Sparsity in Continuous-Depth Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural Ordinary Differential Equations (NODEs) have proven successful in learning dynamical systems in terms of accurately recovering the observed trajectories. While different types of sparsity have been proposed to improve robustness, the generalization properties of NODEs for dynamical systems beyond the observed data are underexplored. We systematically study the influence of weight and feature sparsity on forecasting as well as on identifying the underlying dynamical laws. Besides assessing existing methods, we propose a regularization technique to sparsify "input-output connections" and extract relevant features during training. Moreover, we curate real-world datasets consisting of human motion capture and human hematopoiesis single-cell RNA-seq data to realistically analyze different levels of out-of-distribution (OOD) generalization in forecasting and dynamics identification respectively. Our extensive empirical evaluation on these challenging benchmarks suggests that weight sparsity improves generalization in the presence of noise or irregular sampling. However, it does not prevent learning spurious feature dependencies in the inferred dynamics, rendering them impractical for predictions under interventions, or for inferring the true underlying dynamics. Instead, feature sparsity can indeed help with recovering sparse ground-truth dynamics compared to unregularized NODEs.

1 Introduction

Extreme over-parameterization has shown to be part of the success story of deep neural networks (Zhang et al., 2016; Allen-Zhu et al., 2018; Li & Liang, 2018). This has been linked to evidence that over-parameterized models are easier to train, perhaps because they develop "convexity-like" properties that help convergence of gradient descent (Brutzkus et al., 2017; Du et al., 2018). However, over-parameterization comes at the cost of additional computational footprint during training and inference (Denil et al., 2013). Therefore, motivation for sparse neural networks are manifold and include: (i) computational efficiency (speed up, memory reduction, scalability) (Hoefler et al., 2021), (ii) imitation of human learning, where neuron activity is typically sparse (Ahmad & Hawkins, 2016), as well as (iii) interpretability (Xu et al., 2022), and (iv) avoiding overfitting or improving robustness (Bartoldson et al., 2020).

Enforcing sparsity in model weights, also called model pruning, and its effects for standard predictive modelling tasks have been extensively studied in the literature (Hoefler et al., 2021). However, sparsity in continuous-depth neural nets for modeling dynamical systems and its effects on generalization properties is still underexplored. Neural Ordinary Differential Equations (NODEs) (Chen et al., 2018) have been introduced as the limit of taking the number of layers in residual neural networks to infinity, resulting in continuous-depth (or continuous-time) neural nets. These networks have shown great promise for modeling noisy and irregular time-series. In this case, the trained NODE model might not represent just *some* dynamics serving the predictive task (e.g., regression or classification via a final linear layer). It can rather be thought of as learning *the underlying* dynamics (i.e., the ODE) that actually govern(s) the evolution of the observed trajectory. In the prior scenario (predictive

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

performance), a large body of literature has focused on regularizing the number of model evaluations required for a single step of the ODE solver for improving the efficiency Finlay et al. (2020); Kelly et al. (2020); Ghosh et al. (2020); Pal et al. (2021); Grathwohl et al. (2019). These regularization techniques rely on learning one out of many possible equivalent dynamical systems giving rise to the same ultimate performance, but are easier and faster to solve. Crucially, these regularizers do not explicitly target sparsity in the model weights or in the features used by the model.

In the latter scenario (inferring dynamical laws), we assume the existence of a ground truth ODE that governs the dynamics of the observed trajectories. Here, two types of sparsity may be of interest. First, motivated by standard network pruning, sparsity in model weights (*weight sparsity* or *model sparsity*) can reduce computational requirements for inference (Liebenwein et al., 2021), see Figure 1(top). Second, it has been argued that *feature sparsity*, i.e., reducing the number of inputs a given output of the NODE relies on, can improve identifiability of the underlying dynamical law (Aliee et al., 2021; Bellot et al., 2021; Liu et al., 2022), see Figure 1(bottom). The motivation for feature sparsity often relies on an argument from causality, where it is common to assume sparse and modular causal dependencies between variables (Schölkopf et al., 2021; Zheng et al., 2019).

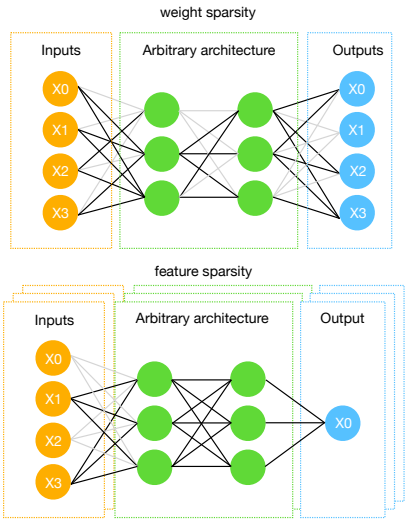


Figure 1: Model vs feature sparsity

In this paper, we empirically assess the impact of various types of sparsity-enforcing methods on the performance of NODEs on different types of OOD generalization for both prediction and for inferring dynamical laws on real-world datasets. While most existing methods target weight sparsity, we highlight that feature sparsity can improve interpretability and dynamical system inference by proposing a new regularization technique for continuous-time models. Specifically, our main contributions are the following:

- We propose PathReg, a differentiable L0-based regularizer that enforces sparsity of "input-output paths" and leads to both feature and weight sparsity.
- We extend L0 regularization (Louizos et al., 2018) and LassoNet (Lemhadri et al., 2021) to NODEs and perform extensive comparisons with existing models including vanilla NODE (Chen et al., 2018), C-NODE (Aliee et al., 2021), GroupLasso for NODEs (Bellot et al., 2021), as well as our own PathReg.
- We define a useful metric for evaluating feature sparsity, demonstrate the differences between weight and feature sparsity, and explore whether and when one implies the other.
- To cover at least part of the immense diversity of time-series problems, each of which implying different types of OOD challenges, we curate large, real-world datasets consisting of human motion capture (`mocap.cs.cmu.edu`) as well as human hematopoiesis single-cell RNA-seq (Luecken et al., 2021) data for our empirical evaluations. We will release these datasets openly and hope they benefit further research on inferring dynamical laws from observed trajectories.

2 Background

2.1 Continuous-depth neural nets

Among the plethora of deep learning based methods to estimate dynamical systems from data (Lim & Zohren, 2021), we focus on Neural Ordinary Differential Equations (NODEs). In NODEs, a neural network with parameters θ is used to learn the function $f_\theta \approx f$ from data, where f defines a (first order) ODE in its explicit representation $\dot{X} = f(X, t)$ (Chen et al., 2018). Starting from the initial observation $X(a)$ at some time $t = a$, an explicit iterative ODE solver is deployed to predict $X(t)$ for $t \in (a, b]$ using the current derivative estimates from f_θ. The parameters θ are then updated via backpropagation on the mean squared error (MSE) between predictions and observations. As discussed extensively in the literature, NODEs can outperform traditional ODE parameter inference techniques in terms of reconstruction error, especially for non-linear dynamics f (Chen et al., 2018;

Dupont et al., 2019). In particular, one advantage of NODEs over previous methods for inferring non-linear dynamics such as SINDy (Brunton et al., 2016) is that no dictionary of non-linear basis functions has to be pre-specified.

A variant of NODEs for second order systems called SONODE exploits the common reduction of higher-order ODEs to first order systems (Norcliffe et al., 2020). The second required initial condition, the initial velocity $\dot{X}(a)$, is simply learned from $X(a)$ via another neural network in an end-to-end fashion. Our experiments build on the public NODE and SONODE implementations. However, our framework is readily applicable to most other continuous-depth neural nets including Augmented NODE (Dupont et al., 2019), latent NODEs (Rubanova et al., 2019), and neural stochastic DEs (Li et al., 2020; Oganesyan et al., 2020).

2.2 Sparsity and generalization

Two prominent motivations for enforcing sparsity in over-parameterized deep learning models are (a) pruning large networks for efficiency (speed, memory), (b) as a regularizer that can improve interpretability or prevent overfitting. In both cases, one aims at preserving (as much as possible) i.i.d. generalization performance, i.e., performance on unseen data from the same distribution as the training data, compared to a non-sparse model (Hoefler et al., 2021).

Another line of research has explored the link between generalizability of neural nets and causal learning (Schölkopf et al., 2021), where generalization outside the i.i.d. setting, is conjectured to require an underlying *causal model*. Deducing true laws of nature purely from observational data could be considered an instance of inferring a causal model of the world. Learning the correct causal model enables accurate predictions not only on the observed data (next observation), but also under distribution shifts for example under interventions (Schölkopf et al., 2021). A common assumption in causal modeling is that each variable depends on (or is a function of) *few* other variables (Schölkopf et al., 2021; Zheng et al., 2019; Ng et al., 2020; Bolstad et al., 2011). In the ODE context, we interpret the variables (or their derivatives) that enter a specific component f_i as causal parents, such that we can write $\dot{X}_i = f(pa(X_i), t)$ (Mooij et al., 2013). Thus, feature sparsity translates to each variable X_i having only few parents $pa(X_i)$, which can also be interpreted as asking for "simple" dynamics. Since weight sparsity as well as regularizing the number of function evaluations in NODEs can also be considered to bias them towards "simpler dynamics", these notions are not strictly disjoint, raising the question whether one implies the other.

