Modeling gene regulatory dynamics in neural differentiation at the single cell level using a machine learning approach

Yixing Hu, MSc

Department of Human Genetics, Mcgill University, Montreal

Oct 2021

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science (MSc) in Human Genetics

Abstract

Cellular differentiation is an important process where progenitor cells progressively develop into mature cells with specialized functions. Understanding the molecular characteristics and underlying regulatory mechanisms of cell fate is a central goal in biological research. Advances in single-cell sequencing technology enable the exploration of cellular differentiation at unprecedented resolution. In this thesis, I focus on characterizing and modeling of cellular differentiation using machine learning approaches. First, I present a random forest approach to identify the most discriminant genes for different cell populations in the developing brain. This method was able to identify key gene markers that revealed dorsal–ventral patterning in a heterogeneous class of progenitors present in a mouse developmental time-series dataset. Next, as cellular differentiation is marked by continuous changes in gene expression and is not well described by static cell populations, I present a framework to model the dynamics of cell fate decisions based on ordinary differential equations (ODE). I train this model on previously reported trajectory data for neural differentiation, and show that the model is able to interpolate and predict the gene expression dynamics across unobserved regions in this trajectory and extend the system dynamics for neural differentiation data. Finally, by training the model on datasets that contain rate of change information for each gene (RNA velocity), I demonstrate that the model has the capacity to predict the effects of gene deletions to the cell's overall gene expression profile with a prediction accuracy of 90%. In summary, the Neural ODE method has the ability to learn the gene regulatory dynamics from single cell data and predict the dynamics of individual genes as well as perturbation response.

Résumé

La maturation cellulaire est un processus au cours duquel les cellules progénitrices se développent progressivement en cellules matures de fonctions spécialisées. Une meilleure compréhension des caractéristiques moléculaires et des mécanismes réglementaires du destin cellulaire est un objectif central dans la recherche biologique. Des progrès dans la technologie de séquençage des cellules simples permettent l'analyse de la différenciation cellulaire à une résolution sans précédent. Dans cette thèse, je modélise la maturation cellulaire à l'aide d'approches d'apprentissage machine. Je commence par présenter une approche utilisant des forêts d'arbres décisionnels pour identifier les gènes les plus discriminants pour différentes populations cellulaires dans le cerveau en développement. Cette méthode a permis d'identifier des marqueurs génétiques dans des données du développement du cerveau murin qui ont révélé une structuration dorso-ventrale dans une classe hétérogène de progéniteurs. Ensuite, comme la maturation cellulaire est marquée par un continuum d'expression et n'est pas bien décrite par des populations cellulaires statiques, je présente une méthode pour modéliser les dynamiques de décisions sur le destin cellulaire utilisant des équations différentielles ordinaires (EDO). J'entraîne ce modèle sur des données de trajectoire rapportées pour la différenciation neurale. Je démontre que le modèle est capable d'interpoler et de prédire la dynamique d'expression génique dans des régions non observées de cette trajectoire et peut étendre la dynamique du système. Enfin, en entraînant le modèle sur des données contenant le taux de changement pour chaque gène (vélocité de l'ARN), je démontre que le modèle est capable de prédire les effets de suppression de gènes sur le profil d'expression global de la cellule avec une précision de prédiction de 90 %. Dans l'ensemble, la méthode Neural EDO est capable d'apprendre la dynamique de régulation des gènes à partir de données de cellules simples et peut prédire la dynamique des gènes individuels ainsi que les effets des perturbations.

Contents

List of abbreviations

AUC Area under the curve.

DCA deep count autoencoder network.

DPT Diffusion pseudotime.

GRISLI Gene Regulation Inference for Single-cell with LInear differential equations.

INF Interferon.

IPSC Induced pluripotent stem cells.

LOESS locally estimated scatterplot smoothing.

MGE Medial ganglionic eminence.

MSE Mean squared error.

NK Natrual killer.

ODE Ordinary differential equations.

PCA Principle component analysis.

RGC Radial glia cells.

RNA Ribonucleic acid.

SAE Scaled absolute error.

t-SNE t-distributed stochastic neighbor embedding.

TCR T-cell receptor.

TSCAN Tools for Single Cell Analysis.

UMAP Uniform Manifold Approximation and Projection.

List of Figures

List of Tables

Acknowledgments

The work presented in this thesis was made possible by the generous funding from my supervisor Claudia. I would first like to express my gratitude to Claudia for taking me into the lab and giving me the opportunity to work on this project even if it deviates away from the main focus of the lab. Thank you for all the help throughout the course of my studies. I would also like to thank members of my supervisor committee Dr. Majewski and Dr. Lemieux for their continued guidance throughout the years. I also like to express gratitude to all the wonderful people with whom I had the pleasure of working with: Steven, Nick, Yang, Samantha, Selin and Marie. I would also like to thank Paul Francois for his help and his enthusiasm for helping me with this project. Last but not least, I would like to express my thanks to my family and friends for supporting me in these times.

Contribution of Authors

Claudia L. Kleinman supervised this project, and the data used in this project is made available by the analysis done by Alexis Blanchette-Cohen and Selin Jessa [20]. The selection of cell populations to form the pseudo-time trajectory was made possible by the help of Selin Jessa and Claudia Kleinman. Claudia Kleinman also led the interpretation of the markers found by the random forest method, in addition to the experimental design and data analysis throughout this study.

1 Introduction

1.1 Background

Understanding the molecular characteristics and underlying regulatory mechanisms of cell fate is a central goal of modern biology. Cellular differentiation is an important process where progenitor cells progressively develop into mature cells with specialized functions. Disruption of these processes has deleterious effects on phenotype, which frequently underlies disease. In complex tissues, such as the brain, studying developmental processes in *vivo* is challenging, as cell populations and their transcriptional profiles shift dramatically over time.

Recent technological advances have allowed the study of gene expression at single-cell resolution [50], which provides unprecedented opportunities to study differentiation processes for complex tissues *invivo* [11] [40] [44]. Single-cell data addresses two key limitations of standard bulk level RNA sequencing: discovering cell classes in a population and tracing the development of each class. The high resolution of single-cell sequencing technology provides a platform for detailing the cell classes present. Subsequent differential expression analyses can identify the gene signatures defining each class, hence providing insight to the genetic programs driving cellular differentiation [32] [30] [36]. However, given the scale of these datasets and the novelty of the analytical methods, novel methodology to infer gene expression profiles and their dynamics over time is needed.

In this study, I focus on developing computation methods for understanding cellular differentiation. The thesis will be presented in 2 parts. First, I focus on characterizing molecular signatures of cell populations in the developing brain and present a method for identifying their most discriminant gene marker combinations. Second, I turn my focus to modeling gene dynamics across differentiation, where changes in transcriptomic profiles form a continuous trajectory. I present a framework to predict the expression dynamics of cellular states, as well as the cellular response to perturbation.

1.2 Part I: Molecular signatures of cell populations in the developing brain

In single cell sequencing experiments, identifying the key differences between cell populations is an important step in understanding the structure of the data. This process is often facilitated by identifying the gene markers unique to each population of cells in the data. Differential expression analysis is a common approach for identifying those gene markers [55] [20]. In this approach, statistical tests are used to decide whether, for a given gene, an observed difference in expression between cell population is significant. Each gene is tested independently and ranked by fold change and p-value. However, the limitation of this approach is that each gene is tested individually, without considering the combinatorial effect of genes. However, it is often the concurrent expression of genes that labels cell identity. For example, in immune cells, T-cell populations are marked by surface T-cell receptors (TCRs). Within the T-cell population, surface markers such as CD4 and CD8 delineate subtypes like T-helpers and cytotoxic T-cells, while the absence of these markers defines double-negative T cells. Here, cell identity is marked by the presence or absence of the expression of certain marker genes, which also labels their cellular function [22] [49]. This identification scheme shares the same logic that informs the use of decision trees in machine learning.

In the decision tree algorithm, the tree is composed of a set of decision nodes that are represented by logical statements. A toy example of a decision tree representing the T-cell classification process is shown in figure 1. The root of the tree (topmost node) is the first query on the data, or decision, which in this case is whether the cell expresses TCR. Depending on the outcome, subsequent nodes are assessed sequentially until a final decision is reached. The decision tree algorithm is a highly interpretable and simple method for classifying cells that takes combinations of features, in this case genes, into account to classify different cell populations. This algorithm is designed to find the most informative features or genes, in a purely data driven matter, to classify the different cell populations in

the data.

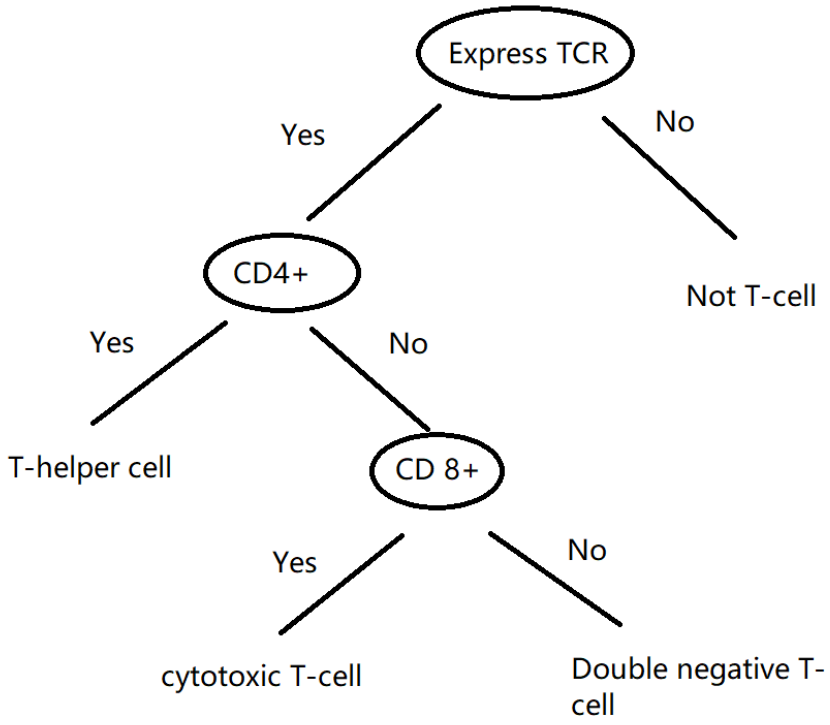


Figure 1: A schematic of a decision tree representing T-cell classification

The nature of single cell data is noisy and prone to dropouts, which can impose potential problems with the decision tree algorithm [24]. The nature of the decision tree algorithm is constructed based on sequential queries on the data. In the toy example, the first query is on the expression of TCR. However, if the expression of TCR is not detected due to sampling noise, an incorrect decision would be made. A more robust approach is to construct multiple trees, hence a forest, to make the predictions based on weighted majority vote. This describes the random forest algorithm, where multiple different decision trees are constructed in the process. A toy example of a random forest approach for classifying cells is shown in figure 2. In this example, three decision trees are used, and the weight of each tree is assumed to be the same. In this scenario, for a cell that expresses CD8 but not TCR due to dropout, two of the trees forming the majority vote will still correctly predict it to be a cytotoxic

T-cell. It has been proven that by using a collection of different trees, the variance of the predictions is decreased, thus making it a more robust method. The random forest algorithm has been used for cell annotation [2] as well as clustering [31] in single cell analysis, and showed promising performance in both cases. While clustering approaches focus on grouping of cell populations in an unsupervised manner, cell annotation approaches emphasize the classification of cell populations by training the model using labeled data, which does not require the interpretation of the features/genes used by the model. In marker selection, the emphasis is on the identification of genes/features that characterize each cell population, where random forest methods have also been used for this purpose [1] [4]. However, most of these approaches combine random forest with additional gene selections steps, reducing the interpretability of the markers found by these algorithms. Thus, for this project, I will be applying a random forest approach, without the addition of other methods, to identify gene markers in single cell data.

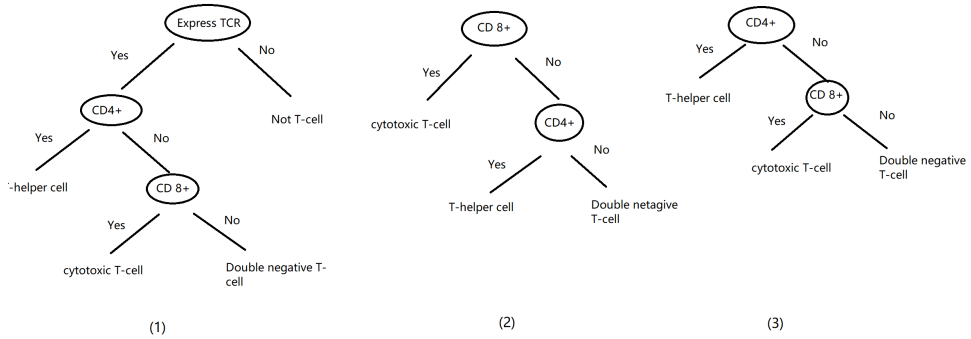


Figure 2: A schematic of a random forest classifier representing T-cell classification

1.3 Part II: Modeling cellular dynamics during differentiation

1.3.1 Overview

The characterization of molecular profiles described so far, through the identification of gene sets that discriminate cell populations in a sample, is not sufficient to capture the most important aspects of cellular differentiation. Cellular differentiation is a dynamic process where gene expression profiles change over time, in a continuous way. We can model this dynamic system as a function, $X(t)$, that predicts how the cell's transcriptomic profile changes throughout the course of differentiation. This model aims to predict $X(t)$ from an initial condition $X(0)$ to learn how gene expression changes over time. In addition, the model aims to predict the change in transcriptomic profiles as a result of gene perturbations, such as a gene deletion. To perform these two tasks, I require datasets that contain information on the time evolution dynamics for the differentiation process.

Single cell data offers a convenient platform for observing this dynamical process. In particular, the change in transcriptomic profile during differentiation can be inferred using pseudo-time trajectory algorithms [44], giving us the information on $X(t)$. In addition, the rate of change in the cell's transcriptome can be estimated via the RNA velocity algorithm [29], giving us the information on $\frac{dX}{dt}$. Although both pseudo-time trajectory algorithms and RNA velocity can extract the dynamics information from single cell data, neither are generative models that can predict on unobserved data, such as forecasting the future transcriptomic states of the cells or the effect of perturbations. Although pseudo-time inference algorithms and RNA velocity are not generative models, models can be built upon the information provided by these algorithms. In this project, I will build my model upon the information provided by pseudo-time and RNA velocity.

1.3.2 Inferring dynamics information from single cell data

Pseudo-time inference

Cell trajectory inference encompasses methods that reconstruct the dynamical information by connecting the discrete 'snapshots' in single cell data into continuous trajectories. Generally, gene expression profiles of cells are compared, and cells are ordered along a time dimension, representing a continuous change in transcriptomic profiles through the course of differentiation. A variety of methods for achieving this aim have already been proposed based on a range of algorithmic approaches. For instance, Monocle uses reverse graph embedding to learn a principal graph that represents this trajectory [52]. In a different approach, diffusion pseudotime (DPT) measures transitions between cells using diffusion-like random walks [16], and TSCAN (Tools for Single Cell Analysis) uses a cluster-based minimum spanning tree approach to order cells along pseudo-time [56]. Studies by [43] and [44] have benchmarked the accuracy and scalability of over 50 pseudo-time inference methods. Among them, Monocle [52] is one of the top ranked trajectory inference methods.

