Deep Generative and Predictive Modeling of Single Cell RNA-Seq Time Series Data

Justin Sanders

Advisor: Ritambhara Singh
Second Reader: Erica Larschan

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science with Honors

Center for Computational Molecular Biology

Brown University

April 2022

Abstract

Recent advances in single-cell RNA sequencing assays have opened the door to understanding biological heterogeneity at the cellular level, offering new insights into the regulatory programs governing dynamic processes such as differentiation, cellular reprogramming, and carcinogenesis. Modeling the temporal pattern of changes to gene networks along such trajectories is necessary for understanding the causal relationships governing cell fate. However, single cell expression data is inherently sparse, noisy, and high dimensional - posing a challenge to computational analysis. In this thesis, we probabilistically model gene expression trajectories across time using methods from deep learning to address these challenges. Our proposed model consists of two components - a variational autoencoder for learning a denoised, low dimensional representation of the expression data and a recurrent neural network for predicting trajectories within the learned latent space. It has previously been shown that single-cell expression profiles lie on a low dimensional manifold in high dimensional space, and here we test the hypothesis is that trajectories on this manifold are more easily inferred than in the original space. We apply our method to two single cell RNA-Seq datasets capturing mouse embryogenesis and stem cell reprogramming trajectories, and show that predictions from our model accurately recapitulate temporal dynamics of the data. Additionally, we demonstrate how predictions made by our model can assist in the discovery of rare cell types and help identify genes that are differentially expressed along trajectories.

Acknowledgements

I would like to thank my advisor for this project, Ritambhara Singh, for welcoming me into her lab and providing her mentorship. Her enthusiasm and support throughout this year have helped me grow as a researcher and encouraged me to pursue computational biology further in graduate school.

I would also like to thank Jeremy Bigness and Jiaqi Zhang, who worked on this project with me in the Singh Lab. Jeremy was super helpful and patient in getting me on-boarded and up to speed with the project, and I'm grateful for all his advice both about research and my career in general. Jiaqi was also always available to help me out when I had questions, and helped produce many of the results that made this thesis possible.

Finally, I am grateful to my previous advisors and mentors Bill Noble, Giancarlo Bonora, and Sorin Istrail for shaping and encouraging my interests in computational biology.

De appel valt niet ver van de boom.

Contents

1 Introduction

1.1 Background

One of the fundamental questions in systems biology is how our body gives rise to such a diverse collection of cell types. Despite all sharing the same underlying genetic code, heart cells, lung cells, and brain cells all express different sets of genes and perform wildly different functions. These varied cell types arise through a process called differentiation, by which a pluripotent cell undergoes precise epigenetic changes to alter gene expression, transitioning it into a more specialized cell type. Understanding the regulatory mechanisms behind this process will have direct applications in the studies of stem cell therapies[22], carcinogenesis[21], and the biology of aging [13][29]. However, the complex regulatory networks and signalling pathways that govern stem cell fate during differentiation are still poorly understood [15].

Recent advances in single-cell RNA sequencing (scRNA-Seq) technologies have allowed for the large-scale measurement of the transcriptome at a cellular resolution. This allows researchers to study differences between gene expression in individual cells as they move along differentiation trajectories. Such analysis has demonstrated that differentiation is a gradual, more continuous process than was previously appreciated [1]. By performing multiple scRNA-Seq experiments at different timepoints on a population of cells undergoing a differentiation process, researchers have produced time series datasets that give a complete picture of key developmental trajectories [28] [10] [25]. Beyond the study of differentiation, scRNA-Seq has also contributed to the discovery of new rare cell types [23] and the understanding of human disease [35] [24], as well as revealed unexpected levels of heterogeneity between individual cells [1]. However, given the immaturity of this new data type, there is still a want of efficient and statistically rigorous computational methods to draw insights from it.

In this paper we present two new methods solving a pair of related tasks in the analysis of single-cell expression data. The first task is probabilistically modeling the underlying distribution of an scRNA-Seq dataset in a way that allows for the *de novo* generation of new expression profiles. Large single-cell sequencing libraries are often prohibitively expensive to produce, so solving this task would allow researchers to up-sample smaller datasets to better discover rare cell types and improve cell clustering. Additionally, it enables our approach to the second task, which is to model the evolution of the probability distribution of gene expression in cells along a trajectory. The goal is to predict the expression of cells at a future point in differentiation given a series of scRNA-Seq measurements performed at past time points. A model that is able to accurately predict cellular trajectories in this way would help researchers better understand the regulatory landscape of differentiation, and would have potential applications to cellular reprogramming and the treatment of developmental diseases.

1.2 Single-Cell Challenges

Working with scRNA-Seq data comes with a host of computational challenges. Because there is such a small amount of biological material in each individual cell, single-cell sequencing methods produce data that is inherently noisy and sparse. In particular, scRNA-Seq data shows a much higher proportion of genes with zero expression measurements in each cell than would be expected based on bulk RNA-Seq data [16]. Correcting for this technical noise is challenging, since the full mechanisms behind the missingness are unknown. Expression measurements may be missing independently at random, or their missingness may be correlated with other seen and even unseen variables. Additionally, there is a growing debate in the field as to whether scRNA-Seq data is actually zero-inflated at all[16], with many arguing that the high proportion of zeros in the data represents a true biological signal that shouldn't be corrected [31][18]. RNA transcript abundance is usually used as a proxy for gene expression in a cell, an assumption which may break down under intermittent patterns of RNA-synthesis [26]. Thus, when developing computational methods for working with this data, researchers must carefully consider and make explicit the assumptions they are making about its distribution. In general, non-parametric methods that are compatible with our current uncertainty about the nature of the data generating distribution are desirable.

Another related challenge that is common to all analysis of high-throughput assays, including scRNA-Seq, is that of batch effects. There is extensive evidence that when multiple repetitions of the same experiment are performed at different times, in different labs, or on different sequencing platforms, the resulting data shows increased correlation between cells from the same batch compared to cells from different batches [34] [14]. This bias can hide biological signals of interest, and introduce misleading structure to the data. For example, when working with time series data, differences in gene expression between timepoints may appear artificially high due to the fact that each timepoint was sequenced in a separate batch. Thus, when integrating data from multiple experiments it is important to develop computational methods that are able to handle and correct for these systematic differences between batches.

Finally, a significant hurdle when performing single-cell analysis is dealing with the high dimensionality of the data. The increase in resolution to the cellular level has added another rapidly growing dimension to our datasets, with recent experiments profiling expression for all 20,000 genes across more than a million cells [6]. This poses a significant challenge to analysis - human intuition often breaks down in such high dimensions, and statistical and machine learning based methods quickly lose power due to the "curse of dimensionality" [36]. Additionally, computational approaches must scale well in terms of runtime and memory usage. Even just realizing a full scRNA-Seq matrix into memory can require upwards of 50GB. In order to address these problems, it is common to apply dimensionality reduction techniques to single cell data. This is feasible because the transcriptional profiles of cells occupy a relatively low dimensional manifold within the high dimensional expression space [17].

Thus if we can learn a projection from the original space onto this manifold, we can perform our computation in a much lower dimensional setting that still captures all of the relevant variability in the data. Two popular methods for this are Principal Component Analysis (PCA) and Uniform Manifold Approximation and Projection (UMAP). Both have been used extensively for generating 2-Dimensional embeddings for the visualization of single cell data [3]. They are also often applied as an initial preprocessing step to reduce the dimensionality of the data before performing downstream analyses [28] [33]. However, both methods have drawbacks. PCA is fully linear, so its embeddings are restricted to being linear combinations of the original features which may fail to properly model non-linear expression manifolds. UMAP embeddings, while non-linear, have been demonstrated to poorly capture the global structure of the data [7]. Therefore, there is still a need for more general and robust computational approaches for handling the high dimensionality of single-cell data.

1.3 Problem Description and Related Work

1.3.1 Structure Model

In this paper, we present two connected models which are useful for the analysis of single-cell expression data. The first, which we dub the structure model, aims to simultaneously learn a low dimensional representation of the data while modeling its underlying distribution in a way that can easily be sampled from. Numerous methods exist for probabilistically modeling scRNA-Seq data, based on both classical statistical models [30][27] and deep learning [12][9]. However, all four methods intrinsically assume that the data is drawn from a Zero-Inflated Negative Binomial (ZINB) distribution. scDesign2 fits a uni-variate ZINB distribution to the expression levels of each gene, and then uses a Gaussian copula to capture gene-gene correlations. ZINB-WaVE also models the paramaters of a ZINB directly, with the parameters for each gene in each cell inferred based on a regression of known and unknown covariates. For the deep learning based approaches, scVAE uses a variational autoencoder with a ZINB posterior and the DCA model uses a vanilla autoencoder with a focus primarily on denoising and removing batch effects. Since there is no constraint on the latent space during training, DCA learns a poorly structured latent representation from which it is non-trivial to draw new samples.

Here we implement our structure model as a variational autoencoder (VAE). We were motivated by the scalability of deep learning models to large amounts of data, as well as the successes of VAEs at modeling complex probability distributions in other domains such as images [19] and language [4]. The architecture of a VAE consists of two dense neural networks, one mapping the input to a low dimensional latent space (encoder), and the other mapping this latent vector back to the original input (decoder). During training, the model is optimized to learn a low dimensional representation of the input from which it can faithfully reconstruct the original. In order to force the latent space learned by our model to be well behaved and easy to sample

from, we impose an additional training objective to force the distribution of points in the latent space to be approximately standard normal (for a full exposition on the theoretical background of VAEs, see section 4.4.1). While our model is similar to scVAE, our approach doesn't make any prior assumptions about the zero-inflatedness of the data. Additionally, we aimed to make our model as general and data-driven as possible, so that it can easily be applied to new datasets without extensive fine-tuning of hyperparameters and data normalization schemes. To test our model, we demonstrate qualitatively that it learns a rich embedding that captures the structure of the data, as well as showing quantitatively that it is able to accurately reproduce the original data from these embeddings. Additionally, we show that generating new data with our structure model can help improve clustering and rare cell type identification, and that our performance on these tasks compares favorably to the similar methods listed above.

