scNODE : Generative Model for Temporal Single Cell Transcriptomic Data Prediction

A Preprint

Jiaqi Zhang
Department of Computer Science
Brown University

Erica Larschan
Department of Molecular Biology, Cell Biology and Biochemistry
Center for Computational Molecular Biology
Brown University

Jeremy Bigness
Center for Computational Molecular Biology
Brown University
jeremy_bigness@brown.edu

Ritambhara Singh
Department of Computer Science
Center for Computational Molecular Biology
Brown University
ritambhara@brown.edu

Abstract

Measurement of single-cell gene expression at different timepoints enables the study of cell development. However, due to the resource constraints and technical challenges associated with the single-cell experiments, researchers can only profile gene expression at discrete and sparsely-sampled timepoints. This missing timepoint information impedes downstream cell developmental analyses. We propose scNODE, an end-to-end deep learning model that can predict *in silico* single-cell gene expression at unobserved timepoints. scNODE integrates a variational autoencoder (VAE) with neural ordinary differential equations (ODEs) to predict gene expression using a continuous and non-linear latent space. Importantly, we incorporate a dynamic regularization term to learn a latent space that is robust against distribution shifts when predicting single-cell gene expression at unobserved timepoints. Our evaluations on three real-world scRNA-seq datasets show that scNODE achieves higher predictive performance than state-of-the-art methods. We further demonstrate that scNODE's predictions help cell trajectory inference under the missing timepoint paradigm and the learned latent space is useful for *in silico* perturbation analysis of relevant genes along a developmental cell path. The data and code are publicly available at https://github.com/rsinghlab/scNODE.

1 Introduction

A fundamental challenge in biology is understanding how gene expression changes over time during cell development (Moris et al., 2016; Fleck et al., 2022). With the advent of high-throughput single-cell RNA sequencing (scRNA-seq), researchers now routinely profile gene expression at single-cell resolution (Hwang et al., 2018; Chen et al., 2019), revealing substantial heterogeneity within the same tissue type or developmental stage (Chen et al., 2022a; Ding et al., 2022). Profiling scRNA-seq data at multiple timepoints allows us to understand how biological processes unfold within cell populations by inferring cellular trajectories, identifying cell fate transitions, and characterizing differentially expressed genes (Qiu et al., 2022; Liu et al., 2022).

Temporal scRNA-seq experiments, however, have significant limitations that impede developmental studies. Due to the substantial time and resources involved in conducting these experiments, researchers generally only profile gene expression at discrete and sparsely-sampled timepoints (Ding et al., 2022). Since it is infeasible to make continuous-time observations, this results in information loss between consecutively measured discrete timepoints. Therefore, it is crucial to develop a model that predicts realistic *in silico* gene expression at any timepoint, whether between (interpolation) or beyond the measured time intervals (extrapolation), to enable single-cell temporal analyses (Fig. 1**A-B**).

Single-cell gene expression prediction has been investigated in previous studies (Cao et al., 2021) through various generative models. Existing methods use generative adversarial networks (GANs) (Marouf et al., 2020) or statistical frameworks (Zappia et al., 2017; Risso et al., 2018; Li and Li, 2019; Sun et al., 2021) to fit characteristics of the observed single-cell gene expression and augment the dataset. However, these methods are designed for single-cell gene expression measured at a single time snapshot and do not consider developmental dynamics.

Several computational methods have been developed to learn gene expression dynamics by inferring cell trajectories (Trapnell et al., 2014; Qiu et al., 2017; Herman et al., 2018; Saelens et al., 2019; Wolf et al., 2019; Li, 2023) from temporal scRNA-seq. However, most of them infer pseudo-times, which analyze cell differentiation based on geometric distances in low-dimensional spaces. Hence, they do not accurately model how cells develop in real physical time. Sagittarius method (Woicik et al., 2023) predicts gene expressions at future timepoints but requires cell matching between timepoints. This matching is hard to obtain for real-world scRNA-seq as the cells are lysed during the experiment. RNA velocity based methods (La Manno et al., 2018; Bergen et al., 2021, 2020) are used to uncover developmental trends and underlying kinetics of gene expression. These methods require deeply sequenced datasets to obtain spliced and unspliced counts as prior information, and most of them are non-generative methods. Waddington-OT (Schiebinger et al., 2019) and TrajectoryNet (Tong et al., 2020) interpolate single-cell gene expression between two timepoints in a developmental trajectory but are unable to extrapolate beyond the observed timepoints.

Recently, a few generative methods have modeled cell differentiation trajectory and predicted gene expression at multiple unobserved time points. For example, PRESCIENT (Yeo et al., 2021) applies principal component analysis (PCA) to reduce high-dimensional gene space to a low-dimensional representation. It then applies neural ordinary differential equations (ODEs) (Chen et al., 2018) in this low-dimensional space to model the cell developmental trajectory. Multiple studies (Ding et al., 2018; Xiang et al., 2021; Tran et al., 2021) have found that PCA-based low-dimensional representations of scRNA-seq expression cannot capture complex relationships in the highly heterogeneous single-cell data. Thus, PCA may have the issue of overcrowding representation (Tran et al., 2021; Kobak and Berens, 2019), where cells of different types are poorly separated in PCA-based low-dimensional space, and cellular variations are lost. MIOFlow (Huguet et al., 2022) replaces PCA with a geodesic variational autoencoder (VAE) to learn non-linear low-dimensional representations that better retain cellular variations. However, this method fixes its low-dimensional space at the beginning, followed by neural ODE modeling. We hypothesize that a fixed low-dimensional representation obtained using observed timepoints may not generalize to an unobserved timepoint, if its underlying distribution differs substantially.

We propose a single-cell neural ODE (scNODE) framework, which integrates a VAE (Kingma et al., 2019) with neural ODE to predict accurate gene expression at any unmeasured timepoint using a dynamic low-dimensional space (Fig. 1**C**). First, scNODE uses VAE to obtain a low-dimensional representation that retains cellular variations. scNODE then predicts single-cell gene expression for unmeasured timepoints with a neural ODE. Importantly, scNODE uses a *dynamic regularization* term to incorporate the dynamic manifold learned from neural ODE as a prior. Therefore, the learned VAE space is robust to the distribution shift between observed and unobserved time points. We evaluate scNODE on three real-world scRNA-seq datasets. Our results demonstrate that scNODE predicts single-cell gene expression more accurately than state-of-the-art methods for unobserved timepoints (interpolation and extrapolation). We further show that scNODE gene expression predictions assist with cell developmental trajectory inference. Finally, scNODE learns an interpretable low-dimensional space, which enables conducting *in silico* perturbation analysis of relevant genes to study cell development.

2 Method

Notation: Let $\mathbf{X}^{(t)} \in \mathbb{R}^{n_t \times p}$ denote the gene expression matrix at the t-th time point of n_t cells by p genes. Given gene expression measurements at observed timepoints $\{\mathbf{X}^{(t)}\}_{t \in \mathcal{T}}$, our goal is to predict gene expression at any timepoint. Here, $\mathcal{T} \subseteq \{0, 1, 2, \cdots\}$ denotes the observed timepoint indices. Note that we assume the same set of highly variable genes (HVGs) is measured at each time point, but due to the destruction of cells during scRNA experiments, a different set of cells is measured at each time point. Furthermore, let $\mathbf{Z}^{(t)} \in \mathbb{R}^{n_t \times d}$ denote the d-dimensional ($d \ll p$) latent variable at the t-th time point.

2.1 scNODE uses VAE to learn complex low-dimensional space

scNODE uses a Variational Autoencoder (VAE) to learn a low-dimensional representation (also known as latent space) of the single-cell gene expression measurements at the observed timepoints. A VAE is a neural network-based generative model that maps high-dimensional data to a low-dimensional representation and is widely used in many single-cell studies (Grønbech et al., 2020; Svensson et al., 2020; Wang and Gu, 2018; Lin et al., 2020), showing superior

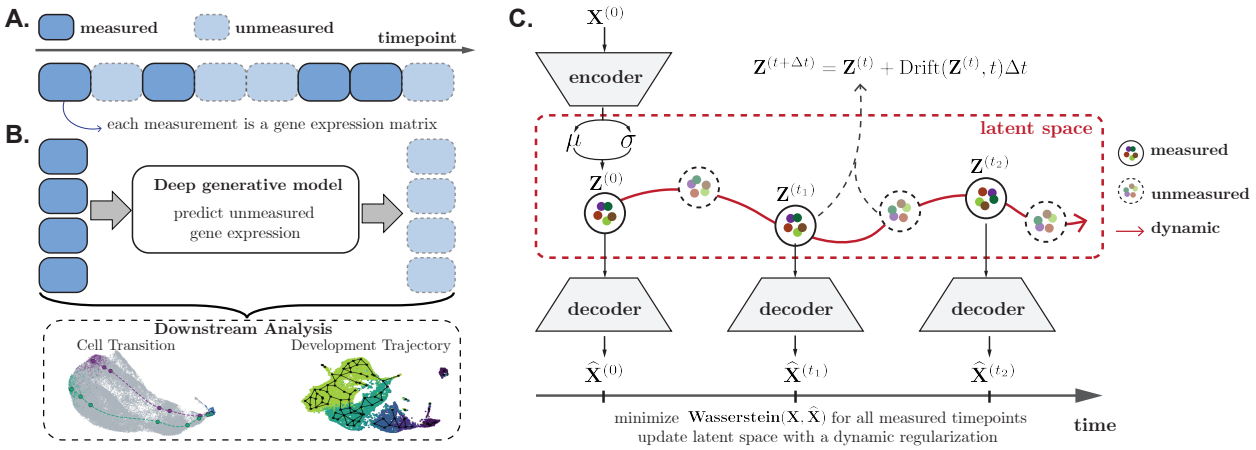

Figure 1: **Model overview.** (**A**) The measurements in temporally recorded single-cell data are sparse and unaligned, such that some timepoints are not measured and measurements at different timepoints contain different sets of cells. The data sets in this study sample cellular populations at uniformly spaced time points, although uniform spacing is not a requirement (as depicted in this graphic). (**B**) scNODE is used to predict expression at unmeasured timepoints. Predictions can be used to analyze cell state transitions and developmental trajectories. (**C**) scNODE uses VAEs to find an optimal low-dimensional representation. scNODE then uses this representation to solve neural ODEs and predict gene expression at any timepoint $t = t_1, t_2, \cdots$ starting from the first timepoint $t = 0$. Optimal models are fitted by minimizing the Wasserstein distance between observations and predictions at measured timepoints. scNODE also introduces a dynamic regularization to learn a latent space robust to distribution shift.

performance. The benefit of using VAEs for single-cell data over PCA is that the non-linearity of neural networks can more effectively capture complex cell relationships and cellular variations.

scNODE first pre-trains a VAE to obtain a latent space that captures the gene expression of single cells at all available training timepoints. Specifically, scNODE trains the VAE component to perform multi-timepoint modeling by inputting the gene expression of all observed cells $\mathbf{X}_{\text{ALL}} = \text{CONCAT}\left(\mathbf{X}^{(t)} \mid t \in \mathcal{T}\right)$. VAE is composed of two neural networks: (1) the encoder network Enc_ϕ maps expression from gene space $\mathbb{R}^p$ to a low-dimensional latent space $\mathbb{R}^d$, parameterized by a Gaussian distribution $\mathcal{N}(\mu, \sigma)$, and (2) the decoder network Dec_θ maps samples in the latent space to gene space in the opposite direction to reconstruct the input. Given $\mathbf{X}_{\text{ALL}}$, VAE learns latent representations through

$$\mu_{\text{ALL}}, \sigma_{\text{ALL}} = \text{Enc}_\phi(\mathbf{X}_{\text{ALL}}), \quad \mathbf{Z}_{\text{ALL}} \sim \mathcal{N}(\mu_{\text{ALL}}, \sigma_{\text{ALL}}), \quad \widehat{\mathbf{X}}_{\text{ALL}} = \text{Dec}_\theta(\mathbf{Z}_{\text{ALL}}). \tag{1}$$

The encoder and decoder networks are parameterized by ϕ and θ, correspondingly. VAE minimizes a combination of the (1) mean squared error (MSE) between input gene expression and the reconstructed gene expression from the decoder and (2) the Kulback-Leibler (KL) divergence (Csiszár, 1975) between the latent distribution and a standard Gaussian prior:

$$\mathcal{L}_{\text{pre}} = \text{MSE}\left(\mathbf{X}_{\text{ALL}},\ \widehat{\mathbf{X}}_{\text{ALL}}\right) + \lambda \text{KL}\left(\mathcal{N}(\mu_{\text{ALL}}, \sigma_{\text{ALL}}),\ \mathcal{N}(0, 1)\right). \tag{2}$$

KL divergence is an information-theoretic quantity that quantifies the distance between two probability distributions. By using a Gaussian prior, the KL divergence forces the encoder to find latent representations of the cells that are well-separated. scNODE pre-trains the VAE component with Eq. 2, such that the encoder latent space preserves the variation of all observed cells. Next, we model the developmental dynamics of this latent space.

