Joint Trajectory Inference for Single-cell Genomics Using Deep Learning with a Mixture Prior

Jin-Hong Du[ab1], Tianyu Chen[c1], Ming Gao[d], Jingshu Wang[e]

[a]Department of Statistics and Data Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[b]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[c]Department of Statistics and Data Science, The University of Texas at Austin, Austin, TX 78712, USA
[d]Booth School of Business, University of Chicago, Chicago, IL 60637, USA
[e]Department of Statistics, University of Chicago, Chicago, IL 60637, USA
[1]Equal contribution.

Abstract

Trajectory inference methods are essential for analyzing the developmental paths of cells in single-cell sequencing datasets. It provides insights into cellular differentiation, transitions, and lineage hierarchies, helping unravel the dynamic processes underlying development, disease progression, and tissue regeneration. However, many existing tools for trajectory inference lack a cohesive statistical model and reliable uncertainty quantification, limiting their utility and robustness. In this paper, we introduce VITAE (**V**ariational **I**nference for **T**rajectory by **A**uto**E**ncoder), a novel statistical approach that integrates a latent hierarchical mixture model with variational autoencoders to infer trajectories. The statistical hierarchical model underpinning the approach enhances interpretability, facilitating the exploration of differential gene expression patterns along the trajectory, while the posterior approximations generated by the variational autoencoder ensure computational efficiency, flexibility, and scalability. Notably, VITAE uniquely enables simultaneous trajectory inference and data integration across multiple datasets, enhancing accuracy in both tasks. We show that VITAE outperforms other state-of-the-art trajectory inference methods on both real and synthetic data under various trajectory topologies. Furthermore, we apply VITAE to jointly analyze three distinct single-cell RNA sequencing datasets of the mouse neocortex, unveiling comprehensive developmental lineages of projection neurons. VITAE effectively mitigates batch effects within and across datasets, aligning cells to elucidate clear trajectories and uncover finer structures that might be overlooked in individual datasets. Additionally, we showcase VITAE's efficacy in integrative analyses of multi-omic datasets with continuous cell population structures.

Keywords— single-cell sequencing, data integration | hierarchical Bayesian models, deep generative models, Jacobian regularizer

Introduction

Single-cell genomics has emerged as an indispensable tool for biologists seeking to unravel cellular diversities and understand cell activities [1, 2]. Many biological processes, including differentiation, immune response, and cancer progression, can be comprehended as continuous dynamic changes within the space of cell types or cell states [3]. Instead of being limited to discrete cell types, cells often display a continuous spectrum of states and actively transition between different cellular states. To tackle this complexity, trajectory inference (TI) has been developed as a computational approach to explore the sequential states of measured cells using single-cell genomics data [4].

While various computational tools have been developed for TI [5–8], there remains a lack of clear definition for the terms "trajectory" and "pseudotime" that can be identified and estimated from single-cell genomics data. Single-cell sequencing data captures only a static snapshot of single-cell omics at specific time points, which may make it challenging to uncover the true temporal lineages sought after by biologists [9, 10]. As a consequence, it becomes essential to construct an explicit and coherent statistical model on single-cell sequencing data that incorporates the trajectory structure. Such a model would also allow for discussions about the efficiency and accuracy of TI methods. Moreover, with the expansion of single-cell sequencing data [11], multiple datasets from different individuals or labs, measuring different omics are increasingly available for the same or closely related biological processes. However, to the best of our knowledge, no existing TI method can simultaneously integrate multiple datasets and infer a shared underlying trajectory while preserving true biological differences across datasets.

In this paper, we present a novel method **VITAE** (**V**ariational **I**nference for **T**rajectory by **A**uto**E**ncoder) for performing TI. VITAE combines a graph-based hierarchical mixture model, which incorporates the trajectory structure within a low-dimensional latent space, with deep neural networks that non-linearly map the latent space to the high-dimensional observed data. We offer a clear definition of both the trajectory backbone and the pseudotime while also providing scalable uncertainty quantification of the estimates through approximate Bayesian inference. To train VITAE, we employ a modified variational autoencoder and introduce the Jacobian regularizer, an additional penalty in the loss function that significantly enhances the stability and accuracy of our algorithm. Moreover, VITAE demonstrates a reliable ability to infer a shared trajectory from multiple datasets by flexible modeling and effectively accounting for confounding variables such as batch effects. With VITAE, we provide comprehensive joint trajectory analyses of three single-cell RNA sequencing (scRNA-seq) datasets on mouse neocortex. We also performed a joint analysis integrating scRNA-seq with the single-cell assay for transposase-accessible chromatin by sequencing (scATAC-seq) of hematopoietic stem cells.

Results

Method overview. We begin our approach by modeling the trajectory backbone of cells as a subgraph of a complete graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where each of the k vertices represents a distinct cell state, and each edge

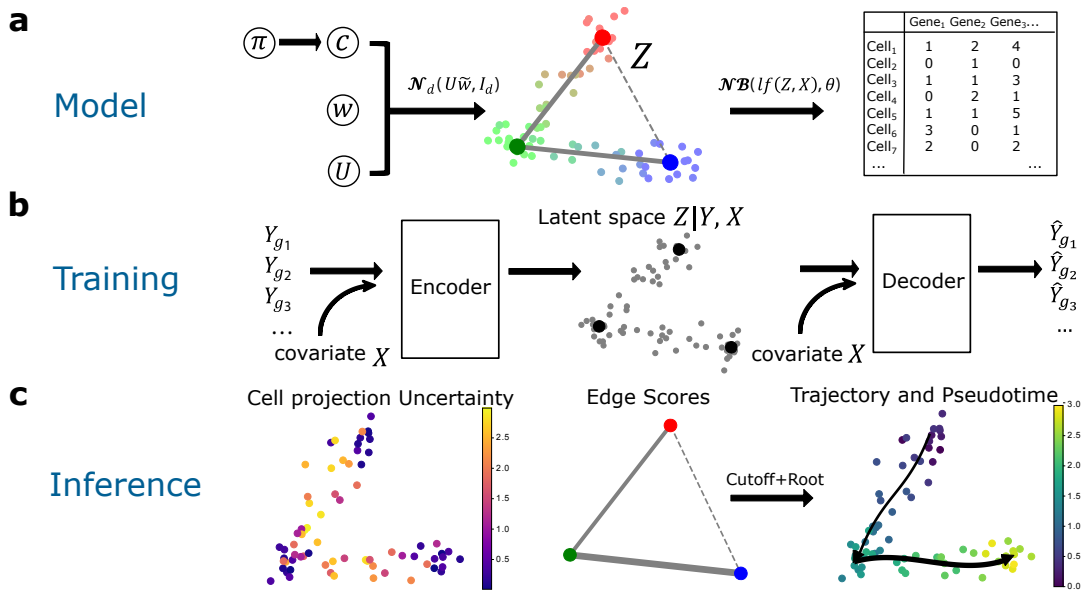

Figure 1: An overview of VITAE. **(a)** Model framework: A hierarchical model is used on the latent representations of cells, which are subsequently nonlinearly mapped to the high-dimensional observed count data. **(b)** Model training: Neural networks are used to approximate the nonlinear mapping to the observed data (decoder) as well as the posterior distributions of the latent space (encoder). This process also allows for adjustment for confounding covariates. **(c)** Trajectory inference: By utilizing posterior approximations, VITAE infers the trajectory backbone by assigning edge scores (represented as the width of gray lines), quantifies the uncertainty in estimated cell positions along the trajectory, and derives estimates for the pseudotime of each cell.

represents a potential transition between two states. As shown in Fig. 1a, to capture the scenario where a cell i can either belong to a specific state or be in the process of transitioning between states, we define its position $\tilde{\boldsymbol{w}}_i \in [0,1]^k$ on the graph as either on a vertex if it is in a steady state or on an edge if it is undergoing a transition. Specifically, we define

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

where $\boldsymbol{e}_j$ is a one-hot vector with the j-th element as 1 and all other elements as 0. The value $w_i \in [0,1]$ represents the relative position of cell i along an edge, indicating its progress along the transition. The goal of trajectory inference is to identify the trajectory backbone, which is a subgraph containing cells positioned on both edges and vertices, as well as to estimate the cell positions $\tilde{\boldsymbol{w}}_i \in [0,1]^k$, representing the pseudotime of cells along the underlying developmental process (see Methods). By quantitatively defining the developmental trajectory and pseudotime as features of a graph, VITAE offers enhanced interpretability and flexibility compared to other existing methods.

To connect the observed single-cell sequencing data with the underlying trajectory backbone graph in the

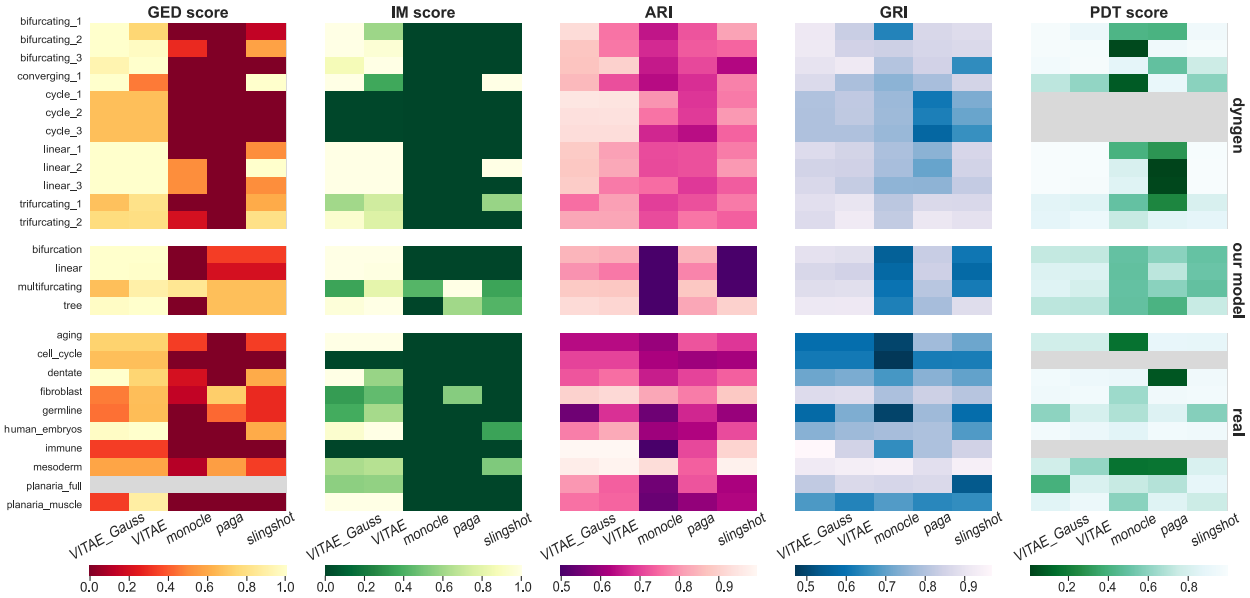

Figure 2: Evaluation results of VITAE and VITAE_Gauss compared with three other TI methods: Monocle 3, PAGA, and Slingshot, on 25 datasets using 5 evaluation scores. A larger score (lighter color) indicates better performance. Each row represents one dataset (Table S1). The PDT scores for datasets with 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. The scores for our methods are averaged over 100 runs with different random seeds.

latent space, we model the observed count Y_{ig} for any cell i and gene g as negative binomial distribution:

$$Y_{ig}|\boldsymbol{Z}_i, \boldsymbol{X}_i \overset{\text{ind}}{\sim} \mathcal{NB}\left(l_i f_g(\boldsymbol{Z}_i, \boldsymbol{X}_i), \theta_g\right) \tag{1}$$

Here, l_i represents the library size of cell i, and θ_g is the dispersion parameter of gene g. The variable $\boldsymbol{Z}_i \in \mathbb{R}^d$ represents the cell i in a latent lower-dimensional space, and its values are linearly dependent on the cell's relative position along the trajectory backbone:

$$\boldsymbol{Z}_i|\tilde{\boldsymbol{w}}_i \sim \mathcal{N}_d(\boldsymbol{U}\tilde{\boldsymbol{w}}_i, \boldsymbol{I}_d)$$

where $\boldsymbol{U} \in \mathbb{R}^{d\times k}$ denotes the positions of the vertices on the latent space. To integrate multiple datasets, VITAE accounts for the effect of confounding variable $\boldsymbol{X}_i$, which could include both discrete and continuous variables such as the dataset ID, batch indices, and cell cycle scores. We incorporate a flexible function $f_g(\boldsymbol{Z}_i, \boldsymbol{X}_i)$, where $f_g(\cdot)$ is represented by a neural network. By employing this setup, VITAE is capable of jointly performing data integration and trajectory inference, resulting in improved accuracy in both tasks.

VITAE is trained using a modified variational autoencoder (VAE, Fig. 1b). Based on the training dataset $\mathcal{D}_N = \{(\boldsymbol{X}_1, \boldsymbol{Y}_1), \ldots, (\boldsymbol{X}_N, \boldsymbol{Y}_N)\}$, one key idea is the incorporation of four penalty terms in the

likelihood-based loss function, defined as follows:

$$
\begin{aligned}
L = & -(1-\alpha)\sum_{i=1}^{N}\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i,\boldsymbol{X}_i) \\
& +\beta\sum_{i=1}^{N}D_{\mathrm{KL}}(q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)\|p(\boldsymbol{Z}_i)) && (2)\\
& -\alpha\sum_{i=1}^{N}\log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i=\boldsymbol{0}_d,\boldsymbol{X}_i) && (3)\\
& +\kappa\,\Omega_{\mathrm{MMD}}(\mathcal{D}_N) && (4)\\
& +\gamma\,\Omega_{\mathrm{Jacobian}}(\mathcal{D}_N). && (5)
\end{aligned}
$$

Compared to standard VAE, the prior $p(\boldsymbol{Z}_i)$ contains the trajectory structure, and the term [2] encourages the posterior positions of $\boldsymbol{Z}_i$ to lie along the trajectory backbone. Then both terms [3] and [4] help adjust for the confounding variables $\boldsymbol{X}_i$ and correct for batch effects. The soft penalty term [3] can effectively preserve biologically meaningful differences across datasets [12] and can also adjust for continuous confounding variables in $\boldsymbol{X}_i$ such as the cell cycle scores. On the other hand, the MMD loss [4] acts as a stronger penalty, ensuring the removal of unwanted variations, and is used when cell populations across batches (e.g., replicates) are known to be exactly the same [13]. Finally, the introduction of the Jacobian regularizer [5] enhances stability in optimization [14] and is a novel contribution to the analysis of single-cell sequencing data, making the estimation procedure robust to small perturbations in the observed counts. More details on the model architecture, loss function, and training procedures are available in Methods and Appendix.