1.3.2 Dynamics Model

The second component of this project is the dynamics model, which aims to predict the future expression of cells based on a series of measurements from past time points. Currently, we are not aware of any previous work that explicitly focuses on this task. However, a number of methods exist for solving the related task of trajectory imputation, where a smooth trajectory is inferred between known start and end expression states. Waddington-OT [28] and TrajectoryNet [32] both use unbalanced optimal transport to generate smooth interpolations between the expression probability distributions at the start and end time points. Prescient [37] models differentiation as a diffusion process over a learned potential function paramaterized by a neural network. Although the authors focus on trajectory interpolation, Prescient is also capable of making forward predictions. Finally, in order to make the computation time tractable, all three of these methods use PCA to reduce the dimensionality of the data before inferring trajectories. Our hypothesis is that if you can learn a low dimensional representation of the data that more faithfully represents the true manifold of cell expression, trajectories will be easier to infer over that manifold. Thus, we decided to train our dynamics model to make predictions within the latent space of our structure model. Specifically, given cells from a series of time points, we first map those cells to their latent representations using the encoder from our VAE, then we predict a latent vector for the next time point in the series, and then finally use the decoder from our VAE to reconstruct the full predicted expression profile.

For this task of predicting future embeddings based on a sequence of past embeddings, we decided to use a Recurrent Neural Network (RNN). RNNs have successful track records in many domains with sequentially structured data such as language modeling and signal processing. Although recent advances in language modeling have moved away from RNNs in favor of models which can better capture long-range dependencies in the data, we think they are still an appropriate choice for this setting

given the relatively short sequence lengths in time series datasets. At a high level, RNNs work by keeping track of an internal 'hidden' state which contains information about the history of the sequence so far. At each time step, this hidden state is updated by applying a dense neural network to the current input. Another dense layer then takes this new hidden state as input and outputs the models prediction for the next element in the series (See 4.2 for more details). This architecture allows our model to capture the dynamic temporal nature of the data within the latent space, and hopefully will let it generalize to future unseen time points. It also builds in flexibility in handling trajectories consisting of an arbitrary number of time points. To validate the effectiveness of our structure model, we test it on a publicly available single-cell RNA-Seq time series dataset. We show it produces accurate predictions which outperform both our baseline and Prescient. Additionally, we show that predictions made by our model can help to discover differentially expressed genes.

2 Results

2.1 The structure model learns an accurate latent representation of scRNA-Seq data

We trained and evaluated our structure model on two large scale single-cell RNA-seq datasets. The first, produced by Schiebinger et al. for their method Waddington-OT, contains the differentiation trajectory of mouse embryonic fibroblasts undergoing cellular reprogramming into induced pluripotent stem cells [28]. The dataset consists of transcription profiles for 259,155 single cells sampled every half day for 18 days. For this project we use only the 17 timepoints from day 0.0 to 8.0, and 4,000 cells from each timepoint, giving a total of 68,000 cells for training (see Methods 4.5). The second dataset, produced by Cao et al. contains 100,000 cells derived from mouse embryos at five timepoints on days 9.5 to 13.5 of gestation [6]. For this data, we used 8,221 cells from each timepoint, giving a total of 41,105 cells for training.

We trained our structure model on both datasets, and evaluated both the quality of the learned low-dimensional embeddings and the accuracy of the re-constructed data. The results are shown in Figure 1. We observe that the 32 dimensional latent representation of the Waddington-OT dataset learned by our model (Figure 1B) still captures the temporal structure observed in the original data (Figure 1A). This indicates that even within the latent space there is enough temporal dependency in the data that our dynamics model can capture when predicting trajectories. Furthermore, we show that the reconstructed data closely matches the true data for both the Waddington-OT dataset (Figure 1C) and the mouse embryo dataset (Figure 1E).

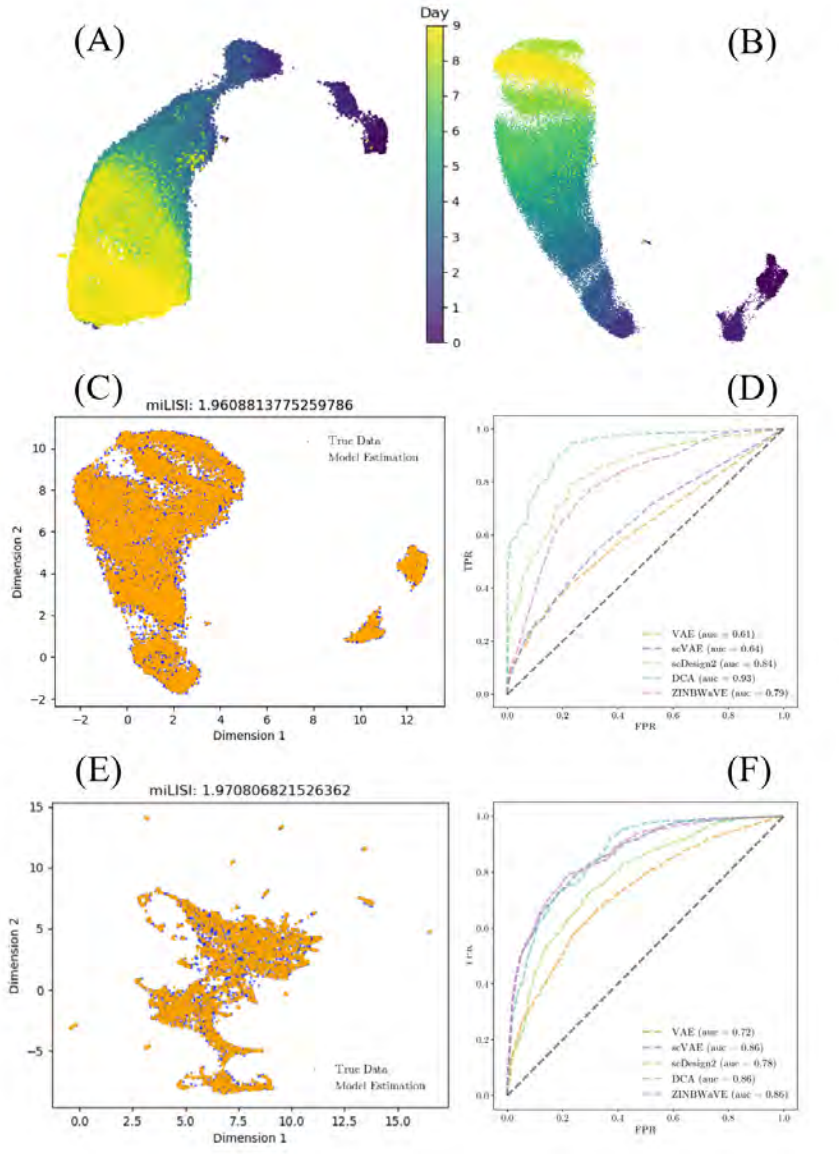


Figure 1: Results for our structure model on the Waddington-OT data-set (A-D) and the mouse embryo dataset (E - F). (A) and (B) show the 2-dimensional UMAP embedding of the original space and of our learned 32-dimensional latent space, respectively. Clearly, our latent representations capture much of the structure in the original data. (C) and (E) show UMAP plots of the true data (blue) alongside de novo *data sampled from our model (orange) for each dataset. A miLISI score, indicating how well mixed the true and simulated data are, is also given. Finally, plots (D) and (F) are AUROC curves showing how well a random forest classifier can separate true and generated cells. Results under this metric for our model (VAE) are shown alongside those of existing approaches (scVAE, scDesign2, DCA, ZINBWaVE).*

One significant challenge for this project was establishing meaningful quantitative metrics for evaluating our models predictions. The two we use here are miLISI and AUROC scores of a classifier. miLISI scores are a measure of how well mixed data with two different labels are, with 1 being perfectly separated and 2 being perfectly mixed (See 4.4 for details). For the AUROC scores, we posited that if the data generated by our model was accurately sampled from the underlying distribution, a binary classifier shouldn't be able to distinguish it from the true data. Thus, we train a simple binary classifier to distinguish the two and plot its ROC curve to measure performance (See 4.4). Under these two metrics, our structure model performs very well. The miLISI scores are 1.96 and 1.97 for the Waddington-OT and mouse embryo datasets respectively, indicating near-perfect mixing. The AUROC scores are 0.61 and 0.72 on the two datasets, meaning that our classifier is performing only somewhat better than random guessing. Additionally, we compared the performance of our structure model to the existing methods scVAE, scDesign2, DCA, and ZINBWaVE, and found our model to show the best performance (Figure 1, D and F).

	Deep Learning Models			Statistical Models	
	Structure model	**scVAE**	**DCA**	**scDesign2**	**ZINBWaVE**
cell avg	0.064	0.177	0.763	0.014	0.107
cell var	0.250	0.274	0.822	0.280	0.513
gene avg	0.065	0.026	1.000	0.106	0.040
gene var	0.363	0.261	0.988	0.106	0.440

Table 1: Results comparing our structure model to existing single cell expression simulation methods on the Waddington-OT dataset. We use the KS-test to compare the predicted distributions of average cell expression, average gene expression, cell expression variance, and gene expression variance to the true distributions.

