Model-based Trajectory Inference for Single-Cell RNA Sequencing Using Deep Learning with a Mixture Prior

Jin-Hong Du , Ming Gao , Jingshu Wang

Department of Statistics, The University of Chicago

Abstract

Trajectory inference methods analyze thousands of cells from single-cell sequencing technologies and computationally infer their developmental trajectories. Though many tools have been developed for trajectory inference, most of them lack a coherent statistical model and reliable uncertainty quantification. In this paper, we present VITAE, a probabilistic method combining a latent hierarchical mixture model with variational autoencoders to infer trajectories from posterior approximations. VITAE is computationally scalable and can adjust for confounding covariates to integrate multiple datasets. We show that VITAE outperforms other state-of-the-art trajectory inference methods on both real and synthetic data under various trajectory topologies. We also apply VITAE to jointly analyze two single-cell RNA sequencing datasets on mouse neocortex. Our results suggest that VITAE can successfully uncover a shared developmental trajectory of the projection neurons and reliably order cells from both datasets along the inferred trajectory.

Keywords— variational inference, hierarchical mixture models, pseudotime ordering, autoencoder

1 Introduction

Single-cell genomics have become immensely popular over the past five years and are becoming indispensable tools for biologists to understand cellular diversities and cell activities (Tang et al., 2011; Tanay & Regev, 2017; Hovestadt et al., 2019). Single-cell RNA sequencing (scRNA-seq), simultaneously measuring tens of thousands of RNAs inside individual cells, is one of the most mature and widely used technology in single-cell genomics. The development of the scRNA-seq technology has boosted new scientific discoveries in many fields, including immunology, cancer studies, neurology, and developmental and stem cell biology (Chen et al., 2019; Fan et al., 2020; Cembrowski, 2019; Griffiths et al., 2018; Kumar et al., 2017).

Many biological processes, such as differentiation, immune response, or cancer expansion, can be described and represented as continuous dynamic changes in cell-type or cell-state space (Lähnemann et al., 2020). Instead of belonging to distinct cell types, the cells can exhibit a continuous spectrum of states and be involved in transitions between different cellular states. As scRNA-seq measures cells at the single-cell resolution, it has become a valuable tool to understand these dynamic processes. Trajectory inference (TI) of scRNA-seq is to study the contiguous states of the measured cells, to infer the underlying developmental trajectories, and to computationally order these cells along the trajectories (Trapnell et al., 2014). Unlike assigning a discrete group ID to each cell, TI methods position each cell onto the inferred continuous trajectories.

Researchers have developed various computational tools for TI. Many of them are from a graphical point of view, exploiting cells' similarities calculated by their gene expressions from scRNA-seq (Qiu et al., 2017; Street et al., 2018; Wolf et al., 2019). The directions along trajectories can also be determined when the trajectories' roots are given based on external biological knowledge. However, since scRNA-seq only captures a static snapshot of each cell's transcriptome at some specific time points, the true temporal lineages or trajectories of the cells that biologists aim to find are not always identifiable from scRNA-seq (Weinreb et al., 2018; Tritschler et al., 2019). Even though there have been more than 70 computational methods proposed for TI (Saelens et al., 2019), to the best of our knowledge, most of them are not based on an explicit or coherent statistical model, and there is not even a clear definition of the "trajectory" that we can identify and estimate from scRNA-seq. As a consequence, it is vague to discuss the efficiency and accuracy in TI, let alone to have useful statistical inference and uncertainty quantification for the estimated trajectories and cell orderings. In addition, as scRNA-seq datasets continue to expand (Svensson et al., 2020), multiple individuals and datasets are becoming available for the same or closely related biological processes. There is not yet an existing TI method that can simultaneously integrate multiple scRNA-seq datasets and infer a shared underlying trajectory.

In this paper, we build a statistical framework to approximate the distribution of scRNA-seq data when the cells evolve along some underlying trajectories. We propose a new method, **VITAE** (**V**ariational **I**nference for **T**rajectory by **A**uto**E**ncoder), to perform TI. VITAE combines a hierarchical mixture model, describing the trajectory structure in a low-dimensional space, with deep neural networks that nonlinearly map the low-dimensional space to the high-dimensional observed RNA counts. Our framework provides an explicit definition of the trajectory backbone and the "pseudotime" of each cell, which are identifiable from scRNA-seq. We can infer all types of trajectories, allowing loops and disconnected states. With the approximate variational inference, we also provide scalable uncertainty quantification on both the inferred trajectory backbone and the estimated cell positions. Finally, we can simultaneously adjust for known confounding variables, such as the cell-cycle and batch effects, and reliably infer a shared trajectory from multiple scRNA-seq datasets.

The structure of the paper is as follows. Section 2 provides a review of existing TI methods. Section 3 describes the VITAE framework and model training. Section 4 discusses the approximate variational inference with VITAE for trajectory inference and uncertainty quantification. Section 5 evaluates the performance of VITAE on both real and synthetic data. Section 6 studies the developmental trajectory of cells in the mouse neocortex by a joint analysis of two scRNA-seq datasets. A Python package implementing VITAE is available at `https://github.com/jaydu1/VITAE`.

2 Review of Existing TI Methods

Since the first TI method for scRNA-seq proposed by Trapnell et al. (2014), many have been developed in the past few years. Recently, Saelens et al. (2019) performed a comprehensive evaluation comparing and benchmarking 45 TI methods on both real and synthetic datasets with various evaluation metrics. Some of these methods can only infer specific types of the trajectory topology, and some may not be scalable to handle large datasets. Among the 45 TI methods, PAGA (Wolf et al., 2019), Slingshot (Street et al., 2018), and the Monocle series (Trapnell et al., 2014; Qiu et al., 2017; Cao et al., 2019) performed the best overall, and are among the most popular TI methods in practice.

Many TI methods, including Monocle, Slingshot, and PAGA, infer the trajectories from a graphical perspective. These methods typically start with some dimension reduction of the observed data and then fit "trees" or "curves" in the reduced space that can best "connect" the cells. They differ in the algorithmic choices to perform dimension reduction and to construct the connectivity graph. For example, Monocle 1 (Trapnell et al., 2014) uses independent component analysis for dimension reduction, then infers the trajectory by building a minimum spanning tree (MST) directly on the cells. Monocle 2 (Qiu et al., 2017) later improves Monocle 1 by introducing cell centroids and a reversed graph embedding on the centroids with the cells projecting onto the embedded space. Similarly, Slingshot (Street et al.,

2018) starts with a given dimension reduction and cell clustering; however, it finds an MST on the cluster centers and simultaneous fits principal curves to infer each cell's pseudotime. PAGA (Wolf et al., 2019) also starts with clustering and dimension reduction. Instead of building MST, it then partitions, prunes and connects the clusters based on a generic statistical model of connectivity among the clusters. Based on PAGA, Monocle 3 (Cao et al., 2019) is recently proposed to learn a principal graph on each of the PAGA partitions to further obtain a finer trajectory. These graph-based TI methods have shown to be successful algorithms providing biologically meaningful trajectory estimation on many datasets. A drawback of these approaches is that they lack a coherent model for the data distribution and an explicit definition of the "trajectory" and "pseudotime" that they aim to estimate.

In contrast, some other methods use a Gaussian process probabilistic model (Campbell & Yau, 2016; Ahmed et al., 2019) to explicitly model the data distributions with a pseudotemporal ordering of the cells. Though these methods have a clear objective and can perform uncertainty quantification, they are restricted to inferring only linear trajectories, where cells have a total order along the linear path. As they do not allow for any branch, they can only be applied to specific dynamic processes. Another type of TI methods (Marco et al., 2014; Schiebinger et al., 2019; Lin & Bar-Joseph, 2019; Tran & Bader, 2020) make use of the temporal information of the cells with a time-series model when the cells are collected from a sequence of time points. For instance, Waddington-OT (Schiebinger et al., 2019) infers developmental trajectories by temporal couplings of the cells with the optimal transport and CSHMM (Lin & Bar-Joseph, 2019) uses a continuous-state hidden Markov model to assign cells to developmental paths. Though the collection time of the cells is often informative for their pseudotime ordering, they may not necessarily be positively correlated. In addition, the estimation of the temporal coupling or the hidden Markov model may be hard to achieve satisfying accuracy, taking into account the technical noise, batch effects, sampling bias, and the high dimensionality of scRNA-seq data.

Recently, the RNA velocity, which is a vector representing the transcriptional dynamics of each cell, has shown its great potential in inferring the cellular differentiation and thus the developmental trajectory of single cells (La Manno et al., 2018; Bergen et al., 2020). Instead of only using the matrix of total RNA counts in scRNA-seq, the estimation of RNA velocities is through both the matrices of spliced and unspliced RNAs. The ratio between spliced and unspliced RNAs provides exciting extra information in identifying the underlying temporal structure of the cells' dynamic process. However, using such information needs a completely different framework and is beyond the scope of this paper.

Motivated by the graph-based TI methods, we propose a probabilistic model endowing the whole procedure with statistical inference and estimate the model with the powerful variational autoencoders, which is computational efficient and flexible to simultaneously adjust for confounding covariates. Deep learning has received tremendous attention from all communities, of which generative modeling with specific priors is well developed as a large and essential branch (Kingma & Welling, 2014; Sohn et al., 2015; Jiang et al., 2017; Zong et al., 2018). For scRNA-seq, deep generative models are also widely applied for various purposes, such as cell clustering (Grønbech et al., 2020; Li et al., 2020), batch correction (Huang et al., 2020), data denoising (Wang et al., 2019; Eraslan et al., 2019) and many other tasks.

3 Model and Estimation

3.1 A hierarchical mixture model for the trajectory structure

Inspired by the common trajectory model proposed in Saelens et al. (2019), we start with the trajectory backbone defined on a complete graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with vertices $\mathcal{N}(\mathcal{G})$ and edges $\mathcal{E}(\mathcal{G})$ (Figure 1a). Let the k vertices in $\mathcal{N}(\mathcal{G})$ be the distinct cell states, we use an edge between two vertices to represent the transitioning between two states. To model the scenario that one cell either belongs to a specific state or is developing from one to another, we assume that each cell is positioned either on one vertex or edge in $\mathcal{G}$. Specifically, let $\tilde{\boldsymbol{w}}_i \in [0, 1]^k$ be the position of cell i on the backbone, we

have

$$\tilde{\boldsymbol{w}}_i = \begin{cases} \boldsymbol{e}_j & \text{if cell } i \text{ is on vertex } j \in \{1, \cdots, k\} \\ w_i \boldsymbol{e}_{j_1} + (1 - w_i)\boldsymbol{e}_{j_2} & \text{if cell } i \text{ is on the edge between vertices } j_1 \text{ and } j_2 \ (j_1 \neq j_2) \end{cases}$$

where $\boldsymbol{e}_j$ is a one-hot vector with jth element 1 and all other elements 0, and the scalar $w_i \in [0, 1]$ describes the relative position of cell i if it is on an edge.

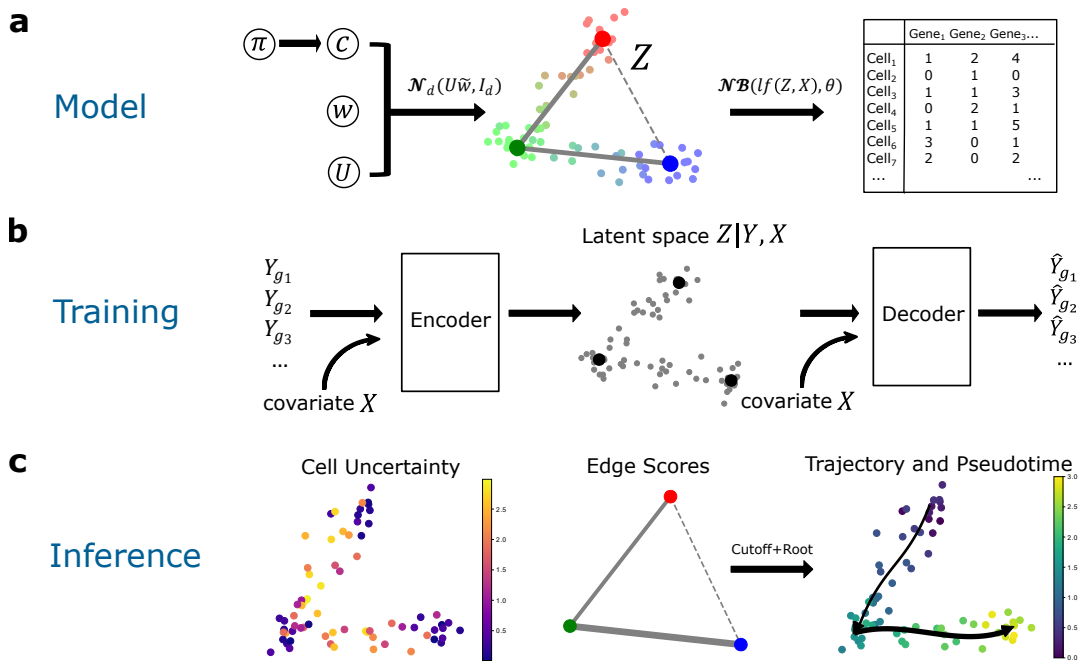

Figure 1: An overview of VITAE. **(a)** Model framework: we describe the latent representations of each cell by a hierarchical model, which are nonlinearly mapped to the high-dimensional observed count data. **(b)** Model training: we use neural networks to approximate the nonlinear mapping to the observed data (decoder) and the posterior distributions of the latent space (encoder), and adjust for confounding covariates. **(c)** Trajectory inference: we approximate the posterior distributions to infer the trajectory backbone by assigning edge scores (width of the gray lines), quantify the uncertainty in the estimated cell positions along the trajectory, and estimate the pseudotime of each cell.

The basic goal of our trajectory inference is to infer the trajectory backbone $\mathcal{B}$ - a subgraph of $\mathcal{G}$ - whose vertices and edges have positive proportions of cells:

$$\mathcal{N}(\mathcal{B}) = \left\{ j \in \mathcal{N}(\mathcal{G}) : \sum_i \mathbb{1}_{\{\tilde{w}_{ij}=1\}} > 0 \right\}$$

$$\mathcal{E}(\mathcal{B}) = \left\{ (j_1, j_2) \in \mathcal{E}(\mathcal{G}) : \sum_i \mathbb{1}_{\{\tilde{w}_{ij_1}>0, \tilde{w}_{ij_2}>0\}} > 0 \right\}$$

Meanwhile, we also aim to estimate the relative positions $\tilde{\boldsymbol{w}}_i$ of each cell on this trajectory backbone, and as in other TI methods, the pseudotime order of the cells along the trajectory. Intuitively, the pseudotime corresponds to the biological progression of a cell through the dynamic process (such as cell differentiation) that results in the trajectory structure (Trapnell et al., 2014). In TI, it refers to the cell's geodesic distance to the root along the trajectory.

Here we give a formal definition of the pseudotime in our framework. Given a root vertex k_0, we first define

pseudotime for each vertex. Assume that each edge $\ell \in \mathcal{E}(\mathcal{B})$ is associated with a duration b_ℓ. Without external information, we set $b_\ell = 1$. The pseudotime for the vertex j can be defined as

$$o_j = \begin{cases} \min \left\{ \sum_{\ell \in \mathcal{L}} b_\ell : \mathcal{L} \subset \mathcal{E}(\mathcal{B}) \text{ forms a path from } k_0 \text{ to } j \right\} \\ \infty \quad \text{if no such } \mathcal{L} \text{ exists} \end{cases} \tag{1}$$

Let $\boldsymbol{o} = (o_1, \cdots, o_k)$, then the pseudotime of the cell i is defined as

$$T_i = \boldsymbol{o}^\top \tilde{\boldsymbol{w}}_i \tag{2}$$

Since $\tilde{\boldsymbol{w}}_i$ has at most two nonzero elements, the pseudotime T_i of a cell equals the pseudotime of the vertex that it is on or the weighted average of the pseudotime of the two vertices of the edge that it belongs to. The pseudotime is only well-defined when there are no loops in $\mathcal{B}$. When there are loops, though the pseudotime T_i is still computable under our definition, we do not think that it is a biologically meaningful quantity to pursue.