In terms of feature sparsity, Aliee et al. (2021); Bellot et al. (2021) study *system identification* and causal structure inference from time-series data using NODEs. They suggest that enforcing sparsity in the number of causal interactions improves parameter estimation as well as predictions under interventions. Let us write out f_θ as a fully connected net with L hidden layers parameterized by $\theta := (W^l, b^l)_{l=1}^{L+1}$ as

$$f_\theta(X) = W^{L+1}\sigma(\ldots\sigma(W^2\sigma(W^1X + b^1) + b^2)\ldots) \tag{1}$$

with element-wise activation function σ, l-th layer weights W^l, and biases b^l. Aliee et al. (2021) then seek to reduce the overall number of parents of all variables by attempting to cancel all contributions of a given input on a given output node through the neural net. In a linear setting, where $\sigma(x) = x$, the regularization term is defined by[1]

$$\|A\|_{1,1} = \|W^{L+1}\ldots W^1\|_{1,1} \tag{2}$$

where $A_{ij} = 0$ if and only if the i-th output is constant in the j-th input. In the non-linear setting, for certain $\sigma(x) \neq x$, the regularizer $\|A\|_{1,1} = \||W^{L+1}|\cdots|W^1|\|_{1,1}$ (with entry wise absolute values on all W^l) is an upper bound on the number of input-output dependencies, i.e., for each output i summing up all the inputs j it is not constant in. Regularizing input gradients (Ross et al., 2017; Ross & Doshi-Velez, 2017; Slavin, 2018) is another alternative to train neural networks that depend on fewer inputs, however it is not scalable to high-dimensional regression tasks.

Bellot et al. (2021) instead train a separate neural net $f_{\theta_i} : \mathbb{R}^n \to \mathbb{R}$ for each variable X_i and penalize NODEs using GroupLasso on the inputs via

$$\sum_{k,i=1}^{n} \|[W_i^1]_{\cdot,k}\|_2 \tag{3}$$

[1]The $1,1$ norm $\|W\|_{1,1}$ is the sum of absolute values of all entries of W.

where W_i^1 is the weight matrix in the input layer of f_i and $[W_i^1]_{\cdot,k}$ refers to the kth column of W_i^1 that should simultaneously (as a group) be set to zero or not. While enforcing strict feature sparsity (instead of regularizing an upper bound), parallel training of multiple NODE networks can be computationally expensive (Figure 1, bottom). While the findings of both works suggest that sparsity of causal interactions helps system identification, their empirical evaluation predominantly focuses on synthetic data settings, leaving performance on real data underexplored.

Another recent work suggests that standard weight or neuron pruning improves generalization for NODEs (Liebenwein et al., 2021). The authors show that pruning lowers empirical risk in density estimation tasks and decreases Hessian's eigenvalues, thus obtaining better generalization (flat minima). However, the effect of sparsity on identifying the underlying dynamics as well as the generalization properties of NODEs to forecast future values are not assessed.

3 Sparsification of neural ODEs

In an attempt to combine the strengths of both weight and feature sparsity, we propose a new regularization technique, called PathReg, which compares favorably to both C-NODE (Aliee et al., 2021) and GroupLasso (Bellot et al., 2021). Before introducing PathReg, we describe how to extend existing methods to NODEs for an exhaustive empirical evaluation.

3.1 Methods

L0 regularization. Inspired by (Louizos et al., 2018), we use a *differentiable* L0 norm regularization method that can be incorporated in the objective function and optimized via stochastic gradient descent. The L0 regularizer prunes the network during training by encouraging weights to get *exactly zero* using a set of non-negative stochastic gates z. For an efficient gradient-based optimization, Louizos et al. (2018) propose to use a continuous random variable s with distribution $q(s)$ and parameters ϕ, where z is then given by

$$s \sim q(s \mid \phi), \qquad z = \min(1, \max(0, s)) \,. \tag{4}$$

Gate z is a hard-sigmoid rectification of s^2 that allows the gate to be *exactly* zero. The regularization term is then defined as the probability of s being positive

$$q(z \neq 0 \mid \phi) = 1 - Q(s \leq 0 \mid \phi) \,, \tag{5}$$

where Q is the cumulative distribution function of s. Minimizing the regularizer pushes many gates to be zero, which implies weight sparsity as these gates are multiplied with model weights. L0 regularization can be added directly to the NODE network f_θ.

LassoNet is a feature selection method (Lemhadri et al., 2021) that uses an input-to-output skip (residual) connection that allows a feature to participate in a hidden unit only if its skip connection is active. LassoNet can be thought of as a residual feed-forward neural net $Y = \mathcal{S}^T X + h_\theta(X)$, where h_θ denotes another feed-forward network with parameters θ, $\mathcal{S}^T \in \mathbb{R}^{n \times n}$ refers to the weights in the residual layer, and Y are the responses. To enforce feature sparsity, L1 regularization is applied to the weights of the skip connection, defined as $||\mathcal{S}||_{1,1}$ (where $\|\cdot\|_{1,1}$ denotes the element-wise L1 norm). A constraint term with factor ρ is then added to the objective to budget the non-linearity involved for feature k relative to the importance of X_k

$$\min_{\theta,\mathcal{S}} \; \mathcal{L}_E(\theta, \mathcal{S}) + \lambda ||\mathcal{S}||_1 \qquad \text{subject to} \;\; \|[W^1]_{\cdot,k}\|_\infty \leq \rho ||[\mathcal{S}]_{\cdot,k}||_2 \text{ for } k \in \{1, \ldots, n\} \,, \tag{6}$$

where $[W^1]_{\cdot,k}$ denotes the kth column of the first layer weights of h_θ, and $[\mathcal{S}]_{\cdot,k}$ represents the kth column of $\mathcal{S}$. When $\rho = 0$, only the skip connection remains (standard linear Lasso), while $\rho \to \infty$ corresponds to unregularized network training.

LassoNet regularization can be extended to NODEs by adding the skip connection either before or after integration (the ODE solver). If added before integration, which we call Inner LassoNet, a linear function of X is added to its time derivative

$$\dot{X} = \mathcal{S}^T X + f_\theta(X, t) \,, \quad X(0) = x_0 \,. \tag{7}$$

Adding the skip connection after integration (and the predictor o), called Outer LassoNet, yields

$$X_{t_1} = \mathcal{S}^T X_t + o(\text{ODESolver}(f_\theta, x_0, t_0, t_1)) \,. \tag{8}$$

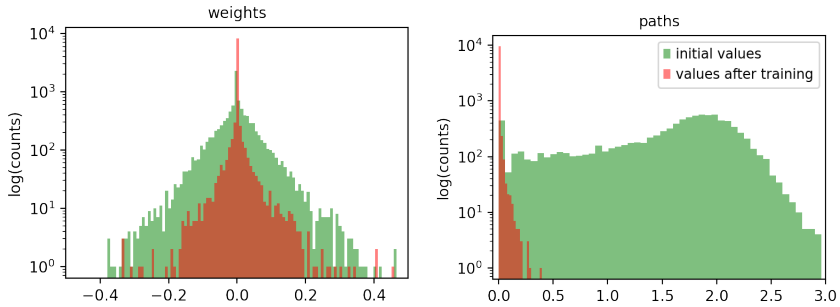


Figure 2: The distributions of network weights and paths weights in Eq. 13 using PathReg applied to single-cell data in Section 4.3. PathReg increases both model and feature sparsity.

PathReg (ours). While L0 regularization minimizes the number of non-zero weights in a network, it does not necessarily lead to feature sparsity meaning that the response variables can still depend on all features including spurious ones. To constrain the number of *input-output paths*, i.e., to enforce feature-sparsity, we regularize the probability of any path throughout the entire network contributing to an output. To this end, we use non-negative stochastic gates $z = g(s)$ similar to Eq. 4, where the probability of an input-output path $\mathcal{P}$ being non-zero is given by

$$q(\mathcal{P} \neq 0) = \prod_{z \in \mathcal{P}} q(z \neq 0 \mid \phi) \tag{9}$$

and we constrain

$$\sum_{i=1}^{\#paths} \prod_{z \in \mathcal{P}_i} q(z \neq 0 \mid \phi) \tag{10}$$

to minimize the number of paths that yield non-zero contributions from inputs to outputs. This is equivalent to regularizing the *gate adjacency matrix* $A_z = G^{L+1} \cdot \ldots \cdot G^1$. Where G^l is a probability matrix corresponding to the probability of the l-th layer gates being positive. Then, $A_{z_{ij}}$ represents the sum of the probabilities of all paths between the i-th input and the j-th output. Ultimately, we thus obtain our PathReg regularization term

$$\|A_z\|_{1,1} = \|G^{L+1} \ldots G^1\|_{1,1} \quad \text{with} \quad G^l_{ij} = q^l(z_{ij} \neq 0 \mid \phi_{i,j})\,. \tag{11}$$

Regularizing $\|A_z\|_{1,1}$, minimizes the number of paths between inputs and outputs and induces no shrinkage on the actual values of the weights. Therefore, we can utilize other regularizers on θ such as $\|A\|_{1,1}$ in conjunction with PathReg similar to Eq. 2. In this work, we consider the following overall loss function

$$\begin{aligned}\mathcal{R}(\theta, \phi) &= \mathbb{E}_{q(s|\phi)}\left[\frac{1}{N}\left(\sum_{i=1}^{N} \mathcal{L}\big(o(f(x_i; \theta \odot g(s)), y_i)\big)\right)\right] + \lambda_0 \|A_z\|_{1,1} + \lambda_1 \|A\|_{1,1} \\ &= \mathcal{L}_E(\theta, \phi) + \lambda_0 \|A_z\|_{1,1} + \lambda_1 \|A\|_{1,1}\end{aligned} \tag{12}$$

where $\mathcal{L}$ corresponds to the loss function of the original predictive task, $\mathcal{L}_E$ the overall predictive loss, o is the NODE-based model with f modeling time derivatives, $g(\cdot) := \min(1, \max(0, \cdot))$, $\lambda_0, \lambda_1 \geq 0$ are regularization parameters, and $\odot$ is entry-wise multiplication. $\mathcal{L}_E$ measures how well the model fits the current dataset. While $\|A\|_{1,1}$ shrinks the actual values of weights θ in an attempt to zero out entire paths, $\|A_z\|_{1,1}$ enforces exact zeros for entire paths at once. When $A_{z_{ij}} = 0$ the output i is constant in input j. In practice, during training the expectation in Eq. 12 is estimated as usual via Monte Carlo sampling.

