CeFlow: A Robust and Efficient Counterfactual Explanation Framework for Tabular Data using Normalizing Flows

Tri Dung Duong[1], Qian Li[2], and Guandong Xu[1⋆]

[1] Faculty of Engineering and Information Technology, University of Technology Sydney, NSW, Australia
[2] School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, WA, Australia

Abstract. Counterfactual explanation is a form of interpretable machine learning that generates perturbations on a sample to achieve the desired outcome. The generated samples can act as instructions to guide end users on how to observe the desired results by altering samples. Although state-of-the-art counterfactual explanation methods are proposed to use variational autoencoder (VAE) to achieve promising improvements, they suffer from two major limitations: 1) the counterfactuals generation is prohibitively slow, which prevents algorithms from being deployed in interactive environments; 2) the counterfactual explanation algorithms produce unstable results due to the randomness in the sampling procedure of variational autoencoder. In this work, to address the above limitations, we design a robust and efficient counterfactual explanation framework, namely CeFlow, which utilizes normalizing flows for the mixed-type of continuous and categorical features. Numerical experiments demonstrate that our technique compares favorably to state-of-the-art methods. We release our source code[3] for reproducing the results.

Keywords: Counterfactual explanation · Normalizing flow · Interpretable machine learning.

1 Introduction

Machine learning (ML) has resulted in advancements in a variety of scientific and technical fields, including computer vision, natural language processing, and conversational assistants. Interpretable machine learning is a machine learning sub-field that aims to provide a collection of tools, methodologies, and algorithms capable of producing high-quality explanations for machine learning model judgments. A great deal of methods in interpretable ML methods has been proposed in recent years. Among these approaches, counterfactual explanation (CE) is the

⋆ Corresponding author: Guandong.Xu@uts.edu.au
[3] https://github.com/tridungduong16/fairCE.git

prominent example-based method involved in how to alter features to change the model predictions and thus generates counterfactual samples for explaining and interpreting models [20, 1, 8, 28, 31]. An example is that for a customer `A` rejected by a loan application, counterfactual explanation algorithms aim to generate counterfactual samples such as "your loan would have been approved if your income was $51,000 more" which can act as a recommendation for a person to achieve the desired outcome. Providing counterfactual samples for black-box models has the capability to facilitate human-machine interaction, thus promoting the application of ML models in several fields.

The recent studies in counterfactual explanation utilize variational autoencoder (VAE) as a generative model to generate counterfactual sample [23, 20]. Specifically, the authors first build an encoder and decoder model from the training data. Thereafter, the original input would go through the encoder model to obtain the latent representation. They make the perturbation into this representation and pass the perturbed vector to the encoder until getting the desired output. However, these approaches present some limitations. First, the latent representation which is sampled from the encoder model would be changed corresponding to different sampling times, leading to unstable counterfactual samples. Thus, the counterfactual explanation algorithm is not robust when deployed in real applications. Second, the process of making perturbation into latent representation is so prohibitively slow [20] since they need to add random vectors to the latent vector repeatedly; accordingly, the running time of algorithms grows significantly. Finally, the generated counterfactual samples are not closely connected to the density region, making generated explanations infeasible and non-actionable. To address all of these limitations, we propose a Flow-based counterfactual explanation framework (CeFlow) that integrates normalizing flow which is an invertible neural network as the generative model to generate counterfactual samples. Our contributions can be summarized as follows:

- We introduce CeFlow, an efficient and robust counterfactual explanation framework that leverages the power of normalizing flows in modeling data distributions to generate counterfactual samples. The usage of flow-based models enables to produce more robust counterfactual samples and reduce the algorithm running time.
- We construct a conditional normalizing flow model that can deal with tabular data consisting of continuous and categorical features by utilizing variational dequantization and Gaussian mixture models.
- The generated samples from CeFlow are close to and related to high-density regions of other data points with the desired class. This makes counterfactual samples likely reachable and therefore naturally follow the distribution of the dataset.

2 Related works

An increasing number of methods have been proposed for the counterfactual explanation. The existing methods can be categorized into gradient-based methods

[28, 21], auto-encoder model [20], heuristic search methods [24, 25] and integer linear optimization [15]. Regarding gradient-based methods, The authors in the study construct the cross-entropy loss between the desired class and counterfactual samples' prediction with the purpose of changing the model output. The created loss would then be minimized using gradient-descent optimization methods. In terms of auto-encoder model, generative models such as variational auto-encoder (VAE) is used to generate new samples in another line of research. The authors [23] first construct an encoder-decoder architecture. They then utilize the encoder to generate the latent representation, make some changes to it, and run it through the decoder until the prediction models achieve the goal class. However, VAE models which maximize the lower bound of the log-likelihood instead of measuring exact log-likelihood can produce unstable and unreliable results. On the other hand, there is an increasing number of counterfactual explanation methods based on heuristic search to select the best counterfactual samples such as Nelder-Mead [9], growing spheres [19], FISTA [4, 27], or genetic algorithms [3, 17]. Finally, the studies [26] propose to formulate the problem of finding counterfactual samples as a mixed-integer linear optimization problem and utilize some existing solvers [2, 1] to obtain the optimal solution.

