Causal normalizing flows: from theory to practice

Adrián Javaloy[1] **Pablo Sánchez-Martín**[1,2] **Isabel Valera**[1,3]

[1]Computer Science Dept., Saarland University, Saarbrücken, Germany
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[3]Max Planck Institute for Software Systems, Saarbrücken, Germany

Abstract

In this work, we bridge the gap between normalizing flows and causal inference. First, we leverage recent results on non-linear ICA to show that causal models are identifiable from observational data given a causal ordering, and thus can be recovered using autoregressive normalizing flows (NFs). Second, we propose a simple design for *causal normalizing flows* to ease learning and capture the underlying causal data-generating process. Third, we describe how to implement the *do-operator* in causal NFs, and thus, how to answer interventional and counterfactual questions. Finally, we empirically validate our proposed design by comparing causal NFs to other approaches for approximating causal models, and demonstrate that causal NFs can be used in real-world problems—where mixed discrete-continuous data and partial knowledge on the causal graph is the norm.

1 INTRODUCTION

The focus of this work is to effectively solve causal inference problems, i.e., answering *what-if* questions about a causal system [22], using only observational data and (potentially partial) knowledge on the causal graph of the underlying structural causal model (SCM). This is exemplified in Fig. 1, where our proposed framework estimated the (unobserved) red and yellow distributions *solely from the blue distribution and partial information about the causal graph.*

In this context, previous works often rely on different deep neural networks (DNNs)—e.g., normalizing flows (NFs) [17, 20, 21], generative adversarial networks (GANs) [15, 31], variational autoencoders (VAEs) [10, 32], Gaussian processes (GPs) [10], or denoising diffusion probabilistic models (DDPMs) [2]—to iteratively estimate the conditional distribution of each observed variable given its direct causes, thus using an independent DNN per observed variable. Some approaches risk error propagation and a high number of parameters, addressed in practice with ad-hoc parameter amortization techniques [20, 21].

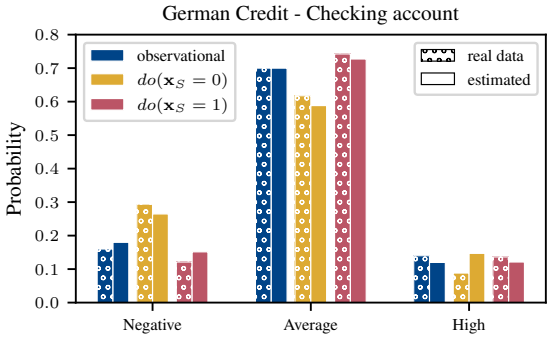


Figure 1: Observational and interventional distributions of the categorical variable *checking account* of the German Credit dataset [7], and their estimated values with a causal NF. $\mathbf{x}_S$ represents the users' sex as a binary variable.

In contrast, we here aim at learning the full causal-generating process using a single DNN, similar to [11, 25, 26, 33] and, in particular, using a *causal normalizing flow*. To this end, we show in §3 that causal NFs can approximate a broad class of SCMs, design causal NFs that inherently satisfy the necessary conditions to capture the underlying SCM, and introduce an implementation of the do-operator to efficiently tackle causal inference tasks. Finally, in §4 we empirically show that causal NFs outperform competing methods, and that they can be used in real-world problems with mixed-type data and partial causal graphs.

2 BACKGROUND

Causality A structural causal model (SCM) [22] is a tuple $\mathcal{M} = (\tilde{\mathbf{f}}, P_{\mathbf{u}})$ describing a data-generating process that transforms a set of d exogenous variables, $\mathbf{u} \sim P_{\mathbf{u}}$, into a set of d endogenous variables, $\mathbf{x}$, according to $\tilde{\mathbf{f}}$:

$$\mathbf{u} \coloneqq (\mathrm{u}_1, \dots, \mathrm{u}_d) \sim P_{\mathbf{u}}\,, \quad \mathrm{x}_i = \tilde{f}_i(\mathbf{x}_{\mathrm{pa}_i}, \mathrm{u}_i) \quad \forall i\,. \tag{1}$$

Accepted for the 6th Workshop on Tractable Probabilistic Modeling at UAI (TPM 2023).

An SCM also induces a causal graph, a powerful tool to reason about the causal dependencies of the system. Namely, the causal graph of an SCM $\mathcal{M} = (\tilde{\mathbf{f}}, P_{\mathbf{u}})$ is the directed graph that describes the functional dependencies of the causal mechanism. Furthermore, the direct causes of the i-th variable (pa_i in Eq. 1) are the *parent* nodes of the i-th node in $\boldsymbol{G}$, and the *ancestors* of this node (which we denoted by an_i) are its (in)direct causes. See Fig. 2 for an example of a causal chain. If $\boldsymbol{G}$ is acyclic, we can also pick a causal ordering π describing which variables do *not* cause others, and which ones *may* cause them.

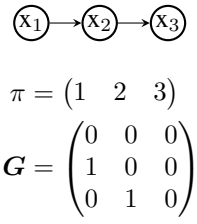


Figure 2: A causal graph, ordering π, and its adjacency matrix $\boldsymbol{G}$.

Normalizing flows We can express the probability of a set of observations using normalizing flows (NFs) [19]. Given an observed random vector $\mathbf{x}$ of size d, an NF $T_{\boldsymbol{\theta}}$ is a neural network that produces $T_{\boldsymbol{\theta}}(\mathbf{x}) =: \mathbf{u} \sim P_{\mathbf{u}}$, with log-density $\log p(\mathbf{x}) = \log p(T_{\boldsymbol{\theta}}(\mathbf{x})) + \log|\det(\nabla_{\mathbf{x}} T_{\boldsymbol{\theta}}(\mathbf{x}))|$, where $P_{\mathbf{u}}$ is a known base distribution. The network parameters $\boldsymbol{\theta}$ are usually learnt via maximum likelihood estimation (MLE) [1]. We focus here on autoregressive normalizing flows (ANFs) [13, 18], where the i-th output of the l-th layer, denoted by z_i^l, is computed as

$$\mathrm{z}_i^l := \tau_i^l(\mathrm{z}_i^{l-1}; \mathbf{h}_i^l), \quad \text{where} \quad \mathbf{h}_i^l := c_i^l(\mathbf{z}_{1:i-1}^{l-1}), \quad (2)$$

with τ_i and c_i termed the transformer and the conditioner, respectively. The transformer is a strictly monotonic function of z_i^{l-1}, while the conditioner can be arbitrarily complex, yet it only takes the variables preceding z_i as input. Thus, ANFs have triangular Jacobian matrices, $\nabla_{\mathbf{x}} T_{\boldsymbol{\theta}}(\mathbf{x})$.

3 CAUSAL NORMALIZING FLOWS

Problem statement We assume a sequence of i.i.d. observations $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ generated according to an unknown SCM $\mathcal{M}$, from which we have partial knowledge of its causal structure. Our objective is to design and learn an ANF $T_{\boldsymbol{\theta}}$, with parameters $\boldsymbol{\theta}$, that captures $\mathcal{M}$ by maximizing the observational likelihood (MLE), i.e.,

$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \log \left[p\left(T_{\boldsymbol{\theta}}\left(\boldsymbol{x}_n\right)\right) \cdot \left|\det(\nabla_{\mathbf{x}} T_{\boldsymbol{\theta}}\left(\boldsymbol{x}_n\right))\right| \right], \quad (3)$$

and that can successfully answer interventional and counterfactual queries, thus enabling causal inference. We refer to these models as *causal normalizing flows*.

Assumptions We make the following common assumptions: i) C^1*-diffeomorphic data-generating process*, i.e., $\tilde{\mathbf{f}}$ is invertible, and both $\tilde{\mathbf{f}}$ and its inverse are continuously differentiable; ii) *no feedback loops*, i.e., the induced causal graph is acyclic; and iii) *causal sufficiency*, i.e., the exogenous variables are mutually independent, $p(\mathbf{u}) = \prod_i p(\mathbf{u}_i)$.

SCMs as TMI maps To bridge the gap between SCMs and ANFs, we resort to triangular monotonic increasing (TMI) maps, which are autoregressive functions whose i-th component is strictly monotonic increasing with respect to its i-th input. Conveniently, an ANF layer (Eq. 2) is a parametric TMI map that can approximate any other TMI map arbitrarily well [19],[1] and any SCM can be rewritten as a tuple $(\mathbf{f}, P_{\mathbf{u}}) \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}$, where $\mathcal{F}$ is the set of all TMI maps, and $P_{\mathbf{u}}$ is the set of all fully-factorized distributions, $p(\mathbf{u}) = \prod_i p(\mathbf{u}_i)$. Given an acyclic SCM $\mathcal{M} = (\tilde{\mathbf{f}}, P_{\mathbf{u}})$ with $\tilde{\mathbf{f}} : \mathbb{X} \times \mathbb{U} \to \mathbb{X}$ as in Eq. 1, we can always unroll $\tilde{\mathbf{f}}$ by recursively replacing each x_i in the causal equation by its function $\tilde{f}_i$ (see App. B.1.1 for an example), obtaining an equivalent non-recursive function $\hat{\mathbf{f}} : \mathbb{U} \to \mathbb{X}$. Since $\mathcal{M}$ is acyclic, this function $\hat{\mathbf{f}}$ is a triangular map. Now, following the causal ordering, we can apply a Knöthe-Rosenblatt (KR) transport [14, 24], replacing each function $\hat{f}_i$ by a composition of conditional quantile functions of the variable x_i given $\mathbf{x}_{\mathrm{pa}_i}$ (which depends on $\mathbf{u}_{\mathrm{an}_i}$) eventually arriving to a TMI map $\mathbf{f}$ as desired.

Isolating the exogenous variables Now that we have SCMs and causal NFs under the family $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$ of TMI maps with fully-factorized distributions, we leverage existing results on identifiability to show that we can find a causal NF $T_{\boldsymbol{\theta}}$ such that the i-th component of $T_{\boldsymbol{\theta}}(\mathbf{x})$ is a function of the true exogenous variable u_i that generated the observed data. More precisely, note that identifying the true exogenous variables of an SCM $\mathcal{M}$ is equivalent to solving a non-linear ICA problem with TMI generators, for which Xi and Bloem-Reddy [30] proved the following:

Theorem 1 (Identifiability)**.** If two elements of the family $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$ produce the same observational distribution, then the two data-generating process differ by an invertible, component-wise transformation of the variables $\mathbf{u}$.

Thm. 1 implies that, if a causal NF $(T_{\boldsymbol{\theta}}, P_{\boldsymbol{\theta}})$ matches the observational distribution of $\mathcal{M} = (\mathbf{f}, P_{\mathcal{M}})$ (both in $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$), then we know that the exogenous variables of the flow differ from the real ones by a function of each component independently, i.e., $T_{\boldsymbol{\theta}}(\mathbf{f}(\mathbf{u})) = \boldsymbol{h}(\mathbf{u}) \sim P_{\boldsymbol{\theta}}$ with $\mathbf{u} \sim P_{\mathcal{M}}$, where, for each i-th component, $h_i(\mathbf{u}) = h_i(\mathrm{u}_i)$ is an invertible function. Fig. 3 graphically illustrates Thm. 1. Furthermore, Thm. 1 also implies that the functional dependencies of the causal NF must agree with that of the SCM, i.e., that $T_{\boldsymbol{\theta}}$ needs to be *causally*

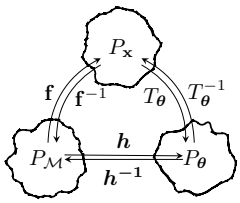


Figure 3: Thm. 1 as a commutative diagram.

[1] While it is common to shuffle the inputs for each layer, we keep the same order across the network.

consistent with $\mathcal{M}$. We formally present this result in the following corollary (the proof can be found in App. A):

Corollary 2 (Causal consistency)**.** If a causal NF $T_{\boldsymbol{\theta}}$ isolates the exogenous variables of an SCM $\mathcal{M}$, then $T_{\boldsymbol{\theta}}$ is causally consistent with the true data-generating process, $\mathcal{M}$.

To sum up, we have shown that *causal NFs are a natural choice to estimate an unknown SCM* by showing that: i) both SCMs and causal NFs fall within the same family $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$; ii) any two elements of this family with identical observational distributions are causally consistent; and iii) they differ by an invertible component-wise transformation.

3.1 CAUSAL NFS FOR REAL-WORLD PROBLEMS

To bring theory to practice, in this section we discuss critical details to ease the optimization of causal NFs, handle mixed discrete-continuous data and partial knowledge on the causal graph, and compute interventions and counterfactuals. Due to space limitations, we provide here a brief explanation, and formalize these ideas in App. B.

Network design Thm. 1 assumes that the causal NF perfectly matches the true observational distribution. In practice, however, reaching the optimal parameters may be tricky. To ease this task, we propose to use a single-layer ANF model going from $\mathbf{x}$ to $\mathbf{u}$ and, given a causal graph $\boldsymbol{G}$, to use the extra information to mask each layer of the causal NF according to $\boldsymbol{I} + \boldsymbol{G}$, such that

$$\mathrm{u}_i = \tau_i(\mathrm{x}_i; \mathbf{h}_i)\,, \quad \text{where} \quad \mathbf{h}_i = c_i(\mathbf{x}_{\mathrm{pa}_i})\,. \tag{4}$$

Despite its simplicity, this architecture enjoys a number of remarkable properties that result in the necessary conditions to approximate the underlying SCM. Specifically, it is:

- **Expressive**. Since an ANF layer is a universal TMI approximator [19], it can approximate any SCM.
- **Causally consistent**. As stated in Cor. 2, the causal NF needs to share the causal dependencies of the SCM at the optima. Given the causal graph $\boldsymbol{G}$, the masking in Eq. 4 ensures that the model is causally consistent by design.
- **Causal-path preserving**. To intervene, the causal dependencies from $\mathbf{u}$ to $\mathbf{x}$ need to follow the same paths as the true causal model. Since ANFs are inverted sequentially, this model can capture all indirect dependencies of $\mathbf{u}$ on $\mathbf{x}$ with no shortcuts, even with a single layer.

Do-operator In order to answer *what-if* queries, we need to use the *do-operator* [23], $do(\mathrm{x}_i = \alpha)$, which simulates a physical intervention on an SCM $\mathcal{M}$, inducing an alternative model $\mathcal{M}^{\mathcal{I}}$ that fixes the observational value $\mathrm{x}_i = \alpha$, removing any causal dependency on x_i. The traditional implementation yields an SCM $\mathcal{M}^{\mathcal{I}} = (\tilde{\mathbf{f}}^{\mathcal{I}}, P_{\mathbf{u}})$ result of replacing the i-th component of $\tilde{\mathbf{f}}$ with a constant function, $\tilde{f}_i^{\mathcal{I}} := \alpha$. Unfortunately, this implementation relies on recursivity of the causal equations, and thus does not generalize to the proposed causal NF. We instead propose to manipulate the SCM by modifying the exogenous distribution $P_{\mathbf{u}}$. Namely, an intervention $do(\mathrm{x}_i = \alpha)$ updates $P_{\mathbf{u}}$, restricting the set of plausible $\mathbf{u}$ to those that yield the intervened value α. We define the intervened SCM as $\mathcal{M}^{\mathcal{I}} = (\tilde{\mathbf{f}}, P_{\mathbf{u}}^{\mathcal{I}})$, where the density of $P_{\mathbf{u}}^{\mathcal{I}}$ is of the form

$$p^{\mathcal{I}}(\mathbf{u}) = \delta\left(\left\{\tilde{f}_i(\mathbf{x}_{\mathrm{pa}_i}, \mathrm{u}_i) = \alpha\right\}\right) \cdot \prod_{j \neq i} p_j(\mathrm{u}_j), \tag{5}$$

with δ being the Dirac delta at the unique value of u_i that yields $\mathrm{x}_i = \alpha$ after applying the causal mechanism $\tilde{f}_i$.