Systematic benchmarking with real and synthetic Datasets. We systematically evaluate the performance of VITAE on 25 real and synthetic single-cell RNA sequencing datasets (Table S1), with various trajectory topologies (Fig. S1). We examine two versions of VITAE: the original likelihood-based VITAE and its accelerated Gaussian counterpart, VITAE_Gauss (see Methods), that make VITAE scalable to handle large datasets. Our assessment involves a comparative analysis against three established TI methods: PAGA [8], Monocle 3 [15], and Slingshot [7]. To ensure an equitable evaluation, we largely adhere to the evaluation framework presented in a third-party benchmarking study [5], and the other three methods are executed through dyno [5], a standardized interface for benchmarking TI methods. Drawing inspiration from the benchmarking study [5], we employ five metrics for assessment. The GED score and IM score measure the accuracy of trajectory backbone recovery. Meanwhile, the adjusted rand index (ARI) and the generalized rand index (GRI) evaluate the precision of predicted cell positions along the trajectory, the latter being independent of clustering. Lastly, the PDT score quantifies the similarity between reference and estimated pseudotime values for cells. All five scores are normalized to a scale of 0 to 1, with higher scores indicating superior performance.

The evaluation results are summarized in Fig. 2. Both VITAE and VITAE_Gauss consistently outperform the other methods. In particular, VITAE demonstrates enhanced accuracy in recovering trajectory backbones, subsequently leading to improved pseudotime estimations. Regarding ARI and GRI, VITAE displays

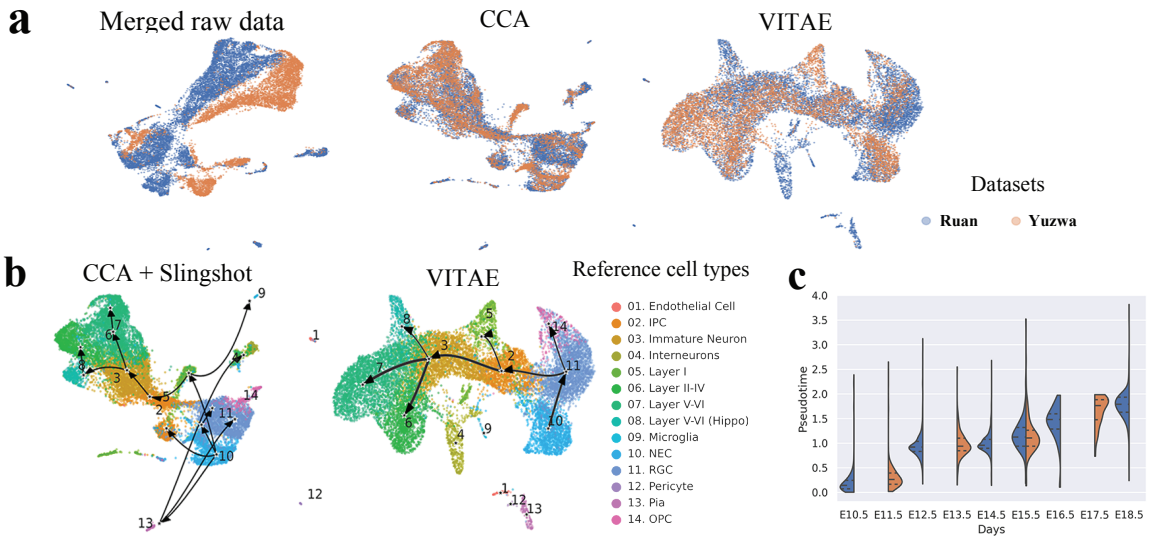

Figure 3: UMAP visualizations of the mouse neocortex in Yuzwa and Ruan datasets. (a) Low-dimensional embeddings colored by dataset sources. We compare the UMAP plots of the cells from the merged raw data, after Seurat CCA integration and by VITAE. The cell cycle scores are adjusted for both Seurat CCA and VITAE. **(b)** The estimated trajectories from Seurat CCA + Slingshot (left) compared with that from the accelerated Gaussian version of VITAE (right). Cells are colored by their reference cell types given in the original papers. For VITAE, the width of each edge is proportional to its edge score. **(c)** The violin plots of estimated pseudotime by VITAE across different embryonic days in the sub-trajectory *NEC-RGC-OPC*.

consistent enhancement across all synthetic datasets, although the improvements in certain real datasets are subtle due to the limited resolution of reference cell positions in real datasets.

Joint analyses of two single-cell RNA sequencing data on developing mouse neocortex. The six-layered neocortex forms the physical center for the highest cognitive functions in mammals [16]. It has been shown that the neuroepithelial cells (NEC) transition into the radial glial cells (RGC), while cortical projection neuron types are generated sequentially by RGCs and intermediate progenitor cells (IPC) [17, 18]. In our study, we employ VITAE to infer a unified trajectory framework through the integration of two distinct single-cell RNA sequencing datasets pertaining to the developing mouse neocortex. Specifically, we delve into Yuzwa's dataset [19], encompassing $6,390$ cortical cells sampled from mouse embryos at E11.5, E13.5, E15.5, and E17.5 time points. Additionally, we incorporate Ruan's dataset [20], which has $10,261$ cells derived from the mouse embryo cortex at E10.5, E12.5, E14.5, E15.5, E16.5, and E18.5 (Table S2). Thus, these datasets capture cells from disparate developmental days within the same brain region. By jointly analyzing these datasets, we unlock the potential to investigate the dynamic progression spanning a comprehensive developmental timeline, from E10.5 to E18.5 in mice.

We conducted a comparison between VITAE and an alternative approach, starting with integration using Seurat CCA [21], followed by trajectory inference using Slingshot on the integrated embeddings. While both VITAE and Seurat CCA are effective at cell mixing and demonstrate comparable retention of reference cell types post-integration, VITAE excels in providing a more refined trajectory inference on the shared structure (Fig. 3ab). In particular, VITAE successfully identified the primary cell lineage, denoted as

NEC-RGC-IPC-Neurons, while excluding microglia cells, pericytes, Pia, and interneurons from the Ruan dataset that do not align with the main trajectory. On the contrary, Slingshot yields disorderly and ambiguous outcomes, attributed in part to subtle imperfections within the Seurat integration process and its inability to recognize disconnected cell types.

Regarding uncertainty quantification, VITAE supplies edge scores for each inferred trajectory edge (Fig. 3b, displayed as the edge widths), signifying our level of confidence regarding edge existence. Additionally, VITAE furnishes estimated mean square errors for the projection of each cell onto the inferred trajectory structure (Fig. S2a).

Moreover, we observe robust correlations between the developmental days of cell collection and the pseudotime estimation along the sub-trajectory *NEC-RGC-OPC* (Fig. 3c, Fig. S2b-d). Although cells collected on even days are from the Ruan dataset, and those collected on odd days mainly originate from the Yuzwa dataset, VITAE effectively aligns cells along pseudotime gradually according to their developmental days. This indicates VITAE's ability to preserve biologically meaningful distinctions across datasets while simultaneously learning a shared trajectory structure.

Integration with the mouse cerebral cortex atlas. Extending our analysis to encompass a more intricate system, we perform an additional shared trajectory inference analysis involving the Ruan and Yuzwa datasets in conjunction with the Di Bella dataset [22] – a comprehensive mouse neocortex atlas with 91,648 cells that contains daily samples from the neocortex throughout embryonic corticogenesis and at the early stage of postnatal development. Due to differences in sequencing technologies, the raw Di Bella dataset exhibits pronounced distinctions in comparison to the Ruan and Yuzwa datasets (Fig. S3). Additionally, substantial batch effects manifest across cells gathered from various embryonic days within the Di Bella dataset, even spanning distinct technical replicates. We aim to use VITAE to mitigate these batch effects within the Di Bella dataset and project cells from the Ruan and Yuzwa datasets onto the Di Bella dataset, thereby facilitating a more comprehensive and integrated analysis.

We compare VITAE with the alternative approach using Seurat CCA for integration and Slingshot for trajectory inference. Additionally, we benchmark with Monocle 3 [15], as employed in the original paper [22], to undertake trajectory inference. Monocle 3 facilitates data integration as its preliminary step and we leverage this to integrate the three datasets before embarking on trajectory inference when using Monocle 3. In alignment with the outcomes presented in the preceding section, VITAE can reveal a more refined shared trajectory structure when juxtaposed against the other two methodologies (Fig. 4a), recovering the main branches identified in the original paper [22] of Di Bella dataset.

We also assess the efficacy of data integration across the three methodologies. The intricacy of data integration arises from the fact that cells originating from different embryonic days possess inherent biological distinctions, albeit with varying degrees of separation. Our objective is to appropriately rectify batch effects across embryonic days while retaining genuine biological variances. First, considering the presence of two technical replicates for Day E18 and Day P1 within Di Bella's dataset, VITAE has the highest median replicate mixing scores (Appendix S4.2) in both the two replicates (Fig. 4b). In contrast, Monocle 3 exhibits the poorest performance in terms of cell type separation, implying an excessive removal of biologically

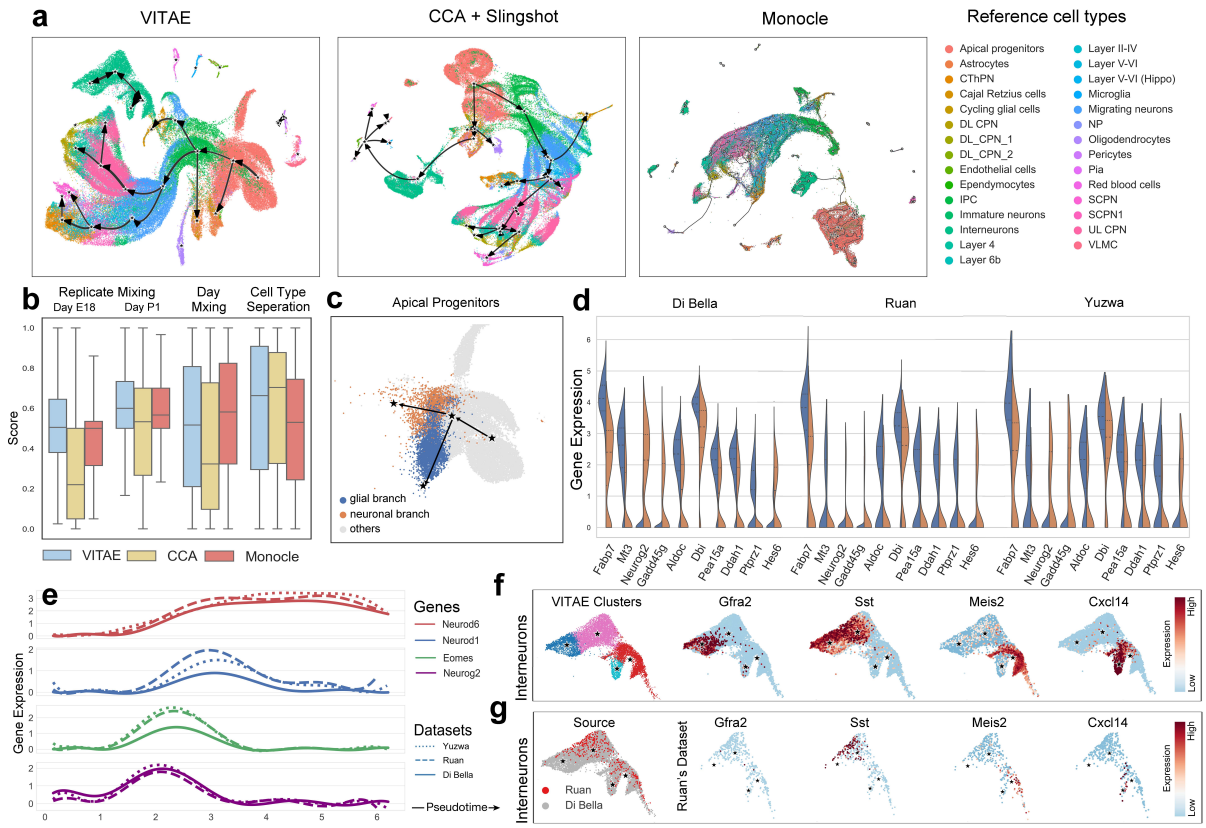

Figure 4: Integrative trajectory analysis on three mouse brain datasets. **(a)** The estimated trajectories on the merged dataset using different methods. Cells are colored by their reference cell types provided in the original paper. **(b)** Comparison of data integration efficacy across three methods measured by the replicate mixing scores, day mixing scores, and cell type separation scores. A higher score indicates better performance. **(c)** Using VITAE, some APs are projected onto the glial branch (blue) or the neural branch (yellow). All other APs are colored gray. **(d)** The violin plots of the top 10 differentially expressed gene expressions between the APs in the glial (blue) and neuronal (orange) branches. Tests are done by combining all three datasets while gene expressions are visualized for each dataset separately. **(e)** Smoothed curve for gene expression change along the estimated pseudotime. **(f)** Four interneuron subtypes identified by VITAE. VITAE automatically identifies four vertices within the interneurons though a single initial vertex is given. **(g)** Interneurons from the Ruan dataset positioned within the four interneurons subtypes.

meaningful cell-type-specific signals. Seurat CCA achieves a comparable degree of cell type separation to VITAE. However, it inadvertently maintains a complete separation between cells from distinct embryonic days (Fig. 4b, Fig. S3), indicating excessive retention of batch effects.

Leveraging the inferred trajectory generated by VITAE, we analyze branching within the apical progenitors (APs), comparing the APs projected onto the glial branch and those projected onto the neuronal branch (Fig. 4c). Through differential analysis, we identify enrichment of radial glia markers (*Fabp7*, *Mt3*, *Dbi*, *Aldoc*, *Ptprz1*) within the APs situated along the glial branch. Conversely, genes such as *Neurog2* and *Hes6* have higher expression in APs projected onto the neural branch, suggesting a potential state of primed neurogenesis. These findings closely mirror the outcomes of the analysis conducted in the original Di Bella dataset paper.

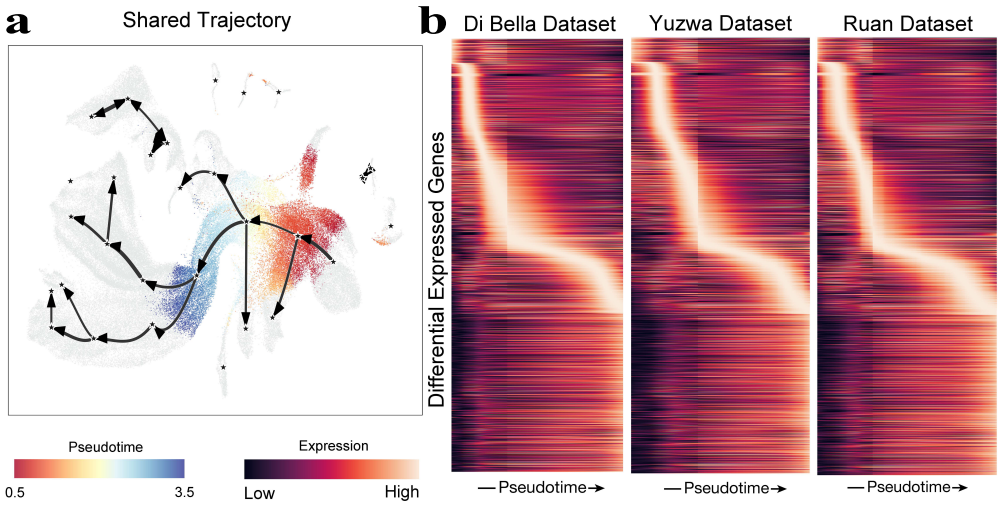

Figure 5: Testing for differentially expressed genes along a sub-trajectory on three mouse brain datasets. **(a)** Highlight of the sub-trajectory containing the apical progenitors, migrating neurons, and immature neurons with cells colored by pseudotime. **(b)** The heatmaps of DE gene expressions for the sub-trajectory along the inferred pseudotime ordering of the cells in each dataset. The DE genes are identified by performing quadratic regression of the gene expressions on the standardized pseudotime ordering with covariates adjusted (t-test, adjusted p-value< 0.05).

