NODAGS-Flow: Nonlinear Cyclic Causal Structure Learning

Muralikrishnna G. Sethuraman
Georgia Institute of Technology

Romain Lopez
Stanford University, Genentech

Rahul Mohan
Genentech

Faramarz Fekri
Georgia Institute of Technology

Tommaso Biancalani
Genentech

Jan-Christian Hütter
Genentech, Corresponding Author

Abstract

Learning causal relationships between variables is a well-studied problem in statistics, with many important applications in science. However, modeling real-world systems remain challenging, as most existing algorithms assume that the underlying causal graph is acyclic. While this is a convenient framework for developing theoretical developments about causal reasoning and inference, the underlying modeling assumption is likely to be violated in real systems, because feedback loops are common (e.g., in biological systems). Although a few methods search for cyclic causal models, they usually rely on some form of linearity, which is also limiting, or lack a clear underlying probabilistic model. In this work, we propose a novel framework for learning nonlinear cyclic causal graphical models from interventional data, called NODAGS-Flow. We perform inference via direct likelihood optimization, employing techniques from residual normalizing flows for likelihood estimation. Through synthetic experiments and an application to single-cell high-content perturbation screening data, we show significant performance improvements with our approach compared to state-of-the-art methods with respect to structure recovery and predictive performance.

1 INTRODUCTION

Understanding the causal relationships between interacting variables is a fundamental problem in science (Sachs et al., 2005; Zhang et al., 2013; Segal et al., 2005) since a causal, or mechanistic understanding is fundamental to correctly predict the effects of previously unobserved interventions on a system. Such systems can be modeled using a directed graph where each variable in the system is associated with a node and the edges represent causal relationships.

With a few notable exceptions (Hyttinen et al., 2012; Richardson, 1996; Mooij and Heskes, 2013; Bongers et al., 2016), most work on causal structure learning relies on the assumption that the underlying graph connecting the variables is a directed *acyclic* graph (DAG). This assumption facilitates the definition of a probability distribution over the observed variables for very general functional relationships. It also provides additional regularization to the estimation problem by narrowing down the class of graphs that are compatible with the observed probability distribution (the *Markov Equivalence Class*) (Richardson, 1996). However, there is compelling evidence that feedback loops are common in many real-world systems, such as those arising in gene-regulatory networks (Sachs et al., 2005; Freimer et al., 2022), violating the acyclicity assumption. These networks can however be probed with a large number of interventions through recent technological advances in biological assays building upon CRISPR/Cas9 and single-cell RNA-Sequencing (Dixit et al., 2016; Frangieh et al., 2021), alleviating the need for the additional regularization provided by the DAG constraint. Moreover, enforcing acyclicity necessitates searching for candidate solutions over large combinatorial search spaces, complicating algorithm design. Combined, this suggests that Cyclic Causal Graphs (CCG) should be better suited to model causal semantics in this regime.

In this work, we present a novel framework for causal discovery that does not rely on the DAG assumption, but instead allows for the presence of cycles in the underlying graph, while also modeling flexible, nonlinear relationships between the observed nodes. It is based on formulating the observation model as the steady state of a discrete dynamical system (Hyttinen et al., 2012). This elegantly allows for cycles in the underlying graph but comes at the cost of necessitating an evaluation of the gradient of the governing functional relationships in the likelihood evaluation.

Preprint Under Review.

We propose to employ the framework of normalizing flows (Papamakarios et al., 2021), in particular contractive residual flows (Behrmann et al., 2019; Chen et al., 2019), to deal with this complication. We provide a comparison of our framework, called NODAGS-Flow, to state-of-the-art structure learning methods on various graph recovery and prediction tasks.

After discussing related work (Section 2), we cover the problem setup including relevant background and modeling assumptions in Section 3. Then, we present the proposed NODAGS-Flow framework (Section 4), solving nonlinear cyclic causal discovery through likelihood optimization with contractive residual flows. Finally, we validate NODAGS-Flow on various synthetic benchmarks and on real-world data with genetic interventions (Section 5). Across the benchmarks, NODAGS-Flow beats state-of-the-art algorithms on nonlinear problems, even in the case when the underlying graph is acyclic, highlighting the practical benefits of NODAGS-Flow.

2 RELATED WORK

In causal discovery, the primary goal is to recover the underlying causal graph and the associated conditional probability distributions from observational and potentially interventional data. We next discuss previous approaches on acyclic and cyclic graphs, followed by the key contributions of our approach.

2.1 Acyclic causal discovery

Most causal discovery algorithms to date deal with the case of acyclic graphs, and they are commonly categorized into constraint-based, scored-based, and hybrid methods. Constraint-based methods such as the PC algorithm (Spirtes et al., 2000; Triantafillou and Tsamardinos, 2015; Heinze-Deml et al., 2018) aim to recover the underlying graph through constraints given by conditional independence relations encoded by causal graphs. Most constraint-based methods suffer from poor scalability and necessitate complicated algorithm design to handle the graphical constraints.

Score-based methods such as GES (Meek, 1997; Hauser and Bühlmann, 2012) learn the graph structure by optimizing a score function over candidate models. A popular choice of score function is given by the likelihood function in frequentist setups or the posterior likelihood in Bayesian formulations, and their regularized variants, such as the Bayesian Information Criterion (BIC). These methods often employ greedy approaches due to the super-exponential size of the search space.

More recently, the NOTEARS methodology (Zheng et al., 2018) introduced a continuous constraint for limiting the search space of the optimization problem to DAGs, closing the gap between DAG learning and continuous optimization and avoiding explicit greedy searches over combinatorial structures. Several extensions followed (Yu et al., 2019; Ng et al., 2020; Zheng et al., 2020; Lee et al., 2019; Brouillard et al., 2020) which allowed for learning DAGs under various assumptions on the underlying causal system. In particular, Brouillard et al. (2020) introduced a Gumbel-Softmax parameterization of the adjacency matrix and interventional masks which extended the method to handle imperfect interventions. While the NOTEARS framework strongly simplifies algorithm design for DAG learning, it necessitates sequentially solving multiple optimization problems, which poses difficulties in applying its nonlinear extensions to larger graphs without further regularization (Lopez et al., 2022).

Hybrid methods combine both previous approaches (Tsamardinos et al., 2006; Solus et al., 2017; Wang et al., 2017). Notably, one proposed hybrid approach (Khemakhem et al., 2021) used autoregressive normalizing flows for the underlying model, but strongly relied on the acyclicity assumption to fix an ordering and on constraint-based methods to estimate the skeleton of the underlying graph.

NODAGS-Flow is a score-based method and closest in spirit to the NOTEARS family of algorithms because it starts from a simple score function and is entirely based on continuous optimization. However, it does extend to the cyclic case and by doing so avoids the need for sequential optimization to handle the DAG constraint. In fact, in the presence of interventional data, we show in Sections 4.3 and 5.1 how it can beat the performance of NOTEARS and similar algorithms in the case where the underlying graph is a DAG.

2.2 Cyclic causal discovery

Cyclic causal discovery methods allow for feedback loops in the underlying causal mechanism, which complicates defining appropriate causal semantics. Early work on this topic extended constraint-based methods to this setting (Richardson, 1996), allowing the recovery of the underlying Markov Equivalence Class. However, exactly recovering cyclic graphs is more challenging than acyclic ones. For example, in the linear case, without resorting to assumptions such as faithfulness or sparsity, cyclic graphs are impossible to identify from purely observational data but can be consistently recovered when the interventions satisfy "pair conditions" for ordered pairs of nodes through the LLC algorithm (Hyttinen et al., 2012). A more thorough treatment of establishing causal semantics for cyclic models can be found in Bongers et al. (2016).

Other notable works are Huetter and Rigollet (2020) and Mooij and Heskes (2013), which use likelihood maximization for cyclic causal discovery, but they both are limited to

either the linear case or rely on a linear approximation of the causal mechanism around the mean of the data, respectively.

