Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers

Divyat Mahajan
Microsoft Research
Bangalore, India
divyatmahajan@gmail.com

Chenhao Tan
University of Colorado Boulder
Boulder, USA
chenhao@chenhaot.com

Amit Sharma
Microsoft Research
Bangalore, India
amshar@microsoft.com

Abstract

Explaining the output of a complex machine learning (ML) model often requires approximation using a simpler model. To construct interpretable explanations that are also consistent with the original ML model, counterfactual examples — showing how the model's output changes with small perturbations to the input — have been proposed. This paper extends the work in counterfactual explanations by addressing the challenge of the feasibility of such examples. For explanations of ML models in critical domains such as healthcare, finance, etc, counterfactual examples are useful for an end-user only to the extent that perturbation of feature inputs is feasible in the real world. We formulate the problem of feasibility as preserving causal relationships among input features and present a method that uses (partial) structural causal models to generate actionable counterfactuals. When feasibility constraints may not be easily expressed, we propose an alternative method that optimizes for feasibility as people interact with its output and provide oracle-like feedback. Our experiments on MNIST, Bayesian networks and the widely used "Adult" dataset show that our proposed methods can generate counterfactual explanations that satisfy feasibility constraints better than previous approaches. Code repository can be accessed here: *https://github.com/divyat09/cf-feasibility*

1 Introduction

Local explanations for a machine learning model are important for people to interpret its output, especially in critical decision-making scenarios such as healthcare, governance, and finance. Techniques for explaining an ML model often involve a simpler surrogate model that yields interpretable information, such as feature importance scores [18]. However, these techniques suffer from an inherent fidelity-interpretability tradeoff due to their use of a simpler model for generating explanations. Highly interpretable explanations may end up approximating too much and be inconsistent with the original ML model (low fidelity), while high fidelity explanations may be as complex as the original ML model and thus less interpretable.

Counterfactual explanations [23] have been proposed as an alternative that are always consistent with the original ML model and arguably may also be interpretable. Counterfactual (CF) explanations present the perturbations in the original input features that could have led to a change in the prediction of the model. For example, consider a person whose loan application has been rejected by an ML

CausalML: Machine Learning and Causal Inference for Improved Decision Making Workshop, 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

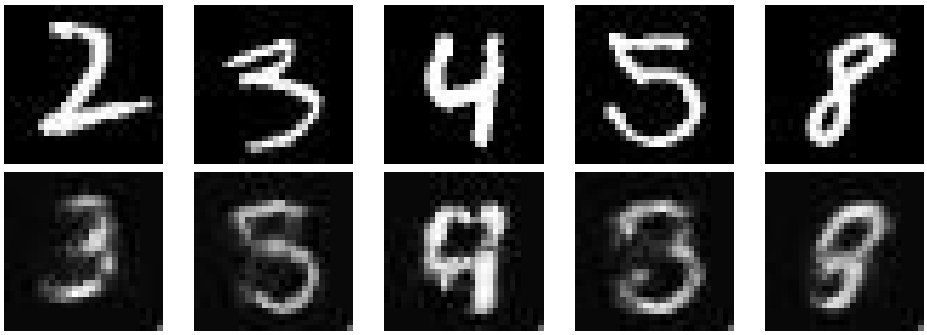


Figure 1: CF examples generated from MNIST images. In each case, a target class (*digit*) was provided to the BaseGenCF method.

classifier. For this person, a CF explanation provides what-if scenarios wherein they would have their loan approved, e.g., *"your loan would have been approved if your income was $10000 more"*. Since the goal is to generate perturbations of an input that lead to a different outcome from the ML model, CF explanation has parallels with adversarial examples [5].

However, one of the biggest challenges with counterfactual explanation is to generate examples that are *feasible* in the real world. Continuing with the loan example, the counterfactual changes in the input should follow some natural laws (e.g., age cannot decrease) and knowledge about interactions between features (e.g., changing education level without changing age is impossible). When explanations are provided specifically to people, a related property is whether they are *actionable* by the end-user [14]. Recent work tries to address feasibility and actionability through adding constraints during generation of CFs. However, these are simply statistical constraints that fail to capture full requirements for feasibility.

Our main contribution is to show that feasibility is fundamentally a causal concept, and cannot be addressed with statistical constraints alone. We formally define feasibility for a CF example based on an underlying structural causal model between input features: a CF example is feasible if the changes satisfy constraints entailed by the causal model. To bring out the difference with statistical constraints, consider a CF example that recommends "increase education level to Masters" without changing the age of a person. Such a CF example is infeasible but will satisfy constraints from past work, including change in only actionable features [21], and being likely given the observed data [4] (it may be likely to observe others with the same age and a Masters degree). Similarly a CF example that recommends "decrease age by 3 years" satisfies the constraint from [17] that intermediate CFs on the path (e.g., people with same features but age reduced by 1 and 2 years) are likely, yet is impossible to act upon. We call this *global* feasibility that must be satisfied by all counterfactual examples, and propose a *causal proximity* regularizer that can be added to any CF generation method. In addition, we also define *local* feasibility that depends on an end-user's preferences.

Our second contribution is to present a generative model for outputting counterfactuals, that provides a substantial speed-up in generating CFs for multiple inputs. We propose a generative model `BaseGenCF` similar to a variational auto-encoder [8] that can be used to directly generate CF examples. Figure 1 shows CF examples generated from MNIST digits. Compared to past work on CF generation that requires solving an optimization problem for every input, a generative model provides significant computational advantage in generating multiple counterfactuals for different inputs.

To model feasibility, we propose two extensions of this model. In practice, it is unlikely that accurate information on causal relationships will be available as a structural causal model. Therefore, we propose a method that uses constraints derived from available causal knowledge to modify the BaseGenCF method. Specifically, we provide an extension to optimize for common unary and binary feasibility constraints. In general, however, optimizing for many complex constraints can be difficult, and in the case of user preferences, no such hard constraints exist. We thus provide a second method that learns from user feedback ("feasibility oracle") and iteratively learns to generate feasible CF examples.

Results on Adult-Income, MNIST, and synthetic Bayesian network datasets show that our proposed methods can generate counterfactual examples that are more feasible than models that do not include

causal assumptions or user feedback. Further, our novel generative model is much faster than existing approaches to generate CFs. To summarize, our contributions include:

- We propose a generative model for CFs that performs better than past approaches on interpretability metrics and is computationally faster.
- To address feasibility, we propose a causal proximity regularizer based on a structural causal model of the data.
- We present two practical methods based on the causal regularizer and show how they can satisfy constraints monotonicity in feature changes.

2 Defining Feasibility of CF Explanations

Throughout, we assume a machine learning classifier, $h : \mathcal{X} \rightarrow \mathcal{Y}$ where $\boldsymbol{x} \in \mathcal{X}$ are the features and $y \in \mathcal{Y}$ is a categorical output. A *valid* counterfactual example for an input $\boldsymbol{x}$ and outcome y is one that changes the outcome of h to the desired outcome y' [14]. Counterfactual generation is usually framed as solving an optimization problem that searches in the feature space to find perturbations that are proximal (close to the original input) but lead to a different output class from the machine learning model. [23] provide the following optimization to generate a CF example $\boldsymbol{x}^{cf}$ for an input instance $\boldsymbol{x}$ given a ML model f, where the target class is y':

$$\underset{\boldsymbol{x}^{cf}}{\operatorname{argmin}} \operatorname{Loss}(f(\boldsymbol{x}^{cf}), y') + \operatorname{Dist}(\boldsymbol{x}, \boldsymbol{x}^{cf}). \tag{1}$$

Loss refers to a classification loss (such as cross-entropy) and Dist refers to a distance metric (such as ℓ_2 distance). That is, we seek to generate counterfactual explanations that belong to a target class y' while still remaining proximal to the original input.

Note that under this formulation, there are two limitations: features of the input $\boldsymbol{x}$ may be changed independently to construct $\boldsymbol{x}^{cf}$, and a new optimization problem needs to be solved for each new input. For the first, we propose a definition for incorporating feasibility below. For the second, Section 3 provides a generative model that once trained, can easily generate multiple new CFs.

2.1 Global Feasibility

Definition 2.1. ***Causal Model [16].*** *A causal model is a triplet $M = \langle U, V, F \rangle$ such that U is a set of exogenous variables, V is a set of endogenous variables that are determined by variables inside the model, and F is a set of functions that determine the value of each $v_i \in V$ (up to some independent noise) based on values of $U_i \bigcup Pa_i$ where $U_i \subseteq U$ and $Pa_i \subseteq V \setminus v_i$.*

Definition 2.2. ***Global Feasibility.*** *Let $\langle \boldsymbol{x}_i, y_i \rangle$ be the input features and the predicted outcome from h, and let y' be the desired output class. Let $M = \langle U, V, F \rangle$ be a causal model over $\mathcal{X}$ such that each feature is in $U \bigcup V$. Then, a counterfactual example $\langle \boldsymbol{x}_{\texttt{cf}}, y_{\texttt{cf}} \rangle$ is globally feasible if it is valid ($y_{\texttt{cf}} = y'$), the change from $\boldsymbol{x}_i$ to $\boldsymbol{x}_{cf}$ satisfies all constraints entailed by the causal model, and all exogenous variables $\boldsymbol{x}^{exog} = U$ lie within the input domain.*

For example, a CF example that changes an individual's age to 300 is infeasible since it violates the limits of the input domain of the age feature. In general, such constraints relating to the input domain can be learned from an i.i.d. sample of data by estimating the joint distribution of features. E.g., [4] use an auto-encoder loss-term to align CF examples to the data distribution.

