Algorithm 1: Message passing GNN with L layers decomposed in three operations

Input: A directed graph $\mathcal{G}$ with d nodes, adjacency matrix $\mathbf{A}$ and node features $\mathbf{X}$.
Output: $\mathbf{H} = \{h_i^L\}_{i=1}^d$.

$h_i^0 = x_i \; \forall i$
for $l = 1, \dots, L$; `// For each layer` l
do
  for $i = 1, \dots, d$; `// For node` i
  do
    $m_{ij}^l = f^m(h_i^{l-1}, h_j^{l-1}; \theta_m^l) \; \forall j \in \mathcal{N}_i$; `// Compute the messages`
    $M_i^l = f^a(\{m_{ij}^l \mid j \in \mathcal{N}_i\})$; `// Compute the aggregated message`
    $h_i^l = f^u(h_i^{l-1}, M_i^l; \theta_u^l)$; `// Compute the node features at layer` l
  end
end

A Background details on message passing Graph Neural Networks

Let us refer as *feature* a vector with dimension F that belongs to $\mathbb{R}^F$. A *directed graph* with d nodes can be represented as $\mathcal{G} := (\mathbf{X}, \mathbf{E})$, where $\mathbf{X} \in \mathbb{R}^{d \times F_X}$ denotes the features of the nodes (the row index identifies the node, i.e., the i-th row contains the F_X-dimensional features of the node i) and $\mathbf{E} \subseteq \{(i,j) \mid i \in [d], j \in [d]\}$ denotes the set of directed edges in the graph from j to i. The adjacency matrix $\mathbf{A} \in \{0,1\}^{d \times d}$ of $\mathcal{G}$ is defined as $A_{ij} = 1$ if there is an edge from j to i and $A_{ij} = 0$ otherwise. Then, a directed graph can be alternatively represented as $\mathcal{G} = (\mathbf{X}, \mathbf{A})$. Given a graph $\mathcal{G}$, a *Graph Neural Network (GNN) with parameters* θ is a function $f_\theta : \mathbb{R}^{d \times F_X} \times \{0,1\}^{d \times d} \to \mathbb{R}^{d \times F_H}$ that takes into account the graph structure contained in the adjacency matrix $\mathbf{A} \in \{0,1\}^{d \times d}$ and transforms the node features $\mathbf{X}$ into different features $\mathbf{H} \in \mathbb{R}^{d \times F_H}$, i.e., $\mathbf{H} = f(\mathbf{X}, \mathbf{A}; \theta)$ (for readability we consider $f_\theta(\cdot) \equiv f(\cdot; \theta)$). Importantly, at the output of the GNN we have a graph $(\mathbf{H}, \mathbf{E})$ that preserves the structure of the input graph $(\mathbf{X}, \mathbf{E})$.

A GNN based on message passing [12] is a type of spatial convolution GNNs [55] in which information is passed following the message passing process: In each layer l of the GNN each node i receives information from its neighbors $\mathcal{N}_i$, a.k.a. the parents of i. In a message passing GNN, the feature vector of the i-th node at the output of a layer l—i.e., h_i^l—is computed in three steps:

1. `Message.` The *message from node j to node i* is defined as $m_{ij}^l = f^m(h_i^{l-1}, h_j^{l-1}; \theta_m^l)$, where h_i^{l-1} are the features of node i at layer $l-1$, h_j^{l-1} are the features of node j at the previous layer $l-1$, and f^m is a neural network (usually a linear layer) parametrized by θ_m^l.

2. `Aggregator.` The *aggregator* is a function in charge of combining all the incoming messages at each node i into a single message, a.k.a. the aggregated message $M_i^l = f^a(\{m_{ij}^l \mid j \in \mathcal{N}_i\})$. Notice that f^a does not have any parameters. Some choices of f^a are the mean, standard deviation, max or min over the inputs, i.e., messages [8].

3. `Update.` The *update function* $h_i^l = f^u(h_i^{l-1}, M_i^l; \theta_u^l)$ takes the aggregated message and the representation of node i at layer $l-1$ and outputs the new representation for node i at layer l. The function f^u is defined as a neural network (usually a linear layer) with parameters θ_u^l.

Putting the three steps together, we obtain the general form of a message passing based GNN layer as $h_i^l = f^u\left(h_i^{l-1}, f^a\left(\{f^m(h_i^{l-1}, h_j^{l-1}; \theta_m^l) \mid j \in \mathcal{N}_i\}\right); \theta_u^l\right)$. Algorithm 1 describes the propagation of information (i.e., messages) in a GNN with L layers.

B Proofs

For the sake of completeness, in this section we first formalize the meaning of causal factorization, interventions and the abduction step in VCAUSE.

VCAUSE causal factorization refers to the factorization of the joint distribution as

$$p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \prod_i p_{\theta_i}(X_i; \boldsymbol{\eta}_i),$$

where the likelihood parameters $\boldsymbol{\eta}_i = \boldsymbol{\eta}_i(\mathbf{Z}_{an^*(i)})$ are a function of all (and only) the features of the ancestors of i and the features of i.

A VCAUSE intervention is performed by removing all the edges towards the intervened node i, such that $\mathcal{N}_i = \varnothing$, while the rest of the edges remains untouched.

In a *VCAUSE abduction step*, the posterior distribution factorizes as

$$q_\phi(\mathbf{Z} \mid \mathbf{X}) = \prod_i q_{\phi_i}(Z_i; \boldsymbol{\eta}_i^{enc}),$$

where the distribution parameters $\boldsymbol{\eta}_i^{enc} = \boldsymbol{\eta}_i^{enc}(\mathbf{X}_{pa^*(i)})$ are a function of all (and only) the features of node i and the features of its the parents.

Notation. Consider a *causal graph* $\mathcal{G} := (\mathbf{X}, \mathbf{E})$, which is a directed acyclic graph (DAG). Let us define a path of length n from node u to node v in $\mathcal{G}$ as $p(u, v) = (u, w_1, w_2, \ldots, w_{n-1}, v)$, which is an ordered sequence of unique nodes such that i) there exists an edge in $\mathcal{G}$ between concurrent nodes, ii) the first node is u, iii) and the last node is v. We refer to the length of the path as $|p(u, v)|$, i.e., the number of edges in the path, or alternatively, the number of nodes minus one. Let us define $P(u, v)$ as the set of unique paths connecting u to v. Let us define the *shortest path from u to v* as $p^-(u, v)$ (i.e. the path with the minimum number of edges to go from u to v) and its length as $d^-(u, v) = |p^-(u, v)|$. Let us define the *longest path* from u to v as $p^+(u, v)$ (i.e. the path with the maximum number of edges to go from u to v) and its length as $d^+(u, v) = |p^+(u, v)|$. Let us define the *set of ancestors of node i* (i.e., $an(i)$) as the set of nodes with paths to i, i.e., $\{j \mid |P(j, i)| > 0\}$. As for a GNN, we define the number of hidden layers (total number of layers minus one) as N_h.