Unlike GroupLasso (Bellot et al., 2021), PathReg does not require training of multiple networks in parallel. Moreover, unlike C-NODE (Aliee et al., 2021), PathReg leads to exact zeros in the weight matrices and requires no choice of threshold for deciding which paths are considered as zeros (see Figure 2).

3.2 Training sparse models

Sparsity can generally be enforced during or after model training. In this paper, we study two main approaches (Hoefler et al., 2021) to train sparse models: (i) *sparse training*, where we add regularizers

already during model training; (II) *training and iterative sparsification*, where we iteratively increase regularization parameters only after the model has been trained (close to) to convergence. A detailed description of our training procedures and architecture choices for each experiment is provided in Appendix A.

3.3 Assessing feature sparsity

For models regularized with L0 or PathReg the natural measure for feature sparsity is

$$(|W^{L+1}| \odot G^{L+1}) \cdot \ldots \cdot (|W^1| \odot G^1)\,, \tag{13}$$

where absolute values are taken entry-wise. LassoNet also allows for a natural feature sparsity metric by counting the non-zero input weights $\sum_{j=1}^{n} \|W_j^{(1)}\|_2 > \epsilon$ of the first hidden layer, where n denotes the number of input features and ϵ is a small tolerance parameter. To evaluate feature sparsity for other feed-forward continuous-depth models, we count the number of non-zero elements in $A = |W^{L+1}| \cdots |W^1|$ again with entry wise absolute values on all W and within some tolerance ϵ (set to $1e-5$ in our experiments). This amounts to an upper bound on the number of input-output dependencies: while we can have $A_{i,j} \neq 0$ even though $X_j \notin pa_{f_\theta}(X_i)$, $A_{i,j} = 0$ always implies $X_j \notin pa_{f_\theta}(X_i)$.

4 Results

In this section, we present empirical results on several datasets from different domains including human motion capture data and large-scale single-cell RNA-seq data. We demonstrate the effect of the different sparsity methods including L0 (Louizos et al., 2018), C-NODE (Aliee et al., 2021), LassoNet (Lemhadri et al., 2021), GroupLasso (Bellot et al., 2021), and our PathReg on the generalization capability of continuous-depth neural nets. We address different aspects of generalization by assessing the accuracy of these models for time-series forecasting and the identification of the governing dynamical laws. In all examples, we assume that the observed motion can be (approximately) described by a system of ODEs where variables correspond directly to the observables of the trajectories.

4.1 Sparsity for system identification

We motivate sparsity for NODEs starting with a simple second-order NODE as used by Aliee et al. (2021). The differential equations explaining this system are presented in Appendix B.1. The questions that we try to answer here are whether different sparsity methods can (i) learn the true underlying system, and (ii) forecast into the future (extrapolate). For this linear system, the adjacency matrix represents the exact coefficients in Eq. 14. The results summarized in Tables 1 and 2 show that PathReg, C-NODE, LassoNet, and GroupLasso perform well with respect to reconstruction error. Moreover, PathReg, C-NODE, and GroupLasso lead to higher sparsity and outperform other methods for extrapolation. Among these methods, only PathReg and C-NODE are able to learn the true system with the lowest error[2].

Table 1: Results for the synthetic second-order ODE.

	MSE		SPARSITY (%)	
MODEL	TRAIN	EXTRAPOLATION	FEATURES	WEIGHTS
BASE	1.3×10^{-4}	3.2×10^{-3}	47.93	11.11
L0	1.9×10^{-2}	3.3×10^{-2}	0.0	**49.63**
C-NODE	6.60×10^{-5}	4.13×10^{-4}	**75.52**	16.67
PATHREG	$\mathbf{2.30 \times 10^{-5}}$	$\mathbf{3.5 \times 10^{-4}}$	**75.52**	35.17
LASSONET	8.4×10^{-3}	4.0×10^{-2}	0.0	0.0
GROUPLASSO	8.82×10^{-5}	1.50×10^{-3}	61.1	6.0

[2]GroupLasso is originally designed for first-order ODEs. Here, we use an augmented version of that feeding also the initial velocities to the network. Therefore, a direct comparison with its adjacency matrix is not entirely adequate.

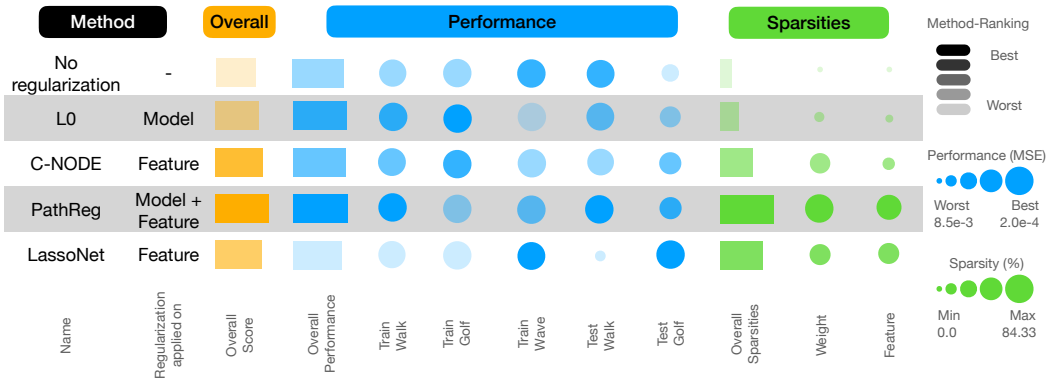


Figure 3: Qualitative result comparison for the human motion capture data. Circle sizes indicate the scaled (continuous) metrics: MSE (blue) and sparsity (in %) (green). The lengths of bar show the mean metric over all sparsities / performances for a method. Different shades of the same color indicate the ranking from best to worst method within a given column. The overall score (orange) is the mean of all circle sizes of the row. PathReg yields the best overall results in this comparison.

4.2 Sparsity improves time-series forecasting

We next demonstrate the robustness of sparse models for both reconstructing and extrapolating human movements using real motion capture data from `mocap.cs.cmu.edu`. Each datapoint is 93-dimensional (31 joint locations with three dimensions each) captured over time. We select three different movements including walking, waving, and golfing, and use 100 frames each for training. After training, we query all models to extrapolate the next 100 frames. For walking and golfing, where multiple trials are available, we also test the model on unseen data in the sense that it has not seen any subset of those specific sequences (despite having trained on other sequences depicting the same type of movement). Generally, for time-series forecasting, it is not straight forward to give precise definitions of in-, and out-of-distribution. In Table 7 in Appendix C, we show a grid of plots illustrating loss and sparsity as a function of the regularization parameter λ of each method on this dataset. We show detailed results in Table 3 in Appendix C and summarize them concisely in Figure 3. We did not manage to scale GroupLasso to this 93-dimensional dataset. While all other models show comparative performance (in terms of MSE error), only PathReg achieves strong levels of both weight and feature sparsity (quantitative comparison is in Table 3). The results show that PathReg outperforms other methods with respect to MSE for both train and test datasets while resulting in the highest model and feature sparsity.

We further examine the extrapolation (in time) capability of these models. We observe that all sparsity regularization techniques improve extrapolation over unregularized NODEs. Figure 4 qualitatively illustrates the reconstruction and extrapolation performance of both the baseline and PathReg for the walking dataset (visually, the results are similar to PathReg for other regularizers). While these models perform well for extrapolation and generalization to unseen data, we show that only PathReg and C-NODE are able to avoid using spurious features for forecasting. Figures 7 and 8 in Appendix C show that PathReg and C-NODE learn sensible and arguably correct adjacency matrices (derived from joint connectedness), whereas L0 and LassoNet fail to sparsify the input-output interactions appropriately. Since we use second-order NODEs for this dataset, the adjacency matrix described in Section 3.3 represents how the acceleration predicted by f_θ of each joint point (e.g., X_i) depends on the positions as well as the velocities of other joint points (e.g., X_j). In case the dependency is non-constant, we infer that the joint point X_i interacts with X_j.

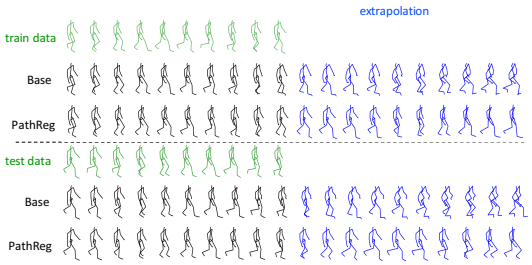


Figure 4: Human-movement reconstructions and extrapolation.