In addition, however, a CF example that decreases age is infeasible since it violates the natural causal model/constraint that age can only increase with time. Such *causal constraints cannot be learned from data alone, and often need extra information* [16]. More complex causal constraints can be defined over pairs or multiple variables. For example, in the loan decision example from above, we can consider v_{p1} as *education-level* and v as *age*, and posit a causal relationship that increasing *education-level* needs years to complete and thus causes *age* to increase. Thus, any counterfactual example that increases *education-level* without increasing *age* is infeasible. That said, a counterfactual example that increases *age* without changing *education-level* may still be feasible since we do not know the full set of causes that may increase *age*. While some of these feasibility constraints can be formulated in simple terms, causal relationships over multiple features can lead to complex constraints. As we show below, they can be defined formally using structural causal models, and implemented as constraints on how features can change (Section 3.2).

Using a causal model to measure global feasibility. We now define a feasibility-compatible notion of distance to utilize the causal knowledge about features and thus constrain independent perturbations over features. We want the counterfactual to be proximal to the data sample not only based on the Euclidean distance between them, but also based on the causal relationships between features.

Suppose we are provided with the structural causal model [16] for the observed data, including the causal graph G over $U \bigcup V$ and the functional relationships between variables. V is the set of all endogenous nodes that have at least one parent in the graph. Now for each node $v \in V$, we know the generating mechanism of v conditioned on its parents, i.e., $v = f(v_{p1}, .., v_{pk}) + \epsilon$ where ϵ denotes independent random noise. We can use the causal knowledge to define a feasibility-compatible notion of $Distance$ for the each node $v \in V$:

$$\text{DistCausal}_{\text{v}}(\boldsymbol{x}_v, \boldsymbol{x}_v^{cf}) = \text{Dist}_{\text{v}}(\boldsymbol{x}_v^{cf}, f(\boldsymbol{x}_{v_{p1}}^{cf}, .., \boldsymbol{x}_{v_{pk}}^{cf})) \tag{2}$$

This distance metric indicates that the perturbation for the feature v should be related to the perturbations in its parents via the mapping f associated with the conditional distribution of the feature v, which implies that the proximity of a CF not just dependent on the original feature value $\boldsymbol{x}_v$. This addresses the problem of independent perturbations in different components of a feature and generates perturbations that follow the underlying causal distribution. That is, a counterfactual is preferable if it is close to the input instance and also preserves the associated change distribution due to the causal structure over its features. For exogenous features U, we still use the ℓ_1/ℓ_2 distance. Hence, the Dist term in the CF generation loss function (Eq. 1) can be modified to generate counterfactuals that preserve causal constraints, where U are the exogenous nodes (i.e., nodes without any parents in the causal graph) and V are the remaining features.

$$\text{DistCausal}(\boldsymbol{x}, \boldsymbol{x}^{cf}) = \sum_{u \in U} \text{Dist}_{\text{u}}(\boldsymbol{x}_u^{cf}, \boldsymbol{x}_u) + \sum_{v \in V} \text{DistCausal}_{\text{v}}(\boldsymbol{x}_v, \boldsymbol{x}_v^{cf}) \tag{3}$$

Since knowing the full causal graph is often impractical, the above approach can also work whenever we have partial knowledge of the causal structure (e.g., some edges in the causal graph). From this partial causal knowledge, we construct a set of nodes V for which we know the generating mechanism for each node $v \in V$ conditioned on its parents and consider the rest of the variables in U.

2.2 Local Feasibility

That said, for a particular user, a globally feasible CF example may still be infeasible due to an end-user's context or personal preferences. Thus, while global feasibility is a necessary condition for feasible CFs, we also need to define local feasibility.

Definition 2.3. ***Local Feasibility****: A CF example is locally feasible for a user if it is globally feasible and satisfies user-level constraints.*

For example, a user may find it difficult to change their city because of family constraints. Thus, a counterfactual example may be locally infeasible due to many user specific factors. Preserving constraints entailed from a causal model provides necessary conditions for a feasible counterfactual, but customization is needed for local feasibility. In Section 3.3, we discuss how an oracle-based method can personalize the feasibility for each person based on their feedback.

3 Generating Feasible CF Explanations

Below we provide a generative model for generating counterfactuals and provide a variational lower bound that can be used to derive a loss function. Being a generative model, it avoids separate, new optimizations for each input and can also be extended to handle causal constraints.

3.1 Base CF Explanation Objective

We define the problem of generating CF examples $\boldsymbol{x}^{cf}$ as building a model that maximizes $\Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x})$ such that $\boldsymbol{x}^{cf}$ belongs to class y'. Our approach is based on an encoder-decoder framework where the task of the encoder is to project input features to a suitable latent space and the task of the decoder is to generate a counterfactual from the latent representation given by the encoder. Analogous to a variational auto-encoder (VAE) [8], we first arrive at a latent representation $\boldsymbol{z}$ for the

input instance $\boldsymbol{x}$ via the encoder $q(\boldsymbol{z}|\boldsymbol{x}, y')$ and then generate the corresponding counterfactual $\boldsymbol{x}^{cf}$ via the decoder $p(\boldsymbol{x}^{cf}|\boldsymbol{z}, y')$. Following the construction in VAEs [8], we first derive the evidence lower bound (ELBO) for generating CF explanations.

Theorem 1. *The evidence lower bound to optimize CF objective* $\Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x})$ *for global feasibility is:*

$$\ln \Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x}) \geq \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) - \mathbb{KL}(Q(\boldsymbol{z}|\boldsymbol{x}, y')||P(\boldsymbol{z}|y', \boldsymbol{x})$$

The proof is in the **Suppl. A.** The prior of the latent variable $\boldsymbol{z}$ is modulated by y' and $\boldsymbol{x}$, but following [20], we simply use $p(\boldsymbol{z}|y', \boldsymbol{x}) \sim \mathcal{N}(\mu_{y'}, \sigma^2_{y'})$, so the KL Divergence can be computed in closed form. $P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x})$ represents the probability of the output $\boldsymbol{x}^{cf}$ given the desired class and latent variable $\boldsymbol{z}$. This can be empirically estimated by the ℓ_1/ℓ_2 loss or any general $\mathrm{Distance}$ metric between input $\boldsymbol{x}$ and $\boldsymbol{x}^{cf}$. That is, without additional assumptions, we are assuming that probability $P(\boldsymbol{x}^{cf})$ is highest near $\boldsymbol{x}$. In addition, this probability expression is conditioned on y', implying that $\boldsymbol{x}^{cf}$ is valid only if belongs to y' class when applied with h. We thus use a classification loss (e.g., hinge-loss) between $h(\boldsymbol{x}^{cf})$ and y', where y' represents the target class and β represents the margin:

$$-\mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) \approx \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')}[\mathrm{Dist}(\boldsymbol{x}, \boldsymbol{x}^{cf}) + \lambda \,\mathrm{HingeLoss}(h(\boldsymbol{x}^{cf}), y', \beta)] \quad (4)$$

where λ is a hyperparameter. To summarize, given the ML model h to be explained, we learn our proposed model by minimizing the following loss function (**BaseGenCFLoss**):

$$\mathop{\mathbb{E}}_{Q(\boldsymbol{z}|\boldsymbol{x},y')}[\mathrm{Dist}(\boldsymbol{x}, \boldsymbol{x}^{cf}) + \lambda \,\mathrm{HingeLoss}(h(\boldsymbol{x}^{cf}), y', \beta)] + KL(Q(\boldsymbol{z}|\boldsymbol{x}, y')||P(\boldsymbol{z}|y', \boldsymbol{x})) \quad (5)$$

where y' is the target counterfactual class. Our loss formulation bears an intuitive resemblance with the standard counterfactual loss formulation (Eq. 1). $\mathrm{HingeLoss}(f(\boldsymbol{x}^{cf}), y', \beta)$ helps us to generate valid counterfactuals with respect to the ML model h, and $\mathrm{Distance}(\boldsymbol{x}, \boldsymbol{x}^{cf})$ helps us to generate counterfactuals that are close to the input feature. The additional third term in the loss function represents the KL divergence between the prior distribution $p(\boldsymbol{z}|y')$ and the latent space encoder $q(\boldsymbol{z}|\boldsymbol{x}, y')$, analogous to the loss term in a VAE [8]. Our encoder-decoder framework can be viewed as an adaptation of VAE for the task of generating counterfactuals. The Hinge Loss function is defined as:

$$HingeLoss(h(\boldsymbol{x}^{cf}), y', \beta) = max\{[\max_{y!=y'}\{h(\boldsymbol{x}^{cf})_y\} - h(\boldsymbol{x}^{cf})_{y'}], -\beta\}$$

The above Hinge Loss formulation encourages classifier's score on target class to be higher than any other class by at least a margin of β. Typically, the Dist function can be defined as the ℓ_1 distance between the input $\boldsymbol{x}$ and the counterfactual $\boldsymbol{x}^{cf}$: $\mathrm{Dist}(\boldsymbol{x}, \boldsymbol{x}^{cf}) = \|\boldsymbol{x} - \boldsymbol{x}^{cf}\|_1$. We refer to this method as **BaseGenCF**. When the true causal model is known, we can replace Dist function by $\mathrm{DistCausal}$ (Eq 3) . We call the resultant method **SCMGenCF**.