Then, we define the *diameter* δ of the graph $\mathcal{G}$ to be the length of the longest shortest path and γ to be the length of the longest path of the graph, which we compute as

$$\delta = \max_{u,v \in \mathcal{G}} d^-(u, v) \quad \text{and} \quad \gamma = \max_{u,v \in \mathcal{G}} d^+(u, v).$$

Lemma 1. *A message passing Graph Neural Network (GNN) has at least N_h hidden layers if and only if the output feature of every node i (i.e., $h_i^{N_h+1}$) receives information from any other node j via paths $p(j, i)$ such that $|p(j, i)| \leq N_h + 1$.*

Proof. **Step 1.** The statement is that the feature of every node i at the output of a GNN with N_h hidden layers (i.e., $h_i^{N_h+1}$) receives information from any other node j via paths $p(j, i)$ such that $|p(j, i)| \leq N_h + 1$. We give a proof by induction on N_h for an arbitrary node i, with input feature to the GNN h_i^0 and output feature $h_i^{N_h+1}$.

Base case: The statement holds for $N_h = 0$. By definition, a message passing GNN with one layer only exchanges messages between neighboring nodes. Hence, i) the output feature of node i is $h_i^1 = f(\{h_i^0\} \cup \{h_j^0 | j \in \mathcal{N}_i\}; \theta)$, which is only a function of the 1-hop ancestors (i.e., parents); ii) information is exchanged via paths that fulfill $|p(j, i)| \leq 1$.

Inductive step: We assume the statement holds for $N_h = k - 1$. In this case, i) the output feature of node i is $h_i^k = f(\{h_i^{k-1}\} \cup \{h_j^{k-1} | j \in \mathcal{N}_i\}; \theta)$, which is a function of the k-hop ancestors; ii) information is exchanged via paths that fulfill $|p(j, i)| \leq k$. For $N_h = k$, the output feature of node i is $h_i^{k+1} = f(\{h_i^k\} \cup \{h_j^k | j \in \mathcal{N}_i\}; \theta)$. Since h_j^k is a function of k-hop ancestors of node j, it follows that h_i^{k+1} is a function of the ancestors of h_j^k and on parents of node i, i.e., the output feature of node i is a function of its $(k + 1)$-ancestors. Then, it follows that for $N_h = k$, information is exchanged via paths that fulfill $|p(j, i)| \leq k + 1$.

Step 2. We assume that the output feature of every node i receives information from any other node j via paths $p(j, i)$ such that $|p(j, i)| \leq N_h + 1$. Then, there exist node i and j such that $|p(j, i)| = N_h + 1$. Then, by definition, a message passing GNN needs at least N_h hidden layers to capture paths $p(j, i)$ with length $|p(j, i)| \leq N_h + 1$. □

Remark. Lemma 1 implies that, for a given GNN with N_h hidden layers, the set of paths through which the output feature of node i is a function of node j is

$$P_{\text{GNN}}(j,i) = \{p(j,i) \mid p(j,i) \in P(j,i) \text{ and } |p(j,i)| \leq N_h + 1\}. \tag{7}$$

Additionally, if $|P_{\text{GNN}}(j,i)| = \varnothing$ then the output feature of node i is not a function of node j.

Proposition 1 (Causal factorization). *VCAUSE satisfies causal factorization, $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \prod_i p_{\theta_i}(X_i \mid \mathbf{Z}_{an^*(i)})$, if and only if the number of hidden layers in the decoder is greater or equal than $\delta - 1$, with δ being the length of the longest shortest path between any two endogenous nodes.*

Proof. Consider a causal graph $\mathcal{G} := (\mathbf{X}, \mathbf{E})$ with diameter δ and a GNN decoder with N_h hidden layers. We assume VCAUSE to satisfy causal factorization, i.e., $\boldsymbol{\eta}_i$ is a function $\mathbf{Z}_{an^*(i)}$ for all i. Therefore, there exist node i and j such that $d^-(j,i) = \delta$ (notice that this implies that j is an ancestor of i). Thus, by Lemma 1, the GNN decoder has $N_h \geq \delta - 1$ hidden layers. The converse is true because Lemma 1 is a bi-conditional statement. □

Proposition 2 (Causal interventions). *VCAUSE captures causal interventions if and only if the number of hidden layers in its decoder is greater than or equal to $\gamma - 1$, with γ being the length of the longest path between any two endogenous nodes in $\mathcal{G}$.*

A causal intervention involves to severe all the incoming edges to the intervened nodes. Thus, VCAUSE can only capture causal interventions, if it can model all the causal paths, i.e., $P_{\text{GNN}}(j,i) = P(j,i)\ \forall i,j$. Otherwise, severing some path will have no effect in the resulting intervention, as we prove next.

Proof. Consider a causal graph $\mathcal{G} := (\mathbf{X}, \mathbf{E})$ with length of the longest path between two nodes γ and a GNN decoder with N_h hidden layers. We assume that the GNN decoder models all the causal paths, i.e., $P_{\text{GNN}}(j,i) = P(j,i)\ \forall i,j$. By definition of γ, there exists at least one node i with an ancestor j such that $d^+(j,i) = \gamma$. Thus, by Lemma 1, the GNN decoder has $N_h \geq \gamma - 1$ hidden layers. The converse is true because Lemma 1 is a bi-conditional statement. □

Proposition 3 (Abduction). *The abduction step of an observed sample $\mathbf{x} = \{x_1, \ldots, x_d\}$ in VCAUSE satisfies that for all i the posterior of Z_i is independent on the subset $\{x_j\}_{j \notin pa^*(i)} \subseteq \mathbf{x}$, if and only if the encoder GNN has no hidden layers.*

Proof. Consider a causal graph $\mathcal{G} := (\mathbf{X}, \mathbf{E})$ and a GNN encoder with N_h hidden layers. We assume the posterior of Z_i is independent on the subset $\{x_j\}_{j \notin pa^*(i)} \subseteq \mathbf{x}$, i.e., the parameters $\boldsymbol{\eta}_i^{enc}$ (the output of the GNN) is a function of $\{x_j\}_{j \in pa^*(i)}$. Then, the GNN only models paths $p(j,i)$ such that $d^+(j,i) = 1$. It follows, by Lemma 1, that the number of hidden layers of the encoder GNN is $N_h = 0$. The converse is true because Lemma 1 is a bi-conditional statement. □

C VCAUSE implementation details

In this section, we extend Section 4.4 and provide further details about the implementation of VCAUSE for complex real-world datasets and causal graphs.