3 Preliminaries

Throughout the paper, lower-cased letters x and $\boldsymbol{x}$ denote the deterministic scalars and vectors, respectively. We consider a classifier $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ that has the input of feature space $\mathcal{X}$ and the output as $\mathcal{Y} = \{1...\mathcal{C}\}$ with $\mathcal{C}$ classes. Meanwhile, we denote a dataset $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$ consisting of N instances where $\boldsymbol{x}_n \in \mathcal{X}$ is a sample, $y_n \in \mathcal{Y}$ is the predicted label of individuals $\boldsymbol{x}_n$ from the classifier $\mathcal{H}$. Moreover, f_θ is denoted for a normalizing flow model parameterized by θ. Finally, we split the feature space into two disjoint feature subspaces of categorical features and continuous features represented by $\mathcal{X}^{\text{cat}}$ and $\mathcal{X}^{\text{con}}$ respectively such that $\mathcal{X} = \mathcal{X}_{\text{cat}} \times \mathcal{X}_{\text{con}}$ and $\boldsymbol{x} = (\boldsymbol{x}^{\text{cat}}, \boldsymbol{x}^{\text{con}})$, and $\boldsymbol{x}^{\text{cat}_j}$ and $\boldsymbol{x}^{\text{con}_j}$ is the corresponding j-th feature of $\boldsymbol{x}^{\text{cat}}$ and $\boldsymbol{x}^{\text{con}}$.

3.1 Counterfactual explanation

With the original sample $\boldsymbol{x}_{\text{org}} \in \mathcal{X}$ and its predicted output $y_{\text{org}} \in \mathcal{Y}$, the counterfactual explanation aims to find the nearest counterfactual sample $\boldsymbol{x}_{\text{cf}}$ such that the outcome of classifier for $\boldsymbol{x}_{\text{cf}}$ is changed to desired output class y_{cf}. We aim to identify the perturbation $\boldsymbol{\delta}$ such that counterfactual instance $\boldsymbol{x}_{\text{cf}} = \boldsymbol{x}_{\text{org}} + \boldsymbol{\delta}$ is the solution of the following optimization problem:

$$\boldsymbol{x}_{\text{cf}} = \underset{\boldsymbol{x}_{\text{cf}} \in \mathcal{X}}{\arg\min}\, d(\boldsymbol{x}_{\text{cf}}, \boldsymbol{x}_{\text{org}}) \quad \text{subject to} \quad \mathcal{H}(\boldsymbol{x}_{\text{cf}}) = y_{\text{cf}} \tag{1}$$

where $d(\boldsymbol{x}_{\text{cf}}, \boldsymbol{x}_{\text{org}})$ is the function measuring the distance between $\boldsymbol{x}_{\text{org}}$ and $\boldsymbol{x}_{\text{cf}}$. Eq (1) demonstrates the optimization objective that minimizes the similarity of the counterfactual and original samples, as well as ensures to change the classifier

to the desirable outputs. To make the counterfactual explanations plausible, they should only suggest minimal changes in features of the original sample. [21].

3.2 Normalizing flow

Normalizing flows (NF) [5] is the active research direction in generative models that aims at modeling the probability distribution of a given dataset. The study [6] first proposes a normalizing flow, which is an unsupervised density estimation model described as an invertible mapping $f_\theta : \mathcal{X} \to \mathcal{Z}$ from the data space $\mathcal{X}$ to the latent space $\mathcal{Z}$. Function f_θ can be designed as a neural network parametrized by θ with architecture that has to ensure invertibility and efficient computation of log-determinants. The data distribution is modeled as a transformation $f_\theta^{-1} : \mathcal{Z} \to \mathcal{X}$ applied to a random variable from the latent distribution $\boldsymbol{z} \sim p_{\mathcal{Z}}$, for which Gaussian distribution is chosen. The change of variables formula gives the density of the converted random variable $\boldsymbol{x} = f_\theta^{-1}(\boldsymbol{z})$ as follows:

$$\begin{aligned} p_{\mathcal{X}}(\boldsymbol{x}) &= p_{\mathcal{Z}}(f_\theta(\boldsymbol{x})) \cdot \left| \det \left(\frac{\partial f_\theta}{\partial \boldsymbol{x}} \right) \right| \\ &\propto \log \left(p_{\mathcal{Z}}(f_\theta(\boldsymbol{x})) \right) + \log \left(\left| \det \left(\frac{\partial f_\theta}{\partial \boldsymbol{x}} \right) \right| \right) \end{aligned} \tag{2}$$

With N training data points $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$, the model with respects to parameters θ can be trained by maximizing the likelihood in Equation (3):

$$\theta = \arg\max_\theta \left(\prod_{n=1}^{N} \left(\log(p_{\mathcal{Z}}(f_\theta(\boldsymbol{x}_n))) + \log \left(\left| \det \left(\frac{\partial f_\theta(\boldsymbol{x}_n)}{\partial \boldsymbol{x}_n} \right) \right| \right) \right) \right) \tag{3}$$

4 Methodology

In this section, we illustrate our approach (CeFlow) which leverages the power of normalizing flow in generating counterfactuals. First, we define the general architecture of our framework in section 4.1. Thereafter, section 4.2 and 4.3 illustrate how to train and build the architecture of the invertible function f for tabular data, while section 4.4 describes how to produce the counterfactual samples by adding the perturbed vector into the latent representation.