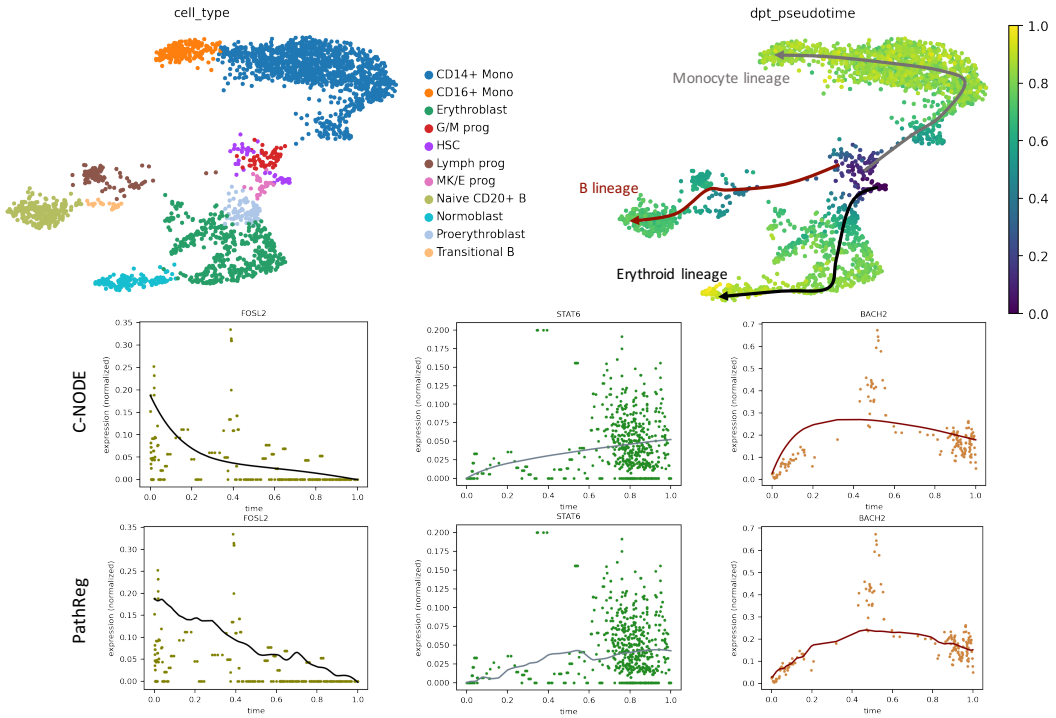


Figure 5: Single-cell RNA-seq Data. Shown are 2D UMAP visualisation of training data colored by cell types and pseudotime (top) where each point is a cell, and the expressions of three genes and their predictions over pseudo time using C-NODE (top) and PathReg (bottom). The predictions across all genes are presented in Figure 9.

4.3 Sparsity improves dynamic identification for single-cell data

Finally, we explore how pruning (i) improves real-world generalization and (ii) helps to learn gene-gene interactions in human hematopoiesis single-cell multiomics data (Luecken et al., 2021) (GEO accession code = GSE194122). The regulatory (or causal) dependencies between genes can explain cellular functions and help understanding biological mechanisms (Aalto et al., 2020). Gene interactions are often represented as a gene regulatory network (GRN) where nodes correspond to genes and directed edges indicate regulatory interactions between genes. Reliable GRN inference from observational data is exceptionally challenging and still an open problem in genomics (Qiu et al., 2020). As human regulatory genes known as transcription factors ($\sim$ 1,600 (Lambert et al., 2018)), do not individually target all genes (potentially up to $\sim$ 20,000), the inferred GRN is expected to be *sparse*. Thus, predicting the expression of a gene should depend on few other genes that can be restricted using sparsity-enforcing regularizers.

For assessing real-world generalization of various sparsity-enforcing methods, we use human bone marrow data of four diverse donors, generated at three different sites using five sequencing batches. In this experiment, we train NODE models with one batch (S1D2), and later test those using other four batches (S1D1, S2D1, S2D4, and S3D6). Notably, time-resolved ground truth dynamics of cells throughout their lifetime are not available, because all existing genome-wide technologies are destructive to the cells (Luecken et al., 2021). Therefore, scRNA-seq only offers static snapshots of data and the temporal order of cell states is commonly estimated based on certain distances between their measured transcriptomes. Following that, we first apply the diffusion pseudo-time (DPT) ordering approach (Haghverdi et al., 2016), a diffusion map based manifold learning technique, to rank the cells. This allows to generate a cell fate mapping from hematopoietic stem cells (HSC) into three biological lineages that are used for training: Monocyte, Erythroid, and B cells (Figure 5). NODEs are then trained using highly-variable genes (529 genes) with time taken to be the inferred pseudo-time (Figure 9).

The train and test MSE across 5 batches and three pruning methods are presented in Table 5 and Table 7 in Appendix C and summarized in Figure 6. While all regularized models achieve decent

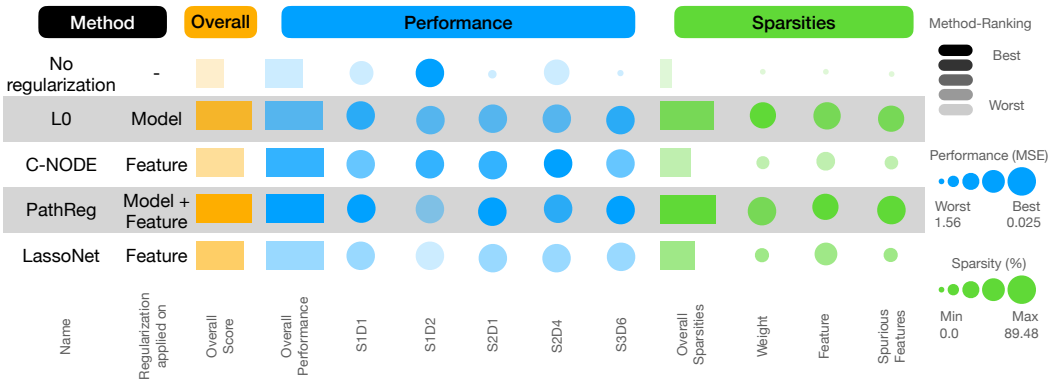


Figure 6: Comparison results using the single-cell RNA-seq Data. PathReg shows best overall results, as well as with regard to performance, and to sparsity.

predictive performance, PathReg outperforms the other methods in most scenarios. We further study the transcription-factor-to-target-gene interactions via the adjacency matrices filtered for transcription factors (columns) and target genes (rows) (see Appendix A for more details). The adjacency matrix A demonstrates what transcription factors (on columns) are used for predicting the expression value of a target gene (on rows). $|A_{ij}| > \epsilon$ for a small threshold $\epsilon = 1e-5$ can then be interpreted as an interaction between gene i and gene j being present in the data (see Figure 11 for an illustrating example). For visualisation purposes, Figure 10 only shows the adjacency matrices across the Erythroid lineage. However, the results are comparable for other lineages. The used list of transcription factors and target genes are extracted using chromatin and sequence-specific motif information (we again refer to Appendix A for more details). Since ground truth dynamics are not known for GRNs, validating the inferred interactions for multiple cell types is a challenging task. However, we observe that PathReg and L0 are able to remove the interactions between pairs known to not directly causally influence each other, including target genes (called spurious features in Table 5), while the other models fail in that regard. Assessed by the literature, we also observe that known erythroid transcription factors are inferred as important by both PathReg and L0 (RUNX1 (Kuvardina et al., 2015), ETV6 (Wang et al., 1998), NFIA (Micci et al., 2013), MAZ (Deen et al., 2021), GATA1 (Zheng et al., 2006).

5 Conclusion and discussion

We have shown the efficacy of various sparsity enforcing techniques on generalization of continuous-depth NODEs and proposed PathReg, a regularizer acting directly on entire paths throughout a neural network and achieving exact zeros. We curate real-world datasets from different domains consisting of human motion capture and single-cell RNA-seq data. Our findings demonstrate that sparsity improves out-of-distribution generalization (for the types of OOD considered) of NODEs when our objective is system identification from observational data, extrapolation of a trajectory, or real-world generalization to unseen biological datasets collected from diverse donors. Finally, we show that unlike weight sparsity, feature sparsity as enforced by PathReg can indeed help in identifying the underlying dynamical laws instead of merely achieving strong in-distribution predictive performance. This is particularly relevant for applications like gene-regulatory network inference, where the ultimate goal is not prediction and forecasting, but revealing the true underlying regulatory interactions between genes of interest. We hope that our empirical findings as well as curated datasets can serve as useful benchmarks to a broader community and expect that extensions of our framework to incorporate both observational and experimental data will further improve practical system identification for ODEs. Finally, our path-based regularization technique may be of interest to other communities that aim at enforcing various types of shape constraints or allowed dependencies into deep learning based models[3].

[3]We highlight that caution must be taken when informing consequential decisions, for example in healthcare, based on causal structures learned from observational data.

References

Aalto, A., Viitasaari, L., Ilmonen, P., Mombaerts, L., and Gonçalves, J. Gene regulatory network inference from sparsely sampled noisy data. *Nature Communications*, 11(1):3493, 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17217-1. URL https://doi.org/10.1038/s41467-020-17217-1.

Ahmad, S. and Hawkins, J. How do neurons operate on sparse distributed representations? a mathematical theory of sparsity, neurons and active dendrites, 2016. URL https://arxiv.org/abs/1601.00720.

Aliee, H., Theis, F. J., and Kilbertus, N. Beyond predictions in neural odes: Identification and interventions, 2021.

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers, 2018.

Bartoldson, B. R., Morcos, A. S., Barbu, A., and Erlebacher, G. The generalization-stability tradeoff in neural network pruning, 2020.