2.2 scNODE uses neural ODE to model cell dynamics

scNODE uses ordinary differential equations (ODEs) to model cell developmental dynamics in the latent space learned by the VAE. An ODE is an equation describing how a quantity x changes with respect to an independent variable y, such that $\mathrm{d}x = f(x; y)\mathrm{d}y$ where function f represents the derivative. Therefore, we can use differential equations to model how gene expression changes with respect to time. However, for the high-dimensional data, finding the solution of the derivative function f through numerical methods is intractable and computationally expensive (Kidger, 2022). Therefore, recent studies adopt neural networks to approximate the derivative function and have proposed neural ODEs (Chen et al., 2018). Neural ODEs (formulated with respect to time) are useful for constructing continuous-time

trajectories and have been explored before to model single-cell development (Matsumoto et al., 2017; Qiu et al., 2022; Chen et al., 2022b).

scNODE uses neural ODEs to parameterize the continuous dynamics of gene expression in the latent space. Specifically, scNODE quantifies changes of cell latent representation $\mathbf{Z}^{(t)}$ (from VAE encoding) at time t through a neural ODE

$$\mathrm{d}\mathbf{Z}^{(t)} = \mathrm{Drift}_{\omega}\left(\mathbf{Z}^{(t)}; t\right) \cdot \mathrm{d}t \tag{3}$$

Here, Drift_{ω} is a non-linear neural network with parameterization ω, modeling the developmental velocities in the latent space, such that $\mathrm{Drift}_{\omega}(\mathbf{Z}^{(t)}; t)$ represents the direction and strength of cellular transitions. scNODE computes the latent representation $\mathbf{Z}^{(0)}$ of cells at the first timepoint $t = 0$ through the encoder network (pre-trained in the previous step) and predicts the subsequent cell states step-wise at any timepoint t as follows:

$$\begin{aligned} \mathbf{Z}^{(0)} \sim \mathcal{N}(\mu_0, \sigma_0) \quad \text{and} \quad \mu_0, \sigma_0 = \mathrm{Enc}_{\phi}(\mathbf{X}^{(0)}), \\ \mathbf{Z}^{(t+\Delta t)} = \mathbf{Z}^{(t)} + \mathrm{Drift}_{\omega}\left(\mathbf{Z}^{(t)}, t\right) \Delta t. \end{aligned} \tag{4}$$

Here, hyperparameter Δt denotes step size and drift term $\mathrm{Drift}_{\omega}\left(\mathbf{Z}^{(t)}, t\right) \Delta t$ represents the forward steps taken in the latent space. While there are several methods that can solve this ODE, in this work, we apply the commonly used first-order Euler method (in Eq. 4) for convenience of explanation. However, in our implementation, one can specify any ODE solver.

To fit the continuous trajectory (controlled by Drift_{ω}) to the observations, scNODE minimizes the difference between the input and the reconstructed gene expression. Specifically, at each measured timepoint $t \in \mathcal{T}$, scNODE uses the decoder Dec_{θ} to convert latent variables $\mathbf{Z}^{(t)}$ generated from Eq. 4 back to the gene space through $\widehat{\mathbf{X}}^{(t)} = \mathrm{Dec}_{\theta}\left(\mathbf{Z}^{(t)}\right)$. Because we have no correspondence between true cells and cells generated from the ODE model, scNODE utilizes the Wasserstein metric (Cuturi, 2013) to measure the distance between distributions defined by ground truth $\mathbf{X}$ and predictions $\widehat{\mathbf{X}}$ as

$$\mathrm{Wass}(\mathbf{X}, \widehat{\mathbf{X}}) = \left(\min_{\Gamma \sim \Pi(\mathbf{X}, \widehat{\mathbf{X}})} \sum_{i,j} D_{ij}^2 \Gamma_{ij} \right)^{1/2} \quad \text{with} \quad D_{ij} = \|\mathbf{X}_i - \widehat{\mathbf{X}}_j\|_2. \tag{5}$$

Here, $\Pi(\mathbf{X}, \widehat{\mathbf{X}})$ denotes the set of all transport plans between each cell of $\mathbf{X}$ and $\widehat{\mathbf{X}}$ and D_{ij} represents the ℓ_2 distance, such that the Wasserstein metric adopts the minimal-cost transport plan Γ to measure the data dissimilarity.

2.3 scNODE uses dynamic regularization for learning a robust latent space

The VAE and its corresponding latent space are trained using training (or observed) timepoints. In the temporal scRNA-seq data where gene expression distribution changes substantially, a fixed low-dimensional representation obtained using training timepoints may not generalize to a testing (or unobserved) timepoint. To overcome this distribution shift issue, scNODE uses a **dynamic regularization** term to update the VAE space dynamically such that it captures both cellular variations and the developmental dynamics of the scRNA-seq data. Specifically, scNODE minimizes the difference between the latent variables generated by the VAE (i.e., $\mathrm{Enc}_{\phi}(\mathbf{X})$) and the dynamics learned by the ODE (i.e., $\mathbf{Z}$). Because we have no correspondence between them, scNODE again uses Wasserstein distance to evaluate their difference at each observed timepoint $t \in \mathcal{T}$ and defines the dynamic regularization as

$$\begin{aligned} \mathcal{R}(\mathcal{T}) = \sum_{t \in \mathcal{T}} \mathrm{Wass}\left(\mathbf{Z}_{\text{enc}}^{(t)},\ \mathbf{Z}^{(t)}\right) \text{ with} \\ \mathbf{Z}_{\text{enc}}^{(t)} \sim \mathcal{N}(\mu_t, \sigma_t) \text{ and } \mu_t, \sigma_t = \mathrm{Enc}_{\phi}(\mathbf{X}^{(t)}), \\ \mathbf{Z}^{(t)} \text{ comes from neural ODE Eq. 4} \end{aligned} \tag{6}$$

Therefore, scNODE jointly updates VAE and neural ODE, optimizing parameters ϕ, θ, and ω, by minimizing the regularized loss function

$$\mathcal{L}_{\text{dyn}} = \sum_{t \in \mathcal{T}} \mathrm{Wass}\left(\mathbf{X}^{(t)},\ \widehat{\mathbf{X}}^{(t)}\right) \ + \ \beta \mathcal{R}(\mathcal{T}), \tag{7}$$

so that the overall dynamics update the final latent space of scNODE through dynamic regularization and corresponding hyperparameter β.

Our dynamic regularization improves upon previous generative models (Schiebinger et al., 2019; Tong et al., 2020; Huguet et al., 2022; Yeo et al., 2021), which fix the latent space at the beginning and then learn the cell dynamics.

scNODE uses the dynamic manifold learned from neural ODE as a prior and enforces VAE latent space to incorporate it using dynamic regularization. Therefore, updating the VAE ensures that scNODE fits the data better and learns a latent space that is robust to distribution shift when generating gene expression for unobserved timepoints. Previous work by Connor et al. (2021) similarly introduced a linear dynamic system-based model to learn a smooth low-dimensional manifold and then incorporated this manifold (as a prior) into VAE to encourage the latent space to fit the data manifold better. This resolved the issue that VAE latent representations sometimes would not match the true data manifold and poorly defined natural paths between data points. Therefore, our dynamic regularization will result in improved representation learning.

3 Results

3.1 Experimental Setup

Datasets We use three publicly available scRNA-seq datasets to demonstrate the capabilities of scNODE in predicting developmental dynamics from real-world single-cell gene expression data. These datasets are summarized in Table 1, have more than 10 timepoints, and cover various species and tissues. For each timepoint, gene expression is measured at a developmental stage (ZB), every hour (DR), or every 12 hours (SC). To make computations tractable, we relabel timepoints with consecutive natural numbers starting from 0. In particular, the meaning of interpolating or extrapolating is defined by these data-specific units. For example, extrapolating one timepoint in the DR dataset means predicting gene expressions for the next hour. We use the data after removing batch effects among different timepoints. In each experiment, we select the top 2000 most highly variable genes (HVGs) from the datasets and normalize the unique molecular identifier (UMI) count expression through a log transformation with pseudo-count.

Table 1: Data descriptions of the three real-world scRNA-seq datasets used in experiments.

ID	Dataset	Species	# Cells	# Timepoints	Source
ZB	zebrafish embryo	*Danio rerio*	38731	12	Broad Single-Cell Portal SCP162 (Farrell et al., 2018)
DR	drosophila	*Drosophila melanogaster*	27386	11	GSE190149 (Calderon et al., 2022)
SC	Schiebinger2019	*Mus musculus*	236285	19	(Schiebinger et al., 2019)

Training and testing We test scNODE 's performance in predicting gene expression at unobserved timepoints. Specifically, for each dataset, we remove several timepoints to test whether scNODE can recover these left-out observations. We design three tasks: (1) **easy** tasks where we remove uniformly spaced timepoints in the middle of the measured time range for interpolating, (2) **medium** task where we remove the last few timepoints for extrapolating beyond the measured time range, and (3) **hard** tasks where we combine the interpolation and extrapolation schemes. Supplementary Table S1 shows the timepoints removed in each task. We consider the left-out timepoints to be the testing set, and the remaining timepoints are used to train the model. In each case, HVGs are selected based on cells corresponding to the training timepoints in order to avoid data leakage.

Baselines We compare scNODE with two main state-of-the-art generative methods that are capable of both interpolating and extrapolating gene expression:

- **PRESCIENT**: Yeo et al. (2021) propose a generative model, called Potential eneRgy undErlying Single Cell gradIENTs (PRESCIENT), to learn the differentiation landscape from single-cell time-series gene expression data. PRESCIENT maps single-cell gene expression to a lower-dimensional PCA space and models cell differentiation with a neural ODE.
- **MIOFlow**: Huguet et al. (2022) integrate geodesic VAE and neural ODE to model the paths of cells in a lower-dimensional latent space while cellular variations are preserved.

Hyperparameter tuning In each task for every dataset, we select corresponding hyperparameters for all methods (our and baselines) that yield the minimum averaged Wasserstein distance using the 3-fold cross-validation scheme. We use Optuna (Akiba et al., 2019) to automatically determine the optimal hyperparameters and use sufficiently large hyperparameter ranges for search and evaluation. The hyperparameter ranges of scNODE and baselines are listed in Supplementary Table S2. We set the latent space dimension as 50, use the first-order Euler ODE solver, and set ODE step size $\Delta t = 0.1$ for all methods, and run each method for sufficient iterations to ensure they converge. We let every model predict 2000 cells at each testing timepoint to ensure a fair metric comparison. Moreover, we evaluate scNODE performance using different hyperparameter settings, conduct ablation studies, and give heuristic guidance on how to set hyperparameters in real-world scenarios in Supplementary Sec. S6.3.

3.2 scNODE can accurately predict gene expression at unobserved timepoints

We compare scNODE 's performance with baseline methods for predicting gene expression for left-out timepoints. Fig. 2 (along with Supplementary Fig. S1∼S3) visualizes true gene expression values and model predictions for left-out testing timepoints in 2D Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018) space. Our results indicate scNODE 's predictions align well with ground truth. Qualitative evaluation of all scRNA-seq datasets indicates that scNODE can accurately predict gene expression at missing timepoints.

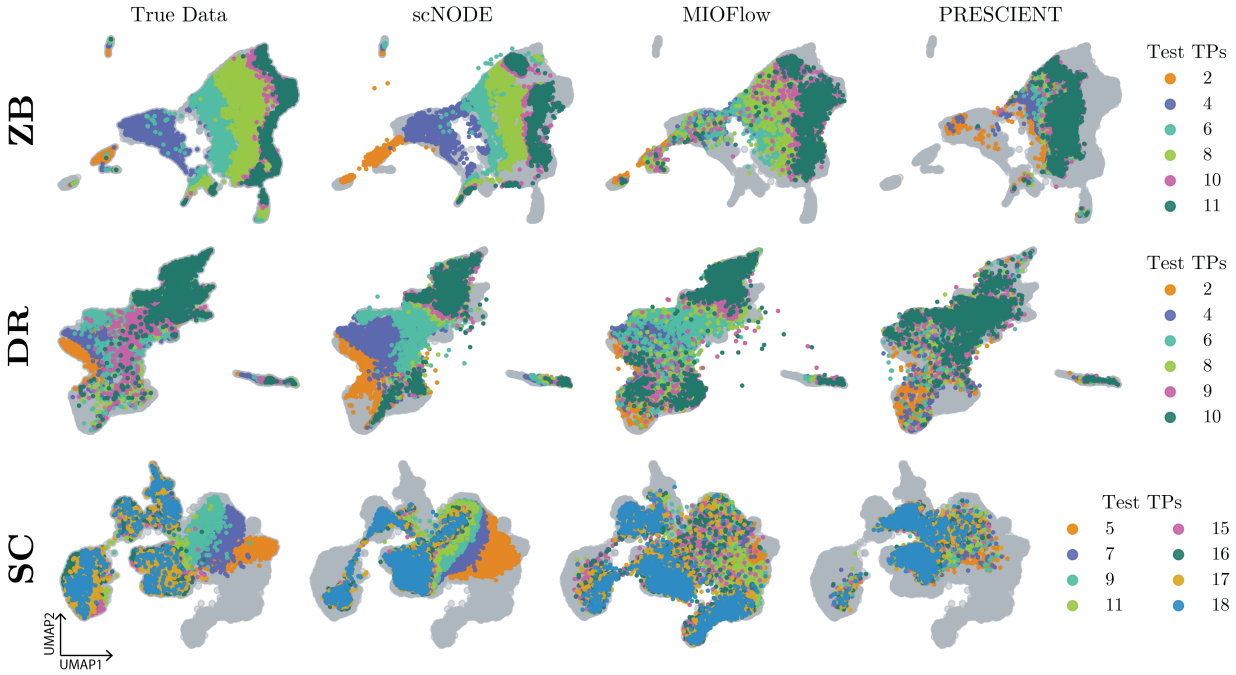

Figure 2: 2D UMAP visualization of true and predicted gene expressions in hard tasks. Gray points represent training data.