One common theme among trajectory inference methods is that cells are placed sequentially in pseudo-time to yield the information of how the transcriptome changes over time. The limitation of this approach is that pseudo-time inference algorithms sorts cells along the time axis, which only generates one trajectory for each lineage differentiation process. The differences between individual cells are generally smoothed out along the trajectory. In contrast, RNA velocity respects the small differences between nearby cells, where the velocity vector $\frac{dX}{dt}$ is inferred for each individual cell [29] [8] [9].

RNA velocity

In a study by [29], G La Manno et.al present an RNA velocity algorithm that is able to infer the rate of transcription for each gene in each cell for single cell RNA seq data. In their framework, the velocity is estimated based on the assumption that partially processed RNA

molecules in a cell reflect the genes that are very recently transcribed, while completely processed molecules reflect the genes that were transcribed earlier in time. This led the authors to examine the relationship between spliced and unspliced RNA profiles to provide dynamic information on the cellular states. In this framework, the transcriptional dynamics are described by the following equation:

$$\begin{aligned} \frac{du(t)}{dt} &= a^{(k)}(t) - \beta u(t) \\ \frac{ds(t)}{dt} &= \beta u(t) - \gamma s(t) \end{aligned} \tag{1}$$

where u and s represent the unspliced and spliced RNA profiles of a gene, respectively. The β and γ terms represents the splicing rate and degradation rate, respectively, while the $a^{(k)}(t)$ term represents the synthesis rate of the unspliced transcript, which varies as a function of time. The RNA velocity of a cell state is defined in this study as the rate of change of the spliced RNA concentrations $\frac{ds(t)}{dt}$.

The RNA velocity $\frac{ds_i}{dt}$ for a particular cell i can be computed as $\beta u_i - \gamma s_i$. Since both s_i and u_i are measured quantities for each cell i, only the parameters β and γ need to be estimated. To do this, the authors make use of the fact that 1) genes are turned on and off for different cellular processes, and these genes are captured within the single cell RNA seq dataset, and 2) at steady state $\frac{ds}{dt} = 0$ and $\frac{du}{dt} = 0$. By setting the derivatives to zero and rearranging the equations in equation 1, one can obtain the expression $\frac{\beta}{\gamma} = \frac{s_{ss}}{u_{ss}}$, where s_{ss} and u_{ss} are the concentrations of the spliced and unspliced counts at steady state. A schematic of the expected graph of plotting u against s is shown in figure 3. By running linear regression to fit the equation $s_{ss} = \frac{\beta}{\gamma} * u_{ss}$, one can obtain the ratio between β and γ from the slope of the line. To perform the regression task, the data points or cells that satisfy the assumed steady state must be selected within the single cell RNA seq dataset. Using the assumption above where the single cell dataset captures both the on and off states of a gene, the regression task can be done by selecting cells where u and s are maximized or when u

and s are both zero. When s and u are zero, the gene is completely off, thus satisfying the assumption $\frac{ds}{dt} = 0$ and $\frac{du}{dt} = 0$. These two assumptions are also satisfied when s and u both reach their maximum value, as the derivative is zero for any given function at its maximum or minimum points. Here, the maximum is approximated by using the maximum observed values in the data. A graphical representation of fitting the regression line to obtain the ratio $\frac{\beta}{\gamma}$ is shown in figure 3. Note that for genes that do not reach their theoretical maximum in the single cell dataset or that are not expressed, the estimations may not be accurate. The velocity for cell i can be computed using $\frac{\hat{ds_i}}{dt} = u_i - \frac{\gamma}{\beta} * s_i$. Here $\frac{\hat{ds_i}}{dt} = \frac{ds_i}{dt} * \frac{1}{\beta}$. Because β and γ cannot be decoupled from the regression model, the estimated velocity is scaled by a constant factor from the original equation.

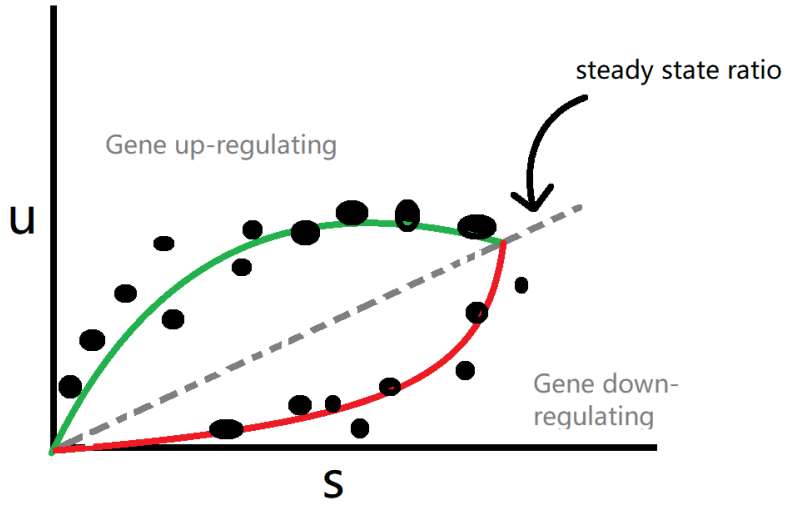


Figure 3: Expected behaviour when plotting the spliced s against the unspliced u transcripts counts for a particular gene in single cell RNA seq data. The dotted line represents the steady state ratio obtained using linear regression. Points above the steady state line represents up-regulation of the gene while the points below represents down-regulation.

In the RNA velocity method, the term $a^{(k)}(t)$ is never explicitly modeled, as it is conveniently not required in the velocity estimate approach. Here, the $a^{(k)}(t)$ term represents the

regulatory relationship between different genes. Because the regulatory relationship between genes is not modeled, even though the RNA velocity method is able to estimate the velocity for each gene in each cell, it does not have the ability to predict the long term time evolution of the expression profile of a gene $s(t)$ over time. It also cannot predict how a perturbation event, such as a gene deletion, can alter the expression profile of other genes. Although the gene regulatory relationship is not directly modeled in their method, RNA velocity data has been used by several methods to infer the regulatory relationship between genes [51] [35] [48].

1.3.3 Modeling of gene regulatory dynamics from single cell data

Causal network inference

Transcriptional states of cells during differentiation are the result of complex regulatory interactions, including activating and repressive interactions between genes and their protein products. These regulatory interactions can be thought of as a network, and much work has been done over the past years to leverage developments from the field of network theory to model biological gene regulatory relationships [5].

In network based approaches, each gene is represented by a node in the network, and the directed edges represent the causal regulatory relationship between genes. For a particular node i, all other nodes with an edge pointing towards i are considered incoming neighbours of i. The expression profile of i (x_i) in the next time step can be predicted as $x_i(t+1) = f_i(X_{i,in})$ where $X_{i,in}$ is the expression levels of the incoming neighbours to gene i, and $f_i(X_{i,in})$ is the function representing how the gene is controlled by its neighbours. Iterative updates can be applied to each of the nodes to predict how all the genes change over time, which is represented by $X(t)$.

To use this approach for predict gene dynamics, one must first accurately infer the network connections from the data, or the X_in for each gene, as well as identify the exact

mathematical relationship, $f_i(X_in)$, between genes and their incoming neighbours.

For instance, *Scribe* [51] is a method that uses the mutual information between the velocity vector and the gene expression profile to infer network connectivity. To determine whether a directed edge exists from two genes a to b, the algorithm computes the mutual information between $a(t)$ and $b(t + \delta t)$, which measure the amount of information that can be obtained on future values of b given the current value of a. In this case, the future value of b is obtained via linear extrapolation $b(t+\delta t) = b(t) + \frac{db}{dt} * \delta t$. Alternatively, another method GRISLI (Gene Regulation Inference for Single-cell with LInear differential equations) uses the linear correlation between the expression profile and the velocity for network inference [35]. Overall, a variety of methods exist to infer networks using the dynamics information captured in single cell expression data, and RNA velocity provides a convenient platform for such inference.

Predicting perturbation response based on gene regulatory networks

Causal inference algorithms primarily focus on identifying edges between genes to construct the gene regulatory network, while the exact regulatory relationships between genes, $f_i(X_{i,in})$, are not always directly modeled. However, several methods exist that model the gene regulatory dynamics based on networks obtained from causal inference methods. In the simplest case, Boolean networks can be use to model the gene regulatory dynamics [3] [7] [34]. In a Boolean network model, the expression of each gene is binarized to either 'on' or 'off', and each gene is represented by a node in the graph. Edges represent the causal relationship between genes, where the edges point from the regulators, such as transcription factors, to their targets. The regulation of each gene is controlled by a Boolean function of its incoming neighbours. The time evolution of each gene i over a time frame of T, $x_i(t+T)$, can be predicted by iteratively applying the update function $x_i(t + 1) = f(X_{i,in}(t))$, where $X_{i,in}(t)$ is the state of the incoming neighbours to gene i at the current time step, and

$x_i(t+1)$ is the state of gene i at the next time step. The time evolution of the entire network can be computed by simultaneously applying the update function to each node. In addition to predicting the time evolution of the system, this method can also be used to predict the perturbation response, such as the effect of gene deletions, by simply setting one of the nodes permanently to the 'off' state while applying time evolution to the rest of the nodes.

As an example, the algorithm $IQCELL$ [17] is a Boolean network based approach that infers gene regulatory networks, as well as the regulatory functions, from pseudo-time trajectories constructed from single cell data. They use the learned network to predict cell transition dynamics as well as perturbation response, and were able to accurately recover over 75% of the causal gene connections in a 39 gene system. However, a small gene set was selected for the model based on prior knowledge of genes involved in the T-cell differentiation process studied. Therefore, applying this method to larger gene sets, and in cases where the genes responsible for the process of interest are unknown, may be challenging. Generally, such network based approaches are limited in their ability to scale to gene sets larger than 200 [47] [46].

A differential equation based approach to model differentiation

In the network based approaches described above, we see that the gene dynamics can be predicted by iteratively applying the update function $x_i(t+1) = f(X_{i,in}(t))$. If we consider infinitesimal changes in time in a continuous manner, the update function can be re-written in the form of an ordinary differential equation (ODE): $x_i(t+\Delta t) = x_i(t) + \frac{dx_i}{dt} * \Delta t$, where $\frac{dx_i}{dt}$ is now a function of its incoming neighbours (i.e., $\frac{dx_i}{dt} = f(X_{i,in})$). This concept is studied in detail by [23], where they demonstrate the mathematical formulation of mapping different types of graphs to ODEs.

A variety of ODE-based approaches have been proposed to model gene expression dynamics, but most still suffer from scaling to larger gene sets or imposing assumptions that

are often violated in gene regulation [41] [21] [37]. However, these limitations have been overcome by a recent method, PRESCIENT, which models the dynamics of single cell data using stochastic ODEs [14]. This method is fast to compute and can incorporate information on a large number of features/genes simultaneously, while imposing limited assumptions [14]. In this approach, the cell state dynamics are modeled as a diffusion process that is represented by ODEs, which in turn are approximated by neural networks. The model is trained on single cell data collected from different time points and has shown to be able to accurately predict the change in population structure at held-out time points, as well as the change in fate bias as a result of gene perturbations [14]. One key difference between PRESCIENT and previous ODE-based methods is that PRESCIENT takes advantage of the high resolution of single cell data by using neural networks as a universal function approximator, and therefore does not impose strict restrictions on what type of functions can be present in the ODEs [14]. This allows PRESCIENT to impose fewer assumptions on the dynamical behaviours of gene regulation.

The limitation of PRESCIENT's approach is that the model is trained on datasets from single cell sequencing sampled at different time points. This requirement of samples separated in real-time restricts which datasets can be used. Moreover, this method has only shown the ability to predict the effect of gene perturbations on the cell's differentiation trajectory, rather than the whole transcriptome. Whether this method can be generalized to predict the overall changes in the cell's transcriptome remains unknown.

Predicting perturbation response based on variational auto-encoders

In contrast to the above methods that mainly emphasize modeling time evolution of gene expression and predict perturbation response as an additional extension, a recent variational auto-encoder (VAE) method [33] uses neural networks to embed the high dimensional single cell data into low dimensions and specifically addresses the perturbation prediction problem.

In this study, the authors use the VAE method to predict the perturbation response in the gut after Salmonella or Heligmosomoides polygyrus (H. poly) infections. In their setup, they use a single cell dataset containing eight different cell types under four different conditions. In the experiment, they held out a cell type in the condition of active infection during training, and evaluated the model's performance on the held out data. They showed that their method was able to predict the effect of perturbation on held out data. In addition, the authors evaluated their method on data from immune cells during interferon beta (INF-β) response. In this case, the INF-β induced natural killer (NK) cells were held out during the training and evaluated during testing. The authors demonstrated that their method was able to performed well in both cases.

Following this line of work, several VAE approaches have been used to predict the perturbation response from single cell data [26] [45] [27]. The aim of these approaches is to predict the unsampled perturbation response of a cell type using the sampled perturbation response of other cells as a reference. Although these VAE methods have been shown to perform well in predicting the perturbation response of cells, they are limited in requiring data of the perturbation of interest to make such predictions [33] [53] [15]. For example, in the study by [33], when predicting the INF-β response in NK cells, data corresponding to INF-β response in other immune cell types was needed, along with data from the unperturbed states of the NK cells and other immune cell types. Therefore, these methods require a unique set of data generated under specific conditions to generate their predictions, such as requiring the perturbation of interest to be included in the data. Thus, VAE based methods are not particularly useful for investigating perturbations that are absent from the training data.

1.3.4 Aim: to build a framework for modeling gene dynamics and predicting perturbation response using machine learning approaches

Both ODE-based methods, such as PRESCIENT [14], and network-based methods, such as $IQCELL$ [17], are able to predict gene regulatory dynamics from single cell data, in addition to predicting perturbation response without perturbation data in training. This ensures that the data required by the models does not scale linearly with respect to the number of perturbations of interest. However, these two methods have their respective limitations. $IQCELL$ does not scale well as the number of genes in the network increases, while PRESCIENT requires data collected from multiple time points and is subjected to sampling bias in the population structure of the data.

Here, I develop a novel method for modeling gene dynamics from single cell data by building upon and combining the above methods, along with pseudo-time inference and RNA velocity, to overcome their individual limitations. In this project, I use an ODE-based approach to model gene regulatory dynamics in differentiation. However, unlike PRESCIENT, I choose to directly model how the cell states change over time using inferred dynamics from the data, obtained by pseudo-time inference and RNA velocity, rather than using the population structure of time series data. I use dynamics information inferred from single cell data such that multiple real time samples are not required. In my method, the regulatory process of each gene is to be described by an equation similar to that of $IQCELL$. Each gene i will be described by a differentiation $\frac{dx_i}{dt} = f(X_{in})$, where X_{in} represents the set of regulators of gene i. To model this process, I use the Neural ODE approach developed by [39], which directly fits an ODE to model the data. The details of my approach will be described in the Methods section, but I will first introduce the Neural ODE method and how it will be used in this project.