C.1 Heterogeneous endogenous variables

As described in Appendix A, each layer l of a GNN uses the same parameters $\theta = \{\theta_m^l, \theta_u^l\}$ (corresponding to f^m and f^u) to update the features of every node, i.e., $h_i^l = f(\{h_i^{l-1}\} \cup \{h_j^{l-1} \mid j \in \mathcal{N}_i\}; \theta)$ with f_θ being reused for all i. Nonetheless, the structural equations of an SCM define a unique function f_i for each node (see **Property 1**). To mimic this behavior, we will rely on *port numbering*. In particular, for a given causal graph $\mathcal{G}$, we uniquely identify each node with an index i and each edge with the pair of indexes of the nodes it connects. Then, we define a *disjoint GNN layer* by the following characteristics:

- The node indexes define unique update functions f_i^u for each node, with parameters θ_{ui}.
- The edge indexes define unique message functions f_{ij}^m for each edge, with parameters θ_{mij}.

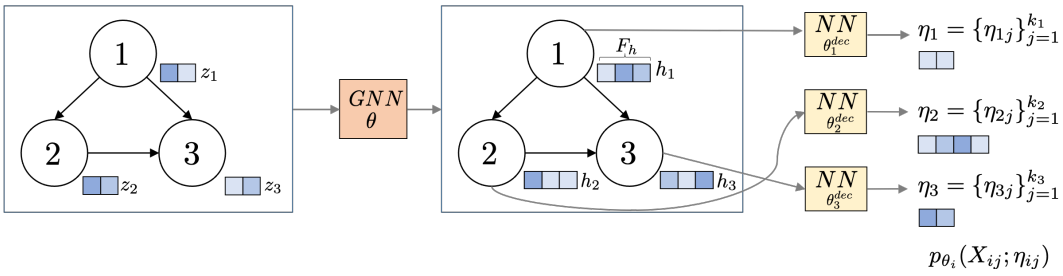

Figure 6: Heterogeneous VCAUSE decoder architecture.

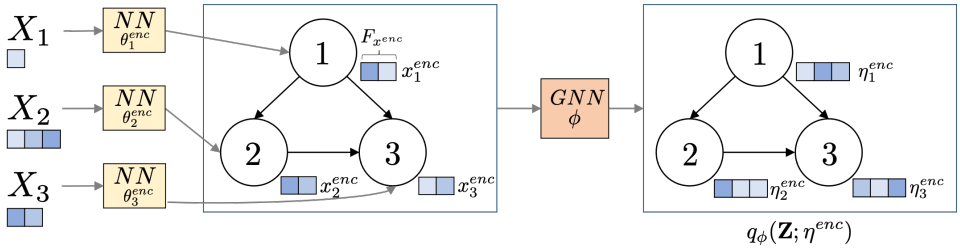

Figure 7: Heterogeneous VCAUSE encoder architecture.

Consequently, parameters are not shared among nodes and we can mimic the diversity of the structural equations of an SCM and model heterogeneous endogenous variables. In our GitHub repository, we present a PyTorch Geometric implementation of the *disjoint GNN layer*.

C.2 Heterogeneous causal nodes

Assume an SCM with d endogenous variables. As described in Section 4.4, is it possible to model an endogenous variable $\mathbf{X}_i$ of the SCM as a heterogeneous node, i.e., $\mathbf{X}_i = \{X_{i1}, \ldots, X_{ik_i}\}$, where k_i is the number of random variables in node i. In this section we describe the implications this has on the design of the encoder and decoder of VCAUSE.

Implications for the decoder. Given the heterogeneous nature of the nodes, the likelihood of VCAUSE factorizes as follows

$$p_\theta(\mathbf{X} \mid \mathbf{Z}) = \prod_{i=1}^{d} p_{\theta_i}(\mathbf{X}_i; \boldsymbol{\eta}_i) = \prod_{i=1}^{d} \prod_{j=1}^{k_i} p_{\theta_i}(X_{ij}; \boldsymbol{\eta}_{ij}) \text{ where } \boldsymbol{\eta}_i = \boldsymbol{\eta}_i(\mathbf{Z}_{an^*(i)}) \text{ and } \boldsymbol{\eta}_{ij} = \boldsymbol{\eta}_{ij}(\mathbf{Z}_{an^*(i)}).$$

Note, each $p(X_{ij}; \boldsymbol{\eta}_{ij})$ can be model with a different distribution, e.g., Gaussian or categorical. This implies that the likelihood parameters $\boldsymbol{\eta}_{ij}$ may differ for each random variable X_{ij} dependent on its type of distribution. However, the decoder GNN transforms the latent features into different features $\mathbf{H} \in \mathbb{R}^{d \times F_h}$, where the feature vector of each node i has the same dimensionality F_h. As a consequence, $\mathbf{H}$ cannot model the diversity in the likelihood parameters $\boldsymbol{\eta}_i$. To overcome such limitation, we add at the output of the GNN decoder a neural network (NN) per node i with parameters θ_i^{dec}. Such a NN transforms h_i, i.e. the output features of node i, into the set of likelihood parameters of each node $\boldsymbol{\eta}_i = \{\boldsymbol{\eta}_{ij}\}_{j=1}^{k_i}$, such that the likelihood parameters of each random variable $\boldsymbol{\eta}_{ij}$ satisfy the constraints of the corresponding likelihood $p(X_{ij}; \boldsymbol{\eta}_{ij})$ (e.g., non-negativity of variance for a Gaussian distribution). See Figure 6 for an illustration.

Implications for the encoder. Due to the heterogeneous nature of nodes, each endogenous variable $\mathbf{X}_i = \{X_{i1}, \ldots, X_{ik_i}\}$, can have a different number of random variables k_i and thus the node i corresponding to it in the GNN will have features of different dimensions. However, as described in Appendix A, a GNN takes as input in general a matrix feature $\mathbf{X} \in \mathbb{R}^{d \times F_{x^{enc}}}$. This implies, the features of every node share the same dimensionality $F_{x^{enc}}$. To overcome this limitation, we include for each node i a neural network (NN) with parameters θ_i^{enc} that transforms the corresponding heterogeneous random variable $\mathbf{X}_i$ into a feature vector with the dimension $F_{x^{enc}}$. See Figure 7 for an illustration.

