Inferring single-cell dynamics with structured dynamical representations of RNA velocity

Spencer Farrell,[1,*] Madhav Mani,[2,3,4] and Sidhartha Goyal[1,5,†]

[1] *Department of Physics, University of Toronto*
[2] *Department of Engineering Sciences and Applied Mathematics, Northwestern University*
[3] *NSF-Simons Center for Quantitative Biology, Northwestern University*
[4] *Department of Molecular Biosciences, Northwestern University*
[5] *Institute of Biomedical Engineering, University of Toronto*

RNA velocity provides directional information for trajectory inference from single-cell RNA-sequencing data. Traditional approaches to computing RNA velocity rely on strict assumptions about the equations describing transcription of unspliced RNA and splicing of unspliced RNA into spliced RNA. This results in issues in scenarios where these assumptions are violated, such as multiple lineages with distinct dynamics and time-dependent rates. In this work we present "LatentVelo", a novel approach to computing a low-dimensional representation of RNA velocity with deep learning. Our approach embeds cells into a latent space with a variational auto-encoder, and describes differentiation dynamics on this latent space with neural ordinary differential equations. These more general dynamics enable accurate trajectory inference, and the latent space approach enables the generation of dynamics-based latent embeddings of cell states and batch correction of cell states and of RNA velocity. By performing simultaneous batch correction of RNA velocity dynamics and gene expression, we also outperform traditional batch correction methods that only consider gene expression. The flexible structure of the model enables modelling a variety of regulatory structures and multi-omic data, or incorporating additional information such as cell-type annotations or experimental metadata to improve the latent embedding. LatentVelo will be available at `https://github.com/Spencerfar/LatentVelo`.

I. INTRODUCTION

Single-cell RNA sequencing enables the inference of developmental differentiation trajectories by computationally reconstructing the developmental process. Traditional approaches are based on the similarity of static snapshots of RNA for individual cells [1–3]. RNA velocity extends these approaches by modelling the dynamical relationship between newer unspliced RNA and mature spliced RNA to infer the direction of the dynamics from these static snapshots. Recent techniques have also been developed to analyze RNA velocity, aiming to improve and expand upon traditional trajectory reconstruction methods [4–9].

However, RNA velocity methods have been limited by a reliance on strict modelling assumptions. In the original formulation, Velocyto [10] modelled cells at an assumed steady-state, with a linear relationship between unspliced and spliced RNA. scVelo [11] relaxed the steady-state assumption by time-dependent modelling of transient cell-states by fitting a set of linear differential equations, treating the unobserved development time as a latent variable. These methods encounter problems with complex dynamical features such as a transcriptional boost, lineage-dependent kinetics, and weak unspliced signal [12–14].

We have developed "LatentVelo", an approach to address these problems in RNA velocity estimation. Rather than learn RNA velocity on gene-space, our model embeds cell states in a learned latent space, and infers structured dynamics on this latent space. Lineage-dependent dynamics are enabled by modelling state-dependent regulation of transcription. By learning dynamics in a latent space, LatentVelo can extract the low-dimensional dynamical system describing cell differentiation. Additionally, modelling the latent space dynamics enables batch correction and the incorporation of additional information such as annotated cell-type labels as in scANVI [15] or temporal information from sequencing batches. LatentVelo also enables constructing general dynamical models, and enabling various structured models of regulation and multi-omic data.

Recently, several other methods have been proposed to address some of these issues to improve the accuracy and reliability of RNA velocity methods. VeloAE [16] develops an extension of the steady-state model by projecting high-dimensional noisy unspliced and spliced vectors onto a lower dimension latent space with an autoencoder, and then uses the steady-state linear model to estimate velocities on this latent space. UniTVelo [17] allows for a more general form of spliced RNA dynamics which relaxes assumptions on the transcription rate, includes a unified cell-dependent latent time (rather than per gene as in scVelo), and introduces a method of dealing with low signal-to noise

* spencer.farrell@utoronto.ca
† goyal@physics.utoronto.ca

in unspliced reads. DeepVelo [18] uses a deep neural network to estimate cell-specific kinetic rate parameters to model variable and lineage-dependent kinetics. VeloVAE [19] uses a variational Bayesian approach to model a unified cell-dependent latent time, cell-specific transcription rates, and celltype specific splicing and degradation rates. MultiVelo [20] integrates chromatin accessibility into the linear dynamical model to improve velocity estimates. Alternatively, work has been done on gene selection for velocity analysis [12, 21].

"LatentVelo" has similarities with these methods. LatentVelo embeds cell states in a latent space with an auto-encoder like VeloAE, but includes dynamics rather than just modelling steady-state cells. LatentVelo models dynamics with a unified cell developmental time like UniTVelo and VeloVAE. Like DeepVelo and VeloVAE, LatentVelo models lineage-dependent dynamics. LatentVelo also includes the scTour model [22] as a special case, where only the dynamics of a single data-type are considered (e.g. just spliced RNA).

We benchmark LatentVelo with synthetic data and real developmental, regeneration, and reprogramming data. We show that our approach strongly outperforms the traditional RNA velocity methods Velocyto and scVelo at inferring trajectories, and has comparable performance or better to recent methods like UniTVelo and DeepVelo. We also show that our approach is able to do batch correction of RNA velocity by projecting batches onto a common latent space independent of technical variation between the samples. Batch correction is unaddressed by the other recent models of RNA velocity, presenting a novel application of LatentVelo. This work is also the first to do a systematic evaluation of batch correction for RNA velocity.

II. OVERVIEW OF LATENTVELO

A. RNA velocity

RNA velocity methods estimate the direction of differentiation using static snapshots of unspliced and spliced RNA. These observations are two different quantities in the processes of transcription, splicing, and degradation of RNA:

$$\emptyset \underset{transcription}{\rightarrow} \text{Unspliced} \underset{splicing}{\rightarrow} \text{Spliced} \underset{degradation}{\rightarrow} \emptyset. \tag{1}$$

Using the causal structure of these processes and the corresponding measurements, we aim to estimate the time-derivative of the spliced RNA for single-cells $\frac{ds}{dt}$, which characterizes the future direction of differentiation for a cell.

B. LatentVelo

LatentVelo is formulated as a variational auto-encoder (VAE) that embeds RNA count data into latent states $\hat{\mathbf{z}}$ along with a latent developmental time $\hat{t}$ per cell, which provides a time-ordering of the cells. Variational auto-encoders are approximate Bayesian models that use a neural network based encoder to approximate the posterior distribution of a model [23, 24]. In our case, we estimate the posterior distribution $p(\hat{\mathbf{z}}, \hat{t}|\mathbf{s}, \mathbf{u})$ of latent states and latent developmental times.

Dynamics on this latent space are described by a neural ordinary differential equation [25], using the inferred latent time as the time variable in the differential equation. The basic form of the latent dynamics are,

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{f}(\mathbf{z}(t)), \tag{2}$$

$$\mathbf{z}(0) = \mathbf{z}_0, \tag{3}$$

where the dynamics are determined by a neural network $\mathbf{f}$ and an initial state $\mathbf{z}_0$. These dynamics are incorporated into the VAE by constraining the encoded state state $\hat{\mathbf{z}}$ with the solution to these dynamics at the corresponding latent time for the cell $\mathbf{z}(\hat{t})$, e.g. minimizing $|\hat{\mathbf{z}} - \mathbf{z}(\hat{t})|$. This encourages the encoder to output latent states $\hat{\mathbf{z}}$ that follow the latent dynamics, and encourages the latent dynamics $\mathbf{z}(t)$ to match the output of the encoder. These dynamics are considered "unstructured" in the sense that $\mathbf{f}$ is a general dense feed-forward neural network, and interactions between all components of $\mathbf{z}$ are allowed. A very similar model with these unstructured dynamics was also recently developed in scTour [22].

However, these unstructured models do not utilize biological knowledge of the the dynamics of splicing. We add this structure by separately representing the different data modalities in the latent space,

$$\frac{d\mathbf{z}_u(t)}{dt} = \mathbf{f}_u(\mathbf{z}_u(t)) \tag{4}$$

$$\frac{d\mathbf{z}_s(t)}{dt} = \mathbf{f}_s(\mathbf{z}_u(t), \mathbf{z}_s(t)). \tag{5}$$

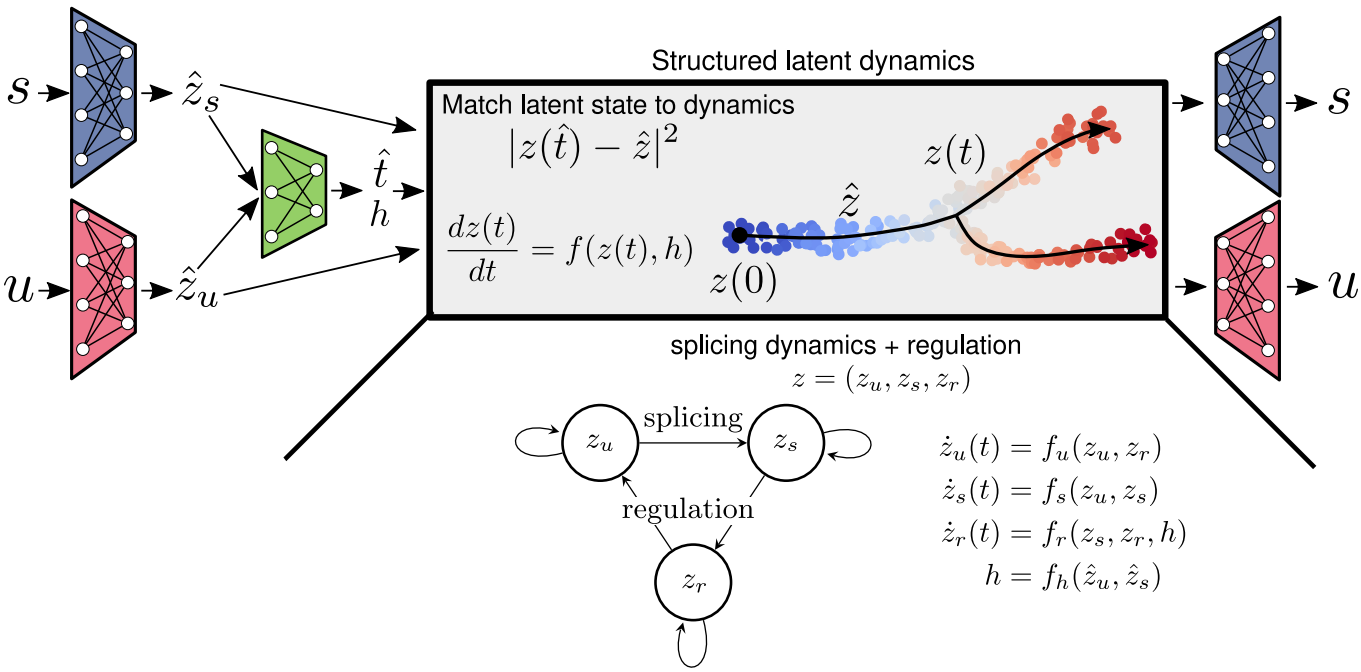

FIG. 1. **LatentVelo.** Our model is a variational autoencoder with structured dynamics on the latent space. An encoder (left trapezoids) encodes the transcriptomic cell state (i.e. unspliced (u) and spliced (s) RNA counts) into corresponding latent states $\hat{z}_s$ and $\hat{z}_u$, and a latent developmental time t and regulation state $\mathbf{h}$. The regulation state h conditions dynamics for each cell to follow a particular branch. We match these latent states to the dynamics obtained by structured dynamics on this latent space, $\mathbf{z}(t)$, described by ODEs starting from an initial state $\mathbf{z}(0)$. These structured dynamics can take a variety of forms, here we show regulated splicing dynamics incorporating state-dependent regulation of transcription. In the shown notation, we use $\hat{\mathbf{z}}$ to denote latent cell states estimated directly from the encoder with the input data, and $z(t)$ to represent latent cell states estimated from the latent dynamics. When fitting the model, the distance between the encoder latent states and the dynamics latent states is minimized.

The structure comes in by separately modelling spliced $\mathbf{z}_s$ and unspliced $\mathbf{z}_u$ latent representations, and restricting the interactions between them to model splicing dynamics. This results in an estimate of a latent representation of RNA velocity, $d\mathbf{z}_s/dt$. This model is a generalized form of traditional RNA velocity methods – modelling the dynamics of transcription, splicing, and degradation with non-linear functions of spliced and unspliced RNA, and including interactions between different genes through this low-dimensional latent space.

However, this form of dynamics does not address the issue of multiple lineages. Lineage-dependent dynamics can be incorporated by including regulation of transcription with state-dependent dynamics of a new latent variable controlling regulation $\mathbf{z}_r$,

$$\frac{d\mathbf{z}_u(t)}{dt} = \mathbf{f}_u(\mathbf{z}_u(t), \mathbf{z}_r(t)) \tag{6}$$

$$\frac{d\mathbf{z}_s(t)}{dt} = \mathbf{f}_s(\mathbf{z}_u(t), \mathbf{z}_s(t)), \tag{7}$$

$$\frac{d\mathbf{z}_r(t)}{dt} = \mathbf{f}_r(\mathbf{z}_s(t), \mathbf{z}_r(t), \mathbf{h}), \tag{8}$$

$$\mathbf{h} = \mathbf{f}_h(\hat{\mathbf{z}}_s, \hat{\mathbf{z}}_u). \tag{9}$$

The variable $\mathbf{h}$ is a constant state-dependent parameter controlling the regulatory dynamics for a particular cell. Since $\mathbf{h}$ is estimated from the cell-state, it enables dynamics to depend on cell-states or lineages. Since the dynamics are deterministic and start from a constant $\mathbf{z}(0)$, without $\mathbf{h}$ we would expect no branching of the dynamics. We can think of $\mathbf{h}$ as conditioning the dynamics to follow a particular branch of the system. This enables us to model complex multi-lineage dynamics. Note that while the VAE includes stochasticity in the latent embedding, there is no stochasticity in the dynamics. We address this in the discussion as a future direction.

We enforce the direction of splicing (unspliced $\rightarrow$ spliced) by regularizing a positive correlation between $\dot{s}$ and u and a negative correlation between $\dot{s}$ and s, e.g., $\text{corr}(\dot{s}, u) > 0$ and $\text{corr}(\dot{s}, s) < 0$ per gene. By only weakly including

this regulation, we allow time-dependent rates rather than the strict linear assumption of some previous models. This regularization is also done in DeepVelo [18]. Further details are in the methods.

Any desired structure to the latent dynamics can be implemented, for example regulation of splicing can be included with an interaction from z_r to z_s, or the addition of multi-omic data, which we discuss in the supplemental with chromatin accessibility from ATAC-seq. We have not explored other variations to the splicing and regulation structure.

The standard VAE version of the model uses a standard Gaussian prior on the latent state $\hat{\mathbf{z}} \sim \text{Normal}(0, 1)$ and a logit-normal prior on the latent times $\hat{t} \sim \text{LogitNormal}(0, 1)$. We can add more structure to the latent embedding by including cell-type information, similar to scANVI [15]. This approach modifies the prior of the latent space to be cell-type specific, rather than a standard Gaussian for all cells. This modification prevents the prior from ignoring biologically relevant clusters, and in particular, we show below that this improves batch correction. The full details of this modification are in the methods.