Using Neural ODE for forecasting time series data

For this project, I model the cellular expression dynamics with an ODE-based approach. To do this, I use the neural ODE method introduced by [39]. This method uses neural networks to represent a system of differential equations to describe the underlying dynamics of a system. Here, I focus on the study done by [54] that emphasizes modeling of time series data. The objective of this method is to perform time series forecasting by modeling the system as ODEs. This method uses a neural network to model the dynamics as a differential equation in the form of $\frac{dX}{dt} = f(X)$, where X is a vector representing all the variables in the system, and $f(X)$ is represented by a neural network. The time series forecast of the system $X(t)$ can be predicted by performing numeric integration on $f(X)$.

2 Materials and Methods

2.1 Part I: Marker selection using a random forest approach

2.1.1 Overview

The objective of this approach is to identify the most discriminant combination of genes for a particular cell population in single cell data. Here, I assume that the cells have already been assigned into distinct populations, which is frequently done using a clustering algorithm. The approach introduced here takes expression profiles of cells and their cluster label as inputs, and outputs the most discriminant gene markers for each cluster. A random forest algorithm is used to construct decision trees for the classification of cells into clusters based on their expression profiles. Following the classification, each decision tree using a different combination of genes is evaluated based on how accurately it assigns the cells to its corresponding class. The genes used by the top performing trees are selected as cluster markers.

2.1.2 Method details

Data pre-processing

This method uses the random forest algorithm to identify gene markers in single cell data. The random forest algorithm is a machine learning approach where a model is first trained and then its performance is evaluated. The data used to fit the model parameters is considered the training data, while the data used to evaluate the model is the test data. In this method, the cells in a single cell sequencing dataset are split randomly into a training set and a test set. The ratio between the amount of cells in the training data versus the test data is $75 : 25$. For a dataset with N number of clusters, the method uses a one-vs-all approach, where each decision tree classifier is designed to predict whether a cells belongs to a particular cluster j or not rather than predicting which cluster the cell belongs to in

N clusters. Thus, each cluster has a set of trees dedicated to its identity. When training the decision trees for a particular cluster, the training data is re-sampled such that there is an equal number of cells inside the cluster and outside the cluster. Meanwhile, for the cells outside the cluster, the cells are sampled such that all clusters have an equal representation of cells.

Training the model

During training, the algorithm constructs a set of decision trees to classify cells into each cluster using only a small number of genes for each tree. A common approach to construct decision trees is to consecutively add nodes to the tree by selecting features, in this case genes, that maximize the model improvement at each step. As an example, figure 4 shows the construction process. For each gene i in a gene set, the algorithm evaluates the model's performance of adding i into the existing tree. Among all the possibilities, this greedy approach will select the one that gives the maximum improvement at each step. The computation complexity to construct trees using this approach is $O(k * n * D)$, where k is the number of genes to be included in the tree, n is the size of the gene set to choose from, and D is the number of samples, in this case cells, in the training data. This approach only searches a subset of the possible arrangements, making the computation feasible. For a given gene set and training dataset, this deterministic approach will always construct the same tree, which is not ideal for searching through different trees and finding different gene markers. To accomplish this, a randomization step is added. To construct each tree, a subset of genes (default of 100) is randomly chosen from the set of all possible genes, and the decision tree construction algorithm is applied to the subset. This process is repeated until the total number of desired trees has been constructed.

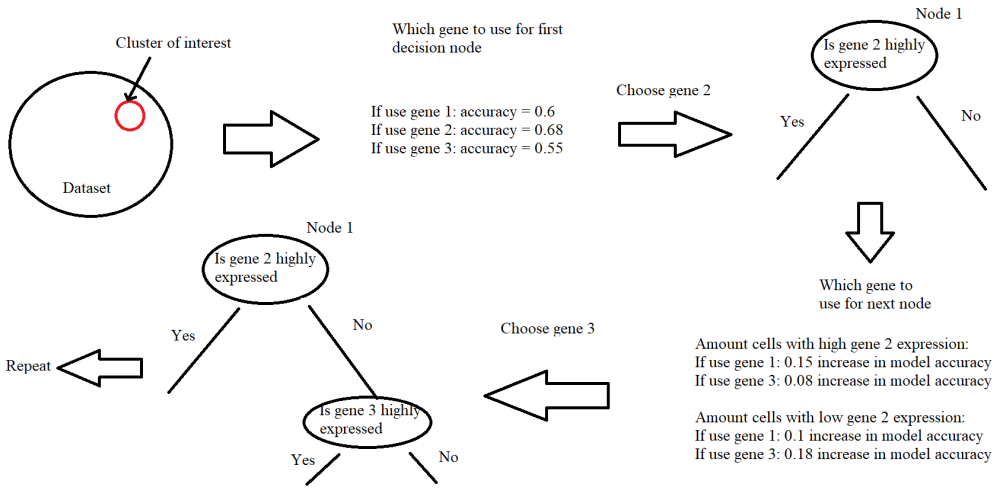


Figure 4: A schematic of a decision tree construction algorithm

Identifying marker genes

After a set of decision trees are constructed for labeling cells to their corresponding clusters, each tree is evaluated and the genes in the top ranked trees are used as the gene markers. Here, the trees are ranked based on the error rate of the trees. As each tree is used to classify if a cell belongs to the cluster or not, the genes used by the top ranked trees for the identification of a particular cluster j are used as the gene markers for that cluster. In the decision tree construction algorithm, each gene is allowed to be either upregulated or downregulated for cells inside the cluster versus outside the cluster, thus allowing identification of negative markers with this method.

2.2 Part II: Modeling differentiation dynamics

2.2.1 Overview

To model the gene regulatory network from single cell expression data, a system of ordinary differential equations (ODE) is used in the form of $\frac{dX}{dt} = f(X)$, where $X = [x_1, ...x_k]$ represents the gene expression levels (also referred to as cell states) of the k number of genes, x_1 to x_k, to be modeled. Here, I assume the dynamics of all cells in the data are described by a single system of ODEs regardless of their cell types. The cell states of the differentiated cells of varying cell types are assumed to be different attractors, defined as having cell states X that give rise to a zero velocity vector (i.e., $\frac{dX}{dt} = 0$), of the ODE system. The differentiation process is modeled as the time evolution of cell states from an initial condition $X(t = 0)$, which is often sampled from the progenitor population. To predict the time evolution process starting at an initial state $X(0)$, numeric integration is applied on the ODEs via $X(t) = \int_{t_0}^{t} \frac{dX}{dt}$. The function $f(X)$ is approximated by a neural network. To train the model, I use two different types of data: 1) pseudo-time trajectory data (See 2.2.2 for details), and 2) RNA velocity data (See 2.3 for details). In the first case, the inferred pseudo-time trajectory is used to train the model, where the data resembles consecutively measured cell states across time represented by $X(t)$. In the second case, RNA velocity information $\frac{dX}{dt}$ is inferred from single cell data and used directly to train the model.

When using pseudo-time data, the model tries to find a system of ODEs, $\frac{dX}{dt} = f(X)$, such that when integrating the ODEs, $\int_{t_0}^{t} f(X)dt$, the model can reconstruct the training data trajectory $X(t)$. When using RNA velocity data, since the velocity information for each gene in each cell $\frac{dX}{dt}$ is given in training, the differential equation is directly modeled using neural network regression.

2.2.2 Predicting gene expression dynamics using model train on Pseudo-time data

Data collection

Single cell data collected from the mouse brain at three developmental time points (E12, E15, and P0) with annotated cell type labels reported in Jessa *et al*. [20] was used. From this dataset, we extracted cells belonging to the following cell types of the interneuron lineage: the ventral radial glial cells (RGC), medial ganglionic eminence (MGE) derived inhibitory neurons, inhibitory progenitors, and mature inhibitory neurons. The top 995 most variable genes within the selected cells were used by Monocle [52] to order the cells and yield a pseudo-time trajectory. To remove confounding signals from cell cycle, cells were scored based on their cell cycle activity (Figure 8), the $G2M$ score, which is defined as the GSVA (gene set variation analysis) score [19] of the "HALLMARK_G2M_checkpoint" gene set [25]. Cells that had a $G2M$ score above 0.4 were labeled as cycling and removed.

The retained cells were used as input for the Monocle algorithm [52], which assigns each cell in the data matrix a pseudo-time value representing its inferred time position along the trajectory. The inferred trajectory, which includes multiple branch points, is shown in figure 9. To only focus on linear trajectories without branching, the longest path along the trajectory was selected for downstream processing.

In addition to the single cell data collected from time points E12, E15, and P0, we also use a dataset sampled at different time points (E13 and E16) to evaluate the model performance. The data is processed using the same conditions to obtain the pseudo-time trajectory. The model is tested on how well it predicts the trajectory generated in this test dataset.

Smoothing pseudo-time data

The pseudo-time inference algorithm Monocle is used to assign each cell a time value along the differentiation trajectory. Here, the time assigned to each cell represents its maturation

level and the time separation between consecutive data points varies across the trajectory, thus representing irregularly sampled time series data. By aligning the cells sequentially along the time axis, the continuous changes in expression of each gene can be obtained by fitting a smooth line across time. As single cell data is noisy and prone to dropouts, locally estimated scatterplot smoothing ($LOESS$) is used to smooth the data. $LOESS$ smooths the data by fitting a linear regression model locally on the data, where a weight function is applied to control the contribution of each data point. A Gaussian weight function is used when applying $LOESS$. Although the original data before smoothing is considered irregularly sampled time series data, data points can be re-sampled on the smoothed trajectory to produce evenly sampled training data in which the time separation between consecutive measurements is a constant. After $LOESS$ smoothing was applied to the data, 3000 points were sampled evenly across the pseudo-time trajectory to train the model.

Model training

During the training process, initial conditions $X(i)$ are randomly sampled from the smoothed trajectory data. Then, starting from $X(i)$, the model is asked to consecutively predict future data points $X(t > i)$ up to a certain length N represented by $X(i+1\hat{...}N+i)$. The error/loss function is computed as $L = \|X(i+1,\hat{\ldots}, N+i) - X(1+i...N+i)\|_2/N$, which is the mean squared error between the model predictions and the data. The prediction $X(i+1...N+i)$ is generated via numeric integration of the ODE learned by the neural network. The integration method used in this case is Euler's method, where the future states are iteratively computed using the recursive equation $X(n+1) = X(n) + \frac{dX}{dt} \times \Delta t$. Gradient decent using the *Adam* optimizer is used to train the model with a learning rate of 0.001.

Model evaluation

The performance of the model is evaluated by hiding parts of the data during training, and assessing the model's ability to interpolate and extrapolate the gene dynamics of the hidden

data. The sequential change in the transcriptomic profile, for K cells, during differentiation $X(0)...X(K)$ is created by sorting the cells along the time axis using the values obtained from Monocle [52]. A continuous segment of this sequence is taken as the hideout data represented by $X(i)...X(i+N)$ where $0 \leq i \leq K-N$, and N taken as 8% of the total number of cells in the trajectory. In the extrapolation case, the hideout data is taken from $X(0)...X(N)$ or $X(K-N)...X(K)$, such that the model needs to predict outside of its observed range within the training data. In the interpolation case, the hideout data $X(i)...X(i+N)$ lies in the middle of the trajectory rather than at the endpoints, and is sampled such that $K-2N \leq i \leq N$.

In addition to evaluating the model on hideout data, the model is also evaluated on an independent test dataset containing cells from the same lineage, but collected at different time points. The test data undergoes similar data processing as the training data to construct the trajectory. In the evaluation process, the model uses $X(0)$ from the test data to predict the future states.

2.3 Predicting perturbation response using RNA velocity data

2.3.1 Overview

In pseudo-time trajectory reconstruction, cells are sorted along a time axis representing their progress during differentiation. This approach generates a separate trajectory for the differentiation of each cell type lineage. The data $X(t)$ generated in this process only captures the expectation value of how the transcriptomic profile changes over time during differentiation. The cell-to-cell variation, which is rich in information, is smoothed out during the process. In contrast, RNA velocity captures cell-to-cell variation by providing the velocity vector at the single cell level. The cell-to-cell deviations can be seen as the results of small perturbations applied to the cells, which provides a better estimate of the causal relationship between genes. RNA velocity estimates the velocity vector $\frac{dX}{dt}$ for each cell in the dataset, where the deviations between cells are retained. The velocity vector $\frac{dX}{dt}$ is then used to train the model by modeling the velocity as a function of the cell state: $\frac{dX}{dt} = f(X)$.

Prior to training the model, several data processing steps are used to impute and extract the relevant information from the single cell sequencing data. Starting from the read counts of the sequencing experiment, the *velocyto* [29] package is used to extract the spliced and unspliced transcript counts.Since single cell data is noisy and prone to dropouts, an imputation step is used to denoise the data. This imputation step is done using the deep count auto-encoder (DCA) method developed by [13]. After denoising, the normalized gene expression data for both spliced and unspliced transcripts is used as input for the *scvelo* [29] package for RNA velocity inference. An additional data re-sampling step is performed for data balancing. The balanced data is then used to train the model. In this section, I will describe the steps of this process sequentially, including 1) imputation using DCA, 2) RNA velocity inference with *scvelo*, 3) data balancing, 4) model construction, and 5) model evaluation.

Step 1: Data denoising using DCA

Single cell data is noisy and prone to dropouts. This negatively impacts both the velocity estimate by *scvelo* as well as the modeling process. To reduce the effect of noise, I used the DCA method developed by [13] to estimate the expected gene expression profile of each cell. DCA, which stands for deep count auto-encoder, is an imputation method that uses a deep learning approach by fitting a zero inflated negative binomial (ZINB) model to the data. The ZINB distribution is described by the following equation:

$$Pr(Z) = \pi + (1 - \pi)NB(Z); \text{if } Z = 0 \tag{2}$$

$$Pr(Z) = (1 - \pi)NB(Z); \text{if } Z \neq 0 \tag{3}$$

where $NB(Z)$ is the negative binomial distribution of the random variable Z, and π is the probability of dropout. The DCA method assumes that the data has a ZINB distribution, and uses a variational auto-encoder approach to estimate the dropout probability π, the expected expression value μ, and the variance θ for each gene conditioned on the input X. In other words, for each gene x_i in X, the neural network estimates the function $\pi_i, \mu_i, \theta_i = f_i(X)$. The parameters are estimated with maximum likelihood estimation by using the negative log likelihood of the ZINB distribution as the loss function. DCA is used on both the spliced and unspliced transcript counts, and the imputed normalized expression values (for both spliced and unspliced transcripts) are used for subsequent analysis.

Step 2: RNA velocity inference using *scvelo*

The imputed data is used to compute RNA velocity using the package *scvelo* [29]. The mathematics and details of this method are discussed in the Introduction in 1.3.2. The top 1000 genes based on expression level (detected with a minimum number of counts) and high dispersion were selected for downstream processing. After inferring the velocity information $\frac{dX}{dt}$, we can fit a regression model that predicts $\frac{dX}{dt}$ from X, where X is the imputed expression

profile of the cell. However, single cell sequencing datasets often have sampling bias that is dependent on the tissue sample, the age of the organism, and other technical factors. An unequal representation of different cell populations may lead to the model emphasizing the larger populations and underrepresenting less abundant cell populations.