A major limitation of the miLISI and AUROC metrics is that both are performed on low dimensional embeddings of the data - miLISI is performed on 2D plots, and the classifier for AUROC calculations is trained on the first 50 principle components. Thus, we wanted an additional metric to compare true and simulated data in the original gene expression space. Our first idea was to calculate the Pearson correlations (PCC) and Spearman Correlations (SCC) between the true and generated data - two metrics which are commonly used in regression tasks. However, while our model performed very well, obtaining near perfect correlations on the Waddington-OT (PCC = 0.999, SCC = 0.966) and mouse embryo (PCC = 0.997, SCC=.984) datasets, we felt this was a poor measure of performance. The high degree of sparsity combined with the large number of low-variance genes in scRNA-Seq data make correlation coefficients an inadequate measure of interesting biological similarity. Instead, they are more indicative of whether or not a model is simply capturing the sparsity and technical noise of the data. Instead, we opted to follow an approach used in a review paper by Crowell et al. [8] where the distributions of uni-variate summary statistics of our data are compared using the non-parametric Kolmogorov–Smirnov (KS) test. We

use these tests to measure the similarity in the distributions of mean gene expression, mean cell expression, gene expression variance, and cell expression variance between our true and generated data. Our results show that we are competitive with the existing methods scVAE, scDesign2, and ZINBWaVE and that we perform much better than DCA (Table 1). Our performance relative to the non deep learning based approaches scDesign2 and ZINBWaVE is particularly noteworthy, as those methods explicitly model these four parameters when fitting a distribution to the data.

2.2 Data augmentation with the structure model improves clustering and cell type assignment

Before presenting the results for the dynamics model, we provide an example application demonstrating that the structure model is useful on its own merits. Although the size of single-cell RNA-Seq libraries is growing at a staggering rate, large scale experiments with hundreds of thousands to millions of cells are still challenging and expensive to run. When working with smaller datasets, clustering and cell type labeling performance can suffer due to the lower number of cells. This is a particular challenge for researchers who want characterize rare cell types, which may not be clearly identifiable as a cluster without a sufficiently large dataset. Here, we show in both a simulated and real-world setting that augmenting available data with data sampled from our model can improve cell type labeling.

First, as a proof of concept we tested our structure model and the four existing baseline methods on a simulated single-cell dataset produced by Splatter [38]. Splatter is a tool for generating realistic artificial scRNA-Seq datasets by drawing cells from a ZINB distribution with gene-wise mean expression sampled from a Gaussian distribution. This is valuable for bench-marking computational methods because it allows us to obtain ground-truth cell type labels by drawing different cells from different underlying distributions. For this experiment, we used Splatter to generate a dataset of 10,000 cells drawn from three clusters. We then test whether a random forest classifiers ability to separate one cluster from the others can be improved by augmenting the data with additional cells sampled from our structure model. Figure 2B shows the result of this experiment at various sampling levels, which measures the proportion of cells coming from the target cluster. We let this range from 50% representing a common cell type to 1% representing a very rare cell type. We find that data augmentation with our structure model significantly improves the precision of cell type labels. Additionally, we outperform all the existing single-cell modeling methods at this task, including the surprisingly effective simple baseline of up-sampling the available data by including the available cells multiple times.

To show that this approach also works on real data, we tested it on our mouse embryo dataset. Since real world data no longer has ground truth cell type labels, we instead used K-nearest neighbors clustering on the full dataset to assign labels. The goal now is to see whether augmenting a downsampled version of the data using

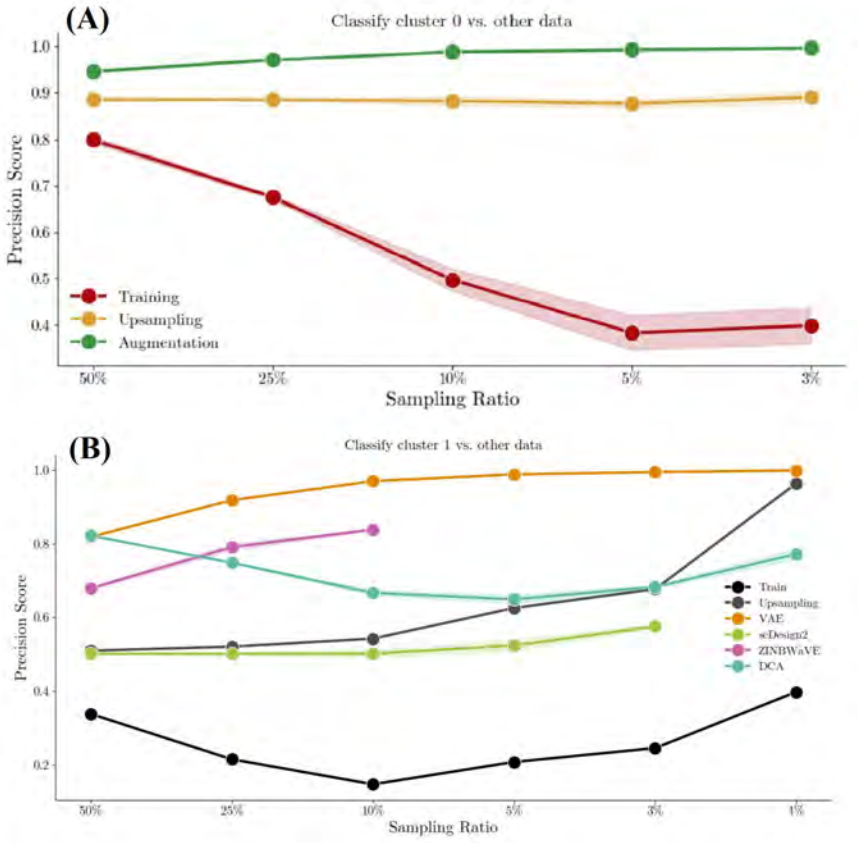
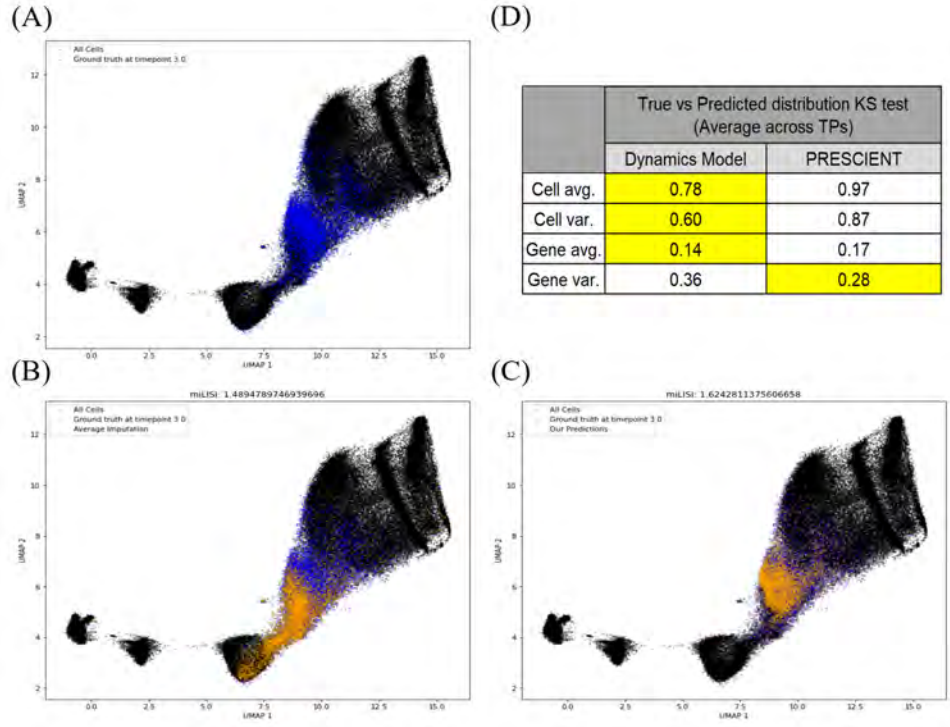


Figure 2: The results of our experiments using our model to improve cell type labelling in (A) the Mouse dataset and (B) simulated single-cell data from Splatter. Plots show how the Precision of a Random Forest Classifier at separating one cluster from the others changes as the amount of original data from the target cluster varies from 50% to 3%. The baselines "training" and "upsampling" refer to using only the available data and to randomly upsampling the available data back to the original dataset size, respectively. In (B), we show how the performance of data augmentation with our model (VAE) compares to existing approaches (scDesign2, DCA, ZINBWaVE). Data points are not shown for methods when the number of cells remaining after downsampling is below the minimum threshold allowed by their implementation.

our structure model could re-capitulate the clusters obtained when all the data was available. As shown in Figure 2A, our structure model performs well on this task, especially for clusters with proportionally very few cells. This indicates that our structure model may be a valuable tool in helping to cluster and study rare cell types in settings where large scale single-cell data-sets are unavailable.

2.3 The dynamics model can accurately generate cells from unseen future timepoints

Having successfully trained our structure model and evaluated the accuracy of its predictions, we moved on to training our dynamics model within the learned latent space. For this we again decided to use the Waddington-OT data-set due to its large number of timepoints and clear temporal structure. Unlike the mouse embryo dataset, cells in the Waddington-OT dataset showed a pronounced and consistent change in expression over time for our model to capture. For each timepoint t in the range 2.0 to 8.0, we trained our RNN on the latent space representation of data from times 0.0 to $t-1$. We then evaluated the prediction made by our models against the ground-truth expression data for cells at the unseen timepoint t. Figure 3 shows the predictions made by our model for timepoint 3.0 as an example.



	True vs Predicted distribution KS test (Average across TPs)	
	Dynamics Model	PRESCIENT
Cell avg.	0.78	0.97
Cell var.	0.60	0.87
Gene avg.	0.14	0.17
Gene var.	0.36	0.28

Figure 3: Results demonstrating the accuracy of predictions made by our dynamics model using time point 3.0 as an example. Figure (A) shows a UMAP plot of the true data for that timepoint in blue, set against the full trajectory in black for reference. (B) compares this to the true data from the previous timepoint (2.5) as a baseline, while (C) compares it to the data predicted by our model when trained on the six timepoints from 0.0 to 2.5. The miLISI mixing score is shown for each, indicating our model (1.62) outperforms the baseline (1.49). Finally, (D) shows the average performance across all timepoints of our dynamics model compared to PRESCIENT, as measured by KS tests statistics between four pairs of uni-variate distributions of the true and predicted data.

