CELLFLOW: A GENERATIVE FLOW-BASED MODEL FOR SINGLE-CELL COUNT DATA

Alessandro Palma[*], **Till Richter**[*], **Hanyi Zhang**
Helmholtz Centre Munich
Munich, Germany
{alessandro.palma,till.richter,hanyi.zhang}@helmholtz-munich.de

Andrea Dittadi, Fabian J.Theis
Helmholtz Centre Munich
Munich, Germany
{andrea.dittadi, fabian.theis}@helmholtz-munich.de

ABSTRACT

Generative modeling for single-cell RNA-seq has proven transformative in crucial fields such as learning single-cell representations and perturbation responses. However, despite their appeal in relevant applications involving data augmentation and unseen cell state prediction, use cases like generating artificial biological samples are still in their pioneering phase. While common approaches producing single-cell samples from noise operate in continuous space by assuming normalized gene expression, we argue for the necessity of sample generation in a raw transcription count space to favor processing-agnostic data generation and flexible downstream applications. To this end, we propose *cellFlow*, a Flow-Matching-based model that generates single-cell count data. In our empirical study, cellFlow performs on par with existing methods operating on normalized data when evaluated on three biological datasets. By carefully considering raw single-cell distributional properties, cellFlow is a promising avenue for future developments in single-cell generative models.

1 INTRODUCTION

Single-cell transcriptomics has enriched our understanding of biological systems. Mainly driven by advancements in RNA-sequencing technologies, datasets have increased in size and complexity (Angerer et al., 2017). Atlasing efforts such as the Human Cell Atlas underscore this development (Regev et al., 2017). However, cell atlases are limited by existing and available data. The data usable for biological research is thus restricted by the specific experiments conducted. Generative modeling offers a solution: we envision a reliable generative model as an augmentation to single-cell atlases, predicting novel cell states, perturbations (Lotfollahi et al., 2019) or even replacing large data storages with parameterized models able to recreate a desired single-cell dataset. Toward this vision, the reliability and realistic characteristics of generated samples are key to their use.

Generative modeling has already begun to impact the field of single-cell transcriptomics, both based on standard probabilistic parameter estimation (Cao et al., 2021; Crowell et al., 2023) and deep learning models (Sadria & Layton, 2023; Marouf et al., 2020; Heydari et al., 2022). However, current deep learning approaches are tailored to generate continuous normalized data, where the gene expression in different cells has been rescaled to sum to the same value. This strong assumption limits the application of such models to contexts where the heterogenous number of counts in the cell matter biologically or where more complex normalizations are required. Conversely, generative models operating in the count space such as scVI Lopez et al. (2018) are mostly employed for representation learning and not optimized to generate single-cells from noise. To address these issues, we opted to provide a more flexible tool designed to reproduce realistic single-cell raw counts, facilitating compatibility with any downstream pipeline.

[*]Equal contribution

Here, we present cellFlow (Fig. 1), a flow-based generative model for single-cell counts that overcomes the need for data normalization or the use of a low-dimensional continuous latent space. This is achieved by mapping discrete data to the continuous parameter space of a Negative Binomial likelihood modeling overdispersed counts, and subsequently training a flow-based generative model on top of it. As a generative model, we leverage the recently proposed Conditional Flow Matching (Tong et al., 2023b), which showed promising performance both on image-based and biological data. In summary, we propose the following contributions:

- An approach to apply flow-based generative models to discrete data under an assumed likelihood model without a representation bottleneck.
- A formulation of such model for single-cell data based on a Negative Binomial distribution.
- An empirical study on three real-world single-cell RNA-sequencing datasets.

2 Related works

Traditional models for simulating single-cell RNA-seq, such as those based on probabilistic parameter estimation frameworks Zappia et al. (2017), have been complemented by deep-learning-based generative models. Notably, scVI Lopez et al. (2018), a VAE adapted to single-cell data's discrete distributional properties, addresses tasks such as batch integration and representation learning. Other approaches, including VAEs for single-cell generation Heydari et al. (2022), GANs Marouf et al. (2020), and Diffusion Models Luo et al. (2024); Sadria & Layton (2023), also aim for single-cell data generation. Our work builds upon Flow Matching (FM) Lipman et al. (2023), a novel framework for stable normalizing flows, applied previously in trajectory inference Tong et al. (2023a;b) and Optimal Transport Klein et al. (2023). However, our method explores flows as generative models of discrete single-cell counts at cellular resolution, opening new avenues for exploration in the field.