Step 3: Data balancing

Prior to training the model, data balancing is performed to reduce the effect of sampling bias. In most single cell sequencing datasets, not all cell types are represented equally. In attempts to correct for sampling bias, the data is balanced such that each of the cell populations are represented equally. Here a re-sampling process is used to up-sample the under-represented populations such that all the populations are represented with equal proportions. The populations are often labeled by their cell types, but more refined structures are also allowed.

Step 4: Neural network modeling using RNA velocity data

After balancing the data, neural networks were used to model the function $\frac{dX}{dt} = f(X)$, where the neural network model is trained to predict the velocity vector ($\frac{dX}{dt}$) of a cell based on its expression profile (X). In the framework of RNA velocity, the velocity is estimated based on the spliced and unspliced RNA profiles, where the system of differential equations are described in 1.3.2. In those equations, u and s represent the unspliced and spliced RNA profiles of a gene, respectively. The β and γ terms represents the splicing rate and degradation rate, respectively, while the $a^{(k)}(t)$ term represents the rate of transcription for the unspliced RNA. The $a^{(k)}(t)$ term is never explicitly modeled in the RNA velocity algorithm as it is a complex process, and it is not required for the velocity estimation process. However, I will be modeling this gene regulation, where the synthesis rate of the unspliced transcripts is assumed to be controlled by the cell's transcriptomic state X. Here, the expression profile X is equivalent to the spliced RNA counts represented by S. By

rearranging the equations in equation 1, and substituting x_i for s, we can rewrite them in the following form:

$$\frac{dx_i(t)}{dt} = a_i^{(k)}(t) - \frac{du_i(t)}{dt} - \gamma x_i(t) \tag{4}$$

For simplicity, I assume that the term $a^{(k)}(t) - \frac{du(t)}{dt} - \gamma x_i(t)$ can be approximated by a function of the current cell state X:

$$\frac{dx_i(t)}{dt} \approx F(X(t)) \tag{5}$$

Generalizing this across all genes, I can write the system as:

$$\frac{dX(t)}{dt} \approx \hat{F}(X(t)) \tag{6}$$

Where $\hat{F}(X(t))$ is approximated by a neural network. The system dynamics $X(t)$ can then be predicted by applying numeric integration to $\frac{dX(t)}{dt}$.

Model training and neural network structure

To model the system dynamics of the cell state $X(t)$ consisting of N genes, I used neural networks that take the expression levels of those N genes as input to predict $\frac{dX}{dt}$ for that gene set. The mean squared error (MSE) is used as the loss function to train the regression model. The model is trained via gradient decent using the *Adam* optimizer with a learning rate of 0.001. During the training process, the data is split into a training set and a validation set. The validation data consists of a random sample of 10% of the data prior to data balancing. This validation data is used for monitoring over-fitting as well as hyper-parameter tuning.

In this model, the regulation of each gene i is modeled as $\frac{dx_i}{dt} = f_i(X)$, where each differential equation is estimated by an independent multi-input, single-output neural network. Each neural network takes X as input and predicts $\frac{dx_i}{dt}$ for gene i. By doing so, parameters are not shared between the $f_i(X)$ functions, as each function is estimated separately such

that parameters do not favour genes with a higher variance in their velocities. In addition, $L1$ regularization can be used to limit the number of inputs to each network.

L1 Regularization and causal inference

In single cell data, the number of detected genes in the dataset often outnumbers the amount of cells sequenced. It is often more favourable to train machine learning models with more samples than features, as large feature sizes can lead to over-fitting. Here, I used $L1$ regularization to reduce the number of input features to each neural network. For each gene i, the input features $X_{i,in}$ are selected based on how informative they are for the prediction of $\frac{dx_i}{dt}$. The input feature set is unique for each gene. This feature selection step can be interpreted as a causal inference process, and it is used in later steps to improve model performance is described in 3.2.3.

The graph interpretation of the model and relations to causal inference

To model gene regulatory dynamics, I used neural networks to perform the regression task of learning the function mapping the cellular state X to its velocity vector $\frac{dX}{dt}$, where X and $\frac{dX}{dt}$ are vectors of length k, and k is the number of genes included in the model. When using pseudo-time data, training was performed with multi-input, multi-output neural networks. However, when using RNA velocity data, k individual neural neural networks were trained, with each taking X as input and returning $\frac{dx_i}{dt}$ for the gene i. The system of differential equations to be estimated by the neural networks can also be interpreted as a gene regulatory network graph, where the selected input genes X_i for a neural network of gene i can be interpreted as the expression level of gene i regulated by X_i. This mapping of X_i to $\frac{dx_i}{dt}$ is similar to the causal inference algorithm *Scribe* [51] (described in 1.3.3), in which the information between velocities and gene expression values are used to infer causal connections. Here, rather than using mutual information to measure the strength of the connections as performed by *Scribe*, the direct regulatory relationship is estimated from the

velocity data. The regulatory function for each gene or node i is represented in the form of $\frac{dx_i}{dt} = f(X_{in})$, with X_{in} representing the incoming edges to that node. A schematic of this process is shown in figure 5. One issue when training the regression model is that X_{in} can potentially include all of the genes in the data. To reduce the number of incoming edges to each node, $L1$ regularization, a technique able to reduce parameter weights to zero in regression tasks, is used to set certain edge weights to zero. In this regularization process, the regression task for each gene i is to learn the $\frac{dx_i}{dt} = f(W \odot X)$ where W are the edge weights of all incoming edges to gene i (including i itself), and $\odot$ represents element-wise multiplication. By applying $L1$ regularization on the edge weights, one can effectively eliminate weakly linked edges. The exact implementation is described as follows.

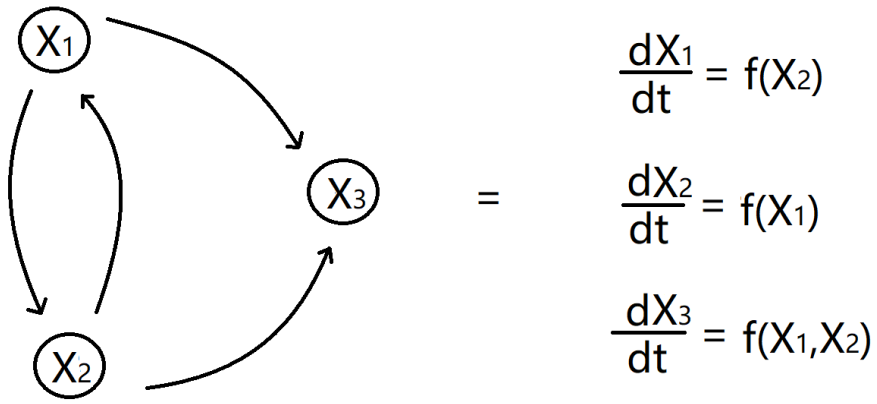


Figure 5: Interpreting the differential equations as a graph

Implementation of causal inference using neural networks via $L1$ regularization

For each neural network that computes the velocity for gene i, a mask vector W_i of length k is multiplied to the input X via element-wise multiplication represented as $\odot$. The penalization term of $\lambda * \|X \odot W_i\|_1$ is added to the loss function, with λ being a user defined hyper-parameter and $\|\|_1$ being the $L1$ norm. This penalization term aims to set elements in W to zero. This process can be interpreted as causal network inference, where

for each equation $\frac{dx_i}{dt} = f_i(X \odot W_i)$, the location of the non-zero elements in W_i represents a causal connection directed to gene i. Although $L1$ regularization has the tendency to set weights to zero, when optimizing the model using iterative methods such as gradient descent, the weights never reach zero due to the stochasticity of the optimization technique. However, weakly connected edges have the tendency to frequently fluctuate between positive and negative weights, and edge weights showing this trend are manually set to zero after a threshold of such fluctuations.

Boundary conditions

For this model, two boundary conditions needed to be applied. The first is that when integrating $\frac{dX}{dt}$, the solution must be non-negative, as $X(t)$ represents the concentration of gene transcripts and cannot take on negative values. Thus, the equation for each gene x_i in X is subjected to the boundary condition of $\frac{dx_i}{dt} = 0$ when $x_i = 0$. Secondly, the solution $X(t)$ must be bounded. This implies that $\frac{dx_i}{dt} = 0$ when x_i has reached its maximum expression level. However, x_i is generally not known a priori, so the empirical maximum observed in the training data is used as the theoretical maximum.

The implementation of these boundary conditions in the neural network is done by utilizing the unit step function $u(x)$ (also known as the heaviside function), where $u(x)$ takes the value of 0 when $x <= 0$ and 1 when $x > 0$. For the first boundary condition, we used $\frac{dX}{dt} = f(X) * u(X)$ to ensure the solution $X(t)$ is non-negative, with $f(X)$ being the output of the neural network. The second boundary condition is similarly applied by modeling $\frac{dX}{dt} = F(X) * u(X_{max} - X)$.

2.3.2 Model evaluation

Evaluation data

To evaluate the model's ability to predict on perturbation response, a published perturbation dataset from [42] was used. In this study, CRISPR interference (CRISPRi) was used

to reduce expression of selected genes in human iPSCs during induced neural differentiation. The cells were sequenced by single cell RNA seq to evaluate the effects of gene knockdowns on the transcriptome. To evaluate model performance in this study, I used the z-scores of differentially expressed genes resulting from the CRISPRi gene knockdowns reported in [42].

Evaluate model accuracy

To evaluate the model performance, I binarized the results of the test data, with each gene binarized as either up-regulated or down-regulated as a result of the gene deletion. Predicting the effect of gene deletions with this model consists of three steps: 1) choose a set of initial conditions $X(t = 0)$, 2) perform integration on the system with and without the perturbation, and 3) compare the expression profiles between the perturbed and non-perturbed final state of the system for the differential expression profile.

1) Prior to predicting the system dynamics, the appropriate initial conditions must be selected. Since the test data is collected from neurons, the initial conditions must be sampled from cell populations or clusters that resemble the test data as closely as possible. In this case, 100 random initial conditions were sampled from the granule cell clusters in the training data.

2) Starting from the sampled initial conditions, numeric integration is performed until the system reaches equilibrium. In this process, each initial condition is used to construct 2 trajectories: one with the perturbation, and one without. While the non-perturbed trajectory is obtained by integrating the system of equations $\frac{dX}{dt}$ normally, the perturbed trajectory is obtained by integrating the equations while permanently setting the values of the perturbed gene x_i and its velocity component $\frac{dx_i}{dt}$ to zero during the integration process.

3) The difference between the final states generated by the two different processes are computed and used to evaluate the model's prediction against the test data. For each gene, the difference between the final expression value in the perturbed trajectory and in the non-

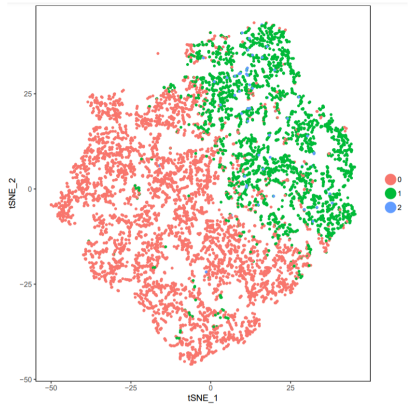
perturbed trajectory indicates whether the gene is up or down-regulated as a result of the perturbation. This is then binarized to a categorical outcome and compared to the test data to evaluate the model accuracy.

3 Results

3.1 Part I: Marker selection using random forest approach identifies key gene markers for neuron populations

In this study, I constructed a random forest approach to identify combination of gene markers. The random forest marker selection method ranks gene combinations based on how well they performed at classifying cells for each cluster. Figure 6 shows a sample output from the algorithm. Panel A shows the cluster assignment of the cells, and panel B shows the table of markers found by the algorithm for those clusters. In this table, the first two columns indicate whether the genes in the next two corresponding columns are up or down-regulated in the cluster. Each row represents the performance of a decision tree in identifying the cells inside the cluster. The false positive and false negative rates are shown for each tree, and the trees are ranked by $\frac{1}{e}$ (shown on the conf_index column in figure 6), where e is the error rate evaluated on balanced test data. Figure 6 panel C shows the expression levels of a positive and negative marker pair that distinguishes cluster 1. In this example, the markers for cluster 2 all have a very high score. However, this is due to cluster 2 having very few cells, which causes the algorithm to overfit.

A)



B)

X1.1	X2.1	X1	X2	false.positive	false.negative	conf_index	Cluster
+	+	Lhx2	Aldoc	0.2100911	0.2422641	4.098369	0
+	+	Lhx2	Lpar1	0.2040428	0.2416635	4.012316	0
+	+	Aldoc	Gas1	0.1925967	0.2523628	3.984891	0
+	+	Lpar1	Gas1	0.2225123	0.2408184	3.973138	0
+	+	Lhx2	Alcam	0.2414140	0.2183480	3.947802	0
+	+	Lhx2	Lix1	0.2385321	0.2545455	3.928571	0
+	+	Lhx2	Ppp2r2b	0.2547170	0.2190476	3.925926	0
+	+	Lhx2	Tfap2c	0.2503851	0.2155064	3.913200	0
+	+	Dmrta2	Gas1	0.1682730	0.2658854	3.888040	0
+	−	Lhx2	Id2	0.2592593	0.1886792	3.857143	0
−	−	Neurog2	Gas1	0.1980831	0.1782539	4.808064	1
−	−	Aldoc	Gas1	0.1908948	0.2043608	4.531700	1
−	−	Dmrta2	Gas1	0.2159069	0.1930027	4.477001	1
+	−	Tubb2a	Gas1	0.2083116	0.2307470	4.333750	1
−	−	Shisa2	Gas1	0.2295070	0.2067857	4.241136	1
−	−	Gas1	Alcam	0.1855416	0.1636566	4.225522	1
+	−	Bub3	Gas1	0.2000000	0.2400000	4.166667	1
+	−	Lxn	Gas1	0.2436770	0.1234416	4.103793	1
−	+	Gas1	Ccnb1	0.2095238	0.2439024	4.100000	1
−	−	Tfap2c	Gas1	0.1835939	0.2362141	4.069087	1
−	+	Cdkn1b	Ttr	0.0000000	0.0000000	100.000000	2
−	+	Ddit4	Ttr	0.0000000	0.0000000	100.000000	2
+	+	Gadd45g	Ttr	0.0000000	0.0000000	100.000000	2
+	+	Hes6	Ttr	0.0000000	0.0000000	100.000000	2
−	+	Lhx2	Ttr	0.0000000	0.0000000	100.000000	2
+	+	Lig1	Ttr	0.0000000	0.0000000	100.000000	2
+	+	Lmo4	Ttr	0.0000000	0.0000000	100.000000	2
+	+	Mt1	Ttr	0.0000000	0.0000000	100.000000	2
−	+	Prr15	Ttr	0.0000000	0.0000000	100.000000	2
+	+	Slc1a3	Ttr	0.0000000	0.0000000	100.000000	2

C)

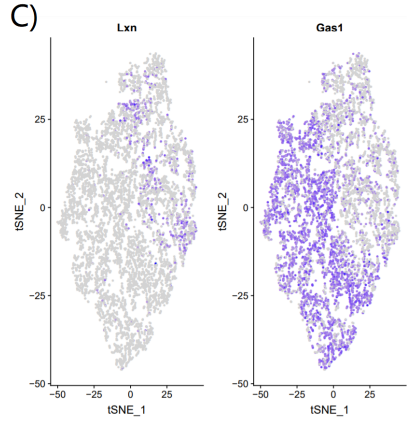


Figure 6: Sample output from random forest marker selection method. A) t-distributed stochastic neighbor embedding (t-SNE) embedding of mouse forebrain progenitor cells from embryonic time points, where cells are colored by a low resolution cluster assignment. B) Table of markers found by the random forest algorithm for each of the clusters. The top markers for each cluster are shown and ranked according to a score (in the conf_score column) computed as one over the overall error rate. C) Gene expression levels in the t-SNE embedding of a positive-negative marker pair found by the algorithm for cluster 1.