Bellot, A., Branson, K., and van der Schaar, M. Graphical modelling in continuous-time: consistency guarantees and algorithms using neural odes, 2021.

Bolstad, A., Veen, B. D. V., and Nowak, R. Causal network inference via group sparse regularization. *IEEE Transactions on Signal Processing*, 59(6):2628–2641, jun 2011.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

Brutzkus, A., Globerson, A., Malach, E., and Shalev-Shwartz, S. Sgd learns over-parameterized networks that provably generalize on linearly separable data, 2017.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Deen, D., Butter, F., Daniels, D. E., Ferrer-Vicens, I., Ferguson, D. C., Holland, M. L., Samara, V., Sloane-Stanley, J. A., Ayyub, H., Mann, M., Frayne, J., Garrick, D., and Vernimmen, D. Identification of the transcription factor maz as a regulator of erythropoiesis. *Blood Advances*, 5, 2021. ISSN 24739537. doi: 10.1182/bloodadvances.2021004609.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and de Freitas, N. Predicting parameters in deep learning, 2013.

Du, S. S., Zhai, X., Póczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. *CoRR*, abs/1810.02054, 2018.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural odes. *arXiv preprint arXiv:1904.01681*, 2019.

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., and Oberman, A. M. How to train your neural ode: the world of jacobian and kinetic regularization, 2020.

Ghosh, A., Behl, H. S., Dupont, E., Torr, P. H. S., and Namboodiri, V. Steer: Simple temporal regularization for neural odes, 2020.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2019.

Haghverdi, L., Büttner, M., Wolf, F. A., Buettner, F., and Theis, F. J. Diffusion pseudotime robustly reconstructs lineage branching. *Nature Methods*, 13(10), 2016. ISSN 15487105. doi: 10.1038/nmeth.3971.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, 2021.

Kelly, J., Bettencourt, J., Johnson, M. J., and Duvenaud, D. Learning differential equations that are easy to solve. In *Neural Information Processing Systems*, 2020. URL `https://arxiv.org/abs/2007.04504`.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Kuvardina, O. N., Herglotz, J., Kolodziej, S., Kohrs, N., Herkt, S., Wojcik, B., Oellerich, T., Corso, J., Behrens, K., Kumar, A., Hussong, H., Urlaub, H., Koch, J., Serve, H., Bonig, H., Stocking, C., Rieger, M. A., and Lausen, J. Runx1 represses the erythroid gene expression program during megakaryocytic differentiation. *Blood*, 125, 2015. ISSN 15280020.

Lambert, S. A., Jolma, A., Campitelli, L. F., Das, P. K., Yin, Y., Albu, M., Chen, X., Taipale, J., Hughes, T. R., and Weirauch, M. T. The human transcription factors. *Cell*, 172(4):650–665, February 2018.

Lemhadri, I., Ruan, F., Abraham, L., and Tibshirani, R. Lassonet: A neural network with feature sparsity. *Journal of Machine Learning Research*, 22(127):1–29, 2021. URL `http://jmlr.org/papers/v22/20-848.html`.

Li, X., Wong, T.-K. L., Chen, R. T. Q., and Duvenaud, D. Scalable Gradients for Stochastic Differential Equations. jan 2020. URL `http://arxiv.org/abs/2001.01328`.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data, 2018.

Liebenwein, L., Hasani, R., Amini, A., and Rus, D. Sparse flows: Pruning continuous-depth models, 2021.

Lim, B. and Zohren, S. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194): 20200209, feb 2021.

Liu, Y., Li, L., and Wang, X. A nonlinear sparse neural ordinary differential equation model for multiple functional processes. *Canadian Journal of Statistics*, 50, 2022. ISSN 1708945X. doi: 10.1002/cjs.11666.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=H1Y8hhg0b`.

Luecken, M. D., Burkhardt, D. B., Cannoodt, R., Lance, C., Agrawal, A., Aliee, H., Chen, A. T., Deconinck, L., Detweiler, A. M., Granados, A. A., Huynh, S., Isacco, L., Kim, Y. J., Klein, D., KUMAR, B. D., Kuppasani, S., Lickert, H., McGeever, A., Mekonen, H., Melgarejo, J. C., Morri, M., Müller, M., Neff, N., Paul, S., Rieck, B., Schneider, K., Steelman, S., Sterr, M., Treacy, D. J., Tong, A., Villani, A.-C., Wang, G., Yan, J., Zhang, C., Pisco, A. O., Krishnaswamy, S., Theis, F. J., and Bloom, J. M. A sandbox for prediction and integration of DNA, RNA, and proteins in single cells. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL `https://openreview.net/forum?id=gN35BGa1Rt`.

Micci, F., Thorsen, J., Panagopoulos, I., Nyquist, K. B., Zeller, B., Tierens, A., and Heim, S. High-throughput sequencing identifies an nfia/cbfa2t3 fusion gene in acute erythroid leukemia with t(1;16)(p31;q24), 2013. ISSN 08876924.

Mooij, J. M., Janzing, D., and Schölkopf, B. From ordinary differential equations to structural causal models: the deterministic case. *arXiv preprint arXiv:1304.7920*, 2013.

Ng, I., Ghassami, A., and Zhang, K. On the role of sparsity and dag constraints for learning linear dags, 2020.

Norcliffe, A., Bodnar, C., Day, B., Simidjievski, N., and Liò, P. On Second Order Behaviour in Augmented Neural ODEs. jun 2020. URL `http://arxiv.org/abs/2006.07220`.

Oganesyan, V., Volokhova, A., and Vetrov, D. Stochasticity in Neural ODEs: An Empirical Study, 2020. ISSN 23318422.

Pal, A., Ma, Y., Shah, V., and Rackauckas, C. Opening the blackbox: Accelerating neural differential equations by regularizing internal solver heuristics, 2021.

Qiu, X., Rahimzamani, A., Wang, L., Ren, B., Mao, Q., Durham, T., McFaline-Figueroa, J. L., Saunders, L., Trapnell, C., and Kannan, S. Inferring causal gene regulatory networks from coupled single-cell expression dynamics using scribe. *Cell Systems*, 10(3):265–274.e11, 2020. ISSN 2405-4712. doi: https://doi.org/10.1016/j.cels.2020.02.003. URL `https://www.sciencedirect.com/science/article/pii/S2405471220300363`.

Ross, A. S. and Doshi-Velez, F. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients, 2017.

Ross, A. S., Hughes, M. C., and Doshi-Velez, F. Right for the right reasons: Training differentiable models by constraining their explanations, 2017.

Rubanova, Y., Chen, R. T., and Duvenaud, D. Latent ODEs for irregularly-sampled time series, 2019. ISSN 23318422.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. Towards Causal Representation Learning. feb 2021. URL `http://arxiv.org/abs/2102.11107`.

Shell, M. Carnegie mellon university motion capture database, 2012.

Slavin, A. B. The neural lasso : Local linear sparsity for interpretable explanations. 2018.

Wang, L. C., Swat, W., Fujiwara, Y., Davidson, L., Visvader, J., Kuo, F., Alt, F. W., Gilliland, D. G., Golub, T. R., and Orkin, S. H. The tel/etv6 gene is required specifically for hematopoiesis in the bone marrow. *Genes and Development*, 12, 1998. ISSN 08909369. doi: 10.1101/gad.12.15.2392.

Wolf, F. A., Angerer, P., and Theis, F. J. SCANPY: Large-scale single-cell gene expression data analysis. *Genome Biology*, 2018. ISSN 1474760X. doi: 10.1186/s13059-017-1382-0.

Xu, S., Bu, Z., Chaudhari, P., and Barnett, I. J. Sparse neural additive model: Interpretable deep learning with feature selection via group sparsity. *arXiv preprint arXiv:2202.12482*, 2022.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization, 2016.

Zheng, J., Kitajima, K., Sakai, E., Kimura, T., Minegishi, N., Yamamoto, M., and Nakano, T. Differential effects of gata-1 on proliferation and differentiation of erythroid lineage cells. *Blood*, 107, 2006. ISSN 00064971. doi: 10.1182/blood-2005-04-1385.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P., and Xing, E. P. Learning sparse nonparametric dags, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The codes are included in the supplementary files.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Explained in Section A and included in the codes.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Included in Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] (Chen et al., 2018; Norcliffe et al., 2020)
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] All the assets used in this paper are publicly accessible.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Models

All neural networks used in this work are fully connected, feed-forward neural networks. First-order NODEs are used for single-cell data, while second NODEs are used for the synthetic example as well as the motion capture data. In the second-order NODEs, the initial velocities are predicted using a neural network with two hidden layers with 20 or 100 neurons depending on the dataset with ELU activation function. The main architecture to infer velocities (or accelerations) also contains two hidden layers of sizes 20 or 100 depending on the size of the input and ELU (or for the synthetic experiments linear) activation function. As an ODE solver, we use an explicit 5-th order Dormand-Prince solver commonly denoted by `dopri5`.

All models are optimized using Adam (Kingma & Ba, 2017) with an initial learning rate of 0.01. We use PyTorch's default weight initialization scheme for the weights. All models can be trained entirely on CPUs on consumer grade Laptop machines within minutes or hours.