The major advantage of our approach is that we end up learning a generative model $p(\boldsymbol{x}^{cf}|\boldsymbol{z}, y')$ for the counterfactuals given the input x. It makes our approach computationally attractive for generating counterfactuals for a series of inputs as compared to past works [4, 14] that solve an optimization problem for each input separately. We can directly sample any number of counterfactuals from the generative model $p(\boldsymbol{x}^{cf}|\boldsymbol{z}, y')$.

Finally it is also possible to add a separate auto-encoder term that models the data distribution, as proposed by [4]. That is, we first train a (variational) auto encoder on the training dataset and use it to penalise counterfactuals that don't obey the training distribution:

$$\mathrm{BaseGenCFLoss}(\boldsymbol{x}) + \mathop{\mathbb{E}}_{Q(\boldsymbol{z}|\boldsymbol{x},y')}[\lambda_{ae} * \|Dist(\boldsymbol{x}^{cf}, AE(\boldsymbol{x}^{cf}))\|] \quad (6)$$

where AE represents the pre-trained Auto Encoder and λ_{ae} is a hyperparameter. We consider this method as a baseline **AEGenCF** and compare it to BaseGenCF in Section 4.

3.2 Approximating Feasibility Constraints

When the exact functional causal mechanism for a variable is unknown, we present an approximation of the `SCMGenCF` method that uses knowledge of certain constraints based on domain knowledge. For example, one may require that Age of a person cannot decrease, or that Education-level shares a monotonic causal relationship with Age, without knowing the true functional form. Here we propose a simple extension to `BaseGenCF` to add constraints for encoding unary and binary constraints:

Unary constraints We consider unary constraints that stipulate whether a feature can increase or decrease. This is added to the `BaseGenCFLoss` (Eq 5) as Hinge Loss on the feature of interest. As an example, for the case that a feature can only increase, the Hinge Loss would be as follows:

$$-\min(0, \boldsymbol{x}_v^{cf} - \boldsymbol{x}_v)$$

Binary constraints Binary constraints capture the nature of causal relationship between two features. One of the most common are monotonic constraints, which we approximate by learning an appropriate linear model for each binary constraint. Let x_1 and x_2 be two features where x_1 causes x_2 and we have a monotonically increasing trend between them. We capture this monotonic trend by learning a linear model between x_1 and x_2, under the constraint that the parameter that relates x_1 to x_2 should be positive (or negative depending on the nature of monotonicity) . This can be learnt by minimizing the following loss function over training data:

$$(\boldsymbol{x}_{v_2} - \alpha - \beta \boldsymbol{x}_{v_1}) - \min(0, \beta)$$

where α and β are parameters that can be learned from training data. After we have learnt the linear model, we can proceed in the exact same fashion as we did for `SCMGenCF`, as now we know the approximate distribution f between the features. We call this **ModelApproxGenCF**.

3.3 Oracle Based Approach

The `ModelApproxGenCF` approach is limited since it might difficult to approximate complex constraints directly. Further, in the case of local feasibility, there may be no explicit constraints that users can easily provide. However, a user may be able to provide yes/no feedback to the generated CF examples. In this section, we thus consider learning feasibility constraints from oracle feedback on generated CFs. Specifically, we assume blackbox access to an Oracle that provides a feasibility score for each (input, counterfactual) pair. We assume that the Oracle exposes implicit user preferences or causal knowledge through a black-box interface. Similar ideas of capturing knowledge implicitly using oracles can be found in [7].

Let us represent the Oracle O's scoring mechanism as follows: Given any input pair $(\boldsymbol{x}, \boldsymbol{x}^{cf})$ the oracle outputs 1 if the CF example is (locally) feasible, otherwise it outputs 0. Hence, in order to generate feasible counterfactuals, our task is to maximize the Oracle score. Consider g as any general mechanism to generate counterfactuals over a dataset, i.e., $\forall \boldsymbol{x} \in \mathcal{X}, \boldsymbol{x}_{cf} = g(\boldsymbol{x})$, then the task to generate feasible counterfactuals is: $\max_g \sum_{\boldsymbol{x} \in \mathcal{X}} O(\boldsymbol{x}, g(\boldsymbol{x}))$.

The oracle can be interpreted as an expert who guides a method towards generating feasible counterfactuals. However, we only have black-box access to the Oracle and there would usually be a high cost to query the oracle, especially where the oracle represents human judgment. It is thus desirable to maximize the oracle score on generated counterfactuals using as few oracle queries as possible.

Our BaseGenCF approach provides us with a convenient way to learn the oracle's knowledge using a model (o) that generates a score for a counterfactual:

$$o(\boldsymbol{x}, \boldsymbol{x}^{cf}) = \mathop{\mathbb{E}}_{Q(\boldsymbol{z}|\boldsymbol{x}, y')} [\exp\left(-(\boldsymbol{x}^{cf} - \mu)^T (\boldsymbol{x}^{cf} - \mu)\right)]$$

where μ represents the output of the counterfactual generator/decoder $p(\mu|\boldsymbol{z}, y')$ in our framework, with $\boldsymbol{x}$ as the input to the encoder $q(z|\boldsymbol{x}, y')$. The key task is to train the `BaseGenCF` framework such that the decoder becomes a better approximation of the oracle: we expect $o(x, x^{cf})$ to be higher whenever $O(x, x^{cf}) = 1$, and lower when $O(x, x^{cf}) = 0$.

3.3.1 Proposed Algorithm

We now provide the complete training algorithm, called **OracleGenCF**. It has two training phases:

1. **Base Training Phase**: Since we do not have any access to $(\boldsymbol{x}, \boldsymbol{x}^{cf})$ query pairs apriori, we train our `BaseGenCF` and sample counterfactuals q^{cf} from the decoder $p(\boldsymbol{x}^{cf}|\boldsymbol{z}, y')$ to create queries for the oracle O. We refer to this set of queries as Q: $\{(x, q^{cf})\}$.
2. **Oracle Modeling Phase**: We fine-tune the `BaseGenCF` generative framework by ensuring that the score $o(\boldsymbol{x}, \boldsymbol{x}^{cf})$ is a better approximation for Oracle O. We also add the BaseGenCF loss (Eq. 5) to ensure the counterfactuals we generate are still valid and proximal. This ensures that we do not solely focus on the task of learning a good model of the Oracle, because that would only ensure feasibility but may violate proximity and validity. We also add a hyperparameter λ_o that controls the tradeoff between modeling valid counterfactuals and modeling the oracle. The exact loss function used for fine-tuning the framework is:

$$min \sum_{i \in Q} [\text{BaseGenCFLoss}(\boldsymbol{x}_i) + \lambda_o * \|O(\boldsymbol{x}_i, q_i^{cf}) - o(\boldsymbol{x}_i, q_i^{cf}; \mu)\|] \tag{7}$$

3.3.2 Capturing Global & Local feasibility

`OracleGenCF` can be used to capture any general global feasibility constraint. Consider an example of feature v and its causes $(v_1, \ldots, v_p)$, with a positive Individual Treatment Effect (ITE) of each cause on the feature v. This can be captured implicitly by a simple Oracle O.

$$O(\boldsymbol{x}, \boldsymbol{x}^{cf}) = \begin{cases} 1, & \text{if } (\ \forall \text{i } \{\boldsymbol{x}_{v_{pi}}^{cf} > \boldsymbol{x}_{v_{pi}}\} \implies \boldsymbol{x}_v^{cf} > \boldsymbol{x}_v) \text{ or} \\ & (\ \forall \text{i } \{\boldsymbol{x}_{v_{pi}}^{cf} < \boldsymbol{x}_{v_{pi}}\} \implies \boldsymbol{x}_v^{cf} < \boldsymbol{x}_v) \\ 0, & \text{otherwise} \end{cases}$$

Additionally, the method can be used to capture personalized user-specific constraints with a user as the oracle, thus representing human perception. Different users could be modeled using different oracles. The oracle can give labelled data for (multiple) user-specific constraints at once by simply stating a counterfactual as feasible ($O(\boldsymbol{x}, \boldsymbol{x}^{cf}) = 1$) or infeasible ($O(\boldsymbol{x}, \boldsymbol{x}^{cf}) = 0$).

4 Empirical Evaluation

We first evaluate `BaseGenCF` method against previous methods for generating CFs for the MNIST dataset, using interpretability metrics proposed by [22]. We then evaluate its extensions for feasibility, `ModelApproxGenCF` and `OracleGenCF` on `Adult` dataset and simulated Bayesian network datasets.

