

APPENDIX

A CAUSALITY: BASICS

A *causal factorization* as in equation 3 allows to model causal relations. With *Structural Causal Models* (SCM), we are able to express causal relations on a functional level. Following Peters et al. (2017) we define a SCM in the following way:

Definition 1. A Structural Causal Model (SCM) $\mathcal{S} = (S, P_{\mathbf{N}})$ consists of a collection S of D (structural) assignments

$$X_j := f_j(\tilde{\mathbf{X}}_{pa(j)}, N_j), \quad j = 1, \dots, D \quad (10)$$

where $pa(j) \subset \{1, \dots, j-1\}$ are called parents of X_j . $P_{\mathbf{N}}$ denotes the distribution over the noise variables $\mathbf{N} = (N_1, \dots, N_D)$ which are assumed to be jointly independent.

An SCM defined as above produces an acyclic graph G and induces a probability distribution over $P_{\tilde{\mathbf{X}}}$ which allows for the *causal factorization* as in equation 3 Peters et al. (2017). Children of X_i in G are denoted as $ch(i)$ or $ch(X_i)$.

An SCM satisfies the *causal sufficiency* assumption if all the noise variables in Definition 1 are indeed jointly independent. A random variable H in the SCM is called *confounder* between two variables X_i, X_j if it causes both of them. If a confounder is not observed, we call it hidden confounder. If there exists a hidden confounder, the causal sufficiency assumption is violated.

The random variables in an SCM correspond to vertices in a graph and the structural assignments S define the edges of this graph. Two sets of vertices \mathbf{A}, \mathbf{B} are said to be d -separated if there exists a set of vertices \mathbf{C} such that every path between \mathbf{A} and \mathbf{B} is blocked. For details see e.g. Peters et al. (2017). The subscript \perp_d denotes d -separability which in this case is denoted by $\mathbf{A} \perp_d \mathbf{B}$. An SCM generates a probability distribution $P_{\tilde{\mathbf{X}}}$ which satisfies the *Causal Markov Condition*, that is $\mathbf{A} \perp_d \mathbf{B} \mid \mathbf{C}$ results in $\mathbf{A} \perp \mathbf{B} \mid \mathbf{C}$ for sets or random variables $\mathbf{A}, \mathbf{B}, \mathbf{C} \subset \tilde{\mathbf{X}}$. The Causal Markov Condition can be seen as an inherent property of a causal system which leaves marks in the data distribution.

A distribution $P_{\tilde{\mathbf{X}}}$ is said to be *faithful* to the graph G if $\mathbf{A} \perp \mathbf{B} \mid \mathbf{C}$ results in $\mathbf{A} \perp_d \mathbf{B} \mid \mathbf{C}$ for all $\mathbf{A}, \mathbf{B}, \mathbf{C} \subset \tilde{\mathbf{X}}$. This means from the distribution $P_{\tilde{\mathbf{X}}}$ statements about the underlying graph G can be made.

Assuming both, faithfulness and the Causal Markov condition, we obtain that the d -separation statements in G are equivalent to the conditional independence statements in $P_{\tilde{\mathbf{X}}}$. These two assumptions allow for a whole class of causal discovery algorithms like the PC- or IC-algorithm (Spirtes & Glymour, 1991; Pearl, 2009).

The smallest set \mathbf{M} such that $Y \perp_d \mathbf{X} \setminus (\{Y\} \cup \mathbf{M})$ is called *Markov Blanket*. It is given by $\mathbf{M} = \mathbf{X}_{pa(Y)} \cup \mathbf{X}_{ch(Y)} \cup \mathbf{X}_{pa(ch(Y))} \setminus \{Y\}$. The *Markov Blanket* of Y is the only set of vertices which are necessary to predict Y .

B DISCUSSION AND ILLUSTRATION OF ASSUMPTIONS

B.1 EXAMPLES

Domain generalization is in general impossible without strong assumptions (in contrast to classical supervised learning). In our view, the interesting question is “Which strong assumptions are the most useful in a given setting?” For instance, Heinze-Deml et al. (2018) use Assumption 2 to identify causes for birth rates in different countries. If all variables mediating the influence of continent/country (environment variable) on birth rates (target variable) are included in the model (e.g. GDP, Education), this assumption is reasonable. The same may hold for other epidemiological investigations as well. Pfister et al. (2019) suppose Assumption 2 in the field of finance.

Another reasonable example are data augmentations in computer vision. Deliberate image rotations, shifts and distortions can be considered as environment interventions that preserve the relation be-

tween semantic image features and object classes (see e.g. Mitrovic et al. (2020)), i.e. verify assumption 2. In general, assumption 2 may be justified when one studies a fundamental mechanism that can reasonably be assumed to remain invariant across environments, but is obscured by unstable accidental relationships between observable variables.

B.2 CAUSAL SUFFICIENCY

Violation of the causal sufficiency assumption might prevent Assumption 2 to hold true. For instance, a causal graph with edges $H \rightarrow X_1, X_1 \rightarrow Y$ and $H \rightarrow Y$ where H is not observed, violates the causal sufficiency assumption. If the environment influences X_1 , i.e. the graph also contains edge $E \rightarrow X_1$, and the generated distribution satisfies the Causal Markov Condition, it follows that $Y \perp E \mid X_1$ is unachievable. This example also illustrates also that the causal sufficiency assumption is necessary for the principle of ICM.

B.3 ROBUSTNESS

To illustrate the impact of causality on robustness, consider the following example: Suppose we would like to estimate the gas consumption of a car. In a sufficiently narrow setting, the total amount of money spent on gas might be a simple and accurate predictor. However, gas prices vary dramatically between countries and over time, so statistical models relying on it will not be robust, even if they fit the training data very well. Gas costs are an *effect* of gas consumption, and this relationship is unstable due to external influences. In contrast, predictions on the basis of the *causes* of gas consumption (e.g. car model, local speed limits and geography, owner’s driving habits) tend to be much more robust, because these causal relations are intrinsic to the system and not subjected to external influences. Note that there is a trade-off here: Including gas costs in the model will improve estimation accuracy when gas prices remain sufficiently stable, but will impair results otherwise. By considering the same phenomenon in several environments simultaneously, we hope to gain enough information to adjust this trade-off properly.