As a simple baseline to compare our model against, we first show that our models predictions are more accurate than simply using data from the previous timepoint. That is, to argue that our model is truly capturing meaningful temporal dynamics in the data, it should perform better than a naive model which simply assumes that nothing changes from the previous timestep. This baseline actually performs surprisingly well, due both to the low number of genes with differing expression over time and the significant overlap between adjacent timepoints. However, our model still outperforms it, producing predictions that are both more well mixed with the true data (Average Structure miLISI = 1.476, Average Baseline miLISI = 1.204) and harder to separate with a simple binary classifier (Average Structure AUROC = 0.809, Average Baseline AUROC = 0.886).

2.4 Predicting expression from future timepoints improves the identification of differentially expressed genes

Having shown that we can generate accurate expression profiles for cells at unseen future timepoints along a trajectory, we next wanted to explore whether these predictions can be used to gain relevant biological insights. A common task when working with time series expression data is to find genes that are differentially expressed across time. This can help identify the factors driving progress along trajectories and improve our understanding of the regulatory landscape of differentiation. Here, we perform an experiment to test whether augmenting a dataset with predictions made by our dynamics model can improve the detection of differentially expressed genes.

For this experiment we again used the Waddington-OT dataset and considered three different scenarios. In the first, only the data from the four timepoints 2.0 to 3.5 are available as a abseline. In the second, we take this same data but augment it with our dynamics models predictions for timepoint 4.0. Finally, in the third case all the true data from timepoints 2.0 to 4.0 is available. We used the scRNA-Seq trajectory analysis software package Monocle [33] to call deferentially expressed genes in all three cases, and then compared whether the baseline or augmented data yielded genes that were more similar to the ground truth genes obtained using the true day 4.0 data. The results are shown in Figure 4. We see that the area under the Precision-Recall curve, showing how accurately differently expressed genes are discovered at various p-value cutoffs, is much higher using the augmented data than the baseline. Additionally, when called at an FDR of 0.01, we see that using the augmented data allows for the accurate discovery of 32 more differentially expressed genes than the baseline at the expense of just 16 additional false positives. Overall, this demonstrates that the predictions made by our dynamics model can be useful for important downstream analysis' of single-cell RNA-Seq experiments.

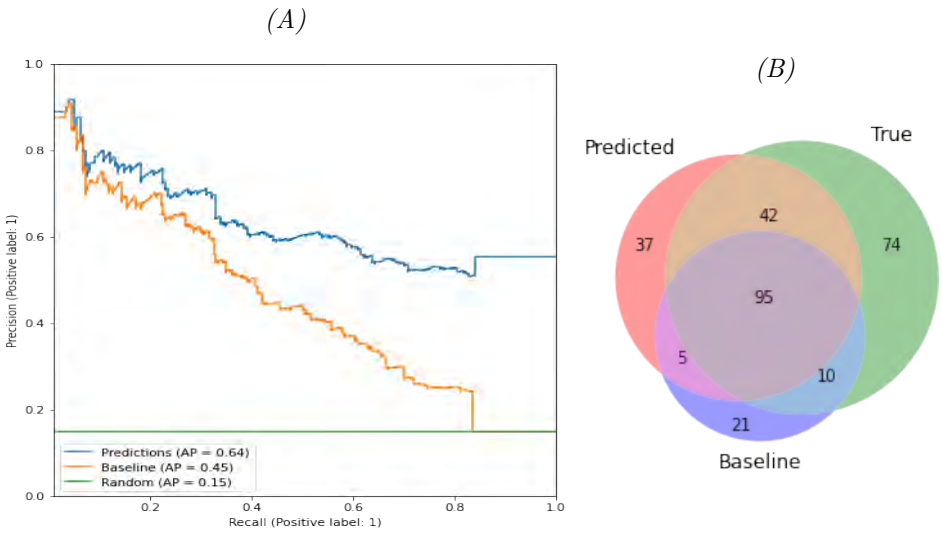


Figure 4: Results showing that augmenting time series data with predictions made by our structure model can improve the discovery of genes that are differentially expressed over time. Using only timepoints 2.0 to 3.5, (A) shows precision-recall curves for calling genes that are differentially expressed based on p-values generated by Monocle. 'Baseline' is run using only the available data, while 'Predictions' uses the data augmented with future predictions for timepoint 4.0 made by our model. The ground truth is obtained by using the true data for timepoint 4.0. 'Random' shows the PR curve obtained by random chance (Only $\sim 13\%$ of genes are differentially expressed). Plot (B) is a Venn diagram showing the overlap of genes called as differentially expressed (0.01 fdr threshold) using our baseline, predicted trajectory, and the true trajectory.

3 Discussion

Here we have presented our structure and dynamics models as tools for analyzing single-cell expression matrices. Our structure model implements a VAE architecture to optimize the variational lower bound of the posterior probability of the data. This allows for efficient sampling from the underlying probability distribution, enabling downstream analyses such as clustering and the discovery of rare cell types. Additionally, it learns a useful latent representation of the data for our dynamics module to operate on, which uses an RNN to predict the evolution of this probability distribution of cells over time. This permits sampling of *in silico* gene expression trajectories which we show can improve the discovery of genes that are differentially expressed during differentiation.

While we test a number of ways in which our models can be applied to generate biological insights, for this project we focused primarily on designing, implementing, and validating our structure and dynamics models. There is still room for much future work in exploring ways that we can apply them to understanding the factors regulating gene expression. An exciting possibility is using our dynamics model to

run *in silico* perturbational screens to see how the introduction or removal of key regulators can affect cell fate. Our model would allow us to quickly and cheaply test thousands of combinations to prioritise those that should be followed up *in vitro*. This has the potential to help uncover aberrant pathways leading to developmental diseases, as well as to aid in the engineering of reprogramming procedures for stem-cell-derived therapies.

Another future direction for this work is to apply interpretability techniques to understand our models predictions. This is a major challenge, as the dense networks for the encoder/decoder of our VAE and RNN tend to permit very little in the way of interpretation. One interesting angle might be to investigate further which dimensions of the data each latent variable encodes. Alternatively, we could look into using a more interpretable attention based architecture for our dynamics module. Despite the recent successes of transformers in the natural language domain, we felt that the task here had neither the scale of data nor the long-range temporal dependencies to necessitate the added complexities of the architecture. However, introducing an attention mechanism in place of our RNN in the dynamics model would allow for easier interpretation, which could make it a direction worth exploring.

4 Methods

4.1 Structure model architecture

4.1.1 Variational Autoencoders

To learn a low dimensional representation of our expression data, we decided to implement our structure model as a variational autoencoder. While architecturally similar to a vanilla autoencoder, variational autoencoders optimize a very different objective and satisfy a number of additional properties that are desirable for our task. Formally, the goal of a variational autoencoder is to take a dataset $X = \{x_1, x_2, ..., x_n\}$ drawn from some unknown underlying distribution $P(x)$ and learn an approximation for P. To do this, we first consider a vector z of latent variables that ideally capture a lot of information in x. For example, in our setting, z might encode properties of the cell that are relevant to its gene expression such as cell type, location in the cell cycle, and metabolic state. If we consider some function f_Φ paramaterized by Φ which maps these latent vectors to our space of X's, we can marginalize $P(x)$ over our latent variables z to get:

$$P(x) = \int_z P_\Phi(x|z)P(z)dz \tag{1}$$

Where $P_\Phi(x|z)$ is a Gaussian distribution giving the probability of obtaining the expression profile x given our latent representation z and our parameters Φ of f. For our VAE model, we are going to implement f_Φ as a dense neural network with

weights and biases Φ. This is a natural choice given the successes of neural networks at learning complex hierarchical functions and scaling to large amounts of data.

The question now is to decide how we can generate $z's$ that capture useful latent information about x. Counterintuitively, for our variational autoencoder we are simply going to use $z \sim N(0,1)$. This is justifiable because there will always exist some function that maps the standard normal distribution to any arbitrary distribution D[5]. Given sufficient capacity, this function may be approximated by our function f_Φ. Therefore, we can conceptually think of f_Φ as capturing both a mapping from $N(0,1)$ to a useful distribution of latent vectors and the mapping from this latent distribution to the expression profile x.

Technically, we now have everything we need to model $P(x)$. We could sample a large number of z's from $N(0,1)$, calculate $P_\Phi(x|z)$ for each in order to calculate $P(x)$ based on equation (1), and then use gradient ascent to update our weights Φ in the direction that maximizes the likelihood of the training data X. Unfortunately, the number of samples of z needed to train this model using this approach would be intractably large. To understand why, consider how small the space of biologically reasonable transcriptional profiles for n genes is compared to all of $\mathbb{R}^n$. $P_\Phi(x|z) \approx 0$ for nearly all z. This means that a very large number of z's will need to be sampled before the model randomly stumbles across an output that looks even remotely similar to a training example. To address this problem, we want a way to generate candidate latent vectors z that are likely to correspond to a given input x. We do this by introducing a new function $Q_\theta(z|x)$ paramaterized by the variables θ, which models the probability of z producing output x. For our VAE, we will model Q_θ as a dense neural network with weights θ that takes in an input x and outputs the vectors μ and ν representing the means and variances for a multi-variate Gaussian. The reason for this formulation will become clear shortly. Although introducing Q_θ solves one issue, we can now no longer use equation (1) to calculate $P(x)$ as we are drawing z's from $Q_\theta(z|x)$ rather than $P(z)$. Instead, we will need to introduce the notion of KL-divergence, which measures the distance between two probability distributions based on their relative entropy.