Using the random forest approach, discriminant gene markers were identified that revealed dorsal–ventral patterning in the brain. Figure 2.1 shows the gene markers identified by the random forest approach, which were not identified by differential expression analysis.

The genes $Pax6$ and $Lhx2$ are markers produced by the algorithm which have been shown to be related to dorsal patterning in the brain [20], while the genes $Nkx2-1$, $Ascl1$, $Dlx2$, $Rbp1$ mark ventral patterning. These results are included in the publication [20], and this method is being used in the lab for identifying gene markers to facilitate labeling cell populations during analysis of single cell data.

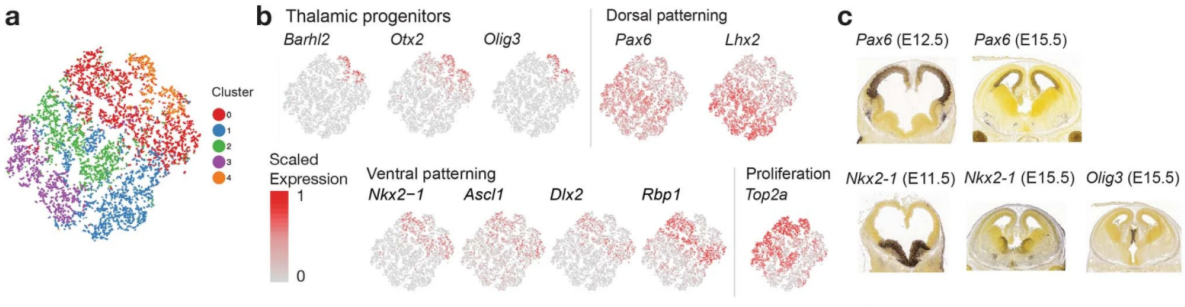


Figure 7: a) TSNE embedding of mouse forebrain progenitor populations from embryonic time points. Cells are colored by cluster assignment. b) t-SNE embedding colored by expression of top discriminant gene markers for each cluster, identified using a random forest-based approach. c) *In situ* hybridization of selected discriminant marker genes from the Allen Brain Atlas (2008 Allen Institute for Brain Science. Allen Developing Mouse Brain Atlas. Available from: developingmouse.brain-map.org)

3.2 Part II: Modeling differentiation dynamics

3.2.1 Interpolate and extrapolate system dynamics

To model the gene regulatory dynamics during differentiation of inhibitory neurons, I used a machine learning approach with single cell expression data of samples collected from the mouse brain at 3 time points: E12, E15, and P0 [20]. The random forest marker selection method was used to facilitate labeling of cell populations in these samples. From this dataset, we selected cells of the interneuron lineage. To remove the effect of cell cycle on trajectory inference, cells are scored based on cell cycle activity (figure 8), the $G2M$ score, and cells with a score above 0.4 are removed. Cells with a score below 0.4 are used to construct the pseudo-time trajectory using (figure 9) and train the machine learning model (see section 2.2.2 for more details).

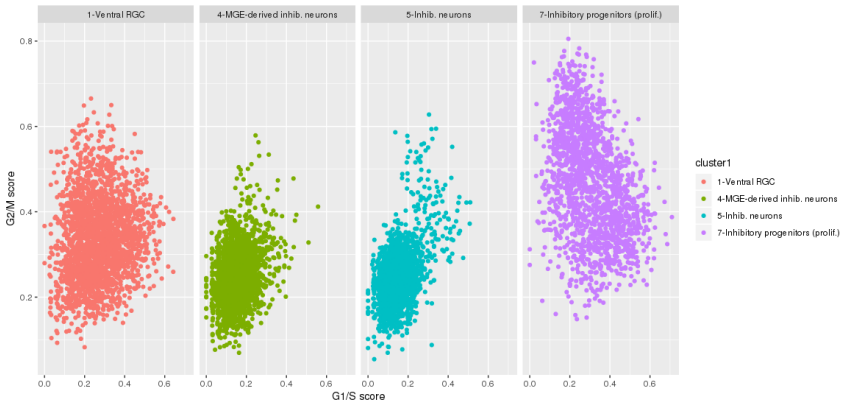


Figure 8: Cell cycle scoring for cell clusters selected for pseudo-time trajectory construction from single cell data collected from $E12$, $E15$, and $P0$.

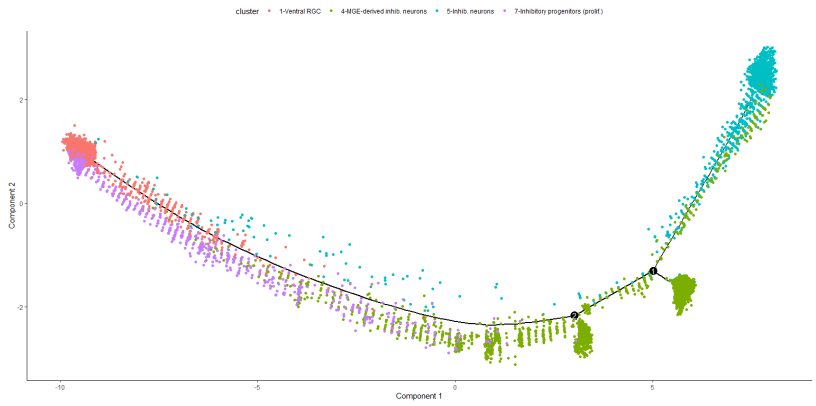


Figure 9: Pseudo-time trajectory reconstructed from single cell data of cells along the inhibitory neuron lineage at $E12$, $E15$, and $P0$ using Monocle [52].

To evaluate the model's ability to learn the gene dynamics in neural differentiation, we hide different parts of the data along the differentiation trajectory and ask the model to interpolate or extrapolate the system dynamics towards unobserved regions along pseudo-time. For the evaluation data, we sampled consecutive data points starting at different regions in the trajectory by removing 8% of the data. The training data was then smoothed using $LOESS$, and the smoothed data was used to train the model. The mean squared error (MSE) of the NODE predictions was then compared to a linear interpolation/extrapolation process, with the comparisons shown in table 1. In each case, the initial state $X(t_0)$ represents the earliest or latest time point in the test data, and the model was asked to predict $X(t > t_0)$ or $X(t < t_0)$ depending on the test scenario.

Table 1 shows that the linear extrapolation outperformed the neural ODE method. However, the neural ODE method is able to outperform the linear model in the interpolation case. This may be due to the fact that the gene expression profile is more linear at the two end points of the trajectory. By plotting the density of the cells along the trajectory (figure 10), we see that the cells are more densely sampled at the two end points. Thus, when 8% of the data at the end points of the trajectories was used as the test data, the corresponding

time window is small compared to other sampled segments. This can be seen in Table 1's test data time frame column, where the extrapolation time window is significantly smaller than the interpolation cases, thus favouring linear trends.

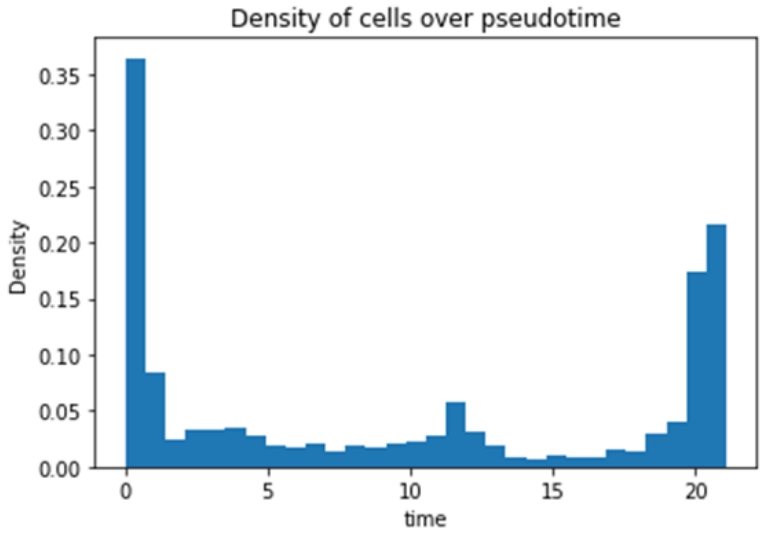


Figure 10: Cell density distribution across pseudo-time. The y-axis represents the percentage of cells distributed across the time x-axis.

We evaluate the model performance by plotting several well characterized genes known to be involved in the inhibitory neuron differentiation pathway across pseudo-time. We asked the model to predict the time evolution of the system, and observe the model is able to extrapolate and interpolate the system dynamics in those genes (figure 11).

In addition, we evaluated the model's ability to properly converge to the final state when part of the data is hidden in the trajectory. We ask the model to predict $X(t)$ for the entire pseudo-time trajectory starting from $X(0)$. Figure 12, show that all of the models trained by hiding different portions of the data were able to converge towards the final state.

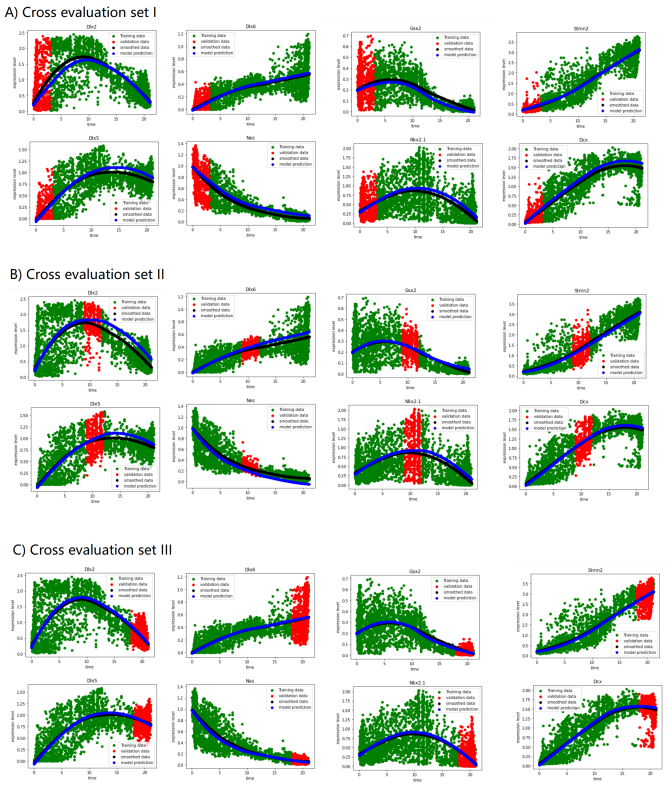


Figure 11: Cross evaluation of selected genes involved in the inhibitory neuron differentiation pathway. Each plot represents the expression level of a gene across pseudo-time. Each dot represents the expression level of a particular cell at its assigned pseudo-time by Monocle [52]. The green data points were used for training, while the red data points were used for testing. The blue line is the model prediction, while the black line is the smoothed version of the data. A) Data from the beginning of the trajectory are hidden and used for evaluation. B) Data from intermediate time points are hidden and used for evaluation. C) Data from end time points are hidden and used for evaluation.

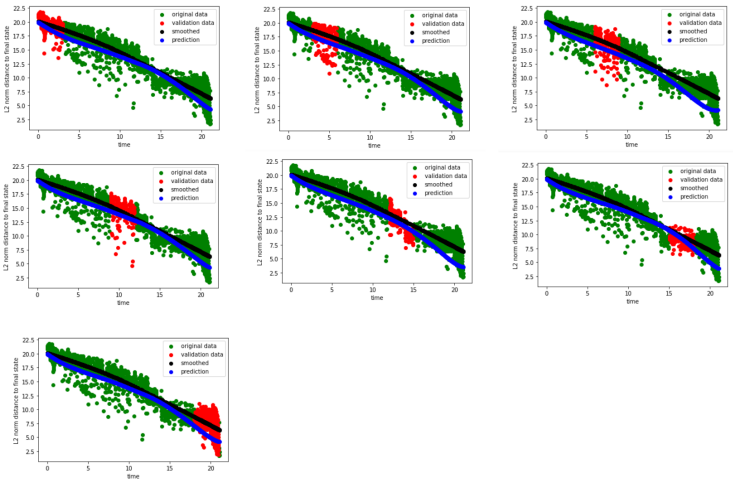


Figure 12: Cross evaluation for overall model performance. Each dot represents the L2 distance between each cell along the trajectory to the final state. The green data points were used for training, and the red data points were used for testing. The blue line is the model performance, and the black line is the smoothed version of the data.

To further evaluate the model performance, an independent test dataset from different developmental time points, $E13$ and $E16$, was used. The trajectory reconstruction process was performed by again selecting cells of the inhibitory neuron lineage. Cells with a $G2M$ score above 0.35 were removed to reduce the effect of cycling cells on trajectory construction (Figure 13). The Monocle trajectory for the test data is shown in Figure 14.

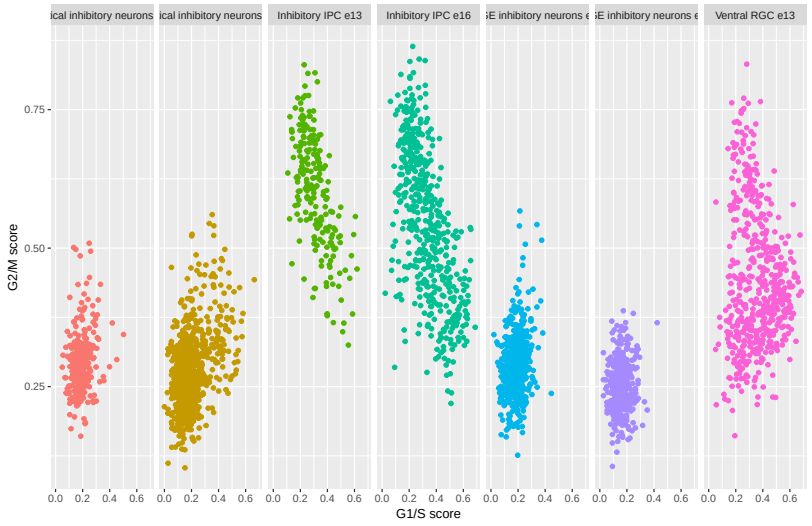


Figure 13: Cell cycle scoring for cell clusters selected for pseudo-time trajectory construction from single cell data collected from $E13$ and $E16$.