In the gas example, countries can be considered as environments that “intervene” on the relation between consumed gas and money spent, e.g. by applying different tax policies. In contrast, interventions changing the impact of motor properties or geography on gas consumption are much less plausible – powerful motors and steep roads will always lead to higher consumption. From a causal standpoint, finding robust models is therefore a causal discovery task Meinshausen (2018).

C NORMALIZING FLOWS

Normalizing flows are a specific type of neural network architecture which are by construction invertible and have a tractable Jacobian. They are used for density estimation and sampling of a target density (for an overview see Papamakarios et al. (2019)). This in turn allows optimizing information theoretic objectives in a convenient and mathematically sound way.

Similarly as in the paper, we denote with \mathcal{H} the set of feature extractors $h: \mathbb{R}^D \rightarrow \mathbb{R}^M$ where M is chosen a priori. The set of all one-dimensional (conditional) normalizing flows is denoted by \mathcal{T} .

Together with a reference distribution p_{ref} , T defines a new distribution $\nu_T = (T(\cdot; h(\mathbf{x})))_{\#}^{-1} p_{ref}$ which is called the push-forward of the reference distribution p_{ref} (Marzouk et al., 2016). By drawing samples from p_{ref} and applying T on these samples we obtain samples from this new distribution. The density of this so-obtained distribution p_{ν_T} can be derived from the change of variables formula:

$$p_{\nu_T}(y \mid h(\mathbf{x})) = p_{ref}(T(y; h(\mathbf{x}))) |\nabla_y T(y; h(\mathbf{x}))|. \quad (11)$$

The KL-divergence between the target distribution $p_{Y|h(\mathbf{x})}$ and the flow-based model p_{ν_T} can be written as follows:

$$\begin{aligned}
\mathbb{E}_{h(\mathbf{X})}[D_{\text{KL}}(p_{Y|h(\mathbf{X})}||p_{\nu_T})] &= \mathbb{E}_{h(\mathbf{X})}\left[\mathbb{E}_{Y|h(\mathbf{X})}\left[\log\left(\frac{p_{Y|h(\mathbf{X})}}{p_{\nu_T}}\right)\right]\right] \\
&= -H(Y|h(\mathbf{X})) - \mathbb{E}_{h(\mathbf{X}),Y}[\log p_{\nu_T}(Y|h(\mathbf{X}))] \\
&= -H(Y|h(\mathbf{X})) + \mathbb{E}_{h(\mathbf{X}),Y}[-\log p_{\text{ref}}(T(y;h(\mathbf{x})) - \log |\nabla_y T(y;h(\mathbf{x}))|)] \quad (12)
\end{aligned}$$

The last two terms in equation 12 correspond to the negative log-likelihood (NLL) for conditional flows with distribution p_{ref} in latent space. The NLL in Section 3 is given in case we assume the reference distribution is the standard normal distribution.

We restate Lemma 1 with a more general notation:

Lemma 1. *Let \mathbf{X}, Y be random variables. We furthermore assume that for each $h \in \mathcal{H}$ there exists one $T \in \mathcal{T}$ with $\mathbb{E}_{h(\mathbf{X})}[D_{\text{KL}}(p_{Y|h(\mathbf{X})}||p_{\nu_T})] = 0$. Then, the following two statements are true*

(a) *Let*

$$h^*, T^* = \arg \min_{h \in \mathcal{H}, T \in \mathcal{T}} -\mathbb{E}_{h(\mathbf{X}),Y}[\log p_{\nu_T}(Y|h(\mathbf{X}))]$$

then it holds $h^ = g^*$ where $g^* = \arg \max_{g \in \mathcal{H}} I(g(\mathbf{X}); Y)$*

(b) *Let*

$$T^* = \arg \min_{T \in \mathcal{T}} \mathbb{E}_{h(\mathbf{X})}[D_{\text{KL}}(p_{Y|h(\mathbf{X})}||p_{\nu_T})]$$

then it holds $h(\mathbf{X}) \perp T^(Y; h(\mathbf{X}))$*

Proof. (a) From equation 12, we obtain $-\mathbb{E}_{h(\mathbf{X}),Y}[\log p_{\nu_T}(Y|h(\mathbf{X}))] \geq H(Y|h(\mathbf{X}))$ for all $h \in \mathcal{H}, T \in \mathcal{T}$. We furthermore have $\min_{T \in \mathcal{T}} -\mathbb{E}_{h(\mathbf{X}),Y}[\log p_{\nu_T}(Y|h(\mathbf{X}))] = H(Y|h(\mathbf{X}))$ due to our assumptions on \mathcal{T} . Therefore, $\min_{h \in \mathcal{H}, T \in \mathcal{T}} -\mathbb{E}_{h(\mathbf{X}),Y}[\log p_{\nu_T}(Y|h(\mathbf{X}))] = \min_{h \in \mathcal{H}} H(Y|h(\mathbf{X}))$. Since we have $I(Y; h(\mathbf{X})) = H(Y) - H(Y|h(\mathbf{X}))$ and only the second term depends on h , statement (a) holds true.

(b) For convenience, we denote $T(Y; h(\mathbf{X})) = R$ and $h(\mathbf{X}) = Z$. We have $\mathbb{E}_Z[D_{\text{KL}}(p_{Y|Z}||p_{\nu_{T^*}})] = 0$ and therefore $p_{Y|Z}(y|z) = p_{\text{ref}}(T(y; z))|\nabla_y T^{-1}(y; z)|$.