We then quantitatively evaluate model predictions. We use the Wasserstein distance to measure the similarity between true and predicted gene expression at each testing timepoint, where a lower value corresponds to more accurate predictions. This evaluation metric has been used in previous studies (Yeo et al., 2021; Tong et al., 2020) because Wasserstein distance learns a soft mapping across the distributions, and we do not have one-to-one cell matching between true and predicted gene expression for single-cell measurements. Table 2 shows the Wasserstein distance of left-out testing timepoints averaged on five trials for defined hard tasks, and Supplementary Table S3 shows metrics for easy and medium tasks. scNODE clearly outperforms the baselines in most cases. In other cases where scNODE has the second best predictions, such as $t = 7$ of SC hard task, scNODE has similar performance as MIOFlow but significantly performs better than PRESCIENT. Moreover, in medium and hard tasks, where extrapolations are required, scNODE can have substantial improvement over baselines. For example, at $t = 15$ of SC hard task (in Table 2), Wasserstein distance of scNODE predictions is around 132, while that of PRESCIENT and MIOFlow are about 150 and 162 respectively. In Supplementary Fig. S4, we further evaluate all model predictions when extrapolating more timepoints. As expected, their extrapolations become less accurate the farther out predictions are made. But scNODE still demonstrates improvement in most cases. Therefore, scNODE has consistent good performance, especially in more challenging prediction tasks.

Next, we validate that scNODE is more robust to distribution shift when testing timepoints have substantially different distributions from training data. Specifically, for the hard task of all three datasets, we compute the distribution shift level for each testing timepoint as the averaged pairwise ℓ_2 distance between cells from training and testing timepoints. Hence, a higher value indicates the testing point has a more significant difference from the training data. Moreover, we define scNODE 's improvement as the difference between the Wasserstein distances of the predictions made by scNODE and the second-best baseline, such that a higher value indicates that scNODE obtains more improvements. Fig. 3 shows that scNODE improvements positively correlate with the distribution shift level (Spearman correlation $\rho \geq 0.3$ and is as high as 0.93.). Therefore, due to the dynamic regularization introduced in our framework, scNODE is more robust to the distribution shift between the observed and unobserved timepoints and obtains significant improvements over baseline models.

Table 2: Wasserstein distance of predictions in hard tasks (inter- and extrapolation). **Bold** numbers denote the best prediction, and underlined numbers represent the second best.

Method	ZB / Hard Task					
	Interpolation				Extrapolation	
	$t = 2$	$t = 4$	$t = 6$	$t = 8$	$t = 10$	$t = 11$
scNODE	**579.10**	**508.55**	**440.92**	**517.81**	**652.36**	**707.10**
MIOFlow	580.18	516.59	453.61	536.35	671.23	734.42
PRESCIENT	1381.96	1002.62	730.974	701.29	916.51	973.17

Method	DR / Hard Task					
	Interpolation			Extrapolation		
	$t = 2$	$t = 4$	$t = 6$	$t = 8$	$t = 9$	$t = 10$
scNODE	445.82	**464.78**	535.78	**600.18**	585.60	**718.20**
MIOFlow	**443.56**	469.51	**532.93**	617.48	680.41	852.02
PRESCIENT	524.38	511.61	539.38	621.31	**575.45**	718.56

Method	SC / Hard Task							
	Interpolation				Extrapolation			
	$t = 5$	$t = 7$	$t = 9$	$t = 11$	$t = 15$	$t = 16$	$t = 17$	$t = 18$
scNODE	55.22	**59.89**	**103.26**	**140.81**	**132.86**	**148.89**	**137.90**	**151.13**
MIOFlow	**55.07**	61.80	108.72	156.51	162.12	191.40	189.39	215.74
PRESCIENT	85.36	87.47	114.16	142.03	150.53	161.59	147.23	155.06

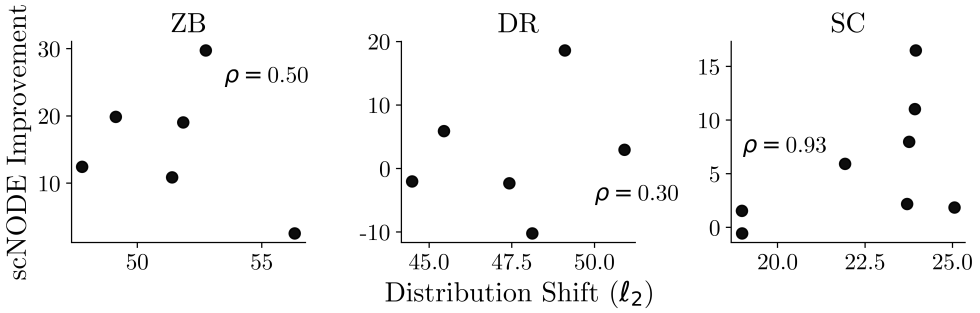

Figure 3: scNODE improvements over baseline models vs. distribution shift levels of testing points on hard tasks. Each point denotes one testing timepoint with scNODE improvements averaged on five trials. ρ: Spearman's ρ correlations between improvements and distribution shift.

3.3 scNODE predictions for unobserved timepoints help recover cell trajectories

To determine if scNODE can aid with temporal downstream analysis, we carry out trajectory analysis on both real and predicted single-cell gene expression. Specifically, we focus on the hard task of all three datasets. Fig. 4 visualizes the ZB data with all timepoints, after the removal of testing timepoints, and with predictions from all models. We apply partition-based graph abstraction (PAGA) (Wolf et al., 2019), a method that computes the topological structures of cellular populations and has been used in previous studies (Saelens et al., 2019; Luecken and Theis, 2019; Han et al., 2020) to understand cell developmental topologies. We find that after timepoint removal, the topology breaks down due to the gaps between timepoints, which can impede the trajectory analysis, while scNODE predictions recover smooth and continuous trajectories. To quantitatively compare scNODE with other models in helping infer cell trajectories, we use the Ipsen-Mikhailov (IM) distance (Ipsen, 2004; Saelens et al., 2019) to measure the similarity between the cell trajectory graphs constructed in each case. $\text{IM}(\mathcal{G}_1, \mathcal{G}_2)$ is a graph similarity measurement defined as the square-root difference between the Laplacian spectrum of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. It ranges from 0 to 1, where 0 indicates maximum similarity between two graph structures and 1 indicates maximum dissimilarity. We find $\text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{removal}}) = 0.200$ and $\text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{scNODE}}) = 0.093$, indicating $\mathcal{G}_{\text{scNODE}}$ is more similar to $\mathcal{G}_{\text{true}}$ than $\mathcal{G}_{\text{removal}}$ such that scNODE predictions help recovering cell trajectories. Moreover, $\text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{scNODE}})$ being smaller than $\text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{MIOFlow}})$ and $\text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{PRESCIENT}})$ (Fig. 4) implies that scNODE predictions for missing timepoints best help to infer cell trajectories. We observe same trend for DR and SC datasets as well, where IM distance is lowest for scNODE (Supplementary Fig. S8).

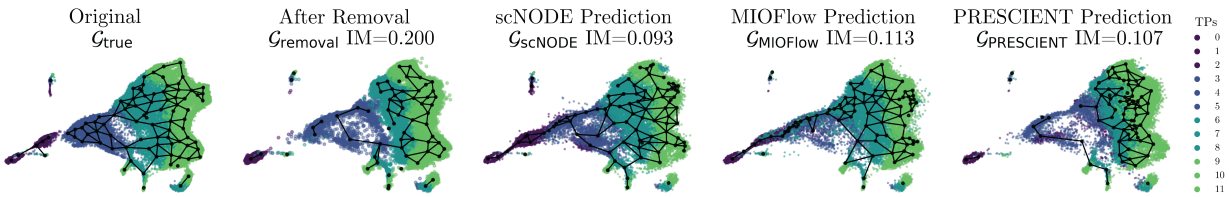

Figure 4: Cell trajectories of ZB data with all timepoints, after removal of timepoints in the hard task, and with model predictions. The connective structure is constructed with PAGA, where black nodes represent cell clusters and edges connect two nodes if their expressions are similar. We show the IM index between $\mathcal{G}_{\text{true}}$ and the corresponding graph.

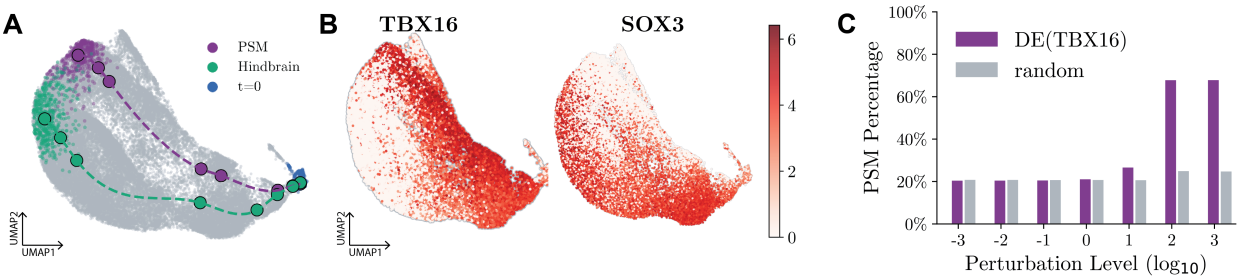

Figure 5: scNODE perturbation analysis results. (**A**) 2D UMAP visualization of the least action path between cells at the starting point ($t = 0$) and the PSM and Hindbrain cell populations, respectively. (**B**) Gene expression value of the top-rank DE gene of PSM path (TBX16) and Hindbrain path (SOX3). (**C**) The ratio of PSM cells in predictions of different perturbation levels when perturbing DE genes and random non-DE genes.

3.4 scNODE 's interpretable latent space assists with perturbation analysis

scNODE 's ability to learn an informative and robust latent space allows it to detect key driver genes for cell developmental paths and simulate cells with perturbed gene expression. This ability can be useful for performing *in silico* perturbation predictions. We validate this hypothesis by using the differential dynamics learned from the ZB dataset. We choose ZB data because the original dataset provides cell type labels at the last timepoint, which enables us to evaluate the perturbation predictions.

We first train scNODE with all timepoints in the ZB dataset and map all cells to the latent space with the learned scNODE encoder. In the latent space, we construct a most probable path between any two points through the Least Action Path (LAP) method (Qiu et al., 2022, 2012; Wang et al., 2014), which finds the optimal path between two cell states while minimizing its action and transition time (see Supplementary Sec. S6.5). Here, we construct LAP paths from cells at the starting point (i.e., $t = 0$) to presomitic mesoderm (PSM) and Hindbrain cell populations (Fig. 5**A**)

Once the optimal path is constructed, we use the Wilcoxon rank-sum test to find differentially expressed (DE) genes for each path, representing key driver genes for the developmental trajectories. Fig. 5 shows the expression of top-rank DE genes for the PSM path (TBX16) and Hindbrain path (SOX3), such that their respective expression levels vary along the path (as well as in adjacent regions). Warga et al. (2013) have found TBX16 regulates intermediate mesoderm cell fate. Other studies (Dee et al., 2008; Vriz et al., 1996) have found SOX3 regulates neural fates and is involved in central nervous system development.

Finally, we perturb the expression profiles of key genes in all cells at the starting timepoint by multiplying their expression values with different levels of coefficient $\{10^{-3}, \cdots, 10^{3}\}$ to mimic overexpressing and knocking-out, and let scNODE predict trajectories for the perturbed gene expression. The perturbations are expected to result in changes in cell fates. We classify predicted cells with a Random Forest (RF) classifier (Breiman, 2001) trained with all unperturbed cells. We find overexpressing TBX16 expression results in the increment of PSM cell ratios (Fig. 5**C**) to over 70%, while perturbing random non-DE genes leads to no changes in cell ratios. Moreover, the ratio of Hindbrain cells decreased by around 25% when we overexpressed SOX3 expressions (Supplementary Fig. S9). These results indicate that scNODE learns an interpretable latent space, where we can detect development-related DE genes and conduct *in silico* perturbation analysis.

4 Conclusion

We propose a generative model called scNODE to predict gene expression during cell development. Given a time-series scRNA-seq data set, our model can both interpolate and extrapolate gene expression. Predicting gene expression for unobserved timepoints enables critical downstream analyses, such as differential gene expression analysis, perturbation analysis, and trajectory inference. To make accurate predictions, scNODE uses a non-linear dimensionality reduction approach, neural ODE, and dynamic regularization. The regularization improves the robustness of the learned latent representation to distribution shifts in unobserved timepoints, resulting in better predictive performance.