4.1 General architecture of CeFlow

Figure 1 generally illustrates our framework. Let $\boldsymbol{x}_{\text{org}}$ be an original instance, and f_θ denote a pre-trained, invertible and differentiable normalizing flow model on the training data. In general, we first construct an invertible and differentiable function f_θ that converts the original instance $\boldsymbol{x}_{\text{org}}$ to the latent representation $\boldsymbol{z}_{\text{org}} = f(\boldsymbol{x}_{\text{org}})$. After that, we would find the scaled vector $\boldsymbol{\delta}_z$ as the perturbation and add to the latent representation $\boldsymbol{z}_{\text{org}}$ to get the perturbed representation $\boldsymbol{z}_{\text{cf}}$ which goes through the inverse function f_θ^{-1} to produce the counterfactual

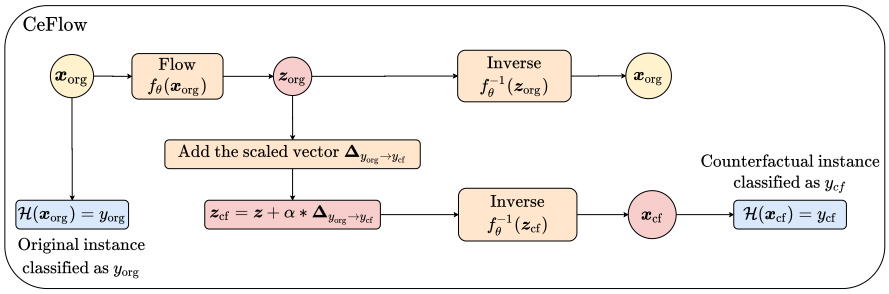

Fig. 1: Counterfactual explanation with normalizing flows (CeFlow).

instance $\boldsymbol{x}_{\text{cf}}$. With the counterfactual instance $\boldsymbol{x}_{\text{cf}} = f_\theta^{-1}(\boldsymbol{z}_{\text{org}} + \boldsymbol{\delta}_z)$, we can re-write the objective function Eq. (1) into the following form:

$$\begin{cases} \boldsymbol{\delta}_z = \arg\min_{\boldsymbol{\delta}_z \in \mathcal{Z}} d(\boldsymbol{x}_{\text{org}}, \boldsymbol{\delta}_z) \\ \mathcal{H}(\boldsymbol{x}_{\text{cf}}) = y_{\text{cf}} \end{cases} \tag{4}$$

One of the biggest problems of deploying normalizing flow is how to handle mixed-type data which contains both continuous and categorical features. Categorical features are in discrete forms, which is challenging to model by the continuous distribution only [10]. Another challenge is to construct the objective function to learn the conditional distribution on the predicted labels [30, 14]. In the next section, we will discuss how to construct the conditional normalizing flow f_θ for tabular data.

4.2 Normalizing flows for categorical features

This section would discuss how to handle the categorical features. Let $\{\boldsymbol{z}^{\text{cat}_m}\}_{m=1}^M$ be the continuous representation of M categorical features $\{\boldsymbol{x}^{\text{cat}_m}\}_{m=1}^M$ for each $\boldsymbol{x}^{\text{cat}_m} \in \{0, 1, ..., K-1\}$ with $K > 1$. Follow by several studies in the literature [10, 12], we utilize variational dequantization to model the categorical features. The key idea of variational dequantization is to add noise $\boldsymbol{u}$ to the discrete values $\boldsymbol{x}^{\text{cat}}$ to convert the discrete distribution $p_{\mathcal{X}^{\text{cat}}}$ into a continuous distribution $p_{\phi_{\text{cat}}}$. With $\boldsymbol{z}^{\text{cat}} = \boldsymbol{x}^{\text{cat}} + \boldsymbol{u}_k$, ϕ_{cat} and θ_{cat} be models' parameters, we have following objective functions:

$$\begin{aligned} \log p_{\mathcal{X}^{\text{cat}}}(\boldsymbol{x}^{\text{cat}}) &\geq \int_{\boldsymbol{u}} \log \frac{p_{\phi_{\text{cat}}}(\boldsymbol{z}^{\text{cat}})}{q_{\theta_{\text{cat}}}(\boldsymbol{u}|\boldsymbol{x}^{\text{cat}})} du \\ &\approx \frac{1}{K} \sum_{k=1}^{K} \log \prod_{m=1}^{M} \frac{p_{\phi_{\text{cat}}}(\boldsymbol{x}^{\text{cat}_m} + \boldsymbol{u}_k)}{q_{\theta_{\text{cat}}}(\boldsymbol{u}_k|\boldsymbol{x}^{\text{cat}})} \end{aligned} \tag{5}$$

Followed the study [12], we choose Gaussian dequantization which is more powerful than the uniform dequantization as $q_{\theta_{\text{cat}}}(\boldsymbol{u}_k|\boldsymbol{x}^{\text{cat}}) = \text{sig}\left(\mathcal{N}\left(\boldsymbol{\mu}_{\theta_{\text{cat}}}, \boldsymbol{\Sigma}_{\theta_{\text{cat}}}\right)\right)$ with mean $\boldsymbol{\mu}_{\theta_{\text{cat}}}$, covariance $\boldsymbol{\Sigma}_{\theta_{\text{cat}}}$ and sigmoid function sig($\cdot$).