4.1 Evaluating BaseGenCF against past approaches

We consider the MNIST dataset [10] which contains 70,000 labeled 28x28 images of handwritten digits between 0 and 9. We train a neural network model on the dataset to predict the digit classes. For explaining this model, we consider the counterfactual generation task on 5 digits (2, 3, 4, 5, 8); with the respective target counterfactual classes (3, 5, 9, 3, 9) as shown in Figure 1. We take a subset of 100 samples for each of the 5 digits for training the `BaseGenCF` method. Details on the model architecture for the ML model and `BaseGenCF` are in the Suppl. **B.1**

For evaluation, we use two metrics defined by [22]:`IM1` metric which measures the ratio between the reconstruction of x_{cf} using AutoEncoders trained on target class and original class data:

$IM1 = \frac{\|\boldsymbol{x}^{cf} - AE_t(\boldsymbol{x}^{cf})\|}{\|\boldsymbol{x}^{cf} - AE_o(\boldsymbol{x}^{cf})\|}$

`IM2` metric, that measures the difference between the reconstruction of x_{cf} using AutoEncoders trained on the target class and all classes:

$IM2 = \frac{\|AE(\boldsymbol{x}^{cf}) - AE_t(\boldsymbol{x}^{cf})\|}{\|\boldsymbol{x}^{cf}\|}$

To ensure a fair comparison; we used the same ML model and AutoEncoder architecture as [22] for computing the Interpretability Metrics (Table 1). The results reported for "Ours" approach are the mean and standard deviation over 10 runs.

`BaseGenCF` achieves a better IM1 and IM2 score than other approaches, even though it does not explicitly have an autoencoder term in its loss, unlike (B, C, D, E, F) approaches that use the AE or Prototype loss. These results are probably since our `BaseGenCF` generative model, being trained on

the full train data, is able to capture the feature distribution better. That said, it is harder to interpret IM2 metric; we provide more details in Suppl. **B.2**.

In addition, the time taken to generate CFs by BaseGenCF is order of magnitude lower than other approaches. That said, we do have a fixed training time (**172.81** $\pm$ **3.97** seconds); thus our approach will be efficient for deployments where it can be trained once and used for generating multiple CFs for different inputs.

Method	Time (s)	Gradient Steps	IM1	IM2(*10)
A: V	13.06 ± 0.23	5158 ± 82	1.56 ± 0.03	1.65 ± 0.04
B: VA	8.40 ± 0.38	2380 ± 113	1.36 ± 0.02	1.60 ± 0.03
C: VP	2.37 ± 0.09	751 ± 31	1.23 ± 0.02	1.46 ± 0.03
D: VAP	2.05 ± 0.08	498 ± 27	1.26 ± 0.02	1.29 ± 0.03
E: P	4.39 ± 0.04	1794 ± 12	1.20 ± 0.02	1.52 ± 0.03
F: PA	2.86 ± 0.06	773 ± 16	1.22 ± 0.02	1.29 ± 0.03
Ours	$\mathbf{0.033 \pm 0.001}$	**600 (Training)**	$\mathbf{1.07 \pm 0.07}$	$\mathbf{0.12 \pm 0.03}$

Table 1: Results for the MNIST dataset. Metrics for other approaches are from Table 1 in [22].

4.2 Evaluating on feasibility constraints

As we have mentioned before, a counterfactual could be infeasible due to many reasons, but for our evaluation, we assume that there is a constraint that completely captures the feasibility of a counterfactual. To design this constraint, we either infer it from the causal model (simulated datasets) or from domain knowledge (real-world dataset).

4.2.1 Datasets.

Simple-BN. We consider a toy dataset of 10,000 samples with three features $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ and one outcome variable (y). The causal relationships between them are modeled as:

$$p(x_1) \sim N(\mu_1, \sigma_1); \qquad p(\boldsymbol{x}_2) \sim N(\mu_2, \sigma_2)$$
$$p(x_3|x_1, x_2) \sim N(k_1 * (x_1 + x_2)^2 + b_1, \sigma_3); k_1 > 0, b_1 > 0;$$
$$p(y|x_1, x_2, x_3) \sim Bernoulli(\sigma(k_2 * (x_1 * x_2) + b_2 - x_3)); k_2 > 0, b_2 > 0$$

Note that we require x_1 and x_2 to be positive, so they follow a truncated normal distribution. The intuition is that $\boldsymbol{x}_3$ is determined by a monotonically increasing function of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. At the same time, y is positively affected by an increase in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ but negatively affected by $\boldsymbol{x}_3$. Thus, a naive counterfactual method may not satisfy the monotonic constraint on $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and $\boldsymbol{x}_3$. Specifically, the global monotonicity constraint is defined as:

" ($\boldsymbol{x}_1, \boldsymbol{x}_2$ increase $\implies$ $\boldsymbol{x}_3$ increases) AND ($\boldsymbol{x}_1, \boldsymbol{x}_2$ decrease $\implies$ $\boldsymbol{x}_3$ decrease)"

We report results for a hand-crafted set of parameters such that there is tradeoff between proximity and monotonic feasibility constraint: $\mu_1 = 50$, $\mu_2 = 50$, $\sigma_1 = 15$ $\sigma_2 = 17$, $\sigma_3 = 0.5$, $k_1 = 0.0003$, $k_2 = 0.0013$, $b_1 = 10$, $b_2 = 10$.

Sangiovese [13]. This is a conditional linear Bayesian network on the effects of different agronomic settings on quality of Sangiovese grapes [2]. It has 14 features and a categorical output for quality, with a sample size of 10,000. The true causal model is known. The features are all continuous except Treatment which has 16 levels. For simplicity, we remove the categorical variable Treatment since it leads to 16 different linear functions. For feasibility, we test a monotonic constraint over two variables, $BunchN$ and $SproutN$. Specifically, the global monotonicity constraint is defined as:

" ($SproutN$ increase $\implies$ $BunchN$ increases) AND ($SproutN$ decrease $\implies$ $BunchN$ decrease)"

Adult [9]. We consider a real-world dataset, `Adult`. The outcome y is binary $y = 0$ (Low Income), and $y = 1$ (High Income). Since we do not have a causal model, we design two constraints that capture feasibility using domain knowledge:

C1: $\boldsymbol{x}^{cf}_{Age} \geq \boldsymbol{x}_{Age}$;

C2: $(\boldsymbol{x}_{Ed}^{cf} > \boldsymbol{x}_{Ed} \implies \boldsymbol{x}_{Age}^{cf} > \boldsymbol{x}_{Age})$ AND $(\boldsymbol{x}_{Ed}^{cf} = \boldsymbol{x}_{Ed} \implies \boldsymbol{x}_{Age}^{cf} \geq \boldsymbol{x}_{Age})$

`C1` represents a unary constraint that Age cannot decrease in CF explanations. `C2` represents a monotonic constraint that increase in Educational level should increase Age, and if Educational level remains the same, age should not decrease. `C2` also includes an additional constraint that Education level cannot decrease. Hence, if Education level decreases then its an infeasible counterfactual, regardless of the change in Age (since in practice Education level does not usually decrease). To make the CF generation task more challenging, we sample data points with the Age feature greater than 35 and outcome class $y = 0$ and data points with the value of feature Age less than 45 and the outcome class $y = 1$. This creates a setup in which higher age data points are more correlated with the low income class group and vice-versa. We obtain a dataset of size 15691 and consider the task of generating CFs with the target class as $y = 1$.

4.2.2 Setup

For all experiments, the ML classifier h is implemented as a neural network with two hidden layers. Continuous features are scaled to (0-1) range and categorical features are represented as one-hot encoded vectors. Each proposed method for counterfactual generation is trained using a 80-10-10% training, validation and test dataset respectively. For the `OracleGenCF` method, we additionally generate the query set Q using 10% of the training dataset with 10 counterfactuals per data point. Details regarding the ML model and (encoder, decoder) in BaseGenCF, and hyperparameter tuning for all methods are provided in Suppl. **C.1**.

Baselines: We report results for all proposed methods (`BaseGenCF, AEGenCF, ModelApproxCF, SCMGenCF` and `OracleGenCF`), except for the `Adult` where we cannot apply `SCMGenCF` due to lack of causal knowledge about the dataset. For comparison, we evaluate `OracleGenCF` on global constraints, though it can also handle local constraints. Suppl. **C.1** describes the loss terms used for modeling constraints in `ModelApproxGenCF` for different datasets. Throughout, we also compare these methods to the Contrastive Explanations method (CEM) proposed by [4].

Evaluation Metrics: We define the following metrics to evaluate CF explanations; considering the case of N data points $\{x_i\}$, with K CF's $\{x_{i,j}^{cf}\}$ sampled for each data point.