We find that the primary challenge for the temporal scRNA-seq prediction task is extrapolating to multiple timepoints beyond the last observed timepoint, as the accuracy deteriorates the farther timepoint predictions. Moreover, time units in predictions are defined by time intervals in the data. For example, a generative model cannot accurately produce gene expression for every hour if the model is trained on observations that are only measured every day, especially when the development is rapid or contains complex trajectories.

In future work, we will incorporate knowledge about cell proliferation - which is known to be important to cellular differentiation - into the model to improve extrapolating performance. Also, although RNA velocity methods can be used as complementary approaches to scNODE to learn fine-grained dynamics from coarse-grained timepoints. We will also incorporate other modalities, such as single-cell chromatin accessibility (scATAC) datasets, which exhibit a regulatory mechanism that scRNA-seq data might not capture.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Ball, T. M., Squeglia, L. M., Tapert, S. F., and Paulus, M. P. (2020). Double dipping in machine learning: problems and solutions. *Biological psychiatry. Cognitive neuroscience and neuroimaging*, 5(3):261.

Bergen, V., Lange, M., Peidli, S., Wolf, F. A., and Theis, F. J. (2020). Generalizing rna velocity to transient cell states through dynamical modeling. *Nature biotechnology*, 38(12):1408–1414.

Bergen, V., Soldatov, R. A., Kharchenko, P. V., and Theis, F. J. (2021). Rna velocity—current challenges and future perspectives. *Molecular systems biology*, 17(8):e10282.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

Calderon, D., Blecher-Gonen, R., Huang, X., Secchia, S., Kentro, J., Daza, R. M., Martin, B., Dulja, A., Schaub, C., Trapnell, C., et al. (2022). The continuum of drosophila embryonic development at single-cell resolution. *Science*, 377(6606):eabn5800.

Cao, Y., Yang, P., and Yang, J. Y. H. (2021). A benchmark study of simulation methods for single-cell rna sequencing data. *Nature communications*, 12(1):6911.

Chen, G., Ning, B., and Shi, T. (2019). Single-cell rna-seq technologies and related computational data analysis. *Frontiers in genetics*, page 317.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Chen, W., Guillaume-Gentil, O., Rainer, P. Y., Gäbelein, C. G., Saelens, W., Gardeux, V., Klaeger, A., Dainese, R., Zachara, M., Zambelli, T., et al. (2022a). Live-seq enables temporal transcriptomic recording of single cells. *Nature*, 608(7924):733–740.

Chen, Z., King, W. C., Hwang, A., Gerstein, M., and Zhang, J. (2022b). Deepvelo: Single-cell transcriptomic deep velocity field learning with neural ordinary differential equations. *Science Advances*, 8(48):eabq3745.

Connor, M., Canal, G., and Rozell, C. (2021). Variational autoencoder with learned latent structure. In *International conference on artificial intelligence and statistics*, pages 2359–2367. PMLR.

Csiszár, I. (1975). I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, pages 146–158.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26.

Dee, C. T., Hirst, C. S., Shih, Y.-H., Tripathi, V. B., Patient, R. K., and Scotting, P. J. (2008). Sox3 regulates both neural fate and differentiation in the zebrafish ectoderm. *Developmental Biology*, 320(1):289–301.

Ding, J., Condon, A., and Shah, S. P. (2018). Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature communications*, 9(1):2002.

Ding, J., Sharon, N., and Bar-Joseph, Z. (2022). Temporal modelling using single-cell transcriptomics. *Nature Reviews Genetics*, 23(6):355–368.

Farrell, J. A., Wang, Y., Riesenfeld, S. J., Shekhar, K., Regev, A., and Schier, A. F. (2018). Single-cell reconstruction of developmental trajectories during zebrafish embryogenesis. *Science*, 360(6392):eaar3131.

Farrell, S., Mani, M., and Goyal, S. (2023). Inferring single-cell transcriptomic dynamics with structured latent gene expression dynamics. *Cell Reports Methods*, 3(9).

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouvé, A., and Peyré, G. (2019). Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690. PMLR.

Fleck, J. S., Jansen, S. M. J., Wollny, D., Zenk, F., Seimiya, M., Jain, A., Okamoto, R., Santel, M., He, Z., Camp, J. G., et al. (2022). Inferring and perturbing cell fate regulomes in human brain organoids. *Nature*, pages 1–8.

Grønbech, C. H., Vording, M. F., Timshel, P. N., Sønderby, C. K., Pers, T. H., and Winther, O. (2020). scvae: variational auto-encoders for single-cell gene expression data. *Bioinformatics*, 36(16):4415–4422.

Han, X., Zhou, Z., Fei, L., Sun, H., Wang, R., Chen, Y., Chen, H., Wang, J., Tang, H., Ge, W., et al. (2020). Construction of a human cell landscape at single-cell level. *Nature*, 581(7808):303–309.

Hao, Y., Hao, S., Andersen-Nissen, E., Mauck, W. M., Zheng, S., Butler, A., Lee, M. J., Wilk, A. J., Darby, C., Zager, M., et al. (2021). Integrated analysis of multimodal single-cell data. *Cell*, 184(13):3573–3587.

Herman, J. S., Sagar, n., and Gruen, D. (2018). Fateid infers cell fate bias in multipotent progenitors from single-cell rna-seq data. *Nature methods*, 15(5):379–386.

Heumos, L., Schaar, A. C., Lance, C., Litinetskaya, A., Drost, F., Zappia, L., Lücken, M. D., Strobl, D. C., Henao, J., Curion, F., et al. (2023). Best practices for single-cell analysis across modalities. *Nature Reviews Genetics*, pages 1–23.

Huguet, G., Magruder, D. S., Tong, A., Fasina, O., Kuchroo, M., Wolf, G., and Krishnaswamy, S. (2022). Manifold interpolating optimal-transport flows for trajectory inference. *Advances in Neural Information Processing Systems*, 35:29705–29718.

Hwang, B., Lee, J. H., and Bang, D. (2018). Single-cell rna sequencing technologies and bioinformatics pipelines. *Experimental & molecular medicine*, 50(8):1–14.

Ipsen, M. (2004). Evolutionary reconstruction of networks. *Function and regulation of cellular systems*, pages 241–249.

Kidger, P. (2022). On neural differential equations. *arXiv preprint arXiv:2202.02435*.

Kingma, D. P., Welling, M., et al. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Kobak, D. and Berens, P. (2019). The art of using t-sne for single-cell transcriptomics. *Nature communications*, 10(1):5416.

La Manno, G., Soldatov, R., Zeisel, A., Braun, E., Hochgerner, H., Petukhov, V., Lidschreiber, K., Kastriti, M. E., Lönnerberg, P., Furlan, A., et al. (2018). Rna velocity of single cells. *Nature*, 560(7719):494–498.

Li, Q. (2023). sctour: a deep learning architecture for robust inference and accurate prediction of cellular dynamics. *Genome Biology*, 24(1):1–33.

Li, W. V. and Li, J. J. (2019). A statistical simulator scdesign for rational scrna-seq experimental design. *Bioinformatics*, 35(14):i41–i50.

Lin, E., Mukherjee, S., and Kannan, S. (2020). A deep adversarial variational autoencoder model for dimensionality reduction in single-cell rna sequencing analysis. *BMC bioinformatics*, 21(1):1–11.

Liu, B., Hu, X., Feng, K., Gao, R., Xue, Z., Zhang, S., Zhang, Y., Corse, E., Hu, Y., Han, W., et al. (2022). Temporal single-cell tracing reveals clonal revival and expansion of precursor exhausted t cells during anti-pd-1 therapy in lung cancer. *Nature Cancer*, 3(1):108–121.

Luecken, M. D. and Theis, F. J. (2019). Current best practices in single-cell rna-seq analysis: a tutorial. *Molecular systems biology*, 15(6):e8746.

Marouf, M., Machart, P., Bansal, V., Kilian, C., Magruder, D. S., Krebs, C. F., and Bonn, S. (2020). Realistic in silico generation and augmentation of single-cell rna-seq data using generative adversarial networks. *Nature communications*, 11(1):166.

Matsumoto, H., Kiryu, H., Furusawa, C., Ko, M. S., Ko, S. B., Gouda, N., Hayashi, T., and Nikaido, I. (2017). Scode: an efficient regulatory network inference algorithm from single-cell rna-seq during differentiation. *Bioinformatics*, 33(15):2314–2321.

McCabe, S., Torres, L., LaRock, T., Haque, S. A., Yang, C.-H., Hartle, H., and Klein, B. (2020). netrd: A library for network reconstruction and graph distances. *arXiv preprint arXiv:2010.16019*.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Moris, N., Pina, C., and Arias, A. M. (2016). Transition states and cell fate decisions in epigenetic landscapes. *Nature Reviews Genetics*, 17(11):693–703.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Qiu, X., Ding, S., and Shi, T. (2012). From understanding the development landscape of the canonical fate-switch pair to constructing a dynamic landscape for two-step neural differentiation. *PloS one*, 7(12):e49271.

Qiu, X., Mao, Q., Tang, Y., Wang, L., Chawla, R., Pliner, H. A., and Trapnell, C. (2017). Reversed graph embedding resolves complex single-cell trajectories. *Nature methods*, 14(10):979–982.

Qiu, X., Zhang, Y., Martin-Rufino, J. D., Weng, C., Hosseinzadeh, S., Yang, D., Pogson, A. N., Hein, M. Y., Min, K. H. J., Wang, L., et al. (2022). Mapping transcriptomic vector fields of single cells. *Cell*, 185(4):690–711.

Risso, D., Perraudeau, F., Gribkova, S., Dudoit, S., and Vert, J.-P. (2018). A general and flexible method for signal extraction from single-cell rna-seq data. *Nature communications*, 9(1):284.

Saelens, W., Cannoodt, R., Todorov, H., and Saeys, Y. (2019). A comparison of single-cell trajectory inference methods. *Nature biotechnology*, 37(5):547–554.

Schiebinger, G., Shu, J., Tabaka, M., Cleary, B., Subramanian, V., Solomon, A., Gould, J., Liu, S., Lin, S., Berube, P., et al. (2019). Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943.

Sha, Y., Qiu, Y., Zhou, P., and Nie, Q. (2024). Reconstructing growth and dynamic trajectories from single-cell transcriptomics data. *Nature Machine Intelligence*, 6(1):25–39.

Sun, T., Song, D., Li, W. V., and Li, J. J. (2021). scdesign2: a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured. *Genome biology*, 22(1):163.

Svensson, V., Gayoso, A., Yosef, N., and Pachter, L. (2020). Interpretable factor models of single-cell rna-seq via variational autoencoders. *Bioinformatics*, 36(11):3418–3421.

Tong, A., Huang, J., Wolf, G., Van Dijk, D., and Krishnaswamy, S. (2020). Trajectorynet: A dynamic optimal transport network for modeling cellular dynamics. In *International conference on machine learning*, pages 9526–9536. PMLR.

Townes, F. W., Hicks, S. C., Aryee, M. J., and Irizarry, R. A. (2019). Feature selection and dimension reduction for single-cell rna-seq based on a multinomial model. *Genome biology*, 20:1–16.

Tran, D., Nguyen, H., Tran, B., La Vecchia, C., Luu, H. N., and Nguyen, T. (2021). Fast and precise single-cell data analysis using a hierarchical autoencoder. *Nature communications*, 12(1):1029.

Trapnell, C., Cacchiarelli, D., Grimsby, J., Pokharel, P., Li, S., Morse, M., Lennon, N. J., Livak, K. J., Mikkelsen, T. S., and Rinn, J. L. (2014). The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature biotechnology*, 32(4):381–386.

Vriz, S., Joly, C., Boulekbache, H., and Condamine, H. (1996). Zygotic expression of the zebrafish sox-19, an hmg box-containing gene, suggests an involvement in central nervous system development. *Molecular brain research*, 40(2):221–228.

Wang, D. and Gu, J. (2018). Vasc: dimension reduction and visualization of single-cell rna-seq data by deep variational autoencoder. *Genomics, proteomics & bioinformatics*, 16(5):320–331.

Wang, P., Song, C., Zhang, H., Wu, Z., Tian, X.-J., and Xing, J. (2014). Epigenetic state network approach for describing cell phenotypic transitions. *Interface focus*, 4(3):20130068.

Warga, R. M., Mueller, R. L., Ho, R. K., and Kane, D. A. (2013). Zebrafish tbx16 regulates intermediate mesoderm cell fate by attenuating fgf activity. *Developmental biology*, 383(1):75–89.

Woicik, A., Zhang, M., Chan, J., Ma, J., and Wang, S. (2023). Extrapolating heterogeneous time-series gene expression data using sagittarius. *Nature Machine Intelligence*, pages 1–15.

Wolf, F. A., Angerer, P., and Theis, F. J. (2018). Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19:1–5.