Then it holds

$$\begin{aligned}
p_{R|Z}(r|z) &= p_{Y|Z}(T^{-1}(r; z)|z) \cdot |\nabla_y T^{-1}(r; z)| \\
&= p_{\text{ref}}(T(T^{-1}(r; z); z)) \cdot |\nabla_y T(y; z)| \cdot |\nabla_y T^{-1}(r; z)| \\
&= p_{\text{ref}}(r) \cdot 1
\end{aligned}$$

Since the density p_{ref} is independent of Z , we obtain $R \perp Z$ which concludes the proof of (b) \square

Statement (a) describes an optimization problem that allows to find features which share maximal information with the target variable Y . Due to statement (b) it is possible to draw samples from the conditional distribution $P(Y|h(\mathbf{X}))$ via the reference distribution.

Let \mathcal{H}_\perp the set of features which satisfy the invariance property, i.e. $Y \perp E|h(\mathbf{X})$ for all $h \in \mathcal{H}_\perp$. In the following, we sketch why $\arg \min_{h \in \mathcal{H}_\perp} \max_{e \in \mathcal{E}} \min_{T \in \mathcal{T}} \mathcal{L}_{\text{NLL}}^e(T; h) = \arg \max_{h \in \mathcal{H}_\perp} \min_{e \in \mathcal{E}} I(Y^e; h(\mathbf{X}^e))$ follows from Lemma 1.

Let $h \in \mathcal{H}_\perp$. Then, it is easily seen that there exists a $T^* \in \mathcal{T}$ with (1) $\mathcal{L}_{\text{NLL}}(T^*; h) = \min_{T \in \mathcal{T}} \mathcal{L}_{\text{NLL}}(T, h)$ and (2) $\mathcal{L}_{\text{NLL}}^e(T^*; h) = \min_{T \in \mathcal{T}} \mathcal{L}_{\text{NLL}}^e(T, h)$ for all $e \in \mathcal{E}$ since the conditional densities $p(y|h(\mathbf{X}))$ are invariant across all environments. Hence we have $H(Y^e|h(\mathbf{X}^e)) = \mathcal{L}_{\text{NLL}}^e(T^*; h)$ for all $e \in \mathcal{E}$. Therefore, $\arg \min_{h \in \mathcal{H}_\perp} \max_{e \in \mathcal{E}} \min_{T \in \mathcal{T}} \mathcal{L}_{\text{NLL}}^e(T; h) = \arg \max_{h \in \mathcal{H}_\perp} \min_{e \in \mathcal{E}} I(Y^e; h(\mathbf{X}^e))$ due to $I(Y^e; h(\mathbf{X}^e)) = H(Y^e) - H(Y^e|h(\mathbf{X}^e))$.

D HSIC AND WASSERSTEIN

The Hilbert-Schmidt Independence Criterion (HSIC) is a kernel based measure for independence which is in expectation 0 if and only if the compared random variables are independent (Gretton et al., 2005). An empirical estimate of $\text{HSIC}(A, B)$ for two random variables A, B is given by

$$\widehat{\text{HSIC}}(\{a_j\}_{j=1}^n, \{b_j\}_{j=1}^n) = \frac{1}{(n-1)^2} \text{tr}(K H K') \quad (13)$$

where tr is the trace operator. $K_{ij} = k(a^i, a^j)$ and $K'_{ij} = k'(b^i, b^j)$ are kernel matrices for given kernels k and k' . The matrix H is a centering matrix $H_{i,j} = \delta_{i,j} - 1/n$.

The one dimensional Wasserstein loss compares the similarity of two distributions (Kolouri et al., 2018). This loss has expectation 0 if both distributions are equal. An empirical estimate of the one dimensional Wasserstein loss for two random variables A, B is given by

$$\mathcal{L}_W = \|\text{sort}(\{a_j\}_{j=1}^n) - \text{sort}(\{b_j\}_{j=1}^n)\|_2$$

Here, the two batches are sorted in ascending order and then compared in the L2-Norm. We assume that both batches have the same size.

E ALGORITHM

In order to optimize the DG problem in equation 4, we optimize a normalizing flow T_θ and a feed forward neural network h_ϕ as described in Algorithm 1. There is an inherent trade-off between robustness and goodness-of-fit. The hyperparameter λ_I describes this trade-off and is chosen a priori.

Data: Samples from $P_{\mathbf{X}^e, Y^e}$ in different environments $e \in \mathcal{E}_{\text{seen}}$.

Initialize θ, ϕ ;

for number of training iterations **do**

for $e \in \mathcal{E}_{\text{seen}}$ **do**

 Sample minibatch $\{(y_1^e, \mathbf{x}_1^e), \dots, (y_m^e, \mathbf{x}_m^e)\}$ from $P_{Y, \mathbf{X}|E=e}$ for $e \in \mathcal{E}_{\text{seen}}$;

 Compute: $r_j^e = T_\theta(y_j^e; h_\phi(\mathbf{x}_j^e))$;

end

 Update θ, ϕ by descending alongside the stochastic gradient

$$\begin{aligned} \nabla_{\theta, \phi} \Big(\max_{e \in \mathcal{E}_{\text{seen}}} \Big\{ \sum_{i=1}^m \left[\frac{1}{2} \|T_\theta(y_i^e; h_\phi(\mathbf{x}_i^e))\|^2 - \log \nabla_y T_\theta(y_i^e; h_\phi(\mathbf{x}_i^e)) \right] \Big\} \\ + \lambda_I \mathcal{L}_I(\{r_j^e\}_{j,e}, \{h_\phi(\mathbf{x}_j^e), e\}_{j,e}) \Big); \end{aligned}$$

end

Result: In case of convergence, we obtain T_{θ^*}, h_{ϕ^*} with

$$\begin{aligned} \theta^*, \phi^* = \arg \min_{\theta, \phi} \Big(\max_{e \in \mathcal{E}_{\text{seen}}} \Big\{ \mathbb{E}_{\mathbf{X}^e, Y^e} \left[\frac{1}{2} \|T_\theta(Y^e; h_\phi(\mathbf{X}^e))\|^2 - \log \nabla_y T_\theta(Y^e; h_\phi(\mathbf{X}^e)) \right] \Big\} \\ + \lambda_I \mathcal{L}_I(P_R, P_{h_\phi(\mathbf{X}), E}) \Big) \end{aligned}$$

Algorithm 1: DG training with normalizing flows

If we choose a gating mechanisms h_ϕ as feature extractor similar to Kalainathan et al. (2018), then a complexity loss is added to the loss in the gradient update step.