3 Background

3.1 Modeling single-cell RNA-sequencing counts

Gene expression in single cells is captured via scRNA-seq as a vector of discrete counts representing the number of molecules detected experimentally at a measured cell state for a certain gene. Single-cell counts are complex in nature. Cells may vary in size or exhibit different detection rates. Consequently, technical and biological variation is responsible for essential characteristics to be accounted for when modeling scRNA-seq, such as sparsity and overdispersion. A common choice is to assume that each gene in each cell follows a Negative Binomial distribution, which allows a flexible relation between gene-specific means and variances depending on an inverse dispersion parameter.

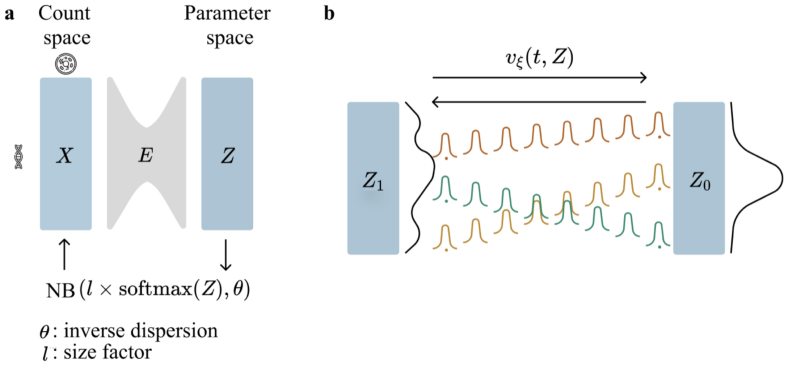

Figure 1: cellFlow framework. **a.** A deep dimensionality-preserving encoder trained via maximum likelihood maps discrete single-cell data to the continuous mean of the Negative Binomial distribution. **b.** A flow-based generative model is trained to transform noise to the output space of the encoder model.

We denote the single-cell gene expression matrix as $\mathbf{X} \in \mathbb{N}_0^{N \times G}$ with N and G the number of cells and genes, respectively. Each entry of $\mathbf{X}_{ng}$ is assumed to follow a Negative Binomial (NB) model:

$$\mathbf{X}_{ng} \sim \mathrm{NB}(\boldsymbol{\mu}_{ng}, \boldsymbol{\theta}_g) \tag{1}$$

where $\boldsymbol{\mu}_{ng} \in \mathbb{R}^+$ is a cell-gene specific mean and $\boldsymbol{\theta}_g \in \mathbb{R}^+$ the gene specific inverse dispersion. In other words, we assume that each cell has an individual mean expression, whereas technical variation is modeled as a single inverse dispersion value learnt per gene across the whole dataset.

The expression value of single genes spans a wide range depending on the technology and biological system employed for the study. In single-cell generative modeling it is common practice to decompose gene expression into the product of gene proportions $\boldsymbol{\rho}_{ng}$ and a size factor $l_n = \sum_{g=1}^{G} \mathbf{X}_{ng}$ which represents the total number of counts in the cell and depends on various factors such as technical effects or cell size. In VAEs, $\boldsymbol{\mu}$ is a function of a latent variable $\mathbf{z}$:

$$\boldsymbol{\mu} = l\, \boldsymbol{\rho}\,, \quad \boldsymbol{\rho} = \mathrm{softmax}(h_\psi(\mathbf{z}))\,, \tag{2}$$

with h_ψ representing a decoder parameterized by ψ.