Related works that are more disconnected from the literature on causal discovery directly start with modeling causal or mechanistic relationships as arising from a dynamical system and have shown promise in modeling biological systems (Yuan et al., 2021; Nilsson et al., 2022). However, they lack a precise likelihood model.

Compared to these approaches, NODAGS-Flow provides a clearly defined and extensible likelihood model that handles nonlinear causal relationships.

2.3 Contributions

NODAGS-Flow endows the graph with semantics similar to those in Mooij and Heskes (2013) and Hyttinen et al. (2012), modeling the data as generated from the steady state of a dynamical system with an explicit noise model. However, instead of linear functional relationships, we allow for a rich class of nonlinear structural functions. Contrary to methods like NOTEARS (Zheng et al., 2018) and its nonlinear extensions which necessitate solving a series of optimization problems to deal with the acyclicity constraint, NODAGS-Flow consists of only a single optimization, thus significantly simplifying algorithm design. In particular, our model naturally extends the classical notion of a Structural Equation Model (SEM) (Bollen, 1989; Pearl, 2009) and subsumes DAG estimation in these models as a special case.

3 PROBLEM SETUP

3.1 Cyclic Causal Models via Structural Equations

Let $G = (V, E)$ represent a causal graph, where V, E denote the set of vertices and edges, respectively. Each vertex $v_i \in V$ has an associated random variable x_i corresponding to its observation and $x = (x_1, \dots, x_d)$ denotes the complete vector of observations. Following the framework proposed by Bollen (1989) and Pearl (2009), we use a *Structural Equation Model* (SEM), also known as *Structural Causal Model* (SCM), to represent the system. That is,

$$x_i = f_i(x_{\mathrm{pa}(i)}) + \varepsilon_i \quad i = 1, \dots, d, \tag{1}$$

where $\mathrm{pa}(i) \subseteq \{1, \dots, d\} \setminus \{i\}$ is the *parent* set of x_i, f_i encodes the functional dependence of x_i on its parents, also referred to as the *causal mechanism* of x_i. The parent-child relationships defined by the SEM encode the edges in G, i.e., the edge $x_j \to x_i$ exists if and only if $j \in \mathrm{pa}(i)$. The variables $(\varepsilon_1, \dots, \varepsilon_d)$ are known as the *disturbance* variables. By combining equation (1) over $i = 1, \dots, d$ and writing $f = (f_1, \dots, f_d)$, we have the following vectorized form:

$$x = f(x) + \varepsilon. \tag{2}$$

Additionally, the SEM also specifies a probability density $p_E(\varepsilon)$ over the disturbance variables. We assume that the system is free of *confounders*, that is, the disturbance variables are independent of each other. Finally, we define x as the solution to the system (1) for a random draw of ε.

In a classical SEM, the underlying graph is acyclic and a solution to (1) is naturally given by forward substitution along the topological ordering of the graph. Here, we instead explicitly assume that the mapping $x \mapsto \varepsilon = (\mathrm{id} - f)(x)$ is invertible, where id is the identity map, and that both $(\mathrm{id} - f)$ and $(\mathrm{id} - f)^{-1}$ are differentiable. This ensures that there is a unique x that corresponds to each disturbance vector ε. Under these conditions, the probability density of x is well-defined and can be obtained using the change of variable formula for density functions,

$$p_X(x) = p_E\big((\mathrm{id} - f)(x)\big) \big| \det J_{(\mathrm{id}-f)}(x) \big|, \tag{3}$$

where $J_{(\mathrm{id}-f)}$ denotes the Jacobian matrix of the function $(\mathrm{id} - f)$ evaluated at x.

3.2 Modeling Interventions

One of the key aspects of inferring causal models is the ability to predict the behavior of the system under interventions. Following Spirtes et al. (2000) and Pearl (2009), we consider surgical interventions, i.e., all incoming causal influences to the intervened-upon variables are removed. This results in a mutilated graph $\widetilde{G}$ where the intervened-upon nodes in G have no incoming edges. Following the notational convention of Hyttinen et al. (2012), we consider K interventional experiments and denote one such experiment by $\mathcal{E}_k = (\mathcal{I}_k, \mathcal{U}_k)$, where $\mathcal{I}_k$ is the set of intervened-upon nodes and $\mathcal{U}_k$ is the set of passively observed nodes. Let $\boldsymbol{U}_k \in \{0, 1\}^{d \times d}$ be a diagonal matrix such that $(\boldsymbol{U}_k)_{ii} = 1$ if and only if $v_i \in \mathcal{U}_k$. Under the interventional setting $\mathcal{E}_k$, the SEM now becomes

$$x = \boldsymbol{U}_k f(x) + \boldsymbol{U}_k \varepsilon + c, \tag{4}$$

where c denotes the value of the intervened-upon variables, i.e., $c_i = x_i$ if $i \in \mathcal{I}_k$ and 0 otherwise. Equation (4) corresponds to the assumption that the intervened-upon nodes are fixed and the equations for the passively observed variables remain unchanged. Similar to before, we assume that the functions $(\mathrm{id} - \boldsymbol{U}_k f)$ are invertible for all k. From equation (2), the density function x for experiment $\mathcal{E}_k$ is

$$\begin{aligned} p_X(x) = p_X(x_{\mathcal{I}_k}) p_E\big([(\mathrm{id} - \boldsymbol{U}_k f)(x)]_{\mathcal{U}_k}\big) \\ \big| \det J_{(\mathrm{id}-\boldsymbol{U}_k f)}(x) \big|, \end{aligned} \tag{5}$$

where $p_E\big([(\mathrm{id}-\boldsymbol{U}_k f)(x)]_{\mathcal{U}_k}\big)$ denotes subsetting the likelihood to only the variables in $\mathcal{U}_k$. Here, we assume surgical

interventions as introduced above and that the intervention targets are known.

Given a set of interventions, we would like to learn the underlying parent-child relations in the graph by maximizing the likelihood of the generated data. This requires the computation of $|\det J_{(\mathrm{id}-\boldsymbol{U}_k f)}(x)|$ to be tractable for each sample. To that end, we employ normalizing flows to model the map $x \mapsto \varepsilon$ (see Section 4.1). Normalizing flows provide a rich class of functions for which the Jacobian determinant is easily computable.

4 NODAGS-FLOW: RESIDUAL FLOW FOR CAUSAL LEARNING

In this section, we present the individual components of the NODAGS-Flow framework, namely contractive residual flows for calculating the log-det term, neural network architectures for modeling f, diagonal preconditioning to enable DAG learning, and finally the full score function that is being optimized. For ease of notation, in the following, we collect all model parameters into a single vector θ if not explicitly noted otherwise.

4.1 Contractive Residual Flows for Causal Learning

Residual flows are a class of invertible functions of the form

$$z' = z + g(z). \tag{6}$$

The name comes from the resemblance to the structure of residual networks (He et al., 2016). We note that solving (2) for ε, the relationship governing our causal model is $\varepsilon = x - f(x)$, which is of the same form as (6) with $g(z) = -f(z)$. To ensure that our model is well-defined, we need the invertibility of this transformation, along with invertibility for every possible intervention in (4). The Contractivity of f is one constraint that guarantees this invertibility. In the following, we outline the machinery introduced by Behrmann et al. (2019); Chen et al. (2019) to exploit this constraint for tractable generative modeling, which we adapt for structure learning.

A function $f : \mathbb{R}^d \to \mathbb{R}^d$ is said to be *contractive* if there exists a constant $L < 1$ such that for any two points $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathbb{R}^d$,

$$\|f(\boldsymbol{z}_1) - f(\boldsymbol{z}_2)\| \leq L\|\boldsymbol{z}_1 - \boldsymbol{z}_2\|.$$

It then follows from the Banach fixed point theorem (Rudin, 1953) that if f is contractive, then the residual transformation $\mathrm{id} - \boldsymbol{U}_k f$ is invertible for any masking matrix $\boldsymbol{U}_k$.