However, the convenience of our integrated VITAE framework significantly facilitates the execution of this analysis. Additionally, we are able to confirm that this divergence trend within the AP population holds consistent across all three datasets (Fig. 4d).

We also visualize the expressions of IPC marker *Eomes*, neurogenesis marker *Neurog2*, migration-associated gene *Neurod1*, and neuronal differentiation gene *Neurod6* along the estimated pseudotime (Fig. 4e). We observe that the expression of *Neurog2* precedes that of *Neurod1* and *Neurod6*, confirming its role as a proneural gene within the dorsal telencephalon [20]. Despite the presence of lab-specific batch effects, these genes consistently exhibit coherent patterns across all three datasets.

Finally, we focus on subtype analysis of the interneurons. With the joint analyses of all three datasets, VITAE is able to automatically identify four biologically different interneuron cell subtypes (Fig. 4f, Fig. S4). What's particularly noteworthy is that while three of these subtypes align with those previously identified through clustering in the original Di Bella paper that analyzes solely its own atlas data, VITAE reveals an additional subtype characterized by the expression of *Gfra2*. This subtype potentially represents the GABAergic interneuron population, as suggested by previous studies [23, 24]. Furthermore, VITAE reliably positions the small population of interneurons from the Ruan dataset within these four identified subtypes (Fig. 4g). Only a minority of interneurons from the Ruan dataset exhibit characteristics associated with the *Gfra2*+, *Sst*+, and *Meiz2*+ subtypes, as they predominantly emerge during the later stages of embryonic days (Fig. S5). Without the joint analysis facilitated by VITAE, it would be challenging to identify these interneurons solely from the Ruan dataset.

Differential analysis along estimated trajectories. VITAE is also designed to identify differentially expressed genes along any estimated sub-trajectory (Appendix). We focus on examining gene expression

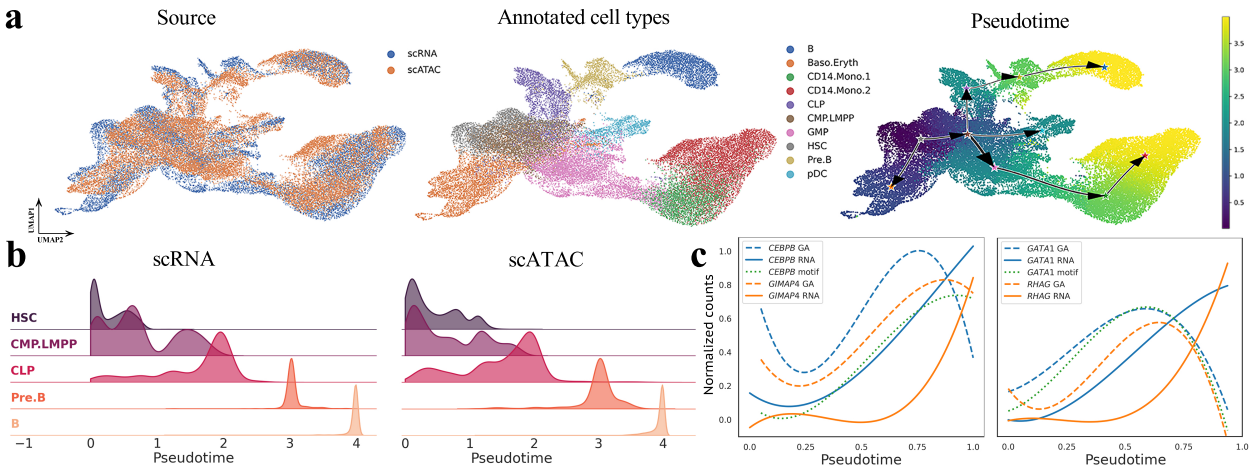

Figure 6: Integrative trajectory inference of multi-omic human hematopoiesis data. (a) UMAP visualization of VITAE's low-dimensional embedding of cells, colored by source, annotated cell types, and pseudotime. **(b)** Distribution of cells per cell type along the estimated pseudotime of the B cell lineage. **(c)** Lineage dynamics of the *CEBPB* and *GATA1* genes. The gene activity (GA) scores, TF (motif) scores, and gene expressions precede the gene expressions of their target genes, *GIMP4* and *RHAG* respectively, along the estimated pseudotime in each lineage.

dynamics along the sub-trajectory spanning RGC to IPC to migrating neurons (Fig. 5a). Despite that cells are from three distinct sources for this analysis, a consistent pattern emerges in the phase transitions of these differentially expressed genes across all three datasets (Fig. 5b). This confirms VITAE's ability to align cells originating from disparate datasets while preserving their inherent biological information.

Joint trajectory analysis on scATAC-seq and scRNA-seq datasets. We further apply VITAE to conduct a combined trajectory analysis of cells using both scRNA-seq and scATAC-seq data obtained from the same human hematopoiesis population of healthy donors [25]. The joint analysis of the two modalities can help to identify the cell types within the ATAC-seq data and infer active regulators for the RNA-seq data without requiring the two modalities to be measured on the same individual cells [26].

We use VITAE to simultaneously integrate the two modalities and infer a developmental trajectory for hematopoietic stem cells (HSC). VITAE successfully achieved a uniform mixing of cells from both modalities while preserving the distinction between different reference cell types (Fig. 6a, Fig. S6a). VITAE reveals the two major lineages (Fig. 6a), including the B cell lineage (*HSC-CMP.LMPP-CLP-Pre.B-B*) and the monocytic lineage (*HSC-CMP.LMPP-GMP-CD14.Mono.1-CD14.Mono.2*). We observed consistent patterns of continuous cell type transitions along the pseudotime in both modalities and within these two lineages (Fig. 6b, Fig. S6b).

By joint profiling of chromatin accessibility and gene expression within the same individual cells, earlier papers suggest that changes in chromatin accessibility may prime cells for lineage commitment [27]. By projecting cells onto the shared trajectory, VITAE effectively aligned cells with only one of the two omics, allowing us to visualize similar regulatory dynamics. For instance, we examined the gene *CEBPB*, a key transcriptional factor (TF) in monocytic development, and observed a clear progression of its gene activity score, gene expression level, and TF activity score, followed by the expression level of its target gene *GIMP4*.

Similar trends were also identified for the gene *GATA1*, a pivotal TF in the erythroid lineage, and its target gene *RHAG*. These findings underscore the power of VITAE in elucidating regulatory relationships within multi-omics data.

Discussion

In this paper, we propose VITAE, a hierarchical mixture model combined with VAE, for trajectory inference. Compared with existing TI methods, VITAE provides a coherent probabilistic framework to explicitly define estimable trajectories and pseudotime while accommodating diverse trajectory topologies and effectively accounting for confounding covariates. VITAE harnesses the power of variational autoencoders to approximate posterior quantities, facilitating the inference of trajectory backbones and cell positions from these approximations. Moreover, VITAE incorporates a Jacobian regularizer as a penalty to enhance the model's estimation robustness.

While numerous tools exist for integrating multiple scRNA-seq datasets, they often struggle when dealing with cells exhibiting continuous trajectory structures that do not neatly fit into distinct cell types. Our approach, which seamlessly combines trajectory analysis with data integration, excels at regularizing the latent space, resulting in a cleaner shared trajectory while preserving biologically meaningful distinctions in each dataset.

However, it's essential to acknowledge the limitations of our approach. First, the approximate variational inference employed to infer trajectories and quantify cell position uncertainties overlooks the estimation uncertainties inherent in our encoder and decoder. Consequently, while our uncertainty quantification offers valuable insights, it may not precisely reflect the true posterior uncertainties. Additionally, our current framework exclusively accommodates multi-omics data with identical input dimensions. Future work will focus on extending VITAE to effectively leverage multi-omics data with distinct features for joint trajectory inference.

Materials and Methods

Definition of the trajectory backbone and pseudotime. In VITAE, the trajectory backbone, denoted as $\mathcal{B}$, is defined as a subgraph of $\mathcal{G}$ that only includes edges with positive proportions of cells:

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

To determine the pseudotime of a cell, we first define the pseudotime for each vertex. Let each edge $\ell \in \mathcal{E}(\mathcal{B})$ be associated with a duration b_ℓ (with a default value of $b_\ell = 1$ for all edges). Given a root vertex k_0, the pseudotime for vertex j is defined as:

$$o_j = \begin{cases} \min \left\{ \sum_{\ell \in \mathcal{L}} b_\ell : \mathcal{L} \subset \mathcal{E}(\mathcal{B}) \text{ forms a path from } k_0 \text{ to } j \right\} \\ \infty \quad \text{if no such } \mathcal{L} \text{ exists.} \end{cases} \tag{6}$$

Then for a specific cell i with its position $\tilde{\boldsymbol{w}}_i$ on the graph, its pseudotime is defined as:

$$T_i = \boldsymbol{o}^\top \tilde{\boldsymbol{w}}_i \tag{7}$$

Intuitively, T_i equals the pseudotime of the vertex the cell is on or the weighted average of the pseudotime of the two vertices of the edge to which the cell belongs.

Hierarchical model for the distributions of gene expressions. In equation (1), we model the observed counts, such as those from scRNA-seq with Unique Molecular Identifiers (UMI), using a Negative Binomial (NB) distribution with gene-specific dispersion parameters θ_g. This choice is suitable for capturing the stochasticity in scRNA-seq data, accounting for both biological and technical noise [28]. However, for certain cases where the NB distribution may not be adequate, such as scRNA-seq without UMI or single-cell ATAC sequencing data with numerous excessive zeros, we adopt the assumption that the observed counts follow zero-inflated Negative Binomial (ZINB) distributions, as proposed by [29]:

$$Y_{ig}|\boldsymbol{Z}_i, \boldsymbol{X}_i \overset{\text{ind}}{\sim} \phi_{ig}\delta_0 + (1-\phi_{ig})\mathcal{NB}\left(l_i f_g(\boldsymbol{Z}_i, \boldsymbol{X}_i), \theta_g\right)$$

where $\phi_{ig} = h_g(\boldsymbol{Z}_i, \boldsymbol{X}_i)$ is the zero inflation probability with $h_g(\cdot)$ being unknown non-linear functions.

To facilitate model estimation, we introduce a hierarchical prior on the cell positions $\tilde{\boldsymbol{w}}_i$. Let the latent category variable c_i take values from the set $1, 2, \ldots, K$, where $K = k(k+1)/2$ represents the total count of potential edges and vertices within the graph $\mathcal{G}$. Then $c_i \sim \text{Multinomial}(1, \boldsymbol{\pi})$ serves as an indicator of the chosen edge or vertex by the cell, with $\boldsymbol{\pi}$ being empirically estimated during the model training step. Should the cell be positioned on an edge, we assume that its relative placement along the edge is governed by a uniform prior: $w_i \sim \text{Uniform}(0, 1)$. For more comprehensive mathematical elaboration, please refer to the supplementary information in the Appendix.

To reduce the computational cost, we also present an accelerated Gaussian version of VITAE. In this version, the observed counts of the G genes are replaced with the top R ($R = 64$ by default) principle component (PC) scores $\boldsymbol{F}_i$. The distribution of $\boldsymbol{F}_i$ is then approximated by Gaussian distributions. This modification allows VITAE to scale effectively and achieve a computational efficiency comparable to other trajectory inference methods.

Penalties in the loss function As demonstrated in the main text, VITAE's loss function comprises four penalty terms [2]-[5]. While the marginal density function $p(\boldsymbol{Z}_i)$ in [2] involves a complicated hierarchical mixture model, it possesses a tractable closed-form representation (Appendix), facilitating straightforward evaluation. By default, we assign $\alpha = 0.1$ to the soft penalty [3], and incorporate the MMD loss [4] solely in scenarios involving technical replicates or when the soft penalty [3] is insufficient and a stronger penalty is necessary.

In particular, considering two cell groups $\mathcal{P}_1$ and $\mathcal{P}_2$, each comprising n_1 and n_2 cells, respectively. To eliminate all differences between the two groups, the MMD loss is characterized in the latent space by the

following formulation:

$$L_{\text{MMD}}(\mathcal{P}_1, \mathcal{P}_2) = -\sum_{j=1}^{2} \frac{1}{n_j^2} \sum_{i_1, i_2 \in \mathcal{P}_j} k(\boldsymbol{Z}_{i_1}, \boldsymbol{Z}_{i_2}) + \frac{2}{n_1 n_2} \sum_{i_1 \in \mathcal{P}_1, i_2 \in \mathcal{P}_2} k(\boldsymbol{Z}_{i_1}, \boldsymbol{Z}_{i_2}).$$

where $k(\cdot, \cdot)$ is the multi-scale RBF kernel proposed by [30] (see Appendix). Consequently, the MMD penalty [4] is outlined as the cumulative MMD loss across all such pairs of cell groups:

$$\Omega_{\text{MMD}}(\mathcal{D}_N) = \sum_{(s,t) \in \text{all pairs}} L_{\text{MMD}}(\mathcal{P}_s, \mathcal{P}_t).$$

Lastly, the Jacobian regularizer [5] is defined to control the Frobenius norm of the Jacobians of latent variables $\boldsymbol{Z}_i$ with respect to the input $\boldsymbol{Y}_i$. Specifically, the penalty term [5] is defined as:

$$\Omega_{\text{Jacobian}}(\mathcal{D}_N) = \sum_{i=1}^{N} \sum_{j=1}^{d} \sum_{g=1}^{G} \mathbb{E}_{q(\boldsymbol{Z}_i | \boldsymbol{Y}_i, \boldsymbol{X}_i)} \left[\left(\frac{\partial \boldsymbol{Z}_{ij}}{\partial \boldsymbol{Y}_{ig}} \right)^2 \right].$$

The default value for the tuning parameter γ is set to 1.

Model training, estimation, and trajectory inference. We train VITAE with only highly variable genes and use normalized, log-transformed, and scaled gene expressions as input for the model. The initial parameters of VITAE are set by pretraining with $\beta = 0$. When cell labels are unavailable, the number of states k is determined by clustering the pretrained latent space. To estimate VITAE's model parameters, we employ the mini-batch stochastic gradient descent algorithm, ensuring memory usage remains independent of cell count.

