The AAAI 2023 Workshop on Representation Learning for Responsible Human-Centric AI (R^{2}HCAI)

Model-agnostic and Scalable Counterfactual Explanations via Reinforcement Learning

Robert-Florian Samoilescu[1,2] * **Arnaud Van Looveren**[1] **Janis Klaise**[1]

[1] Seldon Technologies Ltd. [2] University Politehnica of Bucharest
rfs@seldon.io, avl@seldon.io, jk@seldon.io

Abstract

Counterfactual instances are a powerful tool to obtain valuable insights into automated decision processes, describing the necessary minimal changes in the input space to alter the prediction towards a desired target. Most previous approaches require a separate, computationally expensive optimization procedure per instance, making them impractical for both large amounts of data and high-dimensional data. Moreover, these methods are often restricted to certain subclasses of machine learning models (e.g. differentiable or tree-based models). In this work, we propose a deep reinforcement learning approach that transforms the optimization procedure into an end-to-end learnable process, allowing us to generate batches of counterfactual instances in a single forward pass. Our experiments on real-world data show that our method i) is model-agnostic (does not assume differentiability), relying only on feedback from model predictions; ii) allows for generating target-conditional counterfactual instances; iii) allows for flexible feature range constraints for numerical and categorical attributes, including the immutability of protected features (e.g. gender, race); iv) is easily extended to other data modalities such as images.

1 Introduction

Machine learning models are widely adopted across industries. However, the black-box nature of machine learning systems makes it difficult to build trust in algorithmic decision making, especially in sensitive and safety critical areas where humans are directly affected. For example, when using machine learning as part of the decision making process for loan approvals, university admissions or employment applications, simple rejection feedback can be misinterpreted or raise serious concerns regarding the ability of an institution to provide equal opportunities to all applicants.

Counterfactual instances (a.k.a. *counterfactual explanations, counterfactuals*) (Wachter, Mittelstadt, and Russell 2017; Karimi et al. 2020; Stepin et al. 2021) are a powerful tool to obtain insight into the underlying decision process of a black-box model, describing the necessary minimal changes in the input space to alter the prediction towards a desired target. To be of practical use, a counterfactual should be *sparse*—close (using some distance measure) to the original instance—and indistinguishable from real instances, that is, it should be *in-distribution*. Thus, for a loan application system that currently outputs a rejection for a given individual, a counterfactual explanation should suggest plausible minimal changes in the feature values that the applicant could perform to get the loan accepted leading to actionable recourse (Joshi et al. 2019).

	IN	CF	Condition
Age	40	40	[40, 45]
Workclass	Private	Private	{Private, Federal-gov, Self-emp-inc}
Education	High School grad	Masters	{High School grad, Bachelors, Masters}
Marital Status	Married	Married	{Married}
Occupation	Sales	White-Collar	{Sales, White-Collar, Admin}
Relationship	Husband	Husband	{Husband}
Race	White	White	{White}
Sex	Male	Male	{Male}
Capital Gain	0	0	[0, 0]
Capital Loss	0	0	[0, 0]
Hours per week	60	60	[60, 60]
Country	Latin-America	Latin-America	{Latin-America}
Prediction	≤ $50/y	> $50k/y	

Figure 1: Conditional counterfactual instance on Adult dataset. IN—original instance, CF—counterfactual instance, Condition—feature range/subset constraints. Grayed out feature values correspond to immutable features. Highlighted in red, the feature changes required to alter the prediction of a black-box model (here from $\leq \$50k/y$ to $> \$50k/y$).

A desirable property of a method for generating counterfactuals is to allow feature conditioning. Real-world datasets include immutable features such as gender or race, which should remain unchanged throughout the counterfactual search procedure. A natural extension of immutability is to restrict a feature to a subset or an interval of values. Following the same loan application example, a customer might be willing to improve their education level from a *High-school graduate* to *Bachelor's* or *Master's*, but not further. Similarly, a numerical feature such as *Age* should only increase for a counterfactual to be actionable. To enable such feature conditioning, we propose to use a conditioning vector to guide the generation process. An example of this scenario on the Adult dataset (Dua and Graff 2017) is shown in Figure 1 where we enforce feature conditioning and immutability. Our method successfully flips the classification label by suggesting a change in the *Education* level, from *High School grad* to *Master's*, and in the *Occupation*, from *Sales* to *White-Collar*, producing an actionable counterfac-

*Work done during an internship at Seldon.

tual explanation with respect to the user-specified feature constraints.

Previous approaches to finding counterfactual instances have focused primarily on iterative procedures by minimizing an objective function that favors sparse and in-distribution results. Some methods are only suitable for differentiable models since they require access to model gradients. Moreover, a separate optimization procedure per instance is computationally expensive, making them impractical for both large amounts of data and high-dimensional data.

Our contributions. We propose a deep reinforcement learning approach that transforms the optimization procedure into a learnable process, allowing us to generate batches of counterfactual instances in a single forward pass. Our training pipeline is model-agnostic and relies only on prediction feedback by querying the black-box model. Furthermore, our method allows target and feature conditioning, and is applicable to a variety of tasks such as multi-class classification and regression. We focus primarily on the tabular data setting and evaluate our method on multiple types of models and datasets, for which we obtain competitive results against existing algorithms. Additionally, we show that our approach is flexible and easily extendable to other data modalities such as images. In summary, our contributions are as follows:

- Model-agnostic, target-conditional framework primarily focused on heterogeneous tabular datasets.
- Flexible feature range constraints for numerical and categorical features.
- Fast counterfactual generation process since no iterative optimization procedure is required when producing counterfactual instances.
- Easily extendable framework to other data modalities.

2 Related work

Counterfactual explanation methods have recently seen an explosive growth in interest by the academic community. We refer the reader to the recent surveys of Karimi et al. (2020); Stepin et al. (2021) for a more comprehensive review of the area.

Most counterfactual explanation methods perturb the original instance under proximity constraints until the desired model prediction is achieved (Wachter, Mittelstadt, and Russell 2017; Mothilal, Sharma, and Tan 2020; Cheng, Ming, and Qu 2020) or conduct a heuristic search (Martens and Provost 2014; Laugel et al. 2017). These approaches require a separate optimization process for each explained instance and often result in out-of-distribution counterfactuals when the perturbations are applied in the (high-dimensional) input space. Attempts to generate more realistic, in-distribution counterfactuals include the addition of auxiliary losses such as the counterfactual's reconstruction error using a pre-trained autoencoder (Dhurandhar et al. 2018) or by guiding the perturbations towards class-specific prototypes (Van Looveren and Klaise 2019). Other methods leverage pre-trained generative models such as conditional GANs (Mirza and Osindero 2014; Liu et al. 2019) or variational autoencoders (Joshi et al. 2019) to improve the realism of the proposed counterfactuals.

Many of the previous methods rely on gradient-based optimization which restricts the application to differentiable models. In practice we are often only able to query the model, thus we are interested in the model-agnostic, black-box setting. This is especially relevant for (mixed-type) tabular data where non-differentiable models such as Random Forests (Statistics and Breiman 2001) or XGBoost (Chen and Guestrin 2016) remain very popular. LORE (Guidotti et al. 2018) is a model agnostic method which, similar to LIME (Ribeiro, Singh, and Guestrin 2016), learns a local interpretable surrogate model around the instance of interest. The decision rules learned by the local model are used to change the original prediction towards a desired target. Similarly, White and Garcez (2019) extract local rules from decision trees to generate counterfactual explanations. DiCE (Mothilal, Sharma, and Tan 2020) aims to generate a diverse set of black box counterfactual explanations by using determinantal point processes (Kulesza and Taskar 2012) under both proximity and feature level user constraints. The search process can be done at random or via a genetic algorithm. Hashemi and Fathi (2020) follow the same paradigm and refine the sampling procedure to favour in-distribution counterfactuals. Sharma, Henderson, and Ghosh (2019) also use evolutionary strategies which allow for feature conditioning to obtain counterfactual explanations. Minimum Observable (MO) (Wexler et al. 2019) returns the closest instance in the training set which belongs to the target class. The distance is measured by a combination of the $\mathcal{L}_1$ and $\mathcal{L}_0$ norms of respectively the standardized numerical and categorical features.

The methods mentioned so far either need access to a training set when generating the explanation or rely on a time consuming iterative search procedure for every explained instance. This bottleneck can be resolved by using class-conditional generative models which are able to generate batches of sparse, in-distribution counterfactual instances in a single forward pass for various data modalities such as images, time series or tabular data (Van Looveren et al. 2021; Oh, Yoon, and Suk 2020; Mahajan, Tan, and Sharma 2019). However, end-to-end training of the counterfactual generator requires backpropagation of the gradients through the model, limiting the applicability to differentiable models. We propose a model-agnostic counterfactual generator able to generate batches of explanations with a single prediction. The generator is trained using reinforcement learning (RL) and is able to learn solely from sparse model prediction rewards. Our method allows for flexible feature conditioning and can easily be extended to various data modalities.

3 Reinforcement learning background

Consider a standard model-free RL problem where an agent learns an optimal behavior policy by repeated interactions with the environment, intending to maximize the expected cumulative reward received from the environment. One approach to learn an optimal policy is to approximate the Q-

function, which estimates the reward that an agent will obtain by applying a particular action in a given state. Thus, if we know the optimal action-value function $Q^*(s, a)$, in a state s and for any action a, then the optimal policy is given by $a^*(s) = \arg\max_a Q^*(s, a)$.

For a discrete action space, the maximization procedure requires the evaluation of the state-action pair for the available actions and retrieving the action that maximizes the Q function for a given state. On the other hand, for a continuous action space, an exhaustive search is impossible and alternatively maximizing the Q function through an iterative procedure can be a computational bottleneck. Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2016) addresses those issues by interleaving a state-action function approximator Q (the critic) of $Q^*(s, a)$ with learning an approximator μ (the actor) for the optimal action $a^*(s)$. The method assumes that the critic is differentiable with respect to the action argument allowing to optimize the actor's parameters efficiently through gradient-based methods. DDPG approximates the solution of the expensive maximization procedure through a learnable process, $\mu(s) \approx \arg\max_a Q(s, a)$.

Similar to other deep Q-learning algorithms (Mnih et al. 2013), DDPG uses a replay buffer for sample efficiency. Instead of immediately discarding the current experience, the method stores it in the replay buffer for later use. Each training phase consists of sampling a batch of experiences uniformly and taking a gradient step. Updates based on old experience prevent overfitting and increase training stability.

4 Method

4.1 Problem statement

A counterfactual explanation of a given instance represents a sparse, in-distribution example that alters the model prediction towards a specified target. Following the notation of Van Looveren et al. (2021), let x be the input instance, M a black-box model, $y_M = M(x)$ the model prediction on x and y_T the target prediction. Our goal is to produce a counterfactual instance $x_{CF} = x + \delta_{CF}$ where δ_{CF} represents a sparse perturbation vector such that $y_T = M(x_{CF})$. Instead of solving an optimization problem for each input instance, we train a generative model which models the counterfactual instances x_{CF} directly and allows for feature level constraints via an optional conditioning vector c. A conditional counterfactual explanation x_{CF} therefore depends on the tuple $s = (x, y_M, y_T, c)$.