Although Banach's fixed point theorem guarantees invertibility, we have no analytical form for the inverse. However, the inverse can be obtained via fixed-point iterations. That is, starting with an arbitrary x_0, repeatedly compute $x_{k+1} = f(x_k) + \varepsilon$ for all $k > 0$. The Banach fixed point theorem guarantees that this procedure converges. Moreover, the rate of convergence is exponential in the number of iterations k and bounded by $O(L^k)$. This fixed-point iteration also provides an explicit interpretation of the causal semantics of the system in terms of a discrete dynamical system with fixed disturbances.

To efficiently approximate contractive functions and evaluate (5), two technical challenges remain: enforcing contractivity and evaluating the log-determinant of the Jacobian. To address the first, we employ neural networks to approximate f and note that a fixed Lipschitz constant can be enforced on a neural network layer by rescaling its weights by its spectral norm as shown by Behrmann et al. (2019) and Miyato et al. (2018). The composition of multiple such Lipschitz layers is still a Lipschitz function.

To address the second challenge, we employ the unbiased estimator of the log-det term introduced in Chen et al. (2019). Since f is contractive, by extending the power series expansion of $\log(1 + x)$ to matrices, we have

$$\begin{aligned}\log|\det J_{(\mathrm{id}-f)}(x)| &= \log|\det(\boldsymbol{I} - J_f(x))| \\ &= -\sum_{k=1}^{\infty} \frac{1}{k}\mathrm{Tr}\Big\{J_f^k(x)\Big\},\end{aligned} \tag{7}$$

where $\boldsymbol{I}$ denotes the identity matrix. The contractivity of f guarantees the convergence of the above series. The trace of $J_f^k(x)$ can be efficiently computed using the *Hutchinson trace estimator* (Hutchinson, 1989):

$$\mathrm{Tr}\Big\{J_f^k(x)\Big\} = \mathbb{E}_w[w^\top J_f^k(x) w], \tag{8}$$

where w is a random vector with zero mean and unit covariance. Behrmann et al. (2019) evaluate the above power series by truncating it to a finite number of terms. However, this approach has the drawback of being biased. Chen et al. (2019) improve on this by adding additional randomization to this evaluation, truncating the power series at a *random* cut-off $n \sim p(N)$, where p is a probability distribution over natural numbers $\mathbb{N}$, and re-weighting the terms in the power series to obtain an unbiased estimator. Hence the final estimator we use in NODAGS-Flow is now given by,

$$\log|\det J_{(\mathrm{id}-f)}(x)| = -\mathbb{E}_{n,w}\left[\sum_{k=1}^{n} \frac{w^\top J_f^k(x) w}{k \cdot P(N \geq k)}\right]. \tag{9}$$

Here, we choose $n \sim \mathrm{Poi}(N)$, a Poisson distribution with intensity N that we treat as a hyperparameter.

4.2 Parametrization and Sparsity Penalization

A first naïve implementation of the causal mechanism f through a Multi-Layer Perceptron (MLP) did not produce

promising results due to the presence of self-cycles (dependencies from a node v to itself). To address this, and to simultaneously add sparsity penalization on the dependency structure of f, we add a dependency mask $\boldsymbol{M}' \in \{0,1\}^{d\times d}$ with zero-diagonal that we apply via masking entries of x. That is, we introduce an MLP g_θ and set

$$[f_\theta(x)]_i = [g_\theta(\boldsymbol{M}'_{i,*} \odot x)]_i, \quad i = 1, \dots, d, \tag{10}$$

where $\odot$ denotes the Hadamard product. Similar to Brouillard et al. (2020) and Lopez et al. (2022), to enable efficient learning of $\boldsymbol{M}'$ during training, we model its entries as draws from a Gumbel-Softmax distribution $\boldsymbol{M}' \sim \boldsymbol{M}_\theta$ (Jang et al., 2016) with straight-through gradient estimation. Sparsity penalization is then achieved by adding $\lambda\, \mathbb{E}_{\boldsymbol{M}'\sim\boldsymbol{M}_\theta}[\|\boldsymbol{M}'\|_1]$ to the loss function for a regularization parameter $\lambda > 0$, where the expectation can be calculated explicitly from the parameters of $\boldsymbol{M}_\theta$. The Gumbel-Softmax parametrization $\boldsymbol{M}_\theta$ also offers access to an estimator for the underlying graph.

We note that in the special case of a 1-layer MLP, $f_\theta(x) = \sigma(\boldsymbol{W}^\top x)$ for a weight matrix $\boldsymbol{W}$ and an activation function σ, we can achieve the above more efficiently via enforcing a zero diagonal on $\boldsymbol{W}$ and direct ℓ_1-penalization on the entries of $\boldsymbol{W}$.

4.3 Extension to Non-Contractive DAGs via Preconditioning

Although contractivity is sufficient for the invertibility of $\mathrm{id} - f$, it is not a necessary condition. Indeed, for the case of DAGs, the causal mechanism f need not be contractive for $\mathrm{id} - f$ to be invertible, as the fixed point iterations will always converge after d steps. However, f being contractive is still convenient to efficiently estimate the absolute Jacobian-determinant as explained above. Via a diagonal rescaling (or preconditioning) of the model parameters, we can significantly increase the space of models that can be represented by contractive functions f. In particular, this includes all models whose underlying graph is a DAG, as shown in the following proposition.

Proposition 1 (Non-contractive to Contractive)**.** *Let (G, f) represent a causal DAG and its causal mechanism. If f is a non-contractive function, then there exists $\tilde{f}$ of the form $\tilde{f} = \Lambda \circ f \circ \Lambda^{-1}$, where Λ denotes multiplication with a diagonal matrix with positive diagonal entries such that $\tilde{f}$ is contractive.*

We refer to the appendix for proof of the above proposition. Proposition 1 allows us to rewrite the SEM purely in terms of a contractive function ($\tilde{f}$) and a diagonal matrix (Λ), when the underlying graph is a DAG. That is,

$$x = \Lambda^{-1} \circ \tilde{f} \circ \Lambda(x) + \varepsilon. \tag{11}$$

Hence, for a given observed set $\mathcal{U}_k$, the logarithm of the determinant of the Jacobian now becomes

$$\begin{aligned}
&\log|\det J_{\Lambda^{-1}\circ(I-\boldsymbol{U}_k\tilde{f})\circ\Lambda}| \\
&\quad= \log|\det\Lambda^{-1}| + \log|\det\Lambda| + \log|\det J_{(I-\boldsymbol{U}_k\tilde{f})}| \\
&\quad= \underbrace{\log|\det\Lambda| - \log|\det\Lambda|}_{=0} + \log|\det J_{(I-\boldsymbol{U}_k\tilde{f})}| \\
&\quad= \log|\det J_{(I-\boldsymbol{U}_k\tilde{f})}|,
\end{aligned} \tag{12}$$

which only depends on a contractive function and hence can be estimated efficiently using the procedure detailed in Section 4.1. In training the model, we treat Λ as a learnable parameter to be optimized via the log-likelihood function.