Discrete data To account for discrete data, we use the model considered by Xi and Bloem-Reddy [30] that includes observational noise, and consider a continuous version of the observed discrete variables by adding independent noise $\varepsilon \in [0, 1]$ (e.g., from a standard uniform) to them, such that the real distribution is still recoverable. Intuitively, our approach assumes that discrete variables correspond to the integer part of (noisy) continuous variables generated according to an SCM fulfilling our assumptions, such that both our theoretical and practical insights still apply.

Partial knowledge While we rarely know the entire causal graph $\boldsymbol{G}$, we often have a good grasp on some causal relationships between a subset of observed variables. When only partial knowledge on the graph is available, we can instead work with a modified acyclic graph $\tilde{\boldsymbol{G}}$ obtained by finding the strongly connected components as in [28], where subsets of variables with unknown causal relationships are treated as a block. This allows us to reuse our theoretical results for known parts of the graph, akin to the block identifiability results from von Kügelgen et al. [29].

4 EMPIRICAL EVALUATION

Here, we empirically validate the proposed causal NF, comparing it with previous works, and showing its utility through a real-world use-case. See App. C for more details.

4.1 NON-LINEAR SCMS

Experimental setup We compare our causal NF with two relevant works: i) CAREFL [11], an NF with knowledge on the causal ordering and affine layers; and ii) VACA [26], a variational auto-encoding GNN with knowledge on the graph. For fair comparison, every model uses the same budget for hyperparameter tuning, our causal NF uses affine layers, and CAREFL has been fixed and uses the proposed do-operator from §3.1 (see App. B.3). We consider three non-linear synthetic SCMs: i) TRIANGLE, a 3-node SCM with a dense causal graph; ii) LARGEBD [9], a 9-node SCM with non-Gaussian $P_{\mathbf{u}}$ and made out of two chains with common initial and final nodes; and iii) SIMPSON [9], a 4-node SCM simulating a Simpson's paradox [27].

Table 1: Comparison, on three non-linear SCMs, of the proposed causal NF, VACA [26], and CAREFL [11] with the do-operator proposed in §3.1. Results averaged over five runs.

Dataset	Model	Performance			Time Evaluation (µs)		
		KL	ATE_{RMSE}	CF_{RMSE}	Training	Evaluation	Sampling
TRIANGLE NLIN [26]	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.12_{0.03}}$	$\mathbf{0.13_{0.02}}$	$\mathbf{0.52_{0.07}}$	$\mathbf{0.58_{0.07}}$	$\mathbf{1.07_{0.12}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$\mathbf{0.12_{0.03}}$	$0.17_{0.03}$	$\mathbf{0.57_{0.18}}$	$\mathbf{0.83_{0.26}}$	$\mathbf{1.68_{0.62}}$
	VACA	$7.71_{0.60}$	$4.78_{0.01}$	$4.19_{0.04}$	$28.82_{1.21}$	$23.00_{0.55}$	$70.65_{3.70}$
LARGEBD NLIN [9]	Causal NF	$\mathbf{1.51_{0.04}}$	$\mathbf{0.02_{0.00}}$	$\mathbf{0.01_{0.00}}$	$\mathbf{0.52_{0.10}}$	$\mathbf{0.60_{0.17}}$	$\mathbf{3.05_{0.66}}$
	CAREFL†	$1.51_{0.05}$	$0.05_{0.01}$	$0.08_{0.01}$	$0.84_{0.47}$	$1.18_{0.17}$	$8.25_{1.29}$
	VACA	$53.66_{2.07}$	$0.39_{0.00}$	$0.82_{0.02}$	$164.92_{11.10}$	$137.88_{15.72}$	$167.94_{25.75}$
SIMPSON SYMPROD [9]	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.07_{0.01}}$	$\mathbf{0.12_{0.02}}$	$\mathbf{0.59_{0.17}}$	$\mathbf{0.60_{0.11}}$	$\mathbf{1.51_{0.30}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$0.10_{0.02}$	$0.17_{0.04}$	$\mathbf{0.49_{0.15}}$	$0.81_{0.19}$	$1.91_{0.33}$
	VACA	$13.85_{0.64}$	$0.89_{0.00}$	$1.50_{0.04}$	$49.26_{4.09}$	$37.78_{3.41}$	$79.20_{14.60}$

Table 2: Accuracy, F1-score, and counterfactual unfairness of the audited classifiers over five runs. Causal NFs enable both fair classifiers and accurate unfairness metrics.

		full	unaware	fair $\mathbf{x}$	fair $\mathbf{u}$
Logistic	f1	$72.28_{6.16}$	$72.37_{4.90}$	$59.66_{8.57}$	$73.08_{4.38}$
	accuracy	$67.00_{3.83}$	$66.75_{2.63}$	$54.75_{5.91}$	$66.50_{3.70}$
	unfairness	$5.84_{2.93}$	$2.81_{0.72}$	$0.00_{0.00}$	$0.00_{0.00}$
SVM	f1	$76.04_{2.86}$	$76.80_{5.82}$	$68.28_{5.74}$	$77.39_{1.52}$
	accuracy	$69.50_{3.11}$	$71.00_{3.83}$	$59.25_{2.99}$	$69.75_{1.26}$
	unfairness	$6.65_{2.45}$	$2.78_{0.40}$	$0.00_{0.00}$	$0.00_{0.00}$

Results The results are summarized in Tab 1. In a nutshell: *the proposed causal NF outperforms both CAREFL and VACA in terms of performance and computational efficiency.* VACA shows poor performance, and is considerably slower due to the complexity of GNNs. Our causal NF outperforms CAREFL in counterfactual estimation tasks with identical observational fitting, showing the importance of being causally consistent; and it is quicker than CAREFL as well, since best-performing CAREFL architectures have in general more than one layer.

4.2 USE-CASE: FAIRNESS AUDITING AND CLASSIFICATION

We follow now the use-case of Sánchez-Martín et al. [26] on the German Credit dataset [7] to show the potential impact of causal NFs. Extra details and results appear in App. D.

Experimental setup As proposed by Chiappa [3], we use a partial graph which groups the 7 discrete features of the dataset in 4 different blocks with known causal relationships, putting in practice the results from §3.1. The goal here is to train a causal NF that captures well the underlying SCM, and use it to train and evaluate classifiers that predict the (additional) binary feature *credit risk*, while remaining counterfactually fair w.r.t. the binary variable *sex*, x_S.

In this setting, we call a binary classifier $\kappa : \mathbb{X} \to \{0, 1\}$ counterfactually fair [16] if, for all possible factual values $\mathbf{x}^{\text{f}} \in \mathbb{X}$, the counterfactual unfairness remains zero. That is, if the difference between $P(\kappa(\mathbf{x}^{\text{cf}}) = 1 \,|\, do(\mathrm{x}_S = s), \mathbf{x}^{\text{f}})$ for $s = 0, 1$ is zero on average, where $\mathbf{x}^{\text{cf}}$ is a counterfactual sampled from $P(\mathbf{x}^{\text{cf}} \,|\, do(\mathrm{x}_S = s), \mathbf{x}^{\text{f}})$.

Following Sánchez-Martín et al. [26], we audit: a model that takes all observed variables (*full*); an *unaware* model that leaves the sensitive attribute x_S out; a fair model that only considers non-descendant variables of x_S (*fair* $\mathbf{x}$); and, to demonstrate the ability to learn a counterfactually fair classifier, we include a classifier that takes $\mathbf{u} = T_{\boldsymbol{\theta}}(\mathbf{x})$ as input, but leaves u_S out (*fair* $\mathbf{u}$).

Results Tab 2 summarizes the performance and unfairness of the classifiers, using logistic regression [6] and SVMs [5]. Here, we observe that by taking the non-sensitive exogenous variables from the causal NF, the obtained classifiers achieve comparable or better accuracy than the rest of the classifiers, while at the same time being counterfactually fair. Moreover, the estimations of unfairness obtained with the causal NF match our expectations [16], with *full* being the most unfair, followed by *aware* and the two fair models. With this use-case, we demonstrate that *causal NFs may indeed be a valuable asset for real-world causal inference problems.*

5 CONCLUDING REMARKS

In this work, we have shown that causal NFs are a natural choice to learn a broad class of causal data-generating processes in a principled way. Specifically, we have proven that causal NFs can match the observational distribution of an underlying SCM, and that in doing so the ANF needs to be causally consistent. We have shown that a single-layer ANF model going from $\mathbf{x}$ to $\mathbf{u}$ that uses the causal graph $\boldsymbol{G}$ meets the necessary conditions for approximating the underlying SCM. Moreover, we have provided causal NFs with a do-operator to efficiently solve causal inference tasks. Finally, we have empirically validated our findings and demonstrated that our causal NF framework: i) outperforms competing methods; and ii) can deal with mixed-type data and partial knowledge of the causal graph.

ACKNOWLEDGEMENTS

We would like to thank Batuhan Koyuncu and Jonas Klesen for their invaluable feedback. Adrián Javaloy is funded by DFG grant 389792660 as part of TRR 248 – CPEC, see https://perspicuous-computing.science. Pablo Sánchez Martín also thanks the DFG through the Cluster of Excellence "Machine Learning – New Perspectives for Science", EXC 2064/1, project number 390727645 for generous funding support. The authors also thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Pablo Sánchez Martín.

Bibliography

[1] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[2] Patrick Chao, Patrick Blöbaum, and Shiva Prasad Kasiviswanathan. Interventional and counterfactual inference with diffusion models. *ArXiv preprint*, abs/2302.00860, 2023. Link.

[3] Silvia Chiappa. Path-specific counterfactual fairness. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7801–7808. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33017801. Link.

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. Link.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[6] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[7] Dheeru Dua and Casey Graff. UCI machine learning repository, 2021. Link.

[8] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7509–7520, 2019. Link.

[9] Tomas Geffner, Javier Antorán, Adam Foster, Wenbo Gong, Chao Ma, Emre Kıcıman, Ajay Sharma, A. Lamb, Martin Kukla, Nick Pawlowski, Miltiadis Allamanis, and Cheng Zhang. Deep end-to-end causal inference. *ArXiv preprint*, abs/2202.02195, 2022. Link.

[10] Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. Link.

[11] Ilyes Khemakhem, Ricardo Pio Monti, Robert Leech, and Aapo Hyvärinen. Causal autoregressive flows. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 3520–3528. PMLR, 2021. Link.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. Link.

[13] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. Link.

[14] Herbert Knothe. Contributions to the theory of convex bodies. *Michigan Mathematical Journal*, 4:39–52, 1957.

[15] Murat Kocaoglu, Christopher Snyder, Alexandros G. Dimakis, and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. Link.

[16] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4066–4076, 2017. Link.

[17] Arash Nasr-Esfahany, MohammadIman Alizadeh, and Devavrat Shah. Counterfactual identifiability of bijective causal models. *ArXiv preprint*, abs/2302.02228, 2023. Link.

[18] George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2338–2347, 2017. Link.

[19] George Papamakarios, Eric T. Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22: 57:1–57:64, 2021. Link.

[20] Álvaro Parafita and Jordi Vitrià. Estimand-agnostic causal query estimation with deep causal graphs. *IEEE Access*, 10:71370–71386, 2022. doi: 10.1109/ACCESS.2022.3188395.

[21] Nick Pawlowski, Daniel Coelho de Castro, and Ben Glocker. Deep structural causal models for tractable counterfactual inference. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. Link.

[22] J. Pearl. *Causality*. Cambridge University Press, 2009. ISBN 9781139643986. Link.

[23] Judea Pearl. The do-calculus revisited. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*, pages 3–11. AUAI Press, 2012. Link.

[24] Murray Rosenblatt. Remarks on a multivariate transformation. *The Annals of Mathematical Statistics*, 23 (3):470–472, 1952. ISSN 00034851. Link.

[25] Pedro Sanchez and Sotirios A. Tsaftaris. Diffusion causal models for counterfactual estimation. In *CLEaR*, 2022.

[26] Pablo Sánchez-Martín, Miriam Rateike, and Isabel Valera. Vaca: Design of variational graph autoencoders for interventional and counterfactual queries. *ArXiv preprint*, abs/2110.14690, 2021. Link.

[27] Edward H Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(2):238–241, 1951.

[28] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi: 10.1137/0201010. Link.

[29] Julius von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-supervised learning with data augmentations provably isolates content from style. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 16451–16467, 2021. Link.

[30] Quanhan Xi and Benjamin Bloem-Reddy. Indeterminacy in generative models: Characterization and strong identifiability. 2022.

[31] Kevin Xia, Yushu Pan, and Elias Bareinboim. Neural causal models for counterfactual identification and estimation. *ArXiv preprint*, abs/2210.00035, 2022. Link.

[32] Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. Causalvae: Disentangled representation learning via neural structural causal models. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 9593–9602. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.00947. Link.

[33] Matej Zečević, Devendra Singh Dhami, Petar Veličković, and Kristian Kersting. Relating graph neural networks to structural causal models. *ArXiv preprint*, abs/2109.04173, 2021. Link.

Appendix

Table of Contents

A THEORY OF CAUSAL NORMALIZING FLOWS

A.1 STRUCTURAL EQUIVALENCE

Here, we quickly introduce some extra notation used in the appendix. Specifically, to reason about causal dependencies, we introduce the notion of structural equivalence. We say that two matrices $\boldsymbol{S}$ and $\boldsymbol{R}$ are *structurally equivalent*, denoted $\boldsymbol{S} \equiv \boldsymbol{R}$, if both matrices have zeroes exactly in the same positions. Similarly, we say that $\boldsymbol{S}$ is *structurally sparser* than $\boldsymbol{R}$, denoted as $\boldsymbol{S} \preceq \boldsymbol{R}$, if whenever an element of $\boldsymbol{R}$ is zero, the same element of $\boldsymbol{S}$ is zero.

A.2 IDENTIFIABILITY RESULTS

First, we provide a more detailed explanation on the connection between the results from §3 and those from the work of Xi and Bloem-Reddy [30]. We consider it important to clarify that the definition of identifiability that we use is the same as [30, Def. 2]. Specifically, this definition is one better suited for deep learning models, which is concerned with recovering the variables $\mathbf{u}$ and *one* parametrization that perfectly matches the original generator. In other words, with this definition we aim to recover one parametrization of a neural network which provides the generator function, but not the *exact* parametrization of the generator that generated the data.

We also want to clarify that Thm. 1 from the main paper corresponds to [30, Prop. 5.2], which we rewrote (without changing its content) to plain English and to match our particular setting. We now provide the proof for Cor. 2:

Corollary 3 (Causal consistency)**.** If a causal NF $T_{\boldsymbol{\theta}}$ isolates the exogenous variables of an SCM $\mathcal{M}$, then $\nabla_{\mathbf{x}} T_{\boldsymbol{\theta}}(\mathbf{x}) \equiv \boldsymbol{I} - \boldsymbol{G}$ and $\nabla_{\mathbf{u}} T_{\boldsymbol{\theta}}^{-1}(\mathbf{u}) \equiv \boldsymbol{I} + \sum_{n=1}^{\infty} \boldsymbol{G}^n$, where $\boldsymbol{G}$ is the causal adjacency matrix of $\mathcal{M}$. In other words, $T_{\boldsymbol{\theta}}$ is causally consistent with the true data-generating process, $\mathcal{M}$.