III. RESULTS

A. Evaluation

We take a quantitative approach to evaluate RNA velocity by comparing with ground truth velocity on synthetic data, and the direction of known cell-type transitions in real data.

To evaluate LatentVelo on synthetic data, we use the cosine similarity between ground truth and estimated velocities. Since many genes are not directly involved in the differentiation process, we compute this metric on lower dimensional embeddings; the model latent space and 50-dimensional principle component (PC) space.

For real data, we use datasets where the direction of differentiation is known. To quantitatively evaluate models, we use a modified version of the Cross-Boundary Directedness score used for UniTVelo and VeloAE [16, 17]. This score measures the velocity direction of cells on the boundary between two cell-types.

We make two modifications of this score. In previous work this score was computed on a 2D UMAP embedding, but we found this sometimes resulted in inconsistent values not seen in higher-dimensional embeddings. Therefore, we use multiple higher-dimensional embeddings (model latent space, higher dimensional PCA). Additionally, we modify the score to be more robust to noisy boundaries between cell-types, by only checking that the direction of the velocity of a cell is directed towards any cell of the expected cell-type, rather than any specific cell (see methods).

Our updated version quantifies the probability that a cell is likely to transition in the specified direction. Therefore, we can interpret scores above 0.5 to indicate a likely transition in that direction.

We also use the Inner-Cluster Coherence score used for UniTVelo and VeloAE [16, 17]. This score evaluates the coherence of velocity direction for neighboring cells within a cluster or cell-type. At a maximum value of 1, neighboring cells have the same velocity direction. A similar consistency score was also used with scVelo [11] and DeepVelo [18].

We primarily compare our model with scVelo dynamical and stochastic modes [11]. Comparisons with the default settings of UniTVelo unified and independent modes [17], and DeepVelo [18] are shown in the supplemental. Note: scVelo stochastic mode is an updated version of the Velocyto steady-state model, where regression is done for the first 2 moments of the dynamics instead of just the mean [11]. UniTVelo unified mode uses a unified latent time like our model, UniTVelo independent mode uses a gene-specific latent time like scVelo.

We use the kBET and iLISI batch correction metrics and the cLISI biological cluster conservation metrics to evaluate our models performance for batch correction [26]. kBET and iLISI measure how well batches are integrated together, and cLISI measures how well biological clusters are retained when batch correcting. These metrics only evaluate batch correction of gene expression. To evaluate the batch correction of RNA velocity, we compute the average cosine similarity between neighboring cells.

B. Benchmarking on synthetic and real datasets

We use dyngen [27] to generate synthetic datasets of 5000 cells each with a variety of developmental structures: linear (51 genes), bifurcation (65 genes), trifurcation (81 genes), and a binary tree (89 genes). In Figure 2 (top) we show results on these synthetic datasets. We show the 50 principle component velocity cosine similarity and cross-boundary directedness score (CBDir) and inner cluster coherence score (ICCoh). The boxes show the interquartile range over the aggregated cells of all celltypes, the notch shows the median, the white point shows the mean, and the whiskers show 1.5× the interquartile range. Note that the whiskers for velocity cosine similarity extend the entire plot due to noise in the simulation. Separately plotting these "incorrectly" predicted cells shows their simulated velocity going opposite the direction of differentiation.

With synthetic data, we show LatentVelo performs much better than scVelo dynamical and stochastic modes in Figure 2. In Supplemental Figure S4 we compare LatentVelo to UniTVelo and DeepVelo with their default settings. We have not done any exploration of the settings of the comparison models to see if there are better settings, so we present this as a supplemental. We find LatentVelo performs similar or slightly better than UniTVelo unified mode and DeepVelo, and much better than UniTVelo independent mode. For LatentVelo and all of the comparison models, hyperparameters and settings are kept at their defaults on these synthetic datasets. We also test our model on increasing/decreasing transcription, splicing, and degradation rates vs time with simulations from the scVelo linear differential equations in Supplemental Figure S7.

In Figure 2 bottom, we evaluate on 10 real datasets with known trajectory directions, and compare with scVelo stochastic and dynamical modes. Our model is robust, and consistently scores well for the CBDir and ICCoh scores, above scVelo. The comparison with the default settings of UniTVelo unified and independent modes and DeepVelo in shown in Supplemental Figure S4, where we find similar or better performance of LatentVelo.

We also evaluate LatentVelo's ability to model the dynamics of separate lineages with the regulatory parameter $\mathbf{z}_r$. We train a simple logistic regression classifier on $\mathbf{z}_r$ to predict separate lineages, obtaining classification accuracy between 0.9 and 1 for all datasets (shown in Supplemental Figure S6). This shows that the lineages are clearly separated and modelled with distinct dynamics.

For these synthetic and real datasets we keep all model hyperparameters at their default values, except (1) the dimension of $\mathbf{z}_r$, and (2) regularization strength that enforces the direction of splicing. The dimension of $\mathbf{z}_r$ is set based on the number of expected lineages, and we set z_r to be the number of expected lineages minus one, except if we only expect 1 lineage where we use 1 dimension. We have found this to be a good heuristic. Regularization strength is set to the default value 0.1 for all synthetic datasets and most real datasets. For the intestinal organoid datasets, scNT-seq dataset and hindbrain (GABA, Glial) datasets, we increased the regularization strength to 10. One simple heuristic that can be used to select this regularization strength is the relative size between the reconstruction terms of the loss and the regularization terms: if the reconstruction term dominates, regularization should be increased. However, more work needs to be done exploring these settings of LatentVelo, as well as the settings of other models.

Note that while we have used 50 principle components to evaluate cell-type transitions because we can do a common benchmark between all methods, a natural embedding to use when evaluating cell-type transitions with LatentVelo is just the latent embedding inferred by the model.

Since all of the parameters of the neural networks in LatentVelo need to be initialized and training is done in mini-batches, there is some stochasticity in training. This is highlighted in Supplemental Figure S8. For most datasets the model is robust to this, however is some cases there is variation and LatentVelo infers incorrect velocities on a fraction of the random seeds. In these cases, the velocity field is reversed. A similar problem was encountered with UniTVelo, and is addressed by initializing the unified latent time with a diffusion pseudotime based on a specified known root celltype, or fitting the model multiple times and selecting the one that is consistent with any prior knowledge [17].

C. Latent space dynamics correct for batch effects in RNA velocity

Batch correction for RNA velocity is currently an unaddressed problem [13]. Since RNA velocity methods operate on gene-space, the best-performing batch correction methods [26] that typically output a latent embedding or corrected nearest neighbor graph will not work with typical RNA velocity methods. Another challenge of correcting batch effects in RNA velocity is due to the need to simultaneously correct unspliced and spliced RNA counts, whereas batch correction methods typically operate on a single data matrix.

Since our model estimates RNA velocity on a latent space, it naturally allows batch correction in a similar way as auto-encoder batch correction methods, such as scVI and scANVI [15, 28]. In Figure 3, we test our models ability to represent dynamics in the latent space independent of batch effects. We use three different simulated datasets from dyngen with strong batch effects. To generate the simulated data with batch effects with dyngen, cell kinetic parameters are sampled from the same distributions in different realizations with the same gene regulatory networks, as suggested in the original dyngen paper [27].

We compare our model with existing batch correction tools for gene-expression (without velocity). Recently a method using traditional batch correction tools that produce a corrected gene count matrix has been suggested for RNA velocity [29, 30]. We compare this method to LatentVelo by using UniTVelo unified time mode combined with batch correction tools for gene counts. We have chosen UniTVelo for this comparison because we found it generally performed quite well with default settings on the synthetic datasets (Supplemental Figure S4) and scVelo performed poorly. We evaluate these methods with metrics of biological cluster conservation (cLISI) and batch correction (kBET and iLISI) [26] for the latent representation of cell-states, as well as with velocity metrics of CBDir and ICCoh and a novel metric of batch corrected velocity (NN velocity cosine), comparing the cosine similarity of velocities for neighboring cells in different batches.

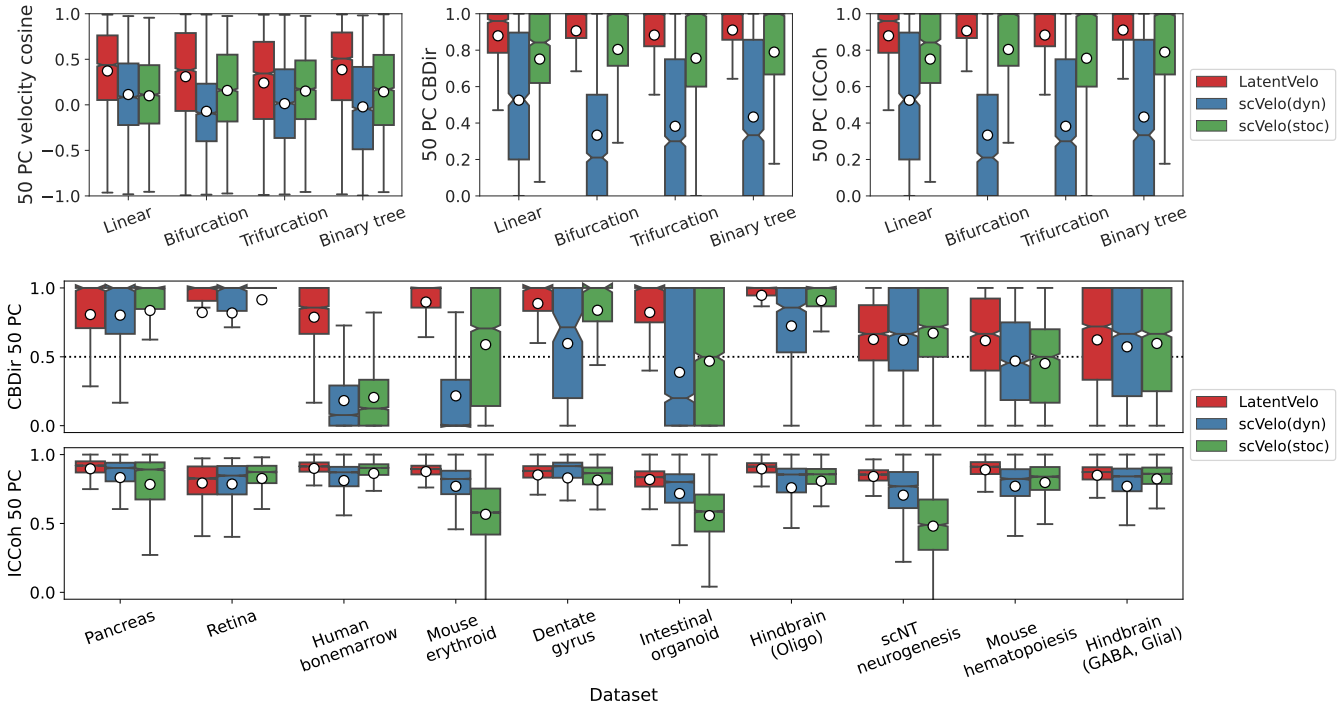

FIG. 2. **Quantitative benchmarking on synthetic and real data. (Top row)** Quantitative metrics for the synthetic datasets. We show velocity cosine similarity, Cross-Boundary Directedness score (CBDir), and Inner-Cluster Coherence score (ICCoh) on the space of 50 principle components (PC). **(Bottom two rows)** Quantitative metrics for real datasets. We show CBDir and ICCoh scores on the space of 50 PC. Higher scores for all metrics are better.

First, we use a simulated dataset with two batches of a bifurcation. In 3a) and b), we show a UMAP representation of the unintegrated data, with the simulation time highlighting the differentiation direction and trajectory milestones used for the CBDir metric. In c), we compare our model with traditional batch correction methods (no velocity) for biological cluster conservation (cLISI) and batch correction (kBET, iLISI) of the latent states. Our model is the only method scoring highly on all metrics, even when compared to these methods specifically designed for batch correction of gene expression. In d), we evaluate batch correction of velocity, and compare with UniTVelo on the unintegrated data or using a previously developed method for RNA velocity (see methods) with batch correction methods that output corrected gene-expression matrices (ComBat [31], scGen [32]) [29, 30]. These methods fail to properly integrate the batches and have worse performance on velocity metrics than our model. In Figure 3e), we show the UMAP plot of the spliced latent state for LatentVelo with celltype annotations (LatentVelo+annot). The model latent space accurately integrates the two batches and RNA velocity accurately follows the direction of differentiation. Both batches are clearly visually integrated in addition to the quantitative metrics. Observing the z_r regulatory parameter at $\hat{t}$, we see that the model accurately describes the lineage dynamics with distinct values of z_r on each lineage, despite the two batches. Additionally, we note that adding celltype annotations to the latent space improves batch correction.

In Figure 3f) and g), we show simulated data for three batches of a linear differentiation trajectory. In Figure 3h), we show batch correction metrics and in i), the velocity metrics. Similar to above, we see our approach is the only one to accurately both integrate the batches and model RNA velocity.

In Figure 3k) and l), we demonstrate our approach on two simulated batches of a bifurcation, but the batches are now sampled at different points in time, leading to the left batch having more points at earlier times and the right batch having more points at later times. We do not give our approach any information about the sample times of the batches, and despite this, the model still is able to integrate the batches and model RNA velocity. Note we do not show the iLISI metric for batch correction here since the batches have different distributions of celltypes and this metric does not condition on cell-types .