4.4 Score Function for Differentiable Causal Learning

Given a set of interventional experiments $\{\mathcal{E}_k\}_{k=1}^K$ and corresponding observations, we would like to learn the graph structure as well as the underlying functions governing the parent-child relations. To that end, similar to previous work (Brouillard et al., 2020; Lopez et al., 2022), we use the log-likelihood of the not-intervened-on nodes as a score function. That is, approximating the log-det term by (9), we consider

$$\begin{aligned}
&\mathcal{L}\Big(\theta, f_\theta, \{x^{(k,i)}\}_{k=1,i=1}^{M,N_k}\Big) = \\
&\quad\sum_{k=1}^{M}\sum_{i=1}^{N_k}\Bigg[\log p_{E,\theta}\Big([(\mathrm{id} - \boldsymbol{U}_k f_\theta)(x^{(k,i)})]_{\mathcal{U}_k}\Big) \\
&\qquad - \mathbb{E}_{n,w}\Bigg\{\sum_{r=1}^{n}\frac{w^\top\big[J^r_{\boldsymbol{U}_k f_\theta}\big(x^{(i,k)}\big)\big]w}{r\cdot P(N\geq r)}\Bigg\}\Bigg],
\end{aligned} \tag{13}$$

where $x^{(i,k)}$ denotes the i-th sample in the k-th experiment, and $p_{E,\theta}$ is parametrized as independent Gaussian distributions with learnable means and standard deviations. Together with ℓ_1 penalization introduced in Section 4.2 with parameter $\lambda > 0$ and the preconditioning in Section 4.3, inference is performed by solving the following optimization problem with stochastic optimization methods:

$$\max_{\theta,\Lambda} \mathcal{L}(\theta, \Lambda^{-1}\circ f_\theta \circ \Lambda) - \lambda\, \mathbb{E}_{\boldsymbol{M}'\sim\boldsymbol{M}_\theta}[\|\boldsymbol{M}'\|_1]. \tag{14}$$

5 EXPERIMENTS

We tested NODAGS-Flow on synthetic and real-world datasets. The performance of NODAGS-Flow is compared with some of the existing state-of-the-art causal discovery algorithms, LLC (Hyttinen et al., 2012) (linear & cyclic graphs), GOLEM (Ng et al., 2020) (linear and acyclic), NOTEARS (Zheng et al., 2018) (linear and acyclic), and DCDI (Brouillard et al., 2020) (nonlinear and acyclic). Of the chosen baselines, only DCDI and LLC are capable of handling interventional data out of the box, the other two

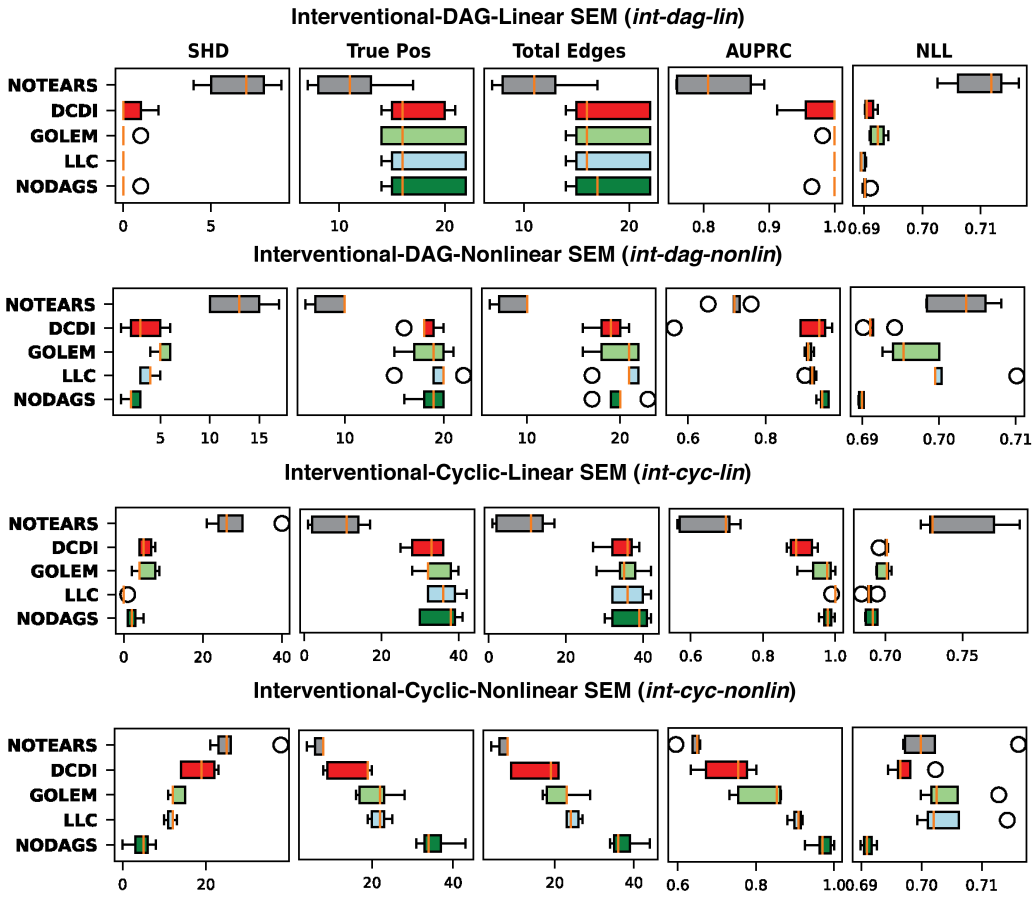

Figure 1: Performance on synthetic interventional data. The box plots show the median and inter-quartile ranges over the independent trails.

algorithms were modified to allow for learning from interventions by summing over different experimental regimes and masking out loss-terms corresponding to intervened-upon nodes as in (13).

5.1 Experiments on synthetic data

We considered both observational and interventional data for the synthetic datasets. The interventions were assumed to be perfect with known targets. Each dataset was generated from graphs with $d = 20$ nodes and for each intervention, 5000 observations were sampled. The observational data consists of 20,000 samples sampled from the graph. For the function f, we considered three different cases, namely (1) a linear function, $f(x) = \boldsymbol{W}^\top x$, (2) a nonlinear function, $f = \texttt{ReLU}(\boldsymbol{W}^\top x)$, a single-layer MLP with ReLU (rectified linear unit) activation, ensuring contractivity by rescaling by the operator norm, (3) a non-contractive nonlinear function, $f = \texttt{SELU}(\boldsymbol{W}^\top x)$, a single-layer MLP with SELU (Scaled Exponential Linear Unit) activation, with the underlying graph being a DAG.

In total we obtain 6 different settings for the synthetic experiments, summarized in Table 1. For the setting *int-dag-nonlin*, the causal mechanism f was taken from case (3) and for the settings *int-cyc-nonlin* and *obs-nonlin*, f was taken from case (2). The latent distribution $p_E(\varepsilon)$ was chosen as a Gaussian distribution with the same variances. The graphs were generated using an Erdős-Rényi random graph model with an expected edge density of 2, allowing for cycles, whereas in case (3), we ensured acyclicity by creating a causal order and ensuring that the parents for each node always come from its predecessor in the causal order. The weight matrices were sampled from the uniform distribution, with post-scaling to ensure that the

Table 1: Synthetic experiment settings.

Setting	Interventions	SEM	Cyclic
int-dag-lin	True	Linear	False
int-dag-nonlin	True	Nonlinear	False
int-cyc-lin	True	Linear	True
int-cyc-nonlin	True	Nonlinear	True
obs-lin	False	Linear	False
obs-nonlin	False	Nonlinear	False

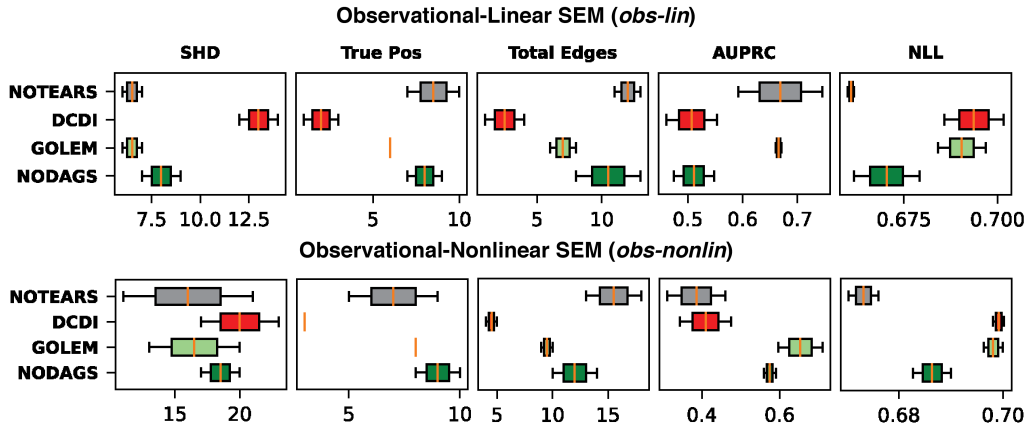

Figure 2: Performance on synthetic observational data. The box plots show the median and inter-quartile ranges over the independent trails.