4.3 Conditional Flow Gaussian Mixture Model for tabular data

The categorical features $\boldsymbol{x}^{\text{cat}}$ going through the the variational dequantization would convert into continuous representation $\boldsymbol{z}^{\text{cat}}$. We then perform merge operation on continuous representation $\boldsymbol{z}^{\text{cat}}$ and continuous feature $\boldsymbol{x}^{\text{con}}$ to obtain values $(\boldsymbol{z}^{\text{cat}}, \boldsymbol{x}^{\text{con}}) \mapsto \boldsymbol{x}^{\text{full}}$. Thereafter, we apply flow Gaussian mixture model [14] which is a probabilistic generative model for training the invertible function f_θ. For each predicted class label $y \in \{1...\mathcal{C}\}$, the latent space distribution $p_{\mathcal{Z}}$ conditioned on a label k is the Gaussian distribution $\mathcal{N}\left(\boldsymbol{z}^{\text{full}} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$ with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$:

$$p_{\mathcal{Z}}(\boldsymbol{z}^{\text{full}} \mid y = k) = \mathcal{N}\left(\boldsymbol{z}^{\text{full}} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{6}$$

As a result, we can have the marginal distribution of $\boldsymbol{z}^{\text{full}}$:

$$p_{\mathcal{Z}}(\boldsymbol{z}^{\text{full}}) = \frac{1}{\mathcal{C}} \sum_{k=1}^{\mathcal{C}} \mathcal{N}\left(\boldsymbol{z}^{\text{full}} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{7}$$

The density of the transformed random variable $\boldsymbol{x}^{\text{full}} = f_\theta^{-1}(\boldsymbol{z}^{\text{full}})$ is given by:

$$p_{\mathcal{X}}(\boldsymbol{x}^{\text{full}}) = \log\left(p_{\mathcal{Z}}(f_\theta(\boldsymbol{x}^{\text{full}}))\right) + \log\left(\left|\det\left(\frac{\partial f_\theta}{\partial \boldsymbol{x}^{\text{full}}}\right)\right|\right) \tag{8}$$

Eq. (7) and Eq. (8) together lead to the likelihood for data as follows:

$$p_{\mathcal{X}}(\boldsymbol{x}^{\text{full}} \mid y = k) = \log\left(\mathcal{N}\left(f_\theta(\boldsymbol{x}^{\text{full}}) \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right) + \log\left(\left|\det\left(\frac{\partial f_\theta}{\partial \boldsymbol{x}^{\text{full}}}\right)\right|\right) \tag{9}$$

We can train the model by maximizing the joint likelihood of the categorical and continuous features on N training data points $\mathcal{D} = \{(\boldsymbol{x}_n^{\text{con}}, \boldsymbol{x}_n^{\text{cat}})\}_{n=1}^N$ by combining Eq. (5) and Eq. (9):

$$\begin{aligned}
\theta^*, \phi_{\text{cat}}^*, \theta_{\text{cat}}^* &= \underset{\theta, \phi_{\text{cat}}, \theta_{\text{cat}}}{\arg\max} \prod_{n=1}^{N} \left(\prod_{\boldsymbol{x}_n^{\text{con}} \in \boldsymbol{\mathcal{X}}^{\text{con}}} p_{\mathcal{X}}\left(\boldsymbol{x}_n^{\text{con}}\right) \prod_{\boldsymbol{x}_n^{\text{cat}} \in \boldsymbol{\mathcal{X}}^{\text{cat}}} p_{\mathcal{X}}\left(\boldsymbol{x}_n^{\text{cat}}\right) \right) \\
&= \underset{\theta, \phi_{\text{cat}}, \theta_{\text{cat}}}{\arg\max} \prod_{n=1}^{N} \left(\log\left(\mathcal{N}\left(f_\theta(\boldsymbol{x}_n^{\text{full}}) \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right) + \log\left(\left|\det\left(\frac{\partial f_\theta}{\partial \boldsymbol{x}_n^{\text{full}}}\right)\right|\right) \right)
\end{aligned} \tag{10}$$