In case we assume that the underlying mechanisms elaborates the noise in an additive manner, we could replace the normalizing flow T_θ with a feed forward neural network f_θ and execute Algorithm 2.

Data: Samples from $P_{\mathbf{X}^e, Y^e}$ in different environments $e \in \mathcal{E}_{\text{seen}}$.

Initialize θ ;

for number of training iterations **do**

for $e \in \mathcal{E}_{\text{seen}}$ **do**

 Sample minibatch $\{(y_1^e, \mathbf{x}_1^e), \dots, (y_m^e, \mathbf{x}_m^e)\}$ from $P_{Y, \mathbf{X}|E=e}$ for $e \in \mathcal{E}_{\text{seen}}$;

 Compute: $r_j^e = y_j^e - f_\theta(\mathbf{x}_j^e)$;

end

 Update θ by descending alongside the stochastic gradient

$$\nabla_\theta \left(\max_{e \in \mathcal{E}_{\text{seen}}} \left\{ \sum_{i=1}^m |r_j^e|^2 \right\} + \lambda_I \mathcal{L}_I(\{r_j^e\}_{j,e}, \{f_\theta(\mathbf{x}_j^e), e\}_{j,e}) \right);$$

end

Result: In case of convergence, we obtain f_{θ^*} with

$$\theta^* = \arg \min_{\theta} \left(\max_{e \in \mathcal{E}_{\text{seen}}} \left\{ \mathbb{E}_{\mathbf{X}^e, Y^e} [|Y^e - f_\theta(\mathbf{X}^e)|^2] \right\} + \lambda_I \mathcal{L}_I(P_R, P_{f_\theta(\mathbf{X}), E}) \right)$$

Algorithm 2: DG training under the assumption of additive noise

If we choose a gating mechanism, minor adjustments have to be made to Algorithm 2 such that we optimize 9. The classification case can be obtained similarly as described in 4.

F IDENTIFIABILITY RESULT

Under certain conditions on the environment and the underlying causal graph, the direct causes of Y become identifiable:

Proposition 1. *We assume that the underlying causal graph G is faithful with respect to $P_{\tilde{\mathbf{X}}, E}$. We further assume that every child of Y in G is also a child of E in G . A variable selection $h(\mathbf{X}) = \mathbf{X}_T$ corresponds to the direct causes if the following conditions are met: (i) $T(Y; h(\mathbf{X})) \perp E, h(\mathbf{X})$ are satisfied for a diffeomorphism $T(\cdot; h(\mathbf{X}))$, (ii) $h(\mathbf{X})$ is maximally informative about Y and (iii) $h(\mathbf{X})$ contains only variables from the Markov blanket of Y .*

Proof. Let $S(\mathcal{E}_{\text{seen}})$ denote a subset of \mathbf{X} which corresponds to the variable selection due to h . Without loss of generality, we assume $S(\mathcal{E}_{\text{seen}}) \subset \mathbf{M}$ where \mathbf{M} is the Markov Blanket. This assumption is reasonable since we have $Y \perp \mathbf{X} \setminus \mathbf{M} \mid \mathbf{M}$ in the asymptotic limit.

Since $pa(Y)$ cannot contain colliders between Y and E , we obtain that $Y \perp E \mid S(\mathcal{E}_{\text{seen}})$ implies $Y \perp E \mid (S(\mathcal{E}_{\text{seen}}) \cup pa(Y))$. This means using $pa(Y)$ as predictors does not harm the constraint in the optimization problem. Due to faithfulness and since the parents of Y are directly connected to Y , we obtain that $pa(Y) \subset S(\mathcal{E}_{\text{seen}})$.

For each subset $\mathbf{X}_T \subset \mathbf{X}$ for which there exists an $X_i \in \mathbf{X}_T \cap \mathbf{X}_{ch(Y)}$, we have $\mathbf{X}_T \not\perp Y \mid E$. This follows from the fact that X_i is a collider, in particular $E \rightarrow X_i \leftarrow Y$. Conditioning on X_i leads to the result that Y and E are not d -separated anymore. Hence, we obtain $Y \not\perp \mathbf{X}_T \mid E$ due to the faithfulness assumption. Hence, for each \mathbf{X}_T with $Y \perp E \mid \mathbf{X}_T$ we have $\mathbf{X}_T \cap \mathbf{X}_{ch(Y)} = \emptyset$ and therefore $\mathbf{X}_{ch(Y)} \cap S(\mathcal{E}_{\text{seen}}) = \emptyset$.

Since $\mathbf{X}_{pa(Y)} \subset S(\mathcal{E}_{\text{seen}})$, we obtain that $Y \perp \mathbf{X}_{pa(ch(Y))} \mid \mathbf{X}_{pa(Y)}$ and therefore the parents of $ch(Y)$ are not in $S(\mathcal{E}_{\text{seen}})$ except when they are parents of Y .

Therefore, we obtain that $S(\mathcal{E}_{\text{seen}}) = \mathbf{X}_{pa(Y)}$

□

One might argue that the conditions are very strict in order to obtain the true direct causes. But the conditions set in Proposition 1 are necessary if we do not impose additional constraints on the

true underlying causal mechanisms, e.g. linearity as done by Peters et al. (2016). For instance if $E \rightarrow X_1 \rightarrow Y \rightarrow X_2$, a model including X_1 and X_2 as predictor might be a better predictor than the one using only X_1 . From the Causal Markov Condition we obtain $E \perp Y \mid X_1, X_2$ which results in $X_1, X_2 \in S(\mathcal{E}_{\text{seen}})$. Under certain conditions however, the relation $Y \rightarrow X_2$ might be invariant across \mathcal{E} . This is for instance the case when X_2 is a measurement of Y . In this cases it might be useful to use X_2 for a good prediction.