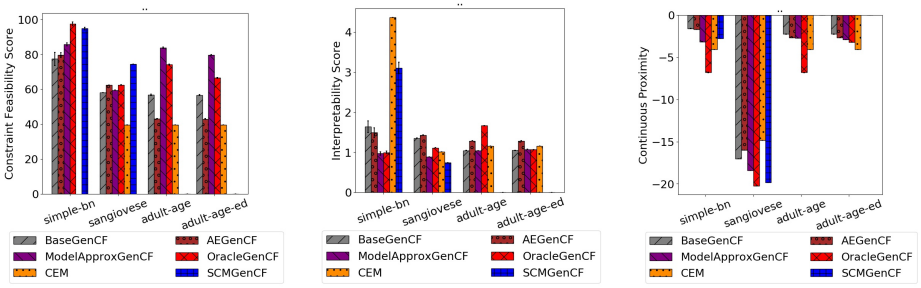
- `Target-Class Validity`: % of CFs whose predicted class by the ML classifier is the same as the target class; $\frac{\sum_{i=1}^{N}\sum_{j=1}^{K}\mathbb{1}[f(x_{i,j}^{cf})==t_c]}{N*K}$
- `Cont-Proximity`: Proximity for continuous features as the average ℓ_1-distance between $\boldsymbol{x}^{cf}$ and $\boldsymbol{x}$ in units of median absolute deviation for each features [14]; It is multiplied by (-1) so that higher values are better. $\frac{\sum_{i=1}^{N}\sum_{j=1}^{K}\sum_{p=1}^{d_{cont}}\frac{x_{i,j}^{cf,p}-x_i^p}{MAD_p}}{N*K*d_{cont}}$
- `Cat-Proximity`: Proximity for categorical features as the total number of mismatches on categorical value between $\boldsymbol{x}^{cf}$ and $\boldsymbol{x}$ for each feature [14]; It is multiplied by (-1) so that higher values are better. $\frac{\sum_{i=1}^{N}\sum_{j=1}^{K}\sum_{p=1}^{d_{cat}}\mathbb{1}[x_{i,j}^{cf,p}\neq x_i^p]}{N*K*d_{cat}}$
- `Constraint Feasibility Score`: For Simple-BN and Sangiovese datasets, the constraints mentioned can be observed as a combination of two sub constraints: X1 and X2. For example, in the case of Simple-BN dataset, X1 corresponds to "($\boldsymbol{x}_1, \boldsymbol{x}_2$ increase $\implies$ $\boldsymbol{x}_3$ increases)"; while X2 correspond to "($\boldsymbol{x}_1, \boldsymbol{x}_2$ decrease $\implies$ $\boldsymbol{x}_3$ decrease)"

  Hence, to ensure good performance at satisfying both the sub-constraints, we define the following metric for constraint feasibility: $\frac{2*S1*S2}{S1+S2}$

  where S1, S2 represent the % of CFs satisfying the sub constraints X1, X2 respectively.

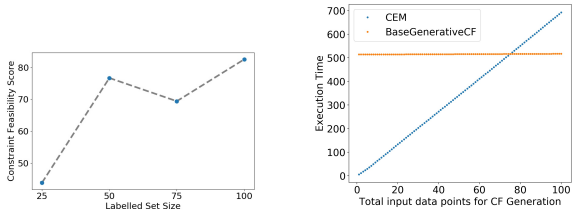
  However in the case of Adult dataset, due to the additional non monotonic constraint of Education level cannot decrease, we simply report the percentage of Counterfactual satisfying the complete constraint `C2` on the Adult dataset.

  For unary constraints, like `C1` on the Adult dataset, we always report the percentage of Counterfactuals satisfying the constraint.
- `Interpretability Score`: The IM1 metric as defined by [22], the ratio of the reconstruction loss of CFs given the target class Auto Encoder and the reconstruction loss given the original class Auto Encoder : $\frac{\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\|\boldsymbol{x}^{cf}-AE_t(\boldsymbol{x}^{cf})\|}{\|\boldsymbol{x}^{cf}-AE_o(\boldsymbol{x}^{cf})\|}}{N*K}$



(a) (b) (c)

Figure 2: Constraint-Feasibility, Interpretability score (lower is better) and Cont-Proximity metrics for the three datasets.



(a) (b)

Figure 3: (a): Constraint-Feasibility as the no. of labelled examples is increased for global constraints in `Adult`. (Other metrics are in Suppl. **C.2**). (b): Time taken to generate CF examples.

- `Causal-Edge Score`: Ratio of the Log Likelihood of CFs x^{cf} and the Log Likelihood of the original data point x w.r.t. to the set of given causal edges distribution V : $\frac{\sum_{i=1}^{N}\sum_{j=1}^{K}\sum_{v\in V}\frac{p(x_v^{cf}|x_{vp1}^{cf},..,x_{vpk}^{cf})}{p(x_v|x_{vp1},..,x_{vpk})}}{N*K}$

Causal-Edge-Score is defined only for `SimpleBN` and `Sangiovese` where the true causal model is known.

4.2.3 Results

Evaluating feasibility. Figure 2 shows `Constraint-Feasibility Score`, `Interpretability Score` and `Cont-Proximity` for different methods, averaged over 10 runs (other metrics are in the Suppl. **C.2**). For Simple-BN dataset, `OracleGenCF` achieves the highest Constraint Feasibility score, `SCMGenCF` achieve the highest Constraint Feasibility score on the Sangiovese dataset, while the `ModelApproxGenCF` achieves the highest Constraint Feasibility score on the Adult dataset.

All the methods achieve perfect score on the `Target-Class Validity` (refer Suppl. **C.2**, Figure 4 (c)). However, across the three datasets, the methods designed to preserve feasibility (`SCMGenCF`, `OracleGenCF` and `ModelApproxGenCF`) perform better than the methods `BaseGenCF`, `AEGenCF` and `CEM` on the `Constraint -Feasibility Score`. `CEM` performs the lowest on Constraint Feasibility score across all datasets, it achieves a score of zero on simple-bn dataset and around 40 percent on the Sangiovese and Adult dataset.

The poor performance of `CEM` on Constraint Feasibility across datasets suggests that feasibility cannot be solely captured by relying on the observed data likelihood. Also, `AEGenCF` leads to worse Constraint Feasibility score than `BaseGenCF` on the Adult dataset, empirically validating our claim that generating CFs that follow observed data distribution [4, 22] do not guarantee feasibility. In the Adult dataset, the feasibility constraint requires Age to be increased, despite a correlation between low Age and High Income in the dataset, illustrating the fact that following observed distribution does not ensure feasibility.

That said, increasing feasibility induces a tradeoff with the continuous proximity. Higher feasibility scores for `OracleGenCF`, `ModelApproxGenCF` and `SCMGenCF` lead to a decrease in `Cont-Proximity` as compared to `BaseGenCF`. Thus, depending on the dataset and underlying causal model, achieving high fraction of feasibile CF examples requires considering a tradeoff with proximity.

BaseGenCF and AEGenCF have similar IM1 score. Figure 2 (b) shows the Interpretability score (IM1 metric). The `BaseGenCF` approach can consistently achieve good interpretability score across datasets as compared to the approaches `AEGenCF`, `CEM` which explicitly optimize for counterfactuals following the observed data distribution. This shows that `BaseGenCF` does not end up learning a distribution too far from that of observed data distribution.

OracleGenCF: Constraint Feasibility increases with no. of labelled CFs. A key question for the Oracle-based method is the number of labelled CF examples it needs. Using the Adult dataset and the non-decreasing Age constraint, we show the `Constraint-Feasibility Score` of `OracleGenCF` as we increase the number of labelled CF examples (Figure 3 (a)). For the global constraint, we find that the Feasibility Score increases with labelled inputs, reaching nearly 80% with 100 labels.

BaseGenCF provides big speed-up compared to CEM. Besides feasibility, `BaseGenCF` is computationally faster than past methods like `CEM`, since it uses a generative model. In Figure 3 (b), we show the time required to generate counterfactual examples for k inputs for the Adult dataset and find that `BaseGenCF` takes less time per CF example as k increases. `CEM` has a similar execution time for each input while `BaseGenCF` takes time for initial training but then negligible time for every new input.

5 Related Work

Our work builds upon the literature on explainable ML [18, 12] by focusing on a specific type of explanation through counterfactual examples [23]. Most CF generation methods rely on separate optimizations for each input, based on original features [19, 14], a latent representation [6], or using a generative adversarial network [11].

To account for feasibility, different statistical notions of feasibility have been proposed, based on adherence to the training distribution [4], distribution of the target class [22], the likelihood of the intermediate points in reaching a CF example [17], and on specifying which features can be changed [21]. [11] rely on GAN's to restrict the explanations in semantically meaningful space. In a critical commentary, [3] raise concern that feasibility cannot be learned only from training data distribution. Related to our framework on expressing feasibility in terms of causal constraints, [14] point to the importance of causal constraints for feasibility, but do not provide a method for generating feasible CF examples. Our paper extends this line of work by formally defining feasibility, providing a theoretical justification of the counterfactual loss and proposing a VAE-based generative model that can preserve causal constraints. In a related line of work, Parafita et al. [15] and [6] also focus on the role of causality towards feasible counterfactual explanations but their approach requires full knowledge of the causal model.

6 Conclusion

Feasibility in CF explanations is hard to quantify due to multiple reasons that can render a counterfactual infeasible. In this work, we provided a generative model and two methods for modeling causal constraints. In future work, we will explore how to learn causal constraints from available data.