Since we don't assume the model M to be differentiable we train the counterfactual generator using RL with s representing the state and the actor network $\mu(s)$ taking the role of the counterfactual generator. This model-agnostic training pipeline is compatible with various data modalities and only uses sparse model prediction feedback as a reward. For a classification model returning the predicted class label the reward can be defined by an indicator function, $R = \mathbb{1}(M(x_{CF}) = y_T)$. The reward for a regression model, on the other hand, is proportional to the proximity of $M(x_{CF})$ to the regression target y_T.

Instead of directly modelling the perturbation vector δ_{CF} in the potentially high-dimensional input space, we first train an autoencoder. The weights of the encoder are frozen and μ applies the counterfactual perturbations in the latent space of the encoder. The pre-trained decoder maps the counterfactual embedding back to the input feature space. Since μ operates in the continuous latent space we use the sample efficient DDPG (Lillicrap et al. 2016) method. For the remainder of the paper, we denote by enc and dec the encoder and the decoder networks, respectively.

4.2 Target conditioning

The counterfactual generation process generalizes to a variety of tasks and target types such as multi-class classification and regression. For classification tasks, we condition on a one-hot encoded representation of the target prediction y_T as well as on the predicted label y_M on the input instance x. While conditioning solely on the class labels represents the most generic use case since it does not require the model output to be probabilistic (e.g. SVM's (Cortes and Vapnik 1995)), we could also condition the counterfactual generator on soft prediction targets such as the probability of the target class or the full prediction distribution over all the classes. Note that our approach does not require the ground truth labels and relies only on querying the black-box model. During training, we sample the target prediction y_T uniformly at random from all possible labels, including $y_T = y_M$ for the identity mapping.

4.3 Feature conditioning

In many real-world applications, some of the input features are immutable, have restricted feature ranges or are constrained to a subset of all possible feature values. These constraints need to be taken into account when generating actionable counterfactual instances. For instance *Age* and *Marital Status* could be features in the loan application example. An actionable counterfactual should however only be able to increase the numerical *Age* feature and keep the categorical *Marital Status* feature unchanged. To achieve this we condition the counterfactual generator on a conditioning vector c, constructed by concatenating per-feature conditioning vectors, defined as follows:

- for a numerical feature with a minimum value a_{min} and a maximum value a_{max}, we append to c the values $-p_{min}$, p_{max}, where $p_{min}, p_{max} \in [0, 1]$. The range $[-p_{min}, p_{max}]$ encodes a shift and scale-invariant representation of the interval $[a - p_{min}(a_{max} - a_{min}), a + p_{max}(a_{max} - a_{min})]$, where a is the original feature value. During training, p_{min} and p_{max} are sampled $\sim Beta(2, 2)$ for each unconstrained feature. Immutable features correspond to setting $p_{min} = p_{max} = 0$. Features allowed to only increase or decrease correspond to setting $p_{min} = 0$ or $p_{max} = 0$, respectively. Following the example in Figure 1, allowing the *Age* feature to increase by up to 5 years is encoded in the conditional vector c by taking $p_{min} = 0, p_{max} = 0.1$, assuming a minimum age of 10 and a maximum age of 60 years in the training set $(5 = 0.1 \cdot (60 - 10))$.

- for a categorical feature of cardinality K we condition the subset of allowed feature values through a binary mask of dimension K. When training the counterfactual generator, the mask values are sampled $\sim Bern(0.5)$. For immutable features, only the original input feature value is set to one in the binary mask. Following the example in Figure 1, the immutability of the *Marital Status* having the current value *Married* is encoded through the binary sequence $(1, 0, 0)$, given an ordering of the possible feature values {*Married*, *Unmarried*, *Divorced*}.

Following the decoding phase, as part of post-processing (denoted by a function *pp*), the numerical values are clipped within the desired range, and categorical values are conditionally sampled according to their masking vector. This step ensures that the generated counterfactual respects the desired feature conditioning before passing it to the model. Note that our method is flexible and allows non-differentiable post-processing such as casting features to their original data types (e.g. converting a decoded floating point *Age* to an integer: 40 = int(40.3)) and categorical mapping (e.g., *Marital Status* distribution/one-hot encoding to the *Married* value) since we rely solely on the sparse model prediction reward.

4.4 Training procedure

The DDPG algorithm requires two separate networks, an actor μ and a critic Q. Given the encoded representation of the input instance $z = enc(x)$, the model prediction y_M, the target prediction y_T and the conditioning vector c, the actor outputs the counterfactual's latent representation $z_{CF} = \mu(z, y_M, y_T, c)$. The decoder then projects the embedding z_{CF} back to the original input space, followed by optional post-processing.

We restrict the components of the embedding representation to $[-1, +1]$ through a *tanh* non-linearity. During the first N steps the latent components are sampled uniformly from $[-1, +1]$ to encourage exploration. Afterwards, Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.1)$ is added to the actor's output. The experience is stored in a replay buffer from which we uniformly sample during the training phase.

The training step consists of simultaneously optimizing the actor and critic networks. The critic regresses on the reward R determined by the model prediction, while the actor maximizes the critic's output for the given instance. The actor also minimizes two objectives to encourage the generation of sparse, in-distribution counterfactuals. The sparsity loss $\mathcal{L}_{sparsity}$ operates on the decoded counterfactual x_{CF} and combines the $\mathcal{L}_1$ loss over the standardized numerical features and the $\mathcal{L}_0$ loss over the categorical ones. The consistency loss $\mathcal{L}_{consist}$ (Zhu et al. 2017) aims to encode the counterfactual x_{CF} back to the same latent representation where it was decoded from, formally define as $\mathcal{L}_{consist} = \|z_{CF} - enc(pp(x_{CF}, c))\|_2^2$.

Note that our setup is equivalent to a Markov decision process with a one-step horizon, which does not require bootstrapping to compute the critic's target, increasing stability and simplifying the training pipeline. A full description of the training procedure is presented in Algorithm 1.

Algorithm 1: training_loop

Input: M - black-box model; λ_s, λ_c - loss hyperparameters;
Output: μ - trained actor network used for counterfactual generation.

Load pre-trained encoder enc, and pre-trained decoder dec.
Randomly initialize the actor $\mu(\cdot; \theta_\mu)$ and the critic $Q(\cdot; \theta_Q)$.
Initialize the replay buffer $\mathcal{D}$.
Define reward function $f(\cdot, \cdot)$.
Define post-processing function pp.
for however many steps **do**
  Sample batch of input data x.
  Construct random target y_T and conditioning vector c.
  Compute $y_M = M(x)$, $z = enc(x)$, $z_{CF} = \mu(z, y_M, y_T, c; \theta_\mu)$.
  Select $\tilde{z}_{CF} = clip(z_{CF} + \epsilon, -1, 1)$, $\epsilon \sim \mathcal{N}(0, 0.1)$.
  Decode $\tilde{x}_{CF} = pp(dec(\tilde{z}_{CF}), c)$.
  Observe $R = f(M(\tilde{x}_{CF}), y_T)$.
  Store $(x, z, y_M, y_T, c, \tilde{z}_{CF}, R)$ in the replay buffer $\mathcal{D}$.
  if time to update **then**
    for however many updates **do**
      Call *training_step(*μ, Q, pp, λ_1, λ_2, $\mathcal{D}$*)*
    end for
  end if
end for

Algorithm 2: training_step

Input: μ - actor network; Q - critic networks; pp - post-processing function; λ_s, λ_c - loss hyperparamters, $\mathcal{D}$ - replay buffer;

Sample uniformly a batch of experiences $\mathcal{B} = \{(x, z, y_M, y_T, c, \tilde{z}_{CF}, R)\}$ from $\mathcal{D}$.
Update critic by one-step gradient descent using

$$\nabla_{\theta_Q} \frac{1}{|B|} \sum_B (Q(z, y_M, y_T, c, \tilde{z}_{CF}) - R)^2$$

Compute $z_{CF} = \mu(z, y_M, y_T, c; \theta_\mu)$, $x_{CF} = dec(z_{CF})$,

$$\mathcal{L}_{max} = -\frac{1}{|B|} \sum_B Q(z, y_M, y_T, c, z_{CF}),$$

$$\mathcal{L}_{sparsity} = \frac{1}{|B|} \sum_B [\mathcal{L}_1(x, x_{CF}) + \mathcal{L}_0(x, x_{CF})],$$

$$\mathcal{L}_{consist} = \frac{1}{|B|} \sum_B (enc(pp(x_{CF}, c)) - z_{CF})^2.$$

Update actor by one-step gradient descent using

$$\nabla_{\theta_\mu} (\mathcal{L}_{max} + \lambda_s \mathcal{L}_{sparsity} + \lambda_c \mathcal{L}_{consist})$$

Table 1: Counterfactual validity—percentage of generated counterfactuals of the desired target label (mean±std over 5 classifiers and 3 random seeds).

	Validity (%)				
Method	**Adult**	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LORE	18.08 ± 5.27	25.95 ± 34.33	15.19 ± 7.75	19.07 ± 9.75	9.53 ± 5.91
MO	91.00 ± 1.12	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
DiCE (random)	99.93 ± 0.16	100.00 ± 0.00	92.67 ± 11.04	99.98 ± 0.04	99.58 ± 0.60
DiCE (genetic)	33.94 ± 11.89	60.86 ± 14.18	72.09 ± 18.40	90.97 ± 8.73	40.93 ± 2.28
Ours	98.59 ± 0.97	99.24 ± 0.76	86.81 ± 13.68	98.27 ± 1.53	99.18 ± 0.97

4.5 Handling heterogeneous data types

Our method is versatile and easily adaptable to multiple data modalities by changing only the autoencoder component. As opposed to homogeneous data such as images where the observed features share the same likelihood function, tabular data require more flexibility to capture relations across heterogeneous data types. We therefore define the decoder as a multi-head, fully connected network for which we model each feature feature individually as follows:

- For a real-valued feature d of an instance x with a corresponding encoding z, we use a Gaussian likelihood model with constant variance given by $p(x_d|z) = \mathcal{N}(x_d|h_d(z), \sigma_d)$, where h_d is the decoder head.
- For categorical features we apply one-hot encoding and model feature values using the categorical distribution. Following the same notation from above, the probability for a category k is given by $p(x_d = k|z) = \frac{exp(-h_{dk}(z))}{\sum_{i=1}^{K} exp(-h_{di}(z)}$. where h is the decoder head, and K is the total number of feature values. During training and generation, we pick the most probable feature according to the output distribution.

4.6 Generating explanations

Our counterfactual generation method does not require an iterative optimization procedure per input instance and allows us to generate batches of conditional counterfactual explanations in a single forward pass. This allows the method to scale to large and high-dimensional datasets. The generation step requires a user-specified target y_T, an optional feature-conditioning tensor c, and the black-box model M to label the given input. The pre-trained encoder projects the test instance onto the latent space where the actor-network generates the embedding representation of the counterfactual, given the target and feature-conditioning vector. Finally, we decode the counterfactual embedding followed by an optional post-processing step which ensures that the constraints are respected, casts features to their desired types and maps categorical variables to the original input space. We provide a full description of the generation procedure in Algorithm 3.

Algorithm 3: Generating explanations

Input: x - original instance, y_T - prediction target, c - optional feature-conditioning vector, M - black-box model.
Output: x_{CF} - counterfactual instance.

