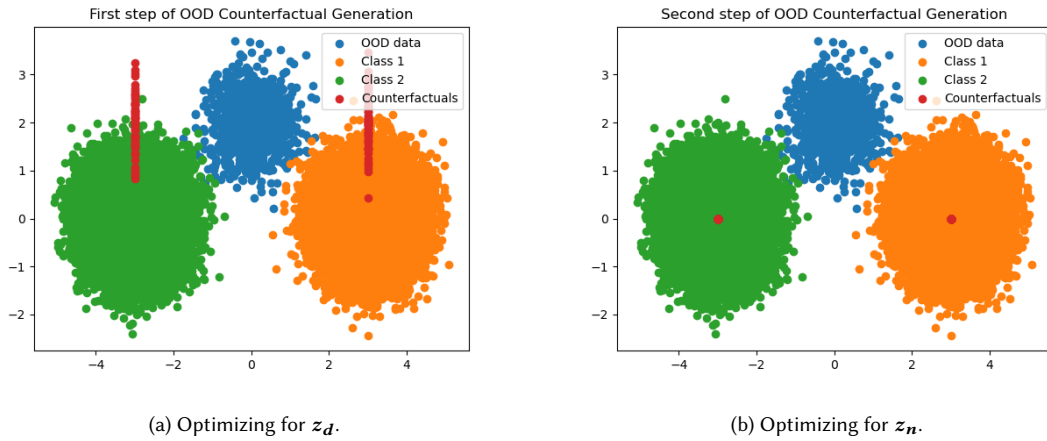
(b) Optimizing for z_n.

Fig. 3. OOD counterfactual generation when optimizing for z_d then z_n.

distinguishing between OOD and ID data. This was done by calculating OOD scores using $\mathcal{L}_{Total}$ and then calculating the AUROC from the OOD scores. The AUROC measures the Area Under the Receiver Operating Characteristic curve. The Receiver Operating Characteristic (ROC) curve plots the relationship between the true positive rate (TPR) and false positive rate (FPR). The area under the ROC curve can be interpreted as the probability that a positive example (ID) will have a higher detection score than a negative example (OOD). The OOD detection performance was compared with the Mahalanobis Distance and the Marginal Mahalanobis Distance. The Marginal Mahalanobis Distance is a variant of the Mahalanobis Distance which fits a single Gaussian to the entire training dataset. From the results in Table. 1, it can be seen that $\mathcal{L}_{Total}$, was the most effective metric at detecting OOD data, and therefore optimizing this objective should lead to the generation of realistic OOD counterfactuals.

(a) Optimizing for z_n and then z_d.

(b) Optimizing for z_d and then z_n.

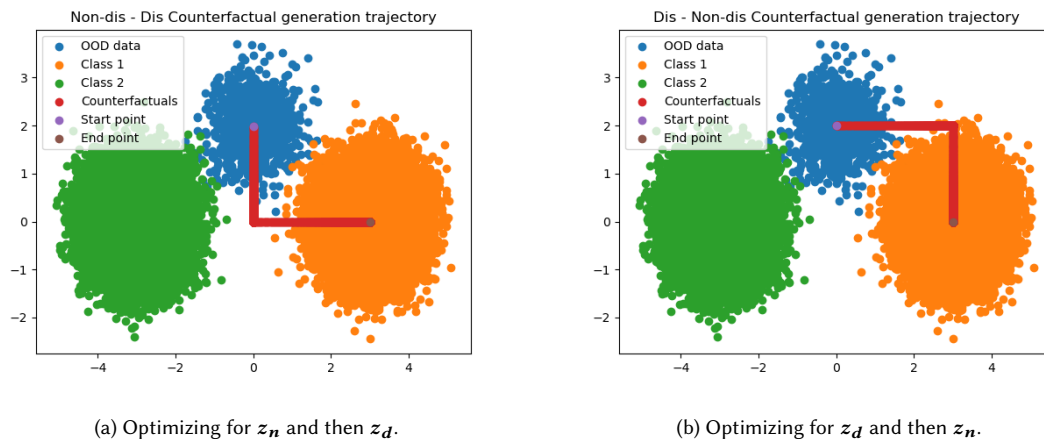Fig. 4. Optimization trajectory of a single OOD data point