In each of these simulated datasets, our model is the only model consistently performing strongly for both batch correction and velocity estimation. We also see that adding annotated cell-type information to the model is useful to improve batch correction. Additionally, our model performs even better at batch correction of gene expression than the batch correction methods scVI, scANVI, ComBat, and scGen(gene). LatentVelo is able to achieve this

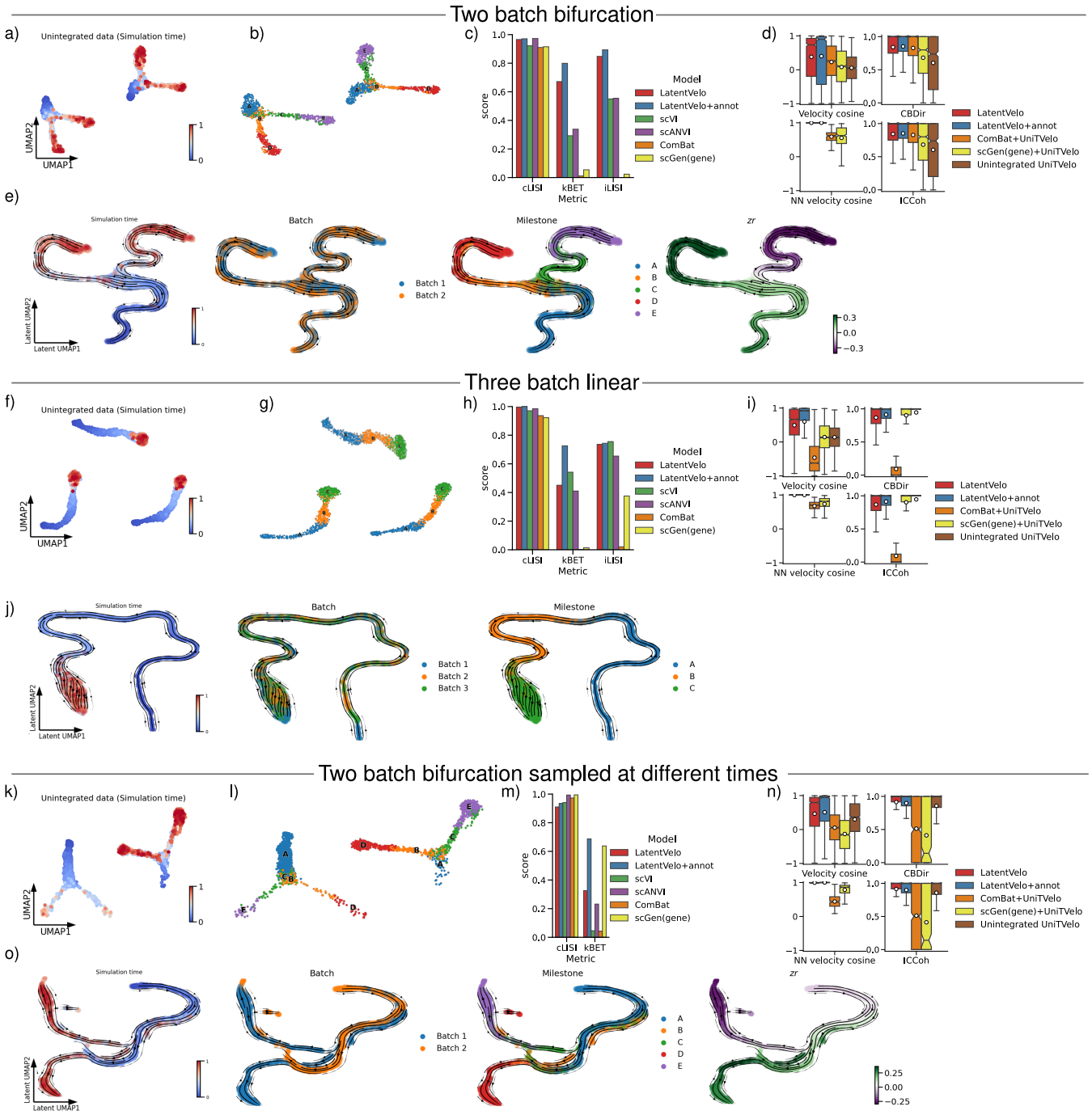

FIG. 3. **Batch effect correction of RNA velocity with LatentVelo. a)** Unintegrated data simulated with dyngen. Two batches with a bifurcation are shown. Simulated time shows the direction of differentiation. **b)** Highlighting the "milestone" cells in the data, which define the direction tested with CBDir scores. **c)** Batch correction (kBET and iLISI) and biological cluster conservation (cLISI) scores for the models. Higher scores are better. **d)** Velocity scores for the models. Nearest neighbor (NN) velocity scores measure batch correction, and the other scores measure velocity accuracy. Higher scores are better. **e)** UMAP of LatentVelo with celltype annotations (LatentVelo+annot) showing simulation time, batch ID, milestone annotations, and regulatory z_r of spliced latent states at $\hat{t}$. Visually, the data are well integrated. **f)** Unintegrated data simulated with dyngen. Three batches with a linear trajectory are shown. Simulated time shows the direction of differentiation. **g)** Highlighting the milestone cells. **h)** Batch correction metrics. Higher scores are better. **i)** Velocity metrics. **j)** UMAP representation of our LatentVelo+annot spliced latent states. Batches are visually well integrated and follow the correct differentiation trajectory. **k)** Simulated dyngen data containing a bifurcation with two batches sampled at different times. **l)** Highlighting milestone cells. **m)** Batch correction metrics **n)** Velocity metrics. **o)** UMAP representation of LatentVelo+annot spliced latent states.

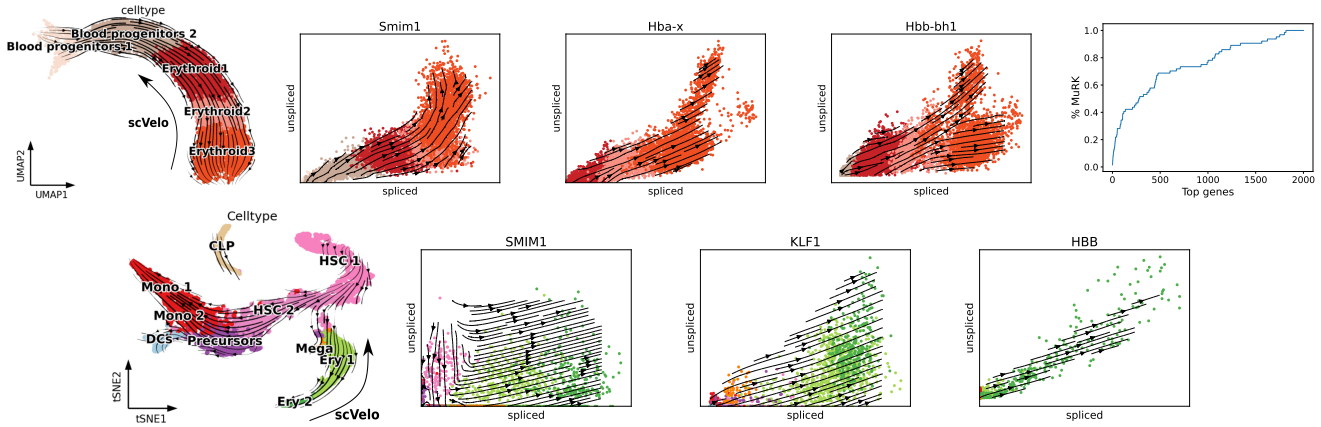

FIG. 4. **Corrected modelled of MuRK genes in erythroid development. (Top row)** UMAP plot a mouse erythoid development dataset. LatentVelo velocity arrows on shown on the plot, and the inferred scVelo direction is shown. We show unspliced vs spliced plots of three genes showing a boost in transcription between Erythroid 2 and Erythroid 3 cells, demonstrating that our model correctly models the velocity of these genes. We also show the many of the best predicted genes in our model are MuRK genes. **(Bottom row)** t-SNE plot of a human bonemarrow dataset. LatentVelo velocity arrows are shown on the plot, and the inferred scVelo direction for the erythroid lineage is shown. We show three genes showing a boost in transcription on the erythroid lineage, demonstrating that our model correctly models the velocity of these genes.

better performance for batch correction by utilizing the splicing dynamics. All model hyperparameters remain at their defaults for these batch correction tests. We also use default parameters for scVI, scANVI, scGen, ComBat, and UniTVelo.

D. Corrected modelling of MuRK genes in erythroid development

Previous work has identified Multiple Regime Kinetic genes (MuRK) that show a transcription boost during erythroid development, resulting in a large increase in unspliced RNA [12, 13]. This results in scVelo dynamical and stochastic modes failing to identify the correct velocity direction. This is due to the dynamics being poorly captured by the linear differential equations of scVelo. We use two datasets containing an erythroid lineage to demonstrate LatentVelo's ability to model dynamics with a transcriptional boost. While scVelo fails on these datasets, recent models such as VeloAE, UniTVelo, and VeloVAE have been previously demonstrated to correctly model a transcriptional boost [16, 17, 19].

First, we use a mouse erythroid development dataset [33]. In the top row of Figure 4, we show that our approach captures the correct differentiation direction from blood progenitors to Erythroid cells, and contrast this with the reversed direction inferred by scVelo.

Second, we would like to show that even though LatentVelo is modelling a low-dimensional representation of RNA velocity, and not the high-dimensional gene space, we can still infer the dynamics of key critical genes. Unspliced-spliced RNA plots are shown for some of the MuRK genes best reconstructed by the VAE (in terms of R^2 score), Smim1, Hba-x, and Hbb-bh2. We compute the velocity of single genes in LatentVelo by transforming the latent space velocities into gene-space velocities by utilizing the decoder: $\dot{\mathbf{s}} = J[D_s](\mathbf{z}_s)\mathbf{f}_s(\mathbf{z}_s, \mathbf{z}_u)$, where $J[D_s](\mathbf{z}_s)$ is the Jacobian of the spliced decoder evaluated at $\mathbf{z}_s$ and $\mathbf{f_s}$ is the latent space velocity. These MuRK genes show a transcriptional boost during the transition between Erythroid 2 and Erythroid 3. In comparison to scVelo where a reversed velocity direction is seen [12], our approach correctly models the velocity direction for these genes.

MuRK genes previously identified in this dataset contain archetypal red blood cell genes essential for red blood cell function [12]. We demonstrate that LatentVelo utilizes these critically relevant genes by showing the percent of top reconstructed genes in terms of R^2 score that are MuRK genes. This shows that even though our model latent space is compressing the 2000 highly variable genes to 20 dimensions, it is still capturing these critical genes.

Note that since this dataset includes 3 batches, we include the batch ID in the encoder and decoder for batch correction with LatentVelo. This is most clearly seen in the Hbb-bh1 gene unspliced vs spliced plot, where two separate blobs of Erythroid 3 cells (dark orange) are seen due to belonging to separate batches. Due to our model including batch correction, when plotting these raw unintegrated counts we see the model accurately predicts different velocities on each of the batches.

We also test our method on a human bonemarrow dataset [34]. Similar to the mouse erythroid dataset, scVelo

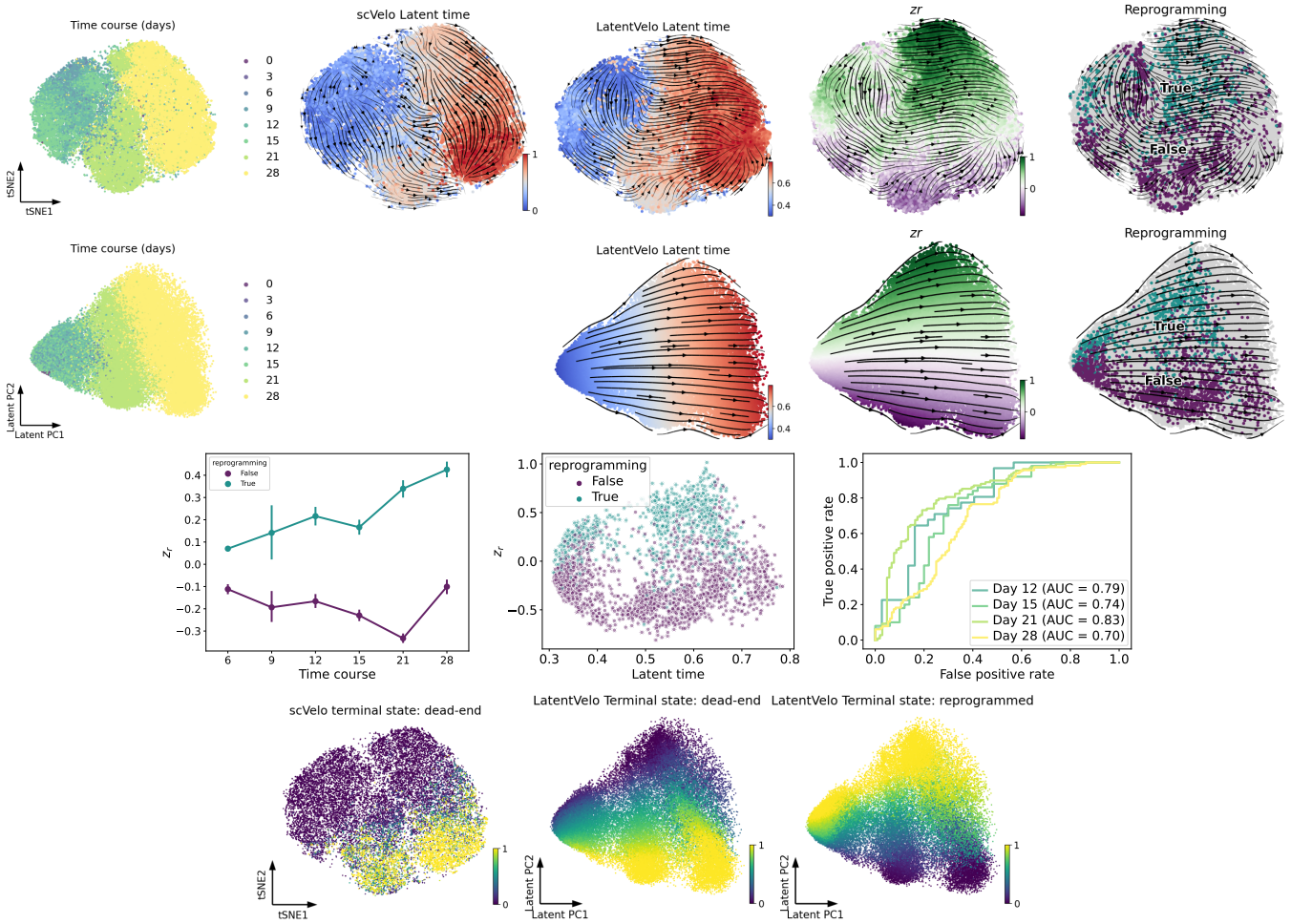

FIG. 5. **Trajectory-dependent dynamics in reprogramming of mouse fibroblasts. (Top row)** t-SNE representation of mouse embryonic reprogramming data [35]. We color this plot by experiment time course (left). We show scVelo velocity and latent time and LatentVelo velocity and latent time. We also show LatentVelo velocity with regulatory parameter z_r and reprogramming outcome (turquoise reprogrammed, purple dead-end) from the experiment. The regulation parameter identifies reprogramming, describing different dynamics for different values of z_r. **(Second row)** We compute a PC representation of the latent spliced latent state $\mathbf{z}_s$, and color by the same variables. On the learned latent state, dynamics are clearly separated by reprogramming outcome. **(Third row)** We show that reprogramming outcome is strongly separated by z_r, including a ROC curve showing z_r is predictive of reprogramming outcome. **(Bottom row)** We also show the tSNE embedding colored by the predicted fate probabilities by CellRank [4] using the scVelo velocities, and latent PC embedding colored by the predicted fate probabilities by CellRank using the LatentVelo velocity estimates. scVelo only infers a terminate state at the dead-end cells, while LatentVelo infers terminal states for reprogrammed and dead-end cells.

has many problems on this dataset, in particular showing incorrect differentiation direction on the erythroid lineage and showing differentiation towards hematopoetic stem cells (HSC). In the bottom row of Figure 4 we show that our method accurately models the direction of differentiation, from HSC to the erythroid lineage, and to the monocyte (Mono 1,2) and dendridic (DCs) lineages. We also highlight some example genes, where our model captures the transcriptional boost behaviour of the SMIM1, KLF1, and HBB genes on the erythroid lineage.

The more general case of time-dependent rates is explored in Supplemental Figure S7. We show that our model is robust to increasing/decreasing transcription, splicing, and degradation rates vs time with simulations.