Now we relate the observed scRNA-seq data with the underlying trajectory backbone graph $\mathcal{B}$. Let $\boldsymbol{Y}_i = (y_{i1}, \cdots, y_{iG})$ be the observed gene expressions of G genes in cell i from scRNA-seq. Though $\boldsymbol{Y}_i$ is the vector of observed counts with complicated dependence across genes, we assume that the dependence can be explained by latent Gaussian variables $\boldsymbol{Z}_i \in \mathbb{R}^d$, which is in a space with a much lower dimension and associated with the trajectory backbone $\mathcal{B}$. We also take into account the effects of known confounding covariates $\boldsymbol{X}_i \in \mathbb{R}^s$ (such as cell-cycle or batch effects, which we take as deterministic variables) on $\boldsymbol{Y}_i$. Thus, we assume the following latent variable model on the observed counts:

$$\begin{aligned} \boldsymbol{Z}_i | \tilde{\boldsymbol{w}}_i &\sim \mathcal{N}_d(\boldsymbol{U}\tilde{\boldsymbol{w}}_i, \boldsymbol{I}_d) \\ Y_{ig} | \boldsymbol{Z}_i, \boldsymbol{X}_i &\overset{\text{ind}}{\sim} \mathcal{NB}\left(l_i f_g(\boldsymbol{Z}_i, \boldsymbol{X}_i), \theta_g\right), \quad g = 1, 2, \cdots, G \end{aligned} \tag{3}$$

Here $\boldsymbol{U} \in \mathbb{R}^{d \times k}$ is the unknown matrix of the vertices positions which, together with the cells' relative positions on $\mathcal{B}$, determines the means of $\boldsymbol{Z}_i$. Then, we assume that the observed counts $\boldsymbol{Y}_i$ nonlinearly depend on the latent variables $\boldsymbol{Z}_i$. The scalar l_i is the known library size of cell i and each $f_g : \mathbb{R}^{d+s} \longrightarrow \mathbb{R}, g = 1, 2, \cdots, G$ is an unknown nonlinear function involving the confounding covariate vector $\boldsymbol{X}_i$. The unknown parameters θ_g are gene-specific dispersion parameters of the Negative Binomial (NB) distribution. Notice that though the edges are assumed to be linear lines in the latent space, they are likely curves in the observed data space via the nonlinear mappings $\{f_g(\cdot),\ g = 1, 2, \cdots, G\}$.

The assumption of using the NB distribution to model scRNA-seq data is based on a detailed review and discussion in Agarwal et al. (2020). Specifically, for scRNA-seq with the unique molecular identifier (UMI) counts, the Negative Binomial distributions can well describe the stochasticity in scRNA-seq, accounting for both the biological and technical noise. However, for scRNA-seq data without UMIs, a Negative Binomial distribution may not be adequate. In that scenario, we assume that the observed counts follow a zero-inflated Negative Binomial (ZINB) distribution as in Risso et al. (2018):

[for non-UMI counts]
$$Y_{ig} | \boldsymbol{Z}_i, \boldsymbol{X}_i \overset{\text{ind}}{\sim} \phi_{ig}\delta_0 + (1 - \phi_{ig})\mathcal{NB}\left(l_i f_g(\boldsymbol{Z}_i, \boldsymbol{X}_i), \theta_g\right) \tag{4}$$

where ϕ_{ig} is the zero inflation probability modeled by a nonlinear mapping function $h_g(\boldsymbol{Z}_i, \boldsymbol{X}_i)$ for $g = 1, \cdots, G$.

The above model defines the trajectory backbone $\mathcal{B}$, cells positions $\tilde{\boldsymbol{w}}_i$, and pseudotime T_i that are identifiable from scRNA-seq data. However, estimating these quantities are still challenging as the vertices and edges are defined on a latent space. Thus, we further impose a hierarchical model on $\tilde{\boldsymbol{w}}_i$ to simplify model estimation. First, we introduce a latent categorical variable c_i as the index of all edges and vertices. Specifically, let c_i take values in $\{1, 2, \ldots, K\}$

where $K = k(k+1)/2$ is the number of all possible edges and vertices in $\mathcal{G}$. It associates with $\mathcal{G}$ through the following categorical assignment symmetric matrix,

$$\boldsymbol{C} = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ k \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \cdots & k \end{array} \\ \begin{pmatrix} 1 & 2 & \cdots & k \\ 2 & k+1 & \cdots & 2k-1 \\ \vdots & \vdots & \ddots & \vdots \\ k & 2k-1 & \cdots & K \end{pmatrix} \end{array}. \tag{5}$$

When c_i equals to $C_{j_1 j_2}$ for $j_1 \neq j_2$, the cell i is at the edge between vertex j_1 and vertex j_2. When c_i equals to C_{jj}, the cell i is at the vertex j. Then we assume the following mixture prior on $\tilde{\boldsymbol{w}}_i$:

$$\begin{aligned} & w_i \overset{\text{i.i.d.}}{\sim} \text{Uniform}(0,1) \\ & c_i \overset{\text{i.i.d.}}{\sim} \text{Multinomial}(1, \boldsymbol{\pi}), \quad \boldsymbol{\pi} \in [0,1]^K \\ & w_i \perp\!\!\!\perp c_i \\ & \tilde{\boldsymbol{w}}_i = w_i \boldsymbol{a}_{c_i} + (1-w_i)\boldsymbol{b}_{c_i} \end{aligned} \tag{6}$$

where $\boldsymbol{a}_c = \boldsymbol{e}_{j_1}$ and $\boldsymbol{b}_c = \boldsymbol{e}_{j_2}$ if $c = C_{j_1 j_2}$ for $j_1 \leq j_2$.

To summarize, combining models (3) and (6), we obtain a hierarchical mixture model for scRNA-seq with an underlying trajectory structure. The unknown parameters are the mean positions of the k cell types in the latent space $\boldsymbol{U}$, the prior probabilities of the K categories $\boldsymbol{\pi}$, the mapping functions $f_g(\cdot), g = 1, 2, \cdots, G$ and the dispersion parameters θ_g (and also $h_g(\cdot)$ for non-UMI counts).

3.2 Variational autoencoder for approximating the posterior distributions

Our hierarchical mixture model for scRNA-seq is well suited for the complex scRNA-seq data as it connects the Gaussian latent variable $\boldsymbol{Z}_i$ with the observed sparse counts $\boldsymbol{Y}_i$ through non-linear mapping functions $f_g(\cdot)$. To introduce a wide class of mapping functions, we model $f_g(\cdot)$ by a neural network and further combine our model with the Variational Autoencoder (VAE) (Kingma & Welling, 2014) for approximating the posterior distributions. Following the variational Bayes approach in VAE and the conditional VAE (Sohn et al., 2015), we use Gaussian distributions to approximate the intractable posterior distribution of $\boldsymbol{Z}_i$ given the observed data $\boldsymbol{Y}_i$ and confounding variables $\boldsymbol{X}_i$:

$$\boldsymbol{Z}_i | \boldsymbol{Y}_i, \boldsymbol{X}_i \dot{\sim} \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{Y}_i, \boldsymbol{X}_i}, \boldsymbol{\Sigma}_{\boldsymbol{Y}_i, \boldsymbol{X}_i}\right). \tag{7}$$

Here the "posterior" mean $\boldsymbol{\mu}_{\boldsymbol{Y}_i, \boldsymbol{X}_i}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{Y}_i, \boldsymbol{X}_i} = \text{diag}(\sigma^2_{\boldsymbol{Y}_i, \boldsymbol{X}_i, 1}, \cdots, \sigma^2_{\boldsymbol{Y}_i, \boldsymbol{X}_i, d})$ are functions of $(\boldsymbol{Y}_i, \boldsymbol{X}_i)$. To guarantee flexibility, they are also modeled by a neural network. As shown in Figure 1b, the neural network for $\boldsymbol{\mu}_{\boldsymbol{Y}_i, \boldsymbol{X}_i}$ and $\boldsymbol{\Sigma}_{\boldsymbol{Y}_i, \boldsymbol{X}_i}$ is the encoder, and the network for $f_g(\boldsymbol{Z}_i, \boldsymbol{X}_i)$ is the decoder. For non-UMI data, as in Eraslan et al. (2019), the functions $h_g(\cdot)$ for zero-inflation parameters are modeled by a neural network in the decoder and share the same hidden layers as $f_g(\cdot)$.

With all the above setup, we can lower bound the log-likelihood of each observation $\boldsymbol{Y}_i$ as

$$\begin{aligned} \log p(\boldsymbol{Y}_i | \boldsymbol{X}_i) &= \log \left[\mathbb{E}_{p(\boldsymbol{Z}_i | \boldsymbol{X}_i)} p(\boldsymbol{Y}_i | \boldsymbol{Z}_i, \boldsymbol{X}_i)\right] \\ &= \log \left[\mathbb{E}_{q(\boldsymbol{Z}_i | \boldsymbol{Y}_i, \boldsymbol{X}_i)} \left(\frac{p(\boldsymbol{Y}_i | \boldsymbol{Z}_i, \boldsymbol{X}_i) p(\boldsymbol{Z}_i)}{q(\boldsymbol{Z}_i | \boldsymbol{Y}_i, \boldsymbol{X}_i)}\right)\right] \end{aligned}$$

$$
\begin{aligned}
&\geq \mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\left[\log\left(\frac{p(\boldsymbol{Y}_i|\boldsymbol{Z}_i,\boldsymbol{X}_i)p(\boldsymbol{Z}_i)}{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\right)\right] \\
&= \mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i,\boldsymbol{X}_i) - D_{\mathrm{KL}}(q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)\|p(\boldsymbol{Z}_i)),
\end{aligned} \tag{8}
$$

where $p(\boldsymbol{Y}_i|\boldsymbol{Z}_i,\boldsymbol{X}_i)$ is the conditional distribution in model (3) or (4), and $q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)$ is the Gaussian approximation of the posterior distribution in (7). This lower bound is often referred to as the evidence lower bound (ELBO) (Kingma & Welling, 2014), where the first term $\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i,\boldsymbol{X}_i)$ denotes the reconstruction likelihood and the second term $-D_{\mathrm{KL}}(q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)\|p(\boldsymbol{Z}_i))$ behaves as a regularizer. The difference between (8) and regular conditional VAE is that we have a mixture model for the trajectory structure encoded in the term $p(\boldsymbol{Z}_i)$, which encourages the posteriors of $\boldsymbol{Z}_i$ to lie along linear edges and vertices. The resulting loss function for one cell is defined as the negative modified ELBOs in (8),

$$
\mathcal{L}(\boldsymbol{Y}_i;\boldsymbol{X}_i,\Theta) = -\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i,\boldsymbol{X}_i) + \beta D_{\mathrm{KL}}(q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)\|p(\boldsymbol{Z}_i)), \tag{9}
$$

where Θ is the set of trainable parameters in our model and β is a tuning parameter to balance the reconstruction error and regularization, which is introduced in Higgins et al. (2016) as the β-VAE. To better adjust for confounding covariates $\boldsymbol{X}_i$, we add another penalty, as introduced in Huang et al. (2020), to obtain our final loss function aggregated over all N cells as:

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{Y}_1,\cdots,\boldsymbol{Y}_N;\boldsymbol{X}_1,\cdots\boldsymbol{X}_N,\Theta) = &-(1-\alpha)\sum_{i=1}^{N}\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i,\boldsymbol{X}_i) \\
&-\alpha\sum_{i=1}^{N}\log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i=\boldsymbol{0}_d,\boldsymbol{X}_i) + \beta\sum_{i=1}^{N}D_{\mathrm{KL}}(q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)\|p(\boldsymbol{Z}_i)).
\end{aligned} \tag{10}
$$

where α is another tuning parameter. Setting $\alpha > 0$ encourages our decoder to reconstruct $\boldsymbol{Y}_i$ using only information from $\boldsymbol{X}_i$, which can help to decorrelate $\boldsymbol{X}_i$ from $\boldsymbol{Z}_i$ so that one can remove the confounding effects of $\boldsymbol{X}_i$ in $\boldsymbol{Z}_i$ more thoroughly.

All three terms in (10) can be approximated by the Monte Carlo method efficiently, as we show in Appendix B. Specifically, though the marginal density function $p(\boldsymbol{Z}_i)$ involves in a complex hierarchical mixture model (6), it still has a closed form representation. Define $\boldsymbol{\alpha}_{\boldsymbol{z}c} = \boldsymbol{U}(\boldsymbol{b}_c - \boldsymbol{a}_c)$ and $\boldsymbol{\beta}_{\boldsymbol{z}c} = \boldsymbol{z} - \boldsymbol{U}\boldsymbol{b}_c$ for any $c = 1, 2, \cdots, K$ where $\boldsymbol{a}_c$ and $\boldsymbol{b}_c$ are constants defined in (6). Then the marginal density of $\boldsymbol{Z}_i$ has the form

$$
p(\boldsymbol{z}) = \sum_{c=1}^{K} p(\boldsymbol{z}|c_i=c)\pi_c = \sum_{c\in\mathcal{N}(\mathcal{G})}\pi_c\varphi_d(\boldsymbol{U}\boldsymbol{b}_c) + \sum_{c\in\mathcal{E}(\mathcal{G})}\frac{\pi_c\sigma_{\boldsymbol{z}c}e^{\frac{t_{\boldsymbol{z}c}}{2}}}{(2\pi)^{\frac{d}{2}-1}}\left[\Phi\left(\frac{1-\nu_{\boldsymbol{z}c}}{\sigma_{\boldsymbol{z}c}}\right) - \Phi\left(-\frac{\nu_{\boldsymbol{z}c}}{\sigma_{\boldsymbol{z}c}}\right)\right].
$$

with

$$
\nu_{\boldsymbol{z}c} = -\frac{\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top\boldsymbol{\beta}_{\boldsymbol{z}c}}{\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top\boldsymbol{\alpha}_{\boldsymbol{z}c}}, \quad \sigma_{\boldsymbol{z}c}^2 = \frac{1}{\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top\boldsymbol{\alpha}_{\boldsymbol{z}c}}, \quad t_{\boldsymbol{z}c} = -\boldsymbol{\beta}_{\boldsymbol{z}c}^\top\boldsymbol{\beta}_{\boldsymbol{z}c} + \frac{(\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top\boldsymbol{\beta}_{\boldsymbol{z}c})^2}{\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top\boldsymbol{\alpha}_{\boldsymbol{z}c}},
$$

where $\varphi_d(\cdot)$ and $\Phi(\cdot)$ are the probability density function and the cumulative density function of d-dimensional and univariate standard Gaussians respectively. With these derivations, the optimization can be efficiently done via stochastic backpropagation (Rezende et al., 2014) on mini-batches of data, with the commonly used amortized variation inference (Stuhlmüller et al., 2013) for VAE.

3.3 Model estimation with practical considerations

For the gene expressions as input, by default, we select $G = 2000$ most highly variable genes following the procedure in Seurat (Stuart et al., 2019), one of the most widely used software in scRNA-seq. We also preprocess the gene expressions following Seurat (see Appendix A) and use the normalized, log-transformed, and scaled gene expressions as the inputs of our neural network. By default, we set $\beta = 2$ and $\alpha = 0.1$ (when there are confounding covariates), which we find to give the best results in practice. We use some $\beta > 1$ to strengthen the regularization on $\boldsymbol{Z}_i$ especially when its dimension is much smaller than the dimension of $\boldsymbol{Y}_i$.

As the stochastic gradient descent algorithm optimizing our loss (10) generally results in a locally optimal solution, we need a good initialization of our parameters, especially for $\boldsymbol{U}$ and $\boldsymbol{\pi}$ defined in (3) and (6). Also, our framework requires a pre-determined number of states k. Inspired by other existing TI methods, we design a three-step algorithm for model initialization and estimation.

The first step is pretraining, where we train the model with $\beta = 0$, to only minimize the reconstruction loss which does not involve the unknown parameters $\boldsymbol{\pi}$ and $\boldsymbol{U}$. The purpose of this step is to obtain better weights for the encoder and the decoder and to get an initial low-dimensional representation, which can be used to initialize $\boldsymbol{U}$ and determine k in the next step.

The second step is to initialize the latent space. We perform cell clustering with the Louvain algorithm (Blondel et al., 2008) on the estimated posterior means $\widehat{\boldsymbol{\mu}}_{\boldsymbol{Y}_i,\boldsymbol{X}_i}$ of $\boldsymbol{Z}_i$ after pretraining. With a default resolution of 0.8 as in Seurat, we let Louvain automatically determine the number of clusters, which we set as the value of k, the number of states in our framework. As in Seurat and other clustering algorithms, we suggest users trying different resolutions (typically between 0.4 to 1.2) in practice to obtain satisfying clustering results. We initialize $\boldsymbol{U}$ with the cluster centers. As $\boldsymbol{\pi}$ involves both the k vertices and $k(k-1)/2$ edges, we have no information yet and just uniformly initialize $\boldsymbol{\pi}$ in this step.