G EXPERIMENTAL SETTING FOR SYNTHETIC DATASET

G.1 DATA GENERATION

In Section 5 we described how we choose different Structural Causal Models (SCM). In the following we describe details of this process.

We simulate the datasets in a way that the conditions in Proposition 1 are met. We choose different variables in the graph shown in Figure 1 as target variable. Hence, we consider different “topological” scenarios. We assume the data is generated by some underlying SCM. We define the structural assignments in the SCM as follows

$$\begin{aligned}
 \text{(a)} \quad f_i^{(1)}(\mathbf{X}_{pa(i)}, N_i) &= \sum_{j \in pa(i)} a_j X_j + N_i && \text{[Linear]} \\
 \text{(b)} \quad f_i^{(2)}(\mathbf{X}_{pa(i)}, N_i) &= \sum_{j \in pa(i)} a_j X_j - \tanh(a_j X_j) + N_i && \text{[Tanhshrink]} \\
 \text{(c)} \quad f_i^{(3)}(\mathbf{X}_{pa(i)}, N_i) &= \sum_{j \in pa(i)} \log(1 + \exp(a_j X_j)) + N_i && \text{[Softplus]} \\
 \text{(d)} \quad f_i^{(4)}(\mathbf{X}_{pa(i)}, N_i) &= \sum_{j \in pa(i)} \max\{0, a_j X_j\} + N_i && \text{[ReLU]} \\
 \text{(e)} \quad f_i^{(5)}(\mathbf{X}_{pa(i)}, N_i) &= \left(\sum_{j \in pa(i)} a_j X_j \right) \cdot (1 + (1/4)N_i) + N_i && \text{[Mult. Noise]}
 \end{aligned}$$

with $N_i \sim \mathcal{N}(0, c_i^2)$ where $c_i \sim \mathcal{U}[0.8, 1.2]$, $i \in \{0, \dots, 5\}$ and $a_i \in \{-1, 1\}$ according to Figure 6. Note that the mechanisms in (b), (c) and (d) are non-linear with additive noise and (e) elaborates the noise in a non-linear manner.

We consider *hard*- and *soft*-interventions on the assignments f_i . We either intervene on all variables except the target variable at once *or* on all parents and children of the target variable (Intervention Location). We consider three types of interventions:

- *Hard-Intervention* on X_i : Force $X_i \sim e_1 + e_2 \mathcal{N}(0, 1)$ where we sample for each environment $e_2 \sim \mathcal{U}([1.5, 2.5])$ and $e_1 \sim \mathcal{U}([0.5, 1.5] \cup [-1.5, -0.5])$
- *Soft-Intervention I* on X_i : Add $e_1 + e_2 \mathcal{N}(0, 1)$ to X_i where we sample for each environment $e_2 \sim \mathcal{U}([1.5, 2.5])$ and $e_1 \sim \mathcal{U}([0.5, 1.5] \cup [-1.5, -0.5])$
- *Soft-Intervention II* on X_i : Set the noise distribution N_i to $\mathcal{N}(0, 2^2)$ for $E = 2$ and to $\mathcal{N}(0, 0.2^2)$ for $E = 3$

Per run, we consider one environment without intervention ($E = 1$) and two environments with either both *soft*- or *hard*-interventions ($E = 2, 3$). We also create a fourth environment to measure a models’ ability for out-of-distribution generalization:

- *Hard-Intervention*: Force $X_i \sim e + \mathcal{N}(0, 4^2)$ where $e = e_1 \pm 1$ with e_1 from environment $E = 1$. The sign $\{+, -\}$ is chosen once for each i with equal probability.
- *Soft-Intervention I*: Add $e + \mathcal{N}(0, 4^2)$ to X_i where $e = e_1 \pm 1$ with e_1 from environment $E = 1$. The sign $\{+, -\}$ is chosen once for each i with equal probability as for the *do-intervention* case.

- *Soft-Intervention II*: Half of the samples have noise N_i distributed due to $\mathcal{N}(0, 1.2^2)$ and the other half of the samples have noise distributed as $\mathcal{N}(0, 3^2)$

We randomly sample causal graphs as described above. Per environment, we consider 1024 samples.

G.2 TRAINING DETAILS

All used feed forward neural networks have two internal layers of size 256. For the normalizing flows we use a 2 layer *MTA-Flow* described in Appendix G.3 with $K=32$. As optimizer we use Adam with a learning rate of 10^{-3} and a L2-Regularizer weighted by 10^{-5} for all models. Each model is trained with a batch size of 256. We train each model for 1000 epochs and decay the learning rate every 400 epochs by 0.5. For each model we use $\lambda_I = 1$ and the HSIC \mathcal{L}_I employs a Gaussian kernel with $\sigma = 1$. The gating architecture was trained without the complexity loss for 200 epochs and then with complexity loss weighted by 5. For the Flow model without gating architecture we use a feed forward neural network h_ϕ with two internal layers of size 256 mapping to an one dimensional vector. In total, we evaluated our models on 1365 created datasets as described in G.1.

Once the normalizing flow T is learned, we predict y given features $h(\mathbf{x})$ using 512 normally distributed samples u_i which are mapped to samples from $p(y|h(\mathbf{x}))$ by the trained normalizing flow $T(u_i, h(\mathbf{x}))$. As prediction we use the mean of these samples.