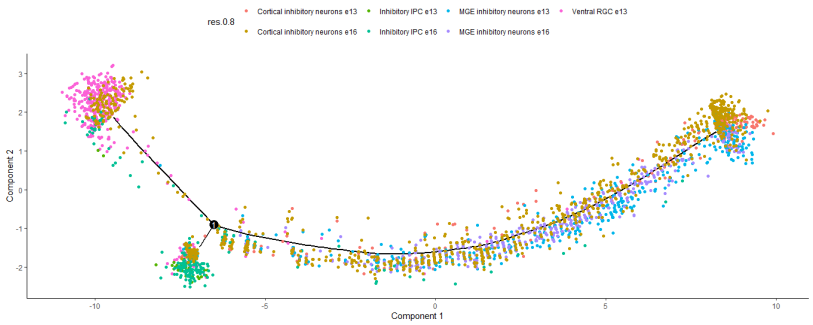


Figure 14: Trajectory reconstructed from single cell data collected from $E13$ and $E16$ using Monocle [52]

To characterize the model's performance on the test data, the model is first evaluated on its predictions of selected genes known to be involved in the inhibitory neuron differentiation pathway (figure 15). We observe for most genes, although slightly deviated, the model is able to accurately capture the overall trend of the genes behaviours over time. Some of the genes used in the training data, such as $Nkx2.1$, are not detected in the testing data, but still required as inputs to the model, are treated as zeros in the initial condition $X(0)$ when

predicting the gene regulatory dynamics $X(t)$. In addition to the individual genes, in figure 16, we see that the predictions also still converge to a similar final state in the test data. To further asses the model prediction on all genes, we compared the MSE of the selected genes to the MSE of all other detected genes. The MSE for the selected genes (omitting the undetected genes in the testing data) was 0.1139, while the MSE for all the genes was 0.0398. Here, we see that the error in the overall data is lower than the error in the selected genes. We suspect that this may result from a higher variance in the selected genes than the rest of the genes used for the modeling process. Thus, we compared the scaled absolute error (SAE) between the two. In the SAE computation, we scaled the absolute error between the predictions and the data for each gene by their variance. The SAE for the selected genes was 4.98, while the SAE for all genes was 3.54.

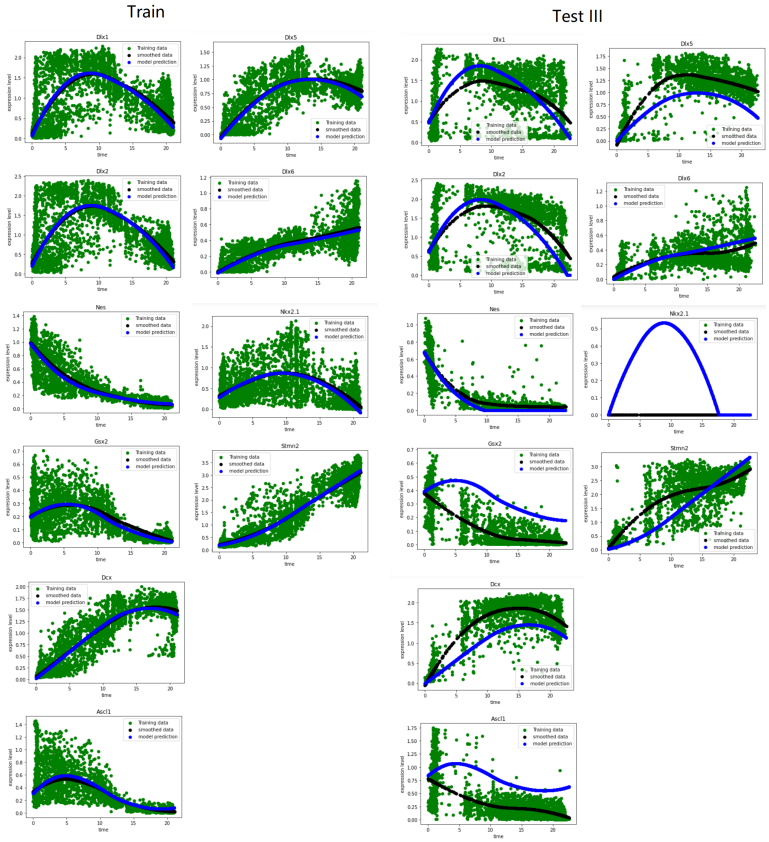


Figure 15: Model performance on selected genes in the training data and testing data. The blue line represents the model predictions and the black line represents the smoothed version of the data.

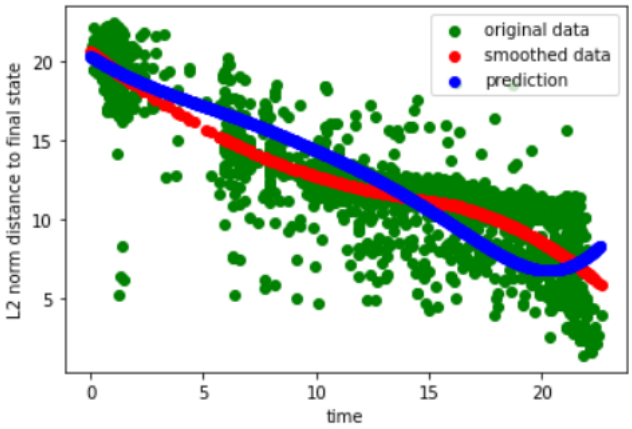


Figure 16: Model performance on the testing data, representing the convergence towards the final state.

To further characterize the model's prediction errors, we randomly sampled 100 initial states in the testing data, and evaluated the MSE and SAE. The average MSE for the selected genes was 0.1863 with a variance of 0.0022. The average MSE for all genes was 0.0563 with a variance of 0.0003. The SAE for the selected genes was 15.71 with a variance of 198.8, and the SAE for all the genes was 12.493 with a variance of 100.3.

3.2.2 Predicting the effect of gene deletions

By modeling the gene regulatory dynamics as an ODE system, it is possible to simulate perturbation response in the form of gene deletions by setting the value of the selected genes to zero during the integration step. The details of the approach are discussed in the Methods section 2.3. Here, to evaluate the model's ability to predict the effects of gene deletions on the overall gene expression profile, a perturb-seq dataset from [42] was used. In this study, peturb-seq was done on iPSC-derived neurons and used to identify the differentially expressed genes as a result of the gene deletion. The Z-scores were reported for the differentially expressed genes between the knockdowns and control. The model performance is evaluated by its accuracy at predicting whether a gene is up or down-regulated as a result of the gene deletion. The details of the test data and the evaluation process is described in section 2.3.2.

For this study, data from the dentate gyrus [18] was used to train the model, as it shares more detected genes with the evaluation dataset compared to our mouse developmental data (that is used in constructing the pseudo-time trajectories). The training data was processed through the steps described in sections 2.3.1, 2.3.1, and 2.3.1. During this data processing step DCA [13] was used to remove noise from the data. To confirm that the DCA imputation step did not drastically change the data structure or interfere with RNA velocity inference, the two-dimensional reduction of the data using Uniform Manifold Approximation and Projection (UMAP) [6] was compared for data processed with DCA (figure 17) and without DCA (figure 18). In the UMAP visualizations, the relative cluster positions are

comparable, while the directions of the velocity vectors for the DCA processed data resemble the non-imputed counterpart. However, in the non-imputed case, the velocity vectors in the mature granule population show a tendency toward the immature granule population, counter to our expectations. In contrast, the velocity vectors in the imputed data shows unidirectional flow from the immature to the mature granule population, which agrees with the known biology.

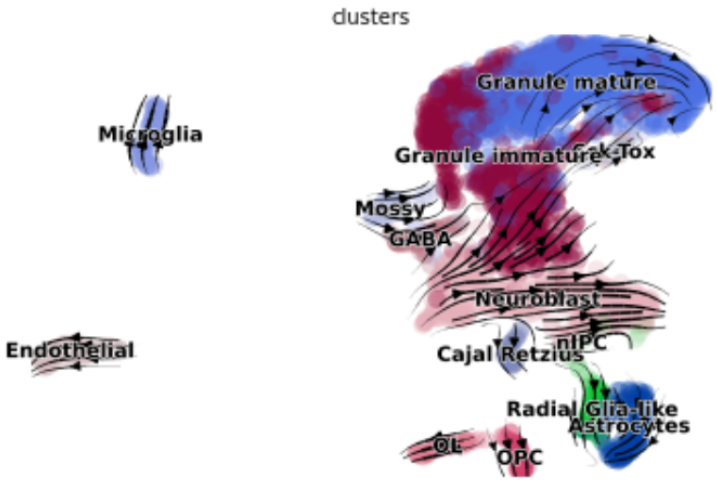


Figure 17: UMAP of Dentate Gyrus dataset after applying DCA to the single cell data with RNA velocity vectors computed by *scvelo*

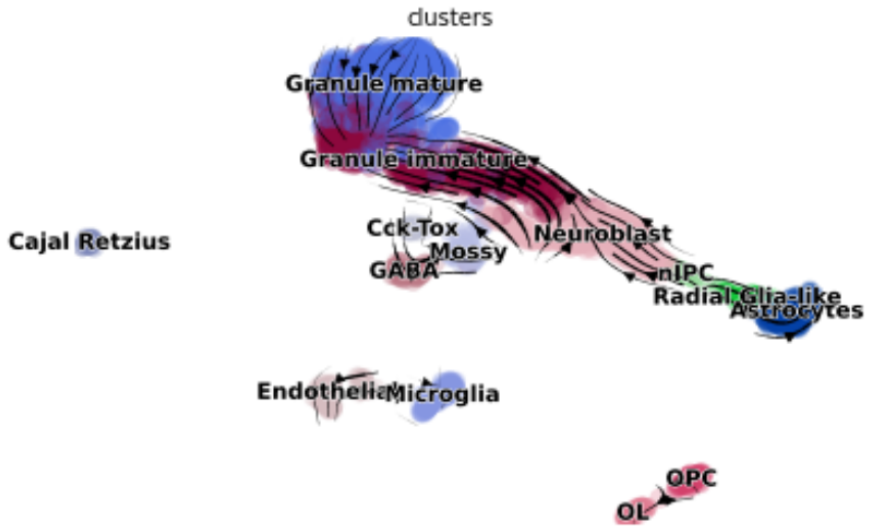


Figure 18: UMAP embedding of Dentate Gyrus dataset with RNA velocity vectors computed by *scvelo*

During the modeling process, the top 1000 genes were included in the model. In this setup, the model was used to predict the effects of 4 different gene deletions on the overall expression profile. To predict the effect the perturbation response, this process was simulated by performing numeric integration on the learned system of equations. The process of generating predictions is described in section 2.3.2. The $UMAP$ embedding of the processed data is shown in figure 17.

To predict the effect of the gene deletions in the test data, cells from the granule cell cluster were set as the initial states of the simulation, following the described steps in section 2.3.2. Numeric integration was applied on the system to compute and compare the final states of the trajectories with and without the gene deletion. For each gene deletion, the simulation was repeated 100 times, each starting with a randomly sampled cell state in the granule cluster. This was done for each of the 4 gene deletions in the test data. The group average for each deletion event was used as the final prediction. The predictions were binarized to either up or down-regulated for each of the affected genes as a result of

the deletion, which reduces this to a binary classification problem. During the binarization process, a threshold value was set on the Z-score to filter out expression value changes that were not significant. Only genes with a Z-score of absolute value above 0.5 were used. The accuracy of the model evaluated on this test data was 91%, while the baseline accuracy was only 68% (shown in Figure 19). The baseline accuracy here was computed as the maximum of either: 1) always predict true or 2) always predict false, which represents the best possible non-educated guess on the outcome of a binary classification problem.

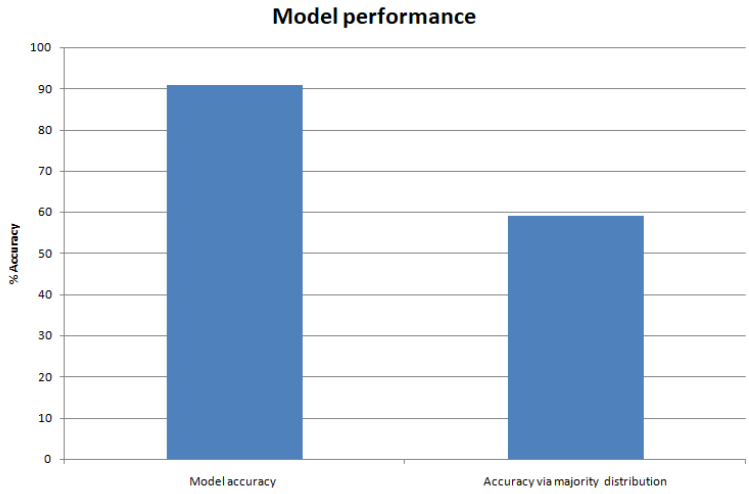


Figure 19: Model's prediction accuracy (left) on predicting the effect of gene deletions, compared to the baseline (right). The model was trained using 1000 genes and evaluated on 4 different gene deletions.

3.2.3 Extending to more genes

To test the limit of how many genes can be modeled using this approach, the number of genes included in the model was increased from 1000 to 2000. In this setup, the model was used to predict the effects of 5 different gene deletions (1 additional gene deletion, which was filtered out in the previous model). The model was trained and evaluated using the same setup as above. However, the model was not able to outperform the baseline, as shown in

figure 20. The increase in dimensionality may be the main cause in the decrease in model performance. To account for this issue, a dimensionality reduction approach was applied. Here, we used $L1$ regularization to reduce the number of variables used for the regression task of each gene by setting certain edge weights to zero. The details of this approach are described in 2.3.1. After applying this approach, the model was again able to outperform the baseline by 15% (shown in figure 21). However, the accuracy was still lower than the previous model including 1000 genes.

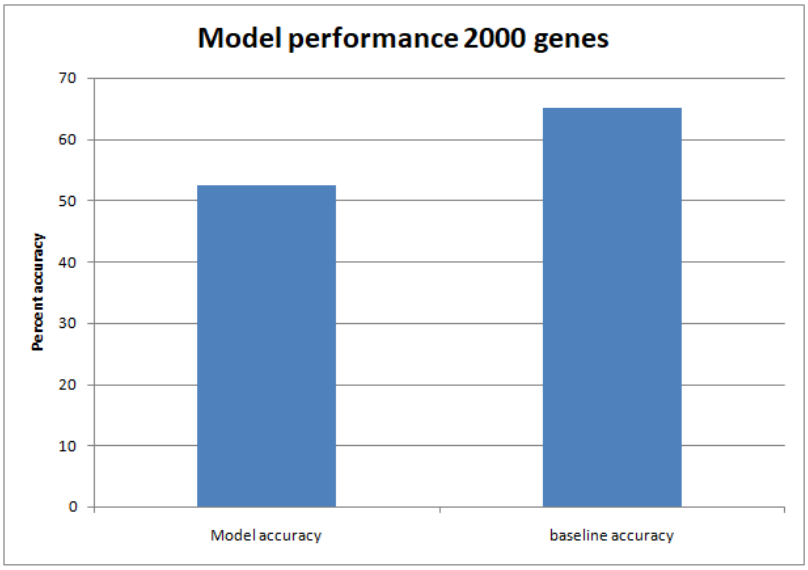


Figure 20: Model's prediction accuracy (left) on predicting the effect of gene deletions, compared to the baseline (right). The model was trained using 2000 genes and evaluated on 5 different gene deletions.