Proof. Assume that we have a flow $T_{\boldsymbol{\theta}}$ that does indeed isolate the exogenous variables, meaning that the i-th output of the flow, $T_{\boldsymbol{\theta}}(\mathbf{x})_i$, is related with the true exogenous variable, u_i, by an invertible function that only depends on it.

As explained in §3), this means that for a variable $\mathbf{u} \sim P_{\mathcal{M}}$, we have that $T_{\boldsymbol{\theta}}(\mathbf{f}(\mathbf{u})) \sim P_{\boldsymbol{\theta}}$ and $T_{\boldsymbol{\theta}}(\mathbf{f}(\mathbf{u})) = \boldsymbol{h}(\mathbf{u}) = (h_1(\mathrm{u}_1), h_2(\mathrm{u}_2), \dots, h_d(\mathrm{u}_d))$.

But we know the true generator, whose i-th exogenous variable is given by $\mathrm{u}_i = f_i^{-1}(\mathbf{x}_{\mathrm{pa}_i}, \mathrm{x}_i)$ (the inverse of f_i w.r.t u_i) and, putting all together,

$$T_{\boldsymbol{\theta}}(\mathbf{x})_i = h_i(\mathrm{u}_i) = h_i(f_i^{-1}(\mathbf{x}_{\mathrm{pa}_i}, \mathrm{x}_i))\,, \tag{6}$$

which is a function of *only* the parents of x_i and x_i itself.

If we call $\mathbf{u} = \mathbf{f}^{-1}(\mathbf{x}) \coloneqq (f_1^{-1}(\mathbf{x}_{\mathrm{pa}_1}, \mathrm{x}_1), f_2^{-1}(\mathbf{x}_{\mathrm{pa}_2}, \mathrm{x}_2), \dots, f_d^{-1}(\mathbf{x}_{\mathrm{pa}_d}, \mathrm{x}_d))$ the inverse of the SCM $\mathcal{M}$ that writes $\mathbf{u}$ as a function of $\mathbf{x}$ (see App. B.1), then it is clear that

$$\nabla_{\mathbf{x}} T_{\boldsymbol{\theta}}(\mathbf{x}) = \nabla_{\mathbf{x}}(\boldsymbol{h} \circ \mathbf{f}^{-1})(\mathbf{x}) = \nabla_{\mathbf{u}} \boldsymbol{h}(\mathbf{u}) \cdot \nabla_{\mathbf{x}} \mathbf{f}^{-1}(\mathbf{x}) = \boldsymbol{D} \cdot \nabla_{\mathbf{x}} \mathbf{f}^{-1}(\mathbf{x}) \equiv \boldsymbol{I} - \boldsymbol{G}\,, \tag{7}$$

where $\boldsymbol{D}$ is a diagonal matrix and $\boldsymbol{G}$ the adjacency matrix of the causal graph induced by $\mathcal{M}$.

Similarly, $T_{\boldsymbol{\theta}}^{-1}(\boldsymbol{h}(\mathbf{u})) = \mathbf{x} = \mathbf{f}(\mathbf{u})$ and $T_{\boldsymbol{\theta}}^{-1}(\boldsymbol{h}(\mathbf{u}))_i = \mathrm{x}_i = f_i(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i)$, which again implies that:

$$\nabla_{\mathbf{u}} T_{\boldsymbol{\theta}}^{-1}(\mathbf{u}) \equiv (\boldsymbol{I} - \boldsymbol{G})^{-1} = \boldsymbol{I} + \sum_{n=1}^{\infty} \boldsymbol{G}^n\,, \tag{8}$$

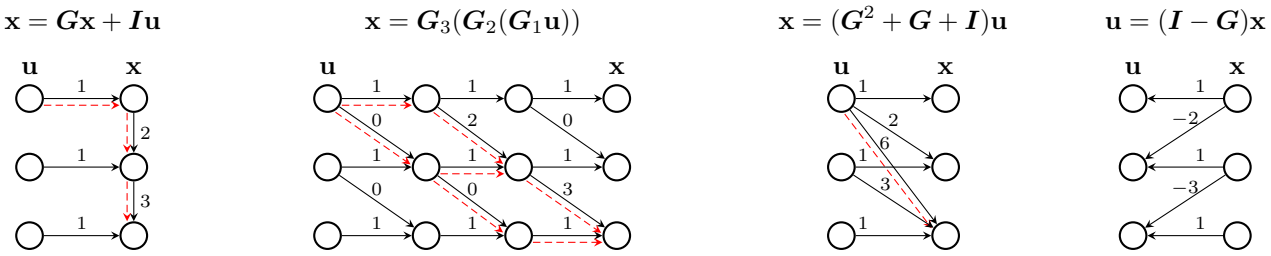
where we have omitted $\boldsymbol{h}$ as its Jacobian matrix is diagonal. Note that the infinite sum above vanishes at $n = \operatorname{diam} \boldsymbol{G}$ since $\boldsymbol{G}$ is triangular with diagonal zero. Q.E.D.

B EXTENSION TO REAL-WORLD SETTINGS

B.1 THE MULTIPLE REPRESENTATIONS OF SCMS

B.1.1 Illustrative example

In this section, we delve a bit deeper into the different equivalent ways of rewriting an SCM through an illustrative example.



(a) Recursive. (b) Unrolled. (c) Compacted. (d) Inverted.

Figure 4: Example of the linear SCM $\{x_1 := u_1 \,;\, x_2 := 2x_1 + u_2 \,;\, x_3 := 3x_2 + u_3\}$ written (a) in its usual recursive formulation; (b) without recursions, with each step made explicit; (c) without recursions, as a single function; and (d) writing $\mathbf{u}$ as a function of $\mathbf{x}$. The red dashed arrows show the influence of u_1 on x_3 for all equations from $\mathbf{u}$ to $\mathbf{x}$, with the compacted version exhibiting shortcuts (see App. B.2). Note that $\boldsymbol{G}_1, \boldsymbol{G}_2, \boldsymbol{G}_3 \preceq \boldsymbol{G} + \boldsymbol{I}$ are any three matrices such that their product equals $\boldsymbol{G}^2 + \boldsymbol{G} + \boldsymbol{I}$.

Recursive SCM As explained in the manuscript, the usual way of describing an SCM $\mathcal{M}$ is by providing its recursive equations. In our example (Fig. 4a), we have a linear SCM of the form

$$\begin{cases} x_1 = u_1 \\ x_2 = 2x_1 + u_2 \\ x_3 = 3x_2 + u_3 \end{cases}, \tag{9}$$

which we can compactly write as $\mathbf{x} = \boldsymbol{G}\mathbf{x} + \boldsymbol{I}\mathbf{u}$. However, the recursive equations are not the most convenient ones, as they entail solving the system iteratively according to its causal dependencies.

Unrolled SCM Instead, we can write the equations as a function from $\mathbf{u}$ to $\mathbf{x}$ directly. To do this, we can proceed and unroll the equations:

$$\begin{cases} x_1 = u_1 \\ x_2 = 2x_1 + u_2 \\ x_3 = 3x_2 + u_3 \end{cases} \Rightarrow \begin{cases} x_1 = u_1 \\ x_2 = 2u_1 + u_2 \\ x_3 = 3(2x_1 + u_2) + u_3 \end{cases} \Rightarrow \begin{cases} x_1 = u_1 \\ x_2 = 2u_1 + u_2 \\ x_3 = 3(2u_1 + u_2) + u_3 \end{cases}, \tag{10}$$

which we can write as a multi-step function:

$$\begin{cases} z_1^1 = u_1 \\ z_2^1 = u_2 \\ z_3^1 = u_3 \end{cases} \Rightarrow \begin{cases} z_1^2 = z_1^1 \\ z_2^2 = 2z_1^1 + z_2^1 \\ z_3^2 = z_3^1 \end{cases} \Rightarrow \begin{cases} x_1 = z_1^2 \\ x_2 = z_2^2 \\ x_3 = 3z_2^2 + z_3^2 \end{cases}, \tag{11}$$

that we can once again compactly write as a series of linear operations $\mathbf{x} = \boldsymbol{G}_3(\boldsymbol{G}_2(\boldsymbol{G}_1\mathbf{u}))$. Note that the matrices $\boldsymbol{G}_1, \boldsymbol{G}_2, \boldsymbol{G}_3$ are not unique, and that they are valid as long as they are at most as sparse as $\boldsymbol{G}$, and produce the same final output.

Compacted SCM Another natural step here is to compress this sequence of linear equations into a single operation. That is, use directly the linear operation described at the end of Eq. 10:

$$\begin{cases} x_1 = u_1 \\ x_2 = 2u_1 + u_2 \\ x_3 = 6u_1 + 3u_2 + u_3 \end{cases}. \tag{12}$$

And we can derive the same result in vectorial form:

$$\mathbf{x} = \boldsymbol{G}\mathbf{x} + \boldsymbol{I}\mathbf{u} \Rightarrow \mathbf{x} = \boldsymbol{G}(\boldsymbol{G}\mathbf{x} + \boldsymbol{I}\mathbf{u}) + \boldsymbol{I}\mathbf{u} \Rightarrow \mathbf{x} = \boldsymbol{G}(\boldsymbol{G}(\boldsymbol{G}\mathbf{x} + \boldsymbol{I}\mathbf{u}) + \boldsymbol{I}\mathbf{u}) + \boldsymbol{I}\mathbf{u} \Rightarrow$$
$$\mathbf{x} = \cancelto{0}{\boldsymbol{G}^3}\mathbf{x} + \boldsymbol{G}^2\mathbf{u} + \boldsymbol{G}\mathbf{u} + \boldsymbol{I}\mathbf{u} = (\boldsymbol{G}^2 + \boldsymbol{G} + \boldsymbol{I})\mathbf{u}\,.$$

Unfortunately, in this form we cannot longer distinguish indirect paths: we have collapsed all paths into direct paths. Even worse, if there were more than one path between two given nodes, we have combined their contributions into a single path, making it quite difficult to disentangle. This effect can be seen as analogous to the common process in cryptography for sharing secrets: if you have two primes (paths), it is fairly easy to multiply them and obtain their product, but if you have their product (collapsed paths), performing prime factorization of the number is prohibitive.

Inverted SCM Finally, we can take any of these different representations and invert the equations to go from $\mathbf{x}$ to $\mathbf{u}$. In this case, it is easier to work with the original equations:

$$\begin{cases} \mathrm{x}_1 = \mathrm{u}_1 \\ \mathrm{x}_2 = 2\mathrm{x}_1 + \mathrm{u}_2 \\ \mathrm{x}_3 = 3\mathrm{x}_2 + \mathrm{u}_3 \end{cases} \Rightarrow \begin{cases} \mathrm{u}_1 = \mathrm{x}_1 \\ \mathrm{u}_2 = \mathrm{x}_2 - 2\mathrm{x}_1 \\ \mathrm{u}_3 = \mathrm{x}_3 - 3\mathrm{x}_2 \end{cases}, \tag{13}$$

and, in vectorial form:

$$\mathbf{x} = \boldsymbol{G}\mathbf{x} + \boldsymbol{I}\mathbf{u} \Rightarrow \mathbf{u} = (\boldsymbol{I} - \boldsymbol{G})\mathbf{x}\,. \tag{14}$$

As discussed in the main manuscript, this turns out to be a really convenient SCM representation to work with, as: i) we can obtain the exogenous variables in one go; and ii) even with a single layer, all indirect paths are preserved.

B.1.2 Non-linear SCM representations

We now discuss how the same representation and reasoning about the causal relationships of a linear SCM from App. B.1.1 can be translated to the general case. To this end, assume that we have a non-linear SCM $\mathcal{M}$ of the form $\mathbf{x} = \tilde{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ where, to ease the reader, imagine that it has the same causal graph as the linear example, so that the reader can use Fig. 4 as a reference again, i.e., assume that

$$\left(\nabla_{\mathbf{x}}\tilde{\mathbf{f}}(\mathbf{x},\mathbf{u}) \neq \mathbf{0}\right) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \left(\nabla_{\mathbf{u}}\tilde{\mathbf{f}}(\mathbf{x},\mathbf{u}) \neq \mathbf{0}\right) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{15}$$

where $\mathbf{0}$ is the constant zero function with the same domain and codomain as the Jacobian matrices.

Recursive SCM Already extensively discussed. This is the representation an SCM is given as.

Unrolled SCM Just as before, we can unroll the equations by having multiple functions $\mathbf{z}^l = \mathbf{f}_l(\mathbf{z}^{l-1})$, and in each one we unroll those equations for which we already know the non-recursive equation of its parents, leaving all the other fixed (identity functions). As before, we can write these multiple layers in different ways, as long as they produce the same final function (after composing them, $\mathbf{f}_1 \circ \mathbf{f}_2 \circ \cdots \circ \mathbf{f}_L = \mathbf{f}$), and that they respect the causal dependencies provided by $\boldsymbol{I} + \boldsymbol{G}$.

Collapsed SCM Just as before, we can expand all the different layers and write a (probably complex) formula that encompasses all changes in a single step. It is easy to show that the composition of these functions has, in general, a Jacobian matrix structurally equivalent to $\boldsymbol{I} + \sum_l^\infty \boldsymbol{G}^l$. Specifically, their composition will be of the form $\prod_l(\boldsymbol{I} + \boldsymbol{G})$.

Reverse SCM Since we assume that each $\tilde{f}_i(\mathbf{x}_{\mathrm{pa}_i}, \mathrm{u}_i)$ is bijective with respect to u_i, we can always compute its inverse to obtain u_i as a function of the observed values, $\mathrm{u}_i = \tilde{f}_i^{-1}(\mathbf{x}_{\mathrm{pa}_i}, \mathrm{x}_i)$. Clearly, its Jacobian matrix will be structurally equivalent to $\boldsymbol{I} + \boldsymbol{G} \equiv \boldsymbol{I} - \boldsymbol{G}$.

Therefore, we can always reason as we did with the linear case, but using Jacobian matrices to talk about causal dependencies between variables, and possibly having a complex and/or non-closed formulation of the generative/abductive functions.

B.2 EFFECTIVE DESIGN OF CAUSAL NORMALIZING FLOWS

We showed in §3 that causal NFs are a natural choice to learn the underlying SCM generating the data, yet reaching the optimal parameters may be tricky in practice. In this section, we expand on the proposed model to guide the optimization towards solutions that do not only provide an accurate fit of the observational distribution, but allow us to also accurately answer to interventional and counterfactual queries.

Let us start with an illustrative example. Suppose that we are given the linear SCM in Fig. 4a, and we want to write the SCM equations as a TMI map to approximate them with a causal NF. As discussed in App. B.1.1, we can unroll the causal equations (Fig. 4b)—resulting in a composition of functions structurally sparser than $\boldsymbol{I} + \boldsymbol{G}$. These functions can be compacted into a single transformation (Fig. 4c), such that each x_i depends on its ancestors, $\mathbf{u}_{\mathrm{an}_i}$. However, note that in this step *shortcuts* appear, making direct and indirect causal paths in this representation indistinguishable—in our example, the indirect causal path from u_1 to x_3 present in Fig. 4a and Fig. 4b does not go anymore through the path that generates x_2, but instead via a *shortcut* that directly connects u_1 to x_3. Alternatively, we can invert the equations to write $\mathbf{u}$ as a function of $\mathbf{x}$ (Fig. 4d), which is structurally equivalent to $\boldsymbol{I} - \boldsymbol{G}$.