References

[1] Algorithms for monitoring and explaining machine learning models. `https://docs.seldon.io/projects/alibi` alibi.

[2] Bayesian network repository. `http://www.bnlearn.com/bnrepository/` sangiovese.

[3] Solon Barocas, Andrew D Selbst, and Manish Raghavan. The hidden assumptions behind counterfactual explanations and principal reasons. *arXiv preprint arXiv:1912.04930*, 2019.

[4] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, pages 592–603, 2018.

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

[7] Theofanis Karaletsos, Serge Belongie, and Gunnar Rätsch. Bayesian representation learning with oracle constraints. *arXiv preprint arXiv:1506.05011*, 2015.

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[9] Ronny Kohavi and Barry Becker. Uci machine learning repository. *https://archive.ics.uci.edu/ml/datasets/adult*, 1996.

[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] Shusen Liu, Bhavya Kailkhura, Donald Loveland, and Yong Han. Generative counterfactual introspection for explainable deep learning. *arXiv preprint arXiv:1907.03077*, 2019.

[12] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[13] Alessandro Magrini, Stefano Di Blasi, and Federico Mattia Stefanini. A conditional linear gaussian network to assess the impact of several agronomic settings on the quality of tuscan sangiovese grapes. *Biometrical Letters*, 2017.

[14] Ramaravind Kommiya Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the ACM FAT* conference (to appear)*, 2020.

[15] Álvaro Parafita and Jordi Vitrià. Explaining visual models by causal attribution. *arXiv preprint arXiv:1909.08891*, 2019.

[16] Judea Pearl. *Causality*. Cambridge University Press, 2009.

[17] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Face: Feasible and actionable counterfactual explanations. *arXiv preprint arXiv:1909.09369*, 2019.

[18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 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*, pages 1135–1144. ACM, 2016.

[19] Chris Russell. Efficient search for diverse coherent explanations. In *Proceedings of FAT**, 2019.

[20] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[21] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 10–19, 2019.

[22] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

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

A Supplementary Materials: Theorem 1 Proof

Theorem 2. *The evidence lower bound to optimize CF objective* $\Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x})$ *for global feasibility is:*

$$\begin{aligned}\ln \Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x}) \geq \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) \\ -\mathbb{KL}(Q(\boldsymbol{z}|\boldsymbol{x}, y')||P(\boldsymbol{z}|y', \boldsymbol{x}))\end{aligned} \tag{8}$$

Proof. An ideal counterfactual generation model approximates $\boldsymbol{x}$ (proximity) and generates $\boldsymbol{x}^{cf}$ that are valid w.r.t desired class y'. Thus for a model we seek to maximize $P(\boldsymbol{x}^{cf}|y', \boldsymbol{x})$ where P is the underlying probability distribution over $\mathcal{X}$.

$$\begin{aligned}&\ln P(\boldsymbol{x}^{cf}|y', \boldsymbol{x}) \\ &= \ln \int P(\boldsymbol{x}^{cf}, \boldsymbol{z}|y', \boldsymbol{x}) d\boldsymbol{z} \\ &= \ln \int Q(\boldsymbol{z}|\boldsymbol{x}, y') \frac{P(\boldsymbol{x}^{cf}, \boldsymbol{z}|y', \boldsymbol{x})}{Q(\boldsymbol{z}|\boldsymbol{x}, y')} dz \\ &\geq \int Q(\boldsymbol{z}|\boldsymbol{x}, y') \ln \frac{P(\boldsymbol{x}^{cf}, \boldsymbol{z}|y', \boldsymbol{x})}{Q(\boldsymbol{z}|\boldsymbol{x}, y')} dz \\ &= \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln \frac{P(\boldsymbol{x}^{cf}, \boldsymbol{z}|y', \boldsymbol{x})}{Q(\boldsymbol{z}|\boldsymbol{x}, y')} \\ &= \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) - \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln \frac{Q(\boldsymbol{z}|\boldsymbol{x}, y')}{P(\boldsymbol{z}|y', \boldsymbol{x})}\end{aligned} \tag{9}$$

Where the inequality above is due to Jensen's inequality. Using the definition of KL-Divergence,

$$\begin{aligned}\ln P(\boldsymbol{x}^{cf}|y', \boldsymbol{x}) \geq \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) \\ - \mathbb{KL}(Q(\boldsymbol{z}|\boldsymbol{x}, y')||P(\boldsymbol{z}|y', \boldsymbol{x}))\end{aligned}$$

□

B Supplementary Materials: MNIST

B.1 Implementation Details

Th ML Model architecture and the BaseGenCF architecture was kept the same as [22]. The details can be seen in this notebook by [22]. We describe the implementation details of the ML model and BaseGenCF below.

B.1.1 ML Model Architecture

Architecture of the ML model to be explained is now described. A slight difference we had to introduce from the architecture of [22] was to make $kernel\text{-}size$ as 3 (instead of 2) and $padding$ as 1 to ensure the spatial dimensions of the image are the same as them after applying Conv Layer. The model was trained for 50 epochs, batch size 32, learning rate 10^{-4} with Adam optimizer and Cross Entropy Loss, with a 80-10-10% training, validation and test dataset respectively. It achieved test accuracy of 96%.

- Conv Layer(out-channels=32, kernel-size=3, stride=1, padding=1), ReLU
- MaxPool(pool-size=2), Dropout(0.3)
- Conv Layer(out-channels=64, kernel-size=3, stride=1, padding=1), ReLU
- MaxPool(pool-size=2), Dropout(0.3)
- Hidden Layer 1(256), Dropout(0.5), ReLU
- Hidden Layer 2(10), Softmax

B.1.2 Auto Encoder Architecture

The training strategy used was exactly the same as mentioned in the notebook by [22].

Architecture of the Encoder used for computing the IM1, IM2 Metrics:

- Conv Layer(out-channel=16, kernel-size=3, stride=1, padding=0), ReLU
- Conv Layer(out-channel=16, kernel-size=3, stride=1, padding=0), ReLU
- MaxPool(pool-size=2)
- Conv Layer(out-channel=1, kernel-size=3, stride=1, padding=0), ReLU

Architecture of the Decoder used for computing the IM1, IM2 Metrics:

- Conv Layer(out-channel=16, kernel-size=3, stride=1, padding=0), ReLU
- UpSample((2,2))
- Conv Layer(out-channel=16, kernel-size=3, stride=1, padding=0), ReLU
- Conv Layer(out-channel=1, kernel-size=3, stride=1, padding=0)

B.1.3 BaseGenCF Architecture

The BaseGenCF Encoder Decoder framework as trained for 25 epochs, batch size 16, learning rate 10^{-4} with SGD optimizer and `BaseGenCFLoss` (Section 3.1), with a 80-10-10% training, validation and test dataset.

Encoder Architecture:

Architecture of Encoder consists of two networks: one is used to estimate the mean and the other is used to estimate the variance of the posterior distribution $q(z|x, y_k)$. Both the networks for estimating mean and variance have the same architecture as described below, with the only difference that the variance network having an additional Sigmoid activation at the end to ensure the variance is positive.

The network consists of sub models: the output from the sub model M1 is concatenated with the target class of the counterfactual and then fed into the sub model M2. We denote the size of the input after sub model M1 as *convoluted-size* and the latent embedding space dimension as *embedding-size*. We used *embedding-size* as 10 and the *convoluted-size* can be computed using the M1 architecture below to be $22 * 22$

Sub Model M1:

- Conv Layer(out-channel=16, kernel-size=3, stride=1, padding=0), ReLU
- Conv Layer(out-channel=16, kernel-size=3, stride=1, padding=0), ReLU
- Conv Layer(out-channel=1, kernel-size=3, stride=1, padding=0), ReLU

Sub Model M2:

- Hidden Layer(convoluted-size+1, embedding-size), BatchNorm
- Sigmoid (In case of variance network only)

Decoder Architecture:

- Hidden Layer(embedding-size+1, convoluted-size), BatchNorm, ReLU
- Hidden Layer(convoluted-size, 2*convoluted-size), BatchNorm, ReLU
- Hidden Layer(2*convoluted-size, 28*28), BatchNorm, Sigmoid

B.2 Additional Results

Table 2 breaks down the performance of the BaseGenCF method on the `IM1` and `IM2` metric by anaylzing the performance on its different components. IM1 Numerator corresponds to the reconstruction loss with the target class Auto Encoder: $\|\boldsymbol{x}^{cf} - AE_t(\boldsymbol{x}^{cf})\|$; while the IM1 Denominator corresponds to the reconstruction loss with the original class Auto Encoder: $\|\boldsymbol{x}^{cf} - AE_o(\boldsymbol{x}^{cf})\|$.

IM2 Numerator corresponds to the difference between the reconstruction loss using AutoEncoders trained on the target class and all classes: $\|AE(\boldsymbol{x}^{cf}) - AE_t(\boldsymbol{x}^{cf})\|$; while the the IM2 Denominator corresponds to the norm of the counterfactual: $\|\boldsymbol{x}^{cf}\|$. Note that any method's performance on the IM2 metric would depend a lot on the norm of counterfactuals generated (IM2 Denominator), which may not be desirable. This limits a proper interpretation of the IM2 Metric as proposed by [22].