Using the estimated model parameters, we approximate posterior distributions for w_i and c_i by sampling from the estimated encoder output $\widehat{q}(\boldsymbol{Z}_i | \boldsymbol{Y}_i, \boldsymbol{X}_i)$. Leveraging these posterior approximations, we infer the trajectory backbone $\widehat{\mathcal{B}}$ under the assumption that $\mathcal{B}$ predominantly comprises sparse edges. Specifically, let $C_{j_1 j_2}$ denote the edge connecting states j_1 and j_2. Its edge score is defined as follows:

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

where $\tilde{c}_i = \arg\max_{c \in \{1,2,\cdots,K\}} \widehat{p}(c_i = c | \boldsymbol{Y}_i, \boldsymbol{X}_i)$ and $\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}$ is the approximate posterior mean of $\tilde{\boldsymbol{w}}_i$. An edge is included in $\widehat{\mathcal{B}}$ when $\tilde{s}_{j_1 j_2} \geq s_0$ (with $s_0 = 0.01$ as the default threshold). Subsequently, cell positions are projected onto $\widehat{\mathcal{B}}$ based on the aforementioned posterior approximations, which also provide uncertainty estimates for these projections.

Given the inferred trajectory $\widehat{\mathcal{B}}$, the user can manually assign a root vertex. We also provide an automatic root selection step following Tempora [31] when cells are gathered across multiple time points. With a chosen root vertex, the pseudotime of other vertices and all cells can be readily derived by substituting estimates into

the definitions [6]-[7]. For a comprehensive understanding of trajectory inference procedures from posterior approximations, refer to Section 1.2 of Appendix.

Benchmarking methods. For systematic benchmarking with real and synthetic datasets, we run PAGA, Monocle 3, and Slingshot via the Dyno platform [5], which converts these TI methods' outputs into an estimated trajectory backbone $\widehat{\mathcal{B}}$, estimated cell positions $\widehat{\tilde{\boldsymbol{w}}}_i$'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 [32] coordinates of cells computed following Seurat's [21] default steps are used.

For experiments on mouse brain datasets [19, 20], we compare the trajectory inference result with Slingshot [7]. To apply Slingshot, Seurat's CCA [21] is first used to integrate both datasets and regress out cell-cycle scores. Then Slingshot is applied to Seurat's results to infer the trajectories.

For experiments on the integration of three mouse brain datasets [19, 20, 22], we compare the results with Slingshot and Monocle3 [33]. For the latter, the batch effects from the different datasets are adjusted by the function `align_cds`, and then the trajectories are learned afterward.

Data, Materials, and Software Availability

- Developing mouse brain datasets. Yuzwa dataset [19] and Ruan dataset [20] are available from the GEO database under accession codes GSE107122 and GSE161690, respectively. For Yuzwa's dataset, we only use cortically derived cells selected by the original paper. We keep the genes that are measured in both datasets (14,707 genes) and merge all 16,651 cells (6,390 and 10,261 cells for each). The cell cycle scores (`S.Score` and `G2M.Score`) of the two datasets are calculated separately by `Seurat` v3. As cell type labels are not provided by the Yuzwa dataset, we perform clustering using Seurat and annotate cell types using the marker genes in the Ruan dataset. The information on the collection days of both datasets is summarized in Table S2.
- Mouse cerebral cortex atlas. Di Bella's preprocessed dataset [22] following the procedure in [22] are available in the `https://singlecell.broadinstitute.org/single_cell/study/SCP1290/molecular-logic-of-cellular-` The cell cycle scores (`S.Score` and `G2M.Score`), cell type annotations, and collection day information are provided in the original paper. We exclude cells labeled as 'Doublet' and 'Low quality cells' and retain 91,648 cells and 19,712 genes with 24 different cell type annotations.
- Integration of the three mouse brain datasets. The previously mentioned merged developing `mouse brain` dataset and the `atlas` dataset are preprocessed separately by the standard procedure (`normalize_total`, `log1p`, `highly_variable_genes`, `scale`) using `scanpy` v1.8.3 [34] with default parameters and exclude cells labeled as 'Doublet' and 'Low quality.' Then `mouse brain` and `atlas` datasets are merged by taking the union of highly variable genes in each dataset. Then we unified different cell type annotations that refer to the same cell type (for example, 'Intermediate progenitors' as 'IPC'). Finally, there are 108,299 cells and 13,183 genes. Empirically, we found that SCPN cells before and after day E16 are always projected to two different latent space regions. So we relabeled the SCPN cells before E16 as SCPN1 to give a more biologically meaningful center when initializing latent space.

- Human hematopoiesis data. We obtain the gene expression, peak matrix, and cell types annotation of human hematopoiesis data of healthy donors from https://github.com/GreenleafLab/MPAL-Single-Cell-2019[25]. The TF activity scores from [25] were computed using chromVAR [35]. Following the preprocessing procedure as in [36], we excluded cells labeled as 'Unknown' and combined the clusters with the same cell-type annotation into one label (for example, 'CLP.1' and 'CLP.2' as 'CLP'). We retain only cell types that are on the two developmental trajectories analyzed in [36], and the 'cDC' and 'CD16.Mono' cell types are removed because of the inadequate number of cells. We calculate the highly variable genes for each dataset by `scanpy` and retain the union of these genes. The raw count matrices containing the selected cells and genes are then concatenated accordingly. Finally, this results in 19,309 cells for scRNA-seq data and 22,685 cells for gene activities of the scATAC-seq data for the analysis, with 4,094 genes measured.

After filtering genes and cells, we apply the standard preprocessing procedure by `scanpy` v1.8.3 [34] before supplying the datasets to our model.

The Python package of VITAE is publicly available at https://github.com/jaydu1/VITAE with MIT license. Python and R scripts for reproducing all results in this paper are also provided in the same repository.

References

[1] Amos Tanay and Aviv Regev. Scaling single-cell genomics from phenomenology to mechanism. *Nature*, 541(7637):331–338, 2017.

[2] Volker Hovestadt, Kyle S Smith, Laure Bihannic, Mariella G Filbin, McKenzie L Shaw, Alicia Baumgartner, John C DeWitt, Andrew Groves, Lisa Mayr, Hannah R Weisman, Alyssa R Richman, Marni E Shore, Goumnerova Liliana, Celeste Rosencrance, Robert A Carter, Timothy N Phoenix, Jennifer L Hadley, Yiai Tong, Jim Houston, Richard A Ashmun, Michael DeCuypere, Tanvi Sharma, Diane Flasch, Antonina Silkov, Keith L Ligon, Scott L Pomeroy, Miguel Rivera, Orit Rozenblatt-Rosen, Jessica M Rusert, Robert J Wechsler-Reya, Li Xiao-Nan, Andreas Peyrl, Johannes Gojo, Dominik Kirchhofer, Daniela Lötsch, Thomas Czech, Christian Dorfer, Christine Haberler, Rene Geyeregger, Angela Halfmann, Charles Gawad, John Easton, Stefan M Pfister, Aviv Regev, Amar Gajjar, Brent A Orr, Irene Slavc, Giles W Robinson, Bradley E Bernstein, Mario L Suvà, and Paul A Northcott. Resolving medulloblastoma cellular architecture by single-cell genomics. *Nature*, 572(7767):74–79, 2019.

[3] David Lähnemann, Johannes Köster, Ewa Szczurek, Davis J McCarthy, Stephanie C Hicks, Mark D Robinson, Catalina A Vallejos, Kieran R Campbell, Niko Beerenwinkel, Ahmed Mahfouz, Luca Pinello, Pavel Skums, Alexandros Stamatakis, Camille S-O Attolini, Samuel Aparicio, Jasmijn Baaijens, Marleen Balvert, Buys de Barbanson, Antonio Cappuccio, Giacomo Corleone, Bas E. Dutilh, Maria Florescu, Victor Guryev, Rens Holmer, Katharina Jahn, Thamar J Lobo, Emma M. Keizer, Indu Khatri, Szymon M. Kielbasa, Jan O. Korbel, Alexey M. Kozlov, Tzu-Hao Kuo, Boudewijn P.F. Lelieveldt, Mandoiu Ion I., John C. Marioni, Tobias Marschall, Felix Mölder, Amir Niknejad, Lukasz Raczkowski, Marcel

Reinders, Jeroen de Ridder, Antoine-Emmanuel Saliba, Antonios Somarakis, Oliver Stegle, Fabian J. Theis, Huan Yang, Alex Zelikovsky, Alice C. McHardy, Benjamin J. Raphael, Sohrab P. Shah, and Alexander Schönhuth. Eleven grand challenges in single-cell data science. *Genome biology*, 21(1):1–35, 2020.

[4] Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J Lennon, Kenneth J Livak, Tarjei S Mikkelsen, and John L Rinn. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature biotechnology*, 32(4):381, 2014.

[5] Wouter Saelens, Robrecht Cannoodt, Helena Todorov, and Yvan Saeys. A comparison of single-cell trajectory inference methods. *Nature biotechnology*, 37(5):547–554, 2019.

[6] Xiaojie Qiu, Qi Mao, Ying Tang, Li Wang, Raghav Chawla, Hannah A Pliner, and Cole Trapnell. Reversed graph embedding resolves complex single-cell trajectories. *Nature methods*, 14(10):979, 2017.

[7] Kelly Street, Davide Risso, Russell B Fletcher, Diya Das, John Ngai, Nir Yosef, Elizabeth Purdom, and Sandrine Dudoit. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics*, 19(1):477, 2018.

[8] F Alexander Wolf, Fiona K Hamey, Mireya Plass, Jordi Solana, Joakim S Dahlin, Berthold Göttgens, Nikolaus Rajewsky, Lukas Simon, and Fabian J Theis. PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *Genome biology*, 20(1):1–9, 2019.

[9] Caleb Weinreb, Samuel Wolock, Betsabeh K Tusi, Merav Socolovsky, and Allon M Klein. Fundamental limits on dynamic inference from single-cell snapshots. *Proceedings of the National Academy of Sciences*, 115(10):E2467–E2476, 2018.

[10] Sophie Tritschler, Maren Büttner, David S Fischer, Marius Lange, Volker Bergen, Heiko Lickert, and Fabian J Theis. Concepts and limitations for learning developmental trajectories from single cell genomics. *Development*, 146(12):dev170506, 2019.

[11] Valentine Svensson, Eduardo da Veiga Beltrame, and Lior Pachter. A curated database reveals trends in single-cell transcriptomics. *Database*, 2020, 2020.

[12] Mo Huang, Zhaojun Zhang, and Nancy R Zhang. Dimension reduction and denoising of single-cell RNA sequencing data in the presence of observed confounding variables. *bioRxiv*, 2020.

[13] Mohammad Lotfollahi, Mohsen Naghipourfar, Fabian J Theis, and F Alexander Wolf. Conditional out-of-sample generation for unpaired data using trvae. *arXiv preprint arXiv:1910.01791*, 2019.

[14] Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529, 2018.

[15] Junyue Cao, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M Ibrahim, Andrew J Hill, Fan Zhang, Stefan Mundlos, Lena Christiansen, Frank J Steemers, Cole Trapnell, and Jay Shendure. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745):496–502, 2019.

[16] Daniel H Geschwind and Pasko Rakic. Cortical evolution: judge the brain by its cover. *Neuron*, 80(3):633–647, 2013.

[17] Magdalena Götz and Wieland B Huttner. The cell biology of neurogenesis. *Nature reviews Molecular cell biology*, 6(10):777–788, 2005.

[18] Luciano Custo Greig, Mollie B Woodworth, Maria J Galazo, Hari Padmanabhan, and Jeffrey D Macklis. Molecular logic of neocortical projection neuron specification, development and diversity. *Nature Reviews Neuroscience*, 14(11):755–769, 2013.

[19] Scott A Yuzwa, Michael J Borrett, Brendan T Innes, Anastassia Voronova, Troy Ketela, David R Kaplan, Gary D Bader, and Freda D Miller. Developmental emergence of adult neural stem cells as revealed by single-cell transcriptional profiling. *Cell reports*, 21(13):3970–3986, 2017.

[20] Xiangbin Ruan, Bowei Kang, Cai Qi, Wenhe Lin, Jingshu Wang, and Xiaochang Zhang. Progenitor cell diversity in the developing mouse neocortex. *Proceedings of the National Academy of Sciences*, 118(10):e2018866118, 2021.

[21] Tim Stuart, Andrew Butler, Paul Hoffman, Christoph Hafemeister, Efthymia Papalexi, William M Mauck III, Yuhan Hao, Marlon Stoeckius, Peter Smibert, and Rahul Satija. Comprehensive integration of single-cell data. *Cell*, 177(7):1888–1902, 2019.

[22] Daniela J Di Bella, Ehsan Habibi, Robert R Stickels, Gabriele Scalia, Juliana Brown, Payman Yadollahpour, Sung Min Yang, Catherine Abbate, Tommaso Biancalani, Evan Z Macosko, et al. Molecular logic of cellular diversification in the mouse cerebral cortex. *Nature*, 595(7868):554–559, 2021.

[23] Ana B Muñoz-Manchado, Carolina Bengtsson Gonzales, Amit Zeisel, Hermany Munguba, Bo Bekkouche, Nathan G Skene, Peter Lönnerberg, Jesper Ryge, Kenneth D Harris, Sten Linnarsson, et al. Diversity of interneurons in the dorsal striatum revealed by single-cell rna sequencing and patchseq. *Cell reports*, 24(8):2179–2190, 2018.

[24] Michael S Fleming, Anna Vysochan, Sónia Paixao, Jingwen Niu, Rüdiger Klein, Joseph M Savitt, and Wenqin Luo. Cis and trans ret signaling control the survival and central projection growth of rapidly adapting mechanoreceptors. *Elife*, 4:e06828, 2015.

[25] Jeffrey M Granja, Sandy Klemm, Lisa M McGinnis, Arwa S Kathiria, Anja Mezger, M Ryan Corces, Benjamin Parks, Eric Gars, Michaela Liedtke, Grace XY Zheng, et al. Single-cell multiomic analysis identifies regulatory programs in mixed-phenotype acute leukemia. *Nature biotechnology*, 37(12):1458–1465, 2019.

[26] Jun Ding, Nadav Sharon, and Ziv Bar-Joseph. Temporal modelling using single-cell transcriptomics. *Nature Reviews Genetics*, 23(6):355–368, 2022.

[27] Sai Ma, Bing Zhang, Lindsay M LaFave, Andrew S Earl, Zachary Chiang, Yan Hu, Jiarui Ding, Alison Brack, Vinay K Kartha, Tristan Tay, et al. Chromatin potential identified by shared single-cell profiling of rna and chromatin. *Cell*, 183(4):1103–1116, 2020.

[28] Abhishek Sarkar and Matthew Stephens. Separating measurement and expression models clarifies confusion in single-cell rna sequencing analysis. *Nature genetics*, 53(6):770–777, 2021.

[29] Gökcen Eraslan, Lukas M Simon, Maria Mircea, Nikola S Mueller, and Fabian J Theis. Single-cell RNA-seq denoising using a deep count autoencoder. *Nature communications*, 10(1):1–14, 2019.