3.2 Continuous normalizing flows

Consider continuous data in $\mathbf{Z} \in \mathbb{R}^G$. Notably, the density $p(\mathbf{z})$ can be modeled as a time-dependent probability path $p : [0,1] \times \mathbb{R}^d \to \mathbb{R}^+$ which transforms a simple distribution $p_0(\mathbf{z})$, such as a standard Gaussian, into $p_1(\mathbf{z}) = p(\mathbf{z})$, where we denote the probability density at time $t \in [0,1]$ as p_t. The probability path is said to be *generated* by a time-dependent vector field $u_t : [0,1] \times \mathbb{R}^G \to \mathbb{R}^G$ if p and u satisfy the continuity equation $\frac{\mathrm{d}p}{\mathrm{d}t} = -\nabla \cdot (p_t u_t)$. The vector field u_t can be seen as modeling the instantaneous change of an invertible time-dependent diffeomorphic transformation of $\mathbf{z}$ indicated as $\phi_t : \mathbb{R}^G \to \mathbb{R}^G$ such that:

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(\mathbf{z}) = u_t(\phi_t(\mathbf{z}))\,, \qquad \phi_0(\mathbf{z}) = \mathbf{z}\,. \tag{3}$$

A *Continuous Normalizing Flow* (CNF) is then defined as $p_t = [\phi_t]_* p_0$, with

$$[\phi_t]_* p_0 = p_0(\phi_t^{-1}(\mathbf{z}))\mathrm{det}\left[\frac{\partial \phi_t^{-1}}{\partial \mathbf{z}}(\mathbf{z})\right] \tag{4}$$

and $*$ representing the push-forward operations. In other words, by deterministically transforming the variable $\mathbf{z}$ following the vector field $u_t(\mathbf{z})$, $\phi_t(\mathbf{z})$ causes the prior density p_0 to evolve into p_1.

3.3 Conditional Flow Matching

We deal with the formulation of CNF which leverages Neural Ordinary Differential Equations (NeuralODEs). Specifically, our goal is to learn a parameterized velocity field $v_\xi(t, \mathbf{z})$ which approximates $u_t(\mathbf{z})$ from Section 3.2. To do so, we resort to Conditional Flow Matching (CFM), a flexible framework to learn normalizing flows without the need to simulate NeuralODEs during training. Briefly, Tong et al. (2023b) express the time marginal $p_t(\mathbf{z})$ as a mixture conditioned on a latent variable $\mathbf{c}$ with density $q(\mathbf{c})$, i.e., $p_t(\mathbf{z}) = \int p_t(\mathbf{z}|\mathbf{c})q(\mathbf{c})\mathrm{d}\mathbf{c}$. The marginal field $u_t(\mathbf{z})$ is related to the conditional field $u_t(\mathbf{z}|\mathbf{c})$ generating $p_t(\mathbf{z}|\mathbf{c})$ by:

$$u_t(\mathbf{z}) = \mathbb{E}_{q(\mathbf{c})}\left[\frac{u_t(\mathbf{z}|\mathbf{c})p_t(\mathbf{z}|\mathbf{c})}{p_t(\mathbf{z})}\right]\,. \tag{5}$$

Given a specific formulation of the conditioning variable $\mathbf{c}$, $u_t(\mathbf{z}|\mathbf{c})$ and $p_t(\mathbf{z}|\mathbf{c})$ are tractable and training $v_\xi(t, \mathbf{z})$ to approximate $u_t(\mathbf{z}|\mathbf{c})$ is equivalent to regressing against $u_t(\mathbf{z})$ up to a constant independent of ξ. In our approach, we follow Optimal Transport CFM (OT-CFM) described in Tong et al. (2023b), which involves defining $\mathbf{c}$ as pairs of samples $(\mathbf{z}_0, \mathbf{z}_1)$, which are sampled from the static Optimal Transport coupling $\mathbf{c} \sim q(\mathbf{c}) = \pi^*(\mathbf{Z}_0, \mathbf{Z}_1)$, where $\mathbf{Z}_0$ and $\mathbf{Z}_1$ represent the source and target sets of observations, respectively. By assuming Gaussian marginals for p_t and connecting $\mathbf{z}_0$ and $\mathbf{z}_1$ through Gaussian flows, both $p_t(\mathbf{z}|\mathbf{c})$ and $u_t(\mathbf{z}|\mathbf{c})$ become computationally manageable:

$$p_t(\mathbf{z}|\mathbf{c}) = \mathcal{N}(t\mathbf{z}_1 + (1-t)\mathbf{z}_0, \sigma^2)\,, \qquad u_t(\mathbf{z}|\mathbf{c}) = \mathbf{z}_1 - \mathbf{z}_0, \tag{6}$$

where σ^2 is a predefined value. Consequently, the CFM loss is expressed as:

$$L_{\mathrm{OT-CFM}} = \mathbb{E}_{t, q(\mathbf{c}), p_t(\mathbf{z}|\mathbf{c})}\|v_\xi(t, \mathbf{z}) - u_t(\mathbf{z}|\mathbf{c})\|_2, \tag{7}$$

with t being sampled from a uniform distribution $U(0,1)$. When p_0 is set as a Gaussian prior over $\mathbb{R}^G$, samples from $\mathcal{N}(\mathbf{0}, \mathbf{I}_G)$ can be used to generate observations from the data distribution.