We remark that the above steps can be applied to any considered acyclic SCM (refer to App. B.1.2 for a more detailed discussion). In particular, we can unroll the equations in a finite number of steps, and we can similarly reason about the causal dependencies through the Jacobian matrices of the generators, $\nabla_{\mathbf{x}} T_{\boldsymbol{\theta}}(\mathbf{x})$ and $\nabla_{\mathbf{u}} T_{\boldsymbol{\theta}}^{-1}(\mathbf{u})$. Moreover, note that the diffeomorphic assumption implies that we can invert the causal equations. Next, inspired by the previous example, we consider the following design for a causal NF:

Abductive model Reminiscent to the abduction step [23], a natural choice is to model the inverse equations of the SCM as in Fig. 4d, hence building a causal NF from $\mathbf{x}$ to $\mathbf{u}$ with a single layer. Under a known causal graph, we replicate the structural sparsity by adequately masking the flow with $\boldsymbol{I} + \boldsymbol{G}$, such that

$$\mathrm{u}_i = \tau_i(\mathrm{x}_i; \mathbf{h}_i)\,, \quad \text{where} \quad \mathbf{h}_i = c_i(\mathbf{x}_{\mathrm{pa}_i})\,. \tag{16}$$

Remarkably, this architecture is capable of capturing all indirect dependencies of $\mathbf{u}$ on $\mathbf{x}$ with no shortcuts at the optima, even with a single layer. This is a result of the autoregressive nature of the ANFs used here to build causal NFs, as they compute the inverse sequentially. In the example of Fig. 4, the indirect influence of u_1 on x_3 via x_2 has to necessarily generate x_2 first (Fig. 4a). In contrast, if we only know the causal ordering, then the causal NF will need to rule out the spurious correlations by learning during training the necessary zeroes to fulfil causal consistency (Cor. 3).

As stated at the beginning of the section, we aim to guide the model towards its global optima. To this end, we now analyse the necessary conditions for the design of a causal NF to be able to accurately approximate and manipulate an SCM.

Expressiveness The least restrictive condition is that the causal NF should be able to reach the optima and a single ANF layer (Eq. 2) is a universal TMI approximator [19].

Causal consistency As stated in Cor. 3, the causal NF needs to share the causal dependencies of the SCM at the optima, meaning that their Jacobian matrices need to be structurally equivalent, i.e., $\nabla_{\mathbf{x}} T_{\boldsymbol{\theta}}(\mathbf{x}) \equiv \boldsymbol{I} - \boldsymbol{G}$ (Fig. 4d), and $\nabla_{\mathbf{u}} T_{\boldsymbol{\theta}}^{-1}(\mathbf{u}) \equiv \sum_{n=1}^{\infty} \boldsymbol{G}^n + \boldsymbol{I}$ (Fig. 4c). Given the (partial) causal graph $\boldsymbol{G}$, the abductive model in Eq. 16 ensures causal consistency when $L = 1$.

Causal path preservation In order to perform interventions with a causal NF (see App. B.3), the NF does not only need to be causally consistent, but the causal dependencies from $\mathbf{u}$ to $\mathbf{x}$ also need to follow the same paths as the true causal model. That is, the causal NF needs to be *causal path preserving*. If we impose a causal ordering (e.g., using an ANF), longer-than-needed dependencies are impossible, e.g., u_1 cannot influence x_2 through x_3 in Fig. 4b due to the network structure. We are left with shortcuts as the only possible deviation from the causal paths. As stated before, the abductive model (Eq. 16) avoid shortcuts w.r.t. the given $\boldsymbol{G}$ by carefully controlling how the information flows, being hence well-suited for causal inference tasks.

B.3 DO-OPERATOR: INTERVENTIONS AND COUNTERFACTUALS

B.3.1 Definition and algorithms

In this section, we extend on the do-operator implementation described in §3.1, and provide the step-by-step algorithms to perform interventions (Alg. 1) and compute counterfactuals (Alg. 2).

Semantics Recalling §3.1, the *do-operator* [23], denoted as $do(\mathrm{x}_i = \alpha)$, is defined as a mathematical operator that simulates a physical intervention on an SCM $\mathcal{M}$, inducing an alternative model $\mathcal{M}^{\mathcal{I}}$ that fixes the observational value $\mathrm{x}_i = \alpha$, and thus removes any causal dependency on x_i. However, the definition does not describe the specifics on how to implement such an operation.

Usual implementation Traditionally, we are given the recursive representation of an SCM (Fig. 4a), as discussed in App. B.1. As such, the do-operator $do(\mathrm{x}_i = \alpha)$ is usually carried out by replacing the i-th equation, i.e., the i-th component of $\tilde{\mathbf{f}}$, with a constant function. That is, by doing $\tilde{f}_i^{\mathcal{I}} := \alpha$. This yields an intervened SCM $\mathcal{M}^{\mathcal{I}} = (\tilde{\mathbf{f}}^{\mathcal{I}}, P_{\mathbf{u}})$ reflecting the data-generating process after such an intervention. Unfortunately, this implementation of the do-operator is quite specific to the recursive representation of the SCM (Fig. 4a), and does not translate well to the other equivalent representations discussed in §3. The reason for this is that these representations compute the observational values $\mathbf{x}$ as a vector function of $\mathbf{u}$, without the iterative sampling process that goes through the intervened value that we replace.

Proposed implementation As discussed in §3, we instead propose to manipulate the SCM by modifying the exogenous distribution $P_{\mathbf{u}}$, while keeping the causal generator $\tilde{\mathbf{f}}$ untouched. Specifically, an intervention $do(\mathrm{x}_i = \alpha)$ updates $P_{\mathbf{u}}$, to have positive density mass on only those values that, when transformed to endogenous variables, the intervened variable yields the intervened value, $\mathrm{x}_i = \alpha$, while keeping the rest of distributions unaltered.

That is, we define the intervened SCM as $\mathcal{M}^{\mathcal{I}} = (\tilde{\mathbf{f}}, P_{\mathbf{u}}^{\mathcal{I}})$, where the density of the updated distribution $P_{\mathbf{u}}^{\mathcal{I}}$ is of the form

$$p^{\mathcal{I}}(\mathbf{u}) \propto p(\mathbf{u}) \cdot \delta_{\left\{\tilde{\mathbf{f}}_i(\boldsymbol{x},\mathbf{u})=\alpha\right\}}(\mathbf{u})\,, \tag{17}$$

and where the distributions of the rest of variables remain the same. Using the acyclic assumption, we know that the only way of altering the value of x_i without altering those of its parents is through u_i and, using the causal sufficiency assumption, we can squeeze the Dirac delta directly in the distribution of the i-th exogenous variable, such that:

$$p^{\mathcal{I}}(\mathbf{u}) = p_i^{\mathcal{I}}(\mathrm{u}_i|\mathbf{u}_{j\neq i}) \cdot \prod_{j\neq i} p_j(\mathrm{u}_j)\,, \qquad \text{with} \qquad p_i^{\mathcal{I}}(\mathrm{u}_i|\mathbf{u}_{j\neq i}) \propto p_i(\mathrm{u}_i) \cdot \delta_{\left\{\tilde{\mathbf{f}}_i(\boldsymbol{x},\mathbf{u})=\alpha\right\}}(\mathbf{u})\,. \tag{18}$$

In the case we consider, where all the generators are bijective given the parent nodes, the set $\delta_{\left\{\tilde{\mathbf{f}}_i(\boldsymbol{x},\mathbf{u})=\alpha\right\}}(\mathbf{u})$ contains a single element, and therefore in the main paper we simply write the i-th density as $p_i^{\mathcal{I}}(\mathrm{u}_i|\mathbf{u}_{j\neq i}) = \delta_{\left\{\tilde{\mathbf{f}}_i(\boldsymbol{x},\mathbf{u})=\alpha\right\}}(\mathbf{u})$. Note, as discussed in the main paper, that the density at this point should be positive, in other words, the element that yields α (and therefore α) should be a plausible value.

Since this implementation does not make any assumption at all in the functional form of the generator, but directly works on the distribution of the exogenous variables, it can be implemented on any SCM representation (see Fig. 4 in App. B.1.1). Notice, however, that in order to properly work, that the data-generating process should be causally consistent (i.e., it isolates $\mathbf{u}$) and causal path-preserving (i.e., without shortcuts) with respect to the original SCM.

Algorithms The step-by-step algorithms to perform interventions and compute counterfactuals, using the described algorithm, are presented in Alg. 1 and Alg. 2, respectively. The only difference between both algorithms is the way that we obtain samples from the observed distribution (generated vs. given).

Algorithm 1 Algorithm to sample from the interventional distribution, $P(\mathbf{x} \mid do(\mathrm{x}_i = \alpha))$.

function SAMPLEINTERVENEDDIST(i, α)
  $\mathbf{u} \sim P_{\mathbf{u}}$
  $\mathbf{x} \leftarrow T_{\boldsymbol{\theta}}^{-1}(\mathbf{u})$ ▷ Sample a value from the observational distribution.
  $\mathrm{x}_i \leftarrow \alpha$ ▷ Set x_i to the intervened value α.
  $\mathrm{u}_i \leftarrow T_{\boldsymbol{\theta}}(\mathbf{x})_i$ ▷ Change the i-th value of $\mathbf{u}$.
  $\mathbf{x} \leftarrow T_{\boldsymbol{\theta}}^{-1}(\mathbf{u})$
  return $\mathbf{x}$ ▷ Return the intervened sample.
end function

Theoretical results Here, we briefly discuss why this implementation works, i.e., why the proposed implementation removes every dependency from the descendants with respect to the ancestors that go through the intervened value, as it is not directly obvious.

To see why, take the usual recursive representation of an SCM in the illustrative example from App. B.1.1 (Fig. 4a), and assume that we do $do(\mathrm{x}_2 = \alpha)$, where $\mathrm{x}_2 = 2\mathrm{x}_1 + \mathrm{u}_2$ in this example. By updating the density $p_2(\mathrm{u}_2)$, we have basically fixed the value of u_2 to be the only one that keeps $\mathrm{x}_2 = \alpha$ given x_1, i.e., $\mathrm{u}_2 = \alpha - 2\mathrm{x}_1$ (this can be clearly seen in Fig. 4d).

Algorithm 2 Algorithm to sample from the counterfactual distribution, $P(\mathbf{x}^{\text{cf}} \mid do(\mathrm{x}_i = \alpha), \mathbf{x}^{\text{f}})$.

function GETCOUNTERFACTUAL($\mathbf{x}^{\text{f}}$, i, α)
  $\mathbf{u} \leftarrow T_{\boldsymbol{\theta}}(\mathbf{x}^{\text{f}})$ ▷ Get $\mathbf{u}$ from the factual sample.
  $\mathrm{x}_i^{\text{f}} \leftarrow \alpha$ ▷ Set x_i to the intervened value α.
  $\mathrm{u}_i \leftarrow T_{\boldsymbol{\theta}}(\mathbf{x}^{\text{f}})_i$ ▷ Change the i-th value of $\mathbf{u}$.
  $\mathbf{x}^{\text{cf}} \leftarrow T_{\boldsymbol{\theta}}^{-1}(\mathbf{u})$
  return $\mathbf{x}^{\text{cf}}$ ▷ Return the counterfactual value.
end function

If we now compute the dependency of x_3 on x_1, we get

$$\frac{\mathrm{d}\,\mathrm{x}_3}{\mathrm{d}\,\mathrm{x}_1} = \frac{\partial \mathrm{x}_3}{\partial \mathrm{x}_2}\frac{\mathrm{d}\,\mathrm{x}_2}{\mathrm{d}\,\mathrm{x}_1} = \frac{\partial \mathrm{x}_3}{\partial \mathrm{x}_2}\left(\frac{\partial \mathrm{x}_2}{\partial \mathrm{x}_1} + \frac{\partial \mathrm{x}_2}{\partial \mathrm{u}_2}\frac{\mathrm{d}\,\mathrm{u}_2}{\mathrm{d}\,\mathrm{x}_1}\right) = \frac{\partial \mathrm{x}_3}{\partial \mathrm{x}_2}\left(2 + 1 \cdot (-2)\right) = 0\,. \tag{19}$$

In layman's terms, the value of u_2 is chosen such that it fixes the value of α, countering any influence that the parents could have on x_2 (or any of its intermediate values), and consequently in any of its descendants.

The general case can be similarly proven. Suppose that we do $do(\mathrm{x}_2 = \alpha)$, and that we want to compute the indirect influence of an ancestor, x_1, on a descendant, x_3, passing through x_2. Since we are fixing the value of u_2 (the input of the network) to produce an observed value x_2 (the output of the network) of α, we can use implicit differentiation to compute the influence of u_1 (and therefore x_1) on x_2 via u_2:

$$\alpha = \mathrm{x}_2(\mathrm{u}_1, \mathrm{u}_2) \xRightarrow{\mathrm{d}\,\mathrm{u}_1} 0 = \frac{\partial \mathrm{x}_2}{\partial \mathrm{u}_1} + \frac{\partial \mathrm{x}_2}{\partial \mathrm{u}_2}\frac{\mathrm{d}\,\mathrm{u}_2}{\mathrm{d}\,\mathrm{u}_1} \Rightarrow \frac{\partial \mathrm{x}_2}{\partial \mathrm{u}_2}\frac{\mathrm{d}\,\mathrm{u}_2}{\mathrm{d}\,\mathrm{u}_1} = -\frac{\partial \mathrm{x}_2}{\partial \mathrm{u}_1}\,, \tag{20}$$

and, similar to Eq. 19, any *indirect* influence of the ancestor, u_1, on the descendant, x_3, through this intermediate variable, x_2, cancels out:

$$\frac{\partial \mathrm{x}_3}{\partial \mathrm{x}_2}\frac{\mathrm{d}\,\mathrm{x}_2}{\mathrm{d}\,\mathrm{u}_1} = \frac{\partial \mathrm{x}_3}{\partial \mathrm{x}_2}\left(\frac{\partial \mathrm{x}_2}{\partial \mathrm{u}_1} + \frac{\partial \mathrm{x}_2}{\partial \mathrm{u}_2}\frac{\mathrm{d}\,\mathrm{u}_2}{\mathrm{d}\,\mathrm{u}_1}\right) = \frac{\partial \mathrm{x}_3}{\partial \mathrm{x}_2} \cdot 0 = 0\,. \tag{21}$$

We have proven this result not only theoretically, but also empirically through the accurate interventions computed for all the experiments from §4 and App. C. Moreover, this lack of correlation between ancestors and descendants through the intervened variables is clearly shown in pair plots such as the ones in Figs. 9 and 10.

B.3.2 Interventions in previous works

We now put our implementation of the do-operator (see §3.1 and App. B.3) into context, by describing how the methods compared in §4, namely CAREFL [11] and VACA [26], proposed to perform interventions with their models.

CAREFL [11] Two different algorithms were proposed to sample from an interventional distribution in CAREFL: i) a sequential algorithm which mimics the usual implementation of the do-operator with the recursive representation of the SCM; and ii) a parallel algorithm that samples the counterfactual in a single call. While the first algorithm works, the parallel one—which is the one actually implemented—only works when intervening on root nodes.

This second algorithm for $do(\mathrm{x}_2 = \alpha)$ is described as follows (see Alg. 2 in [11]): i) sample $\mathbf{u}$ from $P_{\mathbf{u}}$; ii) set u_i to the i-th value obtained by applying the flow, $T_{\boldsymbol{\theta}}$, to an observation with $\mathrm{x}_i = \alpha$ and $\mathrm{x}_j = 0$ for $j \neq i$; and iii) return the value obtained by $T_{\boldsymbol{\theta}}^{-1}$ with $\mathbf{u}$.

While this algorithm resembles the one we proposed, the proposed method does not have into account that the value of u_i to fix α *does depend* on the observed values of its parents, which is clear by looking at the linear illustrative example from Fig. 4d. As a consequence, the algorithm only works when the node has no parents, which is why we replaced it by the one we proposed for the comparisons in §4 and App. C.