To enable a better comparison, here we report the performance of our method BaseGenCF on both components of IM2 metric, We do not have results of [22] on the different components of IM1 and IM2 metric, hence it is unclear why we obtain substantially better results on IM2 Metic (Section 4.1). It may be due to less reconstruction error difference (IM2 Numerator) or due to higher norm counterfactuals (IM2 Denominator) generated by our method.

Metric	**Performance (mean $\pm$ std)**
IM1 Numerator	3.37 ± 0.21
IM1 Denominator	3.32 ± 0.24
IM2 Numerator	1.02 ± 0.09
IM2 Denominator	107.04 ± 0.46

Table 2: Additional results for the MNIST dataset for the BaseGenCF approach

C Supplementary Materials: Bayesian Networks and Adult

C.1 Implementation Details

C.1.1 ML Model Architecture

Architecture of the ML model to be explained as mentioned below. The model for all the datasets was trained for 100 epochs, batch size 32, learning rate 10^{-3} with Adam optimizer and Cross Entropy Loss, with a 80-10-10% training, validation and test datasets. It achieved test accuracy of 87% on Simple-BN, 83.5% on Sangiovese, 89.3% on Adult dataset. The values of *hidden-dim* is chosen as 10 and value of *num-classes* is 2 as its a binary classification task across the three datasets.

The ML models comprises of two layers as shown below:

- Hidden Layer(data-size, hidden-dim)
- Hidden Layer(hidden-dim, num-classes)

C.1.2 BaseGenCF Architecture

Here we provide the implementation details of the base variational encoder decoder used in all our different methods. Both the encoder and decoder are modeled as Neural Networks (NN) with multiple hidden layers and non linear activations. Encoder comprises of two Neural Networks: one NN is used to estimate the mean and other NN is used to estimate the variance of posterior distribution $q(z|x, y_k)$. Both the networks for estimating mean and variance have the same architecture as described below, with the only difference that the variance network having an additional Sigmoid activation at the end to ensure the variance is positive. Similarly, decoder comprises of a neural network to estimate the counterfactual from the latent encoding and the target class.

The latent space dimension is set to 10 for all the different methods and datasets. Both the encoder and the decoder are conditioned on the target counterfactual class. Hence, the `Hidden Layer 1` in the Encoder takes the data point concatenated with the target class as the input. Similarly, the `Hidden Layer 1` in the Decoder takes the latent sample concatenated with the target class as the input.

Across datasets, it was trained to minimise `BaseGenCFLoss` (or its variations depending on the specific method) with SGD optimizer, learning rate 10^{-3} for 50 epochs. The batch size for the different datasets are; Simple-BN : 64, Sangiovese : 512, Adult : 2048.

Encoder Architecture:

- Hidden Layer 1(DataSize+1, 20), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 2(20, 16), BatchNorm, Dropout(0.1), ReLU

- Hidden Layer 3(16, 14), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 4(14, 12), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 5(12, LatentDim)
- Sigmoid (In case of variance network only)

Decoder Architecture:

- Hidden Layer 1(LatentDim+1, 12), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 2(12, 14), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 3(14, 16), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 4(16, 20), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 5(20, DataSize), Sigmoid

C.1.3 Auto Encoder Architecture

The architecture and the training strategy for the Auto Encoder used in the method `AEGenCF` and for computing the `Intepretability Score` are exactly the same as the ones mentioned above for *BaseGenCF Architecture* (Section C.1.2). The only difference being that the Encoder and Decoder are not conditioned on the target class.

C.1.4 Contrastive Explanantions

For experiments involving the Contrastive Explanations (`CEM`) method [4], we used the implementation provided by the open source library ALIBI [1]. Since the choice of auto encoder is not specified in ALIBI, we trained our own auto encoder with the same architecture as defined above in *Auto Encoder Architecture* section.

C.1.5 HyperParameter tuning

For a fair comparison between methods, we optimize hyperparameters using the validation set and use random search for 100 iterations. Since an ideal counterfactual needs to satisfy feasibility, target-class validity, and proximity, we select the hyperparameters that lead to maximum feasibility, while still obtaining more than 90% target class validity and proximity at least τ. τ was conservatively selected to remove models that result in much lower proximity than the BaseGenerativeCF method. In practice, all methods achieved near 100% target-class validity.

The following threshold values of τ were used depending on the dataset:

- BN1: 9.0 (Cont. Proximity)
- Sangiovese: 20.0 (Cont. Proximity)
- Adult: 8.0 (Cont. Proximity) and 5.0 (Cat. Proximity)

We did not include the other metrics like Interpretability Score and Causal Edge score during the hyperparameter tuning. These metrics allow an independent evaluation of the proposed methods.

The final optimal values of the hyper parameters for each dataset are reported in the table 3. As per the `BaseGenCFLoss` equation (section 3.1), we have the two hyperparameters β and λ common across all the approaches; which we refer to as `Margin` and `Validity` in the Table 3.

Additionally, there is an extra hyperparameter involved with approaches like AEGenCF, SCMGenCF, ModelApproxGenCF and OracleGenCF. We denote this extra hyperparameter for approaches besides BaseGenCF, as `Feasibility` in the Table 3.

For AEGenCF, it corresponds to λ_{ae} which controls the trade-off between modelling CF and the AutoEncoder (Eq 6). For OracleGenCF, it corresponds to λ_o which controls the trade-off between modeling CF and the oracle (Eq 7).

In the case of SCMGenCF, it corresponds to the trade-off between Exogenous and Endogenous Loss terms in `DistCausal` loss term, as described below:

$$\text{DistCausal}(\boldsymbol{x}, \boldsymbol{x}^{cf}) = \sum_{u \in U} \text{Dist}_{\text{u}}(\boldsymbol{x}_u^{cf}, \boldsymbol{x}_u) + \lambda_s * \sum_{v \in V} \text{DistCausal}_{\text{v}}(\boldsymbol{x}_v, \boldsymbol{x}_v^{cf})$$

Dataset	Method	Margin	Validity	Feasibility
MNIST	BaseGenCF	0.05	100	NA
Simple-BN	BaseGenCF	0.014	54	NA
	AEGenCF	0.324	72	24
	ModelApproxGenCF	0.087	96	0.1
	OracleGenCF	0.15	150	2350
	SCMGenCF	0.015	85	55
Sangiovese	BaseGenCF	0.161	94	NA
	AEGenCF	0.12	44	84
	ModelApproxGenCF	0.306	71	73
	OracleGenCF	0.02	25	1085
	SCMGenCF	0.319	89	77
Adult-Age	BaseGenCF	0.165	94	NA
	AEGenCF	0.369	73	2
	ModelApproxGenCF	0.764	29	192
	OracleGenCF	0.084	159	5999
Adult-Age-Ed	BaseGenCF	0.165	94	NA
	AEGenCF	0.369	73	2
	ModelApproxGenCF	0.344	76	87
	OracleGenCF	0.117	175	3807

Table 3: Hyperparameter tuning decription for all the methods and datasets

In the case of ModelApproxGenCF, for Unary constraints, it corresponds to the trade-off between `BaseGenCFLoss` (Eq 5) and the Hinge Loss on feature of interest. For the Binary constraints, it would follow the same procedure as for SCMGenCF.

For the Contrastive Explanations (CEM) method, the optimal hyperparameters for each dataset are as follows: (refer to open source library ALIBI [1] for details regarding each hyper parameter)

- Simple BN: Beta (0.608), Kappa (0.021), Gamma (8.0), CSteps (3), Max Iterations (1000)
- Sangiovese: Beta (0.652), Kappa (0.041), Gamma (9.0), CSteps (5), Max Iterations (1000)
- Adult: Beta (0.911), Kappa (0.241), Gamma (0.0), CSteps (9), Max Iterations (1000)

C.1.6 Constraint Modelling in ModelApproxGenCF

For the case of Simple-BN and Sangiovese dataset, the feasibility constraint is monotonic, hence we use the *Binary Constraints* formulation of `ModelApproxGenCF`, as described in the Section: 3.2. For the case of Adult Dataset, the constraint C1 is modelled using the *Unary Constraints* formulation, with a Hinge Loss on the feature Age.

The constraint C2 in Adult Dataset is more complex than the previous constraints, since the feature Education is categorical. We model the constraint C2 under the *Unary Constraints* formulation, since it can be viewed as combinations of two unary constraints: Age cannot decrease and Education cannot decrease. The Hinge Loss on categorical variable Education is implemented by converting the embedding of categorical variable Education into a continuous value. We rank different education levels with increasing score and take a weighted sum of the categorical embedding with the scores assigned for each education category. Hence, we get a continuous score for education feature embedding which is representative of the level/rank of education. Now, we can apply the same Hinge Loss on the continuous values of education feature to put penalty on counterfactuals that decrease the level of education.