1: Load pre-trained actor μ, encoder enc, decoder dec, and post-processing function pp.
2: Compute $z = enc(x)$ input encoding representation.
3: Compute $y_M = M(x)$ model's prediction.
4: Generate $z_{CF} = \mu(z, y_M, y_T, c)$ counterfactual embedding.
5: Decode and post-process $\tilde{x}_{CF} = pp(dec(z_{CF}), c)$ counterfactual instance.

5 Experiments

We evaluate our method on multiple data modalities and various classifiers. We focus primarily on the tabular setting for which we selected five datasets: Adult Census (Dua and Graff 2017), Breast Cancer (Zwitter and Soklic 1988), Covertype (Blackard 1998), Portuguese Bank (Moro, Cortez, and Rita 2014), Spambase (Dua and Graff 2017). We test our method on five black-box classifiers: Logistic Regression (LR), Decision Tree (DT), Linear Support Vector Classifier (LSVC), Random Forest (RF), and XGBoost Classifier (XGBC). Optional for lower dimensional input space such as tabular data, our counterfactual generation method can leverage a pre-trained autoencoder per dataset modeled by a simple feedforward neural network. Note that our optimization procedure is hyperparameter dependent through the sparsity and consistency coefficients λ_s and λ_c, respectively. Although a hyperparameter search can be conducted for each dataset and each classifier to obtain the best performances, we decided to use the same configuration ($\lambda_s = \lambda_c = 0.5$) throughout all experiments. We not only achieved competitive results against the existing baselines, but also demonstrated the practicality and stability of our method, discarding any concerns regarding the hyperparameter selection. More experimental configurations which include the setting without the pre-trained autoencoder are provided in the Appendix A.7-A.10.

We compare our methods with three model-agnostic, gradient-free, tabular-oriented baselines, including their variants: LORE (Guidotti et al. 2018), Minimum Observable (MO) (Wexler et al. 2019), and DiCE (Mothilal, Sharma, and Tan 2020). We analyzed two variants of DiCE: the *random* approach, which incrementally increases the number of features to sample from, and the *genetic* approach, which follows a standard evolutionary procedure. We extend some baselines to allow immutable features (e.g., race, gender)

Table 2: Counterfactual sparsity—$\mathcal{L}_0$ and $\mathcal{L}_1$ distance over standardized numerical features (mean±std over 5 classifiers and 3 random seeds).

Method	**Adult**		**Breast**	**Cover.**		**Portug.**		**Spam.**
	$\mathcal{L}_0$	$\mathcal{L}_1$	$\mathcal{L}_0$	$\mathcal{L}_0$	$\mathcal{L}_1$	$\mathcal{L}_0$	$\mathcal{L}_1$	$\mathcal{L}_1$
LORE	0.09 ± 0.01	0.11 ± 0.04	0.11 ± 0.01	0.23 ± 0.11	0.11 ± 0.03	0.04 ± 0.02	0.12 ± 0.03	0.09 ± 0.03
MO	0.20 ± 0.01	0.39 ± 0.07	0.53 ± 0.06	0.45 ± 0.04	0.46 ± 0.01	0.22 ± 0.08	0.31 ± 0.08	0.30 ± 0.01
DiCE(r)	0.05 ± 0.01	1.76 ± 0.10	0.29 ± 0.05	0.22 ± 0.09	0.44 ± 0.04	0.07 ± 0.01	1.39 ± 0.21	0.28 ± 0.05
DiCE(g)	0.18 ± 0.01	0.09 ± 0.02	0.56 ± 0.04	0.45 ± 0.03	0.92 ± 0.09	0.23 ± 0.04	0.60 ± 0.14	0.31 ± 0.02
Ours	0.11 ± 0.10	0.19 ± 0.06	0.44 ± 0.11	0.41 ± 0.15	0.32 ± 0.11	0.23 ± 0.09	0.15 ± 0.11	0.17 ± 0.02

and range constraints (e.g., strictly increasing age) where needed.

We summarise the performance of all methods on a maximum of 1000 samples, for which we randomly generate target labels that differ from the original classification labels, on the following evaluation metrics:

- *validity*—the percentage of counterfactual instances classified as the intended target.
- *sparsity*—the $\mathcal{L}_0$ and $\mathcal{L}_1$ distance for categorical and standardized numerical features, respectively.
- *in-distributionness*—the target-conditional maximum mean discrepancy (MMD) (Gretton et al. 2012) which we compute by taking the MMD between the counterfactuals of a given target class and the training instances with the same target prediction.

5.1 Validity

We compute the validity as the percentage of counterfactual instances classified by the black-box model as the intended target in Table 1. Our method generates counterfactuals with high validity across multiple datasets and classifiers, competitive with DiCE (random) and MO, and significantly outperforming DiCE (genetic) and LORE. Note that MO can always find a counterfactual as long as there is at least one instance in the training set which complies with the counterfactual constraints and predicted target class since it does a nearest neighbour search on the training set. However, if there is no instance in the training set which satisfies the constraints it becomes impossible to generate a valid counterfactual. We attribute the low validity score of LORE to improper local decision boundary approximation. A change in the prediction of the surrogate model does not guarantee the same desired change towards the target class for the original classification model. Likewise, DiCE (genetic) does not ensure the generation of a valid counterfactual. The evolutionary strategy objective is a weighted sum between the divergence and proximity losses, which can favor sparsity over the classification change.

5.2 Sparsity

We report two sparsity measures between the original input and a valid counterfactual: $\mathcal{L}_0$ distance over categorical and $\mathcal{L}_1$ distance over standardized numerical features. Fair evaluation between methods that achieve different levels of validity is however not trivial. For example, reporting the sparsity over all valid counterfactuals per algorithm can favor methods that achieve a lower validity score. Finding a valid counterfactual instance may require larger displacements from the original input to reach the intended target, which will decrease the sparsity for the methods which managed to succeed and will not affect the ones that failed. On the other hand, performing a pairwise comparison over the intersection of valid counterfactuals can have a similar effect, favoring a low validity method that follows a policy focused on changing a single feature value. In Table 2, we report the $\mathcal{L}_0$ and $\mathcal{L}_1$ metrics across the valid counterfactuals to confirm the sparsity of our results. Most comparable to our proposal are the MO and DiCE (random) approaches which achieve similar validity over four datasets (Adult, Breast Cancer, Portuguese Bank, Spambase). Compared to MO, our method generates significantly sparser counterfactuals across all datasets for both numerical and categorical features. DiCE returns sparser counterfactuals on categorical features but does significantly worse on the numerical ones compared to our method which manages to balance the $\mathcal{L}_0$ and $\mathcal{L}_1$ losses successfully.

5.3 In-distributionness

To evaluate the realism of the generated counterfactuals we compute the target-conditional MMD and compare it across the different methods in Tables 3 and 4. The MMD is conditioned on the target prediction because we want the counterfactual to resemble feasible, in-distribution instances from the target class. See Appendix A.9 for MMD computational details.

Figure 2: UMAP (McInnes, Healy, and Melville 2018) embeddings of the Adult and Breast Cancer training sets, labeled by a logistic regression model.

It is not straightforward to fairly compare different methods given the difference in counterfactual validity. If we only

Table 3: In-distributionness—negative class (0) conditional MMD (mean±std over 5 classifiers and 3 random seeds).

	MMD_0^2 (10^{-1})				
Method	**Adult**	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LORE	0.31 ± 0.06	1.09 ± 0.16	0.08 ± 0.03	0.08 ± 0.03	0.26 ± 0.02
MO	0.45 ± 0.04	0.50 ± 0.14	0.07 ± 0.06	0.16 ± 0.10	0.61 ± 0.11
DiCE (random)	0.56 ± 0.44	1.03 ± 0.10	0.17 ± 0.06	1.75 ± 0.71	0.80 ± 0.19
DiCE (genetic)	0.28 ± 0.03	0.85 ± 0.09	0.24 ± 0.10	0.68 ± 0.17	0.25 ± 0.01
Ours	0.36 ± 0.06	0.36 ± 0.34	0.04 ± 0.02	0.10 ± 0.07	0.32 ± 0.13

Table 4: In-distributionness—positive class (1) conditional MMD (mean±std over 5 classifiers and 3 random seeds).

	MMD_1^2 (10^{-1})				
Method	**Adult**	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LORE	0.29 ± 0.02	0.80 ± 0.35	0.07 ± 0.08	0.19 ± 0.04	0.24 ± 0.05
MO	0.13 ± 0.01	1.53 ± 0.62	0.10 ± 0.06	0.16 ± 0.09	0.76 ± 0.19
DiCE (random)	1.48 ± 0.27	0.46 ± 0.10	0.15 ± 0.07	1.50 ± 0.38	0.75 ± 0.17
DiCE (genetic)	0.34 ± 0.04	0.37 ± 0.20	0.07 ± 0.07	0.15 ± 0.10	0.22 ± 0.07
Ours	0.43 ± 0.13	0.97 ± 0.50	0.12 ± 0.05	0.31 ± 0.21	0.33 ± 0.18

	IN	CF(1)	CF(2)	CF(3)
Age	40	40	40	40
Workclass	Private	Private	Private	Private
Education	Masters	Dropout	Masters	High School grad
Marital Status	Married	Married	Married	Married
Occupation	White-Collar	Service	Professional	Sales
Relationship	Husband	Husband	Husband	Husband
Race	White	White	White	White
Sex	Male	Male	Male	Male
Capital Gain	0	0	0	0
Capital Loss	0	0	0	0
Hours per week	40	40	28	32
Country	United-States	United-States	United-States	United-States
Prediction	> $50k/y	≤ $50k/y	≤ $50k/y	≤ $50k/y

Figure 3: Diverse counterfactual instances via feature conditioning. IN—original instance, CF(i)—counterfactual instance. Grayed out feature values correspond to immutable features.

consider the valid counterfactual instances, the comparison would be biased and is likely to favor methods with low validity which generate few but sparse, easy to flip counterfactuals. Even if we consider both the valid and invalid instances, the comparison depends on how overlapping the class-conditional distributions are. If they are clearly separated, the MMD provides a fair metric for direct comparison, but if the distributions overlap then the invalid instances might end up lowering the MMD which is undesirable. This is illustrated in Figure 2, which visualizes UMAP (McInnes, Healy, and Melville 2018) embeddings of the Adult and Breast Cancer datasets, labeled by a logistic regression classifier. There is considerable overlap between the distributions of the two classes in Adult, leading to lower (better) MMD scores for methods with low validity such as LORE compared to our method. The classes in the Breast Cancer dataset are, however, more clearly separated, resulting in higher (worse) MMD values for LORE due to the invalid counterfactuals. As a result, the conditional MMD as a measure of in-distributionness should not be judged in isolation, but jointly with the validity and sparsity metrics.

DiCE (random) achieves similar validity to our method but reaches a higher (worse) MMD score on three datasets (Adult, Portuguese Bank, Spambase) and performs similarly on the other two. MO actually performs worse on two (Breast Cancer, Spambase) and similarly on the other three datasets compared to ours. This is a strong result for our method since the counterfactuals obtained by MO are actual instances of the target class from the training set.