4.4 Counterfactual generation step

In order to find counterfactual samples, the recent approaches [21, 28] normally define the loss function and deploy some optimization algorithm such as gradient descent or heuristic search to find the perturbation. These approaches however demonstrates the prohibitively slow running time, which prevents from deploying in interactive environment[11]. Therefore, inspired by the study [13], we add

the scaled vector as the perturbation from the original instance $\boldsymbol{x}_{\text{org}}$ to counterfactual one $\boldsymbol{x}_{\text{cf}}$. By Bayes' rule, we notice that under a uniform prior distribution over labels $p(y=k)=\frac{1}{\mathcal{C}}$ for $\mathcal{C}$ classes, the log posterior probability becomes:

$$\log p_{\mathcal{X}}(y=k|\boldsymbol{x}) = \log \frac{p_{\mathcal{X}}(\boldsymbol{x}|y=k)}{\sum_{k=1}^{\mathcal{C}} p_{\mathcal{X}}(\boldsymbol{x}|y=k)} \propto ||f_\theta(\boldsymbol{x}) - \boldsymbol{\mu}_k||^2 \tag{11}$$

We observed from Eq. (11) that latent vector $\boldsymbol{z}=f_\theta(\boldsymbol{x})$ will be predicted from the class y with the closest model mean $\boldsymbol{\mu}_k$. For each predicted class $k \in \{1...\mathcal{C}\}$, we denote $\mathcal{G}_k = \{\boldsymbol{x}_m, y_m\}_{m=1}^{M}$ as a set of M instances with the same predicted class as $y_m = k$. We define the mean latent vector $\boldsymbol{\mu}_k$ corresponding to each class k such that:

$$\boldsymbol{\mu}_k = \frac{1}{M} \sum_{\boldsymbol{x}_m \in \mathcal{G}_k} f_\theta(\boldsymbol{x}_m) \tag{12}$$

Therefore, the scaled vector that moves the latent vector $\boldsymbol{z}_{\text{org}}$ to the decision boundary from the original class y_{org} to counterfactual class y_{cf} is defined as:

$$\boldsymbol{\Delta}_{y_{\text{org}} \to y_{\text{cf}}} = \left| \boldsymbol{\mu}_{y_{\text{org}}} - \boldsymbol{\mu}_{y_{\text{cf}}} \right| \tag{13}$$

The scaled vector $\boldsymbol{\Delta}_{y_{\text{org}} \to y_{\text{cf}}}$ is added to the original latent representation $\boldsymbol{z}_{\text{cf}} = f_\theta(\boldsymbol{x}_{\text{org}})$ to obtained the perturbed vector. The perturbed vector then goes through inverted function f_θ^{-1} to re-produce the counterfactual sample:

$$\boldsymbol{x}_{\text{cf}} = f_\theta^{-1}(f_\theta(\boldsymbol{x}_{\text{org}}) + \alpha \boldsymbol{\Delta}_{y_{\text{org}} \to y_{\text{cf}}}) \tag{14}$$

We note that the hyperparameter α needs to be optimized by searching in a range of values. The full algorithm is illustrated in Algorithm 1.

Algorithm 1 Counterfactual explanation flow (CeFlow)

Input: An original sample $\boldsymbol{x}_{\text{org}}$ with its prediction y_{org}, desired class y_{cf}, a provided machine learning classifier $\mathcal{H}$ and encoder model Q_ϕ.

1: Train the invertible function f_θ by maximizing the log-likelihood:

$$\begin{aligned}\theta^*, \phi^*_{\text{cat}}, \theta^*_{\text{cat}} &= \underset{\theta, \phi_{\text{cat}}, \theta_{\text{cat}}}{\arg\max} \prod_{n=1}^{N} \left(\prod_{\boldsymbol{x}_n^{\text{con}} \in \mathcal{X}^{\text{con}}} p_{\mathcal{X}}\left(\boldsymbol{x}_n^{\text{con}}\right) \prod_{\boldsymbol{x}_n^{\text{cat}} \in \mathcal{X}^{\text{cat}}} p_{\mathcal{X}}\left(\boldsymbol{x}_n^{\text{cat}}\right) \right) \\ &= \underset{\theta, \phi_{\text{cat}}, \theta_{\text{cat}}}{\arg\max} \prod_{n=1}^{N} \left(\log\left(\mathcal{N}\left(f_\theta(\boldsymbol{x}_n^{\text{full}}) \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right) + \log\left(\left|\det\left(\frac{\partial f_\theta}{\partial \boldsymbol{x}_n^{\text{full}}}\right)\right|\right)\right)\end{aligned}$$

2: Compute mean latent vector $\boldsymbol{\mu}_k$ for each class k by $\boldsymbol{\mu}_k = \frac{1}{M}\sum_{\boldsymbol{x}_m \in \mathcal{G}_k} f(\boldsymbol{x}_m)$.

3: Compute the scaled vector $\boldsymbol{\Delta}_{y_{\text{org}} \to y_{\text{cf}}} = \left| \boldsymbol{\mu}_{y_{\text{org}}} - \boldsymbol{\mu}_{y_{\text{cf}}} \right|$.

4: Find the optimal hyperparameter α by searching a range of values.

5: Compute $\boldsymbol{x}_{\text{cf}} = f_\theta^{-1}(f_\theta(\boldsymbol{x}_{\text{org}}) + \alpha \boldsymbol{\Delta}_{y_{\text{org}} \to y_{\text{cf}}})$.