D Experiments: setting, metrics and further results

This section provides a complete description of the experimental set-up, including the (semi-)synthetic datasets (Section D.1), training of VCAUSE , MultiCVAE [17] and CAREFL [18] (Section D.2) and metrics reported in the experiments (Section D.3).

D.1 Datasets

The following (semi-)synthetic datasets are taken from or inspired by [17]. The distribution of exogenous variables $p(\mathbf{U})$ for *triangle*, *chain*and *collider* follows Table 4 with with MoG denoting a mixture of Gaussian distributions.

Table 4: Distribution of exogenous variables $p(\mathbf{U})$ for SCM *triangle*, *chain*, *collider*.

SCM	$p(U_1)$	$p(U_2)$	$p(U_3)$
LIN	$\text{MoG}(0.5\mathcal{N}(-2,1.5)+0.5\mathcal{N}(1.5,1))$	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$
NLIN	$\text{MoG}(0.5\mathcal{N}(-2,1.5)+0.5\mathcal{N}(1.5,1))$	$\mathcal{N}(0,0.1)$	$\mathcal{N}(0,1)$
NADD	$\text{MoG}(0.5\mathcal{N}(-2.5,1)+0.5\mathcal{N}(2.5,1))$	$\mathcal{N}(0,0.25)$	$\mathcal{N}(0,0.0625)$

Collider. The *collider* is a synthetic dataset, which consists of 3 endogenous variables. The structural equations are shown in Table 5. Figure 8 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 3$ nodes, diameter $\delta = 1$ and longest path $\gamma = 1$.

Table 5: Structural equations $\tilde{\mathbf{F}}$ for SCM *collider* with $\mathbf{U} \sim p(\mathbf{U})$ in Table 4. Function $\text{sgn}(x)$ is returning an element-wise indication of the sign of x.

SCM	$\tilde{f}_1 := X_1$	$\tilde{f}_2 := X_2$	$\tilde{f}_3 := X_3$
LIN	U_1	U_2	$0.05X_1 + 0.25X_2 + U_3$
NLIN	U_1	U_2	$0.05X_1 + 0.25(X_2)^2 + U_3$
NADD	U_1	U_2	$-1 + 0.1\,\text{sgn}(U_3)((X_1)^2 + (X_2)^2)U_3$

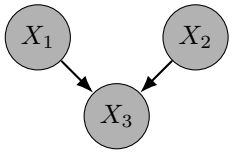

Figure 8: Causal graph for variables $\mathbf{X}$ of SCM *collider*.

Triangle. The *triangle* is a synthetic dataset, which consists of 3 endogenous variables. The structural equations are shown in Table 6. Figure 9 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 3$ nodes, diameter $\delta = 1$ and longest path $\gamma = 2$.

Table 6: Structural Equations $\tilde{\mathbf{F}}$ for SCM *triangle* with $\mathbf{U} \sim p(\mathbf{U})$ in Table 4. Function $\text{sgn}(x)$ is returning an element-wise indication of the sign of x.

SCM	$\tilde{f}_1 := X_1$	$\tilde{f}_2 := X_2$	$\tilde{f}_3 := X_3$
LIN	U_1	$-X_1 + U_2$	$X_1 + 0.25X_2 + U_3$
NLIN	U_1	$-1 + \frac{3}{(1+\exp(-2X_1))} + U_2$	$X_1 + 0.25(X_2)^2 + U_3$
NADD	U_1	$0.25\,\text{sgn}(U_2) * (X_1)^2(1 + (U_2)^2)$	$-1 + 0.1\,\text{sgn}(U_3)((X_1)^2 + (X_2)^2) + U_3$

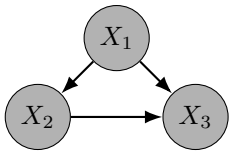

Figure 9: Causal graph for variables $\mathbf{X}$ of SCM *triangle*.

Chain. The *chain* is a synthetic dataset, which consists of 3 endogenous variables. The structural equations are shown in Table 7. Figure 10 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 3$ nodes, diameter $\delta = 2$ and longest path $\gamma = 2$.

Table 7: Structural Equations $\tilde{\mathbf{F}}$ for SCM *chain* with $\mathbf{U} \sim p(\mathbf{U})$ in Table 4. Function $\operatorname{sgn}(x)$ is returning an element-wise indication of the sign of x.

SCM	$\tilde{f}_1 := X_1$	$\tilde{f}_2 := X_2$	$\tilde{f}_3 := X_3$
LIN	U_1	$-X_1 + U_2$	$0.25 * X_2 + U_3$
NLIN	U_1	$-1 + \frac{3}{(1+\exp(-2X_1))} + U_2$	$0.25 * (X_2)^2 + U_3$
NADD	U_1	$0.25 \operatorname{sgn}(U_2)(X_1)^2(1 + (U_2)^2)$	$-1 + 0.1 \operatorname{sgn}(U_3)((X_2)^2) + U_3$

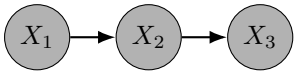

Figure 10: Causal graph for variables $\mathbf{X}$ of SCM *chain*.

M-graph. The *M-graph* is a synthetic dataset, which consists of 5 endogenous variables. Here, the distributions of exogenous variables follow $U_i \sim p(U_i) = \mathcal{N}(0, 1)\ \forall i \in 1 \dots 5$. The structural equations are shown in Table 8 and Figure 11 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 5$ nodes, diameter $\delta = 1$ and longest path $\gamma = 1$.

Table 8: Structural Equations $\tilde{\mathbf{F}}$ for SCM *M-graph* with $U_i \sim p(U_i) = \mathcal{N}(0, 1)\ \forall i \in 1 \dots 5$.

SCM	$\tilde{f}_1 := X_1$	$\tilde{f}_2 := X_2$	$\tilde{f}_3 := X_3$	$\tilde{f}_4 := X_4$	$\tilde{f}_5 := X_5$
LIN	U_1	U_2	$X_1 + U_3$	$-X_2 + 0.5X_1 + U_4$	$-1.5X_2 + U_5$
NLIN	U_1	U_2	$X_1 + 0.5(X_1)^2 + U_3$	$-X_2 + 0.5(X_1)^2 + U_4$	$-1.5(X_2)^2 + U_5$
NADD	U_1	U_2	$X_1 * U_3$	$(-X_2 + 0.5 * (X_1)^2)U_4$	$(-1.5(X_2)^2)U_5$

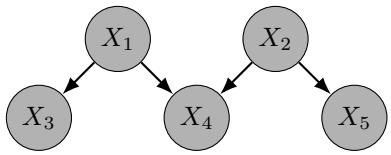

Figure 11: Causal graph for variables $\mathbf{X}$ of SCM *M-graph*.