5.4 Diversity and flexibility to other data modalities

Our method can be extended to generate diverse counterfactuals. The deterministic decoding phase ensures consistency over repeated queries but limits the output to a single possible counterfactual per instance. To increase the diversity, we can sample the conditional vector subject to the user-defined feature constraints. For unconstrained features, we follow the same sampling procedure applied during training, while for constrained ones, we sample a subset of their restricted values. The quantitative results for this setting show that such conditional generation produces in-distribution counterfactuals at the expense of a slight drop in validity performance (see Appendix A.7). Qualitative results are depicted in Figure 3.

To demonstrate the flexibility of our approach, we ran experiments on two image datasets, MNIST—a collection of handwritten digits (LeCun 1998), and CelebA—a collection of face images (Liu et al. 2015) divided into four non-overlapping classes characterized by smiling/non-smiling and young/old classification labels. Our training pipeline remains unchanged and only requires a pre-trained autoencoder for each dataset (see Figure 4).

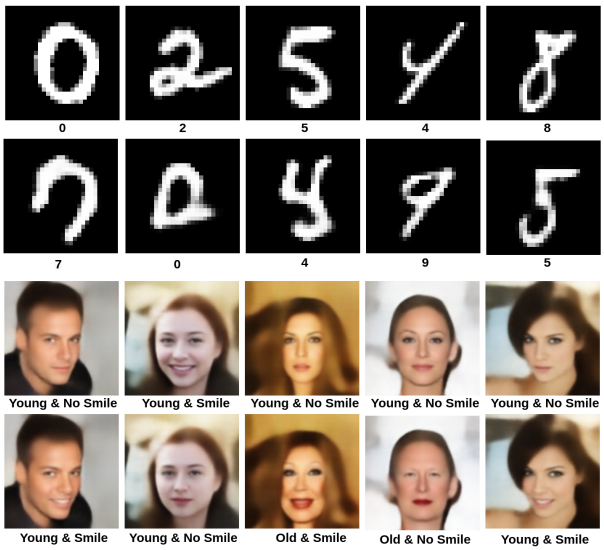

Figure 4: MNIST (top half) and CelebA (bottom half) counterfactual instances. On the odd rows, the original/reconstructed image for MNIST/CelebA, and on the even rows, the counterfactual instance. We use autoencoder reconstructed images for CelebA to compare with the generated counterfactuals as the method is limited by the quality of the autoencoder. Higher quality reconstructions by the autoencoder would lead to higher quality counterfactuals. Labels under images are classifier predictions.

6 Conclusion

Counterfactual instances are a powerful method for explaining the predictions of automated, black-box decision-making systems. They can be particularly useful for actionable recourse by providing direct feedback of what changes to the individual features could be made in order to achieve a specified outcome. Furthermore, being able to condition the explanations on a desired target as well as feasible feature ranges ensures that only relevant explanations are generated.

In this paper, we propose a model-agnostic, conditional counterfactual generation method. We focused primarily on the tabular data setting, and showed that our approach can generate batches of sparse, in-distribution counterfactual instances across various datasets, allowing for target and feature conditioning. Our method is stable during training and does not require extensive hyperparameter tuning. Additionally, we demonstrated that our method is flexible and extendable with minimal effort to other data modalities.

Appendices

A Tabular setup

This section contains all the experimental details related to the tabular setup.

A.1 Datasets

We focus primarily on the tabular setting for which we selected five datasets:

- Adult Census (Dua and Graff 2017)—32,561 instances with 4 numerical and 8 categorical features (we use the pre-processed version from (Klaise et al. 2021)) along with a binary classification label. In all experiments, we allow *Age* to change only in the positive direction, and consider the following features as immutable: *Marital Status*, *Relationship*, *Race*, *Sex*.
- Breast Cancer (Zwitter and Soklic 1988)—699 instances with 9 categorical/ordinal features along with a binary classification label.
- Covertype (Blackard 1998)—581,012 instances with 10 numerical and 2 categorical features along with a multi-classification label (7 classes). For our experiments, we only consider 10% of the available data. Due to high-class imbalance, we apply data balancing during training by adjusting the weights inversely proportional to the class frequencies.
- Portuguese Bank (Moro, Cortez, and Rita 2014)—40,841 instances with 7 numerical and 8 categorical features along with a binary classification label. In all experiments, we allow *age* to change only in the positive direction. We apply the same data balancing as with Covertype.
- Spambase (Dua and Graff 2017)—4,601 instances with 57 numerical features along with a binary classification label.

A.2 Classifiers

We evaluate our method on five classifiers. Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Linear Support Vector Classifier (LSVC) are part of the Scikit-learn machine learning library, and XGBoost Classifier is part of the XGBoost library. Table 5 presents the cross-validation hyperparameters. Missing values correspond to the default ones. We preprocessed each dataset by standardizing the numerical variables and one-hot encoding the categorical ones.

In Table 6 we report the performance (i.e. measured by the accuracy) of each classifier on every dataset for the train and test split, respectively.

A.3 Autoencoders

All tabular autoencoders are fully connected networks with a multi-headed decoder to properly address heterogeneous datatypes. We modeled numerical features using normal distributions with constant variance and categorical ones using categorical distributions. Table 7 summarizes the network architectures of the autoencoder models for each dataset. We trained all models for 100K steps using Adam optimizer (Kingma and Ba 2014) with a batch size of 128 and a learning rate of 1e-3. We defined the reconstruction loss as a weighted combination between Squared Error (SE) averaged across all numerical features and Cross-Entropy (CE) averaged across the categorical ones. In all our experiments, we consider a weight equal to 1 for both loss terms.

Table 5: Tabular classifiers' hyperparameters settings.

Model	Datasets				
	Adult	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LR	C: 10	C: 0.1	C: 100	C: 10	C: 0.1
DT	max_depth: 10, min_samples_split: 5	max_depth: 3, min_samples_split: 4	max_depth: 25, min_samples_split: 3	max_depth: 25	max_depth: 10, min_samples_split: 3
RF	max_depth: 15, min_samples_split: 10, n_estimators: 50	max_depth: 4, min_samples_split: 4, n_estimators: 50	max_depth: 25, n_estimators: 50	max_depth: 25, min_samples_split: 3, n_estimators: 50	max_depth: 20, min_samples_split: 4, n_estimators: 50
LSCV	C: 1	C=0.01	C: 10	C: 100	C=0.01
XGBC	min_child_weight: 0.5, max_depth: 3, gamma: 0.2	min_child_weight: 0.1, max_depth: 3, gamma: 0	min_child_weight: 0.1, max_depth: 20, gamma: 0	min_child_weight: 5.0, max_depth: 4, gamma: 0.01	min_child_weight: 1.0, max_depth: 20, gamma: 0.1

Table 6: Train/test classification accuracy per model and dataset.

	Accuracy train/test(%)				
Model	**Adult**	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LR	85 / 85	97 / 96	94 / 93	85 / 85	61 / 60
DT	87 / 85	95 / 95	97 / 91	98 / 86	98 / 82
RF	88 / 86	97 / 96	99 / 95	100 / 91	100 / 88
LSCV	85 / 85	98 / 96	94 / 93	87 / 87	68 / 68
XGBC	88 / 87	100 / 96	100 / 95	90 / 90	100 / 91

Table 7: Tabular autoencoders' architectures.

Datasets				
Adult	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
Input Layer	Input Layer	Input Layer	Input Layer	Input Layer
Linear(128)	Linear(70)	Linear(128)	Linear(128)	Linear(128)
ReLU()	Tanh()	ReLU()	ReLU()	ReLU()
Linear(15)	Output Layer	Liner(15)	Linear(15)	Linear(50)
Tanh()	-	Tanh()	Tanh()	Tanh()
Linear(128)	-	Linear(128)	Linear(128)	Linear(128)
ReLU()	-	ReLU()	ReLU()	ReLU()
Output Layer	-	Output Layer	Output Layer	Output Layer

A.4 DDPG

The actor and the critic networks follow the same architecture for all data modalities, as displayed in Table 8. We trained the models for 100K steps using a batch size of 128 and Adam optimizer (Kingma and Ba 2014) with the same learning rate of 1e-3 for both actor and critic. We enforced exploration by uniformly sampling the embedding representation in the interval [-1, 1] for 100 training steps. Afterwards, we applied Gaussian additive noise with a constant standard deviation of 0.1 for the rest of the training. The replay buffer has a capacity of 1000 batches, equivalent to 128000 experiences. The learning process starts after ten iterations, corresponding to a replay buffer containing 1280 experiences.

A.5 Algorithms' variants

We analyze the performance of our method by including four of its variants. First, we perform an ablation study on the impact of the autoencoder by removing it from the generation pipeline since the tabular dataset is already low-dimensional (denoted *w/o AE*). To map the input features to the interval $[-1, 1]$, we replace the encoding step with a projection through a *tanh* nonlinearity and the decoding phase with its corresponding inverse transformation. The rest of the training procedure remains identical. Then, we investigate the ability to generate diverse counterfactuals by conditioning the generation process on a randomly sampled conditional vector c. For the two pipeline settings (with and without the AE), we used the same hyperparameters across all the datasets and classifiers. The configuration with the autoencoder, uses a sparsity coefficient $\lambda_s = 0.5$ and a consistency coefficient $\lambda_c = 0.5$. The configuration without the autoencoder only uses a sparsity coefficient $\lambda_s = 0.1$ since the consistency loss is not defined.

We compare our methods with three model-agnostic, gradient-free, tabular-oriented baselines, including their variants: LORE (Guidotti et al. 2018), Minimum Observable (MO) (Wexler et al. 2019), and DiCE (Mothilal, Sharma, and Tan 2020). We analyzed two variants of DiCE: the *random* approach, which incrementally increases the number of features to sample from, and the *genetic* approach, which follows a standard evolutionary procedure. In addition to the two standard DiCE procedures which use soft target labels (e.g., [0.7, 0.3]), we experimented with a hard/binary representation (e.g., [1, 0]) of the output probability distribution (denoted *binary*).

A.6 Benchmarking setup

We summarise the performance of all methods on a maximum of 1000 samples, for which we randomly generate target labels that differ from the original classification labels. To ensure the termination of the counterfactual generation process per instance, we restrict the running time to a window of 10 seconds chosen to exceed the average generation time across all baselines. In contrast, our method generates all counterfactuals within a few seconds. We keep the default hyper-parameters for all baselines, except for LORE, where we reduce the population size by a factor of 10 to fit the time constraints without significantly affecting the overall performance. We extend some baselines to allow immutable features (e.g., race, gender) and range constraints (e.g., strictly increasing age) where needed.

A.7 Validity results

We compute the validity as the percentage of counterfactual instances classified by the black-box model as the intended target. Table 9 summarises the results over all algorithms' variants. For an analysis of the result, see Sections 5.1 and 5.4 from the main paper.

A.8 Sparsity results

We report two sparsity measures between the original input and a valid counterfactual: $\mathcal{L}_0$ distance over categorical and $\mathcal{L}_1$ distance over standardized numerical features. Tables 10 and 11 summarise the results over all algorithms' variants for the $\mathcal{L}_0$ and $\mathcal{L}_1$ distances, respectively. For a analysis of the results, see Section 5.2 from the main paper.