Wolf, F. A., Hamey, F. K., Plass, M., Solana, J., Dahlin, J. S., Göttgens, B., Rajewsky, N., Simon, L., and Theis, F. J. (2019). Paga: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *Genome biology*, 20:1–9.

Xiang, R., Wang, W., Yang, L., Wang, S., Xu, C., and Chen, X. (2021). A comparison for dimensionality reduction methods of single-cell rna-seq data. *Frontiers in genetics*, 12:646936.

Yeo, G. H. T., Saksena, S. D., and Gifford, D. K. (2021). Generative modeling of single-cell time series with prescient enables prediction of cell trajectories with interventions. *Nature communications*, 12(1):3222.

Zappia, L., Phipson, B., and Oshlack, A. (2017). Splatter: simulation of single-cell rna sequencing data. *Genome biology*, 18(1):174.

S1 Related Works

One category of methods learns gene expression dynamics by inferring cell trajectories (Saelens et al., 2019; Herman et al., 2018; Wolf et al., 2019; Trapnell et al., 2014; Qiu et al., 2017). However, most of them are based on pseudo-time inference, which estimates differential time for cells based on cell pair distances in low-dimensional spaces. Therefore, they cannot model developmental dynamics with respect to real physical time.

In addition, a category of approaches uses RNA velocity (La Manno et al., 2018; Bergen et al., 2021, 2020) to uncover developmental trends and underlying kinetics of gene expression. But they have inevitable limitations on gene expression prediction because they use linear ODEs (Matsumoto et al., 2017) that fail to capture complex gene expression dynamics, deeply sequenced datasets to obtain spliced and unspliced counts as prior information (Farrell et al., 2023), or have to use RNA velocity pre-computed from gene expression (Chen et al., 2022b) that causes double dipping (Ball et al., 2020) problems. Furthermore, most RNA velocity models are non-generative and infeasible for predicting gene expression.

Single-cell gene expression prediction has been investigated in previous studies (Cao et al., 2021). For example, Marouf et al. (2020) proposes the use of generative adversarial networks (GAN) to generate realistic augmentations for specific cell types. Moreover, Zappia et al. (2017); Risso et al. (2018); Li and Li (2019); Sun et al. (2021) use statistical models to fit distributions of observed single-cell RNA sequencing (scRNA-seq) data and use the learned information to generate simulations. These models can make accurate predictions that reflect the properties of the experimental data. However, they are designed for gene expression measured at a single time snapshot and do not consider developmental dynamics. Therefore, they are infeasible for predicting temporal scRNA-seq data.

Recently, a few generative methods have introduced procedures to understand cell differentiation based on physical time and predict unobserved timepoints. Sagittarius method (Woicik et al., 2023) predicts gene expressions at future timepoints but mainly focuses on bulk RNA-seq data that requires cell matching between timepoints. This matching is hard to obtain for real-world scRNA-seq as the cells are lysed during the experiment. Although Live-seq (Chen et al., 2022a) enables temporal transcriptomic recording on the same cells, most existing single-cell datasets still have the cell lysing problem. In another line of works, Waddington-OT (WOT) (Schiebinger et al., 2019) interpolates between two timepoints by using an optimal transport (OT) framework to predict cell-cell couplings. However, it cannot extrapolate to future timepoints as the cell-cell coupling is unavailable when the subsequent timepoint is missing. TrajectoryNet (Tong et al., 2020) utilizes continuous normalizing flow to model the paths between cell populations of two time snapshots and predict interpolations. Therefore, WOT and TrajectoryNet are unable to extrapolate beyond the observed timepoints because they require gene expression measurements at two endpoints to learn intermediate cellular developments. In addition, Sha et al. (2024) jointly models developmental dynamics with cell growth. However, it focuses on reconstructing cellular trajectories and inferring underlying gene regulatory networks. It does not show abilities to predict extrapolation to timepoints beyond the measured time range.

PRESCIENT (Yeo et al., 2021) applies principal component analysis (PCA) to reduce high-dimensional gene space to a low-dimensional representation. It then applies neural ordinary differential equations (ODEs) (Chen et al., 2018) in this low-dimensional space to model the cell developmental trajectory. Multiple studies (Ding et al., 2018; Xiang et al., 2021; Tran et al., 2021) have found that PCA-based low-dimensional representations of scRNA-seq expression cannot capture complex relationships in the highly heterogeneous single-cell data. Thus, PCA may have the issue of overcrowding representation (Tran et al., 2021; Kobak and Berens, 2019), where cells of different types are poorly separated in PCA-based low-dimensional space, and cellular variations are lost. MIOFlow (Huguet et al., 2022) replaces PCA with a geodesic variational autoencoder (VAE) to learn non-linear low-dimensional representations that better retain cellular variations. However, this method fixes its low-dimensional space at the beginning, followed by neural ODE modeling. We hypothesize that a fixed low-dimensional representation obtained using observed timepoints may not generalize to an unobserved timepoint, if its underlying distribution differs substantially. Therefore, these methods either use less effective dimensionality reduction approaches or freeze low-dimensional representations so that they have inaccurate predictions at unmeasured timepoints.

On the contrary, our `scNODE` uses variational autoencoder (VAE), a non-linear dimensionality reduction method that captures complex cell relationships. `scNODE` also updates the low-dimensional latent space when learning developmental dynamics, such that the latent space considers overall developments and is more robust against distribution shifts.

S2 Single-Cell Dataset and Pre-Processing

We use three publicly available single-cell RNA sequencing (scRNA-seq) datasets. The ZB dataset is downloaded from Single Cell Portal with identifier SCP126. The DR data is downloaded from `https://shendure-web.gs.washington.edu/content/members/DEAP_website/public/` and SC data are available at

https://broadinstitute.github.io/wot/tutorial/. For all datasets, we use data after removing batch effects among different timepoints.

For each task, we first select highly variable genes (HVGs). HVGs are a subset of genes that contribute strongly to cell-to-cell variation within a cell population. Using only HVGs for analysis helps remove data noise and reduce computational costs. In each case, we detect the top 2000 HVGs only from training timepoints to avoid data leakage. We use *Scanpy* (Wolf et al., 2018) and *Serurat* (Hao et al., 2021) to detect HVGs.

We then normalize expression to remove cell-specific bias before conducting experiments. Specifically, given the unique molecular identifier (UMI) count expression of cell i as $\mathbf{X}_i \in \mathbb{Z}^p_{\geq 0}$ where $\mathbb{Z}_{\geq 0} = \{0, 1, 2, \cdots\}$, we normalize it by total counts over all genes

$$\mathbf{X}_i = \frac{\mathbf{X}_i}{\sum_{j=1}^{p} \mathbf{X}_i} * 10^4, \tag{S8}$$

followed by the log-transformation

$$\mathbf{X}_{ij} = \log(\mathbf{X}_{ij} + 1). \tag{S9}$$

The normalization is cell-specific so that there is no data leakage between training and testing data. Because SC datasets only provide normalized expression data, we normalize the other two datasets.

Table S1: The leave-out timepoints (testing timepoints) for three defined tasks in each dataset. The timepoint index starts with 0. Since the SC dataset has more timepoints, we leave out more timepoints than other datasets in its hard task.

Dataset	**# Timepoints**	**Leave-Out Timepoints (Testing Timepoints)**		
		Easy Task	Medium Task	Hard Task
ZB	12	4, 6, 8	10, 11	2,4,6,8,10,11
DR	11	4, 6, 8	8, 9, 10	2,4,6,8,9,10
SC	19	5, 10, 15	16, 17, 18	5,7,9,11,15,16,17,18

S3 scNODE Training and Inference

Our scNODE is implemented with *Pytorch 1.13* (Paszke et al., 2019) and is trained end-to-end. scNODE training consists of two main steps. scNODE first pre-trains the VAE component (encoder Enc_ϕ and decoder Dec_θ) with all cells of training timepoints. We use Adam optimizer to pre-train scNODE with a learning rate of 0.001 and 200 iterations. Then, scNODE optimizes both VAE and neural ODE components by minimizing the $\mathcal{L}_{\text{dyn}}$. We adopt batch training and use Adam optimizer to train scNODE with a learning rate of 0.001 and 1000 iterations. At each training iteration, we randomly select 32 cells at $t = 0$ as a batch and predict for every training timepoint $t \in \mathcal{T}$. Because the Wasserstein distance computation is expensive, batch training improves training efficiency and enables scNODE usage on large-scale datasets. We use *geomloss* (Feydy et al., 2019) to compute Wasserstein distance with blur = 0.05 and scaling = 0.5. Pseudo-codes of scNODE are provided in Algorithm S1.

Specifically, scNODE training loss function (Eq. 7 in main manuscript) contains a dynamic regularization term to pose cellular dynamics to the latent space and combines structure relation and temporal development of cells. This leads to an informative, interpretable, and robust latent space. At each training timepoint $t \in \mathcal{T}$, we use the encoder to compute latent variables $\mathbf{Z}^{(t)}_{\text{enc}}$ for $\mathbf{X}^{(t)}$ and then compute the Wasserstein distance between it and latent variables $\mathbf{Z}^{(t)}$ generated from neural ODE. Notice at the first timepoint $t = 0$, its latent variable $\mathbf{Z}^{(0)}$ is computed from VAE encoding, such that the regularization $\text{Wass}\left(\mathbf{Z}^{(0)}_{\text{enc}},\ \mathbf{Z}^{(0)}\right) \approx 0$. But for the latter training timepoints, this regularization enforces similarities between structure latent representations from VAE and the dynamic manifold from neural ODE.

At inference, scNODE first uses the trained encoder Enc_ϕ to map gene expression at the first timepoint to the latent space and then samples latent variables for the required number of cells from the latent Gaussian distribution. Then, scNODE uses the ODE solver to predict latent variables of these cells at required timepoints and map them back to the gene space through the decoder Dec_θ. Our scNODE can predict gene expression at any timepoints, including those observed and unobserved ones.

S4 Baseline Models

We compare scNODE with the following baseline models.

Algorithm S1 scNODE

Input: The set of measured timepoint indices $\mathcal{T}$; expression matrices $\{\mathbf{X}^{(t)} \mid t \in \mathcal{T}\}$; hyperparameters Δt, λ, and β; randomly initialized neural networks Enc_ϕ, Dec_θ, and Drift_ω.
(Step I: Pre-Training)
$\mathbf{X}_{\text{ALL}} = \text{CONCAT}\left(\mathbf{X}^{(t)} \mid t \in \mathcal{T}\right)$ // concatenate cells from training timepoints
Optimize ϕ and θ to minimize $\mathcal{L}_{\text{pre}}$ (Eq. 2)
(Step II: Dynamic Modelling)
Optimize ϕ, θ, and ω to minimize $\mathcal{L}_{\text{dyn}}$ (Eq. 7)
Output: ϕ, θ, and ω

- **PRESCIENT**: Yeo et al. (2021) proposes a generative model, called Potential eneRgy undErlying Single Cell gradIENTs (PSCIENT), to learn the differentiation landscape from single-cell time-series gene expression data. PRESCIENT maps gene expression to the PCA space and models cellular differentiation with neural ordinary differential equation (ODE). PRESCIENT suggests using cell proliferation information to improve prediction. But cell proliferation is not available in general cases. Therefore, we do not use such information in our experiments for all methods. We use Python codes on GitHub (`https://github.com/gifford-lab/prescient`) to run PRESCIENT.

- **MIOFlow**: Huguet et al. (2022) integrate geodesic VAE and neural ODE to model the paths of cells in a latent space while cellular structures are preserved. Specifically, MIOFlow first learns low-dimensional cell representation with a geodesic VAE such that latent space keeps the geometry of data manifold. MIOFlow then models cellular differentiation with neural ODE in this latent space. We use Python codes on GitHub (`https://github.com/KrishnaswamyLab/MIOFlow`) to run MIOFlow.

Waddintong-OT (Schiebinger et al., 2019) and TrajectoryNet (Tong et al., 2020) are the other two generative methods. However, they can only predict interpolations between timepoints. Moreover, Yeo et al. (2021) and Huguet et al. (2022) have shown that PRESCIENT and MIOFlow perform better than these two methods. In this work, we focus on generative models that can both interpolate and extrapolate gene expressions, thus excluding Waddington-OT and TrajectoryNet in the comparison.

S5 Hyper-Parameter Tuning

For each method, we select hyperparameters that yield the minimum averaged Wasserstein distance using 3-fold cross-validation. For example, if the dataset has training timepoints $t = 0, 1, 2$ and we want to predict gene expression at testing timepoint $t = 3$. In cross-validation, we equally split the cells at every timepoint of $t = 0, 1, 2$ into three subsets, train the model on two of them, and validate it on the rest one. We repeat this process three times to ensure each subset is used as the validation set once and record the averaged Wasserstein distance on three validation sets. We use Optuna Akiba et al. (2019) to automatically determine the optimal hyperparameters and use sufficiently large hyperparameter ranges for search and evaluation. The search spaces of hyperparameters are shown in Table S2.