Loan. The *loan* is a semi-synthetic dataset from [17], which reflects a loan approval setting in the real-world inspired by German Credit dataset [10]. It consists of 7 endogenous variables: *gender* G, *age* A, *education* E, *loan amount* L, *loan duration* D, *income* I and *savings* S with the following structural equations and distributions of exogenous variables:

$$
\begin{aligned}
f_G: &\quad G = U_G, && U_G \sim \text{Bernoulli}(0.5)\\
f_A: &\quad A = -35 + U_A && U_A \sim \text{Gamma}(10, 3.5)\\
f_E: &\quad E = -0.5 + \left(1 + e^{-\left(-1+0.5G+\left(1+e^{-0.1A}\right)^{-1}+U_E\right)}\right)^{-1} && U_E \sim \mathcal{N}(0, 0.25)\\
f_L: &\quad L = 1 + 0.01(A-5)(5-A) + G + U_L, && U_L \sim \mathcal{N}(0, 4)\\
f_D: &\quad D = -1 + 0.1A + 2G + L + U_D, && U_D \sim \mathcal{N}(0, 9)\\
f_I: &\quad I = -4 + 0.1(A+35) + 2G + GE + U_I, && U_I \sim \mathcal{N}(0, 4)\\
f_S: &\quad S = -4 + 1.5\mathbb{I}_{\{I>0\}} I + U_S, && U_S \sim \mathcal{N}(0, 25)
\end{aligned}
$$

Note, the authors model variables w.r.t. their relative meaning in terms of deviation from the mean. See [17] for further details. Figure 12 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 7$ nodes, diameter $\delta = 2$ and longest path $\gamma = 3$.

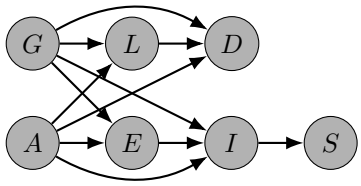

Figure 12: Causal graph for variables $\mathbf{X}$ of SCM *loan*.

D.2 Training and cross-validation

This section details the hyperparameter configurations of VCAUSE , MultiCVAE [17] and [18] for the experiments in Tables 1, 2 and Table 12. Across experiments and models we generate synthetic datasets consisting of 5000 training samples, 2500 test samples and 2500 validation samples and we use a batch size of 1000.

VCAUSE. We train the ELBO [20] for the encoder and the IWAE [3] with $K = 5$ for the decoder. The objective metric is the IWAE with $K = 100$. We use the Rectified Linear Unit (ReLU) as activation function. For the experiments in Table 1 and 2 we trained with a learning rate $\eta = 0.005$ for a maximum of 500 epochs, or alternatively until the objective metric does not improve in 50 epochs. Also, we regularize the training of VCAUSE using a novel *parents dropout:* randomly removing all incoming edges to the nodes with probability $p \in [0, 1)$. In our experiments, we observe that adding this regularization improves overall performance.

We cross-validated the parents dropout rate with values $\{0.1, 0.2\}$, the number of hidden layers of the decoder $\{0, 1, 2\}$ with 16 neurons each and the number of hidden layers in the message passing function f^m in the encoder $\{1, 2\}$ with 16 neurons each. The best models – according to the objective metric – are reported in Table 9. We use a latent variable dimension of 4 and a Gaussian likelihood with a small variance $\sigma^2 = \lambda_{KLD}/2$ with $\lambda_{KLD} = 0.05$.

MultiCVAE. In [17] the authors propose to train a conditional variational autoencoder (CVAE) for each endogenous variable that is not a root node in the causal graph (MultiCVAE). Different from [17], our implementation also models non-root nodes as CVAEs, since our goal is to model the joint distribution, while [17] target counterfactual distributions for algorithmic recourse only. Additionally, we perform the necessary modifications for training on normalized data.

The configuration of the algorithm is displayed in Table 10. The hyperparameter selection for the causal graph *triangle* with the three different types of structural causal equations (LIN, NLIN, NADD) was used as reported by the authors. We also chose the same configuration for *chain* and *collider*. The hyperparameter selection for the causal graph *loan* was obtained for all non-root nodes ($\{X_3, \ldots, X_7\}$) using the provided code by [17] sweeping over different configurations as indicated by the authors resulting in the configuration with the minimum MMD statistic between real and reconstructed samples. As we perform training on normalized data we assume $\lambda_{KLD} = 0.05$ for all SCMs and CVAEs.

CAREFL. In [18] the authors propose CAREFL, an autoregressive causal flows model for causal discovery, which also allows to answer interventional and counterfactual queries. The authors rely on real-valued non-volume preserving (real NVP) transformations, since they mainly focus on the

Table 9: Hyperparemeter selection for our VCAUSE training for the SCMs on the synthetic datasets *triangle*, *collider*, *chain* and *M-graph* and on the semi-synthetic dataset *loan*. Note, the encoder architecture refers to the layers in function f^m, while the decoder architecture refers to the different GNN layers.

SCM		Encoder Arch.	Decoder Arch.	Parents Dropout
chain	LIN	$1 \times 16 \times 16$	$16 \times 16 \times 1$	0.2
	NLIN	$1 \times 16 \times 16$	16×1	0.2
	NADD	$1 \times 16 \times 16$	$16 \times 16 \times 1$	0.1
collider	LIN	1×16	$16 \times 16 \times 1$	0.2
	NLIN	1×16	16×1	0.2
	NADD	1×16	16×1	0.2
triangle	LIN	$1 \times 16 \times 16$	16×1	0.2
	NLIN	$1 \times 16 \times 16$	16×1	0.1
	NADD	$1 \times 16 \times 16$	$16 \times 16 \times 1$	0.2
M-graph	LIN	1×16	1	0.2
	NLIN	1×16	$16 \times 16 \times 1$	0.2
	NADD	1×16	$16 \times 16 \times 1$	0.2
loan	-	1×16	$16 \times 16 \times 16 \times 1$	0.2

Table 10: Hyperparemeter selection for MultiCVAE [17] training for the SCMs on the synthetic datasets with three nodes (i.e., *triangle*, *collider* and *chain*), *M-graph* and for the semi-synthetic dataset *loan*.