VACA [26] Based on GNNs, the approach for intervening on VACA is completely reminiscent to the traditional implementation. Specifically, the authors propose to sever those edges in every layer of the GNN whose endpoints fall in the path generating the intervened variable, so that the ancestors have no way to influence it by design.

While the previous statement is true: ancestors cannot influence the intervened variable nor its descendants, here we argue that this process would require us to "recalibrate" the model, as the middle computations after an intervention change in more complex ways than removing the ancestors from the equation, while keeping the rest unchanged.

To see this, consider the following non-linear triangle SCM:

$$\begin{cases} \mathrm{x}_1 = \mathrm{u}_1 \\ \mathrm{x}_2 = \mathrm{x}_1^2 \mathrm{u}_2 \\ \mathrm{x}_3 = 2\mathrm{x}_1 + \frac{\mathrm{x}_2}{\mathrm{x}_1} + \frac{\mathrm{x}_2}{\mathrm{x}_1^2} + \mathrm{u}_3 \end{cases} \tag{22}$$

which VACA could learn with two layers through the following operations:

$$\begin{cases} \mathrm{z}_1 = \mathrm{u}_1 \\ \mathrm{z}_2 = \mathrm{u}_1 \mathrm{u}_2 \\ \mathrm{z}_3 = \mathrm{u}_1 + \mathrm{u}_2 + \mathrm{u}_3 \end{cases} \qquad \begin{cases} \mathrm{x}_1 = \mathrm{z}_1 \\ \mathrm{x}_2 = \mathrm{z}_1 \mathrm{z}_2 \\ \mathrm{x}_3 = \mathrm{z}_1 + \mathrm{z}_2 + \mathrm{z}_3 \end{cases} . \tag{23}$$

Now, if we were to intervene with $do(\mathrm{x}_2 = \alpha)$, the real SCM would yield:

$$\begin{cases} \mathrm{x}_1 = \mathrm{u}_1 \\ \mathrm{x}_2 = \alpha \\ \mathrm{x}_3 = 2\mathrm{x}_1 + \frac{\alpha}{\mathrm{x}_1} + \frac{\alpha}{\mathrm{x}_1^2} + \mathrm{u}_3 \end{cases} \tag{24}$$

while VACA would yield:

$$\begin{cases} \mathrm{z}_1 = \mathrm{u}_1 \\ \mathrm{z}_2 = \alpha \\ \mathrm{z}_3 = \mathrm{u}_1 + \alpha + \mathrm{u}_3 \end{cases} \qquad \begin{cases} \mathrm{x}_1 = \mathrm{z}_1 \\ \mathrm{x}_2 = \alpha \\ \mathrm{x}_3 = \mathrm{z}_1 + \alpha + \mathrm{z}_3 \end{cases} \Rightarrow \begin{cases} \mathrm{x}_1 = \mathrm{u}_1 \\ \mathrm{x}_2 = \alpha \\ \mathrm{x}_3 = 2\mathrm{x}_1 + 2\alpha + \mathrm{u}_3 \end{cases} , \tag{25}$$

where we can clearly see that the expression for x_3 is different. In contrast, our causal NF would keep the generator as it is, and set u_2 to α/x_1^2, yielding the correct value.

B.4 DISCRETE DATA

In this section, we describe how to extend the results presented in the main text for the case where one observed variable, x_i, is discrete. To this end, we restate the more general data-generating process assumed by Xi and Bloem-Reddy [30], which we used for the theoretical part of the manuscript.

Following the notation of the manuscript, say that we have a data-generating process without recursions, that is, we have a function $\mathbf{f}$ that maps $\mathbf{u}$ to $\mathbf{x}$. Let us assume, without loss of generality, that only the i-th observed variable is discrete, and let us focus on the way this variable is generated, dropping the subindex i along the way to avoid clutter. Now, Xi and Bloem-Reddy [30] additionally consider the existence of a fixed noise distribution P_ε and mechanism g, such that the observed variable x_i is generated as,

$$\mathbf{u} := (\mathrm{u}_1, \mathrm{u}_2, \dots, \mathrm{u}_d) \sim P_{\mathbf{u}} , \qquad \varepsilon \sim P_\varepsilon , \qquad \mathrm{x}_i = g(f_i(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i), \varepsilon) , \tag{26}$$

with $\varepsilon \perp\!\!\!\perp \mathbf{u}$, and where they study the noiseless case under the following assumption: if for two generative processes with $\varepsilon_a \overset{\mathrm{d}}{=} \varepsilon_b$, then $g(f_i(\mathbf{u}_a), \varepsilon_a) \overset{\mathrm{d}}{=} g(f_i(\mathbf{u}_b), \varepsilon_b)$ if and only if $f_i(\mathbf{u}_a) \overset{\mathrm{d}}{=} f_i(\mathbf{u}_b)$, where $\overset{\mathrm{d}}{=}$ denotes equal in distribution.

Just as we do with the rest of variables, we also make the assumption that the observed variable $\tilde{\mathrm{x}}_i$ is the transformation of a continuous exogenous variable, u_i, with a function $\tilde{f}_i$ that fulfils our assumptions (i.e., that $\tilde{f}_i$ is a diffeomorphism) that has undergone a quantization process, i.e., $\mathrm{x}_i = f_i(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i) := \lfloor \tilde{f}_i(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i) \rfloor$. Therefore, it is clear that f_i is no longer bijective, as we are clamping real numbers into integers, and that the observational distribution of x_i is discrete.

We take advantage of the noise assumption above, and dequantize the observed variable x_i by assuming an additive noise mechanism such that $\mathrm{x}_i := f_i(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i) + \varepsilon$, with ε distributed between the unit interval with any continuous distribution (we take in our experiments $P_\varepsilon = \mathcal{U}(0, 1)$). With this process: i) we have made $\tilde{\mathrm{x}}_i$ again a continuous random variable, as

the sum of independents discrete and continuous random variables is a continuous random variable; and ii) the original distribution of the noiseless observed variables is always recoverable $P(\tilde{\mathrm{x}}_i = c) = P(c \leq \mathrm{x}_i \leq c + 1)$.

More importantly, all the theoretical insights from the work of Xi and Bloem-Reddy [30] can still be used, working with the noisy case rather than the noiseless one. Indeed, as for their analysis they assume a single $\mathbf{u}$ in the domain of the generator, we can merge the generator and noise mechanism $g \circ f_i : \mathbb{R} \to \mathbb{R}$ (rather than $g \circ f_i : \mathbb{R} \times [0, 1] \to \mathbb{R}$), by mapping the non-injective part of $\mathbf{u}$ to ε itself, i.e., by using the function $(g \circ f_i)(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i) = \lfloor \tilde{f}_i(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i) \rfloor + F_\varepsilon^{-1}(\tilde{f}_i(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i) - \lfloor \tilde{f}_i(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i) \rfloor)$, where F_ε^{-1} is the quantile function of P_ε. This new function is a diffeomorphism almost everywhere, as it is a composition of a.e. diffeomorphisms, and we have effectively replaced the noise variable by the floating part of $\tilde{f}_i(\mathbf{u}_{\mathrm{an}_i}, \mathrm{u}_i)$ before quantization. Moreover, note that if the noise is uniformly distributed, $P_\varepsilon = \mathcal{U}(0, 1)$, we have that $g \circ f_i = \tilde{f}_i$ (if x_i were discretized by taking its integer part).

In short, by adding noise to discrete variables while keeping them recoverable, we can learn a mapping between continuous variables that learns a version of the generator function before the observed values were somehow discretized. Importantly, the observed discrete distribution is always recoverable, independently of whether we learn the (unknown and unrecoverable) underlying continuous distribution before being discretized.

B.5 PARTIAL KNOWLEDGE

In this section, we explain how to expand our framework to settings in which we have partial information about the causal graph of $\mathcal{M}$. That is, we know the causal ordering π (so we know that half of the causal relationships, the upper diagonal), and we are certain about some other causal relationships (edges on $\boldsymbol{G}$), but not all of them.

To this end, first let us first introduce the way we deal with partial knowledge, and then clarify the theoretical implications that it has with respect to the theory introduced in §3.1.

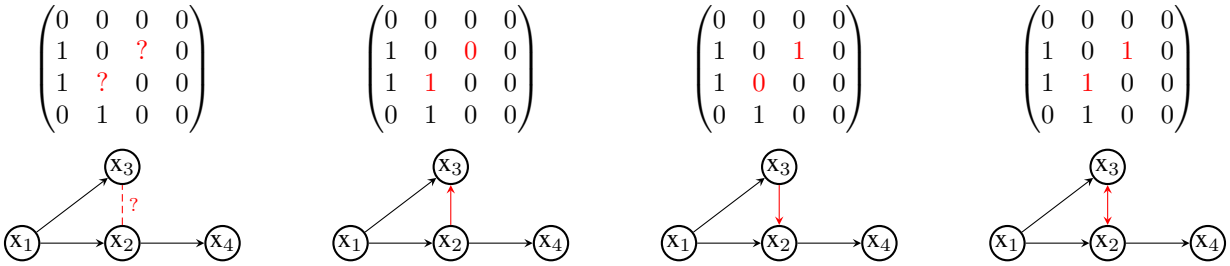
The method Let us motivate the method with an illustrative example. Suppose that we are given an SCM such as the one in Fig. 5a, where we know all relationships but the one between x_2 and x_3. Note that, in this case, we lack even information about the causal ordering. Indeed, there are three possible outcomes: i) the edge $\mathrm{x}_2 \to \mathrm{x}_3$ could exist (Fig. 5b); ii) the edge $\mathrm{x}_3 \to \mathrm{x}_2$ could exist (Fig. 5c); or iii) both could exist simultaneously (Fig. 5d), and hence there is a confounder between them. However, we do not know which of the three options is the correct one.

Let us switch now to Fig. 6. To solve the original problem (Fig. 6a, one natural approach is to group the nodes with unknown relationships—assuming that all unknown edges may exist—and maximize the observed likelihood (Fig. 6b). This, effectively, is equivalent to applying an ANF to the known relationships, and using a general-Jacobian NF to learn the joint of the block variables. However, if we want to keep using exclusively ANFs (Fig. 6c), we can learn the joint distribution within the blocks with an ANF using a fixed ordering (which it can always do, as it is a universal density approximator [19]). The only subtle detail here is that, in that case, we need to increase the granularity of all inter-block edges from node- to block-wise relationships, assuming that an edge exists if it exists for at least one of the elements of the block. To see why this is necessary, assume that the real graph of the example is Fig. 5c, yet we use an ANF with the ordering $\pi = (1\ 2\ 3\ 4)$ and the graph $\boldsymbol{G}$ without inter-block modifications (i.e., the adjacency matrix of Fig. 5b). In that case, we would have that x_4 depends on x_3 through x_2. However, a causally consistent NF w.r.t. $\boldsymbol{G}$ would not be able to model that dependency, and thus x_4 would depend on u_1, u_2, and u_4 but not on u_3. With this approach, the ANF can model every case from Fig. 5, and it would need to remove the extra spurious relationships through optimization.

Therefore, to reuse our existing results from §3.1, the method that we have adopted is the one from Fig. 6c, which can be described in the following steps: i) run Tarjan's algorithm [28] to group all nodes by their SCCs (note that, unlike in the given example, there could be more than one cluster of unknown relationships); ii) choose an ordering that is consistent with the known inter-SCC edges, fixing the edges within the SCCs; iii) move from node-edges to SCC-edges. In practice, this means introducing edges between every pair of edges of two SCCs, if there exists at least one edge between them; and iv) solve the MLE problem with an ANF as in the main manuscript.

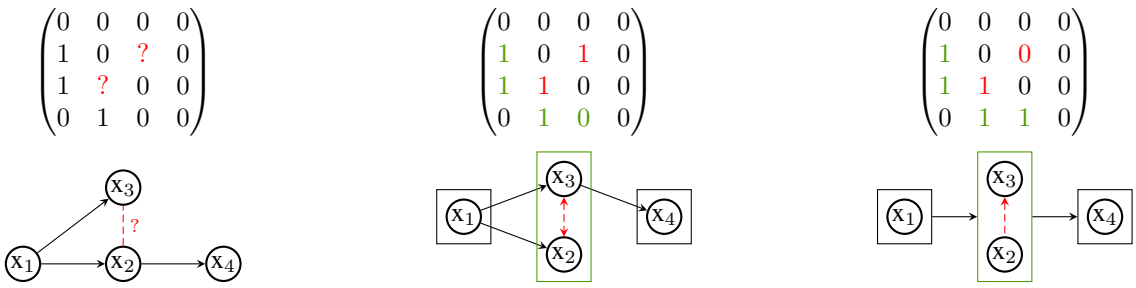
The theory Very conveniently, the method described above fits almost-perfectly into the already-covered theory in §3. To see this, note that our identifiability theory and following results required a fixed ordering, but it does not need to be the causal one: we can always find an equivalent SCM following the selected ordering by applying KR transports as described in §3. Therefore, the technical implications of both Thm. 1 and Cor. 2 still apply to this fixed ordering, we only need to re-state their implications with respect to the true causal data-generating process $\mathcal{M}$.

To this end, we only need to note that all possible graphs, once reduced into a DAG using the partition of SCCs (as in Fig. 6c), are exactly equal. In other words, every possible graph shares *the same causal dependencies between SCCs* with



(a) Partial knowledge. (b) Option 1. (c) Option 2. (d) Option 3.

Figure 5: Example of an SCM with partial knowledge about the causal graph (a) and possible outcomes: (b) in the actual SCM only the edge $\mathrm{x}_2 \to \mathrm{x}_3$ exists; (c) only the edge $\mathrm{x}_3 \to \mathrm{x}_2$ exists; (d) both edges exist (and therefore there exists a confounder between them).



(a) Partial-knowledge SCM. (b) Solved with a general NF. (c) Solved with an ANF.

Figure 6: Illustrative example (same as in Fig. 5) applying our method for partial information. First, we apply Tarjan's algorithm [28] to find the SCCs of the graph (rectangles) and build a new DAG where each node is a subset of the original nodes. If, for the SCCs, we use an NF with a general Jacobian matrix (b), we keep the individual edges and treat each SCC as a block. If we instead use an ANF (c), we pick an arbitrary order within each SCC, and merge the individual edges into SCC-wide edges. Red represents intra-SCC edges, and green inter-SCC edges. See App. B.5 for more details.

the other possible graphs. If we start treating them like block, calling $\{\mathrm{x}_i\} \subset \boldsymbol{S}_i \subset \{1, 2, \dots, d\}$ the SCC of the i-th node, and $\mathrm{pa}_{\boldsymbol{S}_i}$ and $\mathrm{an}_{\boldsymbol{S}_i}$ the parents and ancestors of every node in the SCC $\boldsymbol{S}_i$, then it is clear that we can write for every graph $\tilde{\boldsymbol{G}}$ its observed variables as $\mathrm{x}_i = \tilde{f}_i(\mathbf{x}_{\mathrm{pa}_{\boldsymbol{S}_i}}, \tilde{\mathbf{u}}_{\boldsymbol{S}_i})$ and, more importantly, we can write its "exogenous" variables as a function of the true ones, i.e., $\tilde{\mathbf{u}}_{\boldsymbol{S}_i} = \mathrm{KR}(\mathbf{u}_{\boldsymbol{S}_i})$, where KR is the Knöthe-Rosenblatt transport [14, 24]. Note also that it does not depend on the data, i.e., $\tilde{\mathrm{u}}_{\boldsymbol{S}_i} \perp\!\!\!\perp \mathbf{x} | \mathrm{u}_{\boldsymbol{S}_i}$.