A.9 In-distributionness

We evaluate the realism of the generated counterfactual through the target-conditional MMD. We condition the MMD on the target prediction since we want to measure the in-distributionness of the generated instances against the reference training instances belonging to the target class. We perform a dimensionality reduction step for the MMD computation with a randomly initialized encoder (Rabanser, Günnemann, and Lipton 2019) following an architecture that consists of three linear layers of dimension 32, 16, and 5 with ReLU non-linearities in-between. We use a radial basis function kernel for which we derive the standard deviation from the input data (Van Looveren et al. 2019). Tables 12 and 13 summarise the results over all algorithms' variants. For a analysis of the result, see Section 5.3 from the main paper.

Table 14 summarizes the results obtained for the *Covertype* dataset for all classes, previously presented in Tables 12 and 13 only for the first two most dominant classes along with the rest of binary classification datasets. The scores are averaged across three seeds and all classifiers.

A.10 Results accross different hyperparameters averaged over all datasets, classifiers and seeds.

For both pipeline configurations, we perform a hyperparameter search over the sparsity (λ_s) and consistency (λ_c) coefficients. We evaluate the percentage of the valid counterfactuals and the sparsity through the $\mathcal{L}_0$ and $\mathcal{L}_1$ norm for 1000 instances, where the test set permits. We report the results for the original unbalanced class distribution (denoted as unbalanced) and the results obtained after balancing the test set (denoted as balanced). The balancing procedure consists of sampling uniformly without replacement the same number of instances from the class conditional subset for each class. Tables 15 and 16 summarizes the results averaged across all seeds, classifiers, and datasets for the normal and diverse

Table 8: DDPG actor-critic architectures.

Actor	**Critic**
Linear(input_size, hidden_dim)	Linear(input_size + latent_dim, hidden_dim)
LayerNorm(hidden_dim)	LayerNorm(hidden_dim)
ReLU()	ReLU()
Linear(hidden_dim, hidden_dim)	Linear(hidden_dim, hidden_dim)
LayerNorm(hidden_dim)	LayerNorm(hidden_dim)
ReLU()	ReLU()
Linear(hidden_dim, latent_dim)	Linear(hidden_dim, 1)

Table 9: Counterfactual validity—percentage of generated counterfactuals of the desired target label (mean±std over 5 classifiers and 3 random seeds).

	Validity (%)				
Method	**Adult**	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LORE	18.08 ± 5.27	25.95 ± 34.33	15.19 ± 7.75	19.07 ± 9.75	9.53 ± 5.91
MO	91.00 ± 1.12	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
DiCE (random)	99.93 ± 0.16	100.00 ± 0.00	92.67 ± 11.04	99.98 ± 0.04	99.58 ± 0.60
DiCE (genetic)	33.94 ± 11.89	60.86 ± 14.18	72.09 ± 18.40	90.97 ± 8.73	40.93 ± 2.28
DiCE (binary, random)	100.00 ± 0.00	100.00 ± 0.00	92.66 ± 11.03	99.99 ± 0.03	99.87 ± 0.25
DiCE (binary, genetic)	51.28 ± 5.00	74.57 ± 6.92	77.55 ± 14.49	98.50 ± 1.78	40.72 ± 2.07
Ours (w/o AE)	98.59 ± 0.97	91.24 ± 9.96	91.11 ± 6.58	99.38 ± 1.24	99.72 ± 0.41
Ours (w/o AE, diverse)	97.17 ± 0.87	91.19 ± 3.73	80.07 ± 7.54	98.23 ± 1.20	99.07 ± 0.99
Ours	98.59 ± 0.97	99.24 ± 0.76	86.81 ± 13.68	98.27 ± 1.53	99.18 ± 0.97
Ours (diverse)	97.50 ± 0.58	96.90 ± 1.76	76.55 ± 16.65	96.10 ± 2.78	98.67 ± 1.24

Table 10: Counterfactual sparsity—$\mathcal{L}_0$ distance over categorical features (mean±std over 5 classifiers and 3 random seeds).

	$\mathcal{L}_0$ **distance**				
Method	**Adult**	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LORE	0.09 ± 0.01	0.11 ± 0.01	0.23 ± 0.11	0.04 ± 0.02	-
MO	0.20 ± 0.01	0.53 ± 0.06	0.45 ± 0.04	0.22 ± 0.08	-
DiCE (random)	0.05 ± 0.01	0.29 ± 0.05	0.22 ± 0.09	0.07 ± 0.010	-
DiCE (genetic)	0.18 ± 0.01	0.56 ± 0.04	0.45 ± 0.03	0.23 ± 0.04	-
DiCE (binary, random)	0.05 ± 0.01	0.29 ± 0.05	0.22 ± 0.09	0.07 ± 0.01	-
DiCE (binary, genetic)	0.19 ± 0.01	0.56 ± 0.03	0.46 ± 0.04	0.24 ± 0.05	-
Ours (w/o AE)	0.13 ± 0.10	0.48 ± 0.09	0.47 ± 0.13	0.31 ± 0.07	-
Ours (w/o AE, diverse)	0.11 ± 0.09	0.45 ± 0.10	0.34 ± 0.11	0.25 ± 0.08	-
Ours	0.11 ± 0.10	0.44 ± 0.11	0.41 ± 0.15	0.23 ± 0.09	-
Ours (diverse)	0.10 ± 0.09	0.43 ± 0.10	0.34 ± 0.14	0.17 ± 0.06	-

Table 11: Counterfactual sparsity—$\mathcal{L}_1$ distance over standardized numerical features (mean±std over 5 classifiers and 3 random seeds).

	$\mathcal{L}_1$ **distance**				
Method	**Adult**	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LORE	0.11 ± 0.04	-	0.11 ± 0.03	0.12 ± 0.03	0.09 ± 0.03
MO	0.39 ± 0.07	-	0.46 ± 0.01	0.31 ± 0.08	0.30 ± 0.01
DiCE (random)	1.76 ± 0.10	-	0.44 ± 0.04	1.39 ± 0.21	0.28 ± 0.05
DiCE (genetic)	0.09 ± 0.02	-	0.92 ± 0.09	0.60 ± 0.14	0.31 ± 0.02
DiCE (binary, random)	1.76 ± 0.10	-	0.44 ± 0.04	1.39 ± 0.21	0.28 ± 0.05
DiCE (binary, genetic)	0.08 ± 0.02	-	0.96 ± 0.06	0.59 ± 0.13	0.31 ± 0.02
Ours (w/o AE)	0.23 ± 0.07	-	0.57 ± 0.14	0.18 ± 0.09	0.26 ± 0.09
Ours (w/o AE, diverse)	0.24 ± 0.05	-	0.56 ± 0.10	0.32 ± 0.17	0.23 ± 0.08
Ours	0.19 ± 0.06	-	0.32 ± 0.11	0.15 ± 0.11	0.17 ± 0.02
Ours (diverse)	0.21 ± 0.05	-	0.32 ± 0.07	0.24 ± 0.13	0.15 ± 0.02

Table 12: In-distributionness—negative class (0) conditional MMD (mean±std over 5 classifiers and 3 random seeds).

	$MMD_0^2(10^{-1})$				
Method	**Adult**	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LORE	0.31 ± 0.06	1.09 ± 0.16	0.08 ± 0.03	0.08 ± 0.03	0.26 ± 0.02
MO	0.45 ± 0.04	0.50 ± 0.14	0.07 ± 0.06	0.16 ± 0.10	0.61 ± 0.11
DiCE (random)	0.56 ± 0.44	1.03 ± 0.10	0.17 ± 0.06	1.75 ± 0.71	0.80 ± 0.19
DiCE (genetic)	0.28 ± 0.03	0.85 ± 0.09	0.24 ± 0.10	0.68 ± 0.17	0.25 ± 0.01
DiCE (binary, random)	0.56 ± 0.44	1.03 ± 0.10	0.17 ± 0.06	1.75 ± 0.71	0.80 ± 0.19
DiCE (binary, genetic)	0.25 ± 0.02	0.78 ± 0.12	0.27 ± 0.10	0.67 ± 0.16	0.25 ± 0.02
Ours (w/o AE)	0.56 ± 0.31	1.01 ± 0.41	0.33 ± 0.40	0.07 ± 0.05	0.39 ± 0.53
Ours (w/o AE, diverse)	0.45 ± 0.22	1.02 ± 0.18	0.25 ± 0.20	0.05 ± 0.03	0.27 ± 0.40
Ours	0.36 ± 0.06	0.36 ± 0.34	0.04 ± 0.02	0.10 ± 0.07	0.32 ± 0.13
Ours (diverse)	0.30 ± 0.06	0.65 ± 0.15	0.02 ± 0.01	0.07 ± 0.04	0.28 ± 0.11

Table 13: In-distributionness—positive class (1) conditional MMD (mean±std over 5 classifiers and 3 random seeds).

	$MMD_1^2(10^{-1})$				
Method	**Adult**	**Breast cancer**	**Covertype**	**Portuguese Bank**	**Spambase**
LORE	0.29 ± 0.02	0.80 ± 0.35	0.07 ± 0.08	0.19 ± 0.04	0.24 ± 0.05
MO	0.13 ± 0.01	1.53 ± 0.62	0.10 ± 0.06	0.16 ± 0.09	0.76 ± 0.19
DiCE (random)	1.48 ± 0.27	0.46 ± 0.10	0.15 ± 0.07	1.50 ± 0.38	0.75 ± 0.17
DiCE (genetic)	0.34 ± 0.04	0.37 ± 0.20	0.07 ± 0.07	0.15 ± 0.10	0.22 ± 0.07
DiCE (binary, random)	1.48 ± 0.28	0.46 ± 0.10	0.15 ± 0.07	1.50 ± 0.38	0.75 ± 0.18
DiCE (binary, genetic)	0.33 ± 0.04	0.48 ± 0.22	0.07 ± 0.07	0.15 ± 0.11	0.22 ± 0.07
Ours (w/o AE)	0.34 ± 0.06	1.09 ± 0.42	0.33 ± 0.18	0.20 ± 0.15	0.67 ± 0.50
Ours (w/o AE, diverse)	0.32 ± 0.06	0.50 ± 0.30	0.15 ± 0.10	0.14 ± 0.06	0.35 ± 0.22
Ours	0.43 ± 0.13	0.97 ± 0.50	0.12 ± 0.05	0.31 ± 0.21	0.33 ± 0.18
Ours (diverse)	0.30 ± 0.06	0.28 ± 0.15	0.09 ± 0.03	0.21 ± 0.16	0.23 ± 0.16

Table 14: Covertype MMD for unbalanced datasets with untrained encoder dimensionality reduction.