The last step is to train our whole network, optimizing the loss function (10). To obtain a better initialization of $\boldsymbol{\pi}$, we warm up our optimization algorithm for the first 5 epochs, where we only train the parameters $\widehat{\boldsymbol{\pi}}$ and $\widehat{\boldsymbol{U}}$ in the latent space while keeping the parameters of the encoder and the decoder frozen. We find the warmup step a useful trick for complicated large datasets with many cell types, such as our real data example in Section 6. During both the pretraining and training steps, the optimization is early stopped when the evaluation loss decreases sufficiently slow.

Finally, to make VITAE scalable to efficiently handle large datasets and have a comparable computational cost as other TI methods, we accelerate VITAE for large datasets by reducing the dimension of the input. We replace the expression of G genes with its top R ($R = 64$ by default) principle component (PC) scores $\boldsymbol{F}_i$. The output of the decoder is also replaced by the reconstruction of $\boldsymbol{F}_i$, and the likelihood of $p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i)$ in the loss function(10) is replaced by Gaussian densities assumed on $\boldsymbol{F}_i$ (Appendix B). As shown in Section 5, the accelerated VITAE with Gaussian likelihood provides comparable results with our original Negative Binomial likelihood-based VITAE in general, while it can be much faster than the original scheme. By default, both the encoder and decoder have one hidden layer, with 32 and 16 units for the accelerated and likelihood-based VITAE respectively. The bottleneck layer has a dimension of 8. The hidden layers are all fully-connected with the leaky rectified linear activation function (Maas et al., 2013; Huang et al., 2020) and Batch Normalization (Ioffe & Szegedy, 2015).

4 Trajectory Inference from Posterior Approximations

After the training step, the model returns the estimated parameters, including $\widehat{\boldsymbol{U}}$, $\widehat{\boldsymbol{\pi}}$, the encoder $\widehat{q}(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$, and the decoder $\widehat{f}(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$. Replacing the true posterior density $p(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$ with the approximate posterior density $\widehat{q}(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$, we can also obtain an approximation of the posterior distribution of (w_i, c_i) for each cell. Specifically, let $\boldsymbol{z}_i^{(1)}, \cdots, \boldsymbol{z}_i^{(L)}$ be L ($L = 300$ by default) random samples from $\widehat{q}(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$, and use the fact that

$(w_i, c_i) \perp\!\!\!\perp (\boldsymbol{Y}_i, \boldsymbol{X}_i)|\boldsymbol{Z}_i$, we can approximate the posterior density of (w_i, c_i) as

$$\widehat{p}(w, c|\boldsymbol{Y}_i, \boldsymbol{X}_i) = \frac{1}{L}\sum_{l=1}^{L} \widehat{p}(w, c|\boldsymbol{Z}_i = z_i^{(l)})$$

where $\widehat{p}(w, c|\boldsymbol{Z}_i)$ is obtained by plugging in $\widehat{\boldsymbol{U}}$ and $\widehat{\boldsymbol{\pi}}$ into the true posterior density of (w_i, c_i) given $\boldsymbol{Z}_i$. Notice that the approximate posterior distribution only depends on posterior samples of the latent variables $\boldsymbol{Z}_i$. Similarly, we can get the approximate posterior distributions of c_i, the edge or vertex the cell belongs to, as

$$\widehat{p}(c_i = c|\boldsymbol{Y}_i, \boldsymbol{X}_i) = \frac{1}{L}\sum_{l=1}^{L} \widehat{p}(c_i = c|\boldsymbol{Z}_i = z_i^{(l)})$$

For the cell position $\tilde{\boldsymbol{w}}_i$ on the graph $\mathcal{G}$, as it is a function of w_i and c_i as defined in (6), we can also efficiently obtain the mean $\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}$ and the diagonal elements of the covariance matrix $\Sigma_{\tilde{\boldsymbol{w}}_i}$ of its posterior distribution $\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$. For details in calculating the approximate posterior distributions, see Appendix C. Now we discuss how to infer the trajectory backbone and cell positions along the trajectory with these posterior approximations.

4.1 Infer the trajectory backbone $\mathcal{B}$

The total number of categories $K = O(k^2)$ can be large, even with a moderate number of k. While the trajectory backbone $\mathcal{B}$ typically only sparsely involves a few edges, our estimated $\widehat{\boldsymbol{\pi}}$ is often a dense vector. Inspired by Frühwirth-Schnatter & Malsiner-Walli (2019) on Bayesian Gaussian mixture models, to encourage sparsity, we infer the nonzero edges a posteriori from the data. Specifically, we define a score for each edge, quantifying the strength of the evidence that the edge exists:

$$s_{j_1 j_2} = \frac{|\{i : c_i = C_{j_1 j_2}\}|}{|\{i : \boldsymbol{e}_{j_1}^\top \tilde{\boldsymbol{w}}_i > 0.5 \text{ or } e_{j_1}^\top \tilde{\boldsymbol{w}}_i > 0.5\}|}.$$

where the denominator is added to make sure that we can capture the continuous transitions between cell states even when these states only involve a small proportion of cells in the cell population. Even though $\widehat{\boldsymbol{\pi}}$ is dense, $s_{j_1 j_2}$ can be significantly nonzero for much fewer edges. From another point of view, an edge score is some "test statistics" to evaluate whether the edge exists. To make our algorithm scalable, in practice, instead of obtaining the posterior distributions of $s_{j_1 j_2}$ to dertermine whether it is significantly nonzero or not, we simply use a deterministic version of the edge score as

$$\tilde{s}_{j_1 j_2} = \frac{|\{i : \tilde{c}_i = C_{j_1 j_2}\}|}{|\{i : \boldsymbol{e}_{j_1}^\top \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i} > 0.5 \text{ or } \boldsymbol{e}_{j_1}^\top \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i} > 0.5\}|}$$

where

$$\tilde{c}_i = \underset{c \in \{1, 2, \cdots, K\}}{\arg\max} \ \widehat{p}(c_i = c|\boldsymbol{Y}_i, \boldsymbol{X}_i)$$

and $\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}$ is the approximate posterior mean of $\tilde{\boldsymbol{w}}_i$. Other edge scores choices are discussed in Appendix D.

A larger $\tilde{s}_{j_1 j_2}$ indicates higher confidence assuring that the edge exits in the trajectory backbone $\mathcal{B}$. In practice, we include an edge (j_1, j_2) into the estimated backbone $\widehat{\mathcal{B}}$ if $\tilde{s}_{j_1 j_2} \geq s_0$, where the cutoff s_0 is 0.01 by default. When we are certain that there are no loops in the trajectory, inspired by Trapnell et al. (2014), we further prune $\widehat{\mathcal{B}}$ as the MST of the unpruned graph with $\tilde{s}_{j_1 j_2}$ as edge weights. This typically results in a cleaner shape of our estimated trajectory.

4.2 Project cells onto the inferred trajectory backbone

To obtain the position of each cell on the inferred trajectory, we further project $\tilde{\boldsymbol{w}}_i$ onto $\widehat{\mathcal{B}}$. Given the approximate posterior distributions of $\tilde{\boldsymbol{w}}_i$, we would find a point estimate for the best position of each cell i on $\widehat{\mathcal{B}}$. Specifically, for each cell, we aim to solve the following optimization problem:

$$\begin{aligned} \widehat{\tilde{\boldsymbol{w}}}_i = \arg\min_{\boldsymbol{w}} \quad & \mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \|\tilde{\boldsymbol{w}}_i - \boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & \text{support}(\boldsymbol{w}) \subseteq \widehat{\mathcal{B}}, \\ & \|\boldsymbol{w}\|_1 = 1, \;\; \boldsymbol{w} \succeq \boldsymbol{0}. \end{aligned} \tag{11}$$

Since the support of $\tilde{\boldsymbol{w}}_i$ is restricted to the inferred trajectory backbone $\widehat{\mathcal{B}}$, only one or two entries of $\widehat{\tilde{\boldsymbol{w}}}_i$ can be nonzero, depending on whether the cell is at the vertex or edge of $\widehat{\mathcal{B}}$. Though the L_2 loss in the objective function of optimization problem (11) is not the only choice, it can result in a closed-form solution that allows fast computation.

Proposition 1. *The optimization problem (11) is equivalent to finding*

$$\begin{aligned} \widehat{\tilde{\boldsymbol{w}}}_i = \arg\min_{\boldsymbol{w}} \quad & \|\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i} - \boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & \text{support}(\boldsymbol{w}) \subseteq \widehat{\mathcal{B}}, \\ & \|\boldsymbol{w}\|_1 = 1, \;\; \boldsymbol{w} \succeq \boldsymbol{0}, \end{aligned} \tag{12}$$

where $\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}$ is the mean of $\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$. Denote the jth component of $\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}$ as μ_j and let $\mathcal{EN}(\widehat{\mathcal{B}}) = \mathcal{E}(\widehat{\mathcal{B}}) \cup \left\{(j,j) : j \in \mathcal{N}(\widehat{\mathcal{B}})\right\}$, then the best projection is given by

$$(j_1^*, j_2^*) \in \arg\max_{(j_1,j_2)\in\mathcal{EN}(\widehat{\mathcal{B}})} \quad (\mu_{j_1} - \mu_{j_2})^2 + 2(\mu_{j_1} + \mu_{j_2}) - \mathbb{1}_{\{j_1=j_2\}},$$

which reduces to be a vertex if $j_1^* = j_2^*$, and the corresponding solution $\text{proj}_{(j_1^*,j_2^*)}(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i})$ *has entries*

$$[\text{proj}_{(j_1^*,j_2^*)}(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i})]_j = \begin{cases} \mu_{j_1^*} + \frac{1}{2}\left(1 - \mu_{j_1^*} - \mu_{j_2^*} + \mathbb{1}_{\{j_1^*=j_2^*\}}\right) & j \in \{j_1^*, j_2^*\} \\ 0 & \text{otherwise}, \end{cases}$$

for $j = 1, \ldots, k$.

The proof of Proposition 1 is included in Appendix E. Intuitively, finding the optimal projection of $\tilde{\boldsymbol{w}}_i$ minimizing the L_2 loss is equivalent to simply projecting the posterior mean $\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}$. In addition, as shown in our proof of Proposition 1, most cells will either project onto an edge or an isolated vertex. Notice that the solution for the optimization problem (12) may not be unique. However, it is generally unique due to floating-point computation in practice.

Next, we also want to quantify the uncertainty of the projected position $\widehat{\tilde{\boldsymbol{w}}}_i$ as an estimate of $\tilde{\boldsymbol{w}}_i$. A general metric to evaluate the uncertainty is given by

$$\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}[d(\widehat{\tilde{\boldsymbol{w}}}_i, \tilde{\boldsymbol{w}}_i)]$$

where $d(\cdot,\cdot)$ is a metric or distance function. There can also be many choices of $d(\cdot,\cdot)$. When $d(\boldsymbol{w}, \boldsymbol{w}') = \|\boldsymbol{w} - \boldsymbol{w}'\|_2^2$ is also the L_2 loss, we obtain the projection mean square error (MSE) as

$$\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}[d(\widehat{\tilde{\boldsymbol{w}}}_i, \tilde{\boldsymbol{w}}_i)] = \|\widehat{\tilde{\boldsymbol{w}}}_i - \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}\|_2^2 + \text{tr}\left(\Sigma_{\tilde{\boldsymbol{w}}_i}\right), \tag{13}$$

which is easily computable (Appendix C). Notice that our projection MSE ignores the uncertainty in $\widehat{\mathcal{B}}$ and the

approximation error in $\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$, so it is an underestimate of the true uncertainty. However, we think that the relative magnitude of our projection MSE would still be a useful quantity to compare the projection accuracy across cells. Another pattern, as we will observe later in Section 6, is that the projection MSE is typically smaller when the cells are near vertices. We provide our understanding and a detailed discussion of this pattern in Appendix F.

Finally, we obtain the point estimate of each cell's pseudotime T_i defined in (2). Given the inferred trajectory $\widehat{\mathcal{B}}$, the user can assign a root vertex based on prior biological knowledge. We also provide an automatic root selection step following Tempora (Tran & Bader, 2020) when cells are collected from a series of time points. The idea is to choose the vertex with earliest collection time as the root. Specifically, let r_i be the collection time of cell i, we calculate the "collection time" of vertex j as

$$\gamma_j = \frac{\sum_i \widehat{\tilde{\boldsymbol{w}}}_{ij} r_i}{\sum_i \widehat{\tilde{\boldsymbol{w}}}_{ij}},$$

an weighted average of the collection time of the cells near the vertex. A vertex $k_0 \in \mathcal{N}(\mathcal{B})$ is chosen as the root if it has the smallest collection time γ_j and is not an isolated vertex.

Once the root vertex of the trajectory backbone $\widehat{\mathcal{B}}$ is obtained, it is straightforward to obtain $\widehat{\boldsymbol{o}}$, the estimated pseudotime of the vertices, by plugging in $\widehat{\mathcal{B}}$ into the definition of $\boldsymbol{o}$ in (1). Then a point estimate of the pseudotime T_i of cell i can be given as

$$\widehat{T}_i = \widehat{\boldsymbol{o}}^\top \widehat{\tilde{\boldsymbol{w}}}_i.$$

4.3 Differential gene expression along the trajectory

We also provide a simple linear regression approach to find differentially expressed genes along our inferred trajectory backbone. Compared with existing approaches (Van Buren et al., 2020) to find differentially expressed genes in trajectory inference, we focus on finding genes that are positively or negatively associated with the pseudotime ordering after adjusting for confounding covariates and provide a more scalable way to obtain the p-values of the genes.

Specifically, to find genes that are differentially expressed along the pseudotime ordering for a subset of cells $\mathcal{S}$, we work with the linear regression for each gene g:

$$Y_{ig} = \beta_{0g} + \beta_{1g}\text{rank}(T_i) + \boldsymbol{X}_i^\top \boldsymbol{\gamma}_g + e_{ig}, \quad \mathbb{E}(e_{ig}) = 0, \ \forall\, i \in \mathcal{S} \tag{14}$$

where Y_{ig} is the log-transformed and normalized count for cell i and gene g, and the linear term $\boldsymbol{X}_i^\top \boldsymbol{\gamma}_g$ is to adjust for the confounding effects of known covariates. We allow e_{ig}'s to have unequal variances as Y_{ig}'s are log-transformed and normalized counts. Intuitively, this linear regression is to estimate the correlation between gene g's expression and the pseudotime ordering after correcting for confounding effects, which is defined as

$$\rho_g = \beta_{1g} \frac{\text{sd}[\text{rank}(T_\cdot)]}{\text{sd}(Y_{\cdot g})}. \tag{15}$$

For each gene, we aim to test for the null $H_{0g} : \rho_g = 0$, or equivalently testing $H_{0g} : \beta_{1g} = 0$.

The challenge here is that the true pseudotime T_i is not observed. Instead, we only have estimated pseudotime $\widehat{T}_i$ from the data, which has unknown uncertainty and is correlated with Y_{ig}. As a consequence, obtaining a valid p-value for H_{0g} is impossible at this stage. For simplicity, we plug in $\widehat{T}_i$ and estimate β_{ig} via the ordinary least squares. Treating $\widehat{T}_i$ as the true T_i, we obtain the variance of $\widehat{\beta}_{ig}$ through the sandwich estimator allowing for heterogeneity in e_{ig}. The t-statistics is then defined as $t_g = \widehat{\beta}_{ig}/\widehat{\text{sd}}(\widehat{\beta}_{ig})$. To make a simple correction of the post-estimation bias replacing T_i with the estimated $\widehat{T}_i$, we take the p-value calibration approach following Wang et al. (2017). Specifically, instead of assuming $t_g \sim \mathcal{N}(0, 1)$ under the null, we assume that $t_g \sim \mathcal{N}(0, \sigma^2)$ where σ^2 is estimated from the median absolute deviation (MAD) of $\{t_g,\ g = 1, 2, \cdots, G\}$. To select differentially expressed genes, the calibrated p-values are further

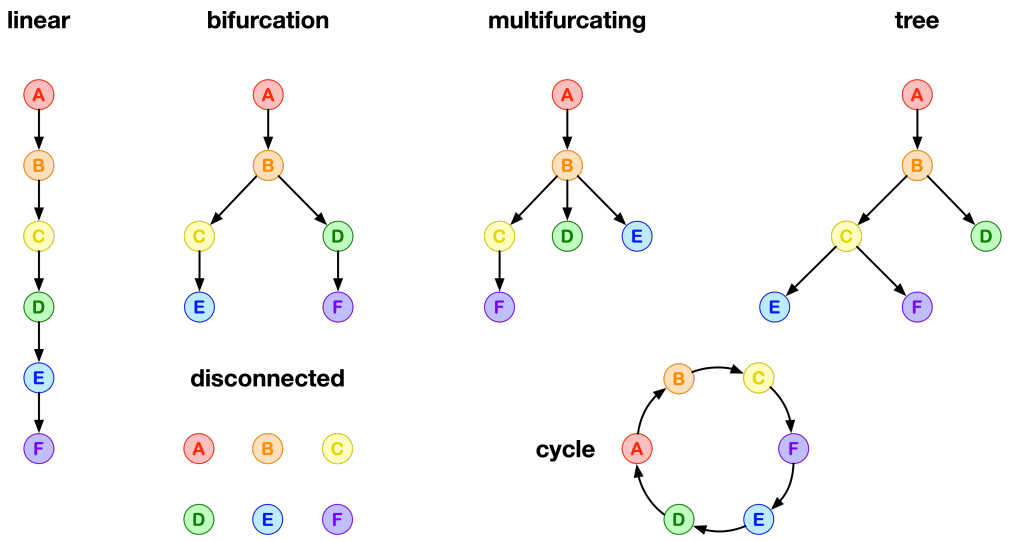

Figure 2: Six different topologies of the underlying trajectories in real and synthetic datasets.

adjusted with the Benjamini-Hochberg procedure for multiple testing correction.