Theorem 4 (Identifiability – Partial knowledge)**.** If an element of $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$, and another from $\tilde{\mathcal{F}} \times \mathcal{P}_{\mathbf{u}}$, where $\mathcal{F}$ and $\tilde{\mathcal{F}}$ are TMI maps with different intra-SCC orders (see above), generate the same observational distribution, then the two processes differ by an invertible, SCC-wise transformation of the variables $\mathbf{u}$.

Proof. Call $\mathcal{M}$ and $\tilde{\mathcal{M}}$ the elements from $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$ and $\tilde{\mathcal{F}} \times \mathcal{P}_{\mathbf{u}}$, respectively.

W.l.o.g. pick $\mathcal{M}$, and apply a KR transport to write it down as another element of $\tilde{\mathcal{F}} \times \mathcal{P}_{\mathbf{u}}$, call it $\hat{\mathcal{M}}$, with identical observational distribution as both $\mathcal{M}$ and $\tilde{\mathcal{M}}$.

Using Thm. 1, we know that the elements of $\tilde{\mathcal{M}}$ and $\hat{\mathcal{M}}$ differ by an invertible, component-wise transformation $\boldsymbol{h}$. Moreover, we can write $\hat{\mathbf{u}}_{\boldsymbol{S}_i}$ as a function of $\mathbf{u}_{\boldsymbol{S}_i}$, as argued above, where $\{i\} \subset \boldsymbol{S}_i \subset \{1, 2, \dots, d\}$ are the indexes of the SCC that contains u_i. Putting it all together:

$$\tilde{\mathrm{u}}_i = h_i(\hat{\mathrm{u}}_i) = h_i(\mathrm{KR}_i(\mathbf{u}_{\boldsymbol{S}_i}))\,, \tag{27}$$

and, in vectorial form, for each SCC $\boldsymbol{S}_i$,

$$\tilde{\mathbf{u}}_{\boldsymbol{S}_i} = \boldsymbol{h}_{\boldsymbol{S}_i}(\mathrm{KR}_{\boldsymbol{S}_i}(\mathbf{u}_{\boldsymbol{S}_i})) = (\boldsymbol{h}_{\boldsymbol{S}_i} \circ \mathrm{KR}_{\boldsymbol{S}_i})(\mathbf{u}_{\boldsymbol{S}_i}) \tag{28}$$

Q.E.D.

Corollary 5 (Causal consistency – Partial knowledge)**.** If a causal NF $T_{\boldsymbol{\theta}}$, with partial knowledge of the causal graph, SCC-wise isolates the exogenous variables of an SCM $\mathcal{M}$, then $T_{\boldsymbol{\theta}}$ is causally consistent with the true data-generating process, $\mathcal{M}$, with respect to each SCC.

Proof. The proof is identical to the one for Cor. 2, but using arguments with respect to the reduced graph after grouping all nodes in their respective SCCs.

Specifically, we can write using Thm. 4 the output of the flow as a function of the exogenous variables, $T_{\boldsymbol{\theta}}(\mathbf{x})_i = \boldsymbol{h}(\mathbf{u}_{\boldsymbol{S}_i})$, and using the true causal generator, we have

$$T_{\boldsymbol{\theta}}(\mathbf{x})_i = \boldsymbol{h}(\mathbf{u}_{\boldsymbol{S}_i}) = \boldsymbol{h}(f_{\boldsymbol{S}_i}(\mathbf{x}_{\mathrm{pa}_{\boldsymbol{S}_i}}, \mathbf{x}_{\boldsymbol{S}_i}))\,, \tag{29}$$

and hence the gradients agree with those from $\mathcal{M}$, *when looking at the reduced graph.* Q.E.D.

Thm. 4 and Cor. 5 provide analogues to those results from the main manuscript. It is important to note, however, that causal consistency refers to the causal relationships between SCCs *as a whole*, i.e., not at the causal relationships between individual nodes. The reason behind this is the same as why we had to introduced spurious edges in Fig. 6c: as we fix an ordering within each SCC, we may not be able to model indirect dependencies of nodes of one SCC to another unless we artificially introduce shortcuts. As such, *every result from the main paper holds*, if we treat each SCC (or block) as a whole. That is, when we reason about SCCs instead of individual nodes (note that if the whole causal graph is known every SCC contains a single node), we can safely talk about SCC-identifiability, causal SCC-consistency, and we can perform interventions and compute counterfactuals on SCCs.

C EXPERIMENTAL DETAILS AND EXTRA RESULTS

In this section, we complement the description of the experimental section from §4, and provide the reader with additional results in the following subsections. First, we describe the details common to every experiment, and delve into the specifics of each experiment in their respective subsections.

Hardware Every individual experiment shown in this paper ran on a single CPU with 8 GB of RAM. To run all experiments, we used a local computing cluster with an automatic job assignment system, so we cannot ensure the specific CPU used for each particular experiment. However, we know that every experiment used one of the following CPUs picked randomly given the demand when scheduled: AMD EPYC 7702 64-Core Processor, AMD EPYC 7662 64-Core Processor, Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz, or Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz.

Training and evaluation methodology For every experiment, we generated with using synthetic SCM a dataset with 20 000 training samples, 2500 validation samples, and 2500 test samples. We ran every model for 1000 epochs, and the results shown in the manuscript correspond to the test set evaluation at the last epoch. For the optimization, we used Adam [12] with an initial learning rate of 0.001, and reduce the learning rate with a decay factor of 0.95 when it reaches a plateau longer than 60 epochs. For hyperparameter tuning, we always perform a grid search with similar budget, and select the best hyperparameter combination according to validation loss, reporting always results from the test dataset in the manuscript. Every experiment is repeated 5 times, and we show averages and standard deviations.

Datasets This section provides all the information of the SCMs employed in the empirical evaluation of §4 of the main paper, and the following subsections. The exogenous variables always follow a standard normal distribution $\mathcal{N}(0, 1)$, except for LARGEBD, where a uniform distribution $\mathcal{U}(0, 1)$ is used instead. Subsequently, we define the 12 SCMs employed—encompassing both linear and non-linear equations—and we additionally provide their causal graph in Fig. 7.

Let us first define the softplus operation as $s(x) = \log(1.0 + e^x)$.

3-CHAIN$_{\text{LIN}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 \tag{30}$$

$$\tilde{f}_2(\mathrm{x}_1, \mathrm{u}_2) = 10 \cdot \mathrm{x}_1 - \mathrm{u}_2 \tag{31}$$

$$\tilde{f}_3(\mathrm{x}_2, \mathrm{u}_3) = 0.25 \cdot \mathrm{x}_2 + 2 \cdot \mathrm{u}_3 \tag{32}$$

3-CHAIN$_{\text{NLIN}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 \tag{33}$$
$$\tilde{f}_2(\mathrm{x}_1, \mathrm{u}_2) = e^{\mathrm{x}_1/2} + \mathrm{u}_2/4 \tag{34}$$
$$\tilde{f}_3(\mathrm{x}_2, \mathrm{u}_3) = \frac{(\mathrm{x}_2 - 5)^3}{15} + \mathrm{u}_3 \tag{35}$$

4-CHAIN$_{\text{LIN}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 \tag{36}$$
$$\tilde{f}_2(\mathrm{x}_1, \mathrm{u}_2) = 5 \cdot \mathrm{x}_1 - \mathrm{u}_2 \tag{37}$$
$$\tilde{f}_3(\mathrm{x}_2, \mathrm{u}_3) = -0.5 \cdot \mathrm{x}_2 - 1.5 \cdot \mathrm{u}_3 \tag{38}$$
$$\tilde{f}_4(\mathrm{x}_3, \mathrm{u}_4) = \mathrm{x}_3 + \mathrm{u}_4 \tag{39}$$

5-CHAIN$_{\text{LIN}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 \tag{40}$$
$$\tilde{f}_2(\mathrm{x}_1, \mathrm{u}_2) = 10 \cdot \mathrm{x}_1 - \mathrm{u}_2 \tag{41}$$
$$\tilde{f}_3(\mathrm{x}_2, \mathrm{u}_3) = 0.25 \cdot \mathrm{x}_2 + 2 \cdot \mathrm{u}_3 \tag{42}$$
$$\tilde{f}_4(\mathrm{x}_3, \mathrm{u}_4) = \mathrm{x}_3 + \mathrm{u}_4 \tag{43}$$
$$\tilde{f}_5(\mathrm{x}_4, \mathrm{u}_5) = -\mathrm{x}_4 + \mathrm{u}_5 \tag{44}$$

COLLIDER$_{\text{LIN}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 \tag{45}$$
$$\tilde{f}_2(\mathrm{u}_2) = 2 - \mathrm{u}_2 \tag{46}$$
$$\tilde{f}_3(\mathrm{x}_1, \mathrm{x}_2, \mathrm{u}_3) = 0.25 \cdot \mathrm{x}_2 - 0.5 \cdot \mathrm{x}_1 + 0.5 \cdot \mathrm{u}_3 \tag{47}$$

FORK$_{\text{LIN}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 \tag{48}$$
$$\tilde{f}_2(\mathrm{u}_2) = 2 - \mathrm{u}_2 \tag{49}$$
$$\tilde{f}_3(\mathrm{x}_1, \mathrm{x}_2, \mathrm{u}_3) = 0.25 \cdot \mathrm{x}_2 - 1.5 \cdot \mathrm{x}_1 + 0.5 \cdot \mathrm{u}_3 \tag{50}$$
$$\tilde{f}_4(\mathrm{x}_3, \mathrm{u}_4) = \mathrm{x}_3 + 0.25 \cdot \mathrm{u}_4 \tag{51}$$

FORK$_{\text{NLIN}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 \tag{52}$$
$$\tilde{f}_2(\mathrm{u}_2) = \mathrm{u}_2 \tag{53}$$
$$\tilde{f}_3(\mathrm{x}_1, \mathrm{x}_2, \mathrm{u}_3) = \frac{4}{1 + e^{-\mathrm{x}_1 - \mathrm{x}_2}} - \mathrm{x}_2^2 + 0.5 \cdot \mathrm{u}_3 \tag{54}$$
$$\tilde{f}_4(\mathrm{x}_3, \mathrm{u}_4) = \frac{20}{1 + e^{0.5 \cdot \mathrm{x}_3^2 - \mathrm{x}_3}} + \mathrm{u}_4 \tag{55}$$

LARGEBD$_{\text{NLIN}}$: Let us define

$$\mathrm{L}(x, y) = s(x + 1) + s(0.5 + y) - 3.0\,, \tag{56}$$

and let us call $\mathrm{CDF}^{-1}(\mu, b, x)$ the quantile function of a Laplace distribution with location μ, scale b, evaluated at x. Then

the structural equations are

$$\tilde{f}_1(\mathrm{u}_1) = s(1.8 \cdot \mathrm{u}_1) - 1 \tag{57}$$
$$\tilde{f}_2(\mathrm{x}_1, \mathrm{u}_2) = 0.25 \cdot \mathrm{u}_2 + \mathrm{L}(\mathrm{x}_1, 0) \cdot 1.5 \tag{58}$$
$$\tilde{f}_3(\mathrm{x}_1, \mathrm{u}_3) = \mathrm{L}(\mathrm{x}_1, \mathrm{u}_3) \tag{59}$$
$$\tilde{f}_4(\mathrm{x}_2, \mathrm{u}_4) = \mathrm{L}(\mathrm{x}_2, \mathrm{u}_4) \tag{60}$$
$$\tilde{f}_5(\mathrm{x}_3, \mathrm{u}_5) = \mathrm{L}(\mathrm{x}_3, \mathrm{u}_5) \tag{61}$$
$$\tilde{f}_6(\mathrm{x}_4, \mathrm{u}_6) = \mathrm{L}(\mathrm{x}_4, \mathrm{u}_6) \tag{62}$$
$$\tilde{f}_7(\mathrm{x}_5, \mathrm{u}_7) = \mathrm{L}(\mathrm{x}_5, \mathrm{u}_7) \tag{63}$$
$$\tilde{f}_8(\mathrm{x}_6, \mathrm{u}_8) = 0.3 \cdot \mathrm{u}_8 + (s(\mathrm{x}_6 + 1) - 1) \tag{64}$$
$$\tilde{f}_9(\mathrm{x}_7, \mathrm{x}_8, \mathrm{u}_9) = \mathrm{CDF}^{-1}\left(-s\left(\frac{\mathrm{x}_7 \cdot 1.3 + \mathrm{x}_8}{3} + 1\right) + 2, 0.6, \mathrm{u}_9\right) \tag{65}$$

SIMPSON$_{\text{NLIN}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 \tag{66}$$
$$\tilde{f}_2(\mathrm{x}_1, \mathrm{u}_2) = s(1 - \mathrm{x}_1) + \sqrt{3/20} \cdot \mathrm{u}_2 \tag{67}$$
$$\tilde{f}_3(\mathrm{x}_1, \mathrm{x}_2, \mathrm{u}_3) = \tanh(2 \cdot \mathrm{x}_2) + 1.5 \cdot \mathrm{x}_1 - 1 + \tanh(\mathrm{u}_3) \tag{68}$$
$$\tilde{f}_4(\mathrm{x}_3, \mathrm{u}_4) = \frac{\mathrm{x}_3 - 4}{5} + 3 + \frac{1}{\sqrt{10}} \cdot \mathrm{u}_4 \tag{69}$$

SIMPSON$_{\text{SYMPROD}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 \tag{70}$$
$$\tilde{f}_2(\mathrm{x}_1, \mathrm{u}_2) = 2 \cdot \tanh(2 \cdot \mathrm{x}_1) + \frac{1}{\sqrt{10}} \cdot \mathrm{u}_2 \tag{71}$$
$$\tilde{f}_3(\mathrm{x}_1, \mathrm{x}_2, \mathrm{u}_3) = 0.5 \cdot \mathrm{x}_1 \cdot \mathrm{x}_2 + \frac{1}{\sqrt{2}} \cdot \mathrm{u}_3 \tag{72}$$
$$\tilde{f}_4(\mathrm{x}_1, \mathrm{u}_4) = \tanh(1.5 \cdot \mathrm{x}_1) + \sqrt{\frac{3}{10}} \cdot \mathrm{u}_4 \tag{73}$$

TRIANGLE$_{\text{LIN}}$:

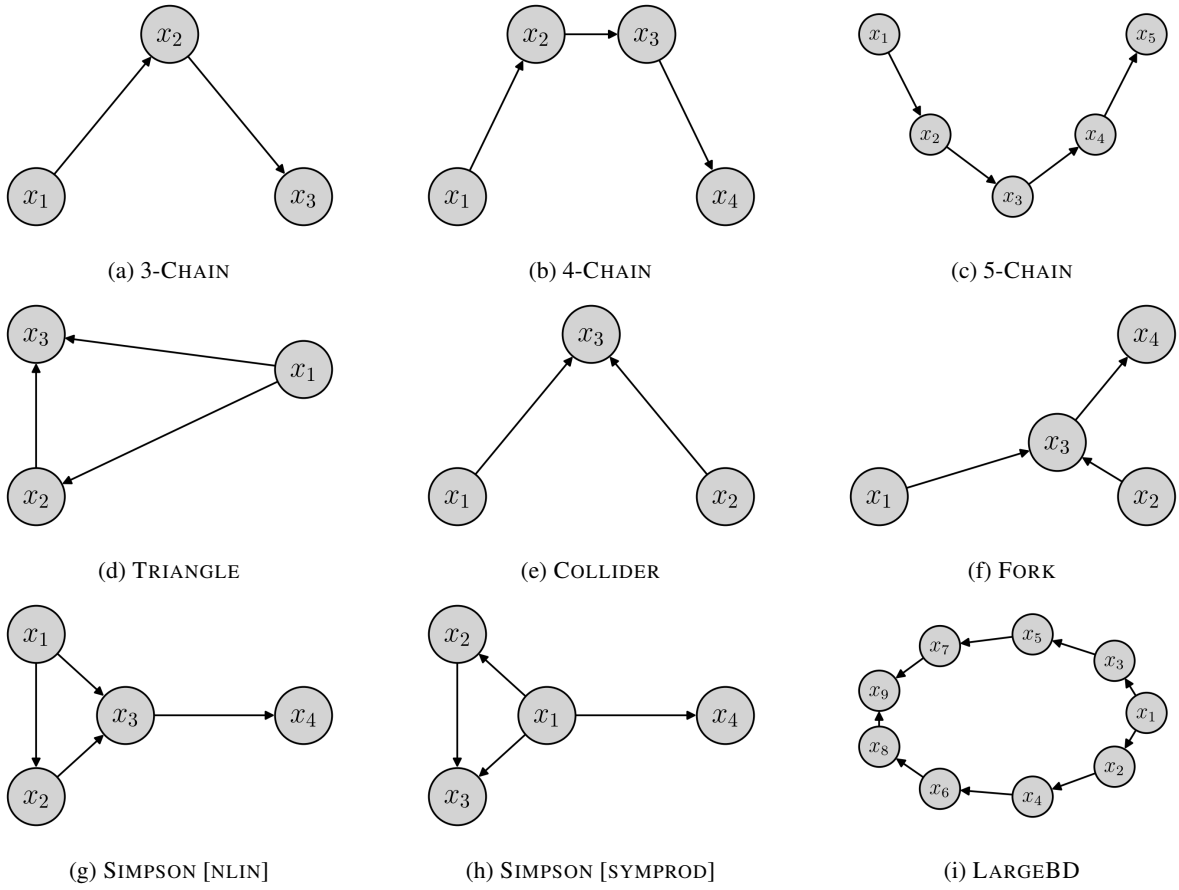
$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 + 1 \tag{74}$$
$$\tilde{f}_2(\mathrm{x}_1, \mathrm{u}_2) = 10 \cdot \mathrm{x}_1 - \mathrm{u}_2 \tag{75}$$
$$\tilde{f}_3(\mathrm{x}_1, \mathrm{x}_2, \mathrm{u}_3) = 0.5 \cdot \mathrm{x}_2 + \mathrm{x}_1 + \mathrm{u}_3 \tag{76}$$

TRIANGLE$_{\text{NLIN}}$:

$$\tilde{f}_1(\mathrm{u}_1) = \mathrm{u}_1 + 1 \tag{77}$$
$$\tilde{f}_2(\mathrm{x}_1, \mathrm{u}_2) = 2 \cdot \mathrm{x}_1^2 + \mathrm{u}_2 \tag{78}$$
$$\tilde{f}_3(\mathrm{x}_1, \mathrm{x}_2, \mathrm{u}_3) = \frac{20}{1 + e^{-\mathrm{x}_2^2 + \mathrm{x}_1}} + \mathrm{u}_3 \tag{79}$$

C.1 ABLATION: BASE DISTRIBUTION

We now assess to which extent a mismatch of the distribution $P_{\mathbf{u}}$ between the SCM and the causal NF negatively affects performance. To this end, we consider two more complex SCMs—SIMPSON [9] and TRIANGLE [26]—and distributions—



(a) 3-CHAIN (b) 4-CHAIN (c) 5-CHAIN

(d) TRIANGLE (e) COLLIDER (f) FORK

(g) SIMPSON [NLIN] (h) SIMPSON [SYMPROD] (i) LARGEBD

Figure 7: Causal graph of the different SCMs considered in §4 and App. C.

Normal and Laplace—for which we either fix or learn their parameters during training. Both SCMs use a standard Normal distribution for $P_{\mathbf{u}}$.

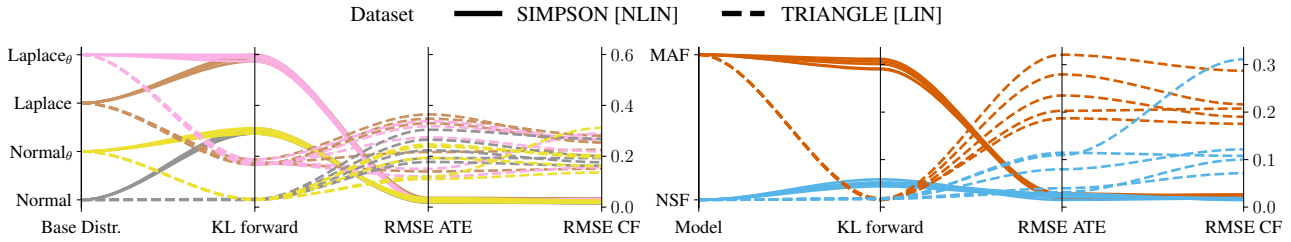
Hyperparameter tuning While we fixed the flow to have a single MAF [18] layer with ELU [4] activation functions, we determined through cross-validation the optimal number of layers and hidden units of the MLP network within MAF. Specifically, we considered the following combinations ($[a, b]$ represents two layers with a and b hidden units): $[16, 16, 16, 16]$, $[32, 32, 32]$, $[16, 16, 16]$, $[32, 32]$, $[32]$, $[64]$. As discussed at the start of the section, we report test results for the configuration with the best validation performance at the last epoch.

Results The results, shown in Fig. 8a, reveal a notable distinction between Normal and Laplace distributions in terms of density estimation. However, this discrepancy appears to have minimal implications for Average Treatment Effect (ATE) and counterfactual estimation. We hypothesize that this disparity originates from dissimilarities in their tails, as it can be inferred by the slight edge of Normal over Laplace on the last column—which measures *per sample* differences—where bigger errors happen at the outliers elements which are, by definition, scarce. Interestingly, with this particular architecture of causal NF, every model struggles to model the denser TRIANGLE SCM.

C.2 ABLATION: FLOW ARCHITECTURE

Considering the observed challenges faced by the Masked Autoregressive Flow (MAF) [18] layer in accurately modelling the TRIANGLE SCM in the previous experiment, we further investigate the potential impact of flow architecture on performance.

Hyperparameter Tuning We cross-validate again the optimal number of layers and hidden units of the MLP internally used by the unique layer of the Causal NF. We consider the following values ($[a, b]$ represents two layers with a and b



(a) Ablation on the base distribution

(b) Ablation on the flow architecture

Figure 8: Performance on the SIMPSON$_{\text{NLIN}}$ and TRIANGLE$_{\text{LIN}}$ datasets of causal NFs with a) different base distributions (Normal and Laplace), where $\boldsymbol{\theta}$ indicates that we learn the parameters of the base distribution; and b) flow architectures. Differences in base distribution affect KL divergence, while the choice of flow architecture influences the overall performance.

hidden units): $[32, 32, 32]$, $[16, 16, 16]$, $[32, 32]$, $[32]$, and $[64]$. As before, test results are reported for the configuration that achieved the best performance on the validation set at the final epoch.

Results Fig. 8b summarizes our results, where we consider a Causal NF with one MAF [18] layer (depicted in orange), and a Causal NF with a Neural Spline Flow (NSF) [8] as layer (depicted in blue). Note that, NSFs are built on top of MAFs and abandoned the realm of affine ANFs, and are thus expected to outperform MAFs in general. We employed the same set of SCMs as in App. C.1.

Our empirical analysis reveals that the NSF consistently outperforms the MAF across the three metrics: observational distribution (measured by the KL divergence), ATE estimation, and counterfactual estimation. Whilst expected, these findings highlight the practical implications of selecting an appropriate flow architecture, which should be taken into consideration by practitioners.

C.3 COMPARISON: EXTRA NON-LINEAR SCMS

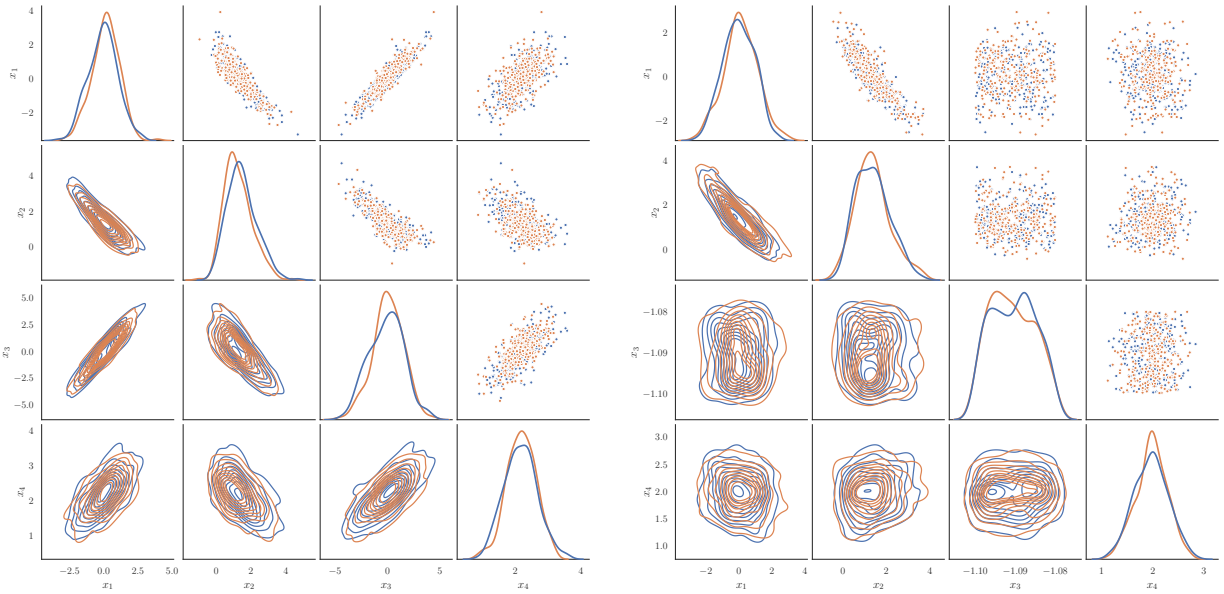
In this section, we complement the results from §4.1 and provide a more extensive comparison of the proposed Causal NF, along with CAREFL [11] and VACA [26], on additional datasets.

Hyperparameter Tuning For VACA, we cross-validated the dropout rate with values $\{0.0, 0.1\}$, the GNN layer architecture with $\{\text{GIN}, \text{PNA}, \text{PNADisjoint}\}$, (see [26] for details), and the number of layers in the MLP prior to the GNN with choices $\{1, 2\}$. For CAREFL, we cross-validated the number of layers in the flow, $\{1, \text{diam}\,\boldsymbol{G}\}$, and the number of layers and hidden units in the MLP composing the flow layers (same format as before), $\{[16, 16, 16], [32, 32], [32], [64]\}$. For Causal NF, we used the abductive model with a single layer, and cross-validated the number of layers and hidden units in the MLP composing the layer of the flow with values $\{[16, 16, 16, 16], [32, 32, 32], [16, 16, 16], [32, 32], [32], [64]\}$. We report test results for the configuration with the best validation performance at the final epoch.

Results Tab 3 shows the performance of each model for all the considered datasets, further validating the conclusions drawn in the main manuscript: the proposed Causal NF consistently outperforms both CAREFL and VACA in terms of performance and computational efficiency. The performance of VACA is notably inferior, and its computation time is significantly longer, primarily due to the complexity of graph neural networks (GNNs). Our Causal NF achieves similar performance to CAREFL in terms of observational fitting, while surpassing it on interventional and counterfactual estimation tasks. Additionally, Causal NF outperforms CAREFL in computational speed. This is to be expected since the optimal CAREFL architectures often have multiple layers, resulting in increased computation time. In contrast, Causal NF has a single layer, reducing computational complexity.

Fig. 9 qualitative proofs the effectiveness of the proposed Causal NF in accurately modelling both observational and interventional distributions for the SIMPSON$_{\text{NLIN}}$ dataset. In this plot, blue represent the real distribution/samples, while orange represents the ones generated by Causal NF. Fig. 9a clearly shows that the model successfully captured the correlations among all variables in the observational distribution. Furthermore, Fig. 9b displays the interventional distribution obtained when we do $do(\text{x}_3 = -1.09)$, i.e., when we intervene on the 25-th empirical percentile of x_3. Remarkably, Causal NF accurately learns the distribution of descendant variables, i.e., x_4, and effectively breaks any dependency between the ancestors of the intervened variable and x_4. Additionally, Fig. 10 shows a similar analysis for 5-CHAIN$_{\text{LIN}}$, when we perform

$do(\mathrm{x}_3 = 2.18)$—which corresponds to intervening on the 75-th percentile of x_3—clearly showing that the correlations not involving the intervened path ($\mathrm{x}_1 \to \mathrm{x}_2$ and $\mathrm{x}_4 \to \mathrm{x}_5$) are preserved.



(a) Observational distribution.

(b) Interventional distribution $do(\mathrm{x}_3 = -1.09)$.

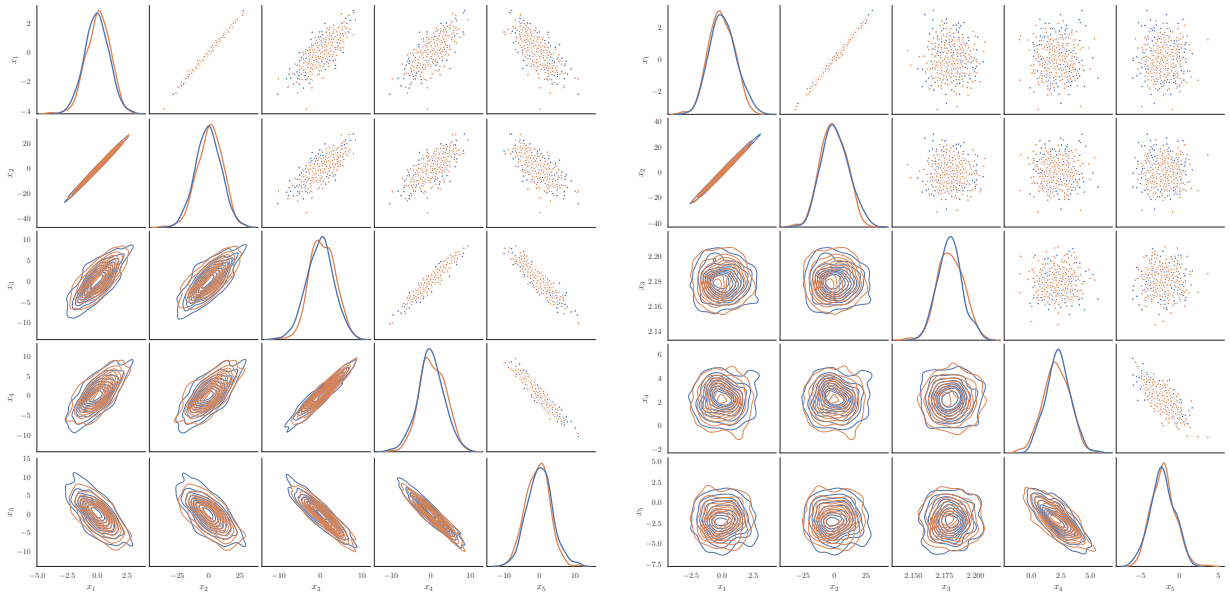
Figure 9: Pair plot of real (in blue) and generated (in orange) data of SIMPSON$_{\text{NLIN}}$. On the left are samples from the true and learnt observational distribution. On the right are samples from the true and learnt interventional distribution when $do(\mathrm{x}_3 = -1.09)$. The plot illustrates that the dependency of x_4 on the ancestors of x_3, namely x_1 and x_2, is effectively broken.