	Target conditional MMD (10^{-1})						
Method	**0**	**1**	**2**	**3**	**4**	**5**	**6**
LORE	0.08 ± 0.03	0.07 ± 0.08	0.52 ± 0.07	1.25 ± 0.14	0.23 ± 0.11	1.20 ± 0.17	0.25 ± 0.04
MO	0.07 ± 0.06	0.10 ± 0.06	0.47 ± 0.09	0.05 ± 0.10	0.05 ± 0.04	0.85 ± 0.09	0.15 ± 0.05
DICE(random)	0.17 ± 0.06	0.15 ± 0.07	0.94 ± 0.72	0.65 ± 0.51	0.63 ± 0.47	0.87 ± 0.18	0.36 ± 0.13
DICE(genetic)	0.24 ± 0.10	0.07 ± 0.07	0.41 ± 0.28	0.59 ± 0.28	0.38 ± 0.17	0.71 ± 0.34	0.44 ± 0.20
DiCE(binary, random)	0.17 ± 0.06	0.15 ± 0.07	0.94 ± 0.72	0.65 ± 0.51	0.63 ± 0.47	0.87 ± 0.18	0.36 ± 0.13
DiCE(binary, genetic)	0.27 ± 0.10	0.07 ± 0.07	0.47 ± 0.26	0.54 ± 0.27	0.36 ± 0.18	0.72 ± 0.33	0.46 ± 0.20
Ours(w/o AE)	0.33 ± 0.40	0.33 ± 0.18	0.65 ± 0.25	1.04 ± 0.51	0.82 ± 0.58	0.78 ± 0.31	0.57 ± 0.41
Ours(w/o AE, diverse)	0.25 ± 0.20	0.15 ± 0.10	0.49 ± 0.26	0.75 ± 0.52	0.37 ± 0.18	0.52 ± 0.34	0.23 ± 0.13
Ours	0.04 ± 0.02	0.12 ± 0.05	0.48 ± 0.21	0.55 ± 0.46	0.33 ± 0.26	0.78 ± 0.48	0.16 ± 0.14
Ours(diverse)	0.02 ± 0.01	0.09 ± 0.03	0.22 ± 0.08	0.66 ± 0.47	0.33 ± 0.22	0.60 ± 0.30	0.14 ± 0.08

version of the algorithm, respectively. Sparser results correspond to lower validity scores. Note that a finer per dataset, per classifier hyperparameter selection procedure may yield better results.

B Image setup

This section contains all the experimental details related to the image setup.

B.1 Datasets

To demonstrate the flexibility of our approach, we ran experiments on two image datasets, MNIST—a collection of handwritten digits (LeCun 1998), and CelebA—a collection of face images (Liu et al. 2015) divided into four non-overlapping classes characterized by smiling/non-smiling and young/old classification labels.

B.2 Classifier

The classification model (CNN) is the same as in (Van Looveren and Klaise 2019; Van Looveren et al. 2021), achieving accuracy on the test set of 99% and 81% for MNIST and CelebA, respectively.

B.3 Autoencoders

The MNIST encoder projects the input instances to a latent representation of dimension 32. We trained the autoencoder for 50K steps with a batch size of 64, optimizing the Binary Cross-Entropy (BCE) using Adam optimizer (Kingma and Ba 2014) with a learning rate of 1e-3. The CelebA encoder projects the input space from 128x128 to a latent representation of dimension 128. We trained the autoencoder for 350K steps with a batch size of 128, optimizing the Mean Squared Error(MSE) using Adam optimizer (Kingma and Ba 2014) with a learning rate of 1e-4. Tables 17 and 18 summarize the autoencoders' architecture for both datasets.

B.4 DDPG

We trained the MNIST and CelebA counterfactual generator for 300K and 100K iterations, with a sparsity coefficient $\lambda_s = 7.5$ for MNIST, and $\lambda_s = 20.0$ for CelebA, and a consistency coefficient $\lambda_c = 0$ for both datasets. The rest of the settings described in A.4 remain unchanged.

Table 15: Comparison validity and sparsity.

	Unbalanced			Balanced		
Method	**Validity**(%)	$\mathcal{L}_0$	$\mathcal{L}_1$	**Validity**(%)	$\mathcal{L}_0$	$\mathcal{L}_1$
$\lambda_s = 0.1$, w/o AE	96.01 ± 6.69	0.35 ± 0.17	0.31 ± 0.18	96.09 ± 5.97	0.34 ± 0.17	0.31 ± 0.17
$\lambda_s = 0.2$, w/o AE	93.89 ± 8.44	0.28 ± 0.14	0.25 ± 0.14	93.89 ± 7.76	0.28 ± 0.14	0.25 ± 0.13
$\lambda_s = 0.3$, w/o AE	90.94 ± 12.24	0.25 ± 0.12	0.23 ± 0.13	90.47 ± 11.50	0.24 ± 0.13	0.23 ± 0.11
$\lambda_s = 0.4$, w/o AE	88.07 ± 14.75	0.24 ± 0.12	0.22 ± 0.12	86.92 ± 14.84	0.23 ± 0.12	0.22 ± 0.11
$\lambda_s = 0.5$, w/o AE	84.63 ± 18.67	0.22 ± 0.11	0.20 ± 0.11	82.02 ± 19.58	0.21 ± 0.11	0.20 ± 0.09
$\lambda_s = 0.1, \lambda_c = 0.0$	99.54 ± 1.15	0.51 ± 0.22	0.37 ± 0.20	99.35 ± 1.15	0.51 ± 0.21	0.37 ± 0.20
$\lambda_s = 0.1, \lambda_c = 0.5$	99.03 ± 2.98	0.46 ± 0.27	0.38 ± 0.21	98.78 ± 3.17	0.48 ± 0.28	0.37 ± 0.20
$\lambda_s = 0.2, \lambda_c = 0.5$	99.02 ± 2.60	0.39 ± 0.23	0.29 ± 0.16	98.65 ± 3.01	0.40 ± 0.24	0.28 ± 0.16
$\lambda_s = 0.3, \lambda_c = 0.5$	97.81 ± 6.74	0.34 ± 0.20	0.26 ± 0.14	97.42 ± 6.66	0.35 ± 0.22	0.24 ± 0.13
$\lambda_s = 0.3, \lambda_c = 1.0$	97.80 ± 6.45	0.37 ± 0.23	0.25 ± 0.13	97.49 ± 6.39	0.38 ± 0.24	0.24 ± 0.12
$\lambda_s = 0.4, \lambda_c = 0.5$	97.23 ± 6.38	0.32 ± 0.19	0.23 ± 0.11	95.99 ± 7.85	0.33 ± 0.20	0.21 ± 0.10
$\lambda_s = 0.4, \lambda_c = 1.0$	97.20 ± 7.33	0.33 ± 0.21	0.22 ± 0.11	95.95 ± 8.43	0.34 ± 0.23	0.21 ± 0.10
$\lambda_s = 0.5, \lambda_c = 0.0$	96.51 ± 9.73	0.39 ± 0.16	0.25 ± 0.13	95.21 ± 10.40	0.39 ± 0.18	0.24 ± 0.12
$\lambda_s = 0.5, \lambda_c = 0.1$	96.13 ± 9.55	0.30 ± 0.16	0.23 ± 0.11	94.65 ± 10.23	0.31 ± 0.17	0.22 ± 0.10
$\lambda_s = 0.5, \lambda_c = 0.2$	96.52 ± 7.89	0.29 ± 0.16	0.22 ± 0.11	94.74 ± 9.97	0.30 ± 0.17	0.20 ± 0.10
$\lambda_s = 0.5, \lambda_c = 0.5$	96.42 ± 7.85	0.30 ± 0.18	0.21 ± 0.11	94.75 ± 10.07	0.31 ± 0.19	0.19 ± 0.09
$\lambda_s = 0.5, \lambda_c = 1.0$	96.57 ± 7.97	0.31 ± 0.20	0.21 ± 0.10	94.89 ± 10.18	0.32 ± 0.21	0.20 ± 0.09
$\lambda_s = 0.5, \lambda_c = 1.5$	95.81 ± 9.94	0.32 ± 0.21	0.22 ± 0.11	93.90 ± 12.01	0.33 ± 0.23	0.20 ± 0.09
$\lambda_s = 0.5, \lambda_c = 5.0$	85.30 ± 23.86	0.40 ± 0.27	0.22 ± 0.11	86.31 ± 19.06	0.40 ± 0.28	0.20 ± 0.10

Table 16: Comparison validity and sparsity for diversity sampling.

	Unbalanced			Balanced		
Method	**Validity**(%)	$\mathcal{L}_0$	$\mathcal{L}_1$	**Validity**(%)	$\mathcal{L}_0$	$\mathcal{L}_1$
$\lambda_s = 0.1$, w/o AE, diverse	96.01 ± 6.69	0.35 ± 0.17	0.31 ± 0.18	96.09 ± 5.97	0.34 ± 0.17	0.31 ± 0.17
$\lambda_s = 0.2$, w/o AE, diverse	93.89 ± 8.44	0.28 ± 0.14	0.25 ± 0.14	93.89 ± 7.76	0.28 ± 0.14	0.25 ± 0.13
$\lambda_s = 0.3$, w/o AE, diverse	90.94 ± 12.24	0.25 ± 0.12	0.23 ± 0.13	90.47 ± 11.50	0.24 ± 0.13	0.23 ± 0.11
$\lambda_s = 0.4$, w/o AE, diverse	88.07 ± 14.75	0.24 ± 0.12	0.22 ± 0.12	86.92 ± 14.84	0.23 ± 0.12	0.22 ± 0.11
$\lambda_s = 0.5$, w/o AE, diverse	84.63 ± 18.67	0.22 ± 0.11	0.20 ± 0.11	82.02 ± 19.58	0.21 ± 0.11	0.20 ± 0.09
$\lambda_s = 0.1, \lambda_c = 0.0$, diverse	96.81 ± 5.36	0.44 ± 0.19	0.41 ± 0.19	96.25 ± 5.72	0.44 ± 0.20	0.40 ± 0.18
$\lambda_s = 0.1, \lambda_c = 0.5$, diverse	96.64 ± 6.42	0.40 ± 0.23	0.40 ± 0.18	96.21 ± 6.75	0.41 ± 0.24	0.38 ± 0.17
$\lambda_s = 0.2, \lambda_c = 0.5$, diverse	96.12 ± 6.92	0.34 ± 0.20	0.31 ± 0.14	95.57 ± 7.35	0.34 ± 0.21	0.29 ± 0.13
$\lambda_s = 0.3, \lambda_c = 0.5$, diverse	95.30 ± 9.06	0.30 ± 0.18	0.27 ± 0.12	94.76 ± 9.08	0.31 ± 0.20	0.26 ± 0.11
$\lambda_s = 0.3, \lambda_c = 1.0$, diverse	95.42 ± 8.77	0.31 ± 0.20	0.27 ± 0.12	94.84 ± 8.66	0.33 ± 0.22	0.26 ± 0.11
$\lambda_s = 0.4, \lambda_c = 0.5$, diverse	94.22 ± 9.85	0.28 ± 0.17	0.25 ± 0.11	93.25 ± 10.71	0.29 ± 0.19	0.23 ± 0.09
$\lambda_s = 0.4, \lambda_c = 1.0$, diverse	94.17 ± 10.47	0.29 ± 0.19	0.25 ± 0.11	93.35 ± 11.01	0.29 ± 0.20	0.23 ± 0.09
$\lambda_s = 0.5, \lambda_c = 0.0$, diverse	93.56 ± 11.62	0.34 ± 0.15	0.28 ± 0.10	92.58 ± 12.04	0.34 ± 0.17	0.26 ± 0.10
$\lambda_s = 0.5, \lambda_c = 0.1$, diverse	93.29 ± 11.90	0.27 ± 0.15	0.25 ± 0.10	92.36 ± 12.40	0.28 ± 0.17	0.23 ± 0.09
$\lambda_s = 0.5, \lambda_c = 0.2$, diverse	93.81 ± 10.81	0.26 ± 0.15	0.24 ± 0.10	92.69 ± 12.19	0.27 ± 0.17	0.22 ± 0.09
$\lambda_s = 0.5, \lambda_c = 0.5$, diverse	93.14 ± 11.29	0.26 ± 0.16	0.23 ± 0.10	92.06 ± 12.66	0.27 ± 0.18	0.21 ± 0.09
$\lambda_s = 0.5, \lambda_c = 1.0$, diverse	93.40 ± 11.45	0.27 ± 0.17	0.23 ± 0.10	92.32 ± 12.57	0.28 ± 0.19	0.21 ± 0.08
$\lambda_s = 0.5, \lambda_c = 1.5$, diverse	92.85 ± 12.18	0.27 ± 0.18	0.23 ± 0.10	91.71 ± 13.51	0.28 ± 0.20	0.22 ± 0.08
$\lambda_s = 0.5, \lambda_c = 5.0$, diverse	81.39 ± 24.63	0.33 ± 0.24	0.24 ± 0.10	83.08 ± 20.38	0.34 ± 0.26	0.22 ± 0.09

Table 17: Image autoencoders' architectures.