Let's start by writing out what it is we are now trying to optimize. First, we want to accurately model $P(x)$, which we will do by minimizing the negative log-likelihood as is customary. Second, we want to learn a good function $Q_\theta(z|x)$ which is close to the true distribution $P(z|x)$. We will achieve this 'closeness' by minimizing the KL-divergence $D_{KL}(Q_\theta(z|x)||P(z|x))$. We will now use the definition of KL-divergence $D_{KL}(Q_\theta(z|x)||P(z|x)) = \mathbb{E}_{z\sim Q}(\log Q_\theta(z|x) - \log P(z|x))$ in order to rewrite our objective into a form that is more easily computable:

$$
\begin{aligned}
-\log P(x) + D_{KL}(Q_\theta(z|x)||P(z|x)) &= -\log P(x) + \mathbb{E}_{z\sim Q}(\log Q_\theta(z|x) - \log P(z|x)) \\
&= \mathbb{E}_{z\sim Q}(\log Q_\theta(z|x) - \log \frac{P_\Phi(x|z)P(z)}{P(x)} - \log P(x)) \\
&= \mathbb{E}_{z\sim Q}(\log Q_\theta(z|x) - \log P_\Phi(x|z) - \log P(z)) \\
&= \mathbb{E}_{z\sim Q}(-\log P_\Phi(x|z)) + D_{KL}(Q_\theta(z|x)||P(z))
\end{aligned}
$$

Since the KL-divergence is always positive, the above provides a variational lower bound on $P(x)$ which we can use to maximize the posterior likelihood of the observed data under our model. Thus, we finally have obtained an objective function for optimizing our weights Φ and θ. Since $P_\Phi(x|z)$ is a multivariate Gaussian, its negative log is proportional to $MSE(x, f_\Phi(z))$. We can also approximate its expectation by sampling many training examples from our dataset. Additionally, the KL-Divergence of $Q_\theta(z|x)$ and $P(z)$ has an easily computable closed form, since both distributions are Gaussians. At this point, it's worth noting that even though we approached the problem entirely from the perspective of Gaussian graphical models, something that looks very much like an autoencoder has emerged. We have an encoder function Q_θ that maps an input gene expression profile x to the means and variances of the latent vector z, and a decoder function f_Φ mapping latent vectors z to output example x's. Writing things in more conventional notation for autoencoders, we have two feed forward neural networks Φ_{enc} and Φ_{dec} which we train using the loss function:

$$
L(x) = MSE(x, \Phi_{dec}(\Phi_{enc}(x))) + D_{KL}(\Phi_{enc}(x)||N(0,1)) \tag{2}
$$

The last thing we need to consider is that in order to optimize our model using this loss function, it needs to be differentiable with respect to the input x. Unfortunately, the loss function as described above is not. We are unable to propagate the derivative through the sampling of z from $N(\mu, \nu)$ in the latent space. We work around this by using the reparameterization trick, where instead of sampling from $N(\mu, \nu)$, we sample from $N(0, 1)$, multiply the result by the variances ν and add the means μ. This way, we have separated the stochasticity of the sampling from the parameters μ and ν themselves, so the gradients can propagate back through them to the weights of Φ_{enc}.

4.1.2 Desirable Properties of VAEs

Now that we have covered the architecture of our structure model, we will discuss why a variational autoencoder is an appropriate choice for our task. The first reason is that we want our model to be generative. Having trained our model on the data from an scRNA-Seq experiment, we want to be able to sample from the underlying distribution of the data to generate *de novo* transcriptional profiles. Since variational autoencoders are formulated for the express purpose of drawing samples from $P(x)$,

they are ideal for this task. We can generate as many new transcriptional profiles as are needed by sampling z's from the latent space. With traditional autoencoders, on the other hand, generating new samples is difficult. You aren't imposing any structure on the latent space, so you have no guarantee that any point which isn't the latent representation of a training example won't be decoded to absolute nonsense. This leads into two desirable properties of the latent space of variational autoencoders - continuity and completeness. Continuity refers to the idea that nearby points in the latent space should be decoded to similar outputs, and completeness is the idea that any point sampled from your latent space should decode to a plausible output. Variational autoencoders introduce continuity into their latent space by enforcing the constraint that the variance of z be close to one, meaning that any input during training may be mapped to a region of latent representations which should all be decoded into something similar to that input. Completeness is achieved by enforcing that the mean of z be close to zero. This makes the latent representations of the training data be clustered tightly around the origin, and thus any latent point sampled from a standard normal should decode into a plausible looking transcriptional profile. Together, these properties are important for the second reason that we elected to use variational autoencoders, which is that we want to be able to safely train the dynamics model and make predictions within the latent space. When we use our RNN to predict the latent representation of a cell, we want to be confident that this latent representation will decode into a biologically possible gene expression profile for that cell. Additionally, it is desirable that improving the RNNs predictions in the latent space should also improve the accuracy of the decoded transcriptional profiles in the original space. Thus, we require that points which are closer together in the latent space also have more similar decoded representations.

4.2 Dynamics model architecture

For our dynamics model, our goal is to predict the distribution of gene expression across cells at a future time point given the distribution of expression observed at previous time points. This task would be nearly intractable in the original high dimensional space of gene expression, so instead we are going to make predictions within the latent space learned by our structure model. More formally, given a series of single-cell RNA-Seq experiments $X_1, ..., X_{t-1}$ taken at time points $1...t-1$ we want to model $P(X_t|X_1, ..., X_{t-1})$. Marginalizing over the latent variables z as we did in (1), this becomes:

$$\int_z P_\Phi(X_t|z_t)P(z_t|z_1, ..., z_{t-1})dz$$

The decoder of our structure model has already learned a good approximation of $P_\Phi(X_t|z_t)$, so all that is left is to learn a model of $P(z_t|z_1, ..., z_{t-1})$. While we could try and model this function by simply concatenating $z_1, ..., z_{t-1}$ together and passing them into a simple dense network, this approach would scale poorly as the number

of time points t increased. Additionally, it would be inflexible to varying numbers of time points, since a network trained to take inputs from say five time points would be unable to make predictions if only four time points were available, as the input vector would be of the wrong dimension. Instead, we decided to use an RNN architecture, which has proven effective at working with sequential data in domains such as language modeling and signal processing.

At a high level, the RNN architecture works by processing the inputs one-by-one sequentially, and keeping track of a hidden state which contains a 'memory' of what the model has seen so far. At each time step the model will update the current hidden state based on the current input, and then use its new hidden state to predict the next output in the sequence. More formally, an RNN model consists of three trainable weight matrices U, W, V and two trainable bias vectors b_h, b_o. At time step i, the model will take in input z_i and the hidden state h_{i-1} and output:

$$h_i = tanh(Uz_i + Wh_{i-1} + b_h)$$
$$\hat{z}_{i+1} = Vh_i + b_o$$

Where h_i is the next hidden state for our model, $\hat{z}_{i+1}$ is our models prediction for the next time step, and tanh is the Hyperbolic Tangent activation function. Note that for the first step when $i = 1$, h is initialized to all zeros.

Given a sequence of latent representations of a cell $z_1, z_2, ..., z_{t-1}$, after feeding them into the RNN architecture described above we will get as output a series of predictions $\hat{z}_2, \hat{z}_3..., \hat{z}_t$. These outputs represent our model predictions of $P(z_2|z_1), P(z_3|z_1, z_2), ..., P(z_t|z_1, ..., z_{t-1})$. During training, we can compare these predictions to the true values of z_i in order to update the weights of our RNN. Since the vectors z contain the means and variances of a Gaussian to be sampled from, the loss function $L(z_i) = D_{KL}(z_i||\hat{z}_i)$ is a natural choice. Thus, our RNN model will learn to capture the observed temporal dynamics of cells within the latent space, which will hopefully generalize to generate future samples along the trajectory with similar underlying biological representations.

Putting it all together, our dynamics model can take in a time series scRNA-Seq dataset and make predictions about the distribution of expression at future time points. It does this by mapping the observed data into the latent space using the encoder of our VAE, using an RNN to predict the next latent representation based on that sequence of observed latent representations, and then decoding the predicted latent vector back into a gene expression profile using the decoder of the VAE. A diagram of the full layout is shown in Figure 5.

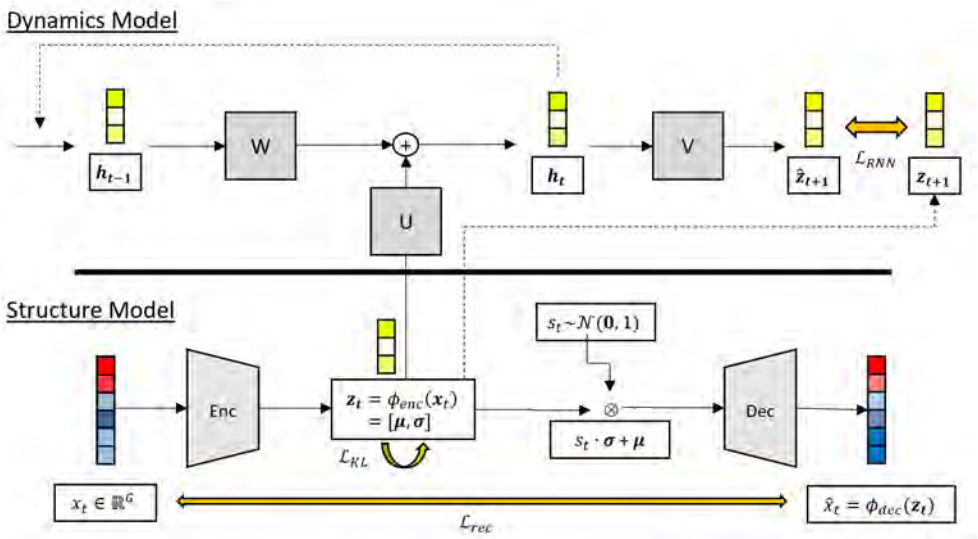


Figure 5: A diagram outlining the architecture of our structure and dynamics models. Dotted lines represent steps taking place at the next timepoint $t + 1$, and orange arrows point out the terms of our various loss functions during training.