OOD approach	AUROC
Mahalanobis Distance	0.908
Marginal Mahalanobis Distance	0.908
Custom metric	0.999

Table 1. Toy OOD detection results

5 TABULAR DATA EXPERIMENT

5.1 Evaluation Metrics

To evaluate the counterfactual generation algorithms, we evaluate how realistic the counterfactuals are and how minimal they have changed from the original data points. To measure the minimality, we calculated the L1 distance between data points, x, and counterfactuals x' given by Eqn. 2 as shown in Section 2. Realism is more difficult to measure because it is poorly defined and in the literature, there are many different definitions used. In this work, we define realism as a counterfactual that is similar to the ID data and therefore similar in the non-discriminative and discriminative latent features of the ID data. Thus, we desire the counterfactuals to have a low NLL for the non-discriminative and the discriminative latent features as given by Eqn. 10 and 11. To measure this, we calculate the AUROC between the ID data (positive examples) and the counterfactual generated from the OOD data (negative examples). In this case, counterfactuals which lead to lower AUROC values are less distinguishable from the ID data and thus more realistic counterfactuals. Therefore, in contrast to the previous use of AUROC in the OOD detection literature, in this work, approaches that have lower AUROC are better for this metric. Additionally, we examined how well the different approaches optimized the Non-dis and Dis loss given by Eqn. 10 and 11.

5.2 Results and Discussion

Approach	Non-dis	Dis	L1	AUROC
OOD CF	1.444	-2.230	11.834	0.860
Proto	1.646	7.798	10.620	0.994
CFI	1.662	8.568	9.598	1.000
CEM	1.726	6.322	8.756	1.000

Table 2. Base metrics-Wine

Approach	Non-dis	Dis	L1	AUROC
OOD CF	-0.260	1.258	3.888	0.052
Proto	-0.070	1.928	5.342	0.364
CFI	2.624	2.290	5.974	0.586
CEM	1.314	2.170	5.580	0.556

Table 3. Base metrics-Diabetes

Approach	Non-dis	Dis	L1	AUROC
OOD CF	0.752	-2.718	8.874	0.382
Proto	0.734	0.370	10.484	0.384
CFI	0.994	2.238	10.930	0.580
CEM	0.968	1.948	10.654	0.606

Table 4. Base metrics-Heart

Approach	Non-dis	Dis	L1	AUROC
OOD CF	-0.538	1.096	3.552	0.312
Proto	0.204	1.284	9.726	0.686
CFI	1.628	1.876	9.922	0.800
CEM	0.972	2.062	9.984	0.896

Table 5. Base metrics-Thyroid

Approach	Non-dis	Dis	L1	AUROC
OOD CF	1.532	-5.118	10.926	0.918
Proto	1.918	-0.078	21.124	0.924
CFI	2.054	4.894	21.956	0.898
CEM	1.986	1.886	22.398	0.946

Table 6. Base metrics-Cardio

We compared the OOD counterfactual (CF) approach with state-of-the-art baselines which are frequently used in the counterfactuals literature. The three baselines are:

- Counterfactual Instances (CFI) [28]
- Counterfactuals Guided by Prototypes (Proto) [17]
- Contrastive Explanation Method (CEM) [5]

To generate counterfactuals with the baselines, we use the authors' implementation in the Alibi package [13]. All the baselines involve training a neural network classifier which contains 2 hidden layers with dimensionality equal to the input dimension and a number of classes equal to the number of ID classes of the data. The classifiers were trained for 500 epochs, using the SGD optimizer with a batch size of 128 and a learning rate of 0.01.