Output: $\boldsymbol{x}_{\text{cf}}$.

5 Experiments

We run experiments on three datasets to show that our method outperforms state-of-the-art approaches. The specification of hardware for the experiment is Python 3.8.5 with 64-bit Red Hat, Intel(R) Xeon(R) Gold 6238R CPU @ 2.20GHz. We implement our algorithm by using Pytorch library and adopt the RealNVP architecture [6]. During training progress, Gaussian mixture parameters are fixed: the means are initialized randomly from the standard normal distribution and the covariances are set to I. More details of implementation settings can be found in our code repository[4].

We evaluate our approach via three datasets: `Law` [29], `Compas` [16] and `Adult` [7]. `Law`[5][29] dataset provides information of students with their features: their entrance exam scores (LSAT), grade-point average (GPA) and first-year average grade (FYA). `Compas`[6][16] dataset contains information about 6,167 prisoners who have features including gender, race and other attributes related to prior conviction and age. `Adult`[7][7] dataset is a real-world dataset consisting of both continuous and categorical features of a group of consumers who apply for a loan at a financial institution.

We compare our proposed method (CeFlow) with several state-to-the-art methods including Actionable Recourse (AR) [26], Growing Sphere (GS) [18], FACE [24], CERTIFAI [25], DiCE [21] and C-CHVAE [23]. Particularly, we implement the CERTIFAI with library PyGAD[8] and utilize the available source code[9] for implementation of DiCE, while other approaches are implemented with Carla library [22]. Finally, we report the results of our proposed model on a variety of metrics including success rate (success), l_1-norm (l_1), categorical proximity [21], continuous proximity [21] and mean log-density [1]. Note that for l_1-norm, we report mean and variance of l_1-norm corresponding to l_1-mean and l_1-variance. Lower l_1-variance aims to illustrate the algorithm's robustness.

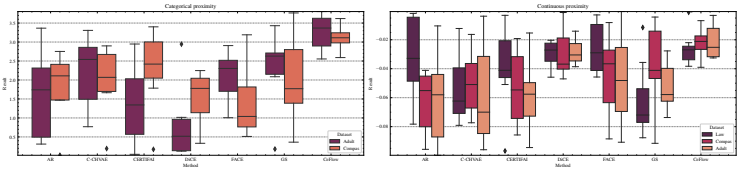

Fig. 2: Baseline results in terms of **Categorical proximity** and **Continuous proximity**. Higher continuous and categorical proximity are better.

[4] https://anonymous.4open.science/r/fairCE-538B
[5] http://www.seaphe.org/databases.php
[6] https://www.propublica.org
[7] https://archive.ics.uci.edu/ml/datasets/adult
[8] https://github.com/ahmedfgad/GeneticAlgorithmPython
[9] https://github.com/divyat09/cf-feasibility

Table 1: Performance of all methods on the classifier. We compute p-value by conducting a paired t-test between our approach (CeFlow) and baselines with 100 repeated experiments for each metric.

Dataset	Method	Performance				p-value		
		success	l_1-mean	l_1-var	log-density	success	l_1	log-density
Law	AR	98.00	3.518	2.0e-03	-0.730	0.041	0.020	0.022
	GS	100.00	3.600	2.6e-03	-0.716	0.025	0.048	0.016
	FACE	100.00	3.435	2.0e-03	-0.701	0.029	0.010	0.017
	CERTIFAI	100.00	3.541	2.0e-03	-0.689	0.029	0.017	0.036
	DiCE	94.00	**3.111**	2.0e-03	-0.721	0.018	0.035	0.048
	C-CHVAE	100.00	3.461	1.0e-03	-0.730	0.040	0.037	0.016
	CeFlow	**100.00**	3.228	**1.0e-05**	**-0.679**	-	-	-
Compas	AR	97.50	1.799	2.4e-03	-14.92	0.038	0.034	0.046
	GS	100.00	1.914	3.2e-03	-14.87	0.019	0.043	0.040
	FACE	98.50	1.800	4.8e-03	-15.59	0.036	0.024	0.035
	CERTIFAI	100.00	1.811	2.4e-03	-15.65	0.040	0.048	0.038
	DiCE	95.50	1.853	2.9e-03	-14.68	0.030	0.029	0.018
	C-CHVAE	100.00	1.878	1.1e-03	-13.97	0.026	0.015	0.027
	CeFlow	**100.00**	**1.787**	**1.8e-05**	**-13.62**	-	-	-
Adult	AR	100.00	3.101	7.8e-03	-25.68	0.044	0.037	0.018
	GS	100.00	3.021	2.4e-03	-26.55	0.026	0.049	0.028
	FACE	100.00	2.991	6.6e-03	-23.57	0.027	0.015	0.028
	CERTIFAI	93.00	3.001	4.1e-03	-25.55	0.028	0.022	0.016
	DiCE	96.00	2.999	9.1e-03	-24.33	0.046	0.045	0.045
	C-CHVAE	100.00	3.001	8.7e-03	-24.45	0.026	0.043	0.019
	CeFlow	**100.00**	**2.964**	**1.5e-05**	**-23.46**	-	-	-

Table 2: We report running time of different methods on three datasets.