E. LatentVelo infers separate trajectories for mouse embryonic fibroblast reprogramming

To demonstrate LatentVelo's ability to learn biologically meaningful features in its latent representation, we use a dataset of mouse embryonic fibroblasts (MEF) reprogramming toward induced endoderm progenitors (iEP) [35]. In

this experiment, over-expression of key transcription factors drives fibroblasts to potentially undergo reprogramming, or develop into a dead-end state. On this dataset, scVelo incorrectly shows differentiation from reprogrammed to dead-end cells, and shows no route to reprogramming [4].

In the top row of Figure 5, we show a tSNE plot of the data. We show the velocity estimated by scVelo and LatentVelo overlayed with their respective inferred latent times. We also show colors showing experimental time course, LatentVelo regulatory parameter z_r, and reprogramming outcome (purple dead-end, green reprogrammed). LatentVelo estimates latent time that correlates with the experimental time-course, and velocity that indicates transitions to both dead-end and reprogrammed cells. In contrast, scVelo shows differentation of reprogrammed cells towards dead-end cells.

Observing the regulator parameter z_r, we see the LatentVelo infers two distinct regimes of dynamics: positive z_r where cells undergo reprogramming, and negative z_r of dead-end cells that do not. This highlights that the regulation parameter z_r can describe different "modes" of the dynamics, and here identifies the differences in reprogrammed vs dead-end cells.

Reprogramming takes a complex path through the data. However in contrast, it becomes clearly apparent in the PC plot of the latent space inferred by LatentVelo shown in the middle row. Here, we see the latent space cleanly separates the reprogrammed/dead end cells, generating a latent representation informative of the reprogramming dynamics. This shows that the latent representation of the dynamics inferred by LatentVelo is biologically meaningful. In the Supplemental Figure S6, we show in the pancreas, retina, bone marrow, hematopoiesis, intestinal organoid, and hindbrain datasets that in general, the regulatory parameter $\mathbf{z}_r$ clearly identifies lineages in this same way, highlighting lineage-dependent dynamics that are not modelled in some other RNA velocity approaches like scVelo.

On the third row, we show the distribution of z_r for reprogrammed/dead-end cells, and z_r vs latent time, showing a clear separation. We also train a logistic regression classifier on z_r (effectively just scaling z_r to represent a probability), and show a ROC curve, demonstrating that it is predictive of reprogrammed cells. This shows that the parameter z_r inferred by the model corresponds to a meaningful biological feature: the reprogramming outcome.

From just visualizing the embedding in 2 dimensions, it cannot be determined that LatentVelo infers terminal states corresponding to reprogrammed and dead-end cells. We use CellRank [4] to analyze the estimated velocities. We use the velocity kernel, which computes a transition matrix based on the estimated velocities. CellRank finds two terminal states, one corresponding to reprogrammed cells and one corresponding to dead-end cells. On the bottom right of Figure 5, we show the probability of transitioning to each of the terminal states. Using velocity estimated from scVelo, only a single terminal state is obtained, corresponding to dead-end cells. This is due to scVelo incorrectly inferring transitions from reprogrammed cells to dead-end cells [4].

F. LatentVelo infers velocity in large multi-lineage systems

In Figure 5 and Supplemental Figure S6, we demonstrated that LatentVelo can learn dynamics in datasets with multiple lineages. However, these datasets had at most 3-4 lineages. We now test LatentVelo on a large-scale many-lineage system.

In Figure 6 we show a dataset of mouse gastrulation, showing development from pluripotent epiblast cells into the ectodermal, mesodermal, and endodermal progenitors of major organs [33]. Note, this is the full dataset from where the erythroid cells from Figure 4 come from (seen in the upper right of the UMAP plot). On this large scale dataset, we have increased the dimension of the latent $\mathbf{z}_s$ and $\mathbf{z}_u$ states to 70 and used a celltype-dependent version of the correlation regularization (see methods) with the celltype annotated version of the model. We highlight the 7-dimensional $\mathbf{z}_r$ space in the middle row of Figure 6, showing that the dynamics of different terminal celltypes are distinct.

Putative developmental directions can be seen by observing the later time point cells (E8.5), and our model's velocity and latent time estimates generally agree with these time points. We also highlight some CBDir transition scores between celltypes, showing the model infers transitions from epiblast to mesoderm, the development of epithelial and erythroid cells, and the development of the gut from endoderm cells. One particular transition that we don't see is the convergence of visceral endoderm cells and definitive endoderm cells to gut cells, we only see the transition from definitive endoderm cells to gut cells.

This dataset demonstrates LatentVelo's ability to learn dynamics on large multi-lineage systems with many different cell types.

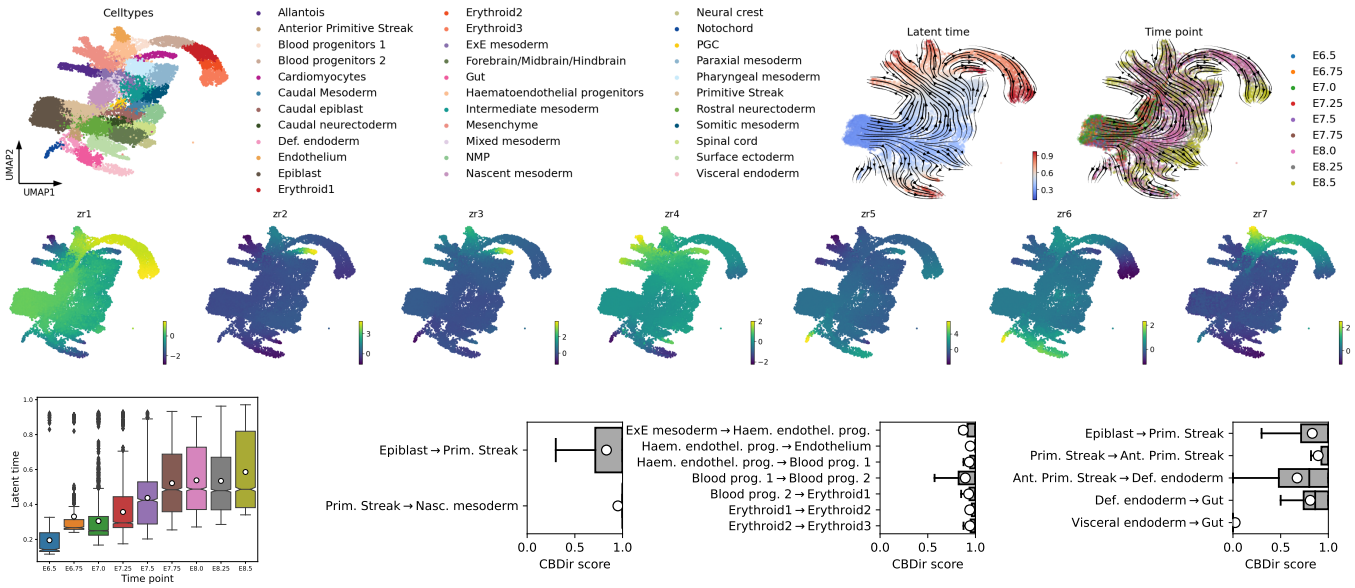

FIG. 6. **Inferring multiple-lineages in mouse gastrulation. (Top row)** UMAP plots showing celltypes during gastrulation, and Latent time and experimental time points with inferred RNA velocity. **(Middle row)** each of the 7 components of $\mathbf{z}_r$ overlayed on the UMAP plot. We can see specific terminal cell-types highlighted in each component. **(Bottom row)** Association between latent time and experimental time, and some CBDir celltype transition scores estimated on the inferred latent space.

IV. DISCUSSION

We have developed LatentVelo, a new model of latent cell developmental dynamics. We have demonstrated that our model is accurate and robust on a wide variety of synthetic and real datasets, outperforming the currently most widely used approach, scVelo [11]. LatentVelo is the first method to batch correct RNA velocity by inferring velocity on a latent embedding, and additionally performs better at batch correction of gene expression than other methods that only attempt to batch correct gene expression and ignore dynamics. LatentVelo can infer distinct RNA velocity dynamics on separate lineages by learning a latent regulatory parameter. We also show that the latent embedding and the latent regulatory parameter can represent biologically meaningful features.

Previous RNA velocity methods have inferred velocity on gene-space, with the only exception being VeloAE, which did not model dynamics [16]. LatentVelo embeds gene expression into a latent space, with the time-progression of gene expression described by latent dynamics. This allows us to infer dynamics-informed embeddings of gene expression, which is a dimensional reduction of the dynamics of the system. Shown with the fibroblast reprogramming dataset [35], the 2 dimensional PC representation of this latent space allows clear visualization of the separate trajectories taken by reprogrammed or dead-end cells. This opens up a new method of studying and characterizing the dynamics of cell differentiation, through the features represented in this latent space. We have left exploring the further implications of this to future work.

LatentVelo's main application is describing complex developmental dynamics in a low-dimensional latent space. Despite modelling dynamics in this low-dimensional latent space, LatentVelo can still model the dynamics of single genes. In Figure 4 we showed that LatentVelo can model the dynamics of MuRK genes in erythroid development. Additionally in Supplemental Figure S5, we show that LatentVelo can model a diverse variety of gene dynamics in pancreatic endocrinogenesis.

Interestingly, we have found scVelo stochastic mode, which computes RNA velocity based on a linear regression between unspliced and spliced RNA at an assumed stead-state, performed much better than scVelo dynamical mode. We believe this highlights the issues with the simple linear differential equation with constant rates approach of scVelo dynamical mode. The recent methods DeepVelo [18], UniTVelo [17], VeloVAE [19], and our approach LatentVelo all have different approaches to addressing this issue.

Since LatentVelo is a variational auto-encoder, we can sample the latent space to generate uncertainty estimates of latent times and velocities. This is shown in Supplemental Figure S2, where we show uncertainty is largest near branching between multiple cell-types. This method of uncertainty estimation is distinct from the method of scVelo [11], which used the consistency of the velocity estimates of neighboring cells.

LatentVelo can be easily extended to multi-omics. In the Supplemental Information, we show an example of this with combined scRNA-seq and ATAC-seq. The extension is done by adding a new variable to the structured latent dynamics $\mathbf{z}_c$, corresponding to chromatin accessibility, and modelling the regulation of chromatin by $\mathbf{z}_r$ and the regulation of transcription by $\mathbf{z}_c$ and $\mathbf{z}_r$. This demonstrates a key feature of LatentVelo: the ability to build *general* structured latent dynamics. This is presented as an example, we leave the more detailed analysis of multi-omic systems and exploration of other forms of structure in the dynamics to future work.

LatentVelo addresses many of the challenges raised in a recent review paper of RNA velocity [13]: multi-modal omics (see Supplemental Information), multi-variate dynamics, batch correction, lineage/time dependent rates, and implicit gene selection by embedding in the latent space. Two challenges not addressed are (1) stochastic dynamics, and (2) normalization. We see potential ways to address these challenges with simple modifications to our model: (1) for stochastic dynamics we can replace the latent ODEs in our model with SDEs [36], and (2) we can approach normalization in a similar way as scVI [28]; including normalization factors as latent variables to infer. Modelling SDEs instead of ODEs is a clear next direction for LatentVelo, in particular by using a Bayesian approach to inferring the latent SDE [36], we can simultaneously learn the prior SDE describing the full population of cells as well as the conditional SDE describing the dynamics of each cell conditioned on a particular branch (as we have done with the parameter $\mathbf{h}$). This would allow the use of the model for perturbed inputs – estimating future developmental trajectories for individual cells.

There are limitations to the evaluation of RNA velocity on real datasets. We have used a modified form of the CBDir metric [16, 17], which quantifies the probability of each cell transitioning to the expected celltype. A substantial limitation of this approach is that it relies on accurate discrete annotations of cell-types. Further development of quantitative metrics for RNA velocity needs to be done.

While we have shown LatentVelo is accurate on many datasets, for some it requires additional hyperparameter adjustment in terms of regularization strength. Indeed, all of the RNA velocity methods we compared with also have multiple settings and hyperparameters, in addition to preprocessing steps [14]. More work needs to be done in this area.

V. METHODS

A. LatentVelo

We embed input spliced and unspliced data into a latent space $\hat{\mathbf{z}}$ with an encoder neural network. A separate encoder is used for spliced and unspliced counts, partitioning the latent space as $\hat{\mathbf{z}} = (\hat{\mathbf{z}}_s, \hat{\mathbf{z}}_u)$. Our model is trained as a Variational Autoencoder (VAE), where we use a prior on the latent space $\hat{\mathbf{z}} \sim \text{Normal}(0, 1)$. We also use an encoder to estimate a latent developmental time with a logit-normal prior $t \sim \text{LogitNormal}(0, 1)$.

Dynamics on the latent space are described by neural ordinary differential equations,

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{f}(\mathbf{z}(t)), \tag{10}$$

where $\mathbf{f}$ is a neural network describing the velocity field of the dynamics. The spliced component of this velocity field represents latent RNA velocity. The structure of $\mathbf{f}$ is as described in 1.

These dynamics are coupled to the auto-encoder by matching the ODE solution $\mathbf{z}(\hat{t})$ at time $\hat{t}$ with the encoded latent state $\hat{\mathbf{z}}$.

We use an approximate posterior factorized as $q(\hat{\mathbf{z}}, \hat{t}|\mathbf{x}) = q(\hat{t}|\hat{\mathbf{z}})q(\hat{\mathbf{z}}|\mathbf{x})$, where $\mathbf{x} = (\mathbf{s}, \mathbf{u})$ are the observed spliced and unspliced counts. Separate encoders for spliced and unspliced latent states are used, such that $q(\hat{\mathbf{z}}|\mathbf{x}) = q(\hat{\mathbf{z}}_s|\mathbf{s})q(\hat{\mathbf{z}}_u|\mathbf{u})$. Additionally, we also use separate decoders, $p(\mathbf{s}, \mathbf{u}|\mathbf{z}) = p(\mathbf{s}|\mathbf{z})p(\mathbf{u}|\mathbf{z})$. Since the data are very noisy, we find that we cannot use a count-based distribution such as a negative-binomial, so instead use a Gaussian with mean functions $\mu(\mathbf{z}) = (\mu(\mathbf{z}_s), \mu(\mathbf{z}_u))$ with smoothed and normalized counts following the scVelo preprocessing procedure [11].

Our model is trained with the loss:

$$L = -\mathbb{E}_{\hat{\mathbf{z}},\hat{t}\sim q}\big[\log \text{Normal}(\mathbf{x}|\mu(\hat{\mathbf{z}}), \sigma)\big] + \text{KL}(q||p) - \mathbb{E}_{\hat{\mathbf{z}},\hat{t}\sim q}\big[\log \text{Normal}(\mathbf{x}|\mu(\mathbf{z}(t)), \sigma) - \log \text{Normal}(\hat{\mathbf{z}}|\mathbf{z}(t), \sigma_z)\big]. \tag{11}$$

Expectations are computed by sampling $\hat{\mathbf{z}}$ and $\hat{t}$ from the encoder.