From Tables 2 - 6, it can be seen that the Dis loss is consistently lower for the OOD CF approach compared to the baselines. Moreover, the Non-dis loss is lower for OOD CF for all datasets except the Heart Dataset. This shows that the approach effectively optimizes both of the metrics of interest. Moreover, despite not having any explicit terms in the optimization to minimize the L_1 distance, it can be seen that OOD CF is able to outperform all the baselines in minimizing the L_1 distance for all datasets except for the Wine dataset. This could suggest that separating the counterfactual generation into two distinct steps, makes it so that the counterfactuals produced have a lower L_1 distance to the original instances. In addition, it can be seen that the AUROC when using the OOD CF approach is significantly better for the Wine, Diabetes as well as Thyroid datasets. Whilst for the larger dimensionality datasets (Cardio, and Heart), the AUROC value is similar to the best-performing baseline. This suggests that having lower values

for the metrics of interest leads to more realistic counterfactuals which also have lower L_1 distances compared to the baselines. This is despite the baselines explicitly regularising to minimize the L_1 distance whilst OOD CF does not. The effectiveness of a counterfactual generation approach that produces realistic ID data points is likely related to the degree to which the density of the ID data points is taken into account. The approach which leads to the least realistic counterfactuals is the CFI approach, which does not take into account the density of data points but instead focuses on generating counterfactuals that are far across the decision boundary. The best-performing baseline is the prototype approach which generates counterfactuals that are near to the prototype of the class. This can be seen as an approach that implicitly maximizes the probability density (likelihood) of a counterfactual. Our approach, which generally outperforms the baselines, can be seen to explicitly maximize the probability density (likelihood) of the counterfactual which consequently leads to the most realistic counterfactuals.

5.3 Ablation - Single Gaussian Counterfactual

Approach	Non-dis	Dis	L1	AUROC
OOD CF	1.444	-2.230	11.834	0.860
OOD SG	1.558	-1.602	13.218	0.916

Table 7. Single Gaussian-Wine

Approach	Non-dis	Dis	L1	AUROC
OOD CF	-0.260	1.258	3.888	0.052
OOD SG	-0.254	2.334	7.472	0.190

Table 8. Single Gaussian-Diabetes

Approach	Non-dis	Dis	L1	AUROC
OOD CF	0.752	-2.718	8.874	0.382
OOD SG	0.682	1.446	11.868	0.888

Table 9. Single Gaussian-Heart

Approach	Non-dis	Dis	L1	AUROC
OOD CF	-0.538	1.096	3.552	0.312
OOD SG	-0.450	1.094	10.586	0.232

Table 10. Single Gaussian-Thyroid

Approach	Non-dis	Dis	L1	AUROC
OOD CF	1.532	-5.118	10.926	0.918
OOD SG	1.518	-4.428	19.538	0.978

Table 11. Single Gaussian-Cardio

To examine the effectiveness of separating the latent features into two partitions, we performed an ablation that examined generating counterfactuals that are optimized to have a high likelihood under a single GMM. This approach is referred to as OOD Single Gaussian (SG). From Table 7 - 11, it can be seen that the results of the L_1 distance are consistently lower for the case where there are two separate partitions of the latent features. Moreover, the AUROC values are lower for OOD CF for all datasets except the Thyroid dataset. This could be due to the Thyroid dataset having lower values for the Non-dis and Dis loss for the OOD SG. This further suggests that low values of Non-dis and Dis loss correlate well with the OOD detection performance. Hence, it can be seen that by separating the latent features into two distinct partitions, we are able to obtain counterfactuals that have lower L_1 distances whilst also being more realistic compared to using a single partition.