4.3 Weighting of loss terms

In the loss function for our VAE shown in equation (2), there is one important component that we left out. In practice, it is often necessary introduce a weighting to the two terms in this loss function lest one term completely dominate another. For example, suppose that in our loss function the values of the MSE term were orders of magnitude larger than values of the KL-divergence term. This would mean our model would be incentivized to focus purely on obtaining an optimal reconstruction of the input at the expense of the latent space structure that motivated our use of VAEs in the first place. On the other hand, if the KL-divergence loss term dominates, the model will learn a latent representation that is perfectly standard normal for every input at the expense of being able to produce accurate outputs. Clearly, we need a way to identify a good middle ground.

In the field of deep learning generally, when working with objective functions that have multiple terms the most commonly used approaches for weighting them are to set weights such that each term has the same magnitude or to run a grid search to find the weighting that works best. However, neither of these approaches are very principled. The first ignores the fact that losses may be highly non-linear and have different minimum values, while the second is very computationally expensive. Additionally, the optimal weightings may be very setting specific, and we want users to be able to apply our method to new datasets without extensive fine tuning. Thus, we wanted a way to automatically set the loss weightings during training. For this, we elected to use a coefficient of variations (CoV) based dynamic loss weighting scheme proposed by Groenendijk et al [11]. This approach works under the assumption that

a loss term has been fully optimized once its variance approaches zero. Therefore, we should weight loss terms based on their coefficients of variation such that during training those terms which still show high variance compared to their mean are given higher importance. Formally, for each term i in our loss function, the weighting α_{it} on batch t should be given by:

$$\alpha_{it} = \frac{1}{\sum_k \frac{\sigma_{kt}}{\mu_{kt}}} * \frac{\sigma_{it}}{\mu_{it}}$$

where μ_{it} and σ_{it} are running averages over the last 100 batches of the mean and standard deviation of term i.

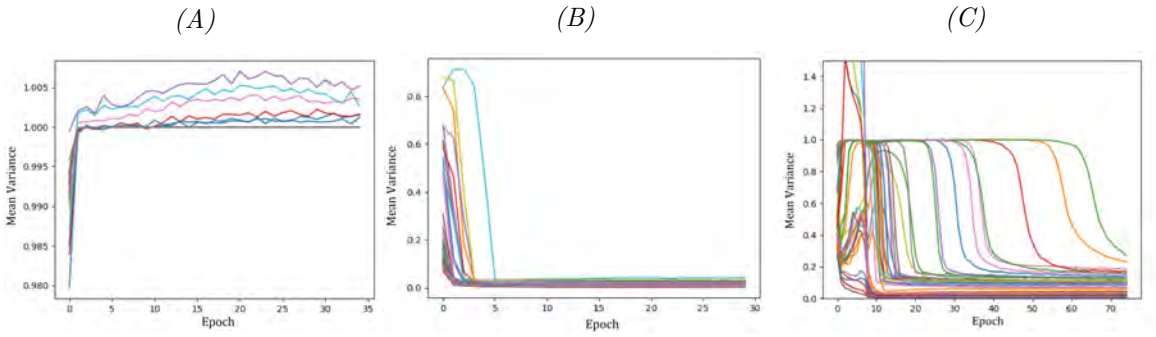


Figure 6: Three plots showing the mean variance $\mathbb{E}_{\mathbb{X}} z_{x,\sigma}$ for each of the 32 latent variables in our variational autoencoder when trained (A) with weighting for the KL-divergence that is too high, (B) with a weighting for the MSE that is too high, and (C) with the CoV dynamic loss weighting method.

To validate the success of using this approach, we performed an analysis method borrowed from Asperti et al to gain insight into the latent variables learned by our autoencoder [2]. Plotting the average variance for each of the 32 latent dimensions across each training epoch, we can see the trade off our model is making between satisfying the KL-divergence loss term (variances near one) and the reconstruction loss term (variances near 0). In Figure 6, we see these plots for three training runs - one where the KL-divergence term is weighted too high, one where the MSE term is weighted too high, and one using the dynamic CoV weighting. The third plot looks very similar to that of a successfully trained image autoencoder in Asperti et al, with latent variables slowly being taken from variance 1 and being put to work, but not collapsing all the way down to variance zero.

4.4 Evaluation Metrics

To understand the quality of predictions made by our model, we employed a number of different evaluation metrics in this paper. Here we will go into more detail about how each was calculated:

- **miLISI** The mean integration Local Inverse Simpson Index is a measure of how well mixed two datasets are. For each point in the two datasets, each of its 30 nearest neighbors are identified and weighted based on a Gaussian kernel. Then, the expected number of these neighbors that would need to be sampled before obtaining a second point from the same dataset is calculated, with the probabilities of choosing a neighbor weighted by its proximity. The average of this expectation is then taken across all points. If this average is 1, that indicates that the neighbors for each point all have the same label, and thus the two datasets are perfectly seperated. On the other hand, if it is two, that means each point has as many neighbors with the same label as with different labels, and the two datasets are perfectly mixed.

- **AUROC** As a way to test if simulated data accurately matches true data, we reasoned that a binary classifier should not be able to tell the two apart. To use this as an evaluation metric for predictions made by our models, we first reduced the dimensionality of both the predicted and true data by taking the first 50 principal components. Then we trained a random forest classifier with 100 trees of max depth 2 to separate the true and predicted data, and calculated its area under the receiver operator characteristic curve (AUROC) to measure its performance. If our generated data is accurate, the model should not be able to tell the two apart better than random guessing, and would have an AUROC of 0.5. Note that this metric is very strict - if there is even a single consistent difference between true and generated data, no matter how small, the classifier will be able to perfectly seperate the two classes and have an AUROC of 1.

- **KS-test** The Kolmogorov–Smirnov (KS) test is a non-parametric test of equality between two distributions. The test statistic measures the maximum vertical distance between two cumulative distribution functions (CDF), with 0 indicating that the two distributions are identical and 1 indicating that they have no overlap. To use this to evaluate predictions made by our models, we calculate the empirical CDF for four uni-variate summary statistics of our data (average expression across cells, average expression across genes, variance across cells, variance across genes) and then use the KS test to compare these to the distributions in the true data.

4.5 Data Preprocessing

For this project we focused our analysis on two single-cell RNA-Seq time series datasets, the Waddington-OT data [28] and the mouse embryo data [6]. Although the full Waddington-OT dataset contains data from across 18 days, at day 8.5 they treat the sample with a collection for transcription factors. To avoid this major perturbation from affecting our results, we only use data from the seventeen timepoints at days 0.0 to 8.0. For both datasets, we wanted to have an equal number of cells

per timepoint to avoid bias during training, so we down-sampled the data for each timepoint to have the same number of cells as the timepoint with the fewest cells. This gave 4,000 cells per timepoint for the Waddington-OT data and 8,221 cells per timepoint for the mouse embryo data.

To preprocess the two datasets we analyzed here, we started by taking the raw TPKM expression matrices and filtering cells with abnormally low/high gene expression. In our experiments, we also found that our model performs best when the raw TPKM expression matrix is normalized by unique UMIs per cell to account for the stochastic variation in sequencing depth between cells. Additionally, we found that our model performs slightly better in the original space of read-counts than in the log space. After filtering and normalizing the raw read matrices for the two experiments, we then used Scanpy's python implementation of Seurat to select the 2,000 most variable genes in the dataset. This is necessary since the vast majority of genes either show zero expression across all cells or are expressed at a nearly constant level in all cells. On the other hand, differentiation is driven by a small number of important regulatory genes. Thus, including only the most variable genes in the input significantly decreases the number of parameters of our model while still capturing those of biological interest.

4.6 Training Scheme

When training our VAE, we divided our data into a 70/15/15 train/validation/test split. We observed marginal evidence of our VAE model overfitting to the data, with validation loss starting to slowly increase while train loss continued to decrease. Thus, based on the training and validation loss curves shown in Figure 7, we decided to stop training after 75 epochs. To avoid the trained structure model leaking any information to the dynamics model about cells at a future unseen time point, we made sure that the structure model was only ever trained on data that would also be visible to the dynamics model during training. Specifically, this meant training a separate structure model to pair with each dynamics model, so for example the structure model would only be trained on day 0.0 - 3.0 data when predicting expression for day 3.5, but would be trained on all day 0.0 - 7.5 data when predicting expression for day 8.0.

After training our VAE, we then use it to map all of our training data into the latent space within which our dynamics model operates. Here, we face the challenge that our RNN requires as input a sequence of measurements across time points, but we have only a single measurement at a single time point for each cell. Although cells are all sampled from the same population, the scRNA-Seq assay is inherently destructive, meaning we can't get measurements from the same cell at multiple different time points. To overcome this, we use an Optimal-Transport approach to match the distributions of cells between time points. The idea is that as a population of cells moves along a differentiation trajectory, the distribution of expression within the population gradually shifts. If we find an optimal mapping between the distributions

at two time points based on some distance metric (in this case KL-divergence), we can match each individual cell in the first time point with its most likely descendent in the second time point, giving us an approximation of how expression of that cell would have progressed if it hadn't been destroyed during sequencing. More formally, we consider the space of matrices M denoting pairings of cells between time points t and $t+1$ with $M^k_{i,j} = 1$ if the ith cell at time point t is matched to the jth cell at time point $t+1$ and $M^k_{i,j} = 0$ otherwise. We then solve the optimization problem:

$$\min_{M^k \in M} \sum_i \sum_j D_{KL}(x_{t,i} || x_{t+1,j}) M^k_{i,j}$$

where $x_{t,i}$ denotes the expression profile of cell i from time point t. The obtained pairings are used as our input sequence for the dynamics model. We also experimented with matching cells probabalistically, where matchings for each epoch were sampled with probability inversely proportional to their cost rather than always using the single lowest cost matching. However, we observed no significant improvement to the models performance under this approach. Thus, we decided it wasn't worth the increased compute needed to re-calculate matchings at each epoch.