SCM		CVAE	Encoder Arch.	Decoder Arch.	Latent Dim.
triangle / collider / chain	LIN	X_1	$1 \times 32 \times 32 \times 32$	$5 \times 5 \times 1$	1
		$X_2 \vert X_1$	$1 \times 32 \times 32 \times 32$	$5 \times 5 \times 1$	1
		$X_3 \vert X_1, X_2$	$1 \times 32 \times 32 \times 32$	$32 \times 32 \times 32 \times 1$	1
	NLIN	X_1	$1 \times 32 \times 32$	$32 \times 32 \times 1$	5
		$X_2 \vert X_1$	$1 \times 32 \times 32$	$32 \times 32 \times 1$	5
		$X_3 \vert X_1, X_2$	$1 \times 32 \times 32 \times 32$	$32 \times 32 \times 1$	1
	NADD	X_1	$1 \times 32 \times 32 \times 32$	$32 \times 32 \times 1$	3
		$X_2 \vert X_1$	$1 \times 32 \times 32 \times 32$	$32 \times 32 \times 1$	3
		$X_3 \vert X_1, X_2$	$1 \times 32 \times 32 \times 32$	$5 \times 5 \times 1$	3
M-graph	LIN	X_1	$1 \times 32 \times 32$	$32 \times 32 \times 1$	1
		$X_2 \vert X_1$	$1 \times 32 \times 32$	$32 \times 32 \times 1$	1
		$X_3 \vert X_1, X_2$	$1 \times 32 \times 32$	$32 \times 32 \times 1$	1
	NLIN	X_1	$1 \times 32 \times 32$	$32 \times 32 \times 1$	1
		$X_2 \vert X_1$	$1 \times 32 \times 32$	$32 \times 32 \times 1$	1
		$X_3 \vert X_1, X_2$	$1 \times 32 \times 32$	$32 \times 32 \times 1$	1
	NADD	X_1	$1 \times 32 \times 32$	$32 \times 32 \times 1$	1
		$X_2 \vert X_1$	$1 \times 32 \times 32$	$32 \times 32 \times 1$	1
		$X_3 \vert X_1, X_2$	$1 \times 32 \times 32$	$32 \times 32 \times 1$	1
loan		X_1	$1 \times 5 \times 5$	2×1	2
		X_2	$1 \times 5 \times 5$	2×1	2
		$X_3 \vert X_1$	$1 \times 5 \times 5$	2×1	2
		$X_4 \vert X_1, X_2$	$1 \times 3 \times 3$	$3 \times 3 \times 1$	1
		$X_5 \vert X_1, X_2, X_4$	$1 \times 5 \times 5$	$3 \times 3 \times 1$	2
		$X_6 \vert X_1, X_2, X_4$	$1 \times 3 \times 3 \times 3$	$3 \times 3 \times 1$	1
		$X_7 \vert X_6$	$1 \times 5 \times 5$	2×1	2

multivariate bi-variate case. As this flow architecture is not suited for general graphs, we use their framework with Neural Spline Autoregressive Flows. We have cross validated the number of flows $\{2, 4, 5\}$ and the number of hidden units of the neural networks $\{10, 32\}$. The objective metric is the log evidence. The final configuration is displayed in Table 11.

Table 11: Hyperparemeter selection for CAREFL [18] training for different SCMs.

SCM		Flows	Hidden Units
chain	LIN	2	10
	NLIN	4	10
	NADD	5	32
collider	LIN	2	10
	NLIN	2	10
	NADD	2	32
triangle	LIN	2	10
	NLIN	5	10
	NADD	4	32
M-graph	LIN	2	10
	NLIN	2	10
	NADD	2	10
loan	-	4	10

D.3 Performance metrics

In the following we describe the metrics used to evaluate the performance of VCAUSE in Section 5. In all experiments we use (semi-)synthetic datasets with access to samples from the ground truth distribution $\{\mathbf{x}_i\}_{i=0}^n \sim P$ as well as from the estimated distribution $\{\hat{\mathbf{x}}_i\}_{i=0}^n \sim Q$.

For the interventional and counterfactual distribution, we perform a set of interventions $\mathcal{I} = \{do(\mathbf{X}_{\mathcal{I}_j} = \alpha_j)\}_j$, where $\mathcal{I}_j \in [d]$ and $\alpha_j \in \{-1.0, -0.5, 0.0, 0.5, 1.0\} \times \sigma_{\mathcal{I}_j}$ with $\sigma_{\mathcal{I}_j}$ as the empirical standard deviation of the intervened variable $\mathbf{X}_{\mathcal{I}_j}$ prior to intervention (i.e., in the observational distribution). Note that we only intervene on one variable at a time. For each intervention in $\mathcal{I}$, we are interested in the estimated distribution of variables causally affected by the intervention $\{\mathbf{X}_i | i \in des(\mathcal{I}_j)\}$, i.e., the set of descendants of the variable intervened. Note that $des(\mathcal{I}_j)$ refers to the set of indexes of the descendants. It follows that we do not intervene on leaf nodes.

Mean Maximum Discrepancy (MMD). The Mean Maximum Discrepancy (MMD) [13] is a kernel-based distance-measure between two distributions P and Q on the basis of samples from both distributions. The smaller the MMD, the more likely it is that the sets of samples are drawn from the same distributions, i.e. the better distributions match. Without access to underlying distribution, we can compute an unbiased empirical squared MMD estimate using a kernel function k as[13]:

$$\widehat{\mathrm{MMD}}^2(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{n(n-1)} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} k\left(\mathbf{x}_i, \mathbf{x}_j\right) + \sum_{i=1}^{n} \sum_{j=1}^{n} k\left(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j\right) - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} k\left(\mathbf{x}_i, \hat{\mathbf{x}}_j\right) \right). \tag{8}$$

In our implementation we use as kernel a mixture of RBF (Gaussian) kernels with different bandwidths and sample size $n = 1000$.