The first two terms of this loss are the standard negative evidence lower bound of a VAE [23, 24], representing the expectation of the negative log-likelihood with a Gaussian, and the KL divergence between the posterior $q(\hat{\mathbf{z}}, \hat{t}|\mathbf{x})$ and the prior, $\text{Normal}(0, 1) \times \text{LogitNormal}(0, 1)$. The two remaining terms include a loss for the decoded solution of the dynamics $\mathbf{z}(\hat{t})$, and a term penalizing the distance between the encoded latent state $\hat{\mathbf{z}}$ and the latent state estimated by the dynamics $\mathbf{z}(\hat{t})$.

By default, we use a 20-dimensional latent state for each observed component (spliced, unspliced) and use ELU activations throughout. We follow VeloAE [16] and use a encoder structured as an initial fully connected dense neural network with 1 hidden layer (of size 25 by default) as the first part, and then use two graph convolution layers using a 30 nearest-neighbor graph computed on the 50 principle components (same as was used for smoothing) as the similarity graph for the second part. This enables the encoder to use information about nearest neighbors when computing the latent embedding. In the case of batch correction, we do not use the graph convolution layers and instead just use a fully connected dense neural network with 2 hidden layers.

The encoders for $\mathbf{h}$ and $\hat{t}$ are also graph convolutional networks with 2 layers of graph convolutions. The differential equation derivative functions $\mathbf{f}_u$ and $\mathbf{f}_r$ have a single hidden layer of size 25. $\mathbf{f}_s$ is linear in $\mathbf{z}_s$ and $\mathbf{z}_u$. We also use a linear decoder, and use separate linear decoders per batch for batch correction. The dimension of $\mathbf{z}_r$ is chosen based on the expected number of branches. For one or two expected branches, we use 1 dimensions, for 3 expected branches we use 2 dimensions. In general, we take the dimension of $\mathbf{z}_r$ to be 1 less than the number of expected branches.

We use 90% of the data for training, and use the other 10% as a validation set to monitor training progress. We train for 50 epochs, and use the model at the epoch with lowest mean-squared error on the validation set. By default we train with the Adam optimizer with a learning rate of 0.01 and a batch size of 100. In large datasets (e.g. mouse gastrulation or Fibroblast reprogramming), we increase the batch size to 1000. In scenarios with exploding gradients, we use gradient clipping to stabilize training. We increase the KL divergence weight in the VAE from 0 to 1 linearly over the first 25 epochs.

The model is implemented in *pytorch* utilizing the *torchdiffeq* package for neural ODEs [25, 37], which critically allows computing gradients of the inferred latent time-points $\hat{t}$ in the solver, rather than marginalizing over $\hat{t}$ by integrating. We found the marginalization approach to be challenging.

B. Enforcing splicing direction

The linear ODEs describing transcription, splicing, and degradation are,

$$\dot{u} = \alpha(t) - \beta u \tag{12}$$

$$\dot{s} = \beta u - \gamma s \tag{13}$$

per gene. These equations enforce the casual relationship of splicing between unspliced and spliced RNA. Generalizing, this same effect can be achieved with $\frac{\partial \dot{s}_g}{\partial u_g} > 0$ and $\frac{\partial \dot{s}_g}{\partial s_g} < 0$.

To constrain the direction of splicing to accurately represent the biology (unspliced to spliced), we need to enforce $\frac{\partial \dot{s}_g}{\partial u_g} > 0$ and $\frac{\partial \dot{s}_g}{\partial s_g} < 0$ for each gene g. Additionally, we also constrain $\frac{\partial \dot{u}_g}{\partial u_g} < 0$. However, enforcing this constraint for general forms of the velocity $\dot{s}$ is difficult, since we need to compute the Jacobian, which is a matrix of size $2N^2$, where N is the number of genes. Computing this would require back-propagation through the entire model back to the input, which is computationally infeasible. Instead, we can use a much faster approach to enforce this direction by using the correlation between velocity and spliced and unspliced counts.

We weakly regularize by the correlation between gene-space velocity and input data,

$$\lambda_{su}\text{corr}(\dot{\mathbf{s}}, \mathbf{u}) + \lambda_{ss}\text{corr}(\dot{\mathbf{s}}, -\mathbf{s}) + \lambda_{uu}\text{corr}(\dot{\mathbf{u}}, -\mathbf{u}). \tag{14}$$

By default we take $\lambda_{su} = \lambda_{su} = \lambda_{uu} = 0.1$. The goal with this term is to just weakly regularize the direction, rather than match the strict linear dependence seen in other models. To compute gene-space velocity we compute the time-derivative of the transformation with the decoder, $\dot{\mathbf{s}} = J[\mu_s](\mathbf{z}_s)\dot{\mathbf{z}}_s$, where $J[\mu_s]$ is the Jacobian of the spliced decoder. Similarly we compute the unspliced gene-space velocity, $\dot{\mathbf{u}} = J[\mu_u](\mathbf{z}_u)\dot{\mathbf{z}}_u$. Note, the Jacobian here is taken with respect to the latent space which has a smaller dimension, and this is a Jacobian vector product, which are easily computable by backpropagation in comparison to the Jacobian discussed above. This regularization can be done per celltype, only including the same type cells in the correlation.

We only apply this regularization to genes that show significant splicing dynamics. These "velocity genes" are identified in a similar way to UniTVelo and scVelo. We fit a linear regression between spliced and unspliced data per gene, then only select genes with R^2 score above 0.05 and below 0.95. In cases of genes with very high R^2, all cells lie on a straight line in the u vs s plane, showing no splicing dynamics. In cases of genes with very low R^2, cells are scattered uniformly in the u vs s plane, showing no splicing dynamics.

C. Incorporating cell-type annotations

We follow scANVI [15], and incorporate cell-type annotations by modifying the prior. We introduce the new latent state $\hat{\mathbf{w}}$, and use the new approximate posterior,

$$q(\hat{\mathbf{z}}, \hat{\mathbf{w}}, \hat{t}|\mathbf{x}, c) = q(\hat{\mathbf{z}}|\mathbf{x})q(\hat{\mathbf{w}}|c, \hat{\mathbf{z}})q(\hat{\mathbf{t}}|\hat{\mathbf{z}}). \tag{15}$$

We place a standard normal prior on $\mathbf{w}$, and model $\hat{\mathbf{z}}$ and $\hat{t}$ as functions of $\mathbf{w}$ and the cell-type annotations c instead of placing a standard normal/logitnormal priors; $p(\hat{\mathbf{z}}|\hat{\mathbf{w}}, c) = \text{Normal}(f_{\mu,z}(\hat{\mathbf{w}}, c), f_{\sigma,z}(\hat{\mathbf{w}}, c))$ and $p(\hat{t}|\hat{\mathbf{w}}, c) = \text{LogitNormal}(f_{\mu,t}(\hat{\mathbf{w}}, c), f_{\sigma,t}(\hat{\mathbf{w}}, c))$.

The effect of this modification is to allow the priors on $\hat{\mathbf{z}}$ and $\hat{t}$ to vary with cell-type, preventing over-regularizing and eliminating biologically relevant cell-type clusters for the sake of matching the restrictive prior.

D. RNA velocity and latent time uncertainty

To estimate uncertainty in latent time, we sample from the inferred posterior distribution $q(\hat{t}|\hat{\mathbf{z}}_u, \hat{\mathbf{z}}_s)$ and compute the standard error of the mean.

To estimate uncertainty in latent RNA velocity, we sample from the inferred posterior distribution of states $q(\hat{\mathbf{z}}_s, \hat{\mathbf{z}}_u|\mathbf{s}, \mathbf{u})$ and compute latent RNA velocity for each sample $\mathbf{v} = \mathbf{f}_s(\hat{\mathbf{z}}_s, \hat{\mathbf{z}}_u)$. We then compute the average cosine similarity of all pairs from these samples as the consistency of velocity,

$$c_v = \mathbb{E}_{i>j \text{ samples}}\big[\cos(\mathbf{v}_i, \mathbf{v}_j)\big]. \tag{16}$$

E. RNA velocity metrics

When ground truth velocities are known (synthetic data), we compare estimated velocities by computing the cosine similarity on a variety of different lower-dimensional embeddings.

When ground truth velocities are not known, we use known cell-type transitions with the Cross-Boundary Directedness metric [16, 17].

$$\text{CBDir}(A \to B, \text{cell } i) = \frac{1}{B \cap \mathcal{N}(i)} \sum_{j \in B \cap \mathcal{N}(i)} \delta\big[\cos(\mathbf{v}_i, \mathbf{s}_{i,j})\big], \tag{17}$$

where $\mathbf{s}_{i,j} = (\mathbf{s}_i - \mathbf{s}_j)/\text{sign}(\mathbf{s}_i - \mathbf{s}_j)$ and $\mathcal{N}(i)$ is the neighborhood of cell i.

We also use the In-Cluster Coherence (ICCoh) metric [16, 17] to evaluate the coherence of velocities within a cluster or cell-type. This score is computed per cell by the average cosine similarity of neighboring cells in the same cluster/celltype,

$$\text{ICCoh}(A, \text{cell } i) = \frac{1}{A \cap \mathcal{N}(i)} \sum_{j \in A \cap \mathcal{N}(i)} \cos(\mathbf{v}_i, \mathbf{v}_j). \tag{18}$$

F. Batch correction metrics

To evaluate batch correction we use the kBET and iLISI metrics [26]. We also use the cLISI metric to evaluate biological cluster conservation [26], ensuring that cell-type clusters are not erased by the batch correction. These metrics are computed with the *scib* package [26].

kBET evaluates the batch composition of the nearest neighbors of a cell, which should match the overall batch composition for a particular cell-type for good batch correction. iLISI and cLISI measure the nearest-neighbor graph structure, evaluating batch mixing (iLISI) or cell-type separation (cLISI).

To evaluate batch correction of RNA velocity, we measure the cosine similarity of nearest neighbor cells in different batches.

G. ComBat and scGen RNA velocity batch effect correction

We follow the approach taken by Hansen and Ranek *et al.* [29, 30] to compare our approach for batch effect correction of RNA velocity. Since traditional RNA velocity methods require cells in gene-space, only batch correction methods that return a corrected gene matrix can be used. Here we use ComBat [31] and scGen (using the corrected gene output) [32]. ComBat is run from the *scanpy* package [38].

Since we need to simultaneously correct spliced and unspliced counts, batch correction is performed on the sum of these counts. We define the sum matrix M and the ratio matrix R,

$$M = S + U, \tag{19}$$

$$R = \frac{S}{S+U}. \tag{20}$$

Batch correction is performed on M. For ComBat, we first $\log(1+x)$ transform this matrix. For scGen, we normalize then $\log(1+x)$ transform.

After batch correcting M to get the corrected matrix $\tilde{M}$, we invert these transforms and then multiply R or $1-R$ to recover corrected spliced and unspliced matrices.

$$S_{\text{corrected}} = \tilde{M}R, \tag{21}$$

$$U_{\text{corrected}} = \tilde{M}(1-R). \tag{22}$$

Then RNA velocity is estimated as before with the same pre-processing steps.

scGen is run with default settings, except we set the dimension of the latent state to be the same as our model, 20.

H. scVI and scANVI batch correction

scVI and scANVI are ran from the scvi-tools package [39]. We run scVI with a negative binomial gene-likelihood, and the same latent dimension as our model, 20. scANVI is trained by starting with the pre-trained scVI model and training for an additional 20 epochs.

I. Comparison velocity models

scVelo stochastic and dynamical modes are run with default settings. DeepVelo is run with mask_zero = False, and correlation regularization set with autoset_coeff_s. UniTVelo is run with the defaults IROOT = $None$ and R2_ADJUST = $True$.

When we compute the transition matrix to embed RNA velocity in 50 dimensional PC space or 2 dimensional UMAP or tSNE for plotting, we use scVelo's function scvelo.tl.velocity_graph with default settings.

J. Datasets

For each dataset we select genes with at least 20 cells with non-zero unspliced and spliced counts, and from these genes select the top 2000 highly variable genes using scVelo preprocessing. We apply the transformation $log(1+x)$ before computing principle components and smooth data by average over the 30 nearest neighbors in 30 dimensional principle component space with the scVelo function scvelo.pp.moments. Input variables to the model are then scaled to standard deviation 1.

Dyngen synthetic datasets. We generate synthetic datasets with dyngen [27]. For each dataset in Figure 2, we simulate 5000 cells. We set 15 target genes and 15 housekeeping genes, and use the number of transcription factors required for the trajectory backbone. We use the backbones "linear", "bifurcating", "binary tree" with 2 modifications, and "trifurcating". We set $\tau = 0.01$, census_interval=1, and use 100 simulations. For the CBDir metric, we use the milestones defined by dyngen as transitions. We set the dimension of $\mathbf{z}_r$ to be 1 on the linear and bifurcation datasets, and 2 on the trifurcation and binary tree datasets.

scVelo linear model synthetic datasets. We use the scVelo linear differential equations to simulate datasets with time-varying alpha (transcription rate), beta (splicing rate), gamma (degradation rate). We simulate 500 cells

with 30 genes. By default, we set $\alpha = 5$, $\beta = 0.5$, $\gamma = 0.5$. For time-varying rates, we select 5 genes to be time-dependent. We set the dimension of $\mathbf{z}_r$ to be 1 on these datasets.

Pancreatic endocrinogenesis. Mouse pancreatic cells sampled at E15.5 [40]. In Figure 2 Initially cycling population is removed to focus on differentiation into terminal cell types. This dataset is downloaded from the CellRank package [4]. The transitions tested with CBDir are (Ngn3 low EP → Ngn3 high EP), (Ngn3 high EP → Fev+), (Fev+ → Delta), (Fev+ → Beta), (Fev+ → Epsilon), (Fev+ → Alpha). We set the dimension of $\mathbf{z}_r$ to be 2 on this dataset. In Supplemental Figure S5 we use the full dataset downloaded with scVelo [11].

Mouse hematopoiesis. Data is from [41], and processed data is downloaded from `https://zenodo.org/record/6110279` [29]. The transitions tested with CBDir are (LTHSC → MPP), (MPP → LMPP), (MPP → CMP), (CMP → GMP), (CMP → MEP). We set the dimension of $\mathbf{z}_r$ to be 2 on this dataset.

Mouse retina development. Data from the Kharchenko lab `http://pklab.med.harvard.edu/peterk/review2020/examples/retina/`. The transitions tested with CBDir are (Neuroblast → PR), (Neuroblast → AC/HC), (Neuroblast → RGC). We set the dimension of $\mathbf{z}_r$ to be 2 on this dataset.

Dentate Gyrus development. Mouse Dentate Gyrus development at two time points P12 and P35 downloaded from the scVelo package [11]. The transitions tested with CBDir are (OPC → OL), (Radial Glia-like → Astrocytes), (Neuroblast → Granule immature). We set the dimension of $\mathbf{z}_r$ to be 3 on this dataset.

Intestinal organoid. Data from [42], downloaded from dynamo [5]. The transitions tested with CBDir are (Stem cells → TA cells), (Stem cells → Goblet cells), (Stem cells → Tuft cells), (TA cells → Enterocytes). We set the dimension of $\mathbf{z}_r$ to be 2 on this dataset.