Unlike our approach here, tradeSeq (Van Buren et al., 2020) fits a generalized linear regression on the raw scRNA-seq counts assuming a Negative Binomial distribution of the data. In order to find more complicated gene expression patterns along the estimated pseudotime, they regress Y_{ig} on the smoothing splines of $\widehat{T}_i$. We only focus on finding genes that are positively or negatively associated with T_i. With a linear model, we can more efficiently find differentially expressed genes. Without parallelization, to find out differentially expressed genes in Section 6, it only takes us less than 4 minutes on 5000 cells and $15k$ genes, while tradeSeq can take more than 20 minutes as evaluated in Van Buren et al. (2020).

5 Validation and Benchmarking with Real and Synthetic Datasets

In this section, we evaluate the performance of VITAE on diverse datasets. These datasets contain both real and synthetic data, with UMI and non-UMI counts and various trajectory topologies. We also use 5 different metrics to measure our accuracy in trajectory inference. Our validations mostly follow the third-party benchmarking paper Saelens et al. (2019) so that we can have a fair evaluation of VITAE.

5.1 Datasets

Following the settings in Saelens et al. (2019), which provided a comprehensive overview and guideline for TI methods, we evaluate VITAE's performance in recovering six types of trajectory topologies, as shown in Figure 2. Our benchmarking datasets include 10 real scRNA-seq data and 15 synthetic datasets, summarized in Table 1.

The datasets from real scRNA-seq studies include 9 datasets from Saelens et al. (2019). Among them, 5 datasets have "gold standard" labels according to Saelens et al. (2019) and are included to cover all trajectory types with at least 200 cells. As most "gold standard" datasets are small, we also include 4 extra datasets (*dentate*, *fibroblast*, *planaria_muscle*, *planaria_full*) with "silver standard" labels. The datasets and labels are all extracted from the Dyno platform[1] (Saelens et al., 2019), except for the dataset *dentate*, whose cells are mislabeled, and we directly extract the

[1] https://github.com/dynverse/dynbenchmark/

Table 1: Detailed summary of datasets used in experiments, including trajectory topology, number of cells N, number of genes G, number of cell types k, and sources of the dataset.

Type	Name	Count Type	Topology	N	G	k	Source
real	aging	non-UMI	linear	873	2815	3	Kowalczyk et al. (2015)
	human_embryos	non-UMI	linear	1289	8772	5	Petropoulos et al. (2016)
	germline	non-UMI	bifurcation	272	8772	7	Guo et al. (2015)
	mesoderm	non-UMI	tree	504	8772	9	Loh et al. (2016)
	cell_cycle	non-UMI	cycle	264	6812	3	Petropoulos et al. (2016)
	dentate	UMI	linear	3585	2182	5	Hochgerner et al. (2018)
	fibroblast	non-UMI	bifurcation	355	3379	7	Treutlein et al. (2016)
	planaria_muscle	UMI	bifurcation	2338	4210	3	Wolf et al. (2019)
	planaria_full	UMI	tree	18837	4210	33	Wolf et al. (2019)
	immune	UMI	disconnected	21082	18750	3	Zheng et al. (2017)
synthetic	linear_1	non-UMI	linear	2000	991	4	dyngen
	linear_2	non-UMI	linear	2000	999	4	dyngen
	linear_3	non-UMI	linear	2000	1000	4	dyngen
	bifurcating_1	non-UMI	bifurcation	2000	997	7	dyngen
	bifurcating_2	non-UMI	bifurcation	2000	991	7	dyngen
	bifurcating_3	non-UMI	bifurcation	2000	1000	7	dyngen
	trifurcating_1	non-UMI	multifurcating	2000	969	10	dyngen
	trifurcating_2	non-UMI	multifurcating	2000	995	10	dyngen
	converging_1	non-UMI	bifurcation	2000	998	6	dyngen
	cycle_1	non-UMI	cycle	2000	1000	3	dyngen
	cycle_2	non-UMI	cycle	2000	999	3	dyngen
	linear	UMI	linear	1900	1990	5	our model
	bifurcation	UMI	bifurcation	2100	1996	5	our model
	multifurcating	UMI	multifurcating	2700	2000	7	our model
	tree	UMI	tree	2600	2000	7	our model

labels from the GEO database (accession number: GSE95315). For each dataset, the Dyno platform also provides its reference trajectory backbone and cell positions, from which we calculate reference pseudotime by our definition. In addition, to evaluate TI methods on a dataset with disconnected states, we create the dataset *immune* by combining purified 10085 B cells, 8385 CD56 NK cells, and 2612 CD14 Monocytes cells from Zheng et al. (2017).

The drawback of using real datasets for benchmarking is that only discrete cell labels are available, though the cells are experiencing continuous transitions. In other words, the true cell positions and ordering along the trajectory are only known at a low resolution. To better evaluate VITAE's performance in estimating the cell positions and pseudotime, we also include synthetic datasets for evaluation. For synthetic datasets, we consider two different simulation approaches. One simulator we use is dyngen (Cannoodt et al., 2020), a multi-modal simulation engine for studying dynamic cellular processes at single-cell resolution. dyngen is also used in Saelens et al. (2019) and provides a delicate way to generate scRNA-seq data starting from gene regulation and transcriptional factors. However, it is limited to generate only a few hundreds of genes. Thus, we only generate 1000 genes for each dyngen dataset and treat the dyngen datasets as non-UMI data as the generated counts are large typically.

We generate four synthetic datasets from our own model framework. First, we train the model on a real data set to obtain a decoder and $\widehat{\boldsymbol{U}}$. Then, we generate each $\tilde{\boldsymbol{w}}_i$ and $\boldsymbol{Z}_i$ and the observed UMI counts $\boldsymbol{Y}_i$ following the hierarchical model (3) and (6). Specifically, we treat the estimated decoder and $\widehat{\boldsymbol{U}}$ as the true $f_g(\cdot)$ and $\boldsymbol{U}$ and design a trajectory backbone by connecting some edges between the vertices. In the four generated datasets, we do not include any confounding covariates $\boldsymbol{X}_i$. We use real dataset *dentate* to generate synthetic datasets *linear* and *bifurcation*, and the real dataset *fibroblast* to generate synthetic datasets *multifurcating* and *tree*.

5.2 Evaluation metrics

We use five different scores to measure TI methods' accuracy in recovering the true trajectory, cell positions, and pseudotime. All these scores range from 0 to 1, and a larger value indicates better performance.

First, to measure the difference between an estimated trajectory backbone $\widehat{\mathcal{B}}$ and the true trajectory backbone $\mathcal{B}$, we compute two scores: the GED score and the IM score, both of which evaluate the difference between two graphs and are invariant to the permutation of vertices. The graph edit distance (Abu-Aisheh et al., 2015) is defined as

$$\text{GED}(\widehat{\mathcal{B}}, \mathcal{B}) = \min_{e_1, \cdots, e_j \in \mathcal{P}(\widehat{\mathcal{B}}, \mathcal{B})} \sum_{i=1}^{j} c(e_i)$$

where $\mathcal{P}(\widehat{\mathcal{B}}, \mathcal{B})$, denotes the set of edit paths transforming graph $\widehat{\mathcal{B}}$ to the graph $\mathcal{B}$ and $c(e_i) \geq 0$ is the cost of an operation e_i. In other words, GED is a symmetric distance measure that quantifies the minimum number of graph edit operations needed to transform $\widehat{\mathcal{B}}$ into an isomorphic graph of $\mathcal{B}$. We then standardize the GED as

$$s_{\text{GED}} = 1 - \min\left\{\text{GED}(\widehat{\mathcal{B}}, \mathcal{B})/k, 1\right\}.$$

so that it ranges between 0 and 1. Besides the GED score, following Saelens et al. (2019), we also compute the IM score based on the Ipsen-Mikhailov distance (Jurman et al., 2015), $\text{IM}(\mathcal{B}, \widehat{\mathcal{B}})$, between the estimated and reference trajectory backbone. The IM distance is symmetric, measures the dissimilarity of adjacency matrices' spectra of the two graphs, and is bounded between 0 and 1. Our IM score is further defined as

$$s_{\text{IM}} = 1 - \text{IM}(\mathcal{B}, \widehat{\mathcal{B}}).$$

Next, we measure the error of the estimated cell position $\widehat{\tilde{\boldsymbol{w}}}_i$ for each cell i. We use two scores, the adjusted rand index (ARI) and the generalized rand index (GRI), both of which are invariant under the permutation of vertices and allow unequal number of vertices. The ARI (Hubert & Arabie, 1985) is a commonly used symmetric measure for the similarity between two clustering results. Following Saelens et al. (2019), the cells are assigned to its nearest discrete state based on their estimated $\widehat{\tilde{\boldsymbol{w}}}_i$ and true values of $\tilde{\boldsymbol{w}}_i$, and the two groupings are compared using ARI. Since such discretization only compares the accuracy of $\widehat{\tilde{\boldsymbol{w}}}_i$ at a low resolution, we also define a GRI score that directly compares $\widehat{\tilde{\boldsymbol{w}}}_i$ with $\tilde{\boldsymbol{w}}_i$.

Our GRI metric is an extension of the rand index (RI). For any two pair of cells i_1 and i_2, we define its "true" similarity as $\rho_{i_1 i_2} = \langle \sqrt{\tilde{\boldsymbol{w}}_{i_1}}, \sqrt{\tilde{\boldsymbol{w}}_{i_2}} \rangle$ where $\sqrt{\boldsymbol{v}} = (\sqrt{v_1}, \cdots, \sqrt{v_k})$ for a vector $\boldsymbol{v}$ of length k. The square-root operation is to guarantee that $\rho_{i_1 i_2} = 1$ if and only if $\tilde{\boldsymbol{w}}_{i_1} = \tilde{\boldsymbol{w}}_{i_2}$. Similarly, we define the estimated similarity between the two cells as $\widehat{\rho}_{i_1 i_2} = \langle \sqrt{\widehat{\tilde{\boldsymbol{w}}}_{i_1}}, \sqrt{\widehat{\tilde{\boldsymbol{w}}}_{i_2}} \rangle$. Then to compare the similar between the estimated and reference cell positions on the trajectory backbone, the GRI is defined as

$$\text{GRI} = \sum_{i_1=1}^{n-1} \sum_{i_2=i_1+1}^{n} \left(1 - |\rho_{i_1 i_2} - \widehat{\rho}_{i_1 i_2}|\right) \Big/ \binom{2}{n}.$$

The rand index is a special case of GRI when all cells are positioned exactly on vertices.

Finally, we also measure the similarity between the reference and estimated pseudotime of cells along the trajectory, which is simply the Pearson correlation between the estimated pseudotime $\widehat{T}_i$ and the reference pseudotime T_i across cells. Specifically, we define the PDT score as

$$s_{\text{PDT}} = \left(\text{Cor}(\boldsymbol{T}, \widehat{\boldsymbol{T}}) + 1\right)/2.$$

which has a range between 0 and 1.

5.3 Evaluation results

We evaluate the performance of two versions of VITAE. VITAE_NB is VITAE with the likelihood-based loss (NB for scRNA-seq with UMI and ZINB for scRNA-seq with non-UMI). The other is VITAE_Gauss, the accelerated version of VITAE introduced in Section 3.3, where the inputs are the PC scores and the Normal density replaces the likelihood in the loss function. For benchmarking, we compare with three state-of-art TI methods: PAGA (Wolf et al., 2019), Monocle 3 (Cao et al., 2019), and Slingshot (Street et al., 2018).

To make the other TI methods have a comparable form of outputs, we run PAGA, Monocle 3, and Slingshot via the Dyno platform, which converts these TI methods' outputs into an estimated trajectory backbone $\widehat{\mathcal{B}}$, estimated cell positions $\widehat{\boldsymbol{w}}_i$'s, and estimated pseudotime $\widehat{T}_i$'s. For a fair comparison, the true number of states k, clustering labels, and root are provided as prior information to all TI methods. For Slingshot, which requires an extra two-dimensional projection of cells as input, the UMAP (McInnes et al., 2018) coordinates of cells computed following Seurat's (Stuart et al., 2019) default steps are used.

The evaluation results are summarized in Figure 3 across all 25 datasets on 5 evaluation scores. For the two versions of VITAE, these scores are averages over 100 different random seeds. As computing the exact GED is NP-hard and costly when k is large, the GED score is not calculated for the *planaria_full* data with $k = 33$. In addition, the PDT score is not meaningful for the *cycle* and *disconnected* structure, so it is also not computed for these trajectory types.

As shown in Figure 3, both versions of VITAE generally have consistently larger scores than the other three TI methods on all five metrics. Specifically, VITAE provides a more accurate recovery of the trajectory backbones, and thus an improved estimation of the pseudotime. In terms of the ARI and GRI, we see consistent improvement of VITAE in all synthetic data, though our improvements are subtle in some real datasets. A possible explanation is that the "true" cell positions and pseudotime in real datasets are discretized, assuming that cells are only placed on vertices, so they might not be sensitive enough to evaluate the accuracy when estimating the real cell positions. In addition, we do see comparable performance of the accelerated version of VITAE compared to our original scheme, showing that VITAE_Gauss can well approximate the likelihood-based loss and makes VITAE scalable to handle large datasets.

6 An Application to the Developing Mouse Neocortex

Finally, we apply VITAE to analyze the developing mouse neocortex. The six-layered neocortex is evolutionarily unique in mammals and forms the physical center for the highest cognitive functions (Geschwind & Rakic, 2013). It has been shown that Neuroepithelial cells (NECs) transition into the radial glial cells (RGCs), while cortical projection neuron types are generated sequentially by RGCs and intermediate progenitor cells (IPCs) (Lui et al., 2011; Götz & Huttner, 2005; Greig et al., 2013). A few scRNA-seq datasets have been generated (Pollen et al., 2014; Yuzwa et al., 2017; Loo et al., 2019; Ruan et al., 2020+) to reveal and understand the developmental process and the dynamic gene expressions in the mouse neocortex. Here, we aim to perform a joint trajectory analysis using both cells from Yuzwa et al. (2017) and Ruan et al. (2020+).