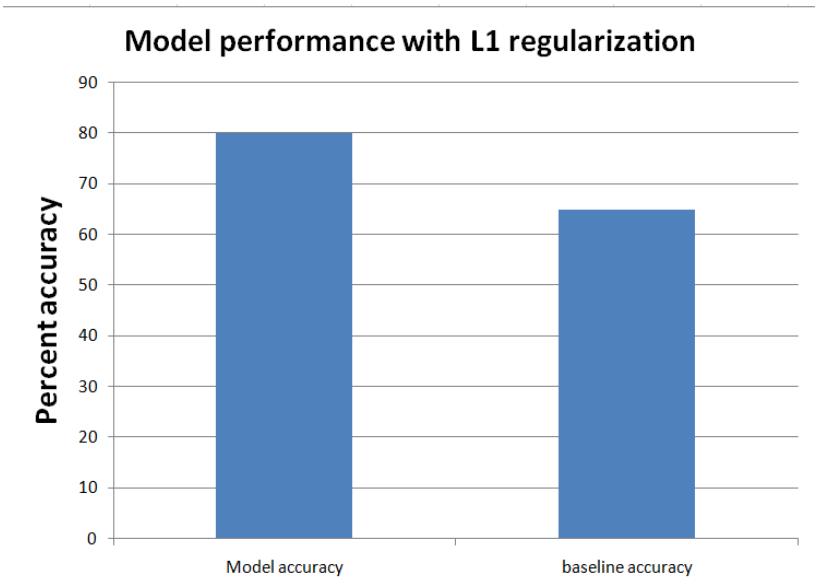


Figure 21: Model's prediction accuracy (left) on predicting the effect of gene deletions, compared to the baseline (right). The model was trained using 2000 genes with L1 regularization and evaluated on 5 different gene deletions.

.

To further test the limits of the model, the number of genes was increased to 3000. In this setup, the model was used to predict the effects of 8 different gene deletions (3 additional gene deletion from the previous case). The same data processing steps were applied, and the model was trained in the two cases with and without the $L1$ regularization. However, in either case, the model was not able to outperform the baseline.

4 Discussion

Understanding the molecular characteristics and underlying regulatory mechanisms of cellular differentiation is an important goal in current research. To study this complex process, the high resolution data generated from single cell RNA sequencing technology provides a convenient platform to investigate molecular changes during differentiation. In this study, I focus on developing computation methods for understanding cellular differentiation, including two methods: a random forest approach to characterize molecular markers defining cell populations, and a method to model cellular dynamics with neural networks.

The random forest approach identified gene markers that differential expression analysis had missed. The advantage of the random forest approach is its ability to evaluate the combinatorial effect of multiple genes to distinguish a population of cells. However, two limitations exist for this approach. First, this method tends to overfit when the number of cells in a cluster is small. This can be seen in figure 6, where the algorithm finds high score markers for a very small cluster as an artifact of the clustering algorithm. This represents an issue with machine learning algorithms in general, where they will often find patterns in the data, even when the patterns are artificial. Second, the markers found by the algorithm are difficult to evaluate on their biological significance. The markers found by the algorithm in figure 7 are cross referenced with prior knowledge of gene expression in brain development. However, not all genes found by the algorithm can be validated in this manner, as the roles of many genes involved in the the developmental process remain unknown, and the random forest algorithm cannot determine the biological significance of its gene set.

In contrast, modeling of cellular dynamics identifies the relative relationship between genes, which can explain the significance of genes in terms of how they impact one another. In this approach, dynamics data inferred from single cell data, in the form of pseudo-time or RNA velocity, is used to fit a system of ODEs in the form of neural networks. This model is able to predict how the cell's transcriptome changes over time during differentiation, as

well as the effect of gene perturbations. As the model uses inferred dynamics data from single cell, it is reliant on the accuracy of the inference algorithm. In particular, pseudo-time inference and RNA velocity are used as the main methods of inferring the dynamics information. In the pseudo-time case, the dynamics information is limited to only one trajectory per differentiation process, where the differences between individual cells, which contains perturbation information, are smoothed out along the trajectory. Thus, the models trained using pseudo-time data are not suited for the prediction of gene perturbations when the number of genes in the model is large. When trying to use this model to predict perturbation response, it predicts no change in the majority of cases where changes should be present, indicating the cause-effect relationship between genes is not correctly captured. In contrast, RNA velocity estimates the velocity information at the cellular level, which captures the deviation between individual cells. The model trained using RNA velocity data is able to predict the effect of perturbations, but shows a decrease in performance as the number of genes in the model increases.

In this experiment, we show that by using a machine learning approach to model the gene regulatory dynamics in single cell data, we can use the model to predict: 1) the change in gene expression profile during differentiation as well as 2) the change in the gene expression profile as a result of gene deletions. This machine learning approach is data driven, and requires minimal prior knowledge on the genes to be modeled. The future goal of this method is to use the model to identify gene targets for disease intervention. The details of how the model can be used to identify gene targets will be discussed in section 4.2.2

4.1 Requirements and limitations on modeling cellular dynamics using the neural ODE approach

4.1.1 Overview

In this section, I discuss the assumptions and limitations of the model. In particular, 1) in order for the model to predict a perturbation response, it requires the corresponding dynamics information to be included in the data, and 2) the number of genes included in the model is limited by the size of the dataset, and the required data may increase exponentially as a function of the number of genes included in the model.

4.1.2 To predict the effect of a perturbation, the corresponding dynamics need to be included in the training data

In this project, I construct a machine learning model that tries to predict the effect of gene deletions on the cell's overall transcriptomic profile, despite the gene deletions not having been observed in the training data. This out-of-sample prediction task is a difficult problem for machine learning methods [38], as most supervised machine learning methods require the sample distribution of the test data to mimic the training data [12] [28] [10]. Since the test data is on gene deletions that were never present in the training data, this assumption appears to be violated. However, as we select highly variable genes to be included in the model, which includes genes that are turned on and off at different stages, the outcome of a gene deletion on the overall transcriptome can be inferred from observing the on the off states of that gene and extending the impact on its downstream targets, which are learned by the model. Here, the underlying assumption is that both the on and off states of the perturbed gene are observed in the training data. For this reason, this method would have difficulties generalizing towards cell populations that are not observed in the training data, as out-of-sample cell types can have active biological programs that are absent in the training

data.

4.1.3 The amount of data required by the model increases exponentially with respect to the number of genes to be modeled, but can be reduced by imposing constraints on the system

In addition to the above requirements, the amount of data points sampled also limits the performance of the model. In the simplest case, each gene can be considered as having one of two states: 'on' or 'off'. The total number of possible configurations for a system with k genes is thus 2^k, although in reality when the expression is continuous, this number grows even larger. If n represents the total number of different configurations of the system, the expected number of samples under uniform distribution, represented by $E(N)$, needed to capture all the different configurations/states in the data is:

$$E(N) = \sum_{i=0}^{2^k-1} (\frac{2^k}{2^k - i}) > 2^k \tag{7}$$

The required data is expected to grow exponentially with respect to the number of genes to be modeled. Thus, the performance of the model would start to decrease as more genes are included in the model, as described in section 3.2.3. Furthermore, as the sampling of cell states is generally not a uniform distribution, this will require more samples than this estimate. To produce a more sample efficient approach, for each equation $\frac{dx_i}{dt} = f(X)$, not all genes are needed. If a constraint is imposed on the system, such as limiting the number of possible regulators of a gene, the total number of configurations can be reduced. Letting l be the upper bound on the number of regulators of a particular gene, the total number of possible configurations for the system is $k2^l$, where $l << k$. This is done by removing weakly correlating genes from each equation via $L1$ regularization. The limitation of this approach is that the gene selection process is stochastic and has a selection bias dependent

on the neural network structure. In particular, different gene sets would be selected when the number of layers or the activation function change. Moreover, the gene set varies even when the network structure is fixed due to the stochastic nature of the training process (although we expect less variation in the gene set than when the network structure is altered).

In summary, to reduce the exponential increase in data requirement, $L1$ regularization is used to reduce the number of possible regulators to each gene. The downside is that biases are introduced during this feature selection process. Thus, fine tuning of the $L1$ regularization parameters is required to select for the optimal trade-off between bias and potential over-fitting.

4.2 Future work

4.2.1 Incorporating hidden states

In my model, I assume that the future state of a cell is only dependent on its current transcriptomic state. However, this may not always be the case, as the epigenetic and proteomic states also play a significant role in this process. For a more comprehensive model, incorporating these layers of information could potentially lead to better performance. Unfortunately, as epigenetic and proteomic information is not yet available at the single cell level, they cannot be directly included in the modeling process. However, they can be incorporated in the form of hidden states in the model. To demonstrate this, consider the following example: first, let us assume that the flow of information between the epigenetic state (E), transcriptomic state (T), and proteomic state (P) can be represented in figure 22.

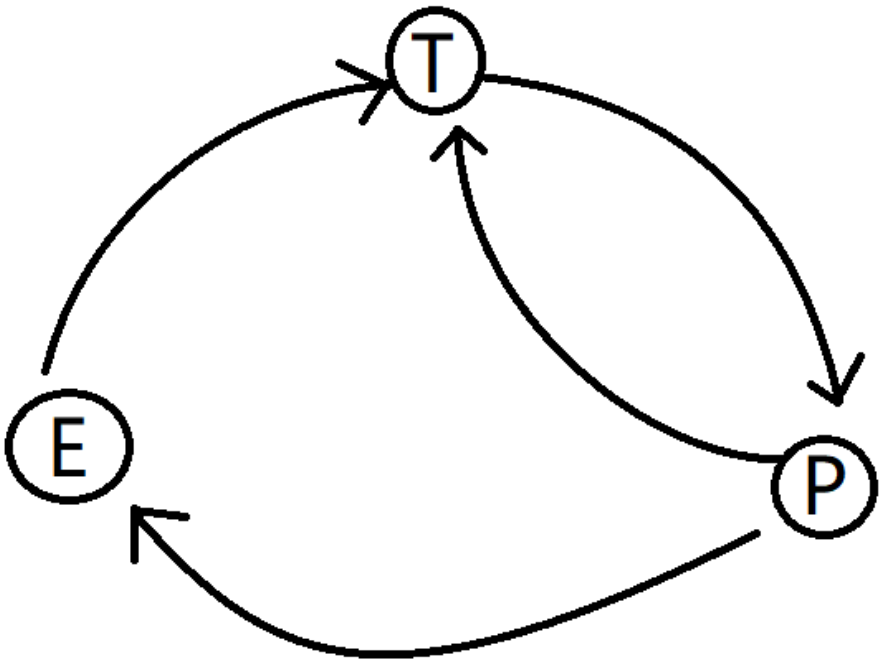


Figure 22: State diagram of incorporating epegenetic and proteomic states into the system. T represents the RNA state, E represents the epigenetic state, and P represents the proteomic state.

Following the diagram, the state transition equation can be described in the following form:

$$E_{t+\Delta t} = f(P_t) \tag{8}$$

$$T_{t+\Delta t} = f(E_t, P_t) \tag{9}$$

$$P_{t+\Delta t} = f(T_t) \tag{10}$$

In this formulation, the future E state (denoted by $E_{t+\Delta t}$) is dependent on the current P state (P_t), where the transcriptomic state $T_{t+\Delta t}$ is dependent on P_t & E_t, and $P_{t+\Delta t}$ is dependent on T_t. Since the E and P are not observed, the goal is to rearrange the equations such that we can express $T_{t+\Delta t}$ only as a function of the past states of T. This can be done by incorporating higher order derivatives, where the information coming from the E and P states can be approximated by the second and third derivatives of T (i.e., $\frac{d^2T_t}{dt^2}$ and $\frac{d^3T_t}{dt^3}$). The details of this formulation are discussed in Appendix A.

4.2.2 Using the machine learning model to control gene regulatory dynamics and identifying optimal gene targets

The future application of this model is to identify the perturbations needed to control certain aspects of the gene regulatory network. As an example, the model can be used to predict the perturbations required to push the cells from an initial state $X(0)$ to a desired target state X_f. During cellular differentiation, the expression profile of a cell irreversibly changes from the naive state towards their differentiated state. The model, in this case, can be used to investigate the potential gene perturbations needed to push the cells from the differentiated state back to its naive state. Here, I will discuss the mathematical procedures for this prediction process.

The model can be used to predict the gene dynamics from an initial condition $X(0)$ by computing the integral $X(T) = \int_0^T \frac{dX}{dt}\dot{dt}$. The dynamics of the system in response to perturbations can be computed via $\hat{X(T)} = \int_0^T (\frac{dX}{dt} + P(\theta, X))\dot{dt}$, where $P(\theta, X)$ are the user-defined perturbations applied to the system with θ being the free-parameters. Here $P(\theta, X)$ represents a system of functions $p_i(\theta_i, X)$ with the same dimensionality as $\frac{dX}{dt}$. The objective is to choose the perturbations that minimize the loss function $\left\| \hat{X(T)} - X_f \right\|_2$, with $\|\|_2$ representing the $L2$ norm. The solution can be found via optimization methods such as gradient descent. For practical applications, the preference is to select solutions that

perturb the minimal number of genes to achieve the desired outcome. This can be seen as enforcing sparsity on $P(\theta, X)$. Let's assume each element $p_i(\theta_i, X)$ in $P(\theta, X)$ takes the form of $p_i(\theta_i, X) = \theta_i \times \frac{dx_i}{dt}$, such that the perturbed version of the equation is $\hat{x_i(T)} = \int_0^T (\frac{dx_i}{dt} + \theta_i \times \frac{dx_i}{dt})dt$. If θ_i is zero, the equation behaves as if no external perturbation is applied to gene i. When $\theta_i < 0$, gene i is down-regulated, while positive values of θ_i implies up-regulation. Thus, to minimize the number of genes to perturb, $L1$ regularization can be applied to θ during the regression process. In summary, by applying a regression task on the loss function in equation 11, one can use the model to identify perturbations to control the cell's transcriptomic state, and will be the future goal of this project.

$$\left\|X(T) - X_f\right\|_2 + \|\theta\|_1 \tag{11}$$

5 Conclusion

In summary, this thesis presents two computational methods for the study of cellular differentiation: one to identify molecular markers of cell populations using a random forest approach, and one to model cellular dynamics with neural networks. While the random forest approach is able to identify combinations of gene markers that differential expression analysis missed, the neural network approach is able to learn the gene regulatory dynamics from single cell data and predict the effect of gene perturbations at the cell level. This neural network approach can be extended in future applications to investigate gene targets for manipulating cell fate.