5.4 Ablation - Only Single Step Counterfactual

Approach	Non-dis	Dis	L1	AUROC
OOD CF	1.444	-2.230	11.834	0.860
OOD SN	1.444	5.658	2.712	0.998
OOD SD	1.754	-2.230	11.188	0.920

Table 12. Single Step-Wine

Approach	Non-dis	Dis	L1	AUROC
OOD CF	-0.260	1.258	3.888	0.052
OOD SN	-0.260	2.104	1.272	0.344
OOD SD	0.822	1.258	3.432	0.162

Table 13. Single Step-Diabetes

Approach	Non-dis	Dis	L1	AUROC
OOD CF	0.752	-2.718	8.874	0.382
OOD SN	0.752	1.420	1.880	0.404
OOD SD	1.068	-2.718	8.560	0.350

Table 14. Single Step-Heart

Approach	Non-dis	Dis	L1	AUROC
OOD CF	-0.538	1.096	3.552	0.312
OOD SN	-0.538	2.008	1.668	0.600
OOD SD	0.488	1.096	2.706	0.640

Table 15. Single Step-Thyroid

Approach	Non-dis	Dis	L1	AUROC
OOD CF	1.532	-5.118	10.926	0.918
OOD SN	1.532	-1.188	3.360	0.868
OOD SD	1.982	-5.118	10.176	0.896

Table 16. Single Step-Cardio

To examine the effectiveness of optimizing for the non-discriminative and discriminative partitions, we performed an ablation that examined generating counterfactuals that are optimized for only one of the two partitions. The approaches are OOD Single Non-dis (SN) and OOD Single Dis (SD) for the case where only the non-discriminative and discriminative partitions are optimized respectively. By comparing the situation between optimizing both partitions and optimizing for a single partition, it can be seen that optimizing for the discriminative partition leads to a much larger change in the L_1 distance compared to optimizing for the non-discriminative partition. Also, the AUROC values for optimizing both partitions compared to a single partition are much better for the lower dimensional datasets (Wine, Diabetes, and Thyroid) as seen from Tables 12, 13 and 15, but are similar for the case of the higher dimensional datasets (Cardio and Heart) as seen in Table 14 and 16. Thus, the benefit of using two steps is more apparent in the case of lower-dimensional datasets.

5.5 Limitations

Our approach is currently suited to problems that have smaller dimensionality which is due to needing to find the minimal subset of latent features that contain discriminative information and this involves examining a different combination of features. PCA, a linear dimensionality reduction approach, is used to obtain the latent features of the data which is effective for representing low dimensional data. But, this may not be effective for larger dimensional datasets where linear dimensionality reduction techniques are insufficient to find latent features that represent the data.

6 RELATED WORK

Counterfactual explanations are used heavily in the interpretability literature. They are useful as post-hoc local explainers which give a contrastive explanation of what needs to change to obtain a desired outcome.

One of the earliest works in counterfactual explanations was the work by Wachter et Al who proposed to perturb data points in the input space to cause it to change class, whilst using additional L_1 regularisation. Following this, there have been many extensions that can be used to enable the generation of counterfactuals with particular properties of interest for several different purposes. One property of interest is to generate counterfactuals that lie close to the data manifold. To encourage this, there has been a significant amount of work on looking at optimizing the counterfactual in the latent space. One of the preliminary works in this area is the work by Balasubramanian et Al [3], where they optimize for the latent features directly by passing it through an autoencoder and then examining the classification loss on the autoencoder output. Other approaches, extend upon this idea. One example is the REVISE method by Joshi et Al which adds an additional regularization loss to encourage sparse changes [10]. Whilst these approaches and our work both encourage the input data point to be close to the data manifold, our work perturbs the input space such that the latent features have a high likelihood, whilst the other approaches directly perturb the latent space to change its classification which can be more computationally efficient. Although, their approaches require training a generative model such as an autoencoder.

Another property desired for the counterfactuals is that they are realistic. To achieve this, there have been several approaches that look at generating counterfactuals with low uncertainty. One approach is the work by Schut et Al which looks at generating counterfactuals which are predicted to have a low entropy by an ensemble of classifiers [25]. In addition, the work done by Antorán et Al calculates the uncertainty of the latent features of a data point using a Bayesian Neural Network and looks at changing the latent features such that the data point becomes more similar to a target ID class whilst also lowering the uncertainty [1]. These works are similar to ours as they focus on the intersection of explainability as well as uncertainty estimation. However, the purpose of these works focuses on ensuring that a data point will change class without becoming OOD whilst our work focuses on moving an OOD data point to become ID. On a practical level, these works differ from our approach as they look at making perturbation to the input in a single step, whilst our approach looks at making it change in two steps. Additionally, these approaches use an ensemble of classifiers or a Bayesian Neural Network to measure uncertainty whilst our work uses a GMM.