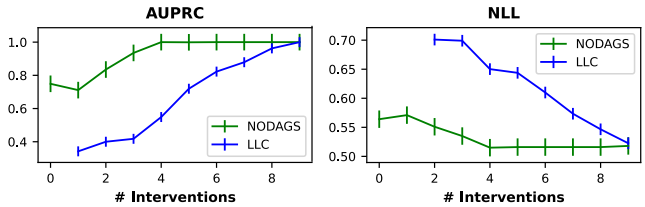

Figure 3: Performance comparison between LLC and NODAGs-Flow as the number of interventions used for training the model is increased from 0 to 9 on a 10-node graph.

overall function is contractive for the first two settings.

Performance evaluation The performance was evaluated with respect to the following metrics: (1) *Structural Hamming Distance* (SHD), (2) Total number of *True Positive* edges (True Pos) predicted, (3) *Total Edges* edges predicted, (4) *Area Under Precision-Recall Curve* (AUPRC), and (5) holdout *Interventional-NLL* (NLL) (Gentzel et al., 2019), the negative log-likelihood over unseen interventions. SHD, True Pos, Total Edges, and AUPRC measure the accuracy of the recovered graph structure whereas NLL measures the predictive power of the model over unseen interventions. For the interventional data sets, the training data consisted of single-node interventions across all nodes. For both the interventional and observational data sets, the test set consisted of interventions on two or three nodes (random with equal probability), randomly sampled from the total set of nodes.

The results of the synthetic experiments are reported in Figures 1 and 2. In both figures, the box plots show the median and the inter-quartile ranges over independent trials. In Figures 1 and 2, each column shows the performance with respect to the metric stated at the top of the column for the different settings shown in Table 1. On linear interventional data (Figure 1, *int-dag-lin* and *int-cyc-lin*), NODAGS-Flow attains comparable performance to that of LLC (which was specifically designed for the interventional linear case), both in terms of graph structure recovery and the prediction of unseen interventions. In *int-dag-lin*, GOLEM and NOTEARS match the performance of LLC and NODAGS-Flow as the setting is well specified for these models. As expected GOLEM and DCDI drop in performance when cycles are introduced (Figure 1, *int-cyc-lin*).

On non-linear interventional data (Figure 1), NODAGS-Flow performs the best when the graph contains cycles (*int-cyc-nonlin*) followed by LLC. When the causal graph is a DAG and the causal mechanism non-contractive (Figure 1, *int-dag-nonlin*), the added learnable parameter Λ allows NODAGS-Flow to learn a non-contractive function by rescaling a contractive function f by Λ. In this case, NODAGS-Flow and DCDI are the best performing models. This highlights the benefits of a larger, potentially simpler search space for structure learning provided by our approach compared to specifically enforcing DAG constraints.

On the other hand, in the observational setting, due to the inherent identifiability issues caused by not confining the search space to DAGs, NODAGS-Flow trails behind the other methods enforcing a DAG constraint, see Figure 2. We exclude LLC in Figure 2 as it is incapable of handling purely observational data.

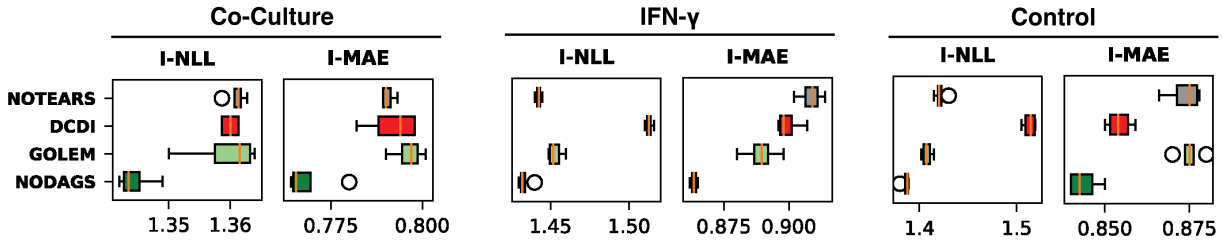

Figure 4: Performance comparison on Perturb-CITE-seq Frangieh et al. (2021) data. The box plots show the median and inter-quartile ranges over the independent trails.

Scaling with Interventions In the previous experiments, we ensured that the models were provided with interventions over all the single nodes in the graph. Here, we test the model's capability to learn the graph structure with limited interventional information. We compare NODAGS-Flow with LLC as we increase the number of interventions provided during training in the case of a linear contractive SEM.

From Figure (3) we can see that NODAGS-Flow requires significantly fewer interventions compared to LLC as it attains close to perfect structure recovery around 4 interventions on a $d = 10$ node graph. It is also important to note that LLC cannot work on purely observational data as it subsets the data according to the performed interventions, whereas NODAGS-Flow can handle both observational and interventional data out of the box.

5.2 Experiment on Real-World Transcriptomics Data

Here, we present an experiment focused on learning a gene regulatory network from gene expression data with genetic interventions (Perturb-seq), a type of dataset that allows to causally investigate biology at an unprecedented scale. Recent advances (Dixit et al., 2016) have made it possible to perform such genetic interventions at large scales (in the order of hundreds or thousands of genes (Replogle et al., 2022)) and be able to measure the effect of full gene expression profile on the order of hundreds of thousands of cells.

We focus on a Perturb-CITE-seq (Frangieh et al., 2021) dataset that investigated drivers of resistance to Immune Checkpoint Inhibitors (ICI). It contains gene expressions taken from 218,331 melanoma cells split over three different conditions, namely: (1) control (57,627 cells) , (2) co-culture (73,114 cells), and (3) interferon (IFN)-γ (87,590 cells). Each measurement contains the identity of the target genes and the expression profiles of each gene in the genome.

Due to practical and computational limitations, we restrict our experiment to a subset of 61 genes out of approximately 20,000 genes in the genome. For interventions, we chose all the single-gene interventions corresponding to the 61 chosen genes. Each condition is considered a separate dataset and we train NODAGS-Flow and the baselines on these datasets separately. Since there is no ground-truth DAG available, we evaluate our model based on its predictive power on unseen interventions. To that end, we perform 5 splits on each of the datasets into 90% training and 10% test interventions. Interventional NLL (`I-NLL`) and the Interventional *Mean Absolute Error* (`I-MAE`) were used as metrics to evaluate the models. `I-MAE` was calculated as the mean of $\|f(x) - x\|_1/d$ over all observations x in a hold-out dataset.

From Figure (4) we can see that NODAGS-Flow outperforms all the baselines with respect to both metrics. Of all the baselines LLC seemed to attain the worst performance, we, therefore, discuss the performance comparison with LLC in more detail in the appendix. This shows that learning cycles in the graph allow for better learning of the underlying distribution and thereby improve the predictive power of the model. Figure 5 shows the cluster map obtained from the adjacency matrix learned by NODAGS-Flow on the Co-culture partition of the Perturb-CITE-seq datasets.

6 DISCUSSION

We proposed NODAGS-Flow, a novel causal discovery approach that is capable of learning nonlinear and cyclic relations between variables through a simple optimization framework, avoiding optimization problems with complex constraints such as NOTEARS. Experiments on synthetic interventional data showed matching performance with state-of-the-art methods (LLC) on linear data and superior performance when recovering nonlinear relationships, both in the case of cyclic and acyclic causal graphical models.