Datasets	
MNIST	**CelebA**
Conv2d(1, 16, kernel_size=(3, 3), padding=1)	ConvBlock(3, 32)
ReLU()	ConvBlock(32, 64)
MaxPool2d(kernel_size=(2, 2), stride=2)	ConvBlock(64, 128)
Conv2d(16, 8, kernel_size=(3, 3), padding=1)	ConvBlock(128, 256)
ReLU()	ConvBlock(256, 512)
MaxPool2d(kernel_size=(2, 2), stride=2)	ConvBlock(512, 512)
Conv2d(8, 8, kernel_size=(3, 3), padding=2)	Flatten()
ReLU()	Linear(2048, 128)
MaxPool2d(kernel_size=(2, 2), stride=2)	Tanh()
Flatten()	Reshape(512, 2, 2)
Linear(128, 32)	TransConvBlock(512, 512)
Tanh()	TansConvBlock(512, 512)
Linear(32, 128)	TransConvBlock(512, 256)
ReLU()	TransConvBlock(256, 128)
Reshape(8, 4, 4)	TransConvBlock(128, 64)
Conv2d(8, 8, kernel_size=(3, 3), padding=1)	TransConvBlock(64, 32)
ReLU()	TransConvBlock(32, 32)
Upsample(scale_factor=2)	TransConvBlock(32, 3)
Conv2d(8, 8, kernel_size=(3, 3), padding=1)	Tanh()
ReLU()	-
Upsample(scale_factor=2)	-
Conv2d(8, 16, kernel_size=(3, 3))	-
ReLU()	-
Upsample(scale_factor=2)	-
Conv2d(16, 1, kernel_size=(3, 3), padding=1)	-
Sigmoid()	-

Table 18: Blocks' architectures.

Blocks	
ConvBlock	**TransConvBlock**
Conv2d(in_channels, out_channels, kernel_size=3, stride=2, padding=1)	ConvTranspose2d(in_channels, out_channels, kernel_size=3, stride=2, padding=1, output_padding=1)
BatchNorm2d(out_channles)	BatchNorm2d(out_channles)
LeakyReLU()	LeakyReLU()

C Tabular and image samples

Qualitative evaluation for tabular and image setup.

C.1 Samples Adult

Age	Workclass	Education	Marital Status	Occupation	Relationship	Race	Sex	Capital Gain	Capital Loss	Hours per week	Country	Label
54	Private	Dropout	Married	Blue-Collar	Husband	White	Male	0	0	40	United-States	≤ $50k/y
54	Private	Dropout	Married	Blue-Collar	Husband	White	Male	5796	0	40	United-States	> $50k/y
17	Private	Dropout	Never-Married	Admin	Own-child	White	Female	0	0	20	United-States	≤ $50k/y
27	Private	Dropout	Never-Married	Admin	Own-child	White	Female	8231	0	20	United-States	> $50k/y
26	Private	High School grad	Never-Married	Sales	Other-relative	White	Female	0	0	20	United-States	≤ $50k/y
26	Private	High School grad	Never-Married	Sales	Other-relative	White	Female	8117	0	20	United-States	> $50k/y
22	Private	High School grad	Separated	Blue-Collar	Own-child	White	Female	0	0	40	United-States	≤ $50k/y
22	Private	High School grad	Separated	Blue-Collar	Own-child	White	Female	9586	0	40	United-States	> $50k/y
44	Local-gov	Bachelors	Never-Married	Professional	Own-child	White	Male	0	0	60	United-States	≤ $50k/y
44	Local-gov	Bachelors	Never-Married	Professional	Own-child	White	Male	8010	0	60	United-States	> $50k/y
32	Federal-gov	Associates	Separated	Other	Not-in-family	White	Male	2174	0	40	United-States	≤ $50k/y
32	Federal-gov	Associates	Separated	Other	Not-in-family	White	Male	8088	0	40	United-States	> $50k/y
46	Private	High School grad	Never-Married	White-Collar	Unmarried	White	Female	0	0	45	United-States	≤ $50k/y
46	Private	High School grad	Never-Married	White-Collar	Unmarried	White	Female	9537	0	45	United-States	> $50k/y
23	Private	High School grad	Married	Admin	Other-relative	White	Male	0	0	40	United-States	≤ $50k/y
23	Private	High School grad	Married	Admin	Other-relative	White	Male	6383	0	40	United-States	> $50k/y
29	Private	High School grad	Married	Blue-Collar	Wife	White	Female	0	0	40	Latin-America	≤ $50k/y
29	Private	High School grad	Married	Blue-Collar	Wife	White	Female	6083	0	40	Latin-America	> $50k/y
30	Private	Dropout	Never-Married	Sales	Other-relative	White	Male	0	0	18	Latin-America	≤ $50k/y
30	Private	Dropout	Never-Married	Sales	Other-relative	White	Male	7803	0	18	Latin-America	> $50k/y
60	Private	High School grad	Married	Blue-Collar	Husband	White	Male	7298	0	40	United-States	> $50k/y
60	Private	High School grad	Married	Blue-Collar	Husband	White	Male	2945	0	40	United-States	≤ $50k/y
35	Private	High School grad	Married	White-Collar	Husband	White	Male	7688	0	50	United-States	> $50k/y
35	Private	High School grad	Married	Blue-Collar	Husband	White	Male	3157	1	50	United-States	≤ $50k/y
39	State-gov	Masters	Married	Professional	Wife	White	Female	5178	0	38	United-States	> $50k/y
39	State-gov	Associates	Married	Blue-Collar	Wife	White	Female	2856	2	38	United-States	≤ $50k/y
44	Self-emp-inc	High School grad	Married	Sales	Husband	White	Male	0	0	50	United-States	> $50k/y
44	Self-emp-inc	High School grad	Married	Blue-Collar	Husband	White	Male	7	1	50	United-States	≤ $50k/y
44	Federal-gov	High School grad	Married	Admin	Husband	White	Male	0	0	40	United-States	> $50k/y
44	Federal-gov	High School grad	Married	Blue-Collar	Husband	White	Male	0	0	40	United-States	≤ $50k/y
39	Private	Bachelors	Separated	White-Collar	Not-in-family	White	Female	13550	0	50	United-States	> $50k/y
39	Private	Bachelors	Separated	White-Collar	Not-in-family	White	Female	4361	1	50	United-States	≤ $50k/y
45	Private	High School grad	Married	Blue-Collar	Husband	White	Male	0	1902	40	?	> $50k/y
45	Private	High School grad	Married	Blue-Collar	Husband	White	Male	0	1655	40	?	≤ $50k/y
50	Private	Bachelors	Married	Professional	Husband	White	Male	0	0	50	United-States	> $50k/y
50	Private	Bachelors	Married	Blue-Collar	Husband	White	Male	58	3	50	United-States	≤ $50k/y
29	Private	Bachelors	Married	White-Collar	Wife	White	Female	0	0	50	United-States	> $50k/y
29	Private	Associates	Married	Blue-Collar	Wife	White	Female	0	0	50	United-States	≤ $50k/y
47	Private	Bachelors	Married	Professional	Husband	White	Male	0	0	50	United-States	> $50k/y
47	Private	Bachelors	Married	Blue-Collar	Husband	White	Male	60	3	50	United-States	≤ $50k/y

Figure 5: Counterfactual instances on Adult dataset. Odd rows correspond to the input instances and even rows correspond to the counterfactual instances. Grayed out features correspond to immutable features and *Age* feature is allowed to increase only. Feature changes are highlighted in red.

Age	Workclass	Education	Marital Status	Occupation	Relationship	Race	Sex	Capital Gain	Capital Loss	Hours per week	Country	Label
38	Self-emp-not-inc	High School grad	Married	Blue-Collar	Husband	White	Male	0	0	60	United-States	≤ $50k/y
38	Self-emp-not-inc	High School grad	Married	Blue-Collar	Husband	White	Male	5744	0	60	United-States	> $50k/y
64	Private	Associates	Married	Blue-Collar	Husband	White	Male	0	0	48	United-States	≤ $50k/y
64	Private	Associates	Married	Blue-Collar	Husband	White	Male	5773	0	48	United-States	> $50k/y
38	Private	Dropout	Married	Blue-Collar	Husband	White	Male	0	0	40	United-States	≤ $50k/y
38	Private	Dropout	Married	Blue-Collar	Husband	White	Male	5861	0	40	United-States	> $50k/y
62	Local-gov	Bachelors	Separated	Professional	Not-in-family	Black	Female	0	0	40	United-States	≤ $50k/y
62	Local-gov	Bachelors	Separated	Professional	Not-in-family	Black	Female	7745	0	40	United-States	> $50k/y
19	Private	Dropout	Never-Married	Sales	Own-child	Asian-Pac-Islander	Female	0	0	19	SE-Asia	≤ $50k/y
29	Private	Dropout	Never-Married	Sales	Own-child	Asian-Pac-Islander	Female	8317	0	19	SE-Asia	> $50k/y
59	Private	High School grad	Separated	Admin	Not-in-family	White	Female	0	0	40	United-States	≤ $50k/y
59	Private	High School grad	Separated	Admin	Not-in-family	White	Female	9447	0	40	United-States	> $50k/y
31	Private	High School grad	Separated	Sales	Not-in-family	White	Female	0	0	32	United-States	≤ $50k/y
31	Private	High School grad	Separated	Sales	Not-in-family	White	Female	9571	0	32	United-States	> $50k/y
50	Private	High School grad	Married	Blue-Collar	Husband	White	Male	0	0	50	United-States	≤ $50k/y
50	Private	High School grad	Married	Blue-Collar	Husband	White	Male	5779	0	50	United-States	> $50k/y
29	Private	Bachelors	Never-Married	Sales	Not-in-family	White	Female	0	0	60	United-States	≤ $50k/y
29	Private	Bachelors	Never-Married	Sales	Not-in-family	White	Female	7786	0	60	United-States	> $50k/y
30	Private	High School grad	Never-Married	Blue-Collar	Not-in-family	White	Male	0	0	40	United-States	≤ $50k/y
30	Private	High School grad	Never-Married	Blue-Collar	Not-in-family	White	Male	9465	0	40	United-States	> $50k/y
31	Private	Bachelors	Married	Admin	Husband	Asian-Pac-Islander	Male	0	0	60	British-Commonwealth	> $50k/y
33	Private	Bachelors	Married	Blue-Collar	Husband	Asian-Pac-Islander	Male	12	0	60	British-Commonwealth	≤ $50k/y
28	Private	Bachelors	Married	White-Collar	Husband	White	Male	0	0	45	United-States	> $50k/y
28	Private	Bachelors	Married	Blue-Collar	Husband	White	Male	9	1	45	United-States	≤ $50k/y
39	Private	Prof-School	Married	Professional	Wife	White	Female	7688	0	60	United-States	> $50k/y
39	Private	Masters	Married	Blue-Collar	Wife	White	Female	4012	8	60	United-States	≤ $50k/y
54	Private	High School grad	Married	Sales	Husband	White	Male	0	0	40	United-States	> $50k/y
54	Private	High School grad	Married	Blue-Collar	Husband	White	Male	6	0	40	United-States	≤ $50k/y
49	Private	High School grad	Married	Sales	Husband	White	Male	5013	0	40	United-States	> $50k/y
49	Private	High School grad	Married	Blue-Collar	Husband	White	Male	3213	0	40	United-States	≤ $50k/y
58	Self-emp-inc	High School grad	Married	Sales	Husband	White	Male	0	0	55	United-States	> $50k/y
58	Self-emp-inc	High School grad	Married	Blue-Collar	Husband	White	Male	12	2	55	United-States	≤ $50k/y
42	Private	Bachelors	Married	White-Collar	Husband	White	Male	0	0	40	United-States	> $50k/y
42	Private	Bachelors	Married	Blue-Collar	Husband	White	Male	45	2	40	United-States	≤ $50k/y
36	Local-gov	Bachelors	Married	Professional	Husband	White	Male	0	1672	50	United-States	> $50k/y
36	Local-gov	Bachelors	Married	Blue-Collar	Husband	White	Male	0	1637	50	United-States	≤ $50k/y
56	Private	Bachelors	Married	Professional	Husband	White	Male	0	0	40	United-States	> $50k/y
56	Private	Bachelors	Married	Blue-Collar	Husband	White	Male	48	2	40	United-States	≤ $50k/y
28	Private	Bachelors	Married	Professional	Wife	White	Female	0	1977	24	United-States	> $50k/y
28	Private	Dropout	Married	Blue-Collar	Wife	White	Female	0	1624	24	United-States	≤ $50k/y