Another work that ensures that the counterfactual has low uncertainty is the work done by Poyiadzi et Al who introduced the FACE approach [21]. The purpose of FACE is to provide users with recourse which is possible whilst there also being a path to achieve the change. This is achieved by generating counterfactuals in high-density regions of data space, whilst traversing a high-density path during its optimization trajectory. Both approaches constrain the path in generating counterfactuals, but, it is done in different ways. FACE ensures traversing along a high-density path whilst our work performs the generation in two distinct steps where we aim to have a high likelihood for both partitions at the end of the two steps.

The approach most similar to our work is the DeDUCE approach by Holtgen et Al [8]. The purpose of DeDUCE is to generate counterfactuals in a computationally efficient manner. DeDUCE looks at generating counterfactuals with minimal epistemic uncertainty by perturbing data to maximize the log-likelihood where the latent feature space is modelled using a GMM. Both our approach and their approach uses a GMM to make it so that the counterfactual has a high density. However, our approach uses two different GMM (one for each type of partition) whilst they use a single partition.

7 CONCLUSION

In this work, we focused on the problem of explaining why data points are classified as OOD. This was done through the use of counterfactuals which used the concept of non-class discriminative and discriminative latent features to iteratively map OOD data to different OOD categories such as far-OOD, near-OOD and ID. This involved:

- Introducing the concept of an OOD counterfactual.
- Proposing a framework for explaining OOD data points through the generation of an OOD counterfactual.

The general framework involved obtaining latent features of the data and then decomposing the latent features of the data into a non-discriminative partition, z_n, and a discriminative partition z_d. Following this, an OOD counterfactual is generated by performing a two-step process where each step in the process perturbs a data point in a way that makes one of the two types of latent features more similar to the ID data. We examined the effectiveness of the OOD counterfactual on the higher dimensional tabular datasets. Compared to the baselines, it was seen that the OOD CF approach was:

- Able to achieve lower Dis and Non-dis loss.
- Generate more realistic counterfactuals shown by the lower AUROC values.

From the ablations, it could be seen that:

- Optimizing using two distinct steps leads to lower L_1 values and more realistic counterfactuals.
- Optimizing both steps compared to a single step is more beneficial for the lower dimensional tabular datasets.

Future work will involve working on the first two parts of the general framework. It would be desirable to develop an approach of representation learning which is both able to capture the variance in the data as well as partition the data into two distinct categories in an end-to-end manner. Additional work could also focus on partitioning the latent features into more than two categories. We hope our work will encourage others to focus on the problem of the explanation of OOD data points through the perspective of non-discriminative and discriminative latent features.

REFERENCES

[1] Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. 2020. Getting a clue: A method for explaining uncertainty estimates. *arXiv preprint arXiv:2006.06848* (2020).

[2] Diogo Ayres-de Campos, Joao Bernardes, Antonio Garrido, Joaquim Marques-de Sa, and Luis Pereira-Leite. 2000. SisPorto 2.0: a program for automated analysis of cardiotocograms. *Journal of Maternal-Fetal Medicine* 9, 5 (2000), 311–318.

[3] Rachana Balasubramanian, Samuel Sharpe, Brian Barr, Jason Wittenbach, and C Bayan Bruss. 2020. Latent-CF: a simple baseline for Reverse Counterfactual Explanations. *arXiv preprint arXiv:2012.09301* (2020).

[4] Peng Cui and Jinjia Wang. 2022. Out-of-Distribution (OOD) Detection Based on Deep Learning: A Review. *Electronics* 11, 21 (2022), 3500.

[5] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. 2018. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Advances in neural information processing systems* 31 (2018).

[6] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Counterfactual visual explanations. In *International Conference on Machine Learning*. PMLR, 2376–2384.