Dataset	AR	GS	FACE	CERTIFAI	DiCE	C-CHVAE	CeFlow
Law	3.030 ± 0.105	7.126 ± 0.153	6.213 ± 0.007	6.522 ± 0.088	8.022 ± 0.014	9.022 ± 0.066	**0.850 ± 0.055**
Compas	5.125 ± 0.097	8.048 ± 0.176	7.688 ± 0.131	13.426 ± 0.158	7.810 ± 0.076	6.879 ± 0.044	**0.809 ± 0.162**
Adult	7.046 ± 0.151	6.472 ± 0.021	13.851 ± 0.001	7.943 ± 0.046	11.821 ± 0.162	12.132 ± 0.024	**0.837 ± 0.026**

The performance of different approaches regarding three metrics: l_1, success metrics and log-density are illustrated in Table 1. Regarding success rate, all three methods achieve competitive results, except the AR, DiCE and CERTIFAI performance in all datasets with around 90% of samples belonging to the target class. These results indicate that by integrating normalizing flows into counterfactuals generation, our proposed method can achieve the target of counterfactual explanation task for changing the models' decision. Apart from that, for l_1-mean, CeFlow is ranked second with 3.228 for `Law`, and is ranked first for `Compas` and `Adult` (1.787 and 2.964). Moreover, our proposed method generally achieves the best performance regarding l_1-variance on three datasets. CeFlow also demonstrates the lowest log-density metric in comparison with other approaches achieving at -0.679, -13.62 and -23.46 corresponding to `Law`, `Compas` and `Adult` dataset. This illustrates that the generated samples are more closely followed the distribution of data than other approaches. We furthermore perform a statistical significance test to gain more insights into the effectiveness of our proposed method in producing counterfactual samples compared with other approaches. Particularly, we conduct the paired t-test between our approach (CeFlow) and other methods on each dataset and each metric with the obtained results on 100 randomly repeated experiments and report the result of p-value in Table 1. We discover that our model is statistically significant with $p < 0.05$, proving CeFlow's effectiveness in counterfactual samples generation

tasks. Meanwhile, Table 2 shows the running time of different approaches. Our approach achieves outstanding performance with the running time demonstrating around 90% reduction compared with other approaches. Finally, as expected, by using normalizing flows, CeFlow produces more robust counterfactual samples with the lowest l_1-variance and demonstrates an effective running time in comparison with other approaches.

Figure 2 illustrates the categorical and continuous proximity. In terms of categorical proximity, our approach achieves the second-best performance with lowest variation in comparison with other approaches. The heuristic search based algorithm such as FACE and GS demonstrate the best performance in terms of this metric. Meanwhile, DiCE produces the best performance for continuous proximity, whereas CeFlow is ranked second. In general, our approach (CeFlow) achieves competitive performance in terms of proximity metric and demonstrates the least variation in comparison with others. On the other hand, Figure 3 shows the variation of our method's performance with the different values of α. We observed that the optimal values are achieved at 0.8, 0.9 and 0.3 for `Law`, `Compas` and `Adult` dataset, respectively.

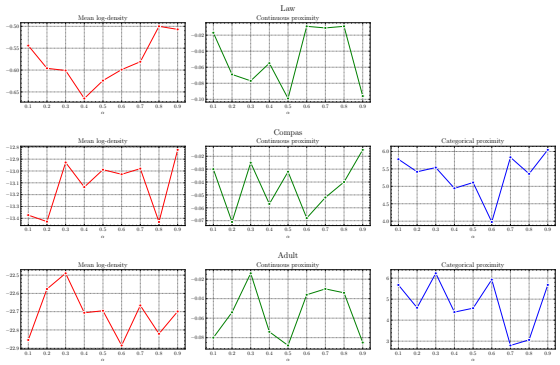

Fig. 3: Our performance under different values of hyperparameter α. Note that there are no categorical features in `Law` dataset.

6 Conclusion

In this paper, we introduced a robust and efficient counterfactual explanation framework called CeFlow that utilizes the capacity of normalizing flows in generating counterfactual samples. We observed that our approach produces more stable counterfactual samples and reduces counterfactual generation time significantly. The better performance witnessed is likely because that normalizing flows can get the exact representation of the input instance and also produce the counterfactual samples by using the inverse function. Numerous extensions

to the current work can be investigated upon successful expansion of normalizing flow models in interpretable machine learning in general and counterfactual explanation in specific. One potential direction is to design a normalizing flow architecture to achieve counterfactual fairness in machine learning models.

Acknowledgement

This work is supported by the Australian Research Council (ARC) under Grant No. DP220103717, LE220100078, LP170100891, DP200101374.