Mouse hindbrain (Oligo). Data of mouse hindbrain oligodendrocyte lineage from [10], downloaded from `http://pklab.med.harvard.edu/ruslan/velocity/oligos/`. The transitions tested with CBDir are (COPs → NFOLs), (NFOLs → MFOLs). We set the dimension of $\mathbf{z}_r$ to be 1 on this dataset.

Mouse hindbrain (GABA, Glial). Processed data of mouse hindbrain with the differentiation of GABAergic interneuron and glial cells is downloaded from DeepVelo [18]. The transitions tested with CBDir are (Neural stem cells → progenitors), (Proliferating VZ progenitors → VZ progenitors), (VZ progenitors → Gliogenic progenitors), (VZ progenitors → Differentiating GABA interneurons), (Differentiating GABA interneurons → GABA interneurons). We set the dimension of $\mathbf{z}_r$ to be 1 on this dataset.

Mouse erythroid. Erythroid lineage of mouse gastrulation [33]. Downloaded from the scVelo package [11]. The transitions tested with CBDir are (Blood progenitors 1 → Blood progenitors 2), (Blood progenitors 2 → Erythroid1), (Erythroid1 → Erythroid2), (Erythroid2 → Erythroid3). We set the dimension of $\mathbf{z}_r$ to be 1 on this dataset.

Human bone marrow. Data from [34], downloaded with scVelo [11]. The transitions tested with CBDir are (HSC 1 → Mega), (HSC 1 → Ery 1), (Ery 1 → Ery 2), (HSC 1 → HSC 2), (HSC 2 → Precursors), (HSC 2 → Mono 2), (Mono 2 → Mono 1), (Precursors → DCs). We set the dimension of $\mathbf{z}_r$ to be 2 on this dataset.

scNT-seq neuron KCl stimulation. Cortical neurons are stimulated with potassium chloride (KCl) for 0, 15, 30, and 60 minutes. Data from [43], downloaded from `https://github.com/wulabupenn/scNT-seq`. The transitions tested with CBDir are the times (0 → 15), (15 → 30), (30 → 60), (60 → 120). We set the dimension of $\mathbf{z}_r$ to be 1 on this dataset.

Mouse embryonic fibroblast reprogramming. Reprogramming of mouse embryonic fibroblasts into induced endoderm progenitor cells [35]. Data was downloaded with the CellRank package [4]. We set the dimension of $\mathbf{z}_r$ to be 1 on this dataset.

Mouse gastrulation. Mouse gastrulation including all progenitors of major organs [33]. Downloaded from the scVelo package [11]. We subset to 20000 cells selected randomly for faster training. We set the dimension of $\mathbf{z}_r$ to be 7 on this dataset.

Embryonic mouse brain. Data downloaded from 10X `https://www.10xgenomics.com/resources/datasets/fresh-embryonic-e-18-mouse-brain-5-k-1-standard-1-0-0`. We followed the pre-processing from MultiVelo

[20]. We set the dimension of $\mathbf{z}_r$ to be 2 on this dataset.

VI. ACKNOWLEDGEMENTS

We thank Eric Johnson and Dominic Skinner for reading and providing feedback on the manuscript.

[1] Louise Deconinck, Robrecht Cannoodt, Wouter Saelens, Bart Deplancke, and Yvan Saeys. Recent advances in trajectory inference from single-cell omics data. *Current Opinion in Systems Biology*, 27:100344, September 2021.

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

[3] Jun Ding, Nadav Sharon, and Ziv Bar-Joseph. Temporal modelling using single-cell transcriptomics. *Nat. Rev. Genet.*, 23(6):355–368, June 2022.

[4] Marius Lange, Volker Bergen, Michal Klein, Manu Setty, Bernhard Reuter, Mostafa Bakhti, Heiko Lickert, Meshal Ansari, Janine Schniering, Herbert B Schiller, Dana Pe'er, and Fabian J Theis. CellRank for directed single-cell fate mapping. *Nat. Methods*, 19(2):159–170, January 2022.

[5] Xiaojie Qiu, Yan Zhang, Jorge D Martin-Rufino, Chen Weng, Shayan Hosseinzadeh, Dian Yang, Angela N Pogson, Marco Y Hein, Kyung Hoi Joseph Min, Li Wang, Emanuelle I Grody, Matthew J Shurtleff, Ruoshi Yuan, Song Xu, Yian Ma, Joseph M Replogle, Eric S Lander, Spyros Darmanis, Ivet Bahar, Vijay G Sankaran, Jianhua Xing, and Jonathan S Weissman. Mapping transcriptomic vector fields of single cells. *Cell*, 185(4):690–711.e45, February 2022.

[6] Ziqi Zhang and Xiuwei Zhang. Inference of high-resolution trajectories in single-cell RNA-seq data by using RNA velocity. *Cell Reports Methods*, 1(6):100095, October 2021.

[7] R Gupta, D Cerletti, G Gut, A Oxenius, and M Claassen. Cytopath: Simulation based inference of differentiation trajectories from RNA velocity fields. August 2021.

[8] Zhanlin Chen, William C King, Aheyon Hwang, Mark Gerstein, and Jing Zhang. DeepVelo: Single-cell transcriptomic deep velocity field learning with neural ordinary differential equations. April 2022.

[9] Ruishan Liu, Angela Oliveira Pisco, Emelie Braun, Sten Linnarsson, and James Zou. Dynamical systems model of RNA velocity improves inference of single-cell trajectory, pseudo-time and gene regulation. *J. Mol. Biol.*, page 167606, April 2022.

[10] Gioele La Manno, Ruslan Soldatov, Amit Zeisel, Emelie Braun, Hannah Hochgerner, Viktor Petukhov, Katja Lidschreiber, Maria E Kastriti, Peter Lönnerberg, Alessandro Furlan, Jean Fan, Lars E Borm, Zehua Liu, David van Bruggen, Jimin Guo, Xiaoling He, Roger Barker, Erik Sundström, Gonçalo Castelo-Branco, Patrick Cramer, Igor Adameyko, Sten Linnarsson, and Peter V Kharchenko. RNA velocity of single cells. *Nature*, 560(7719):494–498, August 2018.

[11] Volker Bergen, Marius Lange, Stefan Peidli, F Alexander Wolf, and Fabian J Theis. Generalizing RNA velocity to transient cell states through dynamical modeling. *Nat. Biotechnol.*, 38(12):1408–1414, August 2020.

[12] Melania Barile, Ivan Imaz-Rosshandler, Isabella Inzani, Shila Ghazanfar, Jennifer Nichols, John C Marioni, Carolina Guibentif, and Berthold Göttgens. Coordinated changes in gene expression kinetics underlie both mouse and human erythroid maturation. *Genome Biol.*, 22(1):197, July 2021.

[13] Volker Bergen, Ruslan A Soldatov, Peter V Kharchenko, and Fabian J Theis. RNA velocity—current challenges and future perspectives. *Mol. Syst. Biol.*, 17(8):e10282, August 2021.

[14] Gennady Gorin, Meichen Fang, Tara Chari, and Lior Pachter. RNA velocity unraveled. February 2022.

[15] Chenling Xu, Romain Lopez, Edouard Mehlman, Jeffrey Regier, Michael I Jordan, and Nir Yosef. Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. *Mol. Syst. Biol.*, 17(1):e9620, January 2021.

[16] Chen Qiao and Yuanhua Huang. Representation learning of RNA velocity reveals robust cell transitions. *Proceedings of the National Academy of Sciences*, 118(49):e2105859118, 2021.

[17] Mingze Gao, Chen Qiao, and Yuanhua Huang. UniTVelo: temporally unified RNA velocity reinforces single-cell trajectory inference. April 2022.

[18] Haotian Cui, Hassaan Maan, and Bo Wang. DeepVelo: Deep learning extends RNA velocity to multi-lineage systems with cell-specific kinetics. April 2022.

[19] Yichen Gu, David Blaauw, and Joshua D. Welch. Bayesian inference of rna velocity from multi-lineage single-cell data. *bioRxiv*, 2022.

[20] Chen Li, Maria Virgilio, Kathleen L Collins, and Joshua D Welch. Single-cell multi-omic velocity infers dynamic and decoupled gene regulation. December 2021.

[21] Yuanhua Huang and Guido Sanguinetti. BRIE2: computational identification of splicing phenotypes from single-cell transcriptomic experiments. *Genome Biol.*, 22(1):251, August 2021.

[22] Qian Li. sctour: a deep learning architecture for robust inference and accurate prediction of cellular dynamics. April 2022.

[23] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[24] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1278–II–1286. JMLR.org, June 2014.

[25] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[26] Malte D Luecken, M Büttner, K Chaichoompu, A Danese, M Interlandi, M F Mueller, D C Strobl, L Zappia, M Dugas, M Colomé-Tatché, and Fabian J Theis. Benchmarking atlas-level data integration in single-cell genomics. *Nat. Methods*, 19(1):41–50, December 2021.

[27] Robrecht Cannoodt, Wouter Saelens, Louise Deconinck, and Yvan Saeys. Spearheading future omics analyses using dyngen, a multi-modal simulator of single cells. *Nat. Commun.*, 12(1):1–9, June 2021.

[28] Romain Lopez, Jeffrey Regier, Michael B Cole, Michael I Jordan, and Nir Yosef. Deep generative modeling for single-cell transcriptomics. *Nat. Methods*, 15(12):1053–1058, December 2018.

[29] Jolene S Ranek, Natalie Stanley, and Jeremy E Purvis. Integrating temporal single-cell gene expression modalities for trajectory inference and disease prediction. March 2022.

[30] Kasper Daniel Hansen. Batch effects in scrna velocity analysis, 2021.

[31] W Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127, April 2006.

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

[33] Blanca Pijuan-Sala, Jonathan A Griffiths, Carolina Guibentif, Tom W Hiscock, Wajid Jawaid, Fernando J Calero-Nieto, Carla Mulas, Ximena Ibarra-Soria, Richard C V Tyser, Debbie Lee Lian Ho, Wolf Reik, Shankar Srinivas, Benjamin D Simons, Jennifer Nichols, John C Marioni, and Berthold Göttgens. A single-cell molecular map of mouse gastrulation and early organogenesis. *Nature*, 566(7745):490–495, February 2019.

[34] Manu Setty, Vaidotas Kiseliovas, Jacob Levine, Adam Gayoso, Linas Mazutis, and Dana Pe'er. Characterization of cell fate probabilities in single-cell data with palantir. *Nat. Biotechnol.*, 37(4):451–460, March 2019.

[35] Brent A Biddy, Wenjun Kong, Kenji Kamimoto, Chuner Guo, Sarah E Waye, Tao Sun, and Samantha A Morris. Single-cell mapping of lineage and identity in direct reprogramming. *Nature*, 564(7735):219–224, December 2018.

[36] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. *International Conference on Artificial Intelligence and Statistics*, 2020.

[37] Ricky T. Q. Chen. torchdiffeq, 2018.

[38] F Alexander Wolf, Philipp Angerer, and Fabian J Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.*, 19(1):15, February 2018.

[39] Adam Gayoso, Romain Lopez, Galen Xing, Pierre Boyeau, Valeh Valiollah Pour Amiri, Justin Hong, Katherine Wu, Michael Jayasuriya, Edouard Mehlman, Maxime Langevin, Yining Liu, Jules Samaran, Gabriel Misrachi, Achille Nazaret, Oscar Clivio, Chenling Xu, Tal Ashuach, Mariano Gabitto, Mohammad Lotfollahi, Valentine Svensson, Eduardo da Veiga Beltrame, Vitalii Kleshchevnikov, Carlos Talavera-López, Lior Pachter, Fabian J. Theis, Aaron Streets, Michael I. Jordan, Jeffrey Regier, and Nir Yosef. A python library for probabilistic analysis of single-cell omics data. *Nature Biotechnology*, Feb 2022.

[40] Aimée Bastidas-Ponce, Sophie Tritschler, Leander Dony, Katharina Scheibner, Marta Tarquis-Medina, Ciro Salinno, Silvia Schirge, Ingo Burtscher, Anika Böttcher, Fabian J Theis, Heiko Lickert, and Mostafa Bakhti. Comprehensive single cell mRNA profiling reveals a detailed roadmap for pancreatic endocrinogenesis. *Development*, 146(12), June 2019.

[41] Sonia Nestorowa, Fiona K Hamey, Blanca Pijuan Sala, Evangelia Diamanti, Mairi Shepherd, Elisa Laurenti, Nicola K Wilson, David G Kent, and Berthold Göttgens. A single-cell resolution map of mouse hematopoietic stem and progenitor cell differentiation. *Blood*, 128(8):e20–e31, August 2016.

[42] Nico Battich, Joep Beumer, Buys de Barbanson, Lenno Krenning, Chloé S Baron, Marvin E Tanenbaum, Hans Clevers, and Alexander van Oudenaarden. Sequencing metabolically labeled transcripts in single cells reveals mRNA turnover strategies. *Science*, 367(6482):1151–1156, 2020.

[43] Qi Qiu, Peng Hu, Xiaojie Qiu, Kiya W Govek, Pablo G Cámara, and Hao Wu. Massively parallel and time-resolved rna sequencing in single cells with scnt-seq. *Nature methods*, 17(10):991–1001, 2020.

[44] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

SUPPLEMENTAL INFORMATION

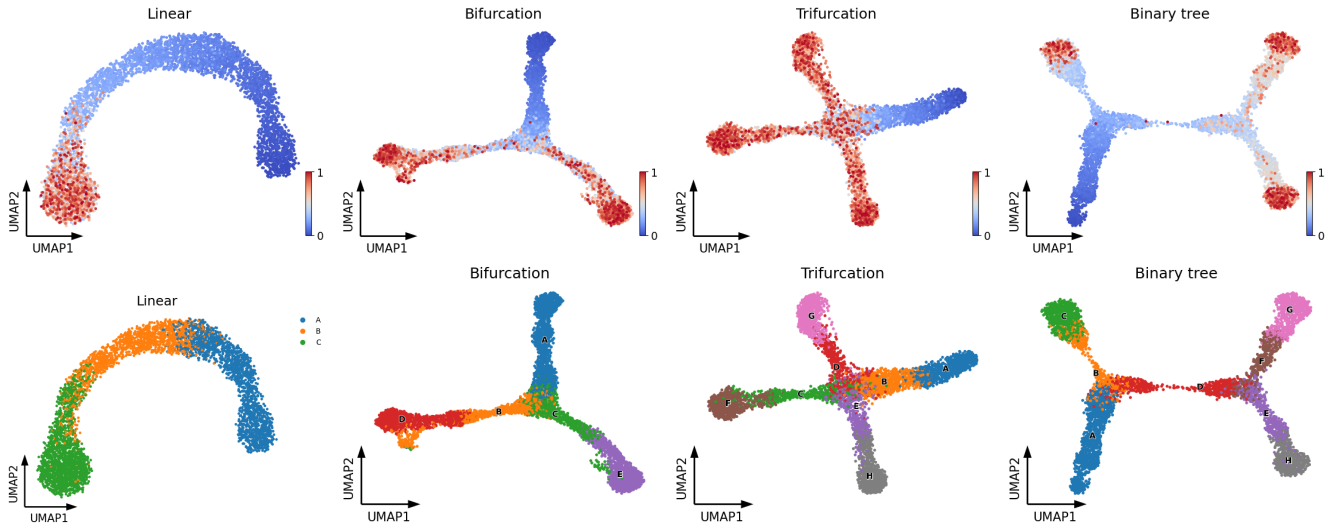

FIG. S1. **Synthetic datasets.** UMAP plots showing the synthetic datasets used to benchmark models. Colored by simulation time and milestone, which is used for CBDir metric and the celltype annotated model.