[30] Mohammad Lotfollahi, Mohsen Naghipourfar, Fabian J Theis, and F Alexander Wolf. Conditional out-of-distribution generation for unpaired data using transfer vae. *Bioinformatics*, 36(Supplement_2):i610–i617, 2020.

[31] Thinh N Tran and Gary D Bader. Tempora: Cell trajectory inference using time-series single-cell RNA sequencing data. *PLoS computational biology*, 16(9):e1008205, 2020.

[32] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3(29):861, 2018.

[33] Junyue Cao, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M Ibrahim, Andrew J Hill, Fan Zhang, Stefan Mundlos, Lena Christiansen, Frank J Steemers, et al. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745):496–502, 2019.

[34] F Alexander Wolf, Philipp Angerer, and Fabian J Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19(1):1–5, 2018.

[35] Alicia N Schep, Beijing Wu, Jason D Buenrostro, and William J Greenleaf. chromvar: inferring transcription-factor-associated accessibility from single-cell epigenomic data. *Nature methods*, 14(10):975–978, 2017.

[36] Yingxin Lin, Tung-Yu Wu, Sheng Wan, Jean YH Yang, Wing H Wong, and YX Wang. scjoint integrates atlas-scale single-cell rna-seq and atac-seq data with transfer learning. *Nature Biotechnology*, 40(5):703–710, 2022.

[37] Kieran R Campbell and Christopher Yau. Order under uncertainty: robust differential expression analysis using probabilistic models for pseudotime inference. *PLoS computational biology*, 12(11):e1005212, 2016.

[38] Sumon Ahmed, Magnus Rattray, and Alexis Boukouvalas. GrandPrix: scaling up the Bayesian GPLVM for single-cell data. *Bioinformatics*, 35(1):47–54, 2019.

[39] Eugenio Marco, Robert L Karp, Guoji Guo, Paul Robson, Adam H Hart, Lorenzo Trippa, and Guo-Cheng Yuan. Bifurcation analysis of single-cell gene expression data reveals epigenetic landscape. *Proceedings of the National Academy of Sciences*, 111(52):E5643–E5650, 2014.

[40] Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, et al. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943, 2019.

[41] Chieh Lin and Ziv Bar-Joseph. Continuous-state hmms for modeling time-series single-cell rna-seq data. *Bioinformatics*, 35(22):4707–4715, 2019.

[42] Gioele La Manno, Ruslan Soldatov, Amit Zeisel, Emelie Braun, Hannah Hochgerner, Viktor Petukhov, Katja Lidschreiber, Maria E Kastriti, Peter Lönnerberg, Alessandro Furlan, et al. RNA velocity of single cells. *Nature*, 560(7719):494–498, 2018.

[43] Volker Bergen, Marius Lange, Stefan Peidli, F Alexander Wolf, and Fabian J Theis. Generalizing rna velocity to transient cell states through dynamical modeling. *Nature biotechnology*, pages 1–7, 2020.

[44] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations*, 2014.

[45] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[46] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1965–1972, 2017.

[47] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

[48] Christopher Heje Grønbech, Maximillian Fornitz Vording, Pascal N Timshel, Casper Kaae Sønderby, Tune H Pers, and Ole Winther. scVAE: Variational auto-encoders for single-cell gene expression data. *Bioinformatics*, 36(16):4415–4422, 2020.

[49] Xiangjie Li, Kui Wang, Yafei Lyu, Huize Pan, Jingxiao Zhang, Dwight Stambolian, Katalin Susztak, Muredach P Reilly, Gang Hu, and Mingyao Li. Deep learning enables accurate clustering with batch effect removal in single-cell RNA-seq analysis. *Nature communications*, 11(1):1–14, 2020.

[50] Jingshu Wang, Divyansh Agarwal, Mo Huang, Gang Hu, Zilu Zhou, Chengzhong Ye, and Nancy R Zhang. Data denoising with transfer learning in single-cell transcriptomics. *Nature methods*, 16(9):875–878, 2019.

[51] Divyansh Agarwal, Jingshu Wang, and Nancy R Zhang. Data denoising and post-denoising corrections in single cell RNA sequencing. *Statistical Science*, 35(1):112–128, 2020.

[52] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

[53] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[54] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[55] Andreas Stuhlmüller, Jacob Taylor, and Noah Goodman. Learning stochastic inverses. *Advances in neural information processing systems*, 26:3048–3056, 2013.

[56] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[57] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[58] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

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

[60] Scott Van Buren, Hirak Sarkar, Avi Srivastava, Naim U Rashid, Rob Patro, and Michael I Love. Compression of quantification uncertainty for scRNA-seq counts. *BioRxiv*, 2020.

[61] Jingshu Wang, Qingyuan Zhao, Trevor Hastie, and Art B Owen. Confounder adjustment in multiple hypothesis testing. *Annals of statistics*, 45(5):1863, 2017.

[62] Matthew Amodio, David Van Dijk, Krishnan Srinivasan, William S Chen, Hussein Mohsen, Kevin R Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha Venkataswamy, et al. Exploring single-cell data with deep multitasking neural networks. *Nature methods*, 16(11):1139–1145, 2019.

[63] Mohammad Lotfollahi, Mohsen Naghipourfar, Malte D Luecken, Matin Khajavi, Maren Büttner, Marco Wagenstetter, Žiga Avsec, Adam Gayoso, Nir Yosef, Marta Interlandi, et al. Mapping single-cell data to reference atlases by transfer learning. *Nature Biotechnology*, 40(1):121–130, 2022.

[64] Monika S Kowalczyk, Itay Tirosh, Dirk Heckl, Tata Nageswara Rao, Atray Dixit, Brian J Haas, Rebekka K Schneider, Amy J Wagers, Benjamin L Ebert, and Aviv Regev. Single-cell RNA-seq reveals changes in cell cycle and differentiation programs upon aging of hematopoietic stem cells. *Genome research*, 25(12):1860–1872, 2015.

[65] Sophie Petropoulos, Daniel Edsgärd, Björn Reinius, Qiaolin Deng, Sarita Pauliina Panula, Simone Codeluppi, Alvaro Plaza Reyes, Sten Linnarsson, Rickard Sandberg, and Fredrik Lanner. Single-cell RNA-seq reveals lineage and X chromosome dynamics in human preimplantation embryos. *Cell*, 165(4):1012–1026, 2016.

[66] Fan Guo, Liying Yan, Hongshan Guo, Lin Li, Boqiang Hu, Yangyu Zhao, Jun Yong, Yuqiong Hu, Xiaoye Wang, Yuan Wei, Wei Wang, Rong Li, Jie Yan, Xu Zhi, Yan Zhang, Hongyan Jin, Wenxin Zhang, Yu Hou, Ping Zhu, Jingyun Li, Ling Zhang, Sirui Liu, Yixin Ren, Xiaohui Zhu, Lu Wen, Yi Q. Gao, Fuchou Tang, and Jie Qiao. The transcriptome and DNA methylome landscapes of human primordial germ cells. *Cell*, 161(6):1437–1452, 2015.

[67] Kyle M Loh, Angela Chen, Pang Wei Koh, Tianda Z Deng, Rahul Sinha, Jonathan M Tsai, Amira A Barkal, Kimberle Y Shen, Rajan Jain, Rachel M Morganti, Ng Shyh-Chang, Nathaniel B Fernhoff, Benson M George, Gerlinde Wernig, Rachel E A Salomon, Zhenghao Chen, Hannes Vogel, Jonathan A Epstein, Anshul Kundaje, William S Talbot, Philip A Beachy, Ang Lay T, and Irving L Weissman. Mapping the pairwise choices leading from pluripotency to human bone, heart, and other mesoderm cell types. *Cell*, 166(2):451–467, 2016.

[68] Hannah Hochgerner, Amit Zeisel, Peter Lönnerberg, and Sten Linnarsson. Conserved properties of dentate gyrus neurogenesis across postnatal development revealed by single-cell rna sequencing. *Nature neuroscience*, 21(2):290–299, 2018.

[69] Barbara Treutlein, Qian Yi Lee, J Gray Camp, Moritz Mall, Winston Koh, Seyed Ali Mohammad Shariati, Sopheak Sim, Norma F Neff, Jan M Skotheim, Marius Wernig, and Stephen R Quake. Dissecting direct reprogramming from fibroblast to neuron using single-cell RNA-seq. *Nature*, 534(7607):391–395, 2016.

[70] Grace XY Zheng, Jessica M Terry, Phillip Belgrader, Paul Ryvkin, Zachary W Bent, Ryan Wilson, Solongo B Ziraldo, Tobias D Wheeler, Geoff P McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga,

Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature communications*, 8(1):1–12, 2017.

[71] Robrecht Cannoodt, Wouter Saelens, Louise Deconinck, and Yvan Saeys. dyngen: a multi-modal simulator for spearheading new single-cell omics analyses. *BioRxiv*, 2020.

[72] Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. An exact graph edit distance algorithm for solving pattern recognition problems. In *4th International Conference on Pattern Recognition Applications and Methods*, 2015.

[73] Giuseppe Jurman, Roberto Visintainer, Michele Filosi, Samantha Riccadonna, and Cesare Furlanello. The HIM glocal metric and kernel for network comparison and classification. In *2015 IEEE International Conference on Data Science and Advanced Analytics*, pages 1–10. IEEE, 2015.

[74] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Appendix

Organization

Below, we outline the structure of the rest of the supplement. Appendix S.0 reviews existing TI methods. Appendix S.1 introduces the proposed trajectory inference method, while the specifications of certain algorithm details are deferred to Appendix S.2. The details of simulation datasets, evaluation metrics, and experiment procedures are presented in Appendix S.3. The details of case studies on mouse brain datasets are provided in Appendix S.4. Lastly, the extra experimental results on real data are included in Appendix S.5.

S.0 Review of Existing TI Methods

Since the first TI method for scRNA-seq was proposed by [4], many have been developed in the past few years. Recently, [5] 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 trajectory topology, and some may not be scalable to handle large datasets. Among the 45 TI methods, PAGA [8], Slingshot [7], and the Monocle series [4, 6, 15] 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 construct the connectivity graph. For example, Monocle 1 [4] uses independent component analysis for dimension reduction, then infers the trajectory by building a minimum spanning tree (MST) directly on the cells. Monocle 2 [6] 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 starts with a given dimension reduction and cell clustering; however, it finds an MST on the cluster centers and simultaneously fits principal curves to infer each cell's pseudotime. PAGA 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 [15] 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 [37, 38] 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 [39–41] makes 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 [40] infers developmental trajectories by temporal couplings of the cells with the optimal transport, and CSHMM [41] 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 [42, 43]. 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 estimation with the powerful variational autoencoders (VAE), which is computationally 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 [44–47]. For scRNA-seq, deep generative models are also widely applied for various purposes, such as cell clustering [48, 49], batch correction [12], data denoising [29, 50] and many other tasks.

S.1 Methods

S.1.1 Model and Estimation

A hierarchical mixture model for the trajectory structure Inspired by the common trajectory model proposed in [5], 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 $\mathcal{E}(\mathcal{G})$. 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 an edge. 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 $w_i \in [0,1]$ describes the relative position of cell i if it is on an edge.

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 that results in the trajectory structure [4].

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

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

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

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

which equals the pseudotime of the vertex that the cell is on or the weighted average of the pseudotime of the two vertices of the edge that the cell belongs to.

Now, we relate the observed single-cell sequencing data with the underlying trajectory backbone graph $\mathcal{B}$. Let $\boldsymbol{Y}_i = (y_{i1}, \cdots, y_{iG})$ be the observed counts of G features in cell i. Though $\boldsymbol{Y}_i$ is the vector of observed counts with complicated dependence across features, 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 $\boldsymbol{Y}_i$. Specifically, we assume the following latent variable model on the observed counts:

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

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 feature-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 NB distributions to model scRNA-seq data is based on a detailed review and discussion in [51]. Specifically, for scRNA-seq with the unique molecular identifier (UMI) counts, NB distributions can describe the stochasticity in scRNA-seq, accounting for both biological and technical noise. However, they may not be adequate for scRNA-seq data without UMI. In that scenario, we assume that the observed counts follow zero-inflated Negative Binomial (ZINB) distributions as in [29]:

$$\text{[for non-UMI counts]} \qquad Y_{ig}|\boldsymbol{Z}_i, \boldsymbol{X}_i \overset{\text{ind}}{\sim} \phi_{ig}\delta_0 + (1-\phi_{ig})\mathcal{NB}\left(l_i f_g(\boldsymbol{Z}_i, \boldsymbol{X}_i), \theta_g\right) \tag{E.2}$$

where $\phi_{ig} = h_g(\boldsymbol{Z}_i, \boldsymbol{X}_i)$ is the zero inflation probability with $h_g(\cdot)$ an unknown non-linear function for $g = 1, \cdots, G$.

The above model defines the trajectory backbone $\mathcal{B}$, cell positions $\tilde{\boldsymbol{w}}_i$, and pseudotime T_i that are identifiable from scRNA-seq data. However, estimating these quantities is 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 $\mathcal{G}$. For the complete graph $\mathcal{G}$, define the categorical assignment symmetric matrix,

$$\boldsymbol{C} = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ k \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \cdots & k \end{array} \\ \begin{pmatrix} 1 & 2 & \cdots & k \\ 2 & k+1 & \cdots & 2k-1 \\ \vdots & \vdots & \ddots & \vdots \\ k & 2k-1 & \cdots & K \end{pmatrix} \end{array}. \tag{E.3}$$

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. We let $C_{j_2 j_1} = C_{j_1 j_2} = j_1 + (j_2 - 1)k$ for $j_1 \leq j_2$ and set $c_i = C_{j_1 j_2}$ if cell i is at the edge between vertex j_1 and vertex j_2, or if cell i is at the vertex j_1 when $j_2 = j_1$. Then we assume the following mixture prior on $\tilde{\boldsymbol{w}}_i$:

$$\begin{aligned} w_i &\overset{\text{i.i.d.}}{\sim} \text{Uniform}(0,1) \\ c_i &\overset{\text{i.i.d.}}{\sim} \text{Multinomial}(1, \boldsymbol{\pi}), \quad \boldsymbol{\pi} \in [0,1]^K \\ w_i &\perp\!\!\!\perp c_i \\ \tilde{\boldsymbol{w}}_i &= w_i \boldsymbol{a}_{c_i} + (1 - w_i)\boldsymbol{b}_{c_i} \end{aligned} \tag{E.4}$$

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 (E.1) and (E.4), 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).