References

1. Artelt, A., Hammer, B.: Convex density constraints for computing plausible counterfactual explanations. arXiv preprint arXiv:2002.04862 (2020)
2. Bliek1ú, C., Bonami, P., Lodi, A.: Solving mixed-integer quadratic programming problems with ibm-cplex: a progress report. In: Proceedings of the twenty-sixth RAMP symposium. pp. 16–17 (2014)
3. Dandl, S., Molnar, C., Binder, M., Bischl, B.: Multi-objective counterfactual explanations. arXiv preprint arXiv:2004.11165 (2020)
4. Dhurandhar, A., Pedapati, T., Balakrishnan, A., Chen, P.Y., Shanmugam, K., Puri, R.: Model agnostic contrastive explanations for structured data. arXiv preprint arXiv:1906.00117 (2019)
5. Dinh, L., Krueger, D., Bengio, Y.: Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516 (2014)
6. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. arXiv preprint arXiv:1605.08803 (2016)
7. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
8. Grath, R.M., Costabello, L., Van, C.L., Sweeney, P., Kamiab, F., Shen, Z., Lecue, F.: Interpretable Credit Application Predictions With Counterfactual Explanations. arXiv:1811.05245 [cs] (Nov 2018), arXiv: 1811.05245
9. Grath, R.M., Costabello, L., Van, C.L., Sweeney, P., Kamiab, F., Shen, Z., Lecue, F.: Interpretable credit application predictions with counterfactual explanations. arXiv preprint arXiv:1811.05245 (2018)
10. Ho, J., Chen, X., Srinivas, A., Duan, Y., Abbeel, P.: Flow++: Improving flow-based generative models with variational dequantization and architecture design. In: International Conference on Machine Learning. pp. 2722–2730. PMLR (2019)
11. Höltgen, B., Schut, L., Brauner, J.M., Gal, Y.: Deduce: Generating counterfactual explanations at scale. In: eXplainable AI approaches for debugging and diagnosis. (2021)
12. Hoogeboom, E., Cohen, T.S., Tomczak, J.M.: Learning discrete distributions by dequantization. arXiv preprint arXiv:2001.11235 (2020)
13. Hvilshøj, F., Iosifidis, A., Assent, I.: Ecinn: efficient counterfactuals from invertible neural networks. arXiv preprint arXiv:2103.13701 (2021)
14. Izmailov, P., Kirichenko, P., Finzi, M., Wilson, A.G.: Semi-supervised learning with normalizing flows. In: International Conference on Machine Learning. pp. 4615–4630. PMLR (2020)

15. Kanamori, K., Takagi, T., Kobayashi, K., Arimura, H.: Dace: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, Christian Bessiere (Ed.). International Joint Conferences on Artificial Intelligence Organization. pp. 2855–2862 (2020)
16. Larson, J., Mattu, S., Kirchner, L., Angwin, J.: How we analyzed the compas recidivism algorithm. ProPublica (5 2016) **9**(1) (2016)
17. Lash, M.T., Lin, Q., Street, N., Robinson, J.G., Ohlmann, J.: Generalized inverse classification. In: Proceedings of the 2017 SIAM International Conference on Data Mining. pp. 162–170. SIAM (2017)
18. Laugel, T., Lesot, M.J., Marsala, C., Renard, X., Detyniecki, M.: Inverse classification for comparison-based interpretability in machine learning. arXiv preprint arXiv:1712.08443 (2017)
19. Laugel, T., Lesot, M.J., Marsala, C., Renard, X., Detyniecki, M.: Comparison-based inverse classification for interpretability in machine learning. In: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. pp. 100–111. Springer (2018)
20. Mahajan, D., Tan, C., Sharma, A.: Preserving causal constraints in counterfactual explanations for machine learning classifiers. arXiv preprint arXiv:1912.03277 (2019)
21. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. pp. 607–617 (2020)
22. Pawelczyk, M., Bielawski, S., Heuvel, J.v.d., Richter, T., Kasneci, G.: Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms. arXiv preprint arXiv:2108.00783 (2021)
23. Pawelczyk, M., Broelemann, K., Kasneci, G.: Learning model-agnostic counterfactual explanations for tabular data. In: Proceedings of The Web Conference 2020. pp. 3126–3132 (2020)
24. Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., De Bie, T., Flach, P.: Face: feasible and actionable counterfactual explanations. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. pp. 344–350 (2020)
25. Sharma, S., Henderson, J., Ghosh, J.: Certifai: A common framework to provide explanations and analyse the fairness and robustness of black-box models. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. pp. 166–172 (2020)
26. Ustun, B., Spangher, A., Liu, Y.: Actionable recourse in linear classification. In: Proceedings of the Conference on Fairness, Accountability, and Transparency. pp. 10–19 (2019)
27. Van Looveren, A., Klaise, J.: Interpretable counterfactual explanations guided by prototypes. arXiv preprint arXiv:1907.02584 (2019)
28. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the gdpr. Harv. JL & Tech. **31**, 841 (2017)
29. Wightman, L.F.: Lsac national longitudinal bar passage study. lsac research report series. (1998)
30. Winkler, C., Worrall, D., Hoogeboom, E., Welling, M.: Learning likelihoods with conditional normalizing flows. arXiv preprint arXiv:1912.00042 (2019)
31. Xu, G., Duong, T.D., Li, Q., Liu, S., Wang, X.: Causality learning: A new perspective for interpretable machine learning. arXiv preprint arXiv:2006.16789 (2020)