D DETAILS ON THE FAIRNESS USE-CASE

In this section, we provide additional details on the use-case of fairness auditing and classification using the German dataset [7], whose causal graph is shown in Fig. 11.

Training For this section, we performed minimal hyperparameter tuning, and only tested a few combinations by hand. We decided to use a Neural Spline Flow (NSF) [8] for the single layer of the Causal NF, which internally uses an MLP with 3 layers, and 32 hidden units each. We use Adam [12] as the optimizer, with a learning rate of 0.01, along with a plateau scheduler with a decay factor of 0.9 and a patience parameter of 60 epochs. The training is performed for 1000 epochs, and the results are reported using 5-fold cross-validation with a $80 - 10 - 10$ split for train, validation, and test data.

Results On addition to the results from §4.2, Fig. 12 shows two pair plots from one of the 5 runs, chosen at random. The true empirical distribution is shown in blue, and the learnt distribution by Causal NF is depicted in orange. Specifically, Fig. 12a illustrates the observational distribution, and Fig. 12b the interventional distribution, obtained when we intervene on the *sex* variable and set it to 1, i.e., $do(\mathrm{x}_1 = 1)$. We can observe that Causal NF achieves a remarkable fit in both cases, demonstrating its capability to handle discrete data, and partial knowledge of the causal graph.



(a) Observational distribution.

(b) Interventional distribution $do(\mathrm{x}_3 = 2.18)$.

Figure 10: Pair plot of real (in blue) and generated (in orange) data of 5-CHAIN$_{\text{LIN}}$. On the left are samples from the true and learnt observational distribution. On the right are samples from the true and learnt interventional distribution when $do(\mathrm{x}_3 = 2.18)$. The plot illustrates that the dependency of x_4 and x_5 on the ancestors of x_3, namely x_1 and x_2, is effectively broken.

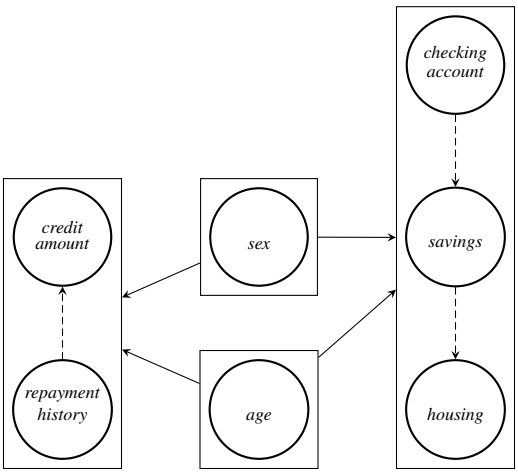
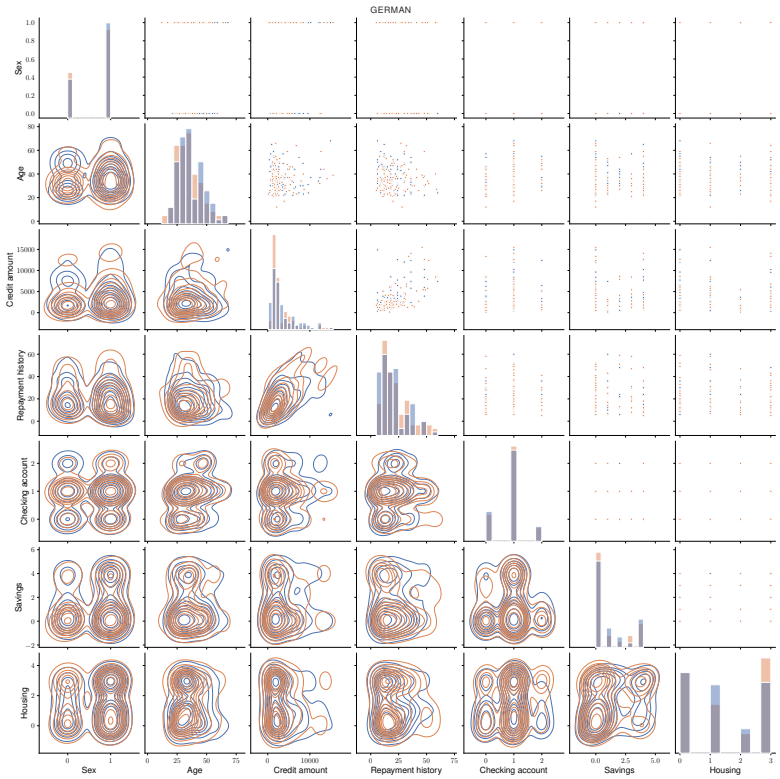


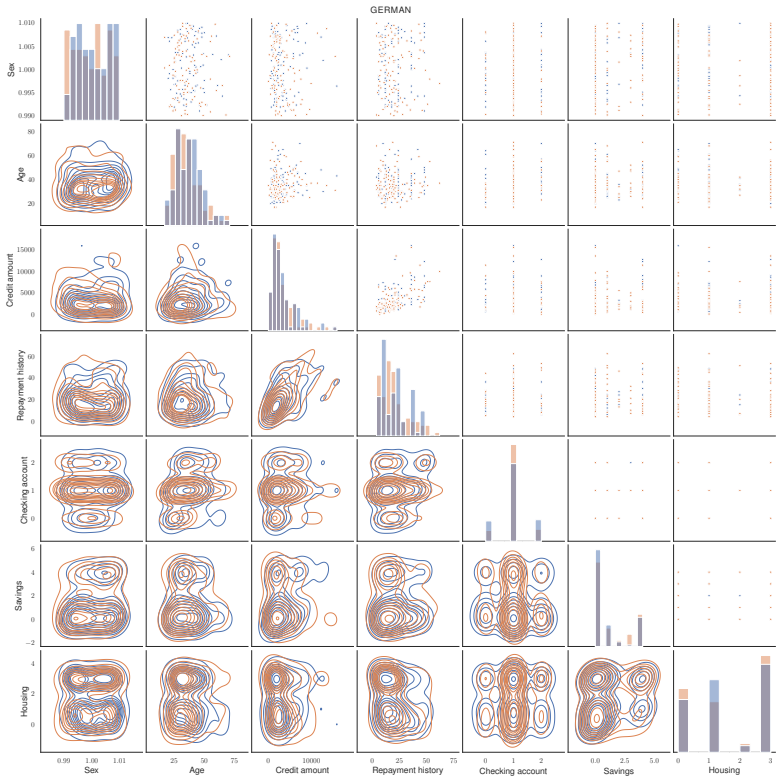
Figure 11: Partial causal graph used for the German Credit dataset [7]. Rectangles show the strongly connected components (SCCs) grouping different variables. Solid arrows represent causal relationships between SCCs, and dashed arrows represent an arbitrary order picked to learn the joint distribution of each SCC with an ANF. See App. B.5 for an in-depth explanation on the proposed method to deal with partial causal graphs using causal NFs.

Table 3: Comparison, on different SCMs, of the proposed Causal NF, VACA [26], and CAREFL [11] with the do-operator proposed in §3.1. Results averaged over five runs.

		Performance			Time Evaluation (µs)		
Dataset	Model	KL	ATE_{RMSE}	CF_{RMSE}	Training	Evaluation	Sampling
3-CHAIN LIN [26]	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.05_{0.01}}$	$\mathbf{0.04_{0.01}}$	$\mathbf{0.41_{0.06}}$	$\mathbf{0.48_{0.10}}$	$\mathbf{0.76_{0.06}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$0.20_{0.13}$	$0.20_{0.09}$	$0.68_{0.24}$	$0.97_{0.33}$	$1.94_{0.77}$
	VACA	$4.44_{1.03}$	$5.76_{0.07}$	$4.98_{0.10}$	$36.19_{1.54}$	$28.33_{0.72}$	$75.34_{4.58}$
3-CHAIN NLIN [26]	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.03_{0.01}}$	$\mathbf{0.02_{0.01}}$	$\mathbf{0.52_{0.06}}$	$\mathbf{0.56_{0.03}}$	$\mathbf{1.02_{0.05}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$\mathbf{0.05_{0.02}}$	$0.04_{0.02}$	$\mathbf{0.60_{0.22}}$	$0.84_{0.22}$	$1.66_{0.41}$
	VACA	$12.82_{1.00}$	$1.54_{0.03}$	$1.32_{0.02}$	$39.45_{4.12}$	$30.93_{2.30}$	$84.36_{9.60}$
4-CHAIN LIN	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.07_{0.02}}$	$\mathbf{0.04_{0.01}}$	$\mathbf{0.56_{0.08}}$	$\mathbf{0.62_{0.15}}$	$\mathbf{1.54_{0.40}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$0.16_{0.07}$	$0.14_{0.04}$	$\mathbf{0.70_{0.28}}$	$0.99_{0.20}$	$2.85_{0.54}$
	VACA	$13.14_{0.73}$	$3.82_{0.01}$	$3.72_{0.05}$	$61.85_{5.06}$	$49.31_{4.11}$	$92.06_{7.93}$
5-CHAIN LIN	Causal NF	$0.01_{0.00}$	$\mathbf{0.12_{0.02}}$	$\mathbf{0.08_{0.01}}$	$\mathbf{0.62_{0.19}}$	$\mathbf{0.69_{0.15}}$	$\mathbf{1.91_{0.44}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$0.47_{0.23}$	$0.46_{0.22}$	$\mathbf{0.79_{0.41}}$	$1.19_{0.25}$	$4.21_{0.87}$
	VACA	$17.31_{0.84}$	$5.95_{0.05}$	$6.06_{0.08}$	$103.75_{10.04}$	$80.81_{11.06}$	$124.52_{20.86}$
COLLIDER LIN [26]	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.02_{0.01}}$	$\mathbf{0.01_{0.00}}$	$\mathbf{0.46_{0.12}}$	$0.56_{0.11}$	$0.95_{0.19}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$\mathbf{0.02_{0.01}}$	$\mathbf{0.01_{0.00}}$	$\mathbf{0.39_{0.07}}$	$\mathbf{0.45_{0.05}}$	$\mathbf{0.74_{0.07}}$
	VACA	$13.45_{0.43}$	$0.22_{0.01}$	$0.86_{0.02}$	$37.22_{3.55}$	$28.77_{4.22}$	$71.21_{6.73}$
FORK LIN [2]	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.03_{0.01}}$	$\mathbf{0.01_{0.00}}$	$\mathbf{0.52_{0.05}}$	$\mathbf{0.59_{0.08}}$	$\mathbf{1.57_{0.57}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$\mathbf{0.04_{0.01}}$	$0.02_{0.00}$	$\mathbf{0.60_{0.17}}$	$0.78_{0.16}$	$\mathbf{2.39_{1.06}}$
	VACA	$8.75_{0.73}$	$0.87_{0.02}$	$1.43_{0.02}$	$45.84_{4.64}$	$34.66_{2.39}$	$73.29_{4.70}$
FORK NLIN [2]	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.07_{0.02}}$	$\mathbf{0.07_{0.00}}$	$\mathbf{0.63_{0.16}}$	$\mathbf{0.74_{0.31}}$	$\mathbf{1.84_{0.84}}$
	CAREFL†	$0.01_{0.01}$	$0.11_{0.04}$	$0.18_{0.07}$	$\mathbf{0.57_{0.17}}$	$\mathbf{0.77_{0.08}}$	$\mathbf{1.96_{0.17}}$
	VACA	$5.09_{0.60}$	$2.01_{0.03}$	$3.19_{0.06}$	$49.22_{5.48}$	$42.13_{2.95}$	$101.02_{18.94}$
LARGEBD NLIN [9]	Causal NF	$\mathbf{1.51_{0.04}}$	$\mathbf{0.02_{0.00}}$	$\mathbf{0.01_{0.00}}$	$\mathbf{0.52_{0.10}}$	$\mathbf{0.60_{0.17}}$	$\mathbf{3.05_{0.66}}$
	CAREFL†	$1.51_{0.05}$	$0.05_{0.01}$	$0.08_{0.01}$	$0.84_{0.47}$	$1.18_{0.17}$	$8.25_{1.29}$
	VACA	$53.66_{2.07}$	$0.39_{0.00}$	$0.82_{0.02}$	$164.92_{11.10}$	$137.88_{15.72}$	$167.94_{25.75}$
SIMPSON NLIN [9]	Causal NF	$\mathbf{0.29_{0.01}}$	$\mathbf{0.04_{0.01}}$	$\mathbf{0.02_{0.00}}$	$\mathbf{0.58_{0.18}}$	$\mathbf{0.63_{0.26}}$	$\mathbf{1.57_{0.64}}$
	CAREFL†	$\mathbf{0.29_{0.01}}$	$\mathbf{0.04_{0.01}}$	$0.03_{0.00}$	$\mathbf{0.69_{0.32}}$	$1.02_{0.26}$	$2.95_{0.79}$
	VACA	$18.97_{0.66}$	$0.60_{0.00}$	$1.19_{0.02}$	$54.76_{7.46}$	$43.69_{7.92}$	$87.01_{16.33}$
SIMPSON SYMPROD [9]	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.07_{0.01}}$	$\mathbf{0.12_{0.02}}$	$\mathbf{0.59_{0.17}}$	$\mathbf{0.60_{0.11}}$	$\mathbf{1.51_{0.30}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$0.10_{0.02}$	$0.17_{0.04}$	$\mathbf{0.49_{0.15}}$	$0.81_{0.19}$	$1.91_{0.33}$
	VACA	$13.85_{0.64}$	$0.89_{0.00}$	$1.50_{0.04}$	$49.26_{4.09}$	$37.78_{3.41}$	$79.20_{14.60}$
TRIANGLE LIN [26]	Causal NF	$0.00_{0.00}$	$0.24_{0.05}$	$0.21_{0.05}$	$\mathbf{0.54_{0.05}}$	$\mathbf{0.56_{0.04}}$	$\mathbf{1.05_{0.07}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$\mathbf{0.15_{0.06}}$	$\mathbf{0.14_{0.03}}$	$\mathbf{0.60_{0.20}}$	$0.75_{0.05}$	$1.50_{0.10}$
	VACA	$3.82_{0.69}$	$7.49_{0.07}$	$7.22_{0.17}$	$27.46_{1.53}$	$21.61_{1.00}$	$67.00_{6.23}$
TRIANGLE NLIN [26]	Causal NF	$\mathbf{0.00_{0.00}}$	$\mathbf{0.12_{0.03}}$	$\mathbf{0.13_{0.02}}$	$\mathbf{0.52_{0.07}}$	$\mathbf{0.58_{0.07}}$	$\mathbf{1.07_{0.12}}$
	CAREFL†	$\mathbf{0.00_{0.00}}$	$\mathbf{0.12_{0.03}}$	$0.17_{0.03}$	$\mathbf{0.57_{0.18}}$	$\mathbf{0.83_{0.26}}$	$\mathbf{1.68_{0.62}}$
	VACA	$7.71_{0.60}$	$4.78_{0.01}$	$4.19_{0.04}$	$28.82_{1.21}$	$23.00_{0.55}$	$70.65_{3.70}$



(a) Observational distribution.



(b) Interventional distribution $do(\mathrm{x}_1 = 1)$.

Figure 12: Pair plot of real (in blue) and generated (in orange) data of German dataset. Above, the samples from the true and learnt observational distributions. Below, the samples from the true and learnt interventional distributions when $do(\mathrm{x}_1 = 1)$. The plot illustrates that Causal NF is able to handle discrete data and correctly intervene.