B Datasets

B.1 Synthetic second-order ODE

We use a second-order, homogeneous, linear system of ODEs with constant coefficients to assess how pruning affects system identification using NODEs. Our system is denoted as:

$$X_0 = -1.5 \cdot X_0 - 0.5 \cdot \dot{X}_0 \tag{14}$$

$$X_1 = -2.7 \cdot X_1 - 1.0 \cdot \dot{X}_1 \tag{15}$$

$$X_2 = -2.0 \cdot X_2 - 1.0 \cdot \dot{X}_0 - 1.0 \cdot \dot{X}_1 - 0.1 \cdot \dot{X}_2 \tag{16}$$

We also set $X(0) = (0.4, 0.7, 1.8)$ and $\dot{X}(0) = (0.3, 0.5, 1.0)$ as initial positions and velocities, respectively.

B.2 Motion capture data

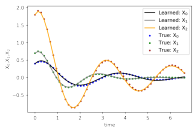
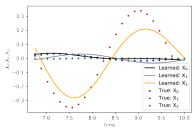
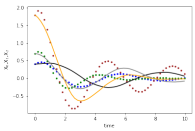
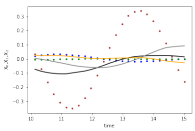
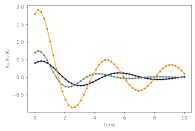
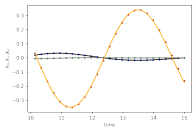
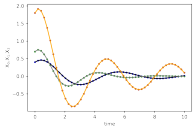
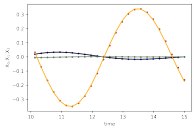
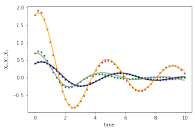
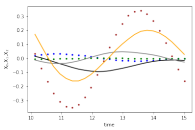
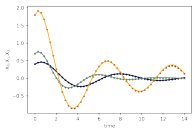
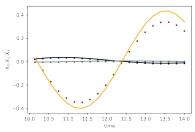
The Carnegie Mellon University Motion Capture Database (Shell, 2012) is a large collection of human motion data. We are only interested in those records where activities are performed by a single person who does not interact with other people and has minimal contact with objects within the environment.The data is presented as 31 joint points in 3D space (in total of 93 variables) and a graph links them. The data is preprocessed as the following:

- **Centering the frames:** The root point is fixed at the origin $(0, 0, 0)$ for the whole duration of the movement. We observe that this method is effective particularly for walking data.
- **Centering the first frame:** For golfing and waving, the root point of the first frame is moved to the origin $(0, 0, 0)$.
- **Setting the floor at $z = 0$:** We find the minimum value of the Z-coordinate throughout the trial and among all joint points. We consider this level to be the "floor" and move all points so that the minimum value of z during the whole trial is at 0 on the Z-axis.

B.3 Single-Cell RNA-seq

To map putative transcription factor (TF) and target gene relationships, we use as a reference a regulatory network generated using the gene expression and chromatin accessibility features available in the human immune cells dataset. Briefly, using the function *rank_genes_groups* from SCANPY (Wolf et al., 2018), we select the top 1,000 genes and 25,000 chromatin peaks, using cell type label as a covariate. Then, for each chromatin peak we map sequence-specific TF motifs using as reference the TF archetypes from ENCODE (n=236), Peaks are mapped to the closest gene using the R package ChIPseeker, with default parameters for the genome build hg38. All chromatin peaks linked to a gene as an enhancer annotation are considered, as long as the mapped distance between peak and promoter is below 200 Kbp. Promoter/intron/exon peaks are always mapped to the harboring gene. Our rule for successfully mapping a TF to a target gene through a chromatin peak is that all TF, chromatin peak, and target gene, have to be simultaneously in the list of features selected

Table 2: Results of the synthetic second-order ODE.

MODEL	ADJACENCY MATRIX	RECONSTRUCTION	EXTRAPOLATION
BASE			
L0			
C-NODE			
PATHREG			
LASSONET			
GROUPLASSO			

in the *rank_genes_groups* function for cell type of interest, and there have to be TF motifs linked to that transcription factor in the chromatin peak. This gives us a total of 113 TFs and 365 potential targets genes across all cell types, that we use for evaluating the models.

C Extended results

C.1 Synthetic second-order ODE

Table 2 shows the inferred adjacency matrices, reconstructed and extrapolated samples. The values in the adjacency matrices show the coefficients of the learned differential equations with the true coefficients presented in Eq. 14, 15,and 16. Only C-NODE and PathReg are able to learn the true dynamics.

C.2 Motion capture data

Tables 3 and 4 show train and test MSE as well as model and feature sparsity for motion capture data. The inferred adjacency matrix for walking motion using PathReg is illustrated in Figure 7. The joint points corresponding to each leg or hand are highlighted with dashed lines. The extended results across different methods and motions are also presented in Figure 8.

Table 3: Train and test MSE for human motion capture data. Shown are average feature and parameter sparsity for three models trained with different motions (walk, wave, and golf). Outer LassoNet is presented in this benchmark, see Table 4 for comparison to Inner LassoNet.

		BASE	L0	C-NODE	PATHREG	LASSONET
TRAIN MSE	WALK	$7.0 \times 10^{-4} \pm 9.4 \times 10^{-5}$	$3.1 \times 10^{-4} \pm 2.0 \times 10^{-5}$	$6.0 \times 10^{-4} \pm 2.0 \times 10^{-4}$	$\mathbf{2.9 \times 10^{-4} \pm 1.8 \times 10^{-4}}$	$7.2 \times 10^{-4} \pm 2.9 \times 10^{-6}$
	WAVE	$2.4 \times 10^{-4} \pm 2.3 \times 10^{-6}$	$\mathbf{2.0 \times 10^{-4} \pm 1.5 \times 10^{-7}}$	$\mathbf{2.0 \times 10^{-4} \pm 2.1 \times 10^{-5}}$	$\mathbf{2.0 \times 10^{-4} \pm 4.8 \times 10^{-4}}$	$5.1 \times 10^{-4} \pm 1.6 \times 10^{-5}$
	GOLF	$6.6 \times 10^{-4} \pm 1.5 \times 10^{-2}$	$7.0 \times 10^{-4} \pm 4.2 \times 10^{-4}$	$6.9 \times 10^{-4} \pm 4.3 \times 10^{-4}$	$6.8 \times 10^{-4} \pm 4.0 \times 10^{-5}$	$\mathbf{3.3 \times 10^{-4} \pm 3.3 \times 10^{-5}}$
TEST MSE	WALK	$2.4 \times 10^{-3} \pm 2.1 \times 10^{-4}$	$2.3 \times 10^{-3} \pm 3.1 \times 10^{-2}$	$2.7 \times 10^{-3} \pm 1.2 \times 10^{-4}$	$\mathbf{2.0 \times 10^{-3} \pm 2.0 \times 10^{-4}}$	$8.5 \times 10^{-3} \pm 1.3 \times 10^{-3}$
	GOLF	$4.5 \times 10^{-3} \pm 2.0 \times 10^{-2}$	$3.1 \times 10^{-3} \pm 1.1 \times 10^{-5}$	$2.8 \times 10^{-3} \pm 4.3 \times 10^{-4}$	$2.7 \times 10^{-3} \pm 1.7 \times 10^{-4}$	$\mathbf{4.8 \times 10^{-4} \pm 3.1 \times 10^{-4}}$
FEAT. SPAR. (%)		0.0 ± 0.0	8.06 ± 2.04	25.33 ± 2.41	$\mathbf{71.33 \pm 2.19}$	57.48 ± 47.21
MODEL SPAR. (%)		0.0 ± 0.0	17.11 ± 1.10	55.66 ± 1.93	$\mathbf{84.33 \pm 2.51}$	57.42 ± 47.20

Table 4: Comparison of Inner- and Outer LassoNet. Train and test MSE for human motion capture data. Shown are average feature and parameter sparsity for three models trained with different motions (walk, wave, and golf). Outer LassoNet shows higher sparsity and better performance except for the walking test data and is therefore included in Table 3.

		INNER LASSONET	OUTER LASSONET
TRAIN MSE	WALK	$7.5 \times 10^{-4} \pm 4.3 \times 10^{-6}$	$7.2 \times 10^{-4} \pm 2.9 \times 10^{-6}$
	WAVE	$6.3 \times 10^{-4} \pm 1.6 \times 10^{-5}$	$5.1 \times 10^{-4} \pm 1.6 \times 10^{-5}$
	GOLF	$7.3 \times 10^{-4} \pm 2.0 \times 10^{-5}$	$3.3 \times 10^{-4} \pm 3.3 \times 10^{-5}$
TEST MSE	WALK	$1.2 \times 10^{-3} \pm 4.6 \times 10^{-5}$	$8.5 \times 10^{-3} \pm 1.3 \times 10^{-3}$
	GOLF	$4.0 \times 10^{-3} \pm 4.6 \times 10^{-4}$	$4.8 \times 10^{-4} \pm 3.1 \times 10^{-4}$
FEAT. SPAR. (%)		7.01 ± 15.66	57.48 ± 47.21
MODEL SPAR. (%)		6.31 ± 15.20	57.42 ± 47.20

C.3 Single-cell data

Tables 5 and 6 compare MSE and sparsity of various models. Figure 9 presents the real and predicted expression of genes over time for three lineages including Monocyte, Erythroid, and B using PathReg. The inferred adjacency matrices are also illustrated in Figure 10.

C.4 Analysis of the effect of regularization on sparsity and performance

The plots in Table 7 show the effects that different regularization weights have on performance and model as well as feature sparsity.