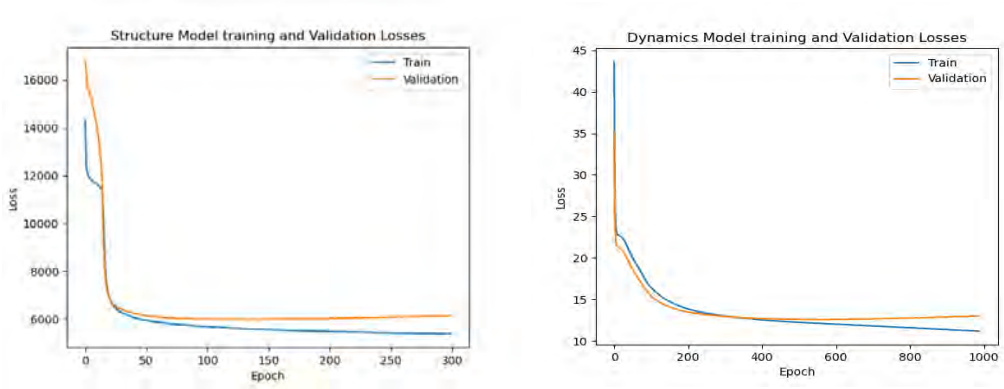


Figure 7: Plots showing Train and Validation loss per epoch while training our Structure (left) and Dynamics (right) models.

When training our RNN with data from time points 0 to t available, we use all data from time points 0 to $t-2$ as training data, data from time point $t-1$ as a validation set, and data from time point t as a test set. Similar to our VAE model, we observed little evidence of over-fitting during training. Based on our loss curves shown in Figure 7, we decided to train for 300 epochs, beyond which our validation loss stopped decreasing. For both our VAE and RNN models we used a batch size of 256, which was the largest batch that could fit into memory on our GPU. Additionally, we performed weight updates during training using the commonly used first-order stochastic optimization algorithm Adam [20].

4.7 Hyperparameter optimization

To explore the effect of various hyperparameters in our model, we ran a grid search over all combinations of the hyperparameters shown in table 2. To accommodate for differences in sample efficiency under different hyperparameter combinations, models were trained for 1000 epochs (far more than would ever be necessary) and evaluated after each epoch. The evaluation at the best performing epoch was then saved. Since the task doesn't permit a nice clean metric like validation accuracy which we can easily optimize for, we instead had to evaluate each run with a number of the metrics described above - Pearson correlation, Spearman correlation, RF-classifier AUROC, miLISI, and the KS-test. Thankfully, all of these metrics corresponded well to each other - we never observed a run with, for example, a very high miLISI but a low Pearson Correlation.

Hyperparameter	**Values**
VAE Learning Rate	0.01, 0.001, 0.0001
RNN Learning Rate	0.01, 0.001, 0.0001
Initialization Type	Glorot, Zeros
Layer Size Determination	Log, Linear
Number of Layers	2, 3, 4
Latent Size	16, 32, 64, 128, 256

Table 2: Values for each hyperparameter that were explored with a grid search. Every possible combination was tested yielding a total of 540 different training runs. Initialization type describes how the initial values of trainable parameters are set (either sampled from a Gaussian or set to zero), and layer size determination refers to how the size of the intermediate hidden layers decrease from the input size to the latent size (either logarithmically or linearly). Based on the results, the final selected hyperparameters were VAE Learning Rate of 0.0001, RNN Learning Rate of 0.0001, Initialization type of Glorot, Layer Size Determination of Linear, Number of layers of 4, and Latent size of 32.

Another challenge we had to navigate was the trade-off between performance of the structure model and performance of the dynamics model. For example, the structure model performs better with larger latent sizes. This is expected, as in the extreme if the latent size was 2000 our auto-encoder wouldn't need to learn a lower dimensional representation at all, and could reconstruct the input perfectly. However, our dynamics model performs better with lower dimensional latent sizes. Remember the hypothesis that trajectories are more easily modeled on low dimensional manifolds is the whole motivation for our architecture in the first place - otherwise we could have just fed the expression profiles straight into an RNN directly. To deal with this trade off, we selected intermediate values for those hyperparameters that worked reasonably well for both models.

5 References

References

[1] J. Acosta, D. Ssozi, and P. van Galen. "Single-Cell RNA Sequencing to Disentangle the Blood System". In: *Arterioscler Thromb Vasc Biol* 41.3 (Mar. 2021), pp. 1012–1018.

[2] Andrea Asperti and Matteo Trentin. *Balancing reconstruction error and Kullback-Leibler divergence in Variational Autoencoders*. 2020. DOI: 10.48550/ARXIV.2002.07514. URL: https://arxiv.org/abs/2002.07514.

[3] Etienne Becht et al. "Dimensionality reduction for visualizing single-cell data using UMAP". In: *Nature Biotechnology* 37.1 (Jan. 2019), pp. 38–44. ISSN: 1546-1696. DOI: 10.1038/nbt.4314. URL: https://doi.org/10.1038/nbt.4314.

[4] Samuel R. Bowman et al. "Generating Sentences from a Continuous Space". In: *CoRR* abs/1511.06349 (2015). arXiv: 1511.06349. URL: http://arxiv.org/abs/1511.06349.

[5] G. E. P. Box and D. R. Cox. "An Analysis of Transformations". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 26.2 (1964), pp. 211–252. ISSN: 00359246. URL: http://www.jstor.org/stable/2984418.

[6] Junyue Cao et al. "The single-cell transcriptional landscape of mammalian organogenesis". In: *Nature* 566.7745 (Feb. 2019), pp. 496–502. ISSN: 1476-4687. DOI: 10.1038/s41586-019-0969-x. URL: https://doi.org/10.1038/s41586-019-0969-x.

[7] Tara Chari, Joeyta Banerjee, and Lior Pachter. "The Specious Art of Single-Cell Genomics". In: *bioRxiv* (2021). DOI: 10.1101/2021.08.25.457696. eprint: https://www.biorxiv.org/content/early/2021/08/26/2021.08.25.457696.full.pdf. URL: https://www.biorxiv.org/content/early/2021/08/26/2021.08.25.457696.

[8] Helena L. Crowell et al. "Built on sand: the shaky foundations of simulating single-cell RNA sequencing data". In: *bioRxiv* (2022). DOI: 10.1101/2021.11.15.468676. eprint: https://www.biorxiv.org/content/early/2022/02/23/2021.11.15.468676.full.pdf. URL: https://www.biorxiv.org/content/early/2022/02/23/2021.11.15.468676.

[9] Gökcen Eraslan et al. "Single-cell RNA-seq denoising using a deep count autoencoder". In: *Nature Communications* 10.1 (Jan. 2019), p. 390. ISSN: 2041-1723. DOI: 10.1038/s41467-018-07931-2. URL: https://doi.org/10.1038/s41467-018-07931-2.

[10] Jeffrey A. Farrell et al. "Single-cell reconstruction of developmental trajectories during zebrafish embryogenesis". In: *Science* 360.6392 (2018), eaar3131. DOI: 10.1126/science.aar3131. eprint: https://www.science.org/doi/pdf/10.1126/science.aar3131. URL: https://www.science.org/doi/abs/10.1126/science.aar3131.

[11] Rick Groenendijk et al. *Multi-Loss Weighting with Coefficient of Variations*. 2020. DOI: 10.48550/ARXIV.2009.01717. URL: https://arxiv.org/abs/2009.01717.

[12] Christopher Heje Grønbech et al. "scVAE: variational auto-encoders for single-cell gene expression data". In: *Bioinformatics* 36.16 (May 2020), pp. 4415–4422. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btaa293. eprint: https://academic.oup.com/bioinformatics/article-pdf/36/16/4415/33965265/btaa293.pdf. URL: https://doi.org/10.1093/bioinformatics/btaa293.

[13] Suleyman Gulsuner et al. "Spatial and Temporal Mapping of De Novo Mutations in Schizophrenia to a Fetal Prefrontal Cortical Network". In: *Cell* 154.3 (2013), pp. 518–529. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2013.06.049. URL: https://www.sciencedirect.com/science/article/pii/S0092867413008313.

[14] S. C. Hicks et al. "Missing data and technical variability in single-cell RNA-sequencing experiments". In: *Biostatistics* 19.4 (Oct. 2018), pp. 562–578.

[15] B. Huang et al. "Decoding the mechanisms underlying cell-fate decision-making during stem cell differentiation by random circuit perturbation". In: *J R Soc Interface* 17.169 (Aug. 2020), p. 20200500.

[16] Ruochen Jiang et al. "Statistics or biology: the zero-inflation controversy about scRNA-seq data". In: *Genome Biology* 23.1 (Jan. 2022), p. 31. ISSN: 1474-760X. DOI: 10.1186/s13059-022-02601-5. URL: https://doi.org/10.1186/s13059-022-02601-5.

[17] Peter V. Kharchenko. "The triumphs and limitations of computational methods for scRNA-seq". In: *Nature Methods* 18.7 (July 2021), pp. 723–732. ISSN: 1548-7105. DOI: 10.1038/s41592-021-01171-x. URL: https://doi.org/10.1038/s41592-021-01171-x.