Estimation squared error for the mean (MeanE). For the interventional distribution, we compute the estimation squared error for the mean (MeanE) as the average (across interventions) of the squared difference between the empirical means of the true and estimated interventional distributions (for the descendants of the intervened variables):

$$\text{MeanE} = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{I}_j \in \mathcal{I}} \frac{1}{|des(\mathcal{I}_j)|} \sum_{i \in des(\mathcal{I}_j)} \left(E[X_i^{\mathcal{I}_j}] - E[\hat{X}_i^{\mathcal{I}_j}] \right)^2 \tag{9}$$

Estimation squared error for the standard deviation (StdE). For the interventional distribution, we compute the estimation squared error for the standard deviation (StdE) as the average (across interventions) of the squared difference between the empirical standard deviation of the true $\tilde{\sigma}(X_i^{\mathcal{I}_j})$ and estimated $\tilde{\sigma}(\hat{X}_i^{\mathcal{I}_j})$ interventional distributions (for the descendants of the intervened variables):

$$\text{StdE} = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{I}_j \in \mathcal{I}} \frac{1}{|des(\mathcal{I}_j)|} \sum_{i \in des(\mathcal{I}_j)} \left(\tilde{\sigma}(X_i^{\mathcal{I}_j}) - \tilde{\sigma}(\hat{X}_i^{\mathcal{I}_j}) \right)^2 \tag{10}$$

Mean squared error (MSE). For the counterfactual distribution, we compute the mean squared error (MSE) as the average (across interventions) of the pairwise squared difference between true and estimated counterfactual values for the descendants of the intervened variable. More in detail, let us define the random variable $T^{\mathcal{I}_j}$ as the *Frobenius norm* of the difference between true $\mathbf{x}^{CF}_{des(\mathcal{I}_j)}$ and estimated $\hat{\mathbf{x}}^{CF}_{des(\mathcal{I}_j)}$ counterfactual values for the descendants of the intervened variable, i.e.,

$$T^{\mathcal{I}_j} = ||\mathbf{x}^{CF}_{des(\mathcal{I}_j)} - \hat{\mathbf{x}}^{CF}_{des(\mathcal{I}_j)}||_2^2, \tag{11}$$

Thus, we can compute the counterfactual MSE as:

$$\text{MSE} = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{I}_j \in \mathcal{I}} \frac{1}{|des(\mathcal{I}_j)|} E\left[T^{\mathcal{I}_j}\right] \tag{12}$$

Standard deviation of the squared error (SSE). Similarly, we can compute the average (across interventions) of the standard deviation of the counterfactual squared error as:

$$\text{SSE} = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{I}_j \in \mathcal{I}} \frac{1}{|des(\mathcal{I}_j)|} \sum_{i \in des(\mathcal{I})} \tilde{\sigma}\left(T^{\mathcal{I}_j}\right), \tag{13}$$

where $\tilde{\sigma}\left(T^{\mathcal{I}_j}\right)$ denotes the empirical standard deviation of $T^{\mathcal{I}_j}$.

D.4 Additional results

In the following we present additional results that empirically show the potential of VCAUSE to model interventional and counterfactual queries. In particular, we report the results for the *collider*, *M-graph*, and *chain* graphs. We remark that the following results are consistent with the ones reported in the main manuscript for the *triangle* and *loan*.

Results for interventional distributions. Table 12 (middle columns) reports the MMD, MeanE, and StdE for the interventional distribution. In accordance with the results shown in the main manuscript, we can observe that i) VCAUSE consistently outperforms other methods in terms of MMD; ii) the three methods provide comparable results in capturing the mean of the interventional distribution (MeanE); and iii) CAREFL and MultiCVAE often fail to capture the standard deviation of the interventional distribution (StdE), while VCAUSE provides a more accurate estimate of the overall interventional distribution (as can be easily seen in the MMD).

Results for the counterfactuals. Table 12 also reports the results for the counterfactual distribution, in terms of MSE and SSE. As reported in the main text, we observe that CAREFL provides more accurate estimates than VCAUSE and MultiCVAE in terms of MSE, which may be explained by the fact that CAREFL performs exact inference as opposed to the approximated inference of the other two approaches. However, CAREFL presents high variance in its results (see SSE). In contrast, VCAUSE leads to regularly lower values of SSE, which suggests more consistent counterfactual estimations across factual samples and interventions.

Table 12: Performance of different methods at estimating the observational, interventional and counterfactual of different SCMs. All metrics are shown in percentage (%).

SCM		Model	Obs.	Interventional			Counterfactuals	
			MMD (%)	MMD (%)	MeanE (%)	StdE (%)	MSE (%)	SSE (%)
collider	LIN	MultiCVAE	1.83±0.65	2.50±0.73	0.53±0.44	**25.75±0.26**	**5.35±1.72**	4.79±3.20
		CAREFL	4.58±1.22	2.40±0.27	**0.24±0.06**	41.97±0.63	**4.93±0.50**	4.84±0.54
		VCAUSE	**1.15±0.64**	**0.91±0.15**	**0.25±0.13**	26.14±0.17	**4.42±0.55**	**3.30±0.55**
	NLIN	MultiCVAE	1.55±0.66	1.56±0.54	**0.14±0.04**	**25.87±0.17**	4.78±0.61	3.98±0.77
		CAREFL	4.14±1.08	2.24±0.42	0.20±0.08	41.60±0.50	4.82±0.40	4.64±0.50
		VCAUSE	**0.89±0.59**	**0.75±0.13**	**0.14±0.07**	26.42±0.09	**2.73±0.30**	**2.03±0.27**
	NADD	MultiCVAE	11.24±10.53	69.13±44.39	16.71±9.89	**33.05±2.07**	27.52±9.74	25.67±5.97
		CAREFL	4.15±0.80	**3.75±0.42**	**0.40±0.13**	54.57±0.69	**6.24±0.19**	**16.61±0.26**
		VCAUSE	**1.20±0.33**	11.70±1.20	8.71±0.75	40.20±1.23	17.07±0.55	21.60±0.49
M-graph	LIN	MultiCVAE	17.85±2.02	40.73±3.71	3.70±0.65	22.06±1.49	26.66±1.09	13.98±0.48
		CAREFL	6.79±1.21	6.82±0.48	0.28±0.10	25.50±0.75	3.92±0.15	5.78±0.10
		VCAUSE	**0.93±0.29**	**1.03±0.11**	**0.16±0.03**	**6.39±0.04**	**2.62±0.06**	**1.38±0.05**
	NLIN	MultiCVAE	14.32±2.57	44.21±7.54	5.22±1.23	22.63±1.41	26.33±0.95	15.72±0.47
		CAREFL	7.32±1.59	7.82±0.54	**0.60±0.07**	27.02±0.86	6.16±0.17	17.76±0.06
		VCAUSE	**3.27±1.30**	**2.22±0.53**	2.03±0.90	**10.59±2.24**	**4.46±0.75**	**4.24±0.55**
	NADD	MultiCVAE	5.26±0.94	10.65±1.16	0.45±0.28	26.02±2.08	23.10±1.41	**23.28±0.82**
		CAREFL	5.76±1.33	7.37±0.44	0.24±0.09	30.36±0.64	**17.03±0.22**	29.12±0.12
		VCAUSE	**1.16±0.35**	**4.08±0.73**	**0.19±0.04**	**13.42±0.92**	18.49±0.39	24.53±1.11
chain	LIN	MultiCVAE	3.04±2.72	4.85±3.67	1.21±1.00	**22.16±0.38**	8.19±2.79	6.75±1.96
		CAREFL	5.88±0.99	4.50±0.37	**0.33±0.09**	49.76±0.98	**6.95±0.95**	8.23±1.19
		VCAUSE	**1.38±0.83**	**1.47±0.43**	0.46±0.12	**22.11±0.15**	9.19±0.67	**6.04±0.39**
	NLIN	MultiCVAE	**2.21±0.74**	8.38±2.21	**4.21±1.06**	**23.64±0.27**	**21.33±2.04**	**14.59±1.37**
		CAREFL	5.21±0.56	13.27±3.94	7.77±3.08	53.83±0.58	**16.68±6.15**	18.39±7.24
		VCAUSE	**2.38±0.73**	**6.52±0.90**	**4.24±0.37**	24.46±0.23	**22.74±2.17**	16.65±1.83
	NADD	MultiCVAE	**2.33±0.73**	59.66±9.01	**0.33±0.19**	**15.68±1.93**	24.88±1.66	42.08±4.70
		CAREFL	7.45±1.21	83.27±15.26	1.50±0.67	125.28±9.54	**9.54±0.74**	47.36±0.32
		VCAUSE	4.62±2.35	**40.00±13.21**	0.84±1.57	37.18±17.29	14.31±1.80	**25.58±1.89**