VAE 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) [44] for approximating the posterior distributions. Following the variational Bayes approach in VAE and the conditional VAE [45], we use Gaussian distributions to approximate the intractable posterior distribution of $\boldsymbol{Z}_i$ given the observed data $\boldsymbol{Y}_i$ and confounding variables $\boldsymbol{X}_i$:

$$\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i \dot{\sim} \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{Y}_i,\boldsymbol{X}_i}, \boldsymbol{\Sigma}_{\boldsymbol{Y}_i,\boldsymbol{X}_i}\right). \tag{E.5}$$

Here the "posterior" mean $\boldsymbol{\mu}_{\boldsymbol{Y}_i,\boldsymbol{X}_i}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{Y}_i,\boldsymbol{X}_i} = \mathrm{diag}(\sigma^2_{\boldsymbol{Y}_i,\boldsymbol{X}_i,1}, \cdots, \sigma^2_{\boldsymbol{Y}_i,\boldsymbol{X}_i,d})$ are functions of $(\boldsymbol{Y}_i, \boldsymbol{X}_i)$. To guarantee flexibility, they are also modeled by a neural network. 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 [29], the functions $h_g(\cdot)$ for zero-inflation parameters are also modeled in the decoder and share the same hidden layers as $f_g(\cdot)$.

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

$$\begin{aligned}\log p(\boldsymbol{Y}_i|\boldsymbol{X}_i) &= \log\left[\mathbb{E}_{p(\boldsymbol{Z}_i|\boldsymbol{X}_i)} p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i)\right] \\ &\geq \mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i) - D_{\mathrm{KL}}(q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)\|p(\boldsymbol{Z}_i)),\end{aligned} \tag{E.6}$$

where $p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i)$ is the true conditional distribution in model (E.1) or (E.2), and $q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)$ is the Gaussian approximation of the posterior distribution in (E.5). This lower bound is often referred to as the evidence lower bound (ELBO) [44], 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 (E.6) 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 (E.6),

$$\mathcal{L}(\boldsymbol{Y}_i; \boldsymbol{X}_i, \Theta) = -\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)} \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i) + \beta D_{\mathrm{KL}}(q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)\|p(\boldsymbol{Z}_i)), \tag{E.7}$$

where Θ represents all unknown parameters, including $\boldsymbol{U}$, $\boldsymbol{\pi}$, the unknown weights in the encoder and the decoder, and the dispersion parameters θ_g. The tuning parameter β, introduced in [52] as the β-VAE, is to balance the reconstruction error and regularization. To better adjust for the confounding $\boldsymbol{X}_i$ and increase robustness, we add three extra (optional) penalties to obtain our final loss function aggregated over all N cells

as:

$$\mathcal{L}(\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_N; \boldsymbol{X}_1, \cdots \boldsymbol{X}_N, \Theta) = -(1-\alpha) \sum_{i=1}^{N} \mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i)} \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i, \boldsymbol{X}_i)$$
$$- \alpha \sum_{i=1}^{N} \log p(\boldsymbol{Y}_i|\boldsymbol{Z}_i = \boldsymbol{0}_d, \boldsymbol{X}_i) + \beta \sum_{i=1}^{N} D_{\mathrm{KL}}(q(\boldsymbol{Z}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i) \| p(\boldsymbol{Z}_i))$$
$$+ \gamma\, \Omega_{\text{Jacobian}}(\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_N; \boldsymbol{X}_1, \cdots \boldsymbol{X}_N) + \kappa\, \Omega_{\text{MMD}}(\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_N; \boldsymbol{X}_1, \cdots \boldsymbol{X}_N). \tag{E.8}$$

where α is a tuning parameter for "soft" batch adjustment [12], Ω_{MMD} is the MMD loss to remove the difference of cells from different sources, and Ω_{Jacobian} is the Jacabian regularizer to stabilize the optimization processes (see Appendix S.2.1 for more details). Setting $\alpha > 0$ ($\alpha = 0.1$ by default) 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. We set $\gamma = 1, \kappa = 0$ as default.

All the terms in (E.8) can be approximated by the Monte Carlo method efficiently, as we show in Appendix S.2.1. Specifically, though the marginal density function $p(\boldsymbol{Z}_i)$ is involved in a complex hierarchical mixture model (E.4), it still has a closed-form representation. The optimization minimizing the loss function (E.8) can be efficiently done via stochastic backpropagation [53] [54] on mini-batches of data, with the commonly used amortized variation inference [55] for VAE.

Model estimation with practical considerations Instead of using all the genes, by default, we select highly variable genes and preprocess the gene expressions by scanpy [34]. Then the normalized, log-transformed, and scaled gene expressions are provided as the inputs of our VAE.

As the optimization of our loss (E.8) generally results in a locally optimal solution, we need a good initialization of our parameters, especially for $\boldsymbol{U}$ and $\boldsymbol{\pi}$ defined in (E.1) and (E.4). 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 Jacobian and MMD regularizers can also be used in this stage. 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.

The second step is to initialize the latent space after pretraining. In our experiments, we use annotated cell types to initialize the latent space and set k to be the number of cell types. However, one can also perform cell clustering with the Louvain algorithm [56] on the estimated posterior means $\widehat{\boldsymbol{\mu}}_{\boldsymbol{Y}_i, \boldsymbol{X}_i}$ of $\boldsymbol{Z}_i$. 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. We introduce a pruning mechanism for better estimating $\boldsymbol{\pi}$. Specifically, we set those π_i's corresponding to edges with the top 50% lengths (the distance between the initial centers) as 0 and freeze them when training.

The last step is to train our whole network, optimizing the loss function (E.8) with $\beta = 1$. During both the

pretraining and training steps, the optimization is early stopped when the evaluation loss decreases sufficiently slowly.

Finally, to make VITAE scalable to 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 G features 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 (E.8) is replaced by Gaussian densities assumed on $\boldsymbol{F}_i$ (Appendix S.2.1). As we will show in the Result Section, the accelerated VITAE with Gaussian densities provides comparable results with our original 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 only has a dimension of 8. The hidden layers are all fully connected with the leaky rectified linear activation function [57] and Batch Normalization [58].

S.1.2 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 $\boldsymbol{Z}_i$. 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 (E.4), 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 S.2.2. Now, we discuss how to infer the trajectory backbone and cell positions along the trajectory with these posterior approximations.

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, the estimated $\widehat{\boldsymbol{\pi}}$ may be dense on the unpruned entries. Inspired by [59] 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:

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

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 determine whether it is significantly nonzero or not, we simply use a deterministic version of the edge score as

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

where $\tilde{c}_i = \arg\max_{c \in \{1,2,\cdots,K\}} \widehat{p}(c_i = c | \boldsymbol{Y}_i, \boldsymbol{X}_i)$ and $\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}$ is the approximate posterior mean of $\tilde{\boldsymbol{w}}_i$. Other edge score choices are discussed in Appendix S.2.3.

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 [4], 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.

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:

$$\begin{aligned} \widehat{\tilde{\boldsymbol{w}}}_i = \arg\min_{\boldsymbol{w}} \quad & \mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i | \boldsymbol{Y}_i, \boldsymbol{X}_i)} \|\tilde{\boldsymbol{w}}_i - \boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & \text{support}(\boldsymbol{w}) \subseteq \widehat{\mathcal{B}},\ \|\boldsymbol{w}\|_1 = 1,\ \boldsymbol{w} \succeq \boldsymbol{0}. \end{aligned} \tag{E.9}$$

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 (E.9) is not the only choice, it can result in a closed-form solution that allows fast computation.

Proposition 1. *The optimization problem (E.9) is equivalent to finding*

$$\begin{aligned} \widehat{\tilde{\boldsymbol{w}}}_i = \arg\min_{\boldsymbol{w}} \quad & \|\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i} - \boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & \text{support}(\boldsymbol{w}) \subseteq \widehat{\mathcal{B}},\ \|\boldsymbol{w}\|_1 = 1,\ \boldsymbol{w} \succeq \boldsymbol{0}, \end{aligned} \tag{E.10}$$

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*

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

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 best projection reduces to be a vertex if* $j_1^* = j_2^*$.

The proof of Proposition 1 is included in Appendix S.2.4. 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 (E.10) 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. 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

$$\mathbb{E}_{\widehat{p}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}[d(\widehat{\tilde{\boldsymbol{w}}}_i, \tilde{\boldsymbol{w}}_i)] = \|\widehat{\tilde{\boldsymbol{w}}}_i - \boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}\|_2^2 + \text{tr}\left(\Sigma_{\tilde{\boldsymbol{w}}_i}\right), \tag{E.11}$$

which is easily computable. 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 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 S.2.5.

Finally, we obtain the point estimate of each cell's pseudotime T_i defined in (7). 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 [31] when cells are collected from a series of time points. The idea is to choose the vertex with the 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 $\omega_j = \sum_i \widehat{\tilde{\boldsymbol{w}}}_{ij} r_i / \sum_i \widehat{\tilde{\boldsymbol{w}}}_{ij}$, a 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 ω_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 (6). Then, a point estimate of the pseudotime T_i of cell i can be given as $\widehat{T}_i = \widehat{\boldsymbol{o}}^\top \widehat{\tilde{\boldsymbol{w}}}_i$.

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 [60] to find differentially expressed genes in TI, 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:

$$Y_{ig} = \beta_{0g} + \beta_{1g}\text{rank}(T_i) + \boldsymbol{X}_i^\top \boldsymbol{\beta}_{Xg} + e_{ig}, \quad \mathbb{E}(e_{ig}) = 0, \ \forall\, i \in \mathcal{S} \tag{E.12}$$

where Y_{ig} is the log-transformed and normalized count for cell i and gene g, and the linear term $\boldsymbol{X}_i^\top \boldsymbol{\beta}_{Xg}$ 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

$$\rho_g = \beta_{1g} \frac{\text{sd}[\text{rank}(T.)]}{\text{sd}(Y_{\cdot g})}. \tag{E.13}$$

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 [61]. 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 adjusted with the Benjamini-Hochberg procedure for multiple testing corrections.

Unlike our approach here, tradeSeq [60] 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 $\widehat{T}_i$. In the simplest case, 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, it only takes us less than 1 minute on 5000 cells and $15k$ genes, while tradeSeq can take more than 20 minutes as evaluated in [60]. Finally, we note that the above testing procedure can be readily generalized to polynomial regression by including the higher-order terms of $\hat{T}_i$. For example, a quadratic regression may help to identify DE genes that exhibit not only linear trends but also quadratic changes. Essentially, this also reduces to linear regression, which can be carried out efficiently, as explained previously.

S.2 Algorithm details and discussions

S.2.1 Loss Function

Basic loss function. Note that the loss for each cell in (E.8) takes the form

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

when $\Omega_{\text{MMD}}\equiv\Omega_{\text{Jacobian}}\equiv 0$. 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,

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

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

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

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}}{=\!=\!=\!

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

Jacobian regularization for stabilizing optimization. To stabilize the training process and make the model to be robust against noises, we adopt the Jacobian regularization method from [14]. Suppose $\boldsymbol{Z}_i \in \mathbb{R}^d$, $\boldsymbol{Y}_i \in \mathbb{R}^G$, $\boldsymbol{Z}_{ij}$ is the j-th coordinate of vector $\boldsymbol{Z}_i$ and $\boldsymbol{Y}_{ig}$ is the g-th coordinate of vector $\boldsymbol{Y}_i$. Then we control the Frobenius norm of the Jacobians of the output of the encoder $\boldsymbol{Z}_i$ with respect to the input $\boldsymbol{Y}_i$:

$$
\Omega_{\text{Jacobian}}(\boldsymbol{Y}_1,\cdots,\boldsymbol{Y}_N;\boldsymbol{X}_1,\cdots\boldsymbol{X}_N)=\sum_{i=1}^{N}\sum_{j=1}^{d}\sum_{g=1}^{G}\mathbb{E}_{q(\boldsymbol{Z}_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)}\left[\left(\frac{\partial\boldsymbol{Z}_{ij}}{\partial\boldsymbol{Y}_{ig}}\right)^2\right]. \tag{E.14}
$$

MMD regularization for removing conditional differences. We can also include the maximum mean discrepancy (MMD) loss as utilized in SAUCIE [62] and scArches [63], which explicitly penalizes the distances between cells from any pairings we want to adjust in the latent space. Let Z_1, Z_1', Z_2, Z_2' be independent samples from to two distributions $\mathcal{P}_1, \mathcal{P}_2$, then the MMD for these two distributions is:

$$
\mathcal{D}(\mathcal{P}_1,\mathcal{P}_2)=\mathbb{E}_{Z_1,Z_1'}[k(Z_1,Z_1')]+\mathbb{E}_{Z_2,Z_2'}[k(Z_2,Z_2')]-2\mathbb{E}_{Z_1,Z_2}[k(Z_1,Z_2)]
$$

where $k(\cdot,\cdot)$ is a kernel function. Empirically, if we want to integrate data from any two cell groups $\mathcal{P}_1$ and $\mathcal{P}_2$ which have n_1 and n_2 cells respectively, we punish large MMD in the latent space. The MMD loss for two groups can be defined as follows:

$$
L_{\text{MMD}}(\mathcal{P}_1,\mathcal{P}_2)=-\frac{1}{n_1^2}\sum_{i_1,i_2\in\mathcal{P}_1}k(\boldsymbol{Z}_{i_1},\boldsymbol{Z}_{i_2})-\frac{1}{n_2^2}\sum_{i_1,i_2\in\mathcal{P}_2}k(\boldsymbol{Z}_{i_1},\boldsymbol{Z}_{i_2})+\frac{2}{n_1n_2}\sum_{i_1\in\mathcal{P}_1,i_2\in\mathcal{P}_2}k(\boldsymbol{Z}_{i_1},\boldsymbol{Z}_{i_2})
$$

where $\boldsymbol{Z}_i$ is the latent vector for cell i. In practice, we employ a multi-scale RBF kernel, proposed in trVAE [30]. The multi-scale RBF kernel is defined as $k(\boldsymbol{Z}_{i_1},\boldsymbol{Z}_{i_2})=\sum_{i=1}^{l}k(\boldsymbol{Z}_{i_1},\boldsymbol{Z}_{i_2},\lambda_i)$ where $k(\boldsymbol{Z}_{i_1},\boldsymbol{Z}_{i_2},\lambda_i)=\exp(-\lambda_i\|\boldsymbol{Z}_{i_1}-\boldsymbol{Z}_{i_2}\|^2)$ and λ_i is a hyper-parameter. We set $l=19$ by default and let $\{\lambda_i\}_{i=1}^{l}=[10^{-6},10^{-5},10^{-4},10^{-3},10^{-2},0.1,1,5,10,15,20,25,30,35,100,10^3,10^4,10^5,10^6]$. Then, the MMD penalty for the whole dataset can be defined as:

$$
\Omega_{\text{MMD}}=\sum_{(s,t)\in\text{all pairs}}L_{\text{MMD}}(\mathcal{P}_s,\mathcal{P}_t).
$$

Suppose we wish to apply the MMD penalty to a set of k cell groups (such as replicates) denoted by $\{\mathcal{P}_1,\mathcal{P}_2,\ldots,\mathcal{P}_k\}$. We can define a set of pairs as pairs $=\{(i,j)\mid 1\le i<j\le k\}$ to represent all possible

pairwise combinations of the k groups.