As shown in Table 2, Dataset A (Ruan et al., 2020+) contains 10261 cortical cells collected from E10.5, E12.5, E14.5, E15.5, E16.5, and E18.5 mouse embryos and Dataset B (Yuzwa et al., 2017) includes 6390 cortically derived cells from E11.5, E13.5, E15.5, and E17.5 mouse embryos. Both datasets sample cells from the same region in the mouse cortex and a joint analysis of them will enable analyzing the dynamic process from a full spectrum of mice from E10.5 to E18.5. We keep the genes that are measured in both datasets (14707 genes), merge all 16651 cells, and preprocess these cells following our default procedure described in Appendix A. As expected, the raw observed gene

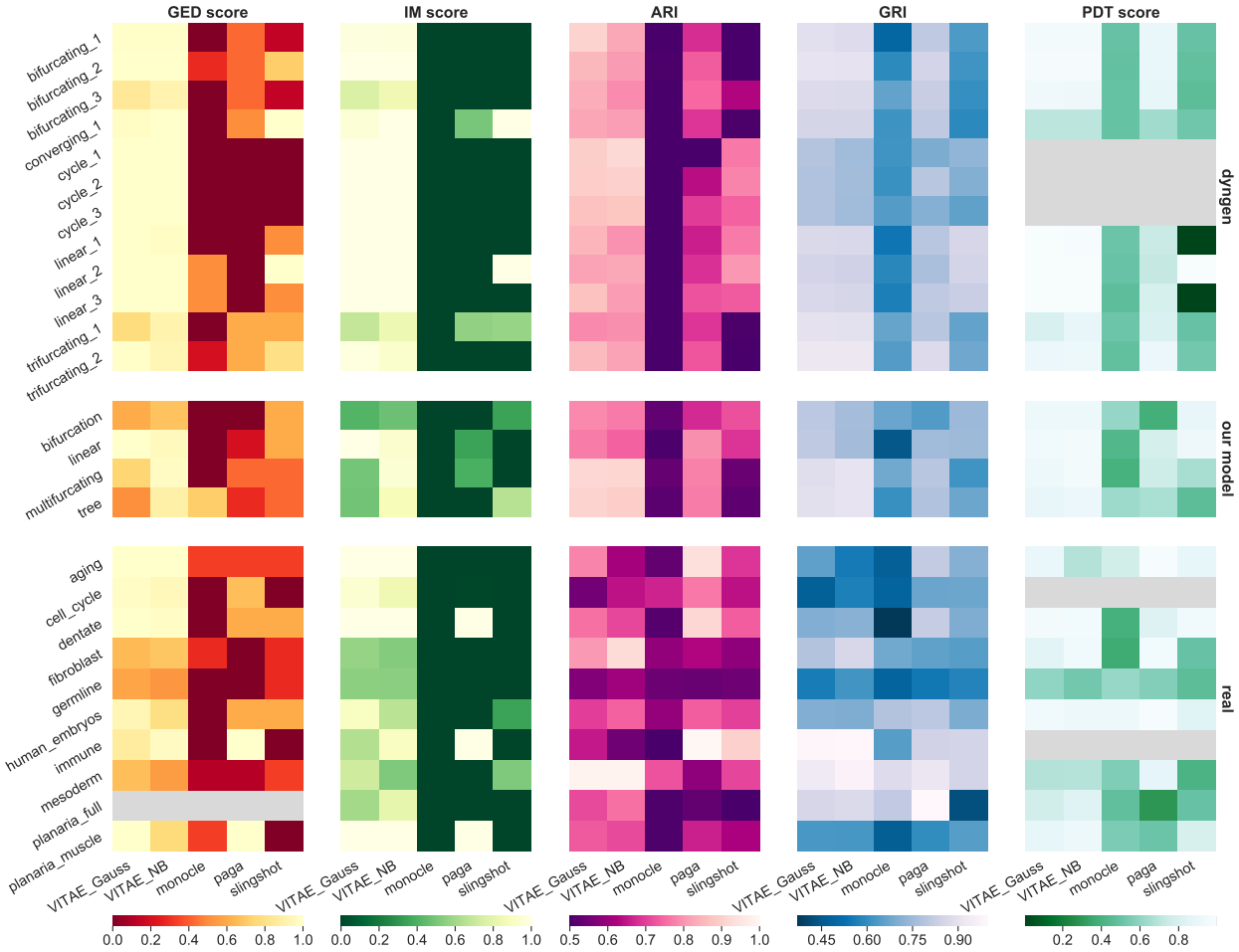

Figure 3: Evaluation results of VITAE_Gauss, VITAE_NB compared with three other TI methods: Monocle 3, PAGA and Slingshot, on 25 datasets with 5 evaluation scores. A larger score (lighter color) indicates better performance, and the datasets are grouped by their source. The PDT scores for datasets with the *cycle* and *disconnected* trajectory topology are not computed as the pseudotime is ill-defined. The GED scores for the *planaria_full* dataset are also not computed due to computational infeasibility.

counts from the two datasets do not merge together due to lab-specific batch effects and the non-overlapping of days in sample collection (Figure 4a).

In order to adjust for confounding batch effect, we include the dataset ID of each cell (whether it is from Dataset A or B) in $\boldsymbol{X}_i$. In addition, as the NECs and RGCs are experiencing cell division, we also adjust for the cell-cycle effects by adding the cell-cycle scores, computed using Seurat, into $\boldsymbol{X}_i$. To perform the joint trajectory analysis, we apply the accelerated VITAE (the VITAE_Gauss) with the adjustment of $\boldsymbol{X}_i$. To determine k, clustering is performed after the pretraining step. We also compare with an alternative approach combining Seurat data integration (Stuart et al., 2019) with Slingshot. Specifically, Seurat is first used to integrate both datasets, regress out cell-cycle scores, and cluster the cells with integrated data. Then Slingshot is applied to the Seurat results to infer the trajectories and pseudotime.

As shown in Figure 4bc, both the Seurat integration and VITAE can successfully remove lab-specific batch effects. To understand whether they retain meaningful biological signals, we then color the cells by their reference major cell types given in the original papers (Figure 5ab). As Yuzwa et al. (2017) did not provide cell type labels, the reference labels of cells in Dataset B are obtained by a separate clustering of Dataset B using Seurat, and the clusters are annotated using the gene markers provided by Ruan et al. (2020+). Note that these reference labels are only used for evaluation

Table 2: Number of cells on different embryonic days.

Embryonic Day	E10.5	E11.5	E12.5	E13.5	E14.5	E15.5	E16.5	E17.5	E18.5
Dataset A	1172	0	2668	0	3742	1014	387	0	1278
Dataset B	0	1418	0	1137	0	2955	0	880	0

and are not used during the trajectory inference of both approaches. Though both approaches can retain meaningful biological signals and merge only the cells in the same major cell type, VITAE can provide a cleaner integration, leading to the recovery of a more biologically meaningful trajectory shared in both datasets (Figure 5b). VITAE successfully identifies the *NEC-RGC-IPC-Neuron* major cell lineage and disconnects the microglia cells, pericytes, Pia, and interneurons in Dataset A that are not derived from the major trajectory. In contrast, the trajectory given by Slingshot after Seurat integration is messy and hard to interpret (Figure 5a).

In terms of uncertainty quantification, VITAE returns the edge score of each edge in the inferred trajectory (Figure 5b, shown as the width of each edge), indicating our confidence in whether an edge exists or not. In addition, we also quantify the uncertainty of each cell's position along the trajectory by the projection MSE defined in (13) (Figure 5c). As discussed in more detail in Appendix F, the projection MSE tends to be smaller for cells that are near the vertices. Comparing the projection MSEs of the cells on different edges, we have larger uncertainty in the cell positions for the IPCs and immature neurons, most of which are actively involved in dynamic transitions.

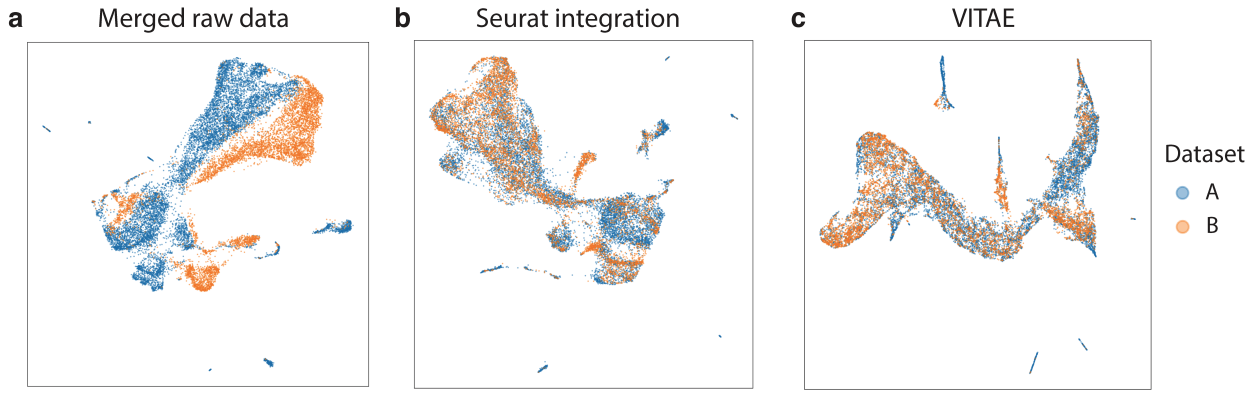

Figure 4: UMAP visualizations of the mouse neocortex datasets A (blue) and B (orange). We compare the UMAP plots of the cells from **(a)** the merged raw data, **(b)** Seurat integration and cell-cycle adjustment with **(c)** the latent space estimated by VITAE.

Now, we further examine the change in gene expression along sub-trajectories. Figure 6 shows the top marker genes for two sub-trajectories. Using a p-value threshold of 0.05, we can identify 790 differential expressed genes along the sub-trajectory *RGC→IPC→Immature Neuron* and 813 differential expressed genes along the sub-trajectory *NEC→RGC→OPC*. The differentially expressed genes are further ranked by their estimated covariates-adjusted correlation $\widehat{\rho}_g = \widehat{\beta}_{1g}\text{sd}[\text{rank}(\widehat{T}_\cdot)]/\text{sd}(Y_{\cdot g})$ defined in (15). For each sub-trajectory, we visualize the expressions of the top 2 genes positively correlated with the pseudotime and the top 2 genes that are negatively correlated. We fit a LOESS curve of each gene for each dataset separately, and find that the gene expression patterns are consistent in both datasets, indicating that our pseudotime ordering is biologically meaningful. All these marker genes have been shown to potentially play essential roles in the development of mouse neocortical neurons (Ruan et al., 2020+). Notice that the sub-trajectory *NEC→RGC→OPC* is positively correlated with the cell collection time, while there is no clear correlation between the pseudotime and cell collection time in the other sub-trajectory *RGC→IPC→Immature Neuron* (Figure 5d, Figure 6). In both two cases, the gene expression patterns across the two datasets are consistent, despite the

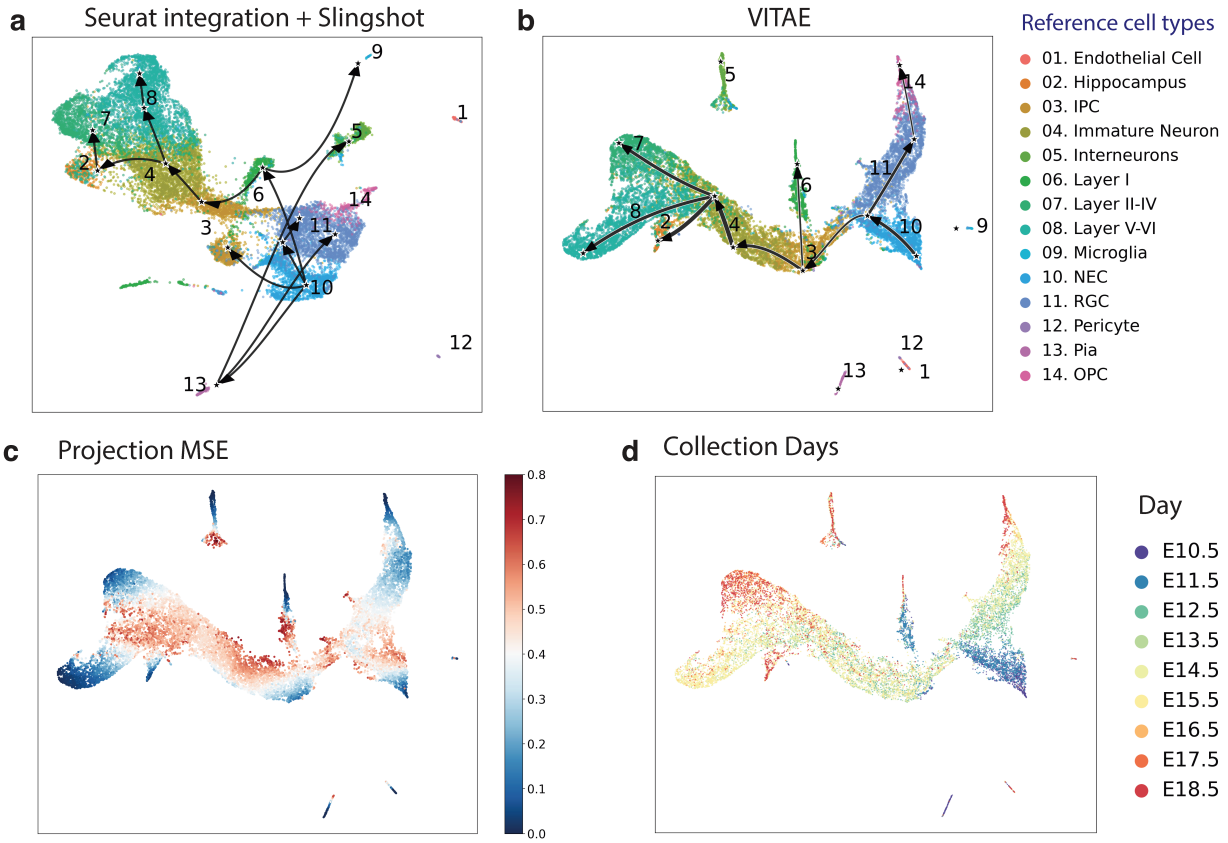

Figure 5: UMAP visualizations of estimated trajectories on the merged dataset of developing mouse neocortex with cells colored by reference cell types. **(a)** The trajectory is given by Slingshot after Seurat data integration, cell cycle adjustment and clustering. **(b)** The trajectory is given by the accelerated VITAE. The width of each edge is proportional to its edge score. **(c)** Cells cells are recolored by their projection MSE defined in (13). **(d)** Cells are recolored by their collection days.

lab-specific batch effects and the non-overlapping of cell collection days between these two datasets.

Finally, in terms of computational cost, our approach takes less than 3 minutes with 1 GPU and less than 10 minutes with 8 CPU cores to infer the underlying trajectory. As a comparison, the data integration and cell-cycle adjustment in Seurat take 11 minutes, and Slingshot takes a further 1 minute using their accelerated version. We have comparable computational costs as these commonly used computational tools for scRNA-seq, while VITAE performs better in recovering the underlying trajectories.

7 Discussion

In this paper, we propose VITAE, a hierarchical mixture model combined with variational autoencoders, for trajectory inference. Compared with existing TI methods, VITAE provides a coherent probabilistic framework for TI in scRNA-seq, to explicitly define the trajectories and pseudotime that are estimable from scRNA-seq, to handle various trajectory topologies, and to simultaneously adjust for confounding covariates. VITAE leverages variational autoencoders to approximate posterior quantities, and infers the trajectory backbone and cell positions from posterior approximations. VITAE is scalable to handle large datasets by the automatic parallelization of deep learning models and the reduction of input dimension with PC scores.

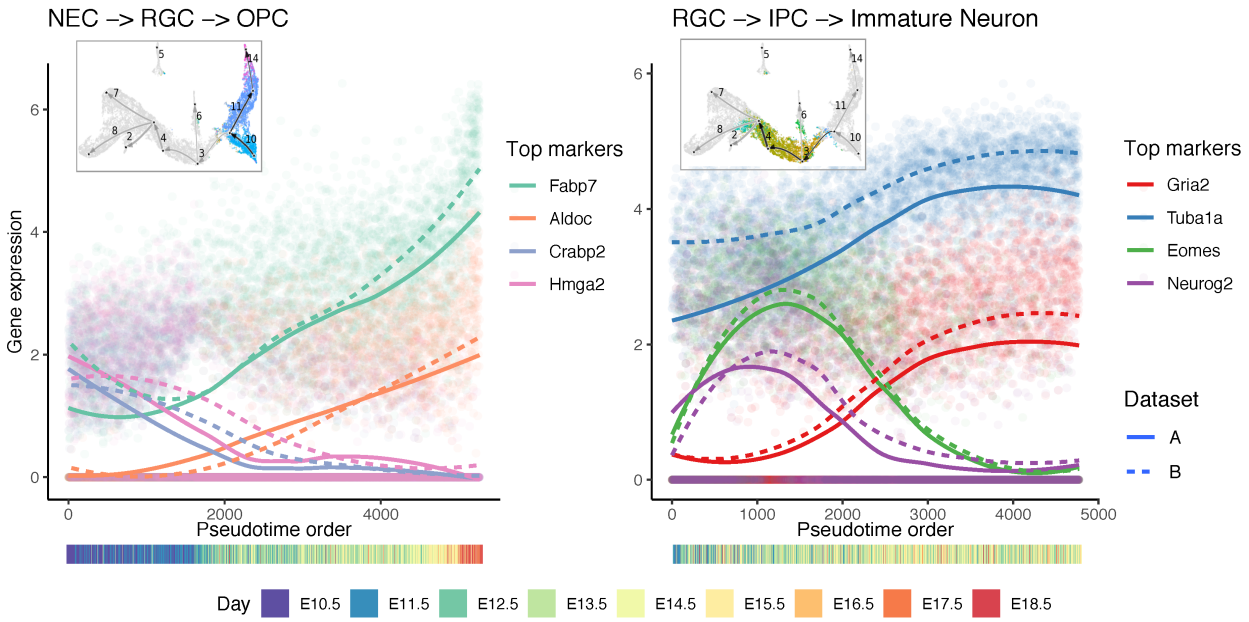

Figure 6: The expression of top marker genes for two sub-trajectories along the inferred pseudotime ordering of the cells. The figures inserted on the top-left corners show the selected cells for each sub-trajectory. For each main figure, a colored point is the log-transformed normalized count of a gene in a cell. For each gene, a separate Loess curve is fitted on the observed expressions in each dataset. The bottom color bars show the collection days of the cells ordered by their inferred pseudotime.