We also presented an application of our approach on real-world gene expression data with genetic interventions (Perturb-CITE-seq), where we learned a gene-regulatory network on 61 genes. NODAGS-Flow was able to achieve better predictive performance on unseen interventions through an interpretable, mechanistic model that allows for feedback loops. We hope that applications on more biological datasets could enhance understanding of

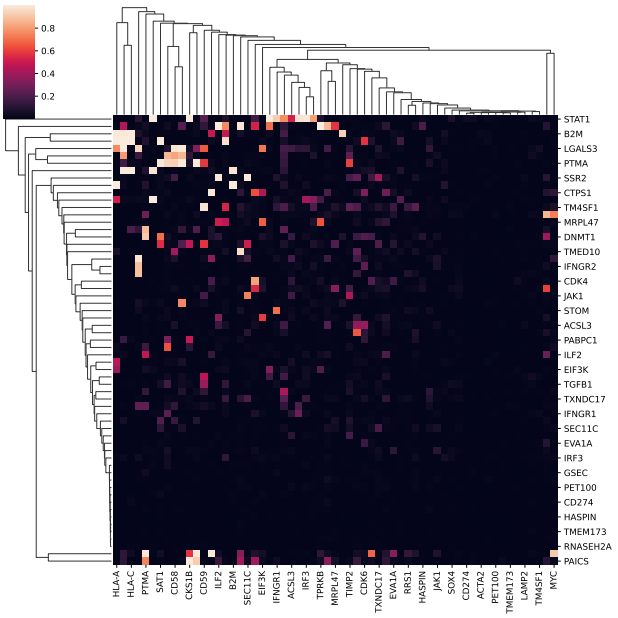

Figure 5: Adjacency matrix of the graph learned by NODAGS-Flow on the Perturb-CITE-seq data set (Co-culture).

transcriptomic regulation and aid in the design of novel perturbations.

Interesting potential extensions to our model include (1) incorporating more realistic measurement noise models, which have been shown to significantly affect the quality of transcriptomic machine-learning tools (Grün et al., 2014; Lopez et al., 2018), (2) explore imperfect interventions and the case where the intervention targets are unknown which is quite common in biological settings, and (3) scaling up the model to handle larger graphs, potentially by incorporating ideas from low-rank models (Segal et al., 2005; Lopez et al., 2022).

7 Acknowledgments

We would like to thank Kathryn Geiger-Schuller and Oana Ursu for helpful suggestions on the pre-processing and gene selection for our experiments on transcriptomics data.

References

Ambrosio, L., Fusco, N., and Pallara, D. (2000). *Functions of bounded variation and free discontinuity problems*. Courier Corporation.

Bakshy, E., Dworkin, L., Karrer, B., Kashin, K., Letham, B., Murthy, A., and Singh, S. (2018). Ae: A domain-agnostic platform for adaptive experimentation. In *Conference on Neural Information Processing Systems*, pages 1–8.

Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. (2019). Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR.

Bollen, K. A. (1989). *Structural equations with latent variables*, volume 210. John Wiley & Sons.

Bongers, S., Peters, J., Schölkopf, B., and Mooij, J. M. (2016). Theoretical aspects of cyclic structural causal models. *arXiv preprint arXiv:1611.06221*.

Brouillard, P., Lachapelle, S., Lacoste, A., Lacoste-Julien, S., and Drouin, A. (2020). Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 33:21865–21877.

Chen, R. T., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. (2019). Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32.

Dixit, A., Parnas, O., Li, B., Chen, J., Fulco, C. P., Jerby-Arnon, L., Marjanovic, N. D., Dionne, D., Burks, T., Raychowdhury, R., Adamson, B., Norman, T. M., Lander, E. S., Weissman, J. S., Friedman, N., and Regev, A. (2016). Perturb-Seq: Dissecting Molecular Circuits with Scalable Single-Cell RNA Profiling of Pooled Genetic Screens. *Cell*, 167(7):1853–1866.e17.

Frangieh, C. J., Melms, J. C., Thakore, P. I., Geiger-Schuller, K. R., Ho, P., Luoma, A. M., Cleary, B., Jerby-Arnon, L., Malu, S., Cuoco, M. S., et al. (2021). Multimodal pooled Perturb-CITE-seq screens in patient models define mechanisms of cancer immune evasion. *Nature genetics*, 53(3):332–341.

Freimer, J. W., Shaked, O., Naqvi, S., Sinnott-Armstrong, N., Kathiria, A., Garrido, C. M., Chen, A. F., Cortez, J. T., Greenleaf, W. J., Pritchard, J. K., and Marson, A. (2022). Systematic discovery and perturbation of regulatory genes in human T cells reveals the architecture of immune networks. *Nature Genetics*, pages 1–12.

Gentzel, A., Garant, D., and Jensen, D. (2019). The case for evaluating causal models using interventional measures and empirical data. *Advances in Neural Information Processing Systems*, 32.

Grün, D., Kester, L., and Van Oudenaarden, A. (2014). Validation of noise models for single-cell transcriptomics. *Nature Methods*, 11(6):637–640.

Hauser, A. and Bühlmann, P. (2012). Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *The Journal of Machine Learning Research*, 13(1):2409–2464.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heinze-Deml, C., Peters, J., and Meinshausen, N. (2018). Invariant causal prediction for nonlinear models. *Journal of Causal Inference*, 6(2).

Huetter, J.-C. and Rigollet, P. (2020). Estimation rates for sparse linear cyclic causal models. In Peters, J. and Sontag, D., editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 1169–1178. PMLR.

Hutchinson, M. F. (1989). A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076.

Hyttinen, A., Eberhardt, F., and Hoyer, P. O. (2012). Learning linear cyclic causal models with latent variables. *The Journal of Machine Learning Research*, 13(1):3387–3439.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*.

Khemakhem, I., Monti, R., Leech, R., and Hyvarinen, A. (2021). Causal autoregressive flows. In *International Conference on Artificial Intelligence and Statistics*, pages 3520–3528. PMLR.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Lee, H.-C., Danieletto, M., Miotto, R., Cherng, S. T., and Dudley, J. T. (2019). Scaling structural learning with NO-BEARS to infer causal transcriptome networks. In *Pacific Symposium on Biocomputing 2020*, pages 391–402. World Scientific.

Lopez, R., Hütter, J.-C., Pritchard, J. K., and Regev, A. (2022). Large-scale differentiable causal discovery of factor graphs. In *Advances in Neural Information Processing Systems*.

Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., and Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058.

Meek, C. (1997). *Graphical Models: Selecting causal and statistical models*. PhD thesis, Carnegie Mellon University.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Mooij, J. M. and Heskes, T. (2013). Cyclic causal discovery from continuous equilibrium data. In *Uncertainty in Artificial Intelligence*.

Ng, I., Ghassami, A., and Zhang, K. (2020). On the role of sparsity and DAG constraints for learning linear dags. *Advances in Neural Information Processing Systems*, 33:17943–17954.

Nilsson, A., Peters, J. M., Meimetis, N., Bryson, B., and Lauffenburger, D. A. (2022). Artificial neural networks enable genome-scale simulations of intracellular signaling. *Nature Communications*, 13(1):1–16.

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. *J. Mach. Learn. Res.*, 22(57):1–64.

Pearl, J. (2009). *Causality*. Cambridge University Press, 2 edition.

Replogle, J. M., Saunders, R. A., Pogson, A. N., Hussmann, J. A., Lenail, A., Guna, A., Mascibroda, L., Wagner, E. J., Adelman, K., Lithwick-Yanai, G., et al. (2022). Mapping information-rich genotype-phenotype landscapes with genome-scale perturb-seq. *Cell*.

Richardson, T. (1996). A discovery algorithm for directed cyclic graphs. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 454–461.

Rudin, W. (1953). *Principles of Mathematical Analysis*. McGraw-Hill Book Company, Inc., New York-Toronto-London.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

Segal, E., Pe'er, D., Regev, A., Koller, D., Friedman, N., and Jaakkola, T. (2005). Learning module networks. *Journal of Machine Learning Research*, 6(4).

Solus, L., Wang, Y., Matejovicova, L., and Uhler, C. (2017). Consistency guarantees for permutation-based causal inference algorithms. *arXiv preprint arXiv:1702.03530*.

Spirtes, P., Glymour, C. N., Scheines, R., and Heckerman, D. (2000). *Causation, prediction, and search*. MIT press.

Triantafillou, S. and Tsamardinos, I. (2015). Constraint-based causal discovery from multiple interventions over overlapping variable sets. *The Journal of Machine Learning Research*, 16(1):2147–2205.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78.