1. Model estimates error in latent time and velocity estimates

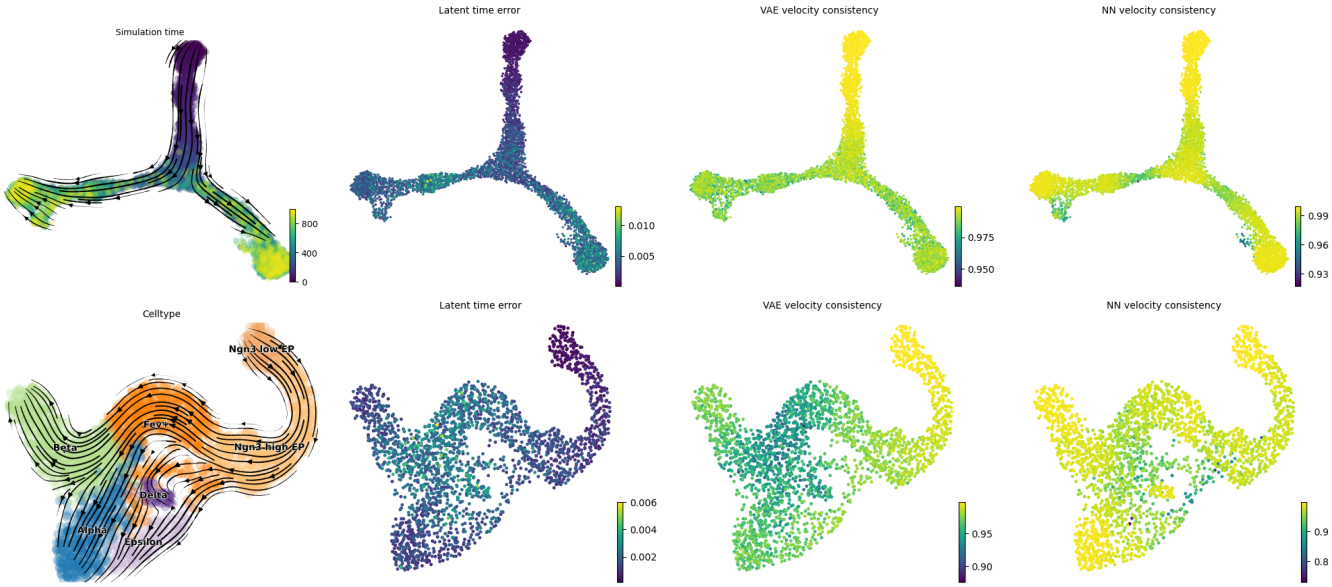

FIG. S2. **Error estimates in latent time and RNA velocity.** (Top) Synthetic bifurcation. (Bottom) Pancreas dataset. Samples from the approximate Bayesian posterior distribution of latent time and RNA velocity are used to estimate the error (standard error of the mean) of latent time and the consistency of RNA velocity. as a comparison, we show the average nearest neighbor velocity consistency, introduced in the scVelo package [11].

Previous models of RNA velocity do not include uncertainty estimates in parameters. Our model infers the approximate Bayesian posterior distribution of the latent time and velocities, enabling the estimation of approximate uncertainties. However, we note that variational Bayesian approximations underestimate uncertainty [44], and so here we only interpret these uncertainty estimates to be *relative* estimates of uncertainty.

In Figure S2, we show the velocity inferred by the model on a UMAP plot, as well as the standard error of the mean of the latent time, the velocity consistency by re-sampling from the VAE latent space, and the consistency of velocity from nearest neighbors (as computed by scVelo [11]). The VAE velocity consistency is computed by the average cosine similarity of all pairs of samples from the VAE.

For the synthetic bifurcation dataset, we show that error increases near and after the bifurcation. Similarly with the pancreas dataset, where error increases in the area near the transitions to Alpha, Beta, Delta, and Epsilon cells, and then decreases in the Alpha and Beta cells after the transition is made. These estimates make sense, showing a decreased uncertainty where cell lineage decisions are being made.

2. Integrating multi-omic data with ATAC-seq

To show our approach can be generalized to any data modality, we use a mouse embryonic brain dataset incorporating both RNA-seq and ATAC-seq, previously used with MultiVelo [20]. The diagram in Figure S3 shows the new structure of the latent variables, where we now include the latent space representation of chromatin accessibility $\mathbf{z}_c$. In this model, chromatin accessibility effects transcription of unspliced RNA $\mathbf{z}_u$, and chromatin dynamics are regulated by z_r. We still also allow the direct regulation of transcription by other methods than just chromatin accessibility, shown by retaining the connection between z_u and z_r as in Figure 1 above.

In Figure S3 the RNA velocity on the UMAP plot shows the differentiation direction from radial glia cells to neurons. The spliced RNA, unspliced RNA, and chromatin accessibility plots vs latent time for 4 different genes show the spliced, unspliced, and chromatin velocities. Velocities are positive when these variables increase, and negative when they decrease, as expected.

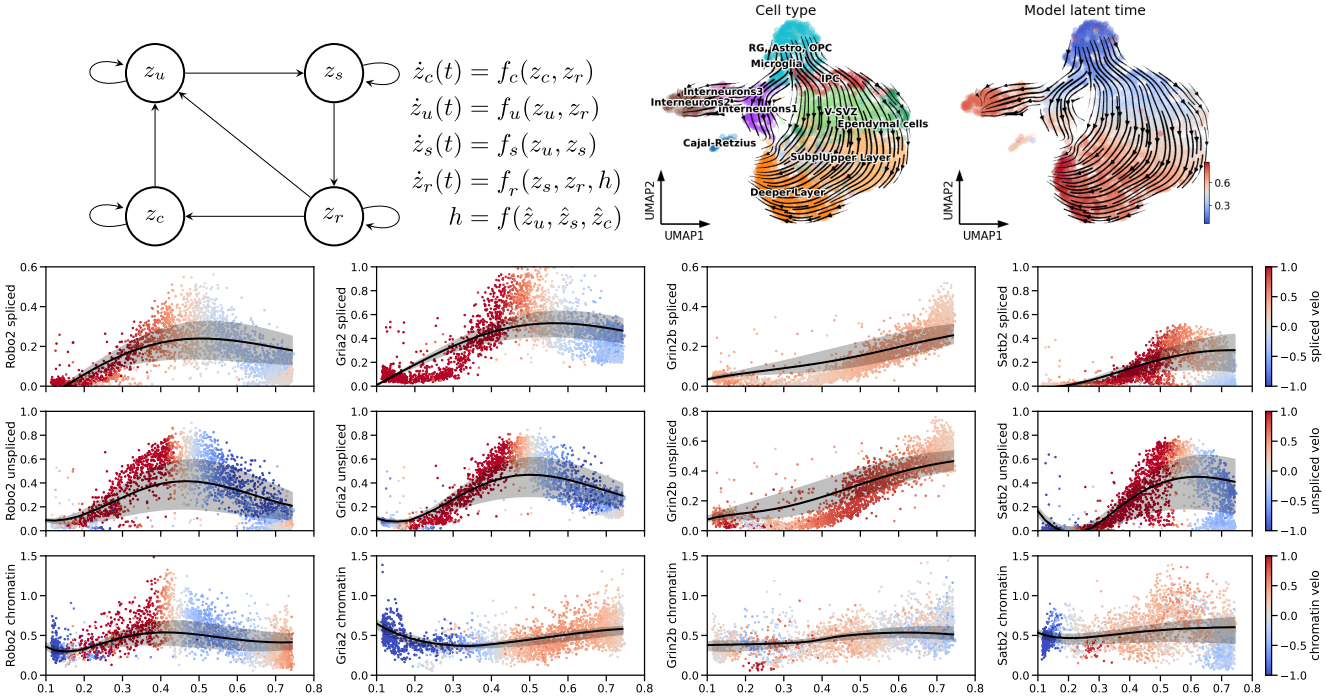

FIG. S3. **Integration of chromatin accessibility.** **a)** Integration of chromatin accessibility into the structured latent dynamics. The latent representation of chromatin accessibility z_c influences transcription of unspliced RNA z_u. Chromatin dynamics are also regulated by z_r. **b)** UMAP and RNA velocity. **c)** Spliced RNA, unspliced RNA, and chromatin accessibility for 4 selected genes, colored by the corresponding chromatin, unspliced, and spliced velocity. We subset to the RG, IPC, VSVZ, Upper Layer, and Deeper Layer cells for these genes.

3. Additional figures

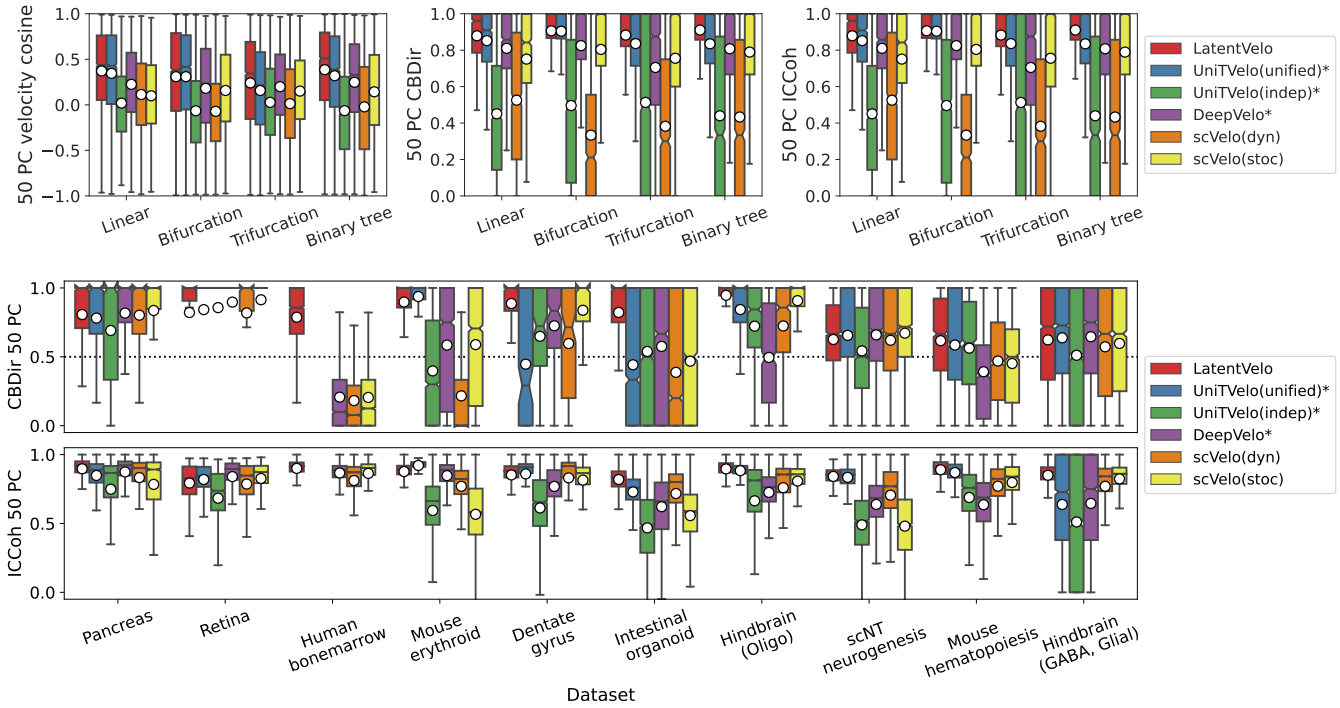

FIG. S4. **Quantitative benchmarking on synthetic and real data with recent RNA velocity methods. (Top row)** Quantitative metrics for the synthetic datasets. We show velocity cosine similarity, Cross-Boundary Directedness score (CBDir), and Inner-Cluster Coherence score (ICCoh) on the space of 50 principle components (PC). **(Bottom two rows)** Quantitative metrics for real datasets. We show CBDir and ICCoh scores on the space of 50 PC. Higher scores for all metrics are better. Note *: all settings and parameters of the comparison models are set at their defaults, we have not attempted to find better settings for these models. The only exploration done was fitting with multiple random seeds.

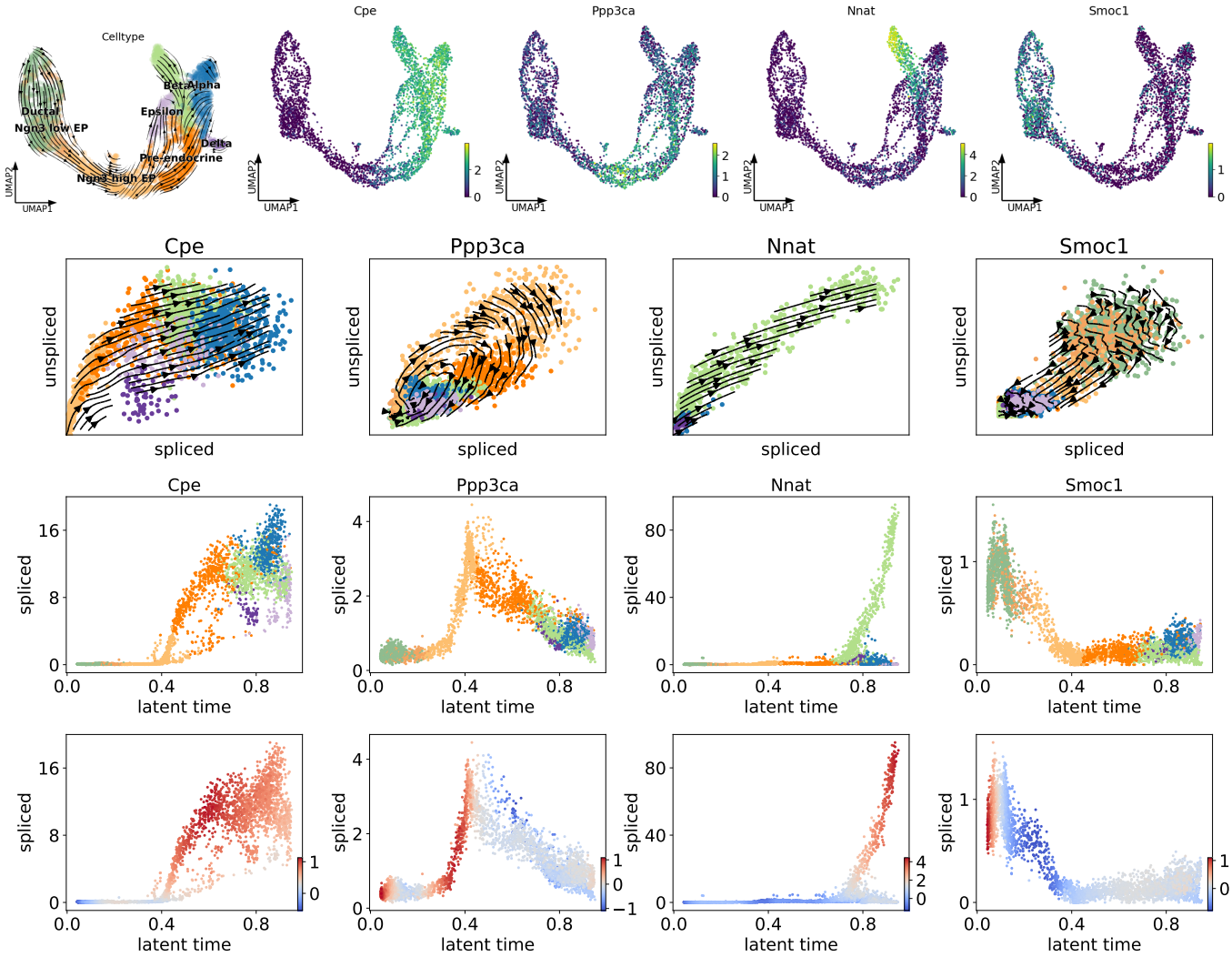

FIG. S5. **LatentVelo can infer the dynamics of single genes despite the compressed latent space.** (Top row) UMAP plot with LatentVelo velocity for pancreatic endocrinogenesis. We also highlight four genes showing a diverse set of dynamics, Cpe, Ppp3ca, Nnat, and Smoc1. (2nd row) we show inferred dynamics for these genes on unspliced vs spliced plots. Velocity on gene-space is computed by multiplying the latent velocity by the Jacobian of the decoder. We see a variety of dynamics, including induction for Cpe, an induction-repression cycle for Ppp3ca, lineage-dependent induction for Nnat, and repression for Smoc1. In the bottom two rows we show spliced RNA vs latent time colored by celltype and by the sign of the inferred velocity. We see positive or negative velocity at the expected points. This demonstrates that LatentVelo correctly infers dynamics for these genes.