E Further details on the counterfactual fairness use-case

In this section we provide further details on dataset, training, metrics and additional results for the use-case of counterfactual fairness in Section 6.

E.1 German Credit Dataset

The German Credit dataset from the UCI repository [10] contains 20 attributes from 1000 loan applicants. We rely on the causal model proposed by in [6] for the following subset of features as exogenous variables $\mathbf{X}$ (see Figure 5): sensitive feature $S = \{sex\}$, and non-sensitive features $C = \{age\}$, $R = \{credit\ amount, repayment\ history\}$ and $H = \{checking\ account, savings, housing\}$. The causal graph in Figure 5 has a diameter $\delta = 1$ and longest path $\gamma = 1$. The goal of a classifier h is to predict $Y = \{creditrisk\}$ from $\mathbf{X}$. We load and pre-process the data using the aif360 library such that the dataset contains binary outcome variable Y (0-bad, 1-good) and a binary sensitive attribute S (0-female, 1-male). Note that the dataset contains 700 labels $Y = 1$ and 300 labels $Y = 0$, i.e., it is imbalanced. It also contains 690 males $S = 1$ and 310 females $Y = 0$. Note also that the causal model contains heterogeneous causal nodes (R and S), as addressed in Section 4.4. For example, [6] assume that the relationship between *credit amount* and *repayment history* is unknown, or that it may be affected by hidden confounders. This leads to an undirected path between the random variables and they are grouped together in one multidimensional causal node R. This applies similarly to node S.

E.2 Training

In this section we provide further information on training VCAUSE on the German Credit dataset [10] and detail the different classifiers in Section 6. We use a 80% training, 10% validation, 10% training data split.

VCAUSE . Training for VCAUSE was performed on normalized data—performing normalization only on the continuous variables, i.e. r.v. C and R in Fig 5. We trained a heterogeneous VCAUSE as described in Section with a message passing function f^m with one hidden layer of 16 neurons, a decoder with one hidden layer of 16 neurons and a latent variable with dimension 4. We trained the model using the PIWAE [41] approach with $\lambda_{KLD} = 0.05$, specifically, the encoder with the IWAE [3] objective with $K = 5$ and the decoder with a β-ELBO with $\beta = 0.5$. We use a parents dropout rate (see Appendix D.2) of 0.2, learning rate of 0.005 and batch size 100.

Classifiers. Classifiers LogisticRegression and SVM are taken from the scikit-learn library and trained with default parameters as well as class_weight = balanced due to the class imbalance of the dataset.

E.3 Metrics

In this section we detail the measures f1-score and unfairness reported in Table 3 as well as accuracy in Table 13.

f1-score. Due to class imbalance, we measure classifier performance with the f1-score. The f1-score is the weighted average of the precision and recall and can assume values between 0 and 1; the higher the values the better. Our implementation relies on the f1_score from the scikit-learn library. We compute in expectation over our training dataset:

$$\text{f1-score} = \mathbb{E}\left[2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}\right]. \tag{14}$$

where precision is the ratio $\frac{TP}{TP+FP}$ with the number of true positives TP and the number of false positives FP and recall is the ratio $\frac{TP}{TP+FN}$ with the number of false positives FP.

Counterfactual (un)fairness. We measure counterfactual unfairness [24] with counterfactual instances $\mathbf{x}^{CF}$ and classifier prediction $h(\mathbf{x}^{CF}) = \hat{y}^{CF}$ as expectation over our training dataset:

$$\text{unfairness} = \mathbb{E}\left[\left|p(y^F) = 1|\mathbf{x}^F) - p(\hat{y}^{CF} = 1|do(S = a'), \mathbf{x}^F)\right|\right] \tag{15}$$

where $a' = 1 - a$ as $S \in \{0, 1\}$.

Accuracy. In Table 13, we report additionally the prediction accuracy as performance measure of classifier h with respect to factuals (samples) $(\mathbf{x}^F, y^F)$ and prediction $h(\mathbf{x}^F) = \hat{y}^F$ in expectation over our training dataset:

$$\text{accuracy} = \mathbb{E}\left[\mathbb{1}\left({y^F}_i = \hat{y}_i^F\right)\right]. \tag{16}$$

Our implementation relies the accuracy_score from the scikit-learn library.

Table 13: Evaluation of counterfactual (un)fairness and performance. All metrics are shown in %. Lower/Larger values of unfairness/f1-score are better.

Metric	Classifier	full	unaware	fair	VCAUSE
↑ accuracy (%)	LR	65.00	62.00	46.00	67.00
	SVM	68.00	65.00	52.00	63.00
↑ f1-score (%)	LR	71.07	68.33	50.00	74.81
	SVM	74.60	72.44	64.71	70.40
↓ unfairness (%)	LR	5.93	2.25	0.16	0.85
	SVM	6.07	2.68	0.20	1.00