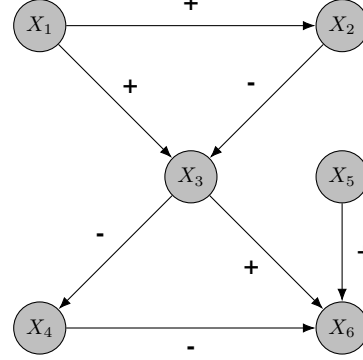


Figure 6: The signs of the coefficients a_j for the mechanisms of the different SCMs

G.3 ONE-DIMENSIONAL NORMALIZING FLOW

We use as one-dimension normalizing flow the *More-Than-Affine-Flow (MTA-Flow)*, which was developed by us. An overview of different architectures for one-dimensional normalizing flows can be found in Papamakarios et al. (2019). For each layer of the flow, a conditioner network C maps the conditional data $h(\mathbf{X})$ to a set of parameters $a, b \in \mathbb{R}$ and $\mathbf{w}, \mathbf{v}, \mathbf{r} \in \mathbb{R}^K$ for a chosen $K \in \mathbb{N}$. It builds the transformer τ for each layer as

$$z = \tau(y | h(\mathbf{X})) := a \left(y + \frac{1}{N(\mathbf{w}, \mathbf{v})} \sum_{i=1}^K w_i f(v_i y + r_i) \right) + b, \quad (14)$$

where f is any almost everywhere smooth function with a derivative bounded by 1. In this work we used a gaussian function with normalized derivative for f . The division by

$$N(\mathbf{w}, \mathbf{v}) := \varepsilon^{-1} \left(\sum_{i=1}^K |w_i v_i| + \delta \right), \quad (15)$$

with numeric stabilizers $\varepsilon < 1$ and $\delta > 0$, assures the strict monotonicity of τ and thus its invertibility $\forall x \in \mathbb{R}$. We also used a slightly different version of the *MTA-Flow* which uses the ELU activation function and – because of its monotonicity – can use a relaxed normalizing expression $N(\mathbf{w}, \mathbf{v})$.

G.4 PC-VARIANT

Since we are interested in the direct causes of Y , the widely applied PC-Algorithm gives not the complete answer to the query for the parents of Y . This is due to the fact that it is not able to orient all edges. To compare the PC-Algorithm we include the environment as system-internal variable and use a conservative assignment scheme where non-oriented edges are thrown away. This assignment scheme corresponds to the conservative nature of the ICP.

For further interest going beyond this work, we consider diverse variants of the PC-Algorithm. We consider two orientation schemes: A *conservative* one, where non-oriented edges are thrown away

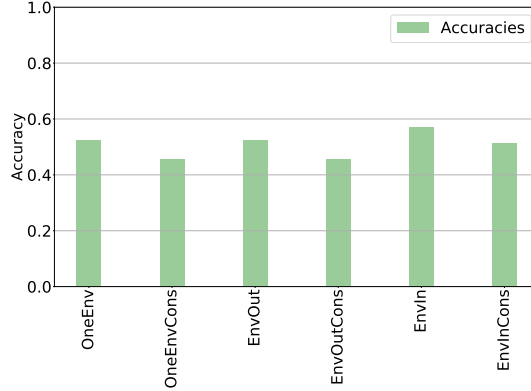


Figure 7: Detection accuracies of direct causes for different variants of the PC-Algorithm. EnvOut means we pool over all environments and EnvIn means the environment is treated as system intern variable E . The suffix Cons means we use the conservative assignment scheme. OneEnv means we only consider the observational environment for inference.

and a *non-conservative* one where non-oriented edges from a node X_i to Y are considered parents of Y .

We furthermore consider three scenarios: (1) the samples across all environments are pooled, (2) only the observational data (from the first environment) is given, and (3) the environment variable is considered as system-intern variable and is seen by the PC-Algorithm (similar as in Mooij et al. (2016)). Results are shown in Figure 7. In order to obtain these results, we sampled 1500 graphs as described above and applied on each of these datasets a PC-Variant. Best accuracies are achieved if we consider the environment variable as system-intern variable and use the non-conservative orientation scheme (EnvIn).

G.5 VARIABLE SELECTION

We consider the task of finding the direct causes of a target variable Y . Our models based on the gating mechanism perform a variable selection and are therefore compared to the PC-Algorithm and ICP. In the following we show the accuracies of this variable selection according to different scenarios.

Figure 8 shows the accuracies of ICP, the PC-Algorithm and our models pooled over all scenarios. Our models perform comparably well and better than the baseline in the causal discovery task.

In the following we show results due to different mechanisms, target variables, intervention types and intervention locations.

Figure 9b shows the accuracies of all models across different target variables. Parentless target variables, i.e. $Y = X_4$ or $Y = X_0$ are easy to solve for ICP due to its conservative nature. All our models solve the parentless case quite well. Performance of the PC-variant depends strongly on the position of the target variable in the SCM indicating that its conservative assignment scheme has a strong influence on its performance. As expected, the PC-variant deals well with $Y = X_6$ which is a childless collider. The causal discovery task seems to be particularly hard for variable $Y = X_6$ for all other models. This is the variable which has the most parents.

The type of intervention and its location seem to play a minor role as shown in Figure 9a and Figure 9a.

Figure 9b shows that ICP performs well if the underlying causal model is linear, but degrades if the mechanism become non-linear. The PC-Algorithm performs under all mechanisms comparably, but not well. ANMG performs quite well in all cases and even slightly better than FlowG in the cases of additive noise. However in the case of non-additive noise FlowG performs quite well whereas ANMG perform slightly worse – probably because their requirements on the underlying mechanisms are not met.

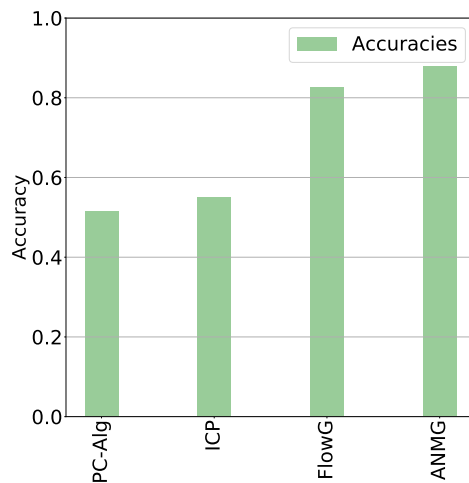
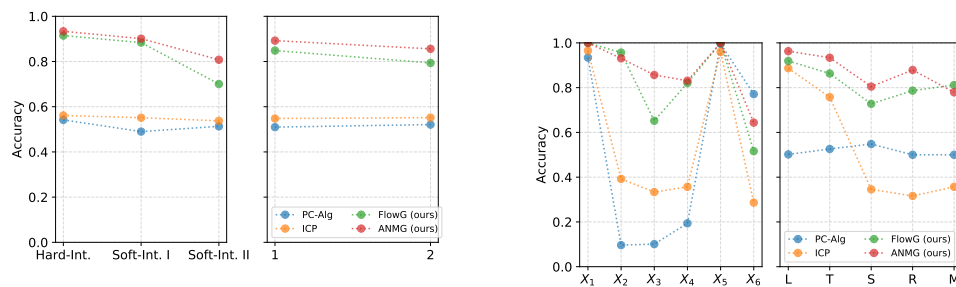


Figure 8: Accuracies for different models across all scenarios. FlowG and ANMG are our models.