References

[1] Brian Aevermann et al. A machine learning method for the discovery of minimum marker gene combinations for cell type identification from single-cell rna sequencing. *Genome Res*, 31, 2021.

[2] Omar Alaqeeli et al. Software benchmark—classification tree algorithms for cell atlases annotation using single-cell rna-sequencing data. *Microbiology Research*, 12, 2021.

[3] David N. Arnosti and Ahmet Ay. Boolean modeling of gene regulatory networks: Driesch redux. *PNAS*, 45, 2012.

[4] Xuanwen Bao et al. Integrated analysis of single-cell rna-seq and bulk rna-seq unravels tumour heterogeneity plus m2-like tumour-associated macrophage infltration and aggressiveness in tnbc. *Cancer Immunol Immunother*, 1, 2021.

[5] Albert-László Barabási and Zoltán N. Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5, 2004.

[6] Etienne Becht et al. Dimensionality reduction for visualizing single-cell data using umap. *Nat Biotechnol*, 37:38–47, 2019.

[7] Alfredo Benso et al. An extended gene protein/products boolean network model including post-transcriptional regulation. *Theor Biol Med Model*, 11, 2014.

[8] Volker Bergen et al. Generalizing rna velocity to transient cell states through dynamical modeling. *Nature Biotechnology*, 2020.

[9] Volker Bergen et al. Rna velocity—current challenges and future perspectives. *Mol Syst Biol*, 26, 2021.

[10] Longbing Cao. Non-iidness learning in behavioral and social data. *The Computer Journal*, 57(9):1358–1370, 2014.

[11] Anna S. E. Cuomo et al. Single-cell rna-sequencing of differentiating ips cells reveals dynamic genetic effects on gene expression. *Nat Communications*, 11, 2020.

[12] Murat Dundar et al. Learning classifiers when the training data is not iid. *IJCAI*, 2007.

[13] Gökcen Eraslan et al. Single-cell rna-seq denoising using a deep count autoencoder. *Nat communications*, 10, 2019.

[14] David K. Gifford Grace H.T. Yeo, Sachit D. Saksena. Generative modeling of single-cell population time series for inferring cell differentiation landscapes. *bioRxiv*, bioRxiv:10.1101/2020.08.26.269332, 2020.

[15] Christopher Heje Grønbech et al. scvae: variational auto-encoders for single-cell gene expression data. *Bioinformatics*, 36, 2020.

[16] Laleh Haghverdi et al. Diffusion pseudotime robustly reconstructs lineage branching. *Nat Methods*, 13, 2016.

[17] Tiam Heydari, Matthew A. Langley, Cynthia Fisher, Daniel Aguilar-Hidalgo, Shreya Shukla, Ayako Yachie-Kinoshita, Michael Hughes, Kelly M. McNagny, and Peter W. Zandstra. Iqcell: A platform for predicting the effect of gene perturbations on developmental trajectories using single-cell rna-seq data. *bioRxiv*, 2021.

[18] Hannah Hochgerner et al. Conserved properties of dentate gyrus neurogenesis across postnatal development revealed by single-cell rna sequencing. *Nat Neurosci*, 2:290–299, 2018.

[19] Sonja Hänzelmann et al. Gsva: gene set variation analysis for microarray and rna-seq data. *BMC Bioinformatics*, 2013.

[20] Selin Jessa et al. Stalled developmental programs at the root of pediatric brain tumours. *Nat Genet*, 51:1702–1713, 2019.

[21] Hongyu Zhao Jiguo Cao, Xin Qi. Modeling gene regulation networks using ordinary differential equations. *Methods Mol Biol*, 2012.

[22] Stephen C. Juvet and Li Zhang. Double negative regulatory t cells in transplantation and autoimmunity: recent progress and future directions. *J Mol Cell Biol*, 4, 2012.

[23] Xiaohan Kang et al. From graph topology to ode models for gene regulatory networks. *PLOS ONE*, 2020.

[24] Jiuyong Li et al. Building diversified multiple trees for classification in high dimensional noisy biomedical data. *Health Inf Sci Syst*, 5, 2017.

[25] Arthur Liberzon et al. The molecular signatures database (msigdb) hallmark gene set collection. *Cell Syst*, 6:417–425, 2015.

[26] Mohammad Lotfollahi et al. Conditional out-of-distribution generation for unpaired data using transfer vae. *Bioinformatics*, 36, 2020.

[27] Mohammad Lotfollahi et al. Out-of-distribution prediction with disentangled representations for single-cell rna sequencing data. *bioRxiv*, 2021.

[28] Alexandra L'Heureux, Katarina Grolinger, Hany F. Elyamany, and Miriam A. M. Capretz. Machine learning with big data: Challenges and approaches. *IEEE Access*, 5:7776–7797, 2017.

[29] Gioele La Manno et al. Rna velocity of single cells. *Nat methods*, 560:494–498, 2018.

[30] Gioele La Manno et al. Molecular architecture of the developing mouse brain. *Nature*, 596, 2021.

[31] Mehrdad Nourani Maziyar Baran Pouyan. Clustering single-cell expression data using random forest graphs. *IEEE J Biomed Health Inform*, 21, 2016.

[32] Markus Mittnenzweig et al. A single-embryo, single-cell time-resolved model for mouse gastrulation. *Cell*, 11, 2021.

[33] Fabian J Theis Mohammad Lotfollahi, F Alexander Wolf. scgen predicts single-cell perturbation responses. *Nat methods*, 16(8):715–721, 2019.

[34] AMohammad Moradi et al. A boolean network control algorithm guided by forward dynamic programming. *PLOS ONE*, 2019.

[35] Jean-Philippe Vert Pierre-Cyril Aubin-Frankowski. Gene regulation inference from single-cell rna-seq data with linear differential equations and velocity inference. *Bioinformatics*, 18, 2020.

[36] Blanca Pijuan-Sala et al. A single-cell molecular map of mouse gastrulation and early organogenesis. *Nature*, 566, 2019.

[37] Jun Zhang Hua Liang Qi Zhang, Yao Yu. Using single-index odes to study dynamic gene regulatory network. *PLOS ONE*, 2018.

[38] Kamiar Rahnama Rad, Wenda Zhou, and Arian Maleki. Error bounds in estimating the out-of-sample prediction error using leave-one-out cross validation in high-dimensions. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 4067–4077. PMLR, 26–28 Aug 2020.

[39] Jesse Bettencourt David Duvenaud Ricky T. Q. Chen, Yulia Rubanova. Neural ordinary differential equations. *arXiv*, arXiv:1806.07366, 2018.

[40] Alexander B. rosenberg et al. Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science*, 360, 2018.

[41] Ellen V. Rothenberg. Causal gene regulatory network modeling and genomics: Second-generation challenges. *Journal of Computational Biology*, 2019.

[42] Connor H.Ludwig Ruilin Tian, Mariam A.Gachechiladze et al. Crispr interference-based platform for multimodal genetic screens in human ipsc-derived neurons. *Cell Press*, 104:239–255, 2019.

[43] Wouter Saelens et al. A comparison of single-cell trajectory inference methods. *Nat biotechnology*, 37, 2019.

[44] Sagar and Dominic Grün. Deciphering cell fate decision by integrated single-cell sequencing analysis. *Annu Rev Biomed Data Sci*, 2020.

[45] Geoffrey F. Schau, Guillaume Thibault, Mark A. Dane, Joe W. Gray, Laura M. Heiser, and Young Hwan Chang. Variational autoencoding tissue response to microenvironment perturbation. In Elsa D. Angelini and Bennett A. Landman, editors, *Medical Imaging 2019: Image Processing*, volume 10949, pages 407 – 415. International Society for Optics and Photonics, SPIE, 2019.

[46] Ning Shi et al. Aten: And/or tree ensemble for inferring accurate boolean network topology and dynamics. *Bioinformatics*, 36, 2020.

[47] Yung-Keun Kwon Shohag Barman. A boolean network inference from time-series gene expression data using a genetic algorithm. *Bioinformatics*, 32, 2018.

[48] Larisa M. Soto et al. scmomentum: Inference of cell-type-specific regulatory networks and energy landscapes. *bioRxiv*, 2021.

[49] Aravind Subramanian et al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *PNAS*, 43, 2005.

[50] Roser Vento-Tormo Sarah A Teichmann Valentine Svensson. Exponential scaling of single-cell rna-seq in the past decade. *Nature Protocols*, 13, 2018.

[51] Li Wang Bingcheng Ren Qi Mao Timothy Durham José L.McFaline-Figueroa Lauren Saunders ColeTrapnell Sreeram Kannan Xiao jie Qiu, Arman Rahimzamani. Inferring causal gene regulatory networks from coupled single-cell expression dynamics using scribe. *Cell Systems*, 10(3):265–274, 2020.

[52] Ying Tang Li Wang Raghav Chawla Hannah Pliner Cole Trapnell Xiaojie Qiu, Qi Mao. Reversed graph embedding resolves complex single-cell developmental trajectories. *Nat methods*, 14:979–982, 2017.

[53] Michael Q. Ding Xinghua Lu Yifan Xue. Learning to encode cellular responses to systematic perturbations with deep generative models. *npj Systems Biology and Applications*, 6, 2020.

[54] David Duvenaud Yulia Rubanova, Ricky T. Q. Chen. Latent odes for irregularly-sampled time series. *arXiv*, arXiv:1907.03907, 2019.

[55] Amit Zeisel et al. Molecular architecture of the mouse nervous system. *Cell*, 174:999–1014., 2018.

[56] Hongkai Ji Zhicheng Ji. Tscan: Pseudo-time reconstruction and evaluation in single-cell rna-seq analysis. *Nucleic Acids Research*, 44, 2016.

A Incorporating hidden states to the model

Starting with the equation in discrete time:

$$T_{t+1} = f(T_{t-2}, T_{t-1}) \tag{12}$$

substituting $t \leftarrow t + 2$

$$T_{t+3} = f(T_t, T_{t+1}) \tag{13}$$

The future states T_{t+1} and T_{t+2} can be expanded in the following form via first order estimate:

$$T_{t+1} = T_t + \Delta t \frac{dT_t}{dt} \tag{14}$$

$$T_{t+2} = T_{t+1} + \Delta t \frac{dT_{t+1}}{dt} \tag{15}$$

$$T_{t+2} = T_t + \Delta t \frac{dT_t}{dt} + \Delta t (\frac{dT_t}{dt} + \Delta t \frac{d^2T_t}{dt^2}) \tag{16}$$

Expanding T_{t+3} in terms of T_t only using first order estimate:

$$T_{t+3} = T_{t+2} + \Delta t \frac{dT_{t+2}}{dt} \tag{17}$$

$$T_{t+3} = T_{t+1} + \Delta t \frac{dT_{t+1}}{dt} + \Delta t (\frac{dT_{t+1}}{dt} + \Delta t \frac{d^2T_{t+1}}{dt^2}) \tag{18}$$

$$T_{t+3} = T_{t+1} + 2\Delta t \frac{dT_{t+1}}{dt} + (\Delta t)^2 \frac{d^2T_{t+1}}{dt^2}) \tag{19}$$

$$T_{t+3} = T_t + \Delta t \frac{dT}{dt}(t) + 2\Delta t(\frac{dT_t}{dt} + \Delta t \frac{d^2T_t}{dt}) + (\Delta t)^2(\frac{d^2T_t}{dt^2} + \Delta t \frac{d^3T_t}{dt^3}) \tag{20}$$

$$T_{t+3} = T_t + 3\Delta t \frac{dT_t}{dt} + 2(\Delta t)^2 \frac{d^2T_t}{dt^2} + (\Delta t)^3 \frac{d^3T_t}{dt^3} \tag{21}$$

Where the derivatives can be modeled as: $\frac{d^3T}{dt^3} = f(T)$ (where $f(T)$ is to be approximated by a neural network). The other derivatives can be computed as: $\frac{d^2T}{dt^2}(t + \Delta t) = \frac{d^2T}{dt^2}(t) + (\Delta t)\frac{d^3T}{dt^3}$, and $\frac{dT}{dt}(t + \Delta t) = \frac{dT}{dt}(t) + (\Delta t) * \frac{d^2T}{dt^2}$. To predict the time evolution of the system $T(t = 0 \rightarrow \infty)$, starting with the initial conditions $T(t = 0)$, $\frac{dT}{dt}(t = 0)$, and $\frac{d^2T}{dt^2}(t = 0)$, the neural network predicts $\frac{d^3T}{dt^3}(t = 0)$. This prediction is then use to predict $\frac{d^2T}{dt^2}(t + \Delta t)$. Subsequently, $\frac{dT}{dt}(t+\Delta t)$ is predicted using $\frac{d^2T}{dt^2}(t)$, and $T(t+\Delta t)$ is predicted using $T(t+\Delta t) = T(t) + \Delta t * \frac{dT}{dt}(t)$. Then, $T(t = 0 \rightarrow \infty)$ is computed by applying this update iteratively. Note that it takes 3 update steps for the neural network $f(T)$ to propagate its information to $T(t)$ in oppose to 1 when we omit the epigenetic and proteomic states.

Here, although the epigenetic states and proteomic states are not directly observed, the system can be model using higher order derivatives in the RNA data. In the case of pseudo-time data, the higher order derivatives can be estimated from consecutive points along pseudo-time. Whereas in RNA velocity data the higher terms can also be computed in a similar matter. As an example, the RNA acceleration (which is computed in [29]) can be computed using the spliced and unspliced counts as:

$$\frac{d^2s}{dt^2} = \beta u + 2\beta us + \gamma s - 2\gamma s^2 \tag{22}$$

while other derivative terms can be computed in a similar manner. In summary, the effects of epigenetic and proteomic information can be included in the model by introducing them as hidden states in aims to improve model performance.

Experiment	test data time frame	linear prediction MSE	Neural ODE MSE
Backward Extrapolation	0.0 - 0.398	7.23e-08	8.96e-06
Forward Extrapolation	19.0 - 21.1	2.76e-07	3.07e-06
Interpolation 1	1.00 - 3.80	8.07e-06	5.71e-06
Interpolation 2	3.00 - 4.89	6.64e-06	9.12e-06
Interpolation 3	5.00 - 9.41	1.06e-4	9.80e-05
Interpolation 4	9.01 - 11.5	1.35e-05	1.76e-05
Interpolation 5	12.1 - 17.9	4.01e-04	1.15e-04
Interpolation 6	18.0 - 20.2	1.05e-05	6.32e-06

Table 1: Cross evaluation of model performance by comparing the mean squared error on the hideout data between the neural ODE method and linear prediction. The models were evaluated on the hideout data. Different portions of the data were used as hideout data, where each segment of the hideout data consists of 8% of the total number of data points. The distribution of the population density of cells at different stages of the differentiation process is not even, therefore the time window length corresponding to each hideout segment is different despite the total number of cells in each segment remaining the same. In the extrapolation case, the hideout data was taken at the ends of the trajectory where the models were to extend the system dynamics beyond the sampled time window of the training data. In the interpolation case, the hideout data was taken in the middle of the trajectory, where the time window of the hideout data lies within the range of the training data. In the extrapolation case, the linear model extrapolates using the slope at the end of the training data. In the interpolation case, the linear model is constructed by connecting the two points between the segments via a straight line.