S.2.2 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 noting 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 S.2.1, 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 \\
&\overset{\boldsymbol{Y}_i|\boldsymbol{Z}_i,\boldsymbol{X}_i \perp\!\!\!\perp (w_i,c_i)|\boldsymbol{Z}_i,\boldsymbol{X}_i}{=\!=\!=\!

$$\approx \frac{1}{M}\sum_{c=1}^{K}\sum_{j=1}^{M}\left[\frac{j}{M}a_c+\left(1-\frac{j}{M}\right)b_c\right]\frac{1}{L}\sum_{l=1}^{L}\frac{\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c,w_i=\frac{j}{M})\widehat{\pi}_c}{\sum_{c'=1}^{K}\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c')\widehat{\pi}'_c}$$

$$\mathrm{diag}(\Sigma_{\tilde{\boldsymbol{w}}_i})\approx \frac{1}{M}\sum_{c=1}^{K}\sum_{j=1}^{M}\left[\frac{j}{M}a_c+\left(1-\frac{j}{M}\right)b_c\right]^2\frac{1}{L}\sum_{l=1}^{L}\frac{\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c,w=\frac{j}{M})\widehat{\pi}_c}{\sum_{c'=1}^{K}\widehat{p}(\boldsymbol{z}^{(l)}|c_i=c')\widehat{\pi}'_c}-\boldsymbol{\mu}^2_{\tilde{\boldsymbol{w}}_i}.$$

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

S.2.3 Edge Score

Below, seven 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 $\frac{1}{N}\sum_{i=1}^{N}p(c_i|\boldsymbol{Y}_i,\boldsymbol{X}_i)$. 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 $s^{\text{mean}}_{j_1j_2}$, 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 $s^{\text{mean}}_{j_1j_2}$ is given by

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

   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 Raw 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 $e_{j_1 j_2}$ to the number of cells at the vertices j_1, j_2 or the edge $e_{j_1 j_2}$:

$$s_{j_1 j_2}^{\text{raw_map}} = \frac{|\{i : \tilde{c}_i = C_{j_1 j_2}\}|}{|\{i : \tilde{c}_i \in \{C_{j_1 j_2, j_1 \neq j_2}\}\}|}.$$

4. Edge Score Based on MAP of c.
   We can also define the edge score of c_i as the proportion of the number of cells at the edge $e_{j_1 j_2}$ to the number of cells at the vertices j_1, j_2 or the edge $e_{j_1 j_2}$:

$$s_{j_1 j_2}^{\text{map}} = \frac{|\{i : \tilde{c}_i = C_{j_1 j_2}\}|}{|\{i : \tilde{c}_i \in \{C_{j_1 j_1}, C_{j_1 j_2}, C_{j_2 j_2}\}\}|}.$$

5. 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 $s_{j_1 j_2}^{\text{map}}$ will be large when the number of cells in nodes j_1 and j_2 is small. To address this problem, we further modify the denominator to get the modified MAP edge score:

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

6. Edge Score Based on MAP of $\mu_{\tilde{w}_i}$.
   Sometimes the large number of cell type k will boost the dimension of $\widehat{p}(c_i = c | \boldsymbol{Y}_i, \boldsymbol{X}_i)$ to $\frac{k(k-1)}{2}$, which may cause the edge score inference sensitive to $\widehat{p}(c_i = c | \boldsymbol{Y}_i, \boldsymbol{X}_i)$ estimation. Therefore, in the dataset containing lots of cell types, we can compute the edge score from $\mu_{\tilde{w}_i}$, which only has k dimension and also contains the position information. First, define a set $MAP(j_1, j_2)$ containing the cells that have the largest $\mu_{\tilde{w}_i}$ value in the vertices j_1, j_2 dimension.

$$MAP(j_1, j_2) := \{i : \arg\max_j e_j^T \mu_{\tilde{w}_i} = (j_1 \text{ or } j_2)\}$$

   Then, the edge score can be defined by the proportion of the number of cells that lie in the middle of two vertices to the number of cells that have the largest $\mu_{\tilde{w}_i}$ value in these vertices' dimensions:

$$s_{j_1, j_2}^{\text{w_base}} = \frac{|\{i : abs(e_{j_1}^T \mu_{\tilde{w}_i} - e_{j_2}^T \mu_{\tilde{w}_i}) < 0.1, i \in MAP(j_1, j_2)\}|}{|MAP(j_1, j_2)|}$$

7. Edge Score Based on Modified MAP of $\mu_{\tilde{w}_i}$.
   When there exists some unbalanced cell type categories, $MAP(j_1, j_2)$ that only care about the 1st

rank dimension of $\mu_{\tilde{w}_i}$ may cause the edge score unexpectedly small when one of the vertices has a large number of cells while the other only has a small number. Then we can modified $MAP(j_1, j_2)$ to consider the largest 2 dimension of $\mu_{\tilde{w}_i}$.

$$MAP2(j_1, j_2) := \{i : \underset{j' \subset [k], |j'|=2}{\arg\max} \sum_{j \in j'} e_j^T \mu_{\tilde{w}_i}, j' = \{j_1, j_2\}\}$$

Then, the edge score can be defined by the proportion of the number of cells that lie in the middle of the two vertices j_1, j_2 to the number of cells that belong to $MAP2(j_1, j_2)$:

$$s_{j_1,j_2} = \frac{|\{i : \max_j e_j^T \mu_{\tilde{w}_i} < 0.55 \cdot (e_{j_1} + e_{j_2})^T \mu_{\tilde{w}_i}), i \in MAP2(j_1, j_2)\}|}{|MAP2(j_1, j_2)|}$$

By default, the package uses edge scores based on the modified MAP of c.

S.2.4 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 (10) is thus equivalent to the optimization problem (11).

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

$$\begin{aligned}
\min_{(j_1,j_2) \in \mathcal{E}(\widehat{\mathcal{B}})} \quad & (\boldsymbol{w}_{j_1} - \mu_{j_1})^2 + (\boldsymbol{w}_{j_2} - \mu_{j_2})^2 - (\mu_{j_1}^2 + \mu_{j_2}^2) \\
\text{s.t.} \quad & \|\boldsymbol{w}\|_1 = w_{j_1} + w_{j_2} = 1, \quad \boldsymbol{w} \succeq \boldsymbol{0}_k.
\end{aligned}$$

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

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

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

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

$$
\begin{aligned}
&\|\text{proj}_{(j_1,j_2)}(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}})-\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}\|_2^2\leq\|\text{proj}_{(m_1,m_2)}(\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}})-\boldsymbol{\mu}_{\tilde{\boldsymbol{w}}_i}\|_2^2\\
\iff\quad&\sum_{k'\neq j_1,j_2}\mu_{k'}^2+\frac{(1-\mu_{j_1}-\mu_{j_2})^2}{2}\leq\sum_{k'\neq m_1,m_2}\mu_{k'}^2+\frac{(1-\mu_{m_1}-\mu_{m_2})^2}{2}\\
\iff\quad&(\mu_{j_1}-\mu_{j_2})^2+2(\mu_{j_1}+\mu_{j_2})\geq(\mu_{m_1}-\mu_{m_2})^2+2(\mu_{m_1}+\mu_{m_2}).
\end{aligned}
\tag{E.15}
$$

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

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

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

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

gives solutions to both optimization problems. ■

S.2.5 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 (12). Specifically, we define our hierarchical model with observed $\boldsymbol{Z}_i$ following (3) and (5) 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 S.1a and we choose 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 S.2.2 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 S.1b 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 S.1b has an M-shape as shown in Figure S.1c, 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 is 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.

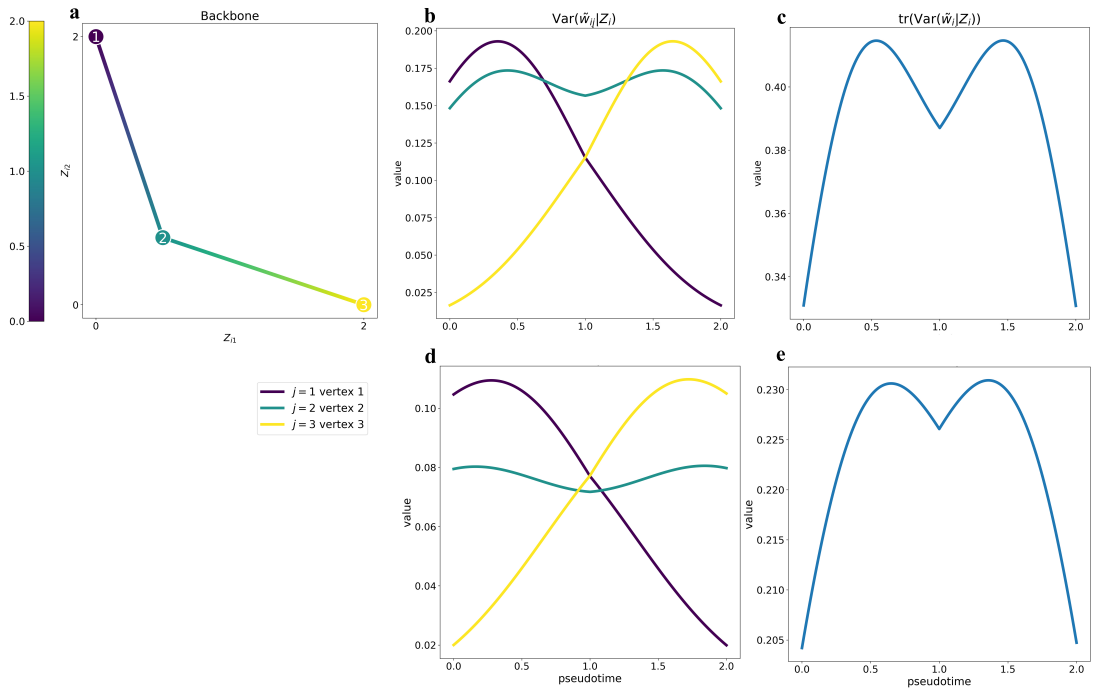

Figure S.1: **(a)** The underlying backbone of the assumed model colored by pseudotime. **(b)** The values of $\text{Var}(\tilde{w}_{ij}|\boldsymbol{Z}_i)$ along the trajectory and **(c)** the values of $\text{tr}(\text{Var}(\tilde{\boldsymbol{w}}_i|\boldsymbol{Z}_i))$ along the trajectory are for the 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 $\boldsymbol{\pi} = (0, 0.5, 0, 0, 0.5, 0)$.

S.3 Simulation

S.3.1 Simulation Datasets

Following the settings in [5], 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 Fig. S.2. Our benchmarking datasets include 10 real scRNA-seq data and 15 synthetic datasets, summarized in Table 1.

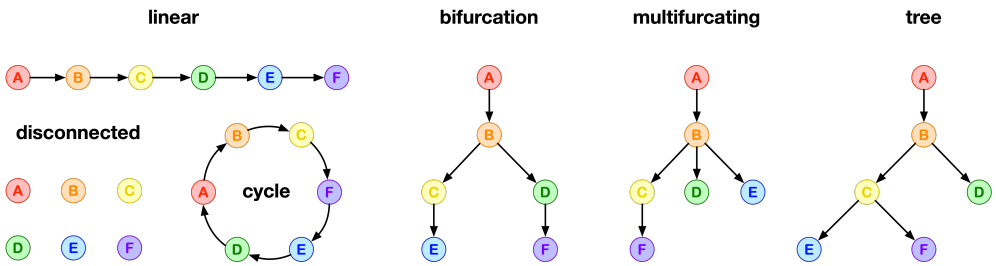

Figure S.2: Six different topologies of the underlying trajectories in real and synthetic datasets.

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	[64]
	human_embryos	non-UMI	linear	1289	8772	5	[65]
	germline	non-UMI	bifurcation	272	8772	7	[66]
	mesoderm	non-UMI	tree	504	8772	9	[67]
	cell_cycle	non-UMI	cycle	264	6812	3	[65]
	dentate	UMI	linear	3585	2182	5	[68]
	fibroblast	non-UMI	bifurcation	355	3379	7	[69]
	planaria_muscle	UMI	bifurcation	2338	4210	3	[8]
	planaria_full	UMI	tree	18837	4210	33	[8]
	immune	UMI	disconnected	21082	18750	3	[70]
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
	cycle_3	non-UMI	cycle	2000	999	3	dyngen
	linear	UMI	linear	1900	1990	5	our model
	bifurcation	UMI	bifurcation	2000	606	6	our model
	multifurcating	UMI	multifurcating	2000	606	6	our model
	tree	UMI	tree	2000	606	6	our model

The datasets from real scRNA-seq studies include 9 datasets from [5]. Among them, 5 datasets have "gold standard" labels according to [5] 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

[5], except for the dataset *dentate*, whose cells are mislabeled, and we directly extract the 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 [70].

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. We consider two different simulation approaches. One simulator we use is dyngen [71], a multi-modal simulation engine for studying dynamic cellular processes at single-cell resolution. dyngen is also used in [5] and provides a delicate way to generate scRNA-seq data starting from gene regulation and transcriptional factors. However, it is limited to generating only a few hundred 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 typically large.

We also 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 models ((E.1) and (E.4)). 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*.

S.3.2 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 [72] 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, following [5], we also compute the IM score based on the Ipsen-Mikhailov distance [73], $\text{IM}(\mathcal{B}, \widehat{\mathcal{B}})$, 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

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
Ruan's Dataset	1172	0	2668	0	3742	1014	387	0	1278
Yuzwa's Dataset	0	1418	0	1137	0	2955	0	880	0

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 an unequal number of vertices. The ARI [74] is a commonly used symmetric measure for the similarity between two clustering results. Following [5], the cells are assigned to their 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 $\tilde{\boldsymbol{w}}_i$, which 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 similarities 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 ranging between 0 and 1 as $s_{\text{PDT}} = \left(\text{Cor}(\boldsymbol{T}, \widehat{\boldsymbol{T}}) + 1\right)/2$.

S.4 Real data

S.4.1 Experiments on mouse brain datasets

To adjust for the confounding batch effect, we include the dataset ID of each cell (whether it is from Yuzwa's or Ruan's dataset) as covariates. 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. To perform the joint trajectory analysis, we apply VITAE_Gauss with the aforementioned covariates. We provide the annotated cell types to initialize the latent space after the pretraining step. For the model hyperparameters, we use 64 PCs, two hidden layers with units 32 and 16 for the encoder, a symmetric structure for the decoder, and 8

units for the latent space. For pre-training and training, we set $\gamma = 1$ for the Jacobian regularizers.

S.4.2 Experiments on the integration of three mouse brain datasets

Experiment setup. To adjust for confounding batch effect, we include the dataset ID of each cell (whether it is from Di Bella's, Yuzwa's, or Ruan's dataset) as covariates. In addition, as the Apical Progenitors are experiencing cell division, we also adjust for the cell-cycle effects by adding the cell-cycle scores (`S.Score` and `G2M.Score`) provided by their own datasets. To perform the joint trajectory analysis, we apply VITAE_Gauss with the aforementioned covariates. We provide the annotated cell types to initialize the latent space after the pretraining step. For the model hyperparameters, we use 96 PCs, two hidden layers with units 48 and 24 for the encoder, a symmetric structure for the decoder, and 16 units for the latent space. Since the dataset is larger and more complicated, we retain more PCs as input and use a wider network structure. For pre-training and training, we set $\gamma = 1$ for the Jacobian regularizers.