[7] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer.

[8] Benedikt Höltgen, Lisa Schut, Jan M Brauner, and Yarin Gal. 2021. DeDUCE: Generating Counterfactual Explanations Efficiently. *arXiv preprint arXiv:2111.15639* (2021).

[9] Haiwen Huang, Joost van Amersfoort, and Yarin Gal. 2021. Decomposing Representations for Deterministic Uncertainty Estimation. *arXiv preprint arXiv:2112.00856* (2021).

[10] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615* (2019).

[11] Dennis Kibler, David W Aha, and Marc K Albert. 1989. Instance-based prediction of real-valued attributes. *Computational Intelligence* 5, 2 (1989), 51–57.

[12] Kivanc Kilicer, Istanbul Bahcesehir, and Adem Karahoca. 2005. Wine recognition analysis by using data mining. In *9th International Research/Expert Conference,'Trends in the Development of Machinery and Associated Technology', TMT 2005*. Citeseer, 26–30.

[13] Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. 2021. Alibi Explain: Algorithms for Explaining Machine Learning Models. *Journal of Machine Learning Research* 22, 181 (2021), 1–7. http://jmlr.org/papers/v22/21-0017.html

[14] Owen Lahav, Nicholas Mastronarde, and Mihaela van der Schaar. 2018. What is interpretable? using machine learning to design interpretable decision-support systems. *arXiv preprint arXiv:1811.10799* (2018).

[15] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems* 31 (2018).

[16] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[17] Arnaud Van Looveren and Janis Klaise. 2021. Interpretable counterfactual explanations guided by prototypes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 650–665.

[18] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[19] Christoph Molnar. 2020. *Interpretable machine learning*.

[20] Guansong Pang and Charu Aggarwal. 2021. Toward explainable deep anomaly detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4056–4057.

[21] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. 2020. FACE: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 344–350.

[22] John Ross Quinlan, Paul J Compton, KA Horn, and Leslie Lazarus. 1987. Inductive knowledge acquisition: a case study. In *Proceedings of the Second Australian Conference on Applications of expert systems*. 137–156.

[23] Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. 2021. A simple fix to mahalanobis distance for improving near-ood detection. *arXiv preprint arXiv:2106.09022* (2021).

[24] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys* 16 (2022), 1–85.

[25] Lisa Schut, Oscar Key, Rory Mc Grath, Luca Costabello, Bogdan Sacaleanu, Yarin Gal, et al. 2021. Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1756–1764.

[26] Harry Surden. 2021. Machine learning and law: an overview. *Research Handbook on Big Data Law* (2021), 171–184.

[27] MTCAJ Thomas and A Thomas Joy. 2006. *Elements of information theory*. Wiley-Interscience.

[28] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.

[29] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, et al. 2020. Contrastive training for improved out-of-distribution detection. *arXiv preprint arXiv:2007.05566* (2020).

[30] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. 2021. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334* (2021).

A FEATURE PARTITIONING ALGORITHMS

The overall procedure to obtain the latent feature partition consisted of many steps. Firstly, for each cardinality, we obtained the set of latent features partitions with the lowest value of $\mathcal{L}_{Partition}$ using Eqn. 9. This set of different latent feature partitions is denoted as $z_{car-min}$ and has associated loss values $L_{Partition-car-min}$. We obtain the mean and standard deviation of $L_{Partition-car-min}$ to normalize it to obtain new values denoted as $L_{Normalized-Partition-car-min}$. We then obtained the lowest value of $L_{Normalized-Partition-car-min}$ which we refer to as $L_{Normalized-min}$ and look choose the smallest cardinality latent feature set with a $L_{Normalized-Partition-car-min}$ within 10% of $L_{Normalized-min}$ as our z_d.