(a) Accuracies of different models for different intervention types and locations. 1 stands for intervention on all variables except Y and 2 stands for interventions on parents and children only.

(b) Accuracies of different models according to target on all variables except Y and 2 stands for interventions on variables and mechanisms of the underlying SCM.

Figure 9: Comparison of models across different scenarios in the causal discovery task.

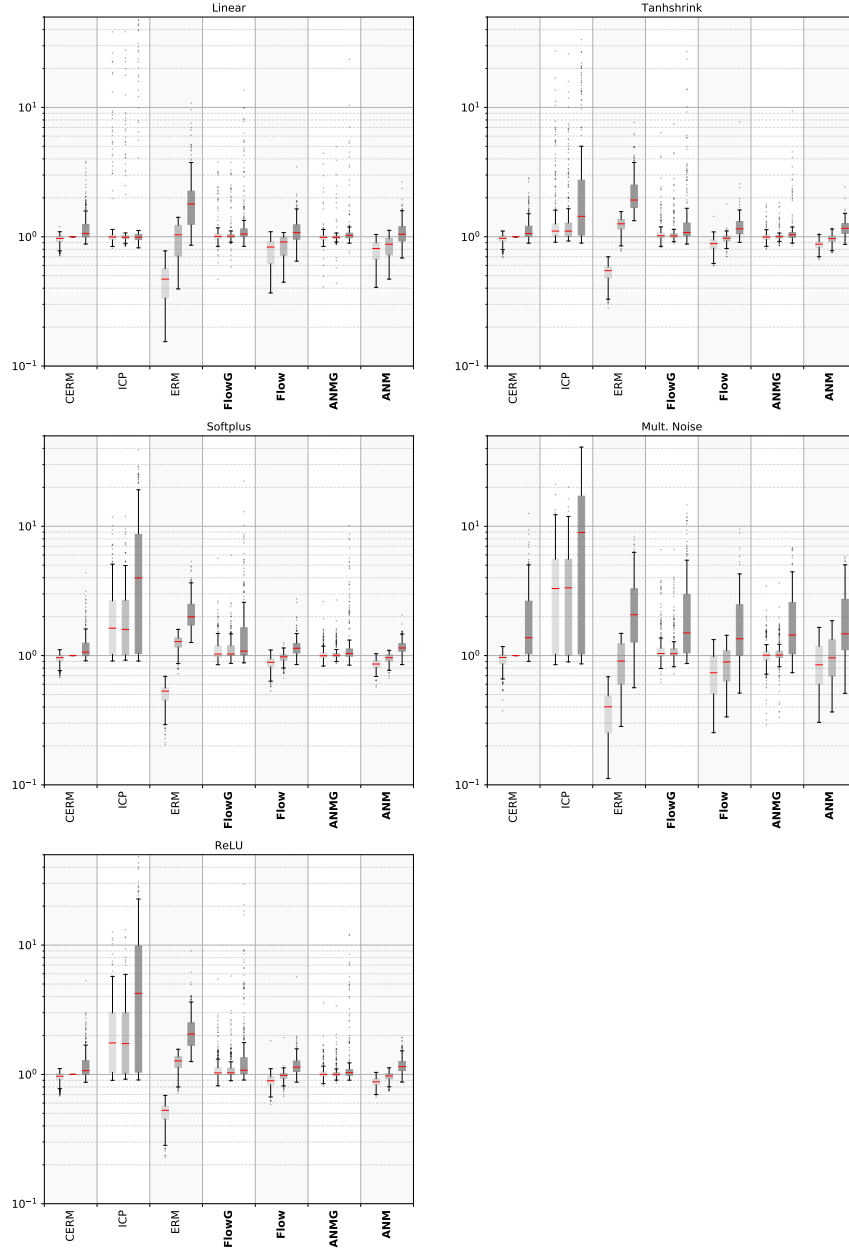


Figure 10: Logarithmic plot of L2 errors, normalized by CERM test error. For each method (ours in bold) from left to right: training error, test error on seen environments, domain generalization error on unseen environments. Scenarios for different mechanisms are shown.

G.6 TRANSFER STUDY

In the following we show the performance of different models on the training set, a test set of the same distribution and a set drawn from an unseen environment for different scenarios. As in Section 5, we use the L2-Loss on samples of an unseen environment to measure out-of-distribution generalization. Figure 10, 11 and 12 show results according to the underlying mechanisms, target variable or type of intervention respectively.