Specifically, we set the latent space dimension as 50 and step size $\Delta t = 0.1$ for our scNODE and all baseline methods. State-of-the-art methods (Townes et al., 2019; Tran et al., 2021; Heumos et al., 2023) also generally choose a latent space of tens of dimensions. For a fair comparison, we set the same latent space size for all methods. Moreover, step size $\Delta t = 0.1$ implies the neural ODE interpolates 10 unformaly spaced timepoints between two consecutive training timepoints. A small step size increases dynamic resolution and a larger step size reduces computational costs. Previous works (Yeo et al., 2021; Huguet et al., 2022) also use $\Delta t = 0.1$ as a proper tradeoff between efficiency and accuracy. We train each method for sufficient iterations to ensure they converge. Specifically, we train scNODE 200 pre-training iterations and 1000 training iterations, PRESCIENT 500 epochs, MIOFlow 1000 pre-training iterations and 100 training epochs. We let every model predict 2000 cells at each predicted timepoints to ensure a fair metric comparison. The datasets used in our experiments have thousands of cells at each timepoint, so we believe 2000 is a reasonable number of predicted cells.

Table S2: Hyperparameter search space of scNODE and baseline methods. "None" means no hidden layers.

Model	Hyperparameters Search Space
scNODE	$d = 50$, $\lambda = 0$ regularize coefficient $\beta \in [0.0, 10.0]$ encoder network size $\in \{\text{None}, [d], [d, d]\}$ decoder network size $\in \{\text{None}, [d], [d, d]\}$ drift network size $\in \{\text{None}, [d], [d, d]\}$
PRESCIENT	$d = 50$ # hidden layers $\in \{1, 2, 3\}$ sd $\in [0.0, 1.0]$ tau $\in [0.0, 0.1]$ gradient clip $\in [0.0, 1, 0]$
MIOFlow	$d = 50$ gae embedded dim $\in \{10, 50, 100\}$ part encoder layers $\in \{[2000, 100], [2000, 100, 100]\}$ layers $\in \{[16], [16, 16], [16, 32, 16]\}$ $\lambda \in [1.0, 100.0]$

Table S3: Wasserstein distance of predictions in the easy (interpolation only) and medium (extrapolation only) tasks for all three real-world datasets. We report average(std.) Wasserstein distance over five trials. Bold numbers denote the best prediction, and underlined numbers represent the second best.

Dataset	ZB				
	Easy (interpolation only)			Medium (extrapolation only)	
	$t = 4$	$t = 6$	$t = 8$	$t = 10$	$t = 11$
scNODE	**433.06(7.1)**	**393.08(3.9)**	**474.37(3.5)**	**608.79(5.1)**	**684.15(11.3)**
MIOFlow	458.60(13.6)	406.51(6.2)	487.62(5.5)	620.52(8.8)	712.38(25.8)
PRESCIENT	840.30(0.0)	693.66(0.0)	705.99(0.0)	989.87(0.0)	1045.14(0.0)

Dataset	DR					
	Easy (interpolation only)			Medium (extrapolation only)		
	$t = 4$	$t = 6$	$t = 8$	$t = 8$	$t = 9$	$t = 10$
scNODE	348.18(1.4)	**404.47(1.6)**	**490.55(2.2)**	**546.81(9.0)**	575.92(24.6)	717.09(27.9)
MIOFlow	**346.71(1.0)**	406.16(2.2)	497.42(2.5)	553.26(2.5)	594.77(19.2)	775.86(38.0)
PRESCIENT	449.68(0.0)	476.85(0.0)	519.6(0.0)	600.77(0.0)	**553.21(0.0)**	**709.49(0.0)**

Dataset	SC					
	Easy (interpolation only)			Medium (extrapolation only)		
	$t = 5$	$t = 10$	$t = 15$	$t = 16$	$t = 17$	$t = 18$
scNODE	58.97(1.7)	**133.91(3.7)**	**118.50(3.2)**	**130.26(6.5)**	**123.80(1.9)**	**128.08(2.5)**
MIOFlow	**56.49(1.8)**	136.15(1.85)	122.56(2.3)	154.48(7.0)	154.52(5.3)	172.74(8.2)
PRESCIENT	104.99(0.0)	138.45(0.0)	154.50(0.0)	177.17(0.0)	163.12(0.0)	168.07(0.0)

S6 More on experiments

S6.1 scNODE can accurately predict expression at unmeasured timepoints

We compare scNODE 's performance with baseline methods for predicting gene expression at left-out timepoints. Tables S3 and S4 show the Wasserstein distance of left-out testing timepoints for defined easy, medium, and hard tasks. We report Wasserstein distances averaged on five repeating trials. scNODE clearly outperforms the baselines in most cases. In other cases where scNODE has the second best predictions, scNODE has a similar performance as MIOFlow and significantly performs better than PRESCIENT. Moreover, in medium and hard tasks, where extrapolations are required, scNODE can have significant superiority over baselines. For example, at $t = 15$ of SC hard task (in Table S4), Wasserstein distance of scNODE predictions is around 132, while that of PRESCIENT and MIOFlow are about 162 and 150 respectively. Therefore, scNODE has better overall performance, especially in more challenging tasks.

Table S4: Wasserstein distance of predictions in hard tasks (inter- and extrapolation). We report average(std.) Wasserstein distance over five trials. **Bold** numbers denote the best prediction, and underlined numbers represent the second best.

Method	ZB / Hard Task					
	Interpolation				Extrapolation	
	$t = 2$	$t = 4$	$t = 6$	$t = 8$	$t = 10$	$t = 11$
scNODE	**579.10(2.2)**	**508.55(3.6)**	**440.92(4.7)**	**517.81(2.2)**	**652.36(3.1)**	**707.10(4.9)**
MIOFlow	580.18(7.5)	516.59(7.9)	453.61(3.4)	536.35(2.3)	671.23(15.9)	734.42(21.7)
PRESCIENT	1381.96(0.0)	1002.62(0.0)	730.974(0.0)	701.29(0.0)	916.51(0.0)	973.17(0.0)

Method	DR / Hard Task					
	Interpolation			Extrapolation		
	$t = 2$	$t = 4$	$t = 6$	$t = 8$	$t = 9$	$t = 10$
scNODE	445.82(4.7)	**464.78(2.6)**	535.78(3.8)	**600.18(7.0)**	585.60(6.8)	**718.20(3.2)**
MIOFlow	**443.56(6.5)**	469.51(1.7)	**532.93(2.5)**	617.48(9.8)	680.41(18.6)	852.02(37.2)
PRESCIENT	524.38(0.0)	511.61(0.0)	539.38(0.0)	621.31(0.0)	**575.45(0.0)**	718.56(0.0)

Method	SC / Hard Task							
	Interpolation				Extrapolation			
	$t = 5$	$t = 7$	$t = 9$	$t = 11$	$t = 15$	$t = 16$	$t = 17$	$t = 18$
scNODE	55.22(0.9)	**59.89(1.0)**	**103.26(2.0)**	**140.81(2.4)**	**132.86(7.3)**	**148.89(10.8)**	**137.90(10.6)**	**151.13(12.7)**
MIOFlow	**55.07(2.0)**	61.80(1.6)	108.72(2.1)	156.51(6.9)	162.12(19.1)	191.40(30.2)	189.39(35.3)	215.74(49.9)
PRESCIENT	85.36(0.0)	87.47(0.0)	114.16(0.0)	142.03(0.0)	150.53(0.0)	161.59(0.0)	147.23(0.0)	155.06(0.0)

We also visualize ground truth and model predictions in the 2D UMAP space to qualitatively compare scNODE with baseline models. Specifically, in each case, we first map all timepoints of ground truth to the 50-dimensional PCA space and then compute the 2D UMAP embedding of them. Then we project each model's predictions to this UMAP space with the PCA and UMAP transformation initialized from ground truth. We use the *umap* package (`https://github.com/lmcinnes/umap`) and set n_neighbors=50 and min_dist=0.1 when computing UMAP embeddings. Fig. S1, S2, and S3 show that our scNODE predictions better align with ground truth.

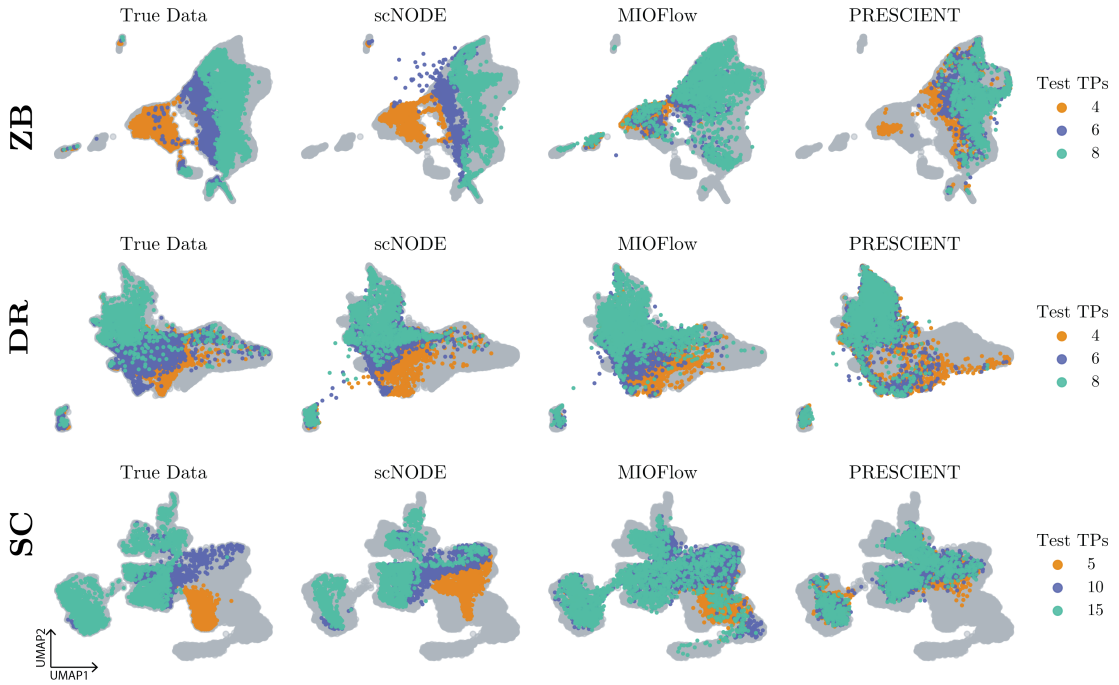

Figure S1: UMAP visualizations of true and model predictions in easy tasks. Gray points represent training data.

Unsurprisingly, extrapolating into the future is more difficult than interpolating because the training data may not contain sufficient information about future developments. Moreover, the distribution of gene expression at future timepoints may shift drastically from training data, so extrapolation is challenging for model generalization. Therefore,

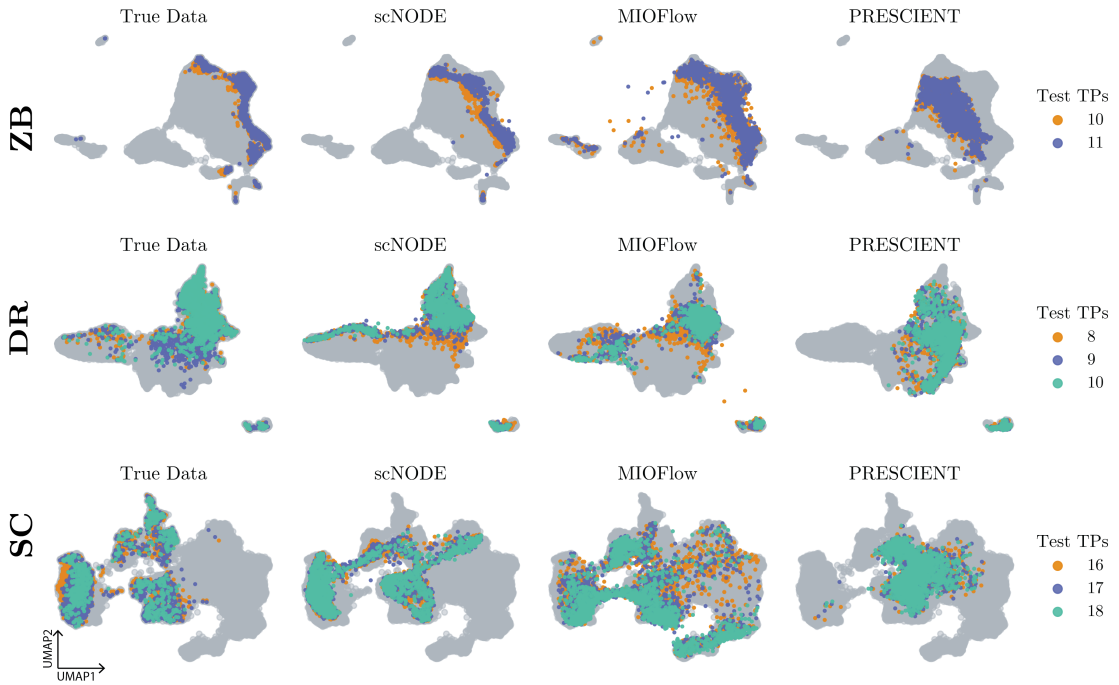

Figure S2: UMAP visualizations of true and model predictions in medium tasks. Gray points represent training data.