Though there are many available tools for integrating multiple scRNA-seq datasets, it is typically challenging to integrate datasets when the cells have a continuous trajectory structure and do not fall into distinct cell types. As shown in Section 6, our approach, which simultaneously performs trajectory analysis with data integration, can better regularize the latent space to infer a cleaner shared trajectory while retaining biologically meaningful differences in each dataset.

Though our approach is powerful for TI, there are still some limitations. First, the approximate variational inference we use to infer the trajectory and quantify cell position uncertainties ignores the estimation uncertainties of our encoder and decoder. As a consequence, though our uncertainty quantification is informative, there is no guarantee that it is close to the true posterior uncertainties. In addition, we still need a more principled statistical framework to find differentially expressed genes along the trajectory, to account for the post-estimation bias and adjust for the confounding effects of known covariates. Since using scRNA-seq counts alone is often not enough to identify the true temporal lineage of cells, we also believe that extending our framework to model both the spliced and unspliced RNA counts with RNA velocities can provide more information in understanding the true dynamic process of cells.

Acknowledgement

This work was completed in part with resources provided by the University of Chicago Research Computing Center. We thank Xiaochang Zhang for sharing the mouse neocortex dataset A and helpful suggestions on our analysis of the mouse neocortex.

References

Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., & Martineau, P. (2015). An exact graph edit distance algorithm for

solving pattern recognition problems. In *4th International Conference on Pattern Recognition Applications and Methods.*

Agarwal, D., Wang, J., & Zhang, N. R. (2020). Data denoising and post-denoising corrections in single cell RNA sequencing. *Statistical Science*, *35*(1), 112–128.

Ahmed, S., Rattray, M., & Boukouvalas, A. (2019). GrandPrix: scaling up the Bayesian GPLVM for single-cell data. *Bioinformatics*, *35*(1), 47–54.

Bergen, V., Lange, M., Peidli, S., Wolf, F. A., & Theis, F. J. (2020). Generalizing rna velocity to transient cell states through dynamical modeling. *Nature biotechnology*, 1–7.

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

Campbell, K. R., & Yau, C. (2016). Order under uncertainty: robust differential expression analysis using probabilistic models for pseudotime inference. *PLoS computational biology*, *12*(11), e1005212.

Cannoodt, R., Saelens, W., Deconinck, L., & Saeys, Y. (2020). dyngen: a multi-modal simulator for spearheading new single-cell omics analyses. *BioRxiv*.

Cao, J., Spielmann, M., Qiu, X., Huang, X., Ibrahim, D. M., Hill, A. J., . . . others (2019). The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, *566*(7745), 496–502.

Cembrowski, M. S. (2019). Single-cell transcriptomics as a framework and roadmap for understanding the brain. *Journal of neuroscience methods*, *326*, 108353.

Chen, H., Ye, F., & Guo, G. (2019). Revolutionizing immunology with single-cell RNA sequencing. *Cellular & molecular immunology*, *16*(3), 242–249.

Eraslan, G., Simon, L. M., Mircea, M., Mueller, N. S., & Theis, F. J. (2019). Single-cell RNA-seq denoising using a deep count autoencoder. *Nature communications*, *10*(1), 1–14.

Fan, J., Slowikowski, K., & Zhang, F. (2020). Single-cell transcriptomics in cancer: Computational challenges and opportunities. *Experimental & Molecular Medicine*, *52*(9), 1452–1465.

Frühwirth-Schnatter, S., & Malsiner-Walli, G. (2019). From here to infinity: sparse finite versus dirichlet process mixtures in model-based clustering. *Advances in data analysis and classification*, *13*(1), 33–64.

Geschwind, D. H., & Rakic, P. (2013). Cortical evolution: judge the brain by its cover. *Neuron*, *80*(3), 633–647.

Götz, M., & Huttner, W. B. (2005). The cell biology of neurogenesis. *Nature reviews Molecular cell biology*, *6*(10), 777–788.

Greig, L. C., Woodworth, M. B., Galazo, M. J., Padmanabhan, H., & Macklis, J. D. (2013). Molecular logic of neocortical projection neuron specification, development and diversity. *Nature Reviews Neuroscience*, *14*(11), 755–769.

Griffiths, J. A., Scialdone, A., & Marioni, J. C. (2018). Using single-cell genomics to understand developmental processes and cell fate decisions. *Molecular systems biology*, *14*(4), e8046.

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

Guo, F., Yan, L., Guo, H., Li, L., Hu, B., Zhao, Y., . . . Qiao, J. (2015). The transcriptome and DNA methylome landscapes of human primordial germ cells. *Cell*, *161*(6), 1437–1452.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., . . . Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations.*

Hochgerner, H., Zeisel, A., Lönnerberg, P., & Linnarsson, S. (2018). Conserved properties of dentate gyrus neurogenesis across postnatal development revealed by single-cell rna sequencing. *Nature neuroscience*, *21*(2), 290–299.

Hovestadt, V., Smith, K. S., Bihannic, L., Filbin, M. G., Shaw, M. L., Baumgartner, A., . . . Northcott, P. A. (2019). Resolving medulloblastoma cellular architecture by single-cell genomics. *Nature*, *572*(7767), 74–79.

Huang, M., Zhang, Z., & Zhang, N. R. (2020). Dimension reduction and denoising of single-cell RNA sequencing data in the presence of observed confounding variables. *bioRxiv*.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of classification*, *2*(1), 193–218.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., & Zhou, H. (2017). Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th international joint conference on artificial intelligence* (pp. 1965–1972).

Jurman, G., Visintainer, R., Filosi, M., Riccadonna, S., & Furlanello, C. (2015). The HIM glocal metric and kernel for network comparison and classification. In *2015 IEEE International Conference on Data Science and Advanced Analytics* (pp. 1–10).

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In Y. Bengio & Y. LeCun (Eds.), *2nd International Conference on Learning Representations.*

Kowalczyk, M. S., Tirosh, I., Heckl, D., Rao, T. N., Dixit, A., Haas, B. J., . . . Regev, A. (2015). Single-cell RNA-seq reveals changes in cell cycle and differentiation programs upon aging of hematopoietic stem cells. *Genome research*, *25*(12), 1860–1872.

Kumar, P., Tan, Y., & Cahan, P. (2017). Understanding development and stem cells using single cell-based analyses of gene expression. *Development*, *144*(1), 17–32.

Lähnemann, D., Köster, J., Szczurek, E., McCarthy, D. J., Hicks, S. C., Robinson, M. D., . . . Schönhuth, A. (2020). Eleven grand challenges in single-cell data science. *Genome biology*, *21*(1), 1–35.

La Manno, G., Soldatov, R., Zeisel, A., Braun, E., Hochgerner, H., Petukhov, V., . . . others (2018). RNA velocity of single cells. *Nature*, *560*(7719), 494–498.

Li, X., Wang, K., Lyu, Y., Pan, H., Zhang, J., Stambolian, D., . . . Li, M. (2020). Deep learning enables accurate clustering with batch effect removal in single-cell RNA-seq analysis. *Nature communications*, *11*(1), 1–14.

Lin, C., & Bar-Joseph, Z. (2019). Continuous-state hmms for modeling time-series single-cell rna-seq data. *Bioinformatics*, *35*(22), 4707–4715.

Loh, K. M., Chen, A., Koh, P. W., Deng, T. Z., Sinha, R., Tsai, J. M., . . . Weissman, I. L. (2016). Mapping the pairwise choices leading from pluripotency to human bone, heart, and other mesoderm cell types. *Cell*, *166*(2), 451–467.

Loo, L., Simon, J. M., Xing, L., McCoy, E. S., Niehaus, J. K., Guo, J., . . . Zylka, M. J. (2019). Single-cell transcriptomic analysis of mouse neocortical development. *Nature communications*, *10*(1), 1–11.

Lui, J. H., Hansen, D. V., & Kriegstein, A. R. (2011). Development and evolution of the human neocortex. *Cell*, *146*(1), 18–36.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (Vol. 30, p. 3).

Marco, E., Karp, R. L., Guo, G., Robson, P., Hart, A. H., Trippa, L., & Yuan, G.-C. (2014). Bifurcation analysis of single-cell gene expression data reveals epigenetic landscape. *Proceedings of the National Academy of Sciences*, *111*(52), E5643–E5650.

McInnes, L., Healy, J., Saul, N., & Großberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, *3*(29), 861.

Petropoulos, S., Edsgärd, D., Reinius, B., Deng, Q., Panula, S. P., Codeluppi, S., . . . Lanner, F. (2016). Single-cell RNA-seq reveals lineage and X chromosome dynamics in human preimplantation embryos. *Cell*, *165*(4), 1012–1026.

Pollen, A. A., Nowakowski, T. J., Shuga, J., Wang, X., Leyrat, A. A., Lui, J. H., ... others (2014). Low-coverage single-cell mrna sequencing reveals cellular heterogeneity and activated signaling pathways in developing cerebral cortex. *Nature biotechnology*, *32*(10), 1053.

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

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning* (pp. 1278–1286).

Risso, D., Perraudeau, F., Gribkova, S., Dudoit, S., & Vert, J.-P. (2018). A general and flexible method for signal extraction from single-cell RNA-seq data. *Nature communications*, *9*(1), 1–17.

Ruan, X., Kang, B., Qi, C., Wang, J., & Zhang, X. (2020+). Progenitor cell diversity in the developing mouse neocortex. *Under revision*.

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

Schiebinger, G., Shu, J., Tabaka, M., Cleary, B., Subramanian, V., Solomon, A., ... others (2019). Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, *176*(4), 928–943.

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems* (pp. 3483–3491).

Street, K., Risso, D., Fletcher, R. B., Das, D., Ngai, J., Yosef, N., ... Dudoit, S. (2018). Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics*, *19*(1), 477.

Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck III, W. M., ... Satija, R. (2019). Comprehensive integration of single-cell data. *Cell*, *177*(7), 1888–1902.

Stuhlmüller, A., Taylor, J., & Goodman, N. (2013). Learning stochastic inverses. *Advances in neural information processing systems*, *26*, 3048–3056.

Svensson, V., da Veiga Beltrame, E., & Pachter, L. (2020). A curated database reveals trends in single-cell transcriptomics. *Database*, *2020*.

Tanay, A., & Regev, A. (2017). Scaling single-cell genomics from phenomenology to mechanism. *Nature*, *541*(7637), 331–338.

Tang, F., Lao, K., & Surani, M. A. (2011). Development and applications of single-cell transcriptome analysis. *Nature methods*, *8*(4), S6–S11.

Tran, T. N., & Bader, G. D. (2020). Tempora: Cell trajectory inference using time-series single-cell RNA sequencing data. *PLoS computational biology*, *16*(9), e1008205.

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

Treutlein, B., Lee, Q. Y., Camp, J. G., Mall, M., Koh, W., Shariati, S. A. M., . . . Quake, S. R. (2016). Dissecting direct reprogramming from fibroblast to neuron using single-cell RNA-seq. *Nature*, *534*(7607), 391–395.

Tritschler, S., Büttner, M., Fischer, D. S., Lange, M., Bergen, V., Lickert, H., & Theis, F. J. (2019). Concepts and limitations for learning developmental trajectories from single cell genomics. *Development*, *146*(12), dev170506.

Van Buren, S., Sarkar, H., Srivastava, A., Rashid, N. U., Patro, R., & Love, M. I. (2020). Compression of quantification uncertainty for scRNA-seq counts. *BioRxiv*.

Wang, J., Agarwal, D., Huang, M., Hu, G., Zhou, Z., Ye, C., & Zhang, N. R. (2019). Data denoising with transfer learning in single-cell transcriptomics. *Nature methods*, *16*(9), 875–878.

Wang, J., Zhao, Q., Hastie, T., & Owen, A. B. (2017). Confounder adjustment in multiple hypothesis testing. *Annals of statistics*, *45*(5), 1863.

Weinreb, C., Wolock, S., Tusi, B. K., Socolovsky, M., & Klein, A. M. (2018). Fundamental limits on dynamic inference from single-cell snapshots. *Proceedings of the National Academy of Sciences*, *115*(10), E2467–E2476.

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

Yuzwa, S. A., Borrett, M. J., Innes, B. T., Voronova, A., Ketela, T., Kaplan, D. R., . . . Miller, F. D. (2017). Developmental emergence of adult neural stem cells as revealed by single-cell transcriptional profiling. *Cell reports*, *21*(13), 3970–3986.

Zheng, G. X., Terry, J. M., Belgrader, P., Ryvkin, P., Bent, Z. W., Wilson, R., . . . Bielas, J. H. (2017). Massively parallel digital transcriptional profiling of single cells. *Nature communications*, *8*(1), 1–12.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations.*

A Data preprocessing

We follow the procedure in Seurat (Stuart et al., 2019) to preprocess the raw scRNA-seq count matrix. First, we remove cells and genes whose expressions are all zero and obtain the raw count data Y_{ig}^{raw}, for $i = 1, \ldots, N$ and $g = 1, \ldots, G^{\text{raw}}$. Then, we normalize each cell by its library size and log-transform the normalized counts. Specifically, we compute the library size for each cell as

$$\text{SF}_i = \frac{1}{N_C} \sum_g Y_{ig}^{\text{raw}},$$

for a fix number N_C (with the default value 10^4 as in Seurat), and transform the raw counts to obtain

$$Y_{ig}^{\text{norm}} = \log\left(\frac{Y_{ig}^{\text{raw}}}{\text{SF}_i} + 1\right).$$

Next, we find highly variable genes following the procedure describe in Stuart et al. (2019). Specifically, we calculate the mean and variance of each gene on the raw count data

$$\mu_g = \frac{1}{N} \sum_i Y_{ig}^{\text{raw}}, \qquad \sigma_g^2 = \frac{1}{N} \sum_i \left(Y_{ig}^{\text{raw}} - \mu_g\right)^2,$$

and then do a local polynomial regression with degree 2 and span 0.3 to regress $\log_{10} \sigma_g^2$ on $\log_{10} \mu_g$. The fitted variance is then used to compute the standardized gene expressions

$$Y_{ig}' = \min\left\{\frac{Y_{ig}^{\text{raw}} - \mu_g}{\widehat{\sigma}_g}, \sqrt{N}\right\},$$

For each gene, we calculate a feature importance score as

$$\text{FI}_g = \frac{1}{N-1} \sum_i Y_{ig}'^2.$$

A larger value of FI_g indicates a more highly variable gene. By default, we select the first $G = \min\{2000, G^{\text{raw}}\}$ genes for downstream analyses. The input of VITAE is obtained by a further scaling of the expression of each selected genes across cells so that each gene has mean 0 and variance 1.

B Loss Function

Note that the loss for each cell in (10) takes the form

$$\begin{aligned}
\mathcal{L}(\boldsymbol{Y}_i; \boldsymbol{X}_i, \Theta) = &-(1-\alpha)\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i) \\
&- \alpha \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i = \boldsymbol{0}_d, \boldsymbol{X}_i) + \beta D_{\text{KL}}(q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)\|p(\boldsymbol{Z}_i)) \\
= &-(1-\alpha)\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i) - \alpha \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i = \boldsymbol{0}_d, \boldsymbol{X}_i) \\
&+ \beta\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i) - \beta\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log p(\boldsymbol{Z}_i).
\end{aligned}$$

Next we show how to compute the above terms efficiently.

1. $\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i)$

$$\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i) \approx \frac{1}{L}\sum_{l=1}^{L}\sum_{g=1}^{G} \log p(Y_{ig}|\boldsymbol{Z}_i = \boldsymbol{z}_i^{(l)}, \boldsymbol{X}_i)$$

where $\boldsymbol{z}_i^{(1)}, \ldots, \boldsymbol{z}_i^{(L)}$ are Monte Carlo samples from the approximate posterior distribution $q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$. Here $p(Y_{ig}|\boldsymbol{Z}_i, \boldsymbol{X}_i)$ depends on the distribution assumptions of Y_{ig}.