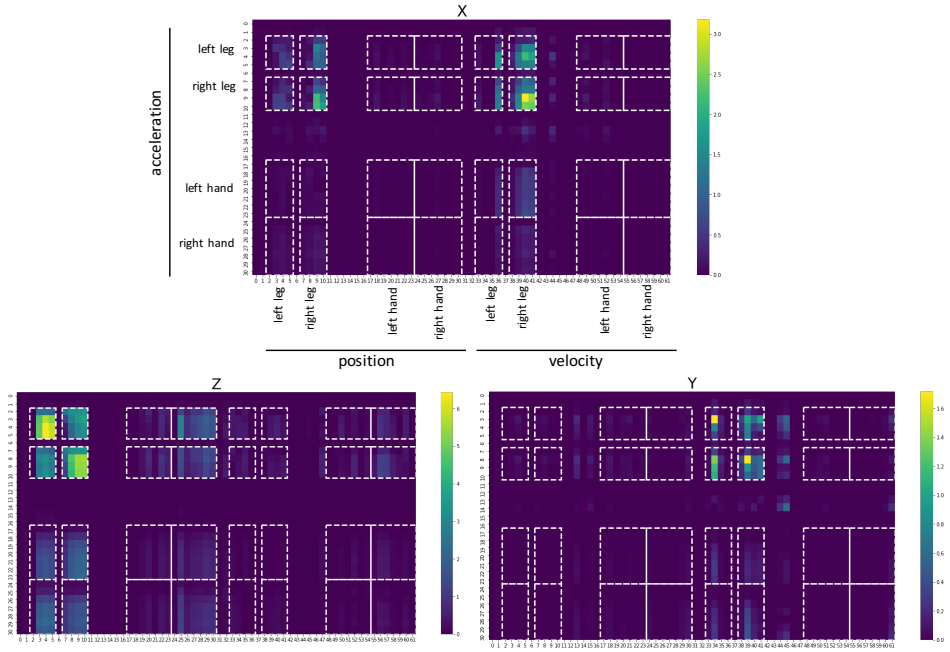


Figure 7: The adjacency matrix for MCD experiment trained with PathReg-regularized NODE. Shown are the interactions among joint points across three dimensions (X, Y, and Z). Here, a second-order NODE is trained that infers acceleration as a function of position and velocity. Then, X_j interacts with X_i if $X_j \in pa(X_i)$ or $\dot{X}_j \in pa(X_i)$, where $\ddot{X}_i = f(pa(X_i), t)$.

Table 5: Generalization for single-cell data. NODEs are trained with batch S1D2 and tested for four other batches. Shown are MSE as well as model and feature sparsity. Inner LassoNet is presented in this benchmark, see Table 6 for comparison to Outer LassoNet.

		BASE	L0	C-NODE	PATHREG	LASSONET
MSE	S1D1	0.34 ± 0.40	$0.038 \pm 5.5 \times 10^{-4}$	0.038 ± 0.097	$\mathbf{0.036 \pm 2.6 \times 10^{-4}}$	$0.044 \pm 6.4 \times 10^{-4}$
	S1D2	$\mathbf{0.033 \pm 1.7 \times 10^{-4}}$	$0.035 \pm 1.2 \times 10^{-4}$	$0.033 \pm 2.4 \times 10^{-4}$	$0.035 \pm 3.6 \times 10^{-4}$	$0.051 \pm 3.8 \times 10^{-4}$
	S2D1	1.39 ± 1.90	$0.042 \pm 1.0 \times 10^{-3}$	0.038 ± 0.021	$\mathbf{0.037 \pm 4.0 \times 10^{-4}}$	$0.046 \pm 4.6 \times 10^{-4}$
	S2D4	0.22 ± 0.27	$0.026 \pm 4.2 \times 10^{-4}$	0.025 ± 0.027	$\mathbf{0.025 \pm 2.9 \times 10^{-4}}$	$0.030 \pm 2.2 \times 10^{-4}$
	S3D6	1.56 ± 1.36	$0.075 \pm 4.2 \times 10^{-4}$	0.079 ± 0.025	$\mathbf{0.072 \pm 3.0 \times 10^{-4}}$	$0.081 \pm 3.4 \times 10^{-4}$
FEAT. SPAR. (%)		0.0 ± 0.0	79.58 ± 0.80	28.83 ± 11.9	$\mathbf{88.36 \pm 0.33}$	32.67 ± 41.56
MODEL SPAR. (%)		0.0 ± 0.0	$\mathbf{84.29 \pm 0.69}$	50.65 ± 11.04	81.06 ± 0.45	67.05 ± 21.31
SPUR. FEAT. SPAR. (%)		0.0 ± 0.0	81.49 ± 0.95	31.81 ± 11.30	$\mathbf{89.48 \pm 0.54}$	33.57 ± 42.06

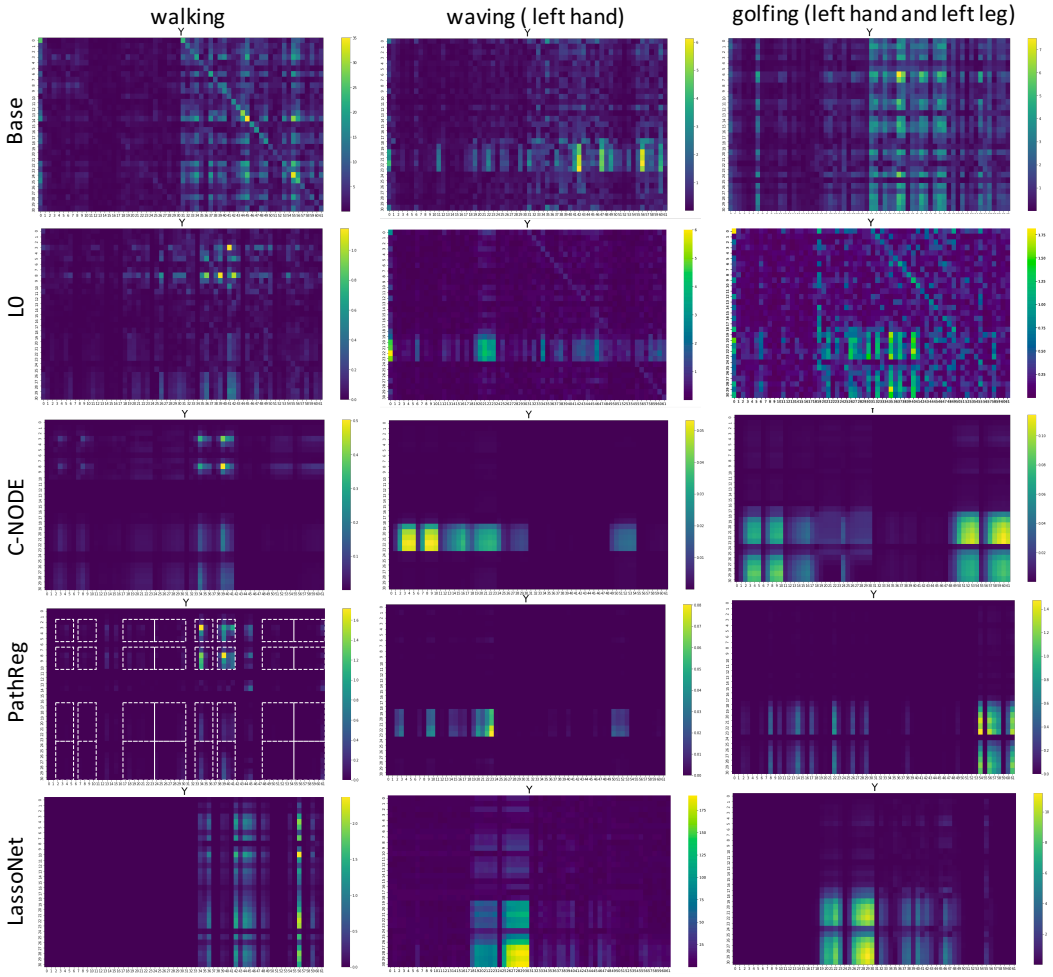


Figure 8: The adjacency matrices for MCD data trained with different pruning methods. PathReg and C-NODE find the most relevant features. For example for waving with the left hand, there are only interactions with the joint points corresponding to the left hand. For the sake of simplicity, we only show the interactions across dimension Y.

Table 6: Comparison Inner LassoNet and Outer LassoNet for single-cell RNA-seq data. While Outer LassoNet produces a higher performance for the in-distribution set S1D2 and higher sparsities, the generalization abilities are worse than the Inner LassoNet. Therefore, Inner LassoNet is used in the benchmark for single-cell data.