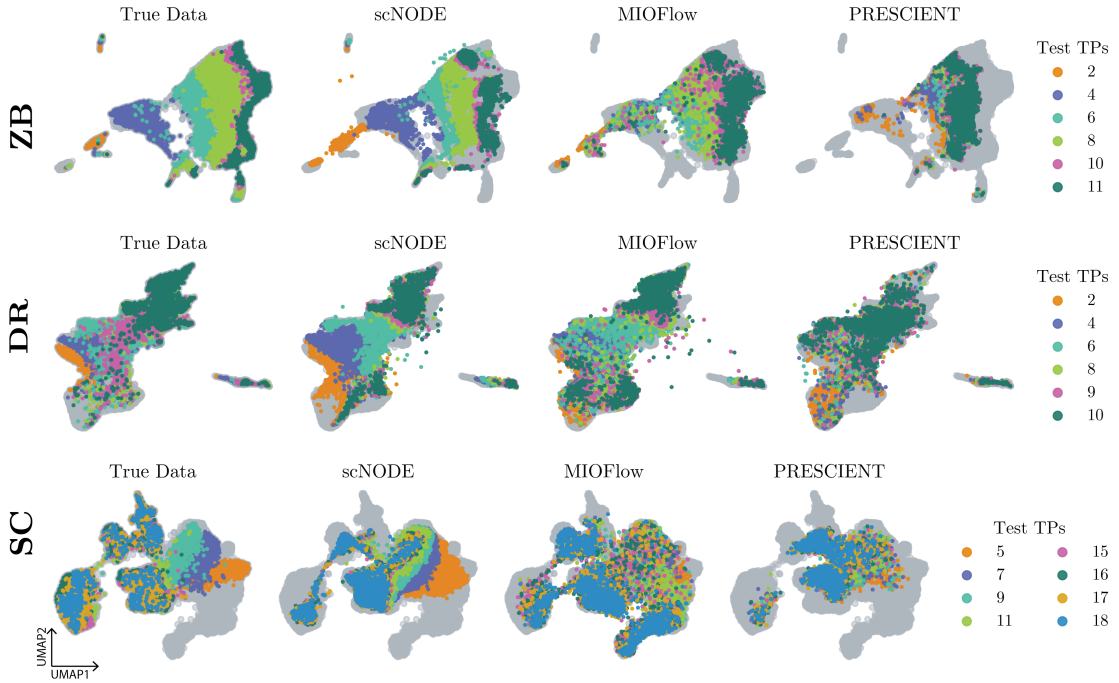

Figure S3: UMAP visualizations of true and model predictions in hard tasks. Gray points represent training data.

to evaluate model predictions in extrapolating multiple timepoints and test the generalization and stability of generative models against distribution shift, we let each model predict different future timepoints and track its performance. Specifically, in the ZB, DR, and SC datasets, we remove the last $\{1, 2 \cdots, 5\}$ timepoints, train each model on the rest of the timepoints, and let them predict extrapolations. In each case, we re-do the preprocessing and re-tune all hyperparameters. Fig. S4 shows the averaged Wasserstein distance in each setting. We notice for all methods, their extrapolations become less accurate the farther out predictions are made. But scNODE consistently performs better in most cases. Especially when extrapolating five timepoints, scNODE significantly outperforms the other two methods in DR and SC datasets, which implies that scNODE better generalizes to unobserved future timepoints that have shifted distributions. Therefore, although accurate extrapolation is a challenge (Woicik et al., 2023) in single-cell gene expression prediction, scNODE still demonstrates significant improvement over state-of-the-art methods in accurately extrapolating future timepoints.

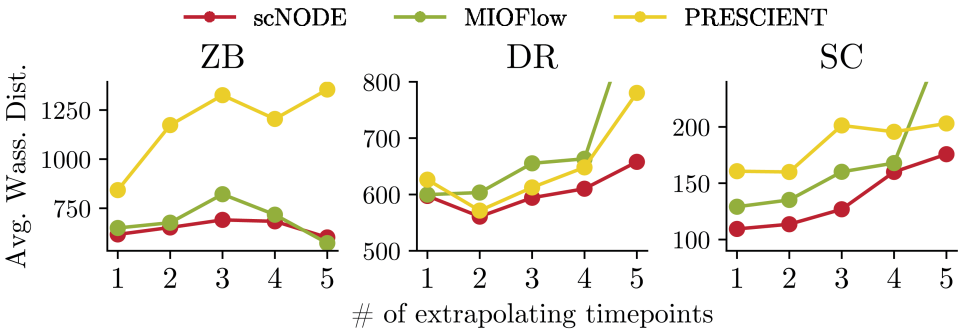

Figure S4: Compare scNODE and baseline model predictions on extrapolating multiple timepoints in ZB, DR, and SC datasets. We show the Wasserstein distance between true and predicted expression averaged across all testing timepoints.

S6.2 scNODE is more robust to distribution shift between observed and unobserved timepoints

We show that scNODE is more robust to distribution shift and obtains significant improvements over baseline models when testing timepoints have substantially different distributions as training data. In three hard tasks, we compute the distribution shift level for each testing timepoint as the averaged pairwise ℓ_2 distance between cells from training and testing timepoints. Specifically, given training data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ and testing data matrix $\mathbf{Y} \in \mathbb{R}^{m \times p}$. The distribution shift level is computed through

$$\text{distribution shift} = \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} \parallel X_i - Y_j \parallel_2 . \tag{S10}$$

Hence, a higher value indicates the testing point has more significant differences from the training data. Moreover, we define the scNODE improvement as the difference between the prediction Wasserstein distance of scNODE and the best baseline, such that a higher value indicates that scNODE obtains more improvements. We compute the Spearman's ρ correlation between distribution level and scNODE improvements for each testing timepoint. Before computing correlation, we remove outliers by fitting the data points with RANdom SAmple Consensus (RANSAC) regression, an outlier-robust regression algorithm, and remove the sample with the largest residual. Fig. 3 in the manuscript implies that scNODE improvements positively correlate with the distribution shift level. The correlations are $\rho = 0.5$, $\rho = 0.3$, and $\rho = 0.93$ for ZB, DR, and SC datasets, respectively. Therefore, scNODE is robust to distribution shift and shows more superiority when testing data are more different.

Last, we validate that scNODE learns a more robust latent space. We use the setting of leaving out the last three timepoints in all three datasets, where testing timepoints are located outside the training time range and have different distributions. We train all methods with training timepoints and compare the latent variables of testing timepoints generated by each model. Fig. S5 shows that PRESCIENT generates an over-mixing distribution of testing timepoints in the latent space, implying it fails at generating samples out of training distribution and has the worst overall performance. scNODE and MIOFlow can model temporal dynamics in the latent space and distinguish between different timepoints. However, MIOFlow latent variables are scattered and random, while scNODE has a more structured latent space, leading to accurate predictions at testing timepoints. This should be one reason for scNODE 's robustness against distribution shift.

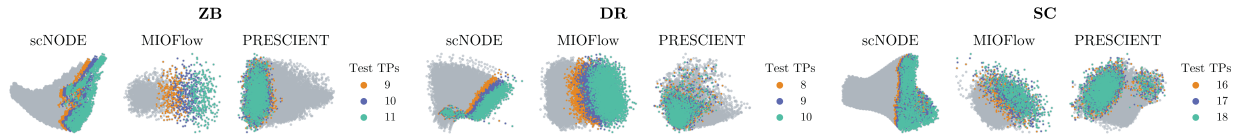

Figure S5: Compare two-dimensional PCA visualization of latent variables when extrapolating the last three timepoints in ZB, DR, and SC datasets.

S6.3 Investigation of scNODE hyperparameters

We evaluate scNODE performance using different hyperparameter settings and give heuristic guidance on how to set hyperparameters in real-world scenarios. All experiments are conducted on the hard tasks. First, we vary the latent dimension from $\{25, 50, 75, \cdots, 200\}$. Table S5 indicates that scNODE predictions are robust to the size of the latent dimensionality. Users can choose to set a reasonable latent dimension based on a tradeoff between accuracy and computational costs. State-of-the-art methods (Townes et al., 2019; Tran et al., 2021; Heumos et al., 2023) generally choose a latent space of 10 to 50 dimensions. We set the latent space size to 50 for all methods in our main comparison.

Table S5: Performance of scNODE predictions on all datasets when using different latent space size (d).

Dataset/Task	Latent Size (d)									
	25	**50**	**75**	**100**	**125**	**150**	**175**	**200**	**avg.**	**std.**
ZB/easy	449.14	429.71	428.37	427.10	429.33	429.09	426.25	419.72	429.84	7.89
ZB/medium	648.85	661.06	634.73	636.39	623.58	622.91	630.47	633.52	636.44	12.02
ZB/hard	577.88	566.15	564.11	569.51	566.54	566.06	573.11	564.70	568.51	4.46
DR/easy	412.52	413.95	417.63	411.83	419.23	413.12	416.49	415.83	415.08	2.46
DR/medium	584.51	613.93	592.91	615.29	632.27	665.69	615.98	596.47	614.64	24.06
DR/hard	557.53	553.08	557.58	555.05	555.97	553.73	556.84	567.12	557.11	4.09
SC/easy	106.45	104.78	109.69	107.83	104.69	109.15	104.36	112.92	107.48	2.81
SC/medium	127.74	124.46	126.13	123.97	123.01	129.53	120.69	121.74	124.66	2.80
SC/hard	124.53	116.06	110.92	113.96	108.16	110.45	107.39	116.93	113.55	5.27

Then, we test the effects of the pre-training phase. Specifically, we compare the accuracy of scNODE predictions when excluding the pre-training step and using different numbers of training timepoints in pre-training. We compare two strategies of choosing pretraining timepoints: randomly select from training timepoints (denoted as "random") or pick the first several training timepoints (denoted as "first"). In each case, we run scNODE 5 trials. Fig. S6 shows that pre-training clearly improves predictions in all cases. Moreover, scNODE can outperform baselines even when not using all training timepoints in the pre-training step. This is because scNODE uses dynamic regularization to update the latent space. To validate the effects of latent space adjustment in scNODE , we also compare the predictions of scNODE when including and excluding the update of latent space. Excluding the adjustment of latent space means fixing the pre-trained VAE and only optimizing neural ODE parameters ω during model learning. Fig. S7 shows that latent space adjustment improves prediction accuracy. Overall, the ablation study shows that pre-training and latent space updates are necessary for accurate predictions.

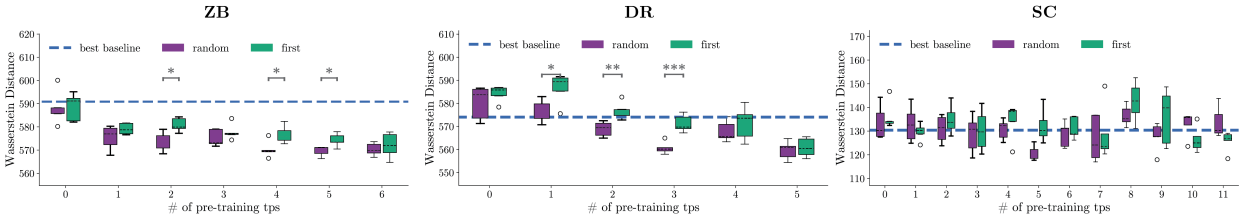

Figure S6: Compare Wasserstein distances of scNODE predictions in hard tasks when using different numbers of pre-training timepoints. Asterisks indicate p-values of the t-test between two strategies. *: p-value≤ 0.05, **: p-value≤ 0.01, and ***: p-value≤ 0.001. Dotted lines indicate the best baseline performance.

scNODE uses a dynamic regularizer with hyperparameter β to update the VAE space dynamically such that it captures both cellular variations and the developmental dynamics of the scRNA-seq data. We test scNODE predictions when using different $\beta \in \{0.25, 0.5, \cdots, 10.0\}$. Table S6 shows that the value of β affects model prediction accuracy, but scNODE is robust relative to the value of β. Moreover, scNODE performs better than baseline models in most cases. In experiments for the main text, we tune β values for scNODE in each case.

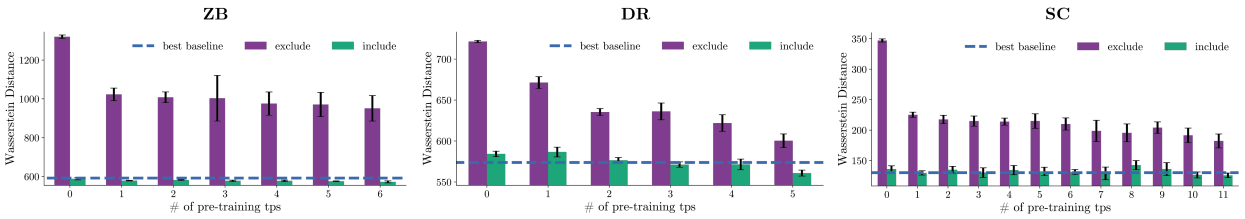

Figure S7: Compare Wasserstein distances of scNODE predictions in hard tasks when including and excluding the update of latent space. Dotted lines indicate the best baseline performance.