(1) For scRNA-seq with UMI, $Y_{ig}|\boldsymbol{Z}_i, \boldsymbol{X}_i$ follows a Negative Binomial distribution $\mathcal{NB}(\lambda_{ig}, \theta_g)$ with a probability mass function,

$$p(y|\boldsymbol{Z}_i, \boldsymbol{X}_i) = \binom{y}{\theta_g + y - 1}\left(\frac{\theta_g}{\lambda_{ig} + \theta_g}\right)^{\theta_g}\left(\frac{\lambda_{ig}}{\lambda_{ig} + \theta_g}\right)^{y},$$

for $i = 1, \ldots, N$. It is noted that only λ_{ig} dependent on $\boldsymbol{Z}_i$ and $\boldsymbol{X}_i$.

(2) For non-UMI data, $Y_{ig}|\boldsymbol{Z}_i, \boldsymbol{X}_i$ follows a zero-inflated Negative Binomial distribution $\mathcal{ZINB}(\lambda_{ig}, \theta_g, \phi_{ig})$ with the probability mass function,

$$p(y|\boldsymbol{Z}_i, \boldsymbol{X}_i) = \begin{cases} \phi_{ig} + (1 - \phi_{ig})\, p_{NB}(0; \lambda_{ig}, \theta_g) & \text{if } y = 0 \\ (1 - \phi_{ig})\, p_{NB}(y; \lambda_{ig}, \theta_g) & \text{if } y > 0, \end{cases}$$

where $p_{NB}(y; \lambda_{ig}, \theta_g)$ is the probability mass function of $\mathcal{NB}(\lambda_{ig}, \theta_g)$.

(3) For the accelerated VITAE with a Gaussian model, $\boldsymbol{Y}_i$ is replaced by the PC scores $\boldsymbol{F}_i$. We assume that each $F_{ig}|\boldsymbol{Z}_i, \boldsymbol{X}_i$ follows a Gaussian distribution $\mathcal{N}(\nu_{ig}, \tau_i^2)$ with probability density function,

$$p(y|\boldsymbol{Z}_i, \boldsymbol{X}_i) = \frac{1}{\sqrt{2\pi}\tau_i}\exp\left(-\frac{(y - \nu_{ig})^2}{2\tau_i^2}\right).$$

2. $\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log p(\boldsymbol{Z}_i)$

$$\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log p(\boldsymbol{Z}_i) = \mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\left[\sum_{c=1}^{K} p(\boldsymbol{z}|c_i = c)p(c)\right] \approx \frac{1}{L}\sum_{l=1}^{L}\sum_{c=1}^{K} p(\boldsymbol{z}^{(l)}|c_i = c)p(c).$$

When $c \in \mathcal{N}(\mathcal{G})$, the conditional density of $\boldsymbol{Z}_i|c_i = c$ is given by

$$p(\boldsymbol{z}|c_i = c) = \int_0^1 p(\boldsymbol{z}|c_i = c, w)\mathrm{d}w = \frac{1}{(2\pi)^{\frac{d}{2}}}\int_0^1 e^{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{U}\boldsymbol{b}_c)^\top(\boldsymbol{z}-\boldsymbol{U}\boldsymbol{b}_c)}\mathrm{d}w = \varphi_d(\boldsymbol{U}\boldsymbol{b}_c).$$

where $\varphi_d(\cdot)$ is the probability density function of the standard d-dimensional multivariate Gaussian distribution. When $c \in \mathcal{E}(\mathcal{G})$, the conditional density of $\boldsymbol{Z}_i|c_i = c$ is given by

$$\begin{aligned} p(\boldsymbol{z}|c_i = c) &= \int_0^1 p(\boldsymbol{z}|c_i = c, w)\mathrm{d}w \\ &= \frac{1}{(2\pi)^{\frac{d}{2}}}\int_0^1 e^{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{U}\tilde{w})^\top(\boldsymbol{z}-\boldsymbol{U}\tilde{w})}\mathrm{d}w \\ &\overset{\substack{\boldsymbol{\alpha}_{\boldsymbol{z}c}=\boldsymbol{U}(\boldsymbol{b}_c-\boldsymbol{a}_c) \\ \boldsymbol{\beta}_{\boldsymbol{z}c}=\boldsymbol{z}-\boldsymbol{U}\boldsymbol{b}_c}}{=\!=\!=\!

$$
\begin{aligned}
&= \frac{1}{(2\pi)^{\frac{d}{2}}} \int_0^1 e^{-\frac{1}{2}(\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top \boldsymbol{\alpha}_{\boldsymbol{z}c} w^2 + 2\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top \boldsymbol{\beta}_{\boldsymbol{z}c} w + \boldsymbol{\beta}_{\boldsymbol{z}c}^\top \boldsymbol{\beta}_{\boldsymbol{z}c})} \mathrm{d}w \\
&= \frac{\sigma_c e^{\frac{t_c}{2}}}{(2\pi)^{\frac{d}{2}-1}} \left[\Phi\left(\frac{1-\nu_{\boldsymbol{z}c}}{\sigma_c}\right) - \Phi\left(-\frac{\nu_{\boldsymbol{z}c}}{\sigma_c}\right) \right],
\end{aligned}
$$

with

$$
\nu_{\boldsymbol{z}c} = -\frac{\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top \boldsymbol{\beta}_{\boldsymbol{z}c}}{\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top \boldsymbol{\alpha}_{\boldsymbol{z}c}}, \quad \sigma_{\boldsymbol{z}c}^2 = \frac{1}{\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top \boldsymbol{\alpha}_{\boldsymbol{z}c}}, \quad t_{\boldsymbol{z}c} = -\boldsymbol{\beta}_{\boldsymbol{z}c}^\top \boldsymbol{\beta}_{\boldsymbol{z}c} + \frac{(\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top \boldsymbol{\beta}_{\boldsymbol{z}c})^2}{\boldsymbol{\alpha}_{\boldsymbol{z}c}^\top \boldsymbol{\alpha}_{\boldsymbol{z}c}}
$$

where $\Phi(\cdot)$ is the cumulative density function of the standard Gaussian distribution.

Therefore, the marginal density of $\boldsymbol{Z}_i$ is given by

$$
p(\boldsymbol{z}) = \sum_{c=1}^{K} p(\boldsymbol{z}|c_i = c)\pi_c = \sum_{c \in \mathcal{N}(\mathcal{G})} \pi_c \varphi_d(\boldsymbol{U}\boldsymbol{b}_c) + \sum_{c \in \mathcal{E}(\mathcal{G})} \frac{\pi_c \sigma_{\boldsymbol{z}c} e^{\frac{t_{\boldsymbol{z}c}}{2}}}{(2\pi)^{\frac{d}{2}-1}} \left[\Phi\left(\frac{1-\nu_{\boldsymbol{z}c}}{\sigma_{\boldsymbol{z}c}}\right) - \Phi\left(-\frac{\nu_{\boldsymbol{z}c}}{\sigma_{\boldsymbol{z}c}}\right) \right].
$$

3. $\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$

   Since the approximate posterior density $q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$ is $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{Y}_i,\boldsymbol{X}_i}, \boldsymbol{\Sigma}_{\boldsymbol{Y}_i,\boldsymbol{X}_i}\right)$, we have

$$
\begin{aligned}
&\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i) \\
=&\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \left[-\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{d} \log \sigma_{\boldsymbol{Y}_i,\boldsymbol{X}_i,i}^2 - \frac{1}{2}\sum_{j=1}^{d} \frac{(Z_{ij} - \mu_{\boldsymbol{Y}_i,\boldsymbol{X}_i,j})^2}{\sigma_{\boldsymbol{Y}_i,\boldsymbol{X}_i,j}^2} \right] \\
=& -\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{d} \log \sigma_{\boldsymbol{Y}_i,\boldsymbol{X}_i,i}^2 - \frac{1}{2}\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})} \left(\sum_{j=1}^{d} \frac{(Z_{ij} - \mu_{\boldsymbol{Y}_i,\boldsymbol{X}_i,j})^2}{\sigma_{\boldsymbol{Y}_i,\boldsymbol{X}_i,j}^2} \right) \\
=& -\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{d} (\log \sigma_{\boldsymbol{Y}_i,\boldsymbol{X}_i,i}^2 + 1).
\end{aligned}
$$

C Posterior Estimation

1. $\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i$

   Since the approximate posterior distribution of $\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i$ is $\mathcal{N}_d\left(\boldsymbol{\mu}_{\boldsymbol{Y}_i,\boldsymbol{X}_i}, \boldsymbol{\Sigma}_{\boldsymbol{Y}_i,\boldsymbol{X}_i}\right)$, we can use $\boldsymbol{\mu}_{\boldsymbol{Y}_i,\boldsymbol{X}_i}$ as the latent representation of $\boldsymbol{Y}_i$.

2. $c_i|\boldsymbol{Y}_i, \boldsymbol{X}_i$

   It is worth noted that a useful property of our model is $\boldsymbol{Y}_i \perp\!\!\!\perp (w_i, c_i)|\boldsymbol{Z}_i, \boldsymbol{X}_i$, which is due to the fact that the probability mass function of $Y_{ig}|\boldsymbol{Z}_i, w_i, c_i, \boldsymbol{X}_i$ is $p(y|\boldsymbol{Z}_i, w_i, c_i, \boldsymbol{X}_i) = p(y|\boldsymbol{Z}_i, \boldsymbol{X}_i)$, which only depends on $\boldsymbol{Z}_i$ and $\boldsymbol{X}_i$. Note that we have already derived the formula for $p(\boldsymbol{z}|c_i = c)$ in Appendix B, we have

$$
\begin{aligned}
p(c|\boldsymbol{Y}_i, \boldsymbol{X}_i) &= \int_{\mathbb{R}} p(c, \boldsymbol{z}|\boldsymbol{Y}_i, \boldsymbol{X}_i)\mathrm{d}z \\
&\xlongequal{\boldsymbol{Y}_i|\boldsymbol{Z}_i,\boldsymbol{X}_i \perp\!\!\!\perp (w_i,c_i)|\boldsymbol{Z}_i,\boldsymbol{X}_i} \int_{\boldsymbol{z}} p(c|\boldsymbol{Z}_i = \boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{Y}_i, \boldsymbol{X}_i)\mathrm{d}\boldsymbol{z} \\
&= \mathbb{E}_{p(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \left[\frac{p(\boldsymbol{z}|c_i = c, \boldsymbol{X}_i)p(c, \boldsymbol{X}_i)}{p(\boldsymbol{z}, \boldsymbol{X}_i)} \right] \\
&\xlongequal{(\boldsymbol{Z}_i,c_i) \perp\!\!\!\perp \boldsymbol{X}_i} \mathbb{E}_{p(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \left[\frac{p(\boldsymbol{z}|c_i = c)p(c)}{p(\boldsymbol{z})} \right]
\end{aligned}
$$

$$\approx \frac{1}{L}\sum_{l=1}^{L}\frac{\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c)\widehat{\pi}_c}{\sum\limits_{c'=1}^{K}\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c')\widehat{\pi}_{c'}} \triangleq \widehat{p}(c|\boldsymbol{Y}_i,\boldsymbol{X}_i).$$

So we can obtained the predicted category of cell i by compute the MAP estimator

$$\widehat{c}_i = \arg\max_c \ \widehat{p}(c|\boldsymbol{Y}_i,\boldsymbol{X}_i).$$

3. $\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i$
   Note that

$$p(w,c|\boldsymbol{Y}_i,\boldsymbol{X}_i) \approx \frac{1}{L}\sum_{l=1}^{L}\frac{\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c,w_i=w)\widehat{\pi}_c}{\sum\limits_{c'=1}^{K}\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c')\widehat{\pi}_{c'}} \triangleq \widehat{p}(w,c|\boldsymbol{Y}_i,\boldsymbol{X}_i).$$

   Let $\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}$ and $\Sigma_{\tilde{\boldsymbol{w}}_i}$ denote the mean and the covariance matrix for the estimated distribution of $\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i$. We have

$$\begin{aligned}
\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i} &= \sum_{c=1}^{K}\int [wa_c+(1-w)b_c]\widehat{p}(w,c|\boldsymbol{Y}_i,\boldsymbol{X}_i)\mathrm{d}w \\
&\approx \tfrac{1}{M}\sum_{c=1}^{K}\sum_{j=1}^{M}\left[\tfrac{j}{M}a_c+\left(1-\tfrac{j}{M}\right)b_c\right]\tfrac{1}{L}\sum_{l=1}^{L}\frac{\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c,w_i=\frac{j}{M})\widehat{\pi}_c}{\sum\limits_{c'=1}^{K}\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c')\widehat{\pi}'_c} \\
\mathrm{diag}(\Sigma_{\tilde{\boldsymbol{w}}_i}) &\approx \tfrac{1}{M}\sum_{c=1}^{K}\sum_{j=1}^{M}\left[\tfrac{j}{M}a_c+\left(1-\tfrac{j}{M}\right)b_c\right]^2\tfrac{1}{L}\sum_{l=1}^{L}\frac{\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c,w=\frac{j}{M})\widehat{\pi}_c}{\sum\limits_{c'=1}^{K}\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c')\widehat{\pi}'_c}-\boldsymbol{\mu}^2_{\tilde{\boldsymbol{w}}_i}.
\end{aligned}$$

   Here, as the integration over w is intractable, we approximate the integral by the rectangular rule with equally spaced M points in $[0,1]$.

D Edge Score

For the complete graph $\mathcal{G}$, recall the definition of the categorical assignment matrix $\boldsymbol{C}$ in (5). Below four kinds of edge scores are presented.

1. Edge Score Based on Posterior Mean of c.
   The first edge score is based on the posterior mean $\frac{1}{N}\sum\limits_{i=1}^{N}p(c_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)$. Let

$$\mathcal{C}_{j_1j_2} = \{i:\mathbb{P}(c_i\in\{C_{j_1j_1},C_{j_1j_2},C_{j_2j_2}\}|\boldsymbol{Y}_i,\boldsymbol{X}_i)>t_1\}$$

   where $t_1=0.5$ is the threshold parameter. We are considering $\mathcal{C}_{j_1j_2}$ instead of the full data set because not every cell contributes to the existence of the edge $e_{j_1j_2}$. Within the set $\mathcal{C}_{j_1j_2}$, proportion of cells at the edge $e_{j_1j_2}$ gives us one score

$$\mathbb{E}\left[\frac{1}{|\mathcal{C}_{j_1j_2}|}\sum_{i\in\mathcal{C}_{j_1j_2}}\mathbb{1}_{\{c_i=C_{j_1j_2}\}}\right] \approx \frac{1}{|\widehat{\mathcal{C}}_{j_1j_2}|}\sum_{i\in\widehat{\mathcal{C}}_{j_1j_2}}\frac{\widehat{p}\left(C_{j_1j_2}|\boldsymbol{Y}_i,\boldsymbol{X}_i\right)}{\widehat{p}(C_{j_1j_1}|\boldsymbol{Y}_i,\boldsymbol{X}_i)+\widehat{p}(C_{j_1j_2}|\boldsymbol{Y}_i,\boldsymbol{X}_i)+\widehat{p}(C_{j_2j_2}|\boldsymbol{Y}_i,\boldsymbol{X}_i)},$$

which is denoted by $s_{j_1j_2}^{\text{mean}}$, where $\widehat{\mathcal{C}}_{j_1j_2} = \{i : \widehat{p}(C_{j_1j_1}|\boldsymbol{Y}_i, \boldsymbol{X}_i) + \widehat{p}(C_{j_1j_2}|\boldsymbol{Y}_i, \boldsymbol{X}_i) + \widehat{p}(C_{j_2j_2}|\boldsymbol{Y}_i, \boldsymbol{X}_i) > t_1\}$.

2. Edge Score Based on Modified Posterior Mean of c.
   A modified version of $s_{j_1j_2}^{\text{mean}}$ is given by

$$s_{j_1j_2}^{\text{modified_mean}} = \frac{\sum\limits_{i\in\widehat{\mathcal{C}}_{j_1j_2}} \widehat{p}\left(C_{j_1j_2}|\boldsymbol{Y}_i, \boldsymbol{X}_i\right)}{\sum\limits_{i\in\widehat{\mathcal{C}}_{j_1j_2}} \left[\widehat{p}(C_{j_1j_1}|\boldsymbol{Y}_i, \boldsymbol{X}_i) + \widehat{p}(C_{j_1j_2}|\boldsymbol{Y}_i, \boldsymbol{X}_i) + \widehat{p}(C_{j_2j_2}|\boldsymbol{Y}_i, \boldsymbol{X}_i)\right]}$$

   where an first-order approximation $\mathbb{E}\left(\frac{X}{Y}\right) \approx \frac{\mathbb{E}X}{\mathbb{E}Y}$ is used.