		INNER LASSONET	OUTER LASSONET
MSE	S1D1	$0.044 \pm 6.4 \times 10^{-4}$	$0.16 \pm 3.9 \times 10^{-2}$
	S1D2	$0.051 \pm 3.8 \times 10^{-4}$	$0.034 \pm 8.5 \times 10^{-5}$
	S2D1	$0.046 \pm 4.6 \times 10^{-4}$	$0.2 \pm 3.1 \times 10^{-2}$
	S2D4	$0.030 \pm 2.2 \times 10^{-4}$	$0.05 \pm 6.8 \times 10^{-3}$
	S3D6	$0.081 \pm 3.4 \times 10^{-4}$	$0.27 \pm 8.4 \times 10^{-2}$
FEAT. SPAR. (%)		32.67 ± 41.56	52.21 ± 6.37
MODEL SPAR. (%)		67.05 ± 21.31	73.20 ± 3.26
SPUR. FEAT. SPAR. (%)		33.57 ± 42.06	52.39 ± 6.36

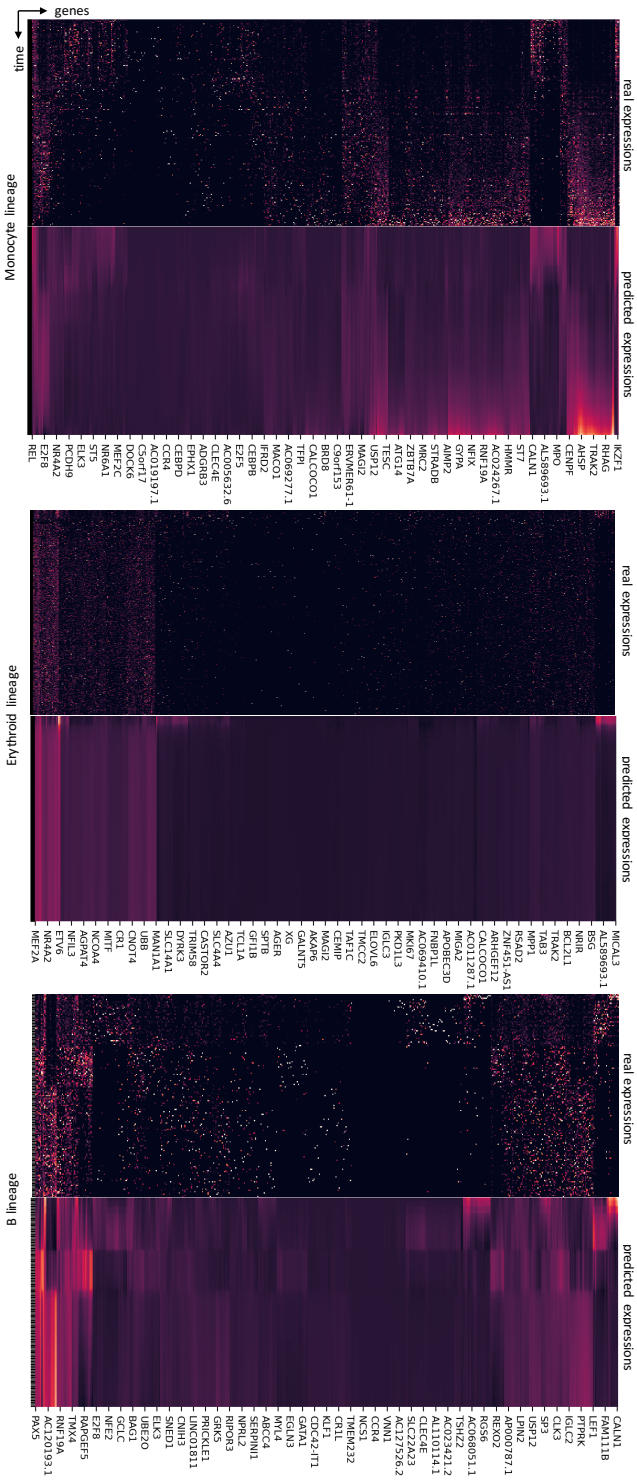


Figure 9: Normalized gene expressions over pseudotime. Shown are real (top) and predicted (bottom) expressions using PathReg.

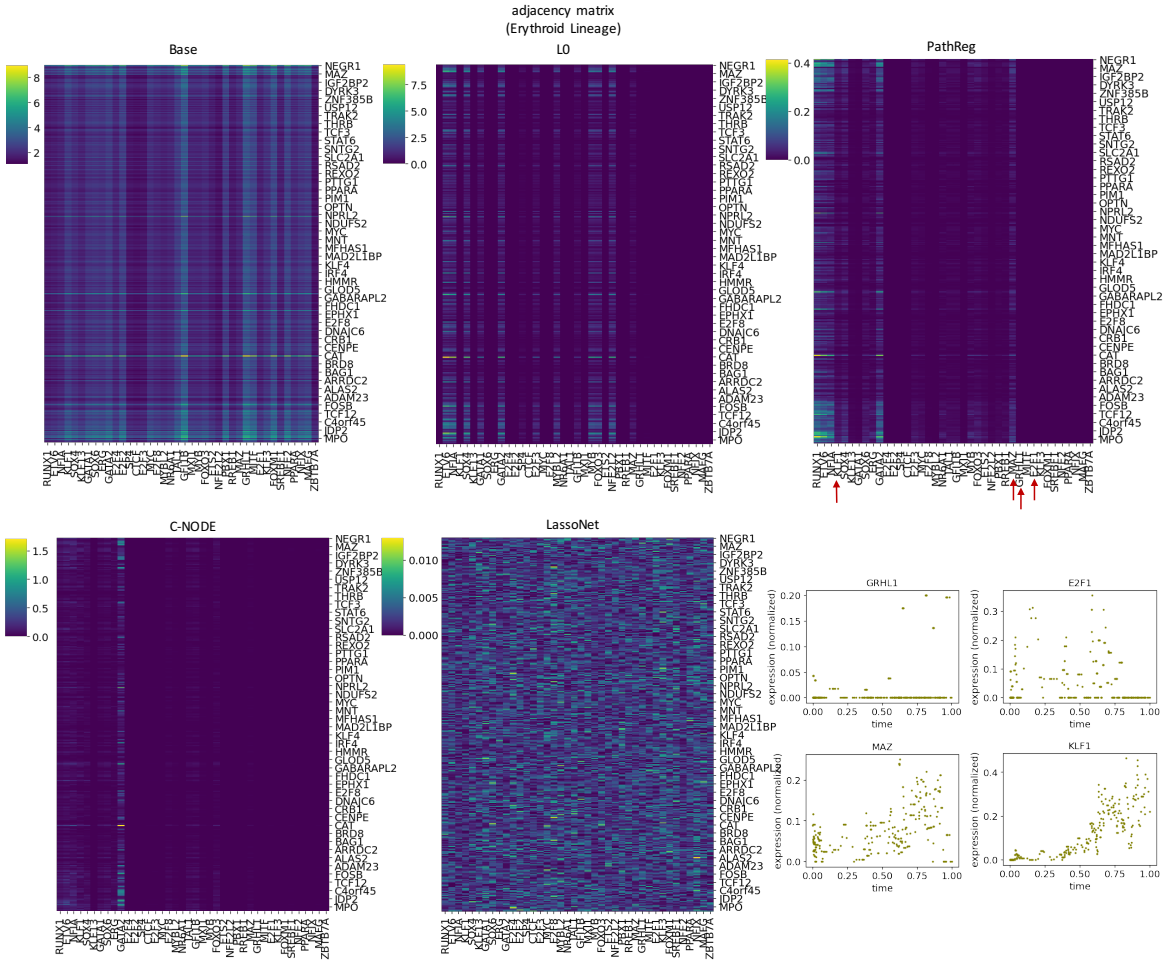


Figure 10: Adjacency matrix comparison. Shown are subsets of absolute adjacency matrices. Columns refer to transcription factors and rows refer to target genes associated with Erythroid lineage. The expression of four genes across pseudo-time are presented on right. Shown are two relevant (MAZ and KLF1) and two spurious genes (GRHL1 and E2F1). Evident by the adjacency matrices, PathReg and L0 remove the links *to* the spurious target genes or *from* the spurious transcription factors.

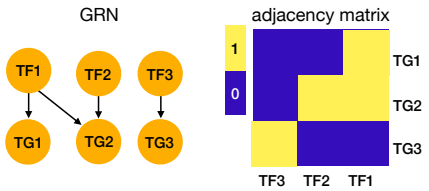
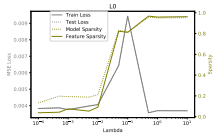
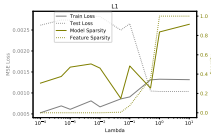
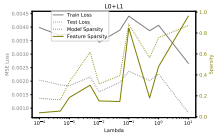
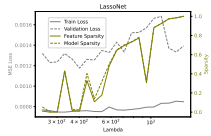
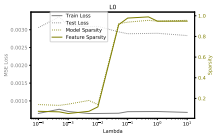
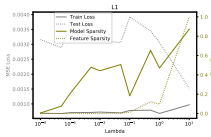
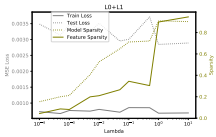
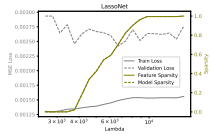
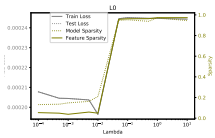
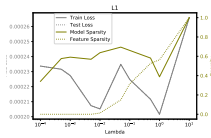
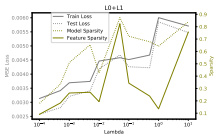
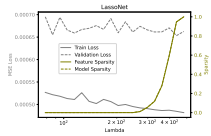
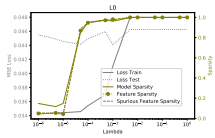
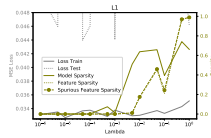
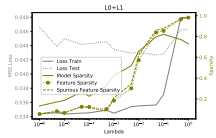
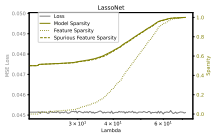


Figure 11: A GRN example compared with the true adjacency matrix (shown is a subset referring to TF-TG interactions) that we expect a model to learn.

Table 7: Effect of the regularization weight on performance and sparsity of different datasets.

DATASET	L0	C-NODE	PATHREG	LASSONET
MCD - WALKING				
MCD - GOLFING				
MCD - WAVING				
SINGLE-CELL