Evaluation metric for integration. We construct a mixing score based on the reduced UMAP space to measure the mixing effect of Day E18 Replicate, Day P1 Replicate, different days, and different cell types within the same day. We first calculate the KNN graph in UMAP space. Suppose we have n categories, and we want to quantify how these n categories are mixed. In the k neighbors of a cell, assume there are x cells that have the same category as this cell; then, for each cell, the mixing metric can be calculated by

$$\text{mixing score} = 1 - \frac{(x - \frac{k}{n})^+ + [k - \frac{k}{n} - (k - x)]^+}{2(1 - \frac{1}{n})k},$$

where $(x - a)^+ = \max\{x - a, 0\}$. The score reaches 1 when $x \leq k/n$ and reaches 0 when $x = k$. The denominator $2(1 - n^{-1})k$ is used to normalize the score between 0 and 1. This score can also be converted to quantify the separation of different categories:

$$\text{separation score} = 1 - \text{mixing score}.$$

Consequently, we use the mixing score to measure the mixing effect of Day E18 Replicate, Day P1 Replicate, and different days. The separation score is applied to quantify the biological cell type conservation on different days.

S.4.3 Experiments on multi-omic human hematopoiesis datasets

To adjust for confounding batch effect, we include the sequencing technique of each cell (scRNA or scATAC) and also use MMD loss calculated within each annotated cell type to remove effects from different sequencing techniques better. For the model hyperparameters, we use 128 PCs, two hidden layers with units 32 and 16 for the encoder, a symmetric structure for the decoder, and 8 units for the latent space. For pre-training, we set $\gamma = \kappa = 0.6$ for the Jacobian and MMD regularizers. For training, we set $\gamma = \kappa = 1$ for the Jacobian and MMD regularizers.

S.5 Extra experimental results

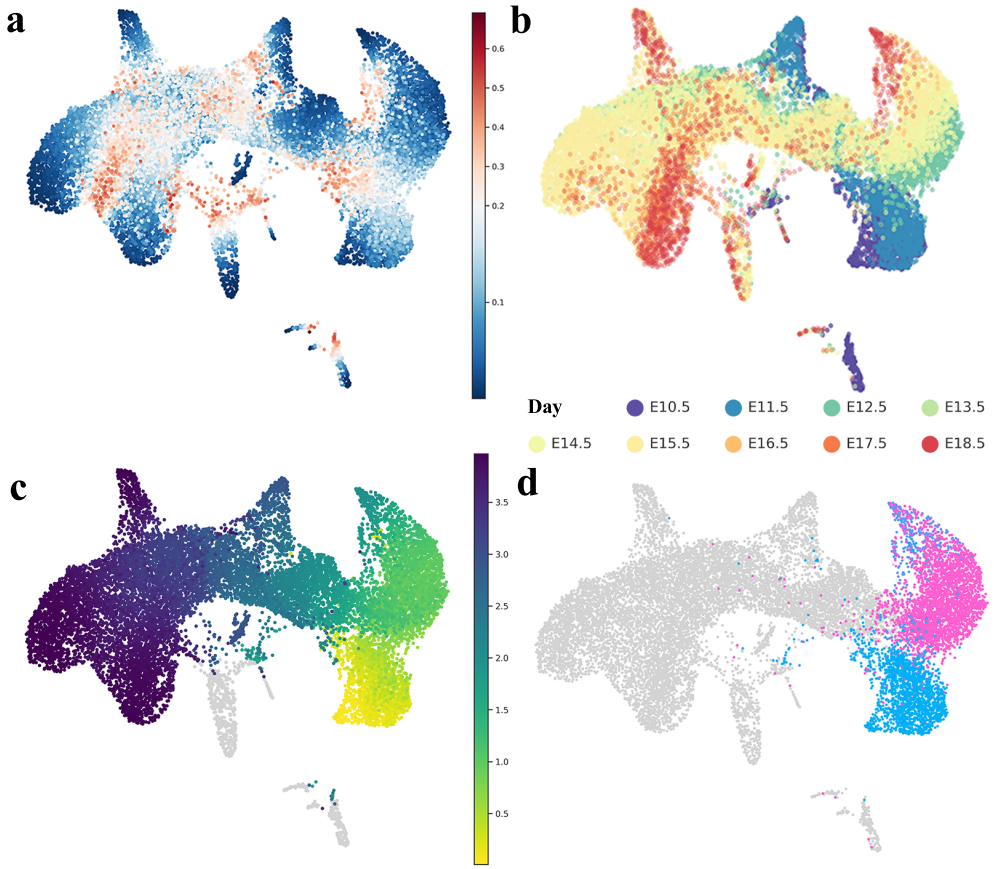

Figure S.3: UMAP visualizations of VITAE's low-dimensional cell representations on the joint analysis of Yuzwa and Ruan datasets. **(a)** Low-dimensional embeddings colored by the projection uncertainty estimated by VITAE. When projected to the inferred trajectory, Cells colored in red have higher uncertainties compared to cells colored in blue. **(b)** Low-dimensional embeddings colored by collection days. **(c)** Low-dimensional embeddings colored by the pseudotime estimated by VITAE. Cells not in the inferred trajectory are colored in gray. **(d)** Low-dimensional embeddings highlighting the NEC-RGC-OPC sub-trajectory where the cell types NEC, RGC, and OPC are highlighted.

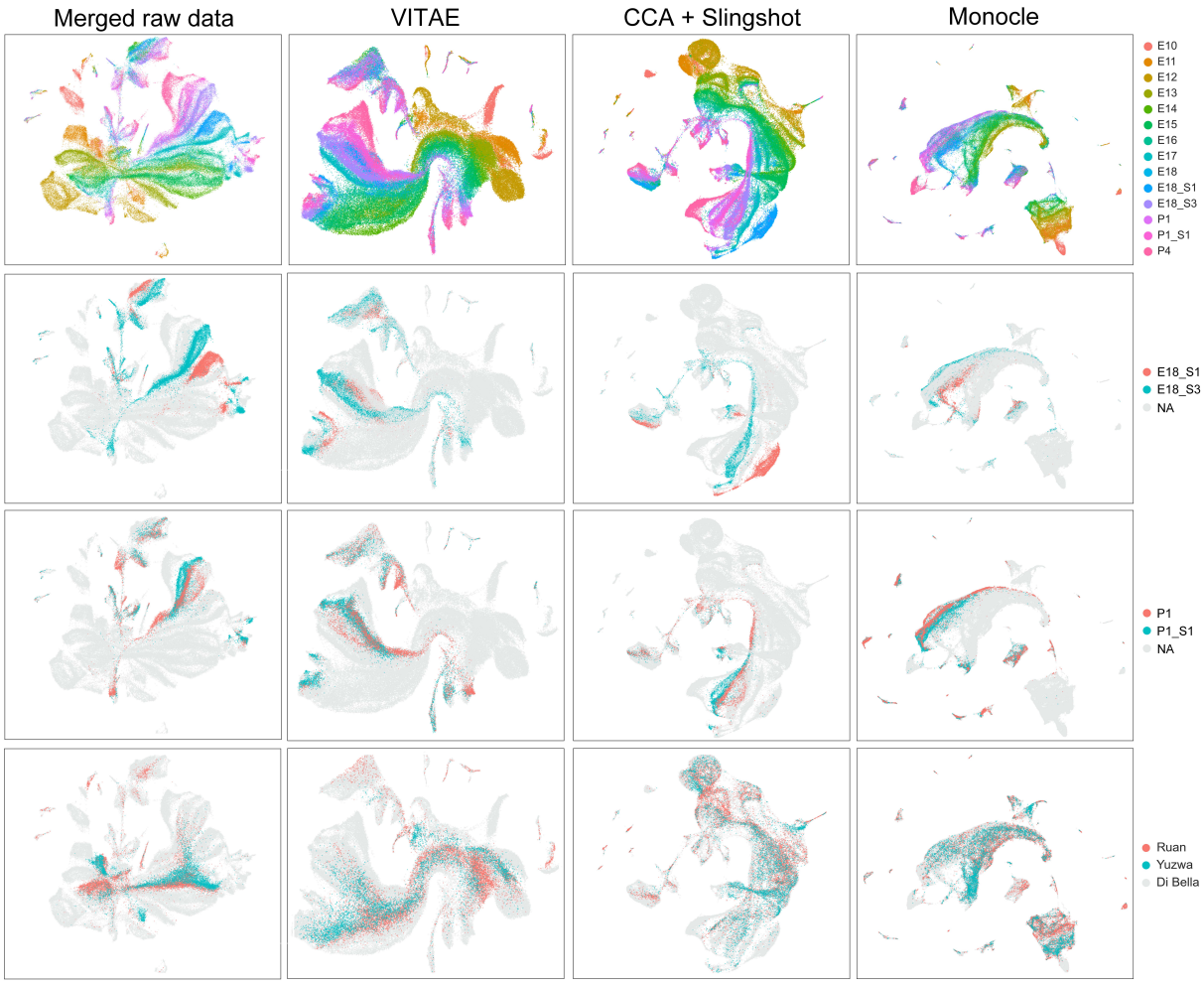

Figure S.4: UMAP visualizations of the joint analysis of Di Bella, Yuzwa and Ruan datasets. Each column represents a method. For the first column, cells are normalized and then concatenated to perform a joint PCA analysis before UMAP. In the first row and last row, cells are colored by their collection days and sources. In the second and third rows, cells from the two replicates at E18 and two replicates at P1 from the Di Bella dataset are highlighted.

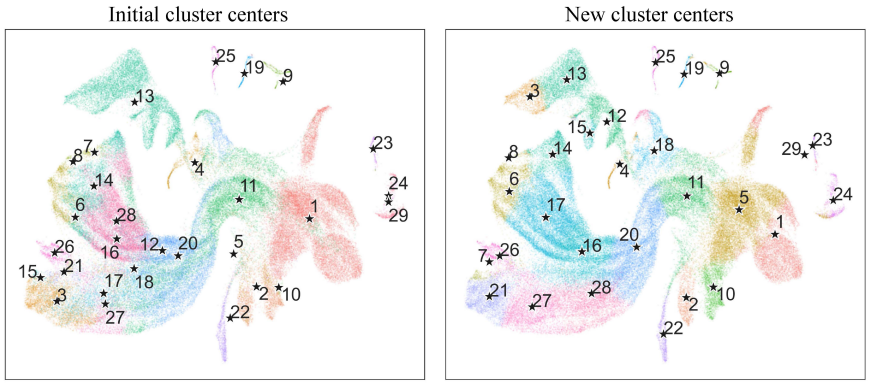

Figure S.5: Locations of the cluster centers before and after VITAE training for the joint analysis of Di Bella, Yuzwa and Ruan datasets VITAE uses the reference cell types to initialize the locations of the vertices before training, and then the vertices will change locations and give new clustering results after training. The left and the right panels show the initial and trained cluster centers, respectively. For interneurons, though only one vertex (cluster 13) is initialized, VITAE can automatically learn four cluster centers after training.

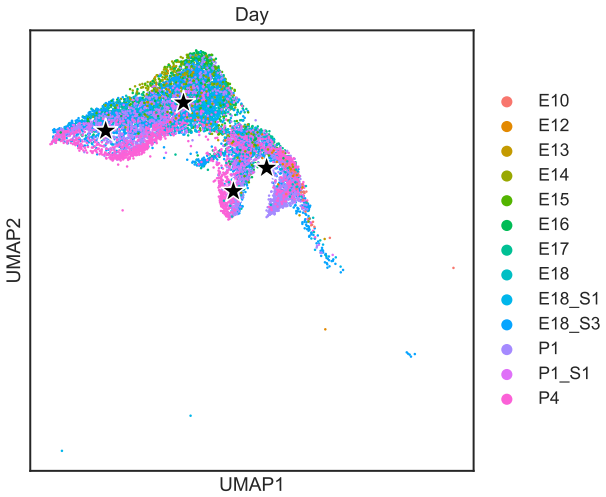

Figure S.6: Interneurons colored by collection days.

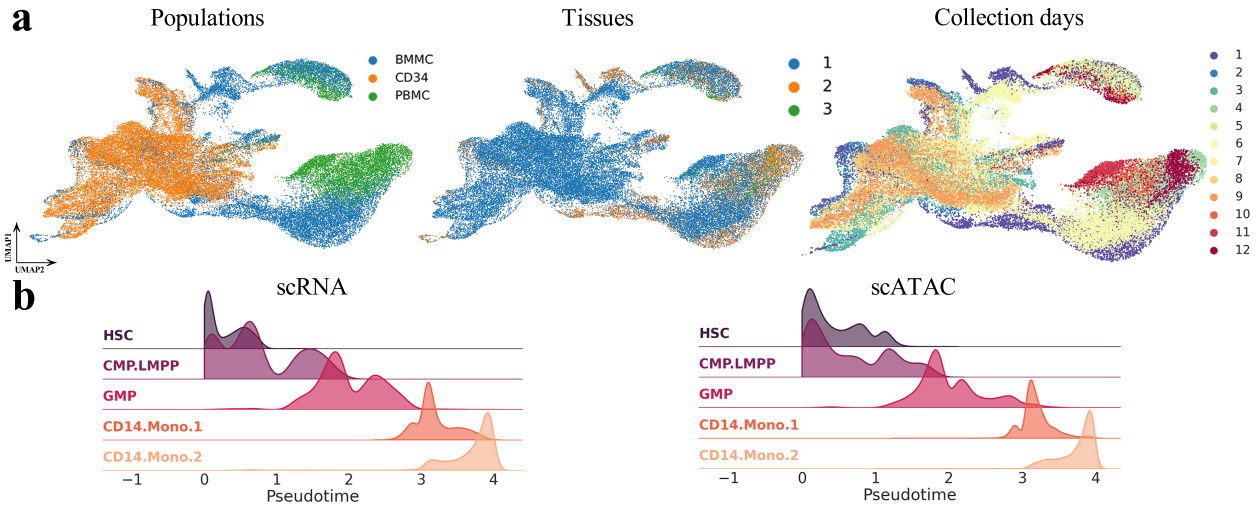

Figure S.7: Integrative trajectory inference of multi-omic human hematopoiesis data. **(a)** UMAP visualization of VITAE's low-dimensional embedding of cells, colored by cell populations ('BMMC', 'CD34', and 'PBMC' stand for bone marrow mononuclear cells, CD34$^+$-enriched bone marrow mononuclear cells, and peripheral blood mononuclear cells, respectively), tissues, replicates, and collection days. VITAE merges the gene expressions (scRNA) and the gene activities (scATAC) score, retains meaningful biological variations and correctly infers the developmental trajectory. **(b)** Distribution of pseudotime of the monocytic lineage for scRNA-seq and scATAC-seq cells.