Table S6: Wasserstein distance of scNODE predictions when varying the regularization coefficient β. We report Wasserstein distance averaged over testing timepoints.

Dataset/Task	**Regularization Coefficient (β)**								**Best Baseline**
	0.25	**0.5**	**0.75**	**1.0**	**2.5**	**5.0**	**7.5**	**10.0**	
ZB/easy	428.54	432.30	432.64	430.98	431.24	433.35	436.32	437.24	450.91
ZB/medium	649.85	651.64	647.88	644.91	646.56	645.86	635.19	636.29	666.45
ZB/hard	571.38	572.96	570.01	569.44	567.20	571.13	569.19	569.03	590.89
DR/easy	413.71	416.57	415.09	415.85	416.21	418.18	420.73	421.04	416.76
DR/medium	637.07	630.52	604.50	629.35	594.94	606.72	597.09	590.90	621.15
DR/hard	559.99	557.36	559.44	564.38	561.52	558.55	563.30	563.52	574.02
SC/easy	102.66	103.95	105.62	106.58	106.47	108.04	106.92	108.40	157.6
SC/medium	125.81	128.40	127.15	122.61	126.36	126.36	129.69	128.83	160.58
SC/hard	113.02	114.71	113.84	126.61	121.65	117.77	118.91	120.66	130.42

S6.4 scNODE predictions help recover cell trajectories

We validate that scNODE can aid cell trajectory inference using the hard task of all three datasets. Specifically, we leave out several timepoints (see hard tasks in Table S1) and let scNODE and baseline model predict them back, in order to see whether model predictions can help with reconstructing cell developmental trajectories. We apply partition-based graph abstraction (PAGA) (Wolf et al., 2019) to construct cell trajectories. PAGA is a method that computes the topological structure of cell populations and represents population structures in interpretable graphs. Specifically, we use Louvain clustering (Blondel et al., 2008) to first cluster cells in the 2D UMAP space and apply PAGA to construct a structural graph of these clusters. We use *Scanpy* for Louvain clustering.

In addition, we use the Ipsen-Mikhailov (IM) distance (Ipsen, 2004) to quantitatively measure the similarity between the cell trajectory graphs constructed in each case. $\text{IM}(\mathcal{G}_1, \mathcal{G}_2)$ is a graph similarity measurement defined as the square-root difference between the Laplacian spectrum of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. It ranges from 0 to 1, where 0 indicates maximum similarity between two graph structures and 1 indicates maximum dissimilarity. We use *nedtrd* package (McCabe et al., 2020) to compute the IM distance. Fig. S8 shows that scNODE predictions help recovering cell trajectories as $\text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{scNODE}}) < \text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{removal}})$. Moreover, $\text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{scNODE}})$ being smaller than $\text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{MIOFlow}})$ and $\text{IM}(\mathcal{G}_{\text{true}}, \mathcal{G}_{\text{PRESCIENT}})$ in all cases implies that scNODE predictions for missing timepoints best help to infer cell trajectories.

S6.5 scNODE in perturbation analysis

We use ZB data to test scNODE 's ability of helping *in silico* perturbation analysis. We train scNODE with all timepoints of ZB data and construct the cellular state path in the latent space. The path is constructed with least action path (LAP), which has been used in previous works (Qiu et al., 2022, 2012; Wang et al., 2014) to construct cell fate transitions. The LAP method aims at finding the optimal path between two cell states while minimizing its action and transition time. Specifically, given starting point $\mathbf{X}_0$ and end point $\mathbf{X}_K$, LAP fins a path discretized as a sequence of K points $\mathcal{P} = \{\mathbf{X}_0, \cdots, \mathbf{X}_K\}$. For each segment constrained between $\mathbf{X}_{k-1}$ and $\mathbf{X}_k$, its tangential velocity is defined as $\mathbf{V}_k = \frac{(\mathbf{X}_k - \mathbf{X}_{k-1})}{\Delta}$ where Δ is the timestep taken by cells from $\mathbf{X}_{k-1}$. Therefore, we define the action $\mathcal{S}$ along the path $\mathcal{P}$ as

$$\mathcal{S} = \frac{1}{2}\sum_{k=1}^{K}\left(\mathbf{V}_k - \text{Drift}_{\omega}(\widetilde{\mathbf{X}}_k)\right)^2 \Delta \quad \text{with } \widetilde{\mathbf{X}}_k = \frac{\mathbf{X}_{k-1} + \mathbf{X} - k}{2}. \tag{S11}$$

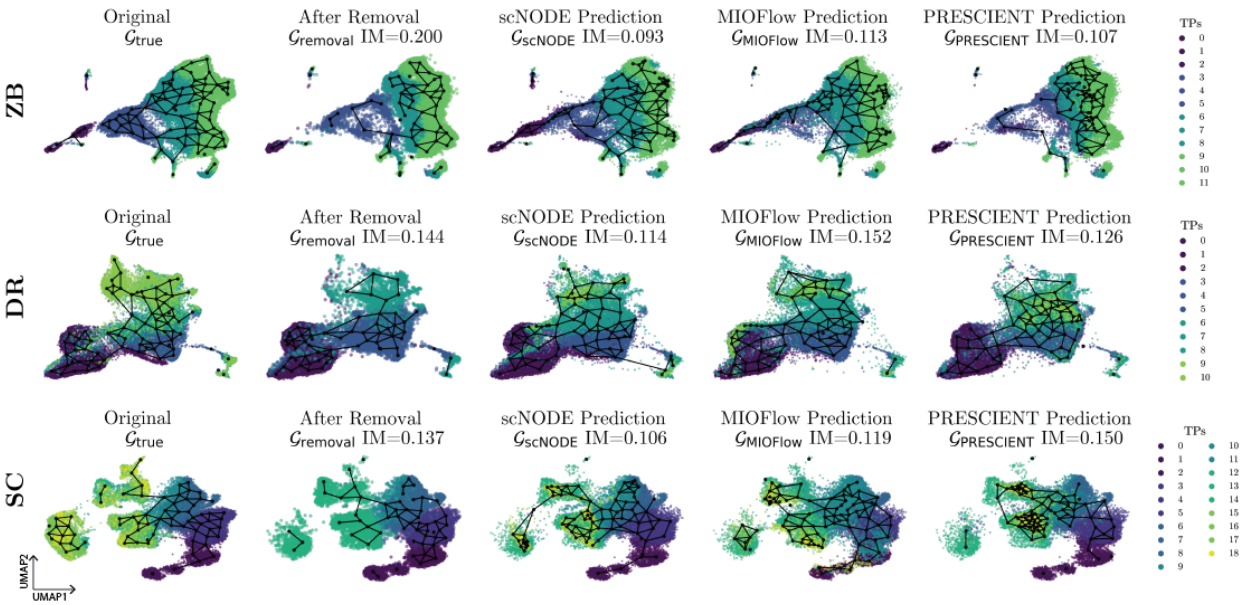

Figure S8: scNODE helps infer cell trajectories. Here are cell trajectories of data with all timepoints, after removal of timepoints in the hard task, and with model predictions. The connective structure is constructed with PAGA, where black nodes represent cell clusters and edges connect two nodes if their expressions are similar. We show the IM index between $\mathcal{G}_{\text{true}}$ and the corresponding graph.

Here, LAP method aims to align the tangential velocity $\mathbf{V}_k$ with the differential velocity $\text{Drift}_\omega(\widetilde{\mathbf{X}}_k)$ learned by scNODE , while having the least transition time. Therefore, the optimal path is

$$\widehat{\mathcal{P}} = \underset{\mathcal{P},\Delta}{\text{argmin}}\, \mathcal{S} = \underset{\mathcal{P},\Delta}{\text{argmin}} \frac{1}{2}\sum_{k=1}^{K}\left(\mathbf{V}_k - \text{Drift}_\omega(\widetilde{\mathbf{X}}_k)\right)^2 \Delta. \tag{S12}$$

Solving Eq. S12 consists of two iterative steps

(1) Minimize action by fixing path $\mathcal{P}$ and varying the timestep Δ through

$$\widehat{\Delta} = \underset{\Delta}{\text{argmin}} \frac{1}{2}\sum_{k=1}^{K}\left(\frac{\mathbf{X}_k - \mathbf{X}_{k-1}}{\Delta} - \text{Drift}_\omega(\widetilde{\mathbf{X}}_k)\right)^2 \Delta. \tag{S13}$$

(2) Minimize action by fixing timestep $\widehat{\Delta}$ and varying path $\mathcal{P}$

$$\widehat{\mathcal{P}} = \underset{\mathbf{X}_1,\cdots,\mathbf{X}_{K-1}}{\text{argmin}} \frac{1}{2}\sum_{k=1}^{K}\left(\frac{\mathbf{X}_k - \mathbf{X}_{k-1}}{\widehat{\Delta}} - \text{Drift}_\omega(\widetilde{\mathbf{X}}_k)\right)^2 \widehat{\Delta}. \tag{S14}$$

The starting ($\mathbf{X}_0$) and end point ($\mathbf{X}_K$) are fixed in the optimization.

We use *scipy.optimize.minimize* to solve these two objective functions. In our experiments, we construct two paths between the first timepoint and two cell populations (PSM and Hindbrain) with $K = 8$ (Fig. S9A). We set the starting point as the center of cells at the first timepoint (t=0) and the endpoint as the center of the cell population. We initialize timestep $\Delta = 1$ and $\mathcal{P}$ as equally spaced points from the starting to end points.

When finding the differentially expressed (DE) genes, we augment the LAP path with its nearest neighbors in the latent space (Fig. S9). Specifically, assuming $\mathcal{P} = \{\mathbf{X}_0, \cdots, \mathbf{X}_K\}$ is the LAP path from $\mathbf{X}_0$ to $\mathbf{X}_K$, we have only 8 cells on the path which is insufficient for DE detection. Therefore, for each $\mathbf{X}_k \in \mathcal{P}$, we find its nearest neighbors in the latent space in order to augment the path. We use *sklearn.neighbors.NearestNeighbors* to search for 10 nearest neighbors. The number of nearest neighbors affects DE detection, where fewer neighbors may be insufficient for good DE detection, and more neighbors may introduce noise. We select 10 neighbors for a proper tradeoff between efficiency and accuracy. Then we can use *Scanpy* to detect DE genes for the augmented path with the Wilcoxon rank-sum test.

Lastly, we perturb the expression profile of key genes in all cells at the starting timepoint by multiplying their expression values with different levels of coefficient $\{10^{-3}, \cdots, 10^3\}$ to mimic overexpressing and knocking-out,

and let scNODE predict trajectories for the perturbed gene expression. The perturbations are expected to result in changes in cell fates. Specifically, we overexpress TBX16 (DE gene of PSM) or SOX3 (DE gene of Hindbrain). We classify predicted cells with a Random Forest (RF) classifier trained with all unperturbed cells. We use *sklearn.ensemble.RandomForestClassifier* with default parameters as the classifier and train it with all unperturbed cell expression as well as cell type labels in original datasets. We compare cell ratios when perturbing DE gene and random non-DE gene and find overexpressing TBX16 expression results in the increment of PSM cell ratios (Fig. S9**B**) to over 70%. Moreover, the ratio of Hindbrain cells decreased when overexpress SOX3 expressions (Fig. S9**C**). These indicate that detected DE genes are related to developmental dynamics. In applications of scNODE , these perturbations can involve multiple genes, use different cell populations, or be carried out at different timepoints.

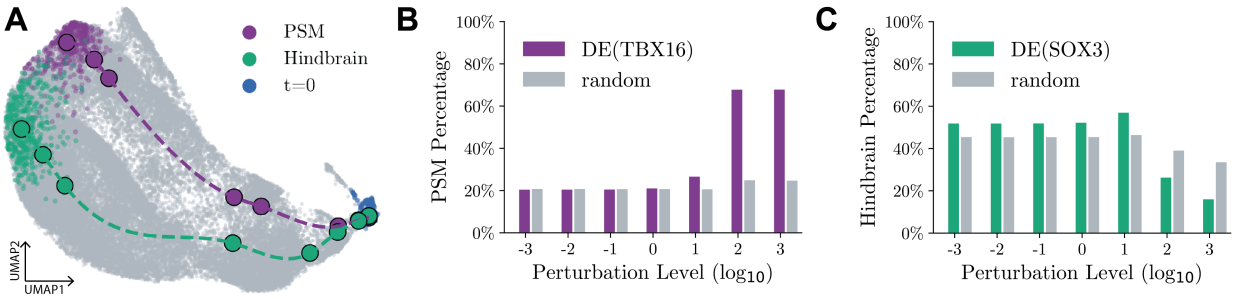

Figure S9: scNODE perturbation analysis results. (**A**) 2D UMAP visualization of the least action path between cells at the starting point ($t = 0$) and PSM/Hindbrain cell populations. (**B**) The ratio of PSM cells in predictions of different perturbation levels when perturbing TBX16 or ten random non-DE genes. (**C**) The ratio of Hindbrain cells in predictions of different perturbation levels when perturbing SOX3 or ten random non-DE genes.