B COUNTERFACTUAL GENERATION ALGORITHMS

Algorithm 1 Counterfactual generation - Natural language

Input: Test sample x, target class t, training data X_{train}
Output: counterfactual x'
Apply PCA on X_{train} to obtain latent features Z_{train}
Separate Z_{train} to obtain non-discriminative latent features $Z_{n,train}$ and discriminative latent features $Z_{d,train}$.
Obtain the discriminative features for the target class t, $Z_{d,train}^t$
Fit a GMM to $Z_{n,train}$ and $Z_{d,train}^t$
Decide the order of feature partitions to optimize in the two-step process
$x' \leftarrow x$
for each partition i $\in$ Range(2) **do**
    $k \leftarrow 0$
    while $k <$ max iter **do**
        Compute the loss, $\mathcal{L}_{CF} = -\log p_i(f_z(x))$
        compute gradient g in input space, $g = \nabla_x(\mathcal{L}_{CF})$
        $x' = x' - (\alpha \cdot g)$
        $k \leftarrow k + 1$
    end while
end for
return x'

Algorithm 2 Counterfactual generation - Pseudocode

1: **Input:** Test sample x, target class t, training data X_train, training labels Y_train, discriminate dimensions dis_dims, non-discriminate dimensions non_dis_dims, partition order, $partitions$
2: **Output:** $counterfactual$
3: $Z\_train = \text{PCA}(X\_train)$
4: $Z\_n\_train = Z\_train[:,non\_dis\_dims]$
5: $Z\_d\_train = Z\_train[:,dis\_dims]$
6: $class\_means = \text{mean}(Z\_d\_train[:,dis\_dims][Y\_train == c], \text{dim}=0)$ for c in unique(Y_train)])
7: $class\_covariance = \text{covariance}(Z\_d\_train[:,dis\_dims][Y\_train == c], \text{dim}=0)$ for c in unique(Y_train)])
8: $dis\_gmm = \text{normal}(class\_means, class\_covariance)$ ▷ Fit normal distribution using mean and covariance
9: $non\_dis\_mean = \text{mean}(Z_{n,train}[:,dis\_dims]$
10: $non\_dis\_covariance = \text{covariance}(Z\_n\_train)$
11: $non\_dis\_gmm = \text{normal}(non\_dis\_means, non\_dis\_covariance)$ ▷ Fit normal distribution using mean and covariance
12: $gmms = \text{order}(non\_dis\_gmm, dis\_gmm, partitions)$ ▷ Order the partitions which is used in the first and second step
13: $x' \leftarrow x$
14: **for** $gmm \in gmms$) **do**
15:     $k = 0$
16:     **while** $k < $ max iter **do**
17:         $loss = \text{NLL}(\text{PCA}(x)))$ ▷ Calculate negative log likelihood
18:         $g = \text{backpropagate}(loss)$ ▷ Backpropagate to calculate gradient
19:         $counterfactual = counterfactual - (\alpha \cdot g)$
20:         $k = k + 1$
21:     **end while**
22: **end for**
23: **return** $counterfactual$

C   DATASETS

Thyroid: The problem is to determine whether a patient referred to the clinic is hypothyroid. Therefore three classes are built: normal (not hypothyroid), hyperfunction and subnormal functioning [22]. Furthermore, we defined the subnormal functioning class as the outlier class and the other two classes are inliers, because subnormal functioning has the lowest number of data points present in the dataset.

Diabetes: The datasets consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on. The outcomes are diabetic or non-diabetic [16]. To define OOD data in this case, we make it so that all data points with the age attribute above the upper quartile are OOD.

Heart: A dataset where each row is composed of 14 attributes. The outcome, in this case, is whether an individual has heart disease or not [11]. To define OOD data in this case, we make it so that all data points with the exercise-induced angina variable set to 1 are OOD.

Wine: These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines [12]. To define OOD data in this case, we make it so that all data points classified as class 2 are OOD.

Cardiotocography: 2126 fetal cardiotocograms (CTGs) were automatically processed and the respective diagnostic features were measured. The CTGs were also classified by three expert obstetricians and a consensus classification label was assigned to each of them. Classification was with respect to a fetal state (N, S, P) [2]. To define OOD data in this case, we make it so that all data points classified as P are OOD.