4 CELLFLOW

4.1 PROBLEM STATEMENT

Our objective is to develop a statistical model utilizing OT-CFM for generating single-cell counts. The challenge lies in handling discrete data, which violates the assumptions of continuous flows. While prior works have explored discrete normalizing flows (Tran et al., 2019) and dequantization methods (Ho et al., 2019), none have addressed the complexities of high-dimensional single-cell data, characterized by sparsity and overdispersion. To tackle this, we leverage the distributional properties of single-cell data, as discussed in Section 3.1, by learning to generate gene expression mapped to a continuous space representing the mean of a Negative Binomial distribution. Realistic single-cell count data can then be sampled from the generated parameter vectors following Eq. (1).

4.2 THE PROBABILISTIC MODEL

Let $\mathbf{X} \in \mathbb{N}_0^{N \times G}$ be a single-cell dataset and $\mathbf{Y} \in (0,1)^{N \times N_y}$ a matrix of one-hot categorical conditions indexed 1 through N_y. Considering a single data point $(\mathbf{x}, \mathbf{y})$ consisting of a row from the data matrices above, we define the following latent variable generative model:

$$p(\mathbf{x}, \mathbf{z}, l, \mathbf{y}) = p(\mathbf{x}|\mathbf{z}, l; \boldsymbol{\theta}) p(\mathbf{z}|l, \mathbf{y}) p(l|\mathbf{y}) p(\mathbf{y}) \tag{8}$$

where $\mathbf{z} \in \mathbb{R}^G$ is a latent variable, l is the size factor, defined as the total number of counts in a cell, and $\boldsymbol{\theta}$ is the inverse dispersion parameter described in Eq. (1). In what follows, we examine our choices of the individual elements of the model in Eq. (8).

Condition probability $p(\mathbf{y})$. We model $p(\mathbf{y})$ as a categorical distribution $\mathrm{Cat}(N_y, \pi_y)$, where N_y is the number of categories and π_y is a vector of per-class probabilities. We fit the vector π_y on the data as the observed proportions of each of the N_y conditions.

Size factor probability $p(l|\mathbf{y})$. We assume that the size factor is log-normally distributed as $\mathrm{LogNormal}(\mu_{l,y}, \sigma_{l,y})$ with category-specific means $\mu_{l,y}$ and standard deviations $\sigma_{l,y}$. Similarly to $p(\mathbf{y})$, we fit the parameters of $p(l|\mathbf{y})$ on the data.

Latent variable probability $p(\mathbf{z}|l, \mathbf{y})$. As $p(\mathbf{z}|l, \mathbf{y})$, we choose a CNF trained via OT-CFM (see Section 3.3). In other words, we learn a parameterized velocity function $v_\xi(t, \mathbf{x}, l, \mathbf{y})$ that generates an Optimal Transport probability flow p_t transporting a standard multivariate Gaussian distribution to the density of $\mathbf{z}$, for which we do not assume any parametric form.

Likelihood model $p(\mathbf{x}|\mathbf{z}, l; \boldsymbol{\theta})$. Finally, we formulate the likelihood model from which generated counts are sampled given $\mathbf{z}$, l and $\mathbf{y}$. As previously noted, we assume a parametric form of $p(\mathbf{x}|\mathbf{z}, l; \boldsymbol{\theta})$ as the Negative Binomial distribution $\mathrm{NB}(l\boldsymbol{\rho}, \boldsymbol{\theta})$ where $\boldsymbol{\rho} = \mathrm{softmax}(\mathbf{z})$. It is important to notice that our formulation is analogous to that in Eq. (2). However, we do not make use of a parameterized decoder h_ψ. That is, we assume that $\mathbf{z}$ already denotes the pre-softmax representation of the gene proportions with no need for additional non