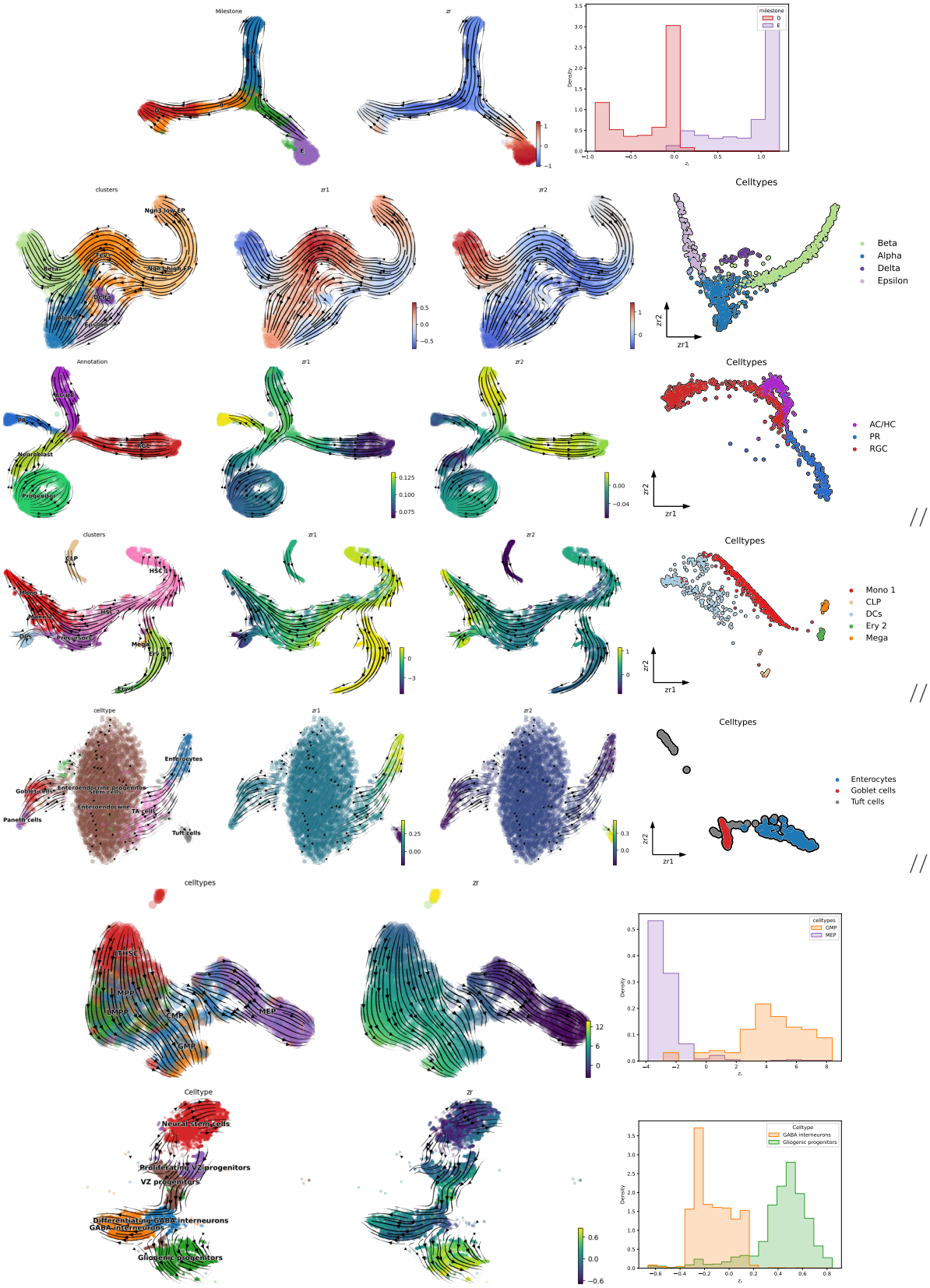

FIG. S6. **LatentVelo regulatory parameter $\mathbf{z}_r$ infers lineages.** Synthetic bifurcation, Pancreas, Retina, Bone marrow, Intestinal organoid, hematopoiesis, and hindbrain UMAP plots shown with velocity arrows and colored by cell-types and regulatory parameters $\mathbf{z}_r$. In the case of 2 regulatory parameters, we show a scatter plot. For 1 regulatory parameter, we show a histogram. These plots show clear separation between the lineages. We also use a logistic regression classifier to predict terminal states, achieving mean 25-fold cross-validation of accuracy of 98% for synthetic bifurcation, 96% for pancreas, 94% for retina, 99% for bonemarrow, 94% for intestinal organoid, 96% for hematopoiesis, and 92% for hindbrain, demonstrating that these clearly separated cell-type clusters accurately represent the different lineages. Since RNA velocity dynamics are parameterized by z_r, these plots demonstrate distinct dynamics on different lineages.

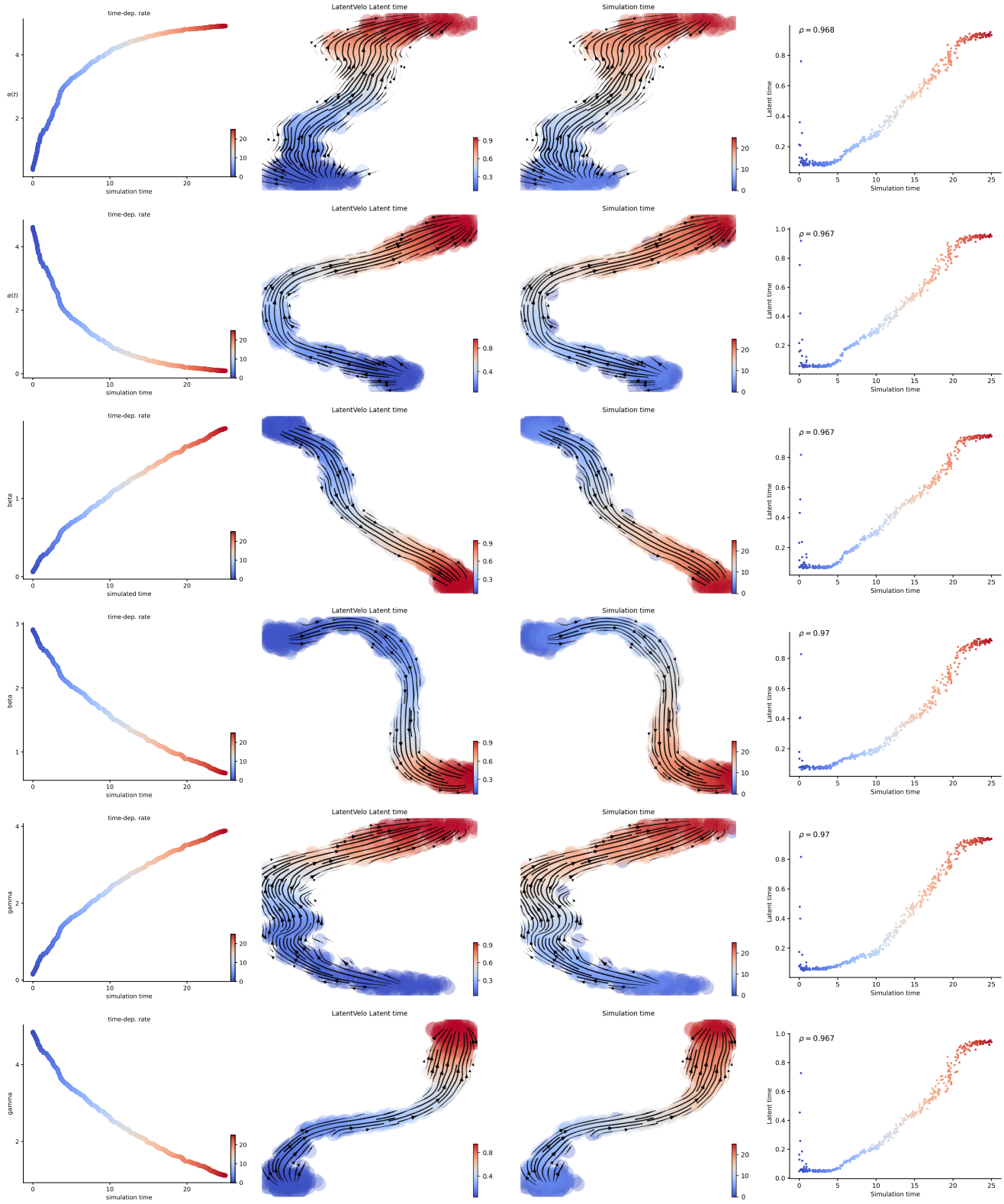

FIG. S7. **Time-dependent rate simulations.** We simulate from the scVelo linear model [11] while varying alpha (transcription rate), beta (splicing rate), and gamma (degradation rate). The simulation is run with 30 genes, and 5 are selected to be time-varying. We show increasing/decreasing for each of alpha, beta, and gamma. We assess our LatentVelo's ability to infer the proper direction of differentiation by rank correlation between latent time and simulation time. We use LatentVelo with 10 latent dimensions for these small datasets.

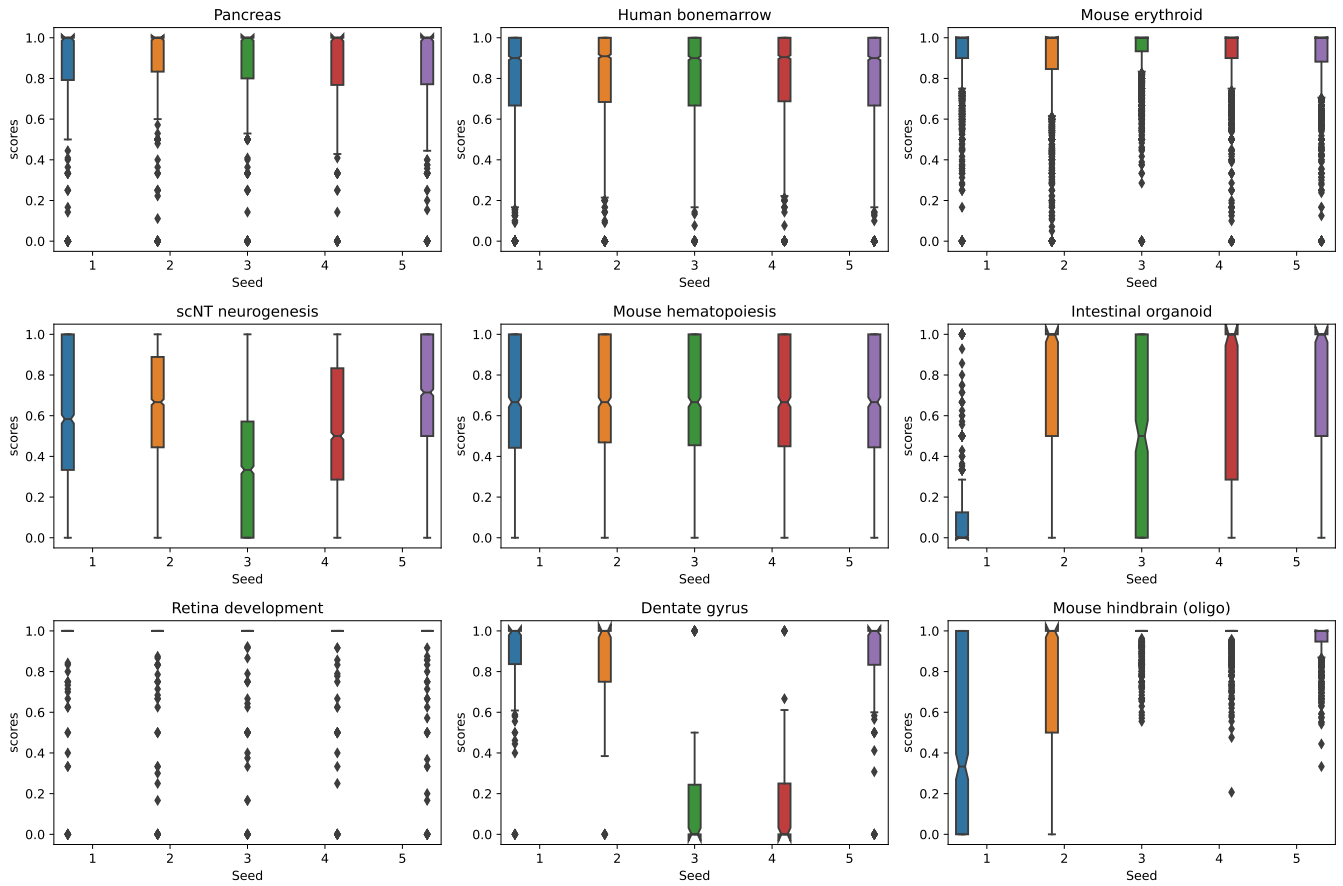

FIG. S8. **Training stochasticity.** We repeatedly train the model with different random seeds, resulting in different training/test sets, different model parameter initializations, and different batch sampling for training. For most datasets the model is robust, however occasionally on some datasets there is some variation. We recommend training the model multiple times.