The vector we used for ranking different educations levels is as follows:

- HS-Grad, School: 0
- Bachelors, Assoc, Some-college: 1
- Masters:2
- Prof-school, Doctorate: 3

C.2 Additional Results

Interpretability Score: Figure 4 (a) and (b), breaks down the performance of the methods on the `Interpretability Score` by analyzing the performance on its different component. Interpretability Score Numerator corresponds to the reconstruction loss with the target class Auto Encoder: $\|\boldsymbol{x}^{cf} - AE_t(\boldsymbol{x}^{cf})\|$; while the Denominator corresponds to the reconstruction loss with the original class Auto Encoder: $\|\boldsymbol{x}^{cf} - AE_o(\boldsymbol{x}^{cf})\|$.

Like the result on the complete Interpretability Score (Figure 2 (b) in the main submission), the results here again suggest that `BaseGenCF` performs as good as `AEGenCF` and better than `CEM` across datasets on Interpretability Score Numerator and Denominator, even though `CEM` and `AEGenCF` are encouraged explicitly to generate counterfactuals closer to observed data distribution.

Feasibility preserving approaches like `ModelApproxGenCF` and `OracleGenCF` do not perform as good as `BaseGenCF` consistently across datasets. This suggests a trade-off between feasibility and interpretability in the current methods.

For the `SCM` method, we find a substantial difference in the reported results based on the overall Interpretability score versus its numerator and denominator components. In the case of Simple-BN dataset, `SCM` generates counterfactuals that achieve low reconstruction loss w.r.t both the target class Auto Encoder (Figure 4 (a)) and the original Auto Encoder (Figure 4 (b)) as compared to other methods. However, it achieves a worse score on the overall `Interpretability Score` (Figure 2 (b)). A likely explanation is that the overall metric computes the average of ratios, whereas Figures 4 (a,b) show the averages of the numerator and denominator respectively. In future work, we will investigate whether the ratio of averages (rather than average of ratios) may be a better, more robust metric.

Target-Class Validity: Figure 4 (c) shows the performance of the methods on `Target-Class Validity`. All the methods achieve near perfect score on this metric across datasets.

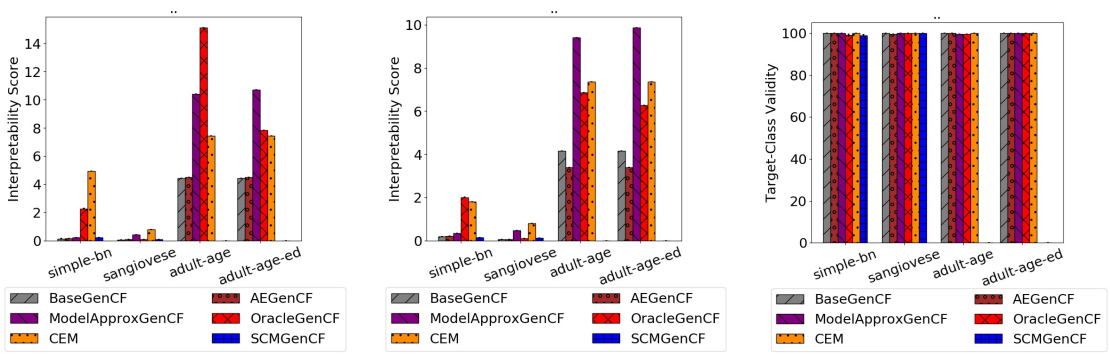
Causal Edge Score: Figure 5 (a) and (b), shows the methods evaluated on the Causal Edge Score metric for the Simple-BN and the Sangiovese dataset. We cannot evaluate methods on the `Adult` dataset under this metric due to lack of causal knowledge about the dataset.

`BaseGenCF` performs as good as `AEGenCF` on both the datasets, even though `AEGenCF` explicitly utilizes the likelihood of observed data to guide CF generation. `CEM` does not achieve a high Causal Edge score either, performing the worst on both datasets.

Among the feasibility preserving approaches, `SCMGenCF` performs good consistently on both the datasets, while `ModelApproxGenCF` and `OracleGenCF` show poor performance for the Simple BN dataset (Figure 5 (a)) despite being better than `BaseGenCF` at the Constraint-Feasibility Score (Figure 2 (a)). This suggests future improvement towards designing a method that preserves both the feasibility and the underlying causal edge distribution.

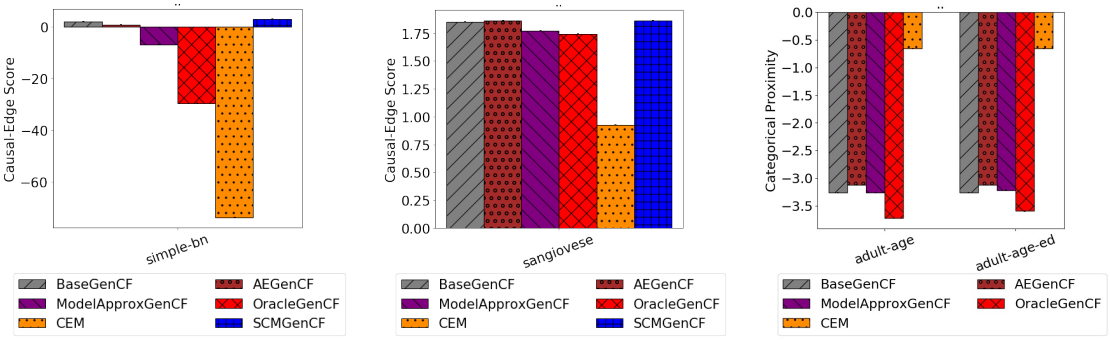
Categorical Proximity: Figure 5 (c) shows the methods evaluated on the Cat-Proximity metric. The dataset `Simple-BN` and `Sangiovese` do not contain any categorical variables, hence we do not include them for this analysis. `CEM` performs better than other methods on Categorical proximity in both the cases of age (C1) and age-ed (C2) constraint. This may be because CEM tends to not vary categorical features. Table 4 provides an example via generated CF examples for the Adult dataset: `CEM` does not change categorical features like Occupation, MaritalStat, Race unlike `ModelApproxGenCF` and `OracleGenCF`. Along with Figure 2 (c), this result demonstrates the tradeoff between feasibility and proximity. `CEM` is worse at preserving constraints (e.g., in Table 4, CF examples by CEM do not increase the value of Age while increasing the Education level to Masters), but achieves higher proximity score for categorical variables.

OracleGenCF Performance Analysis Figure (6) shows the results on Target-Class Validity, Continuous Proximity and Categorical Proximity for the analysis of Oracle performance at preserving global constraints (non decreasing Age constraint) in the `Adult` dataset with increasing labelled CF examples. While we saw substantial variation in the `Constraint-Feasibility` Metric (Figure 3 (a)), we find minor variations on the target-class validity and proximity metrics, as the number of labelled examples increases from 25 to 100.



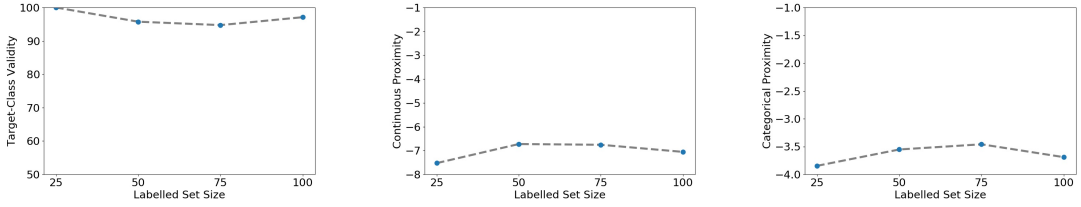
(a) (b) (c)

Figure 4: Interpretability Score Numerator, Interpretability Score Denominator, Target-Class Validity for all datasets



(a) (b) (c)

Figure 5: Causal-Edge Score BN1, Causal-Edge Score Sangiovese and Categorical Proximity metrics across the datasets



(a) (b) (c)

Figure 6: Target-Class Validity, Continuous Proximity and Categorical Proximity as the number of labelled examples is increased for global constraints in the Adult dataset

Adult	Method	AgeYrs	Education	Occupation	WorkClass	Race	HrsWk	MaritalStat	Sex
Original input (outcome: <=50K)		41	Some-college	Blue-collar	Private	Other	40	Single	Male
Counterfactuals (outcome: >50K)	**ModelApproxGenCF**	46	Masters	White-Collar	Private	White	41	Married	Male
		45	Some-college	White-Collar	Private	White	40	Married	Male
		47	Masters	White-Collar	Private	White	41	Married	Male
Counterfactuals (outcome: >50K)	**OracleGenCF**	45	Prof-school	White-Collar	Private	White	42	Married	Male
		44	Masters	White-Collar	Private	White	42	Married	Male
		47	Prof-school	White-Collar	Private	White	43	Married	Male
Counterfactuals (outcome: >50K)	**CEM**	41	Masters	Blue-Collar	Private	Other	39	Single	Male
		41	Some-college	Blue-Collar	Other	Other	39	Single	Male
		40	Masters	Blue-Collar	Private	Other	41	Single	Male

Table 4: Examples of generated counterfactuals on the modified Adult dataset. OracleGenCF and ModelApproxGenCF were trained to preserve the Education-Age causal constraint