Figure 6: Conditional counterfactual instances on the Adult dataset. Odd rows correspond to the input instances and even rows correspond to the counterfactual instances. Grayed out features correspond to immutable features. All instances are conditioned on the same conditional vector corresponding to: 1) *Age* allowed to increase by up to 10; 2) *Workclass* allowed to change to {*Private*, *Without-pay*} or stay the same; 3) *Education* allowed to change to {*Bachelors*, *Masters*, *Dropout*} or stay the same; 4) *Occupation* allowed to change to {*Sales*, *White-Collar*, *Blue-Collar*} or stay the same; 5) *Capital Gain* allowed to increase or decrease by up to 10000; 6) *Capital Loss* allowed to increase or decrease by up to 10000; 7) *Hours per week* allowed to increase or decrease by up to 20; 8) *Country* cannot change. Feature changes are highlighted in red.

C.2 Samples MNIST

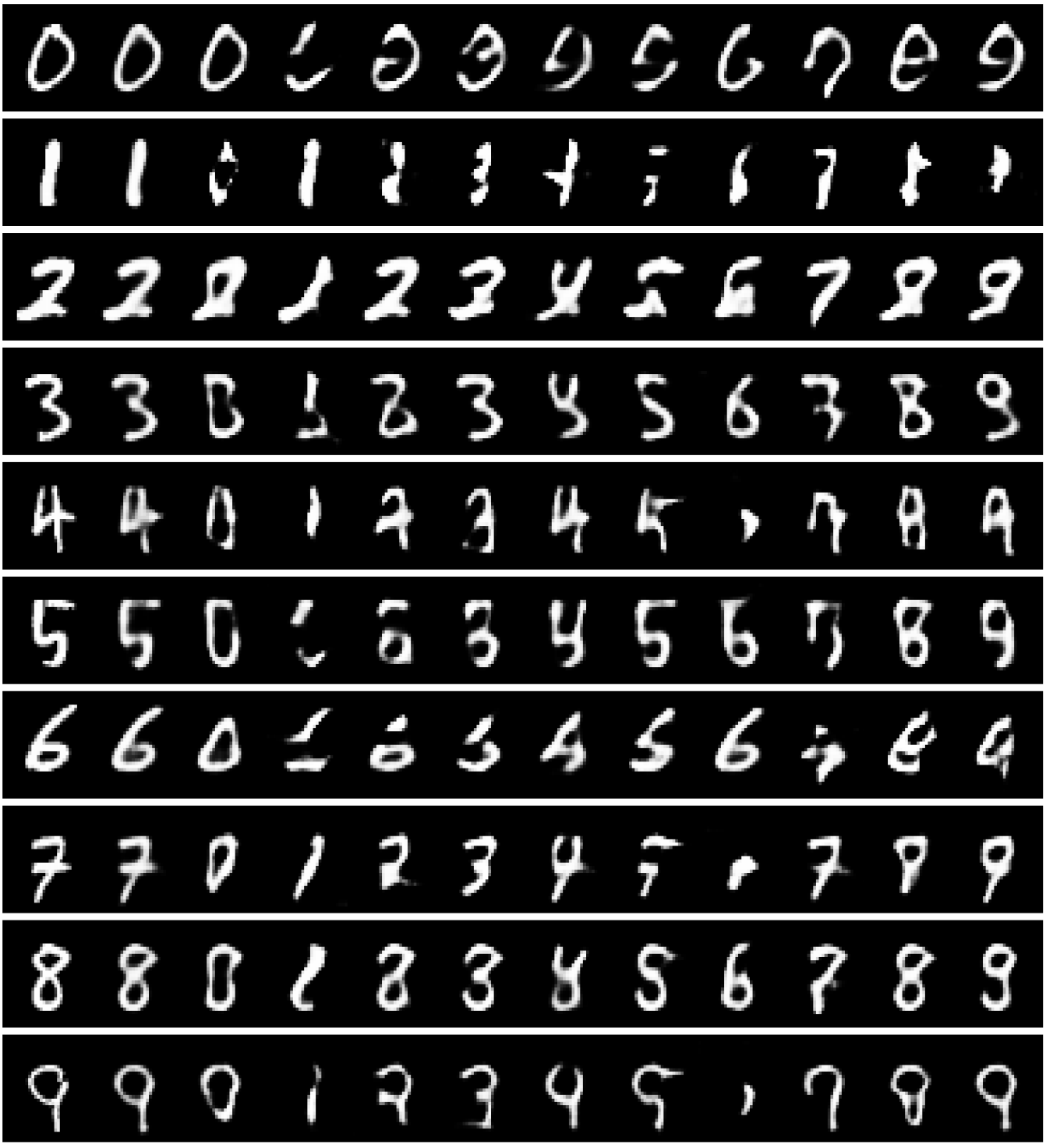

Figure 7: One instance to all classes on MNIST dataset. Starting from left to right: original instance, reconstructed instance, counterfactual instance for each class staring from 0 to 9.

C.3 Samples CELEBA

Figure 8: One instance to all classes on CelebA dataset. Starting from left to right: original instance, reconstructed instance, counterfactual instance for each class: young and smiling, young and not smiling, old and smiling, old and not smiling.

References

Blackard, J. A. 1998. Covertype Data Set.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 785–794. New York, NY, USA: Association for Computing Machinery. ISBN 9781450342322.

Cheng, F.; Ming, Y.; and Qu, H. 2020. DECE: Decision Explorer with Counterfactual Explanations for Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics*.

Cortes, C.; and Vapnik, V. 1995. Support-Vector Networks. *Mach. Learn.*, 20(3): 273–297.

Dhurandhar, A.; Chen, P.-Y.; Luss, R.; Tu, C.-C.; Ting, P.; Shanmugam, K.; and Das, P. 2018. Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13(25): 723–773.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Pedreschi, D.; Turini, F.; and Giannotti, F. 2018. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*.

Hashemi, M.; and Fathi, A. 2020. PermuteAttack: Counterfactual Explanation of Machine Learning Credit Scorecards. *arXiv preprint arXiv:2008.10138*.

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

Karimi, A.-H.; Barthe, G.; Schölkopf, B.; and Valera, I. 2020. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klaise, J.; Looveren, A. V.; Vacanti, G.; and Coca, A. 2021. Alibi Explain: Algorithms for Explaining Machine Learning Models. *Journal of Machine Learning Research*, 22(181): 1–7.

Kulesza, A.; and Taskar, B. 2012. *Determinantal Point Processes for Machine Learning*. Hanover, MA, USA: Now Publishers Inc. ISBN 1601986289.

Laugel, T.; Lesot, M.-J.; Marsala, C.; Renard, X.; and Detyniecki, M. 2017. Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*.

LeCun, Y. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In Bengio, Y.; and LeCun, Y., eds., *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Liu, S.; Kailkhura, B.; Loveland, D.; and Han, Y. 2019. Generative Counterfactual Introspection for Explainable Deep Learning. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 1–5.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Mahajan, D.; Tan, C.; and Sharma, A. 2019. Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers. arXiv:1912.03277.

Martens, D.; and Provost, F. 2014. Explaining Data-Driven Document Classifications. *MIS Q.*, 38(1): 73–100.

McInnes, L.; Healy, J.; and Melville, J. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. Cite arxiv:1802.03426Comment: Reference implementation available at http://github.com/lmcinnes/umap.

Mirza, M.; and Osindero, S. 2014. Conditional Generative Adversarial Nets. arXiv:1411.1784.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Moro, S.; Cortez, P.; and Rita, P. 2014. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62: 22–31.

Mothilal, R. K.; Sharma, A.; and Tan, C. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 607–617.

Oh, K.; Yoon, J. S.; and Suk, H.-I. 2020. Born Identity Network: Multi-way Counterfactual Map Generation to Explain a Classifier's Decision. arXiv:2011.10381.

Rabanser, S.; Günnemann, S.; and Lipton, Z. 2019. Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

Sharma, S.; Henderson, J.; and Ghosh, J. 2019. Certifai: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. *arXiv preprint arXiv:1905.07857*.

Statistics, L. B.; and Breiman, L. 2001. Random Forests. In *Machine Learning*, 5–32.

Stepin, I.; Alonso, J. M.; Catala, A.; and Pereira-Fariña, M. 2021. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9: 11974–12001.

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

Van Looveren, A.; Klaise, J.; Vacanti, G.; and Cobb, O. 2021. Conditional Generative Models for Counterfactual Explanations. *arXiv preprint arXiv:2101.10123*.

Van Looveren, A.; Vacanti, G.; Klaise, J.; Coca, A.; and Cobb, O. 2019. Alibi Detect: Algorithms for outlier, adversarial and drift detection.

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

Wexler, J.; Pushkarna, M.; Bolukbasi, T.; Wattenberg, M.; Viégas, F.; and Wilson, J. 2019. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1): 56–65.

White, A.; and Garcez, A. d. 2019. Measurable counterfactual local explanations for any classifier. *arXiv preprint arXiv:1908.03020*.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2242–2251.

Zwitter, M.; and Soklic, M. 1988. Breast Cancer Data Set.