Wang, Y., Solus, L., Yang, K., and Uhler, C. (2017). Permutation-based causal inference algorithms with interventions. *Advances in Neural Information Processing Systems*, 30.

Yu, Y., Chen, J., Gao, T., and Yu, M. (2019). DAG-GNN: DAG structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR.

Yuan, B., Shen, C., Luna, A., Korkut, A., Marks, D. S., Ingraham, J., and Sander, C. (2021). Cellbox: interpretable machine learning for perturbation biology with application to the design of cancer combination therapy. *Cell systems*, 12(2):128–140.

Zhang, B., Gaiteri, C., Bodea, L.-G., Wang, Z., McElwee, J., Podtelezhnikov, A. A., Zhang, C., Xie, T., Tran, L., and Dobrin, R. (2013). Integrated systems approach identifies genetic nodes and networks in late-onset Alzheimer's disease. *Cell*, 153(3):707–720.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). DAGs with NO TEARS: Continuous optimization for structure learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P., and Xing, E. (2020). Learning sparse nonparametric DAGs. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, pages 3414–3425.

Appendix

The appendix is organized as follows: in Appendix A we present the proof of Proposition 1 followed implementation details and further details regarding the real-world experiment in appendix B and appendix C respectively.

A PROOFS

A.1 Proof of Proposition 1

In this section, we present a detailed proof of Proposition 1.

Proposition 2 (Non-contractive to Contractive)**.** *Let (G, f) represent a causal DAG and its causal mechanism. If $f : \mathbb{R}^d \to \mathbb{R}^d$, is an L-Lipschitz function, then there exists $\tilde{f}$ of the form $\tilde{f} = \Lambda \circ f \circ \Lambda^{-1}$, where Λ denotes multiplication with a diagonal matrix with positive diagonal entries (also denoted by Λ) such that $\tilde{f}$ is contractive.*

Proof. For ease of notation, we assume that f is everywhere differentiable and denote its Jacobian by J_f. The general case can be proved similarly since Lipschitz functions are almost everywhere differentiable by Rademacher's theorem (Ambrosio et al., 2000).

Without loss of generality, we assume that the graph G is topologically sorted along the indices $i = 1, \dots, d$. If not, we can rearrange the dimensions of f accordingly. With this, the Jacobian J_f is a strictly lower triangular matrix. For a desired contractivity constant $0 < c < 1$, we recursively define the entries of the diagonal matrix $\Lambda \in \mathbb{R}^{d \times d}$ as follows:

$$\begin{aligned} \Lambda_{d,d} &= 1, \\ \Lambda_{i,i} &= \frac{d^2 L}{c} \max_{j>i} \Lambda_{j,j}, \quad \text{for } i < d. \end{aligned} \tag{15}$$

Defining $\tilde{f} = \Lambda \circ f \circ \Lambda^{-1}$, we have that $J_{\tilde{f}} = \Lambda J_f \Lambda^{-1}$. For any $x \in \mathbb{R}^d$, the (i, j)-th entry of $J_{\tilde{f}}$ is therefore given by

$$(J_{\tilde{f}}(x))_{i,j} = \frac{\Lambda_{i,i}}{\Lambda_{j,j}} (J_f(\Lambda^{-1} x))_{i,j}. \tag{16}$$

Let $y = \Lambda^{-1} x$. Since $\|z\|_1 \leq \sqrt{d}\|z\|_2$ by the Cauchy-Schwarz inequality and $\|z\|_2 \leq \|z\|_1$ for all $z \in \mathbb{R}^d$, we obtain

$$|(J_f(y))_{i,j}| \leq \sup_{z:\|z\|_1 \leq 1} \|J_f(y)[z]\|_1 \leq \sup_{z:\|z\|_2 \leq 1} \sqrt{d}\|J_f(y)[z]\|_2 \leq \sqrt{d}\|J_f(y)\|_{\text{op}} \leq \sqrt{d}L. \tag{17}$$

In turn, the entries of $J_{\tilde{f}}$ can be bounded as follows: for $i > j$, by combining (17) with the definition of Λ (15), we obtain

$$|(J_{\tilde{f}}(x))_{i,j}| = \frac{\Lambda_{i,i}}{\Lambda_{j,j}} |(J_f(y))_{i,j}| \leq \frac{1}{\Lambda_{j,j}} \sqrt{d} L \max_{k \geq i} \Lambda_{k,k} \leq \frac{c}{d^{3/2}}. \tag{18}$$

For $i \leq j$, since J_f and therefore $J_{\tilde{f}}$ are strictly lower triangular by definition, $(J_{\tilde{f}}(x))_{i,j} = 0$.

Finally, applying similar reasoning as in (17) and the bound in (18), we obtain a bound on the operator norm of $J_{\tilde{f}}$,

$$\left\| J_{\tilde{f}}(x) \right\|_{\text{op}} = \sup_{z:\|z\|_2 \leq 1} \|J_{\tilde{f}}(x)[z]\|_2 \leq \sup_{z:\|z\|_1 \leq \sqrt{d}} \|J_{\tilde{f}}(x)[z]\|_1 = \sqrt{d} \max_j \sum_{i=1}^{d} |(J_{\tilde{f}}(x))_{i,j}| \leq d^{3/2} \frac{c}{d^{3/2}} = c < 1.$$

This concludes the proof, showing that $\tilde{f}$ is contractive. □

B IMPLEMENTATION DETAILS

In this section, we present the implementation details of NODAGS-Flow, the baselines as well as the setup of the experiments.

Table 2: Hyperparameter spaces for all the models.

	Hyperparameter space
NODAGS-Flow	$\log_{10}(\lambda) \in [-4, 2]$ # hidden units $\in \{0, 1, 2, 3\}$ $n_L \in \{5, 10, 15\}$
LLC	$\log_{10}(\lambda) \in [-4, 2]$
GOLEM	$\log_{10}(\lambda) \in [-4, 2]$ $\lambda_{DAG} \in [-3, 3]$
NOTEARS	$\log_{10}(\lambda) \in [-4, 2]$
DCDI	$\log_{10}(\lambda) \in [-4, 2]$

B.1 NODAGS-Flow

We consider the parametric family of neural networks (NN), denoted as g_θ, to model the causal function f. As detailed in section 4.2 of the main paper, the dependency structure (parent-child relations) is encoded by introducing a dependency mask $\boldsymbol{M'} \in \{0, 1\}^{d \times d}$ in the model. $\boldsymbol{M'}$ is then used to mask the inputs for each node as shown in equation (10) of the main paper. For the neural network architecture, we fix each hidden layer to have the same number of neurons ($= d$) and vary the number of hidden layers in the model. If the data is nonlinear we add a `ReLU` activation function to each layer of the neural network, allowing NODAGS-Flow to learn nonlinear parent-child relations.

In order to maintain the contractivity of the neural network, the weights of each layer are rescaled by its spectral norm, similar to Behrmann et al. (2019) and Miyato et al. (2018). This is done every time the weights are updated, that is, after every backward pass. In practice, we choose the Lipschitz constant of the neural network to be 0.9. For computing the log Jacobian determinant, we sample the number of terms in the power series from a Poisson distribution with parameter σ initialized to 2. Additionally, σ is treated as a parameter to be learned during training. During the training stage, we use the Neumann gradient series formulation of the log Jacobian determinant estimator in Behrmann et al. (2019) as this provides a more efficient way for backpropagation over the entries of the Jacobian matrix. Whereas in the validation stage we use the standard estimator for the log Jacobian determinant. The number of hidden layers, regularization parameter λ, and the number of terms used for computing the spectral norm of the weights (n_L) are treated as parameters to be tuned.