3. Edge Score Based on MAP of c.
   Alternatively, we can also compute the edge score based on the MAP of c_i's,

$$\tilde{c}_i = \underset{c\in\{1,2,\cdots,K\}}{\arg\max} \ \widehat{p}(c_i = c|\boldsymbol{Y}_i, \boldsymbol{X}_i).$$

   Then an edge score can be defined as the proportion of the number of cells at the edge $e_{j_1j_2}$ to the number of cells at the vertices j_1, j_2 or the edge $e_{j_1j_2}$,

$$s_{j_1j_2}^{\text{map}} = \frac{|\{i : \tilde{c}_i = C_{j_1j_2}\}|}{|\{i : \tilde{c}_i \in \{C_{j_1j_1}, C_{j_1j_2}, C_{j_2j_2}\}\}|}.$$

4. Edge Score Based on Modified MAP of c.
   Since the assigned categories by MAP of c are generally at the edges instead of the nodes, the estimated denominator will be smaller than the real one, and $s_{j_1j_2}^{\text{map}}$ will be large when the number of cells in nodes j_1 and j_2 are small. To address this problem, we further modify the denominator, to get the modified MAP edge score,

$$s_{j_1j_2}^{\text{modified_map}} = \frac{|\{i : \tilde{c}_i = C_{j_1j_2}\}|}{|\{i : \boldsymbol{e}_{j_1}^\top \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i} > 0.5 \text{ or } \boldsymbol{e}_{j_1}^\top \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i} > 0.5\}|}.$$

E Proof of Proposition 1

Proof. Note that

$$\begin{aligned}
&\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\|\tilde{\boldsymbol{w}}_i - \boldsymbol{w}\|_2^2 = \mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}(\tilde{\boldsymbol{w}}_i^\top \tilde{\boldsymbol{w}}_i) - 2\boldsymbol{w}^\top \tilde{\boldsymbol{w}}_i + \boldsymbol{w}^\top \boldsymbol{w} \\
&\|\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\tilde{\boldsymbol{w}}_i - \boldsymbol{w}\|_2^2 = (\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\tilde{\boldsymbol{w}}_i)^\top \mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\tilde{\boldsymbol{w}}_i - 2\boldsymbol{w}^\top \tilde{\boldsymbol{w}}_i + \boldsymbol{w}^\top \boldsymbol{w},
\end{aligned}$$

we have

$$\begin{aligned}
\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\|\tilde{\boldsymbol{w}}_i - \boldsymbol{w}\|_2^2 = &\|\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\tilde{\boldsymbol{w}}_i - \boldsymbol{w}\|_2^2 \\
&- (\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\tilde{\boldsymbol{w}}_i)^\top \mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\tilde{\boldsymbol{w}}_i + \mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}(\tilde{\boldsymbol{w}}_i^\top \tilde{\boldsymbol{w}}_i),
\end{aligned}$$

where the last two terms do not involve $\boldsymbol{w}$. So, minimizing $\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\|\tilde{\boldsymbol{w}}_i - \boldsymbol{w}\|_2^2$ is equivalent to minimizing $\|\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\tilde{\boldsymbol{w}}_i - \boldsymbol{w}\|_2^2$ over $\boldsymbol{w}$. The optimization problem (11) is thus equivalent to the optimization problem (12).

We divide the problem into two cases: finding the best projection onto edges and the best projection onto vertices.

(1) For projection onto edges, the original problem becomes

$$\min_{(j_1,j_2)\in\mathcal{E}(\widehat{\mathcal{B}})} \quad (\boldsymbol{w}_{j_1} - \mu_{j_1})^2 + (\boldsymbol{w}_{j_2} - \mu_{j_2})^2 - (\mu_{j_1}^2 + \mu_{j_2}^2)$$

$$\text{s.t.} \quad \|\boldsymbol{w}\|_1 = w_{j_1} + w_{j_2} = 1, \quad \boldsymbol{w} \succeq \mathbf{0}_k.$$

Notice that for any edge $(j_1, j_2) \in \mathcal{E}(\widehat{\mathcal{B}})$, the objective function is quadratic in w_{j_1} (and w_{j_2}) since

$$\begin{aligned}
&(w_{j_1} - \mu_{j_1})^2 + (w_{j_2} - \mu_{j_2})^2 - (\mu_{j_1}^2 + \mu_{j_2}^2) \\
=&(w_{j_1} - \mu_{j_1})^2 + (1 - w_{j_1} - \mu_{j_2})^2 - (\mu_{j_1}^2 + \mu_{j_2}^2) \\
=&2\left(w_{j_1} - \frac{1 + \mu_{j_1} - \mu_{j_2}}{2}\right)^2 + \frac{(1 - \mu_{j_1} - \mu_{j_2})^2}{2} \\
\geq&\frac{(1 - \mu_{j_1} - \mu_{j_2})^2}{2},
\end{aligned}$$

with equality holds if and only if $w_{j_1} = \mu_{j_1} + \frac{1-\mu_{j_1}-\mu_{j_2}}{2}$. That is, for each edge $(j_1, j_2) \in \mathcal{E}(\widehat{\mathcal{B}})$, the solution

$$w_{k'} = \begin{cases} \mu_{j_1} + \frac{1-\mu_{j_1}-\mu_{j_2}}{2}, & k' = j_1 \\ \mu_{j_2} + \frac{1-\mu_{j_1}-\mu_{j_2}}{2}, & k' = j_2 \\ 0, & \text{otherwise}, \end{cases}$$

achieves minimum projection error

$$\|\boldsymbol{w} - \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}\|_2^2 = \sum_{k' \neq j_1, j_2} \mu_{k'}^2 + \frac{(1 - \mu_{j_1} - \mu_{j_2})^2}{2}.$$

If we want edge (j_1, j_2) to have the smallest projection error, then for any other edge $(m_1, m_2) \in \mathcal{E}(\widehat{\mathcal{B}})$, we need to have

$$\begin{aligned}
& \|\text{proj}_{(j_1,j_2)}(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}}) - \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}\|_2^2 \leq \|\text{proj}_{(m_1,m_2)}(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}}) - \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}\|_2^2 \\
\iff \quad & \sum_{k' \neq j_1, j_2} \mu_{k'}^2 + \frac{(1 - \mu_{j_1} - \mu_{j_2})^2}{2} \leq \sum_{k' \neq m_1, m_2} \mu_{k'}^2 + \frac{(1 - \mu_{m_1} - \mu_{m_2})^2}{2} \\
\iff \quad & (\mu_{j_1} - \mu_{j_2})^2 + 2(\mu_{j_1} + \mu_{j_2}) \geq (\mu_{m_1} - \mu_{m_2})^2 + 2(\mu_{m_1} + \mu_{m_2}).
\end{aligned} \tag{16}$$

Thus the best projected edge can be obtained by computing

$$(j_1^*, j_2^*) = \underset{(j_1,j_2) \in \mathcal{E}(\widehat{\mathcal{B}})}{\arg\max} \quad (\mu_{j_1} - \mu_{j_2})^2 + 2(\mu_{j_1} + \mu_{j_2}).$$

(2) For the vertex $j \in \mathcal{N}(\widehat{\mathcal{B}})$, the problem becomes

$$\min_{\boldsymbol{w}} \quad (w_j - \mu_j)^2 \qquad \text{s.t.} \quad \|\boldsymbol{w}\|_1 = w_j = 1, \;\; \boldsymbol{w} \succeq \mathbf{0}_k,$$

and the solution is $\text{proj}_j(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}}) = \boldsymbol{e}_j$. The projection error for the vertex i is given by

$$\|\text{proj}_j(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}}) - \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}}\|_2^2 = \sum_{k' \neq j} \mu_{k'}^2 + (\mu_j - 1)^2.$$

Then

$$j_3^* \in \underset{j \in \mathcal{N}(\widehat{\mathcal{B}})}{\arg\min} \|\text{proj}_j(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}}) - \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}}\|_2^2$$

gives the best projection onto vertices.

(3) Consider the following case,

$$
\begin{aligned}
& \|\text{proj}_{j_3^*}(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}}) - \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}\|_2^2 \leq \|\text{proj}_{(j_1^*, j_2^*)}(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}}) - \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}\|_2^2 \\
\iff \quad & \sum_{k' \neq j_3^*} \mu_{k'}^2 + (1 - \mu_{j_3^*})^2 \leq \sum_{k' \neq j_1^*, j_2^*} \mu_{k'}^2 + \frac{(1 - \mu_{j_1^*} - \mu_{j_2^*})^2}{2} \\
\iff \quad & \mu_{j_1^*}^2 + \mu_{j_2^*}^2 + (1 - \mu_{j_3^*})^2 \leq \mu_{j_3^*}^2 + \frac{(1 - \mu_{j_1^*} - \mu_{j_2^*})^2}{2}, \\
\iff \quad & 4\mu_{j_3^*} - 1 \geq (\mu_{j_1^*} - \mu_{j_2^*})^2 + 2(\mu_{j_1^*} + \mu_{j_2^*}) \\
\iff \quad & (\mu_{j_3^*} - \mu_{j_3^*})^2 + 2(\mu_{j_3^*} + \mu_{j_3^*}) - 1 \geq (\mu_{j_1^*} - \mu_{j_2^*})^2 + 2(\mu_{j_1^*} + \mu_{j_2^*}).
\end{aligned}
$$

Define $\mathcal{EN}(\mathcal{G}) = \mathcal{E}(\mathcal{G}) \cup \{(j, j) : j \in \mathcal{N}(\mathcal{G})\}$, then

$$
\underset{(j_1, j_2) \in \mathcal{EN}(\widehat{\mathcal{B}})}{\arg\max} \quad (\mu_{j_1} - \mu_{j_2})^2 + 2(\mu_{j_1} + \mu_{j_2}) - \mathbb{1}_{\{j_1 = j_2\}}
$$

gives solutions to the optimization problems (11) and (12). ■

F Discussion on Uncertainty Quantification

We observed in Section 6 that our calculated projection MSE is always smaller for the cells near vertices. Why do we observe such a pattern? Is it due to the estimation bias in our approximation of the posterior distribution, or is it an intrinsic property of the L2 loss under our mixture model? In order to answer these questions, we consider the case where we observe the latent space $\boldsymbol{Z}_i$, so that we can compute the true posterior distributions of $\boldsymbol{Z}_i | \tilde{\boldsymbol{w}}_i$. Also, we focus on $\text{tr}(\Sigma_{\tilde{\boldsymbol{w}}_i})$, as from our observations it is typically the leading term in (13). Specifically, we define our hierarchical model with observed $\boldsymbol{Z}_i$ following (3) and (6) by setting $d = 2$, $k = 3$ with $\boldsymbol{U} = \begin{pmatrix} 0 & 0.5 & 2 \\ 2 & 0.5 & 0 \end{pmatrix}$. The defined backbone is shown in Figure 7a and we choose the vertex 1 as the root of the trajectory.

First, we take $\boldsymbol{\pi} = \begin{pmatrix} \frac{1}{5} & \frac{1}{5} & 0 & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{pmatrix}$, so that the edges and vertices have equal probabilities. As $\boldsymbol{Z}_i$ is observed, we can compute the true posterior mean and variances from Appendix C for any given $\boldsymbol{Z}_i$. Here we show the posterior variances given $\boldsymbol{Z}_i = \boldsymbol{U}\tilde{\boldsymbol{w}}_i$ which are exactly on the trajectory and discuss how the variances change with $T_i = \boldsymbol{o}^\top \tilde{\boldsymbol{w}}_i$. The cell i is at the vertices when $T_i = 0, 1$ or 2. Figure 7b shows how the posterior variances of each $\tilde{w}_{ij}$ change with T_i. The leading term of the projection MSE $\text{tr}(\Sigma_{\tilde{\boldsymbol{w}}_i})$ summing up the three terms in Figure 7b has an M-shape as shown in Figure 7c, indicating that the cells indeed have smaller projection uncertainties when they are near the vertices.

One explanation of the above phenomenon is that as there are nonzero probabilities exactly at the vertices, the distribution is denser near the vertices. As a consequence, there are less uncertainty on $\tilde{\boldsymbol{w}}_i$ if it is closer to a vertex. However, what we find surprising is that when we set $\boldsymbol{\pi} = \begin{pmatrix} 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \end{pmatrix}$, where the cells can only be on the edges, we still observe an M-shape for the change of $\text{tr}(\Sigma_{\tilde{\boldsymbol{w}}_i})$ along T_i. In other words, the projection MSE from true posteriors is smaller near the vertices no matter whether the vertices have nonzero probabilities or not, though the difference is smaller when the vertices have zero probabilities. As a consequence, we believe that this pattern is an intrinsic property of the L_2 loss under our mixture model.

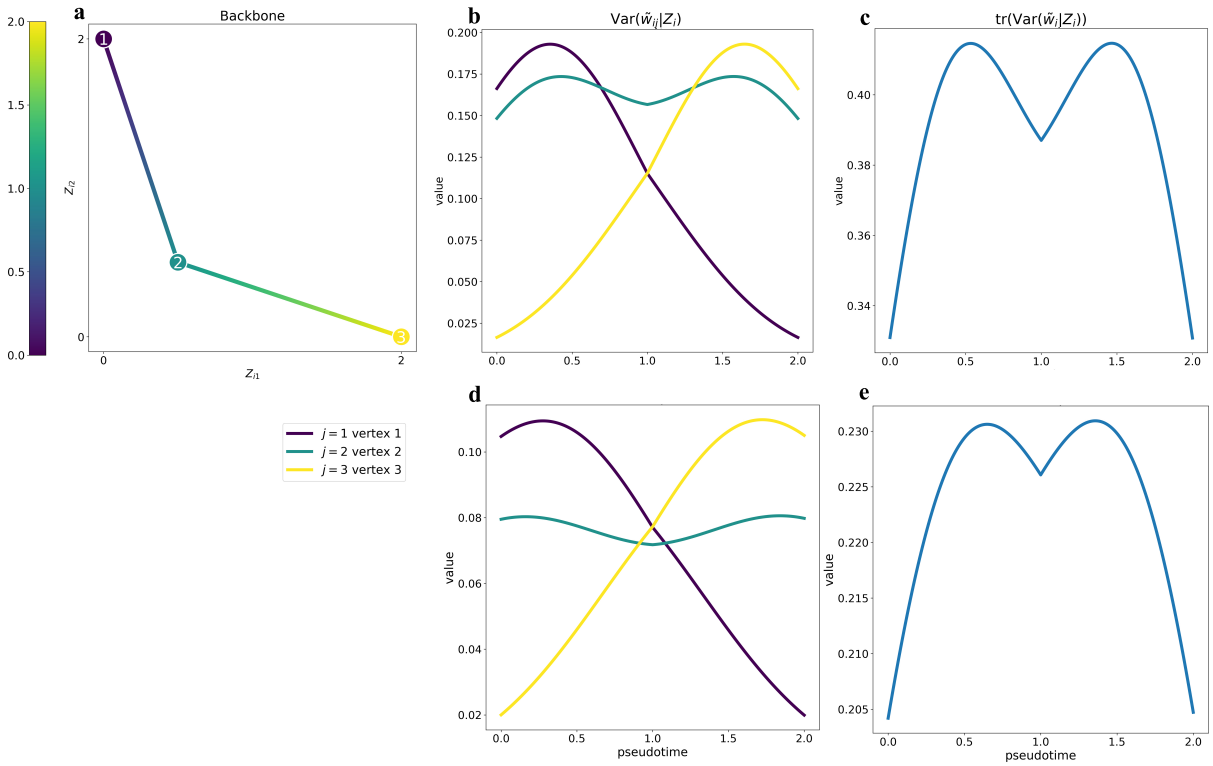

Figure 7: **(a)** The underlying backbone of the assumed model colored by pseudotime. **(b)** The values of $\mathrm{Var}(\tilde{w}_{ij}|\boldsymbol{Z}_i)$ along the trajectory and **(c)** the values of $\mathrm{tr}(\mathrm{Var}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Z}_i))$ along the trajectory are for the case when $\boldsymbol{\pi} = (0.2, 0.2, 0, 0.2, 0.2, 0.2)$. **(d-e)** are the same as **(b-c)** for the case when $\boldsymbol{\pi} = (0, 0.5, 0, 0, 0.5, 0)$.