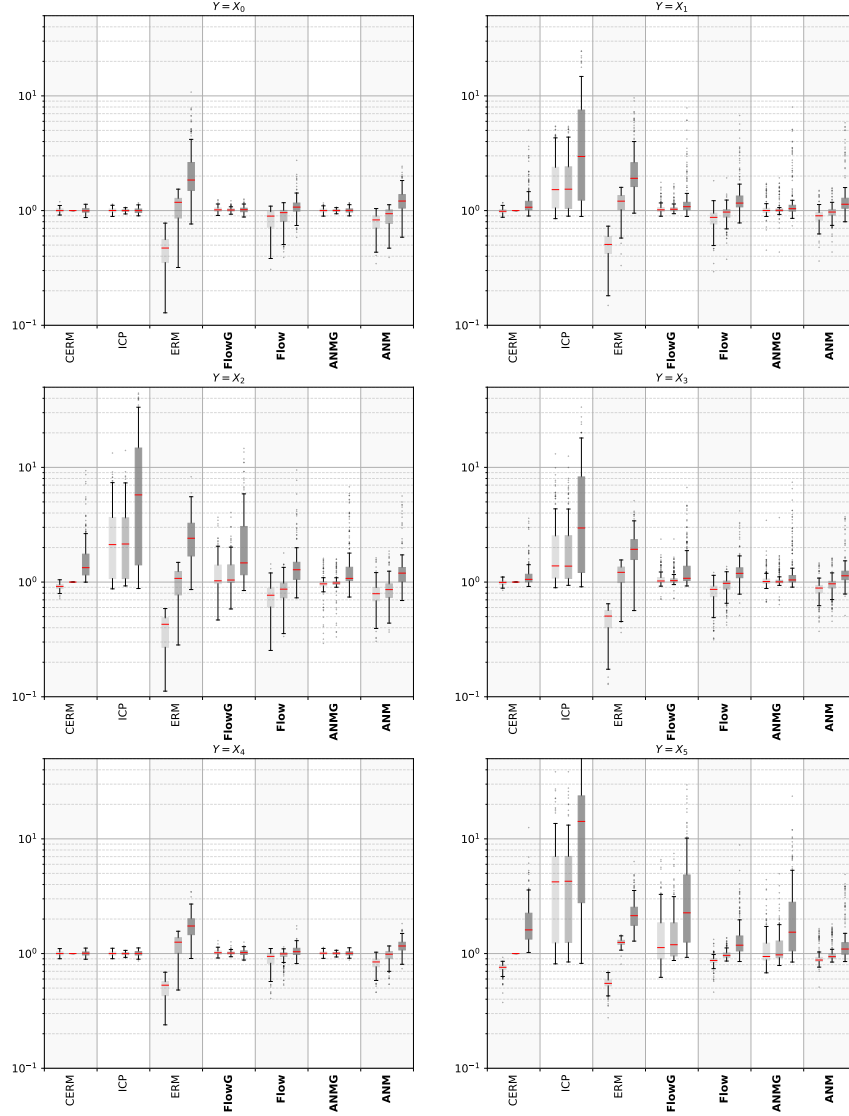


Figure 11: Logarithmic plot of L2 errors, normalized by CERM test error. For each method (ours in bold) from left to right: training error, test error on seen environments, domain generalization error on unseen environments. Scenarios for different target variables are shown.

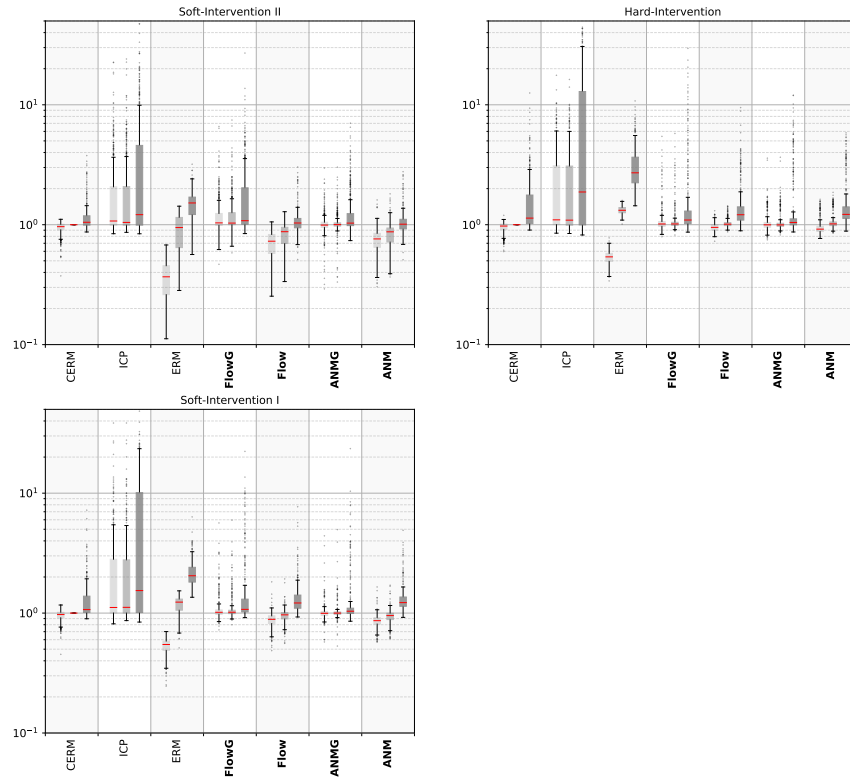


Figure 12: Logarithmic plot of L2 errors, normalized by CERM test error. For each method (ours in bold) from left to right: training error, test error on seen environments, domain generalization error on unseen environments. Scenarios for different intervention types are shown.

H EXPERIMENTAL DETAILS COLORED MNIST

For the training, we use a feed forward neural network consisting of a feature selector followed by a classifier. The feature selector consists of two convolutional layers with a kernel size of 3 with 16 respectively 32 channels followed by a max pooling layer with kernel size 2, one dropout layer and a fully connected layer mapping to 16 feature dimensions. After the first convolutional layer and after the pooling layer a ReLU activation function is applied. For the classification we use a ReLU activation function followed by a linear layer which maps the 16 features onto the two classes corresponding to the labels.

We use the dataset from Arjovsky et al. (2019). 50 000 samples are used for training and 10 000 samples as test set. For training, we choose a batch size of 2048 and train our models for 300 epochs. We choose a starting learning rate of $6 \cdot 10^{-3}$. The learning rate is decayed by 0.33 after 100 epochs. We use an L2-Regularization loss weighted by 10^{-5} . After each epoch we randomly reassign the colors and the labels with the corresponding probabilities. The one-dimensional Wasserstein loss is applied dimension-wise and the maximum over dimensions is computed in order to compare residuals. Since the output of the Wasserstein is very small, we multiplied it by a factor of 10. For the HSIC we use a gaussian kernel with $\sigma = 1$. For Figure 4 we trained our model with $\lambda_I \approx 1.585$.