[18] Tae Hyun Kim, Xiang Zhou, and Mengjie Chen. "Demystifying "drop-outs" in single-cell UMI data". In: *Genome Biology* 21.1 (Aug. 2020), p. 196. ISSN: 1474-760X. DOI: 10.1186/s13059-020-02096-y. URL: https://doi.org/10.1186/s13059-020-02096-y.

[19] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114.

[20] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[21] Ashley Laughney et al. "Regenerative lineages and immune-mediated pruning in lung cancer metastasis". In: *Nature Medicine* 26 (Feb. 2020). DOI: 10.1038/s41591-019-0750-6.

[22] Xiaodong Liu et al. "Modelling human blastocysts by reprogramming fibroblasts into iBlastoids". In: *Nature* 591.7851 (Mar. 2021), pp. 627–632. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03372-y. URL: https://doi.org/10.1038/s41586-021-03372-y.

[23] A. Nguyen et al. "Single Cell RNA Sequencing of Rare Immune Cell Populations". In: *Front Immunol* 9 (2018), p. 1553.

[24] Anoop P. Patel et al. "Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma". In: *Science* 344.6190 (2014), pp. 1396–1401. DOI: 10.1126/science.1254257. eprint: https://www.science.org/doi/pdf/10.1126/science.1254257. URL: https://www.science.org/doi/abs/10.1126/science.1254257.

[25] Chengxiang Qiu et al. "Systematic reconstruction of cellular trajectories across mouse embryogenesis". In: *Nature Genetics* 54.3 (Mar. 2022), pp. 328–341. ISSN: 1546-1718. DOI: 10.1038/s41588-022-01018-x. URL: https://doi.org/10.1038/s41588-022-01018-x.

[26] A. Raj et al. "Stochastic mRNA synthesis in mammalian cells". In: *PLoS Biol* 4.10 (Oct. 2006), e309.

[27] Davide Risso et al. "A general and flexible method for signal extraction from single-cell RNA-seq data". In: *Nature Communications* 9.1 (Jan. 2018), p. 284. ISSN: 2041-1723. DOI: 10.1038/s41467-017-02554-5. URL: https://doi.org/10.1038/s41467-017-02554-5.

[28] Geoffrey Schiebinger et al. "Optimal-Transport Analysis of Single-Cell Gene Expression Identifies Developmental Trajectories in Reprogramming". In: *Cell* 176.4 (2019), 928–943.e22. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2019.01.006. URL: https://www.sciencedirect.com/science/article/pii/S009286741930039X.

[29] R. A. Simmons. "Developmental origins of adult disease". In: *Pediatr Clin North Am* 56.3 (June 2009), pp. 449–466.

[30] Tianyi Sun et al. "scDesign2: a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured". In: *Genome Biology* 22.1 (May 2021), p. 163. ISSN: 1474-760X. DOI: 10.1186/s13059-021-02367-2. URL: https://doi.org/10.1186/s13059-021-02367-2.

[31] Valentine Svensson. "Droplet scRNA-seq is not zero-inflated". In: *Nature Biotechnology* 38.2 (Feb. 2020), pp. 147–150. ISSN: 1546-1696. DOI: 10.1038/s41587-019-0379-5. URL: https://doi.org/10.1038/s41587-019-0379-5.

[32] Alexander Tong et al. *TrajectoryNet: A Dynamic Optimal Transport Network for Modeling Cellular Dynamics*. 2020. DOI: 10.48550/ARXIV.2002.04461. URL: https://arxiv.org/abs/2002.04461.

[33] Cole Trapnell et al. "The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells". In: *Nature Biotechnology* 32.4 (Apr. 2014), pp. 381–386. ISSN: 1546-1696. DOI: 10.1038/nbt.2859. URL: https://doi.org/10.1038/nbt.2859.

[34] Po-Yuan Tung et al. "Batch effects and the effective design of single-cell gene expression studies". In: *Scientific Reports* 7.1 (Jan. 2017), p. 39921. ISSN: 2045-2322. DOI: 10.1038/srep39921. URL: https://doi.org/10.1038/srep39921.

[35] Peter van Galen et al. "Single-Cell RNA-Seq Reveals AML Hierarchies Relevant to Disease Progression and Immunity". In: *Cell* 176.6 (2019), 1265–1281.e24. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2019.01.031. URL: https://www.sciencedirect.com/science/article/pii/S0092867419300947.

[36] Michel Verleysen and Damien François. "The Curse of Dimensionality in Data Mining and Time Series Prediction". In: *Computational Intelligence and Bioinspired Systems*. Ed. by Joan Cabestany, Alberto Prieto, and Francisco Sandoval. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 758–770. ISBN: 978-3-540-32106-4.

[37] Grace Hui Ting Yeo, Sachit D. Saksena, and David K. Gifford. "Generative modeling of single-cell time series with PRESCIENT enables prediction of cell trajectories with interventions". In: *Nature Communications* 12.1 (May 2021), p. 3222. ISSN: 2041-1723. DOI: 10.1038/s41467-021-23518-w. URL: https://doi.org/10.1038/s41467-021-23518-w.

[38] Luke Zappia, Belinda Phipson, and Alicia Oshlack. "Splatter: simulation of single-cell RNA sequencing data". In: *Genome Biology* 18.1 (Sept. 2017), p. 174. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1305-0. URL: https://doi.org/10.1186/s13059-017-1305-0.

6 Supplementary Figures

	Day 2.0		Day 2.5		Day 3.0		Day 3.5	
	Dynamics Model	PRESCIENT	Dynamics Model	PRESCIENT	Dynamics Model	PRESCIENT	Dynamics Model	PRESCIENT
Cell avg.	0.90	0.99	0.52	0.99	0.95	0.99	0.90	1.00
Cell var.	0.55	0.98	0.99	0.94	0.81	0.96	0.86	0.94
Gene avg.	0.10	0.16	0.23	0.15	0.12	0.15	0.21	0.15
Gene var.	0.40	0.24	0.32	0.24	0.42	0.25	0.34	0.22
	Day 4.0		Day 4.5		Day 5.0		Day 5.5	
	Dynamics Model	PRESCIENT	Dynamics Model	PRESCIENT	Dynamics Model	PRESCIENT	Dynamics Model	PRESCIENT
Cell avg.	0.78	0.99	0.55	0.99	0.84	0.98	0.79	0.97
Cell var.	0.43	0.92	0.17	0.87	0.64	0.89	0.62	0.82
Gene avg.	0.14	0.17	0.14	0.17	0.09	0.17	0.12	0.20
Gene var.	0.38	0.26	0.41	0.30	0.34	0.30	0.33	0.29
	Day 6.0		Day 6.5		Day 7.0		Average	
	Dynamics Model	PRESCIENT	Dynamics Model	PRESCIENT	Dynamics Model	PRESCIENT	Dynamics Model	PRESCIENT
Cell avg.	0.52	0.91	0.79	0.86	0.99	0.96	0.78	0.97
Cell var.	0.16	0.72	0.43	0.61	0.90	0.88	0.60	0.87
Gene avg.	0.11	0.21	0.12	0.18	0.12	0.17	0.14	0.17
Gene var.	0.33	0.30	0.37	0.33	0.33	0.33	0.36	0.28

Figure 8: The full results of our Dynamics module on the Waddington-OT dataset, as measured by our KS-test metric. The average across these timepoints is shown in Figure 3D.

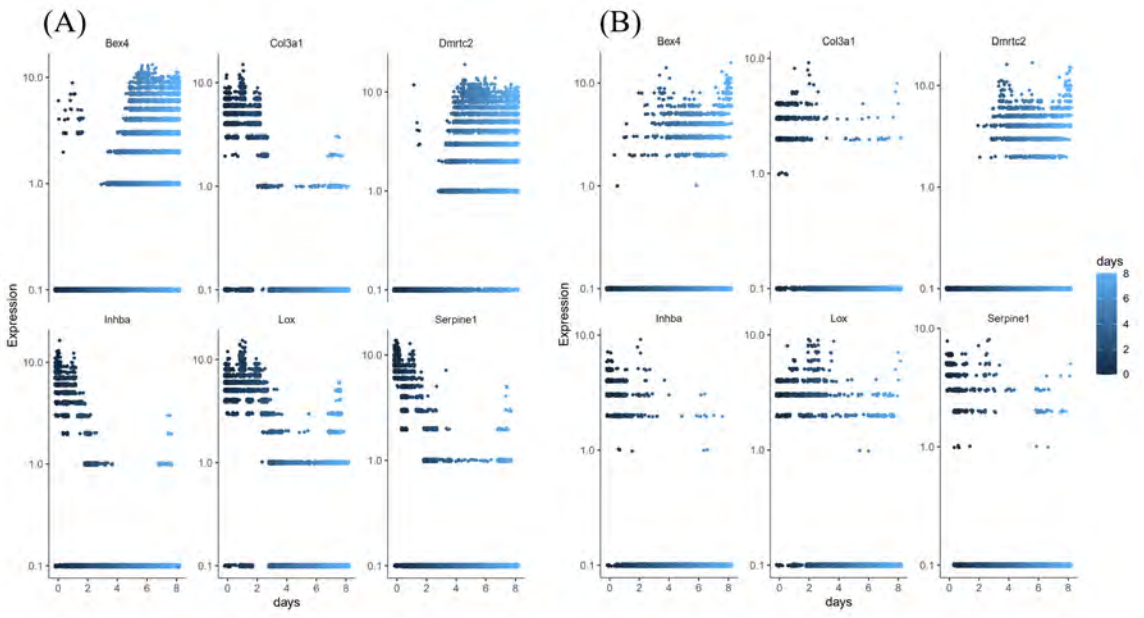


Figure 9: Scatter-plots showing the expression in each cell for six differentially expressed genes in (A) the data predicted by our dynamics model and (B) the true data. We can see that while our model captures the patterns of expression well across time, it does not have the same sparsity as the true data. Differential expression calling and plots produced by Monocle [33].