B.2 Baseline Methods

We now provide the implementation details of the baselines used in our experiments. The LLC algorithm proposed by Hyttinen et al. (2012) was reimplemented and a sparse regularization term was added in order to make LLC solve the same objective as NODAGS-Flow and the other baselines. In accordance with the method proposed by Hyttinen et al. (2012) we assume that the intervened nodes are independent and sampled from the standard normal distribution. Of the other baselines used only DCDI (Brouillard et al., 2020) supports interventional data out of the box. Hence NOTEARS (Zheng et al., 2018) and GOLEM (Ng et al., 2020) were reimplemented along the lines of Lopez et al. (2022). For NOTEARS, DCDI, and GOLEM, we threshold the adjacency matrices (probability of an edge for DCDI) with a threshold t obtained by performing a binary search with $T = 20$ evaluations of an acyclicity test to find the largest possible DAG from the estimated weights matrix.

B.3 Hyperparameter Tuning

For all methods, an exhaustive hyperparameter search was performed using the `Ax` library (Bakshy et al., 2018) that can perform joint Bayesian and bandit optimization over the set of hyperparameters. The list of chosen hyperparameters for each model is summarized in Table 2. `Ax` samples from the range of values provided for each parameter, the models are then trained using the sampled parameters and are then evaluated on the validation set. For the synthetic experiments, the training set consists of single-node interventions over all the nodes in the graph and the validation set consists of interventions over 2-3 randomly chosen nodes. For the Perturb-CITE-seq data set, we randomly split 10% of the interventions to be a part of the validation set and the rest with the training set. We perform separate hyperparameter tuning for the synthetic and the real-world experiments and use the optimal parameter values provided by `Ax` for the respective experiments. Additionally, we fix the learning rate to 10^{-2} and use Adam optimizer (Kingma and Ba, 2015) for maximizing the log-likelihood.

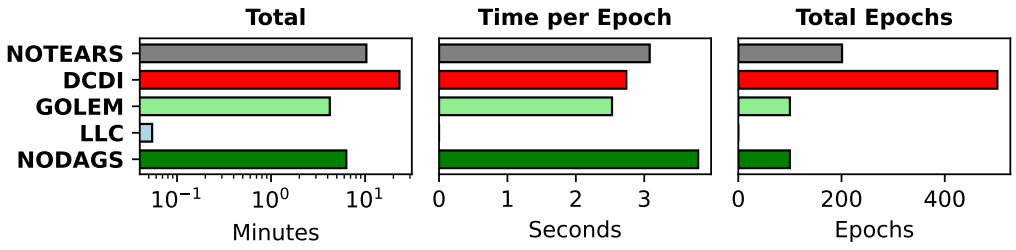

Figure 6: Comparison of the runtime of the chosen

B.4 Compute Time analysis

In Figure 6, we compare the runtime between NODAGS-Flow and the chosen baselines. It is important to note that, LLC (unlike NODAGS-Flow, DCDI, GOLEM, and NOTEARS) is not deep-learning-based method and hence doesn't require any training via stochastic gradient methods. Instead, it solves simple linear regression problems from the estimated covariance matrices for each intervention. This makes LLC considerably faster than the other algorithms as seen in Figure 6. Additionally, due to the lack of training LLC is excluded from Time per Epoch and Total Epochs plots. Aside from LLC, we can see that the other methods are comparable in terms of total runtime and runtime per epoch. In particular, the log-det approximation necessary in every epoch of NODAGS-Flow renders the computational cost per epoch for NODAGS-Flow the highest. However, NODAGS-Flow is overall faster than the only other method relying on a nonlinear mechanism, DCDI, by a factor of more than 3.5 due to not relying on solving a constrained optimization problem via the Augmented Lagrangian Method.

B.5 Code Statement

We implemented our models using the PyTorch library in Python and plan to make it publicly available on GitHub upon publication.

C REAL-WORLD EXPERIMENT

The data set was downloaded from the Single Cell Portal of the Broad Institute (accession code SCP1064). We removed cells containing less than 500 expressed genes and genes that were expressed in less than 500 cells. Due to computational constraints, we chose a subset of 61 genes (Table 3) from the total set of genes in the genome, ensuring that all the chosen genes were perturbed. The three different conditions (co-culture, IFN-γ, and control) were partitioned into distinct data sets. The models were trained and evaluated on each of the three data sets. Figures 8 and 9 show the cluster map of the learnt adjacency matrices for the IFN-γ and control datasets respectively.

Table 3: The list of chosen genes from Perturb-CITE-seq dataset (Frangieh et al., 2021).

ACSL3	ACTA2	B2M	CCND1	CD274	CD58	CD59	CDK4	CDK6
CDKN1A	CKS1B	CST3	CTPS1	DNMT1	EIF3K	EVA1A	FKBP4	FOS
GSEC	GSN	HASPIN	HLA-A	HLA-B	HLA-C	HLA-E	IFNGR1	IFNGR2
ILF2	IRF3	JAK1	JAK2	LAMP2	LGALS3	MRPL47	MYC	P2RX4
PABPC1	PAICS	PET100	PTMA	PUF60	RNASEH2A	RRS1	SAT1	SEC11C
SINHCAF	SMAD4	SOX4	SP100	SSR2	STAT1	STOM	TGFB1	TIMP2
TM4SF1	TMED10	TMEM173	TOP1MT	TPRKB	TXNDC17	VDAC2		

C.1 NODAGS-Flow vs. LLC

Figure 7 shows the performance comparison between NODAGS-Flow and LLC. It can be seen that NODAGS-Flow is able to outperform LLC with respect to both the evaluation metrics and across all three conditions. This shows that learning nonlinear relations does indeed provide an advantage, but we also attribute LLC's poor performance to the mismatch in LLC's treatment of the intervened nodes to its actual behavior. That is, LLC considers the intervened nodes to be

independent with zero mean and unit variance, which is not the case in the data set and hence the significantly worse performance compared to the other baselines.

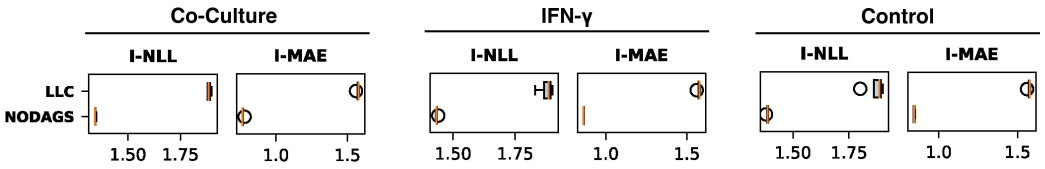

Figure 7: Performance comparison between NODAGS-Flow and LLC on Perturb-CITE-seq data set.

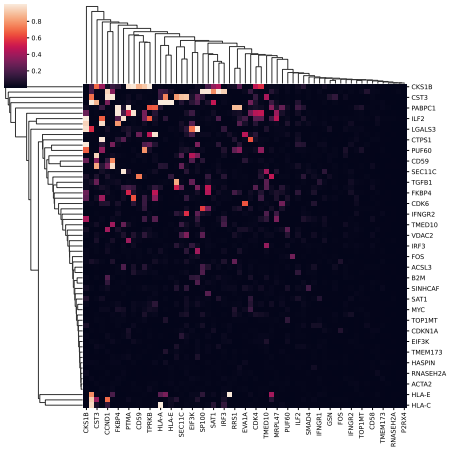

Figure 8: Clustermap of the adjacency matrix learned by NODAGS-Flow on the IFN-γ datasets.

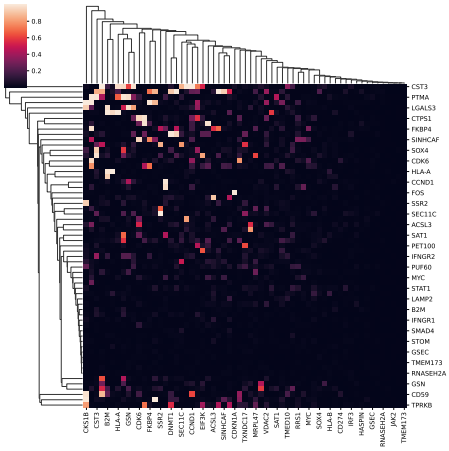

Figure 9: Clustermap of the adjacency matrix learned by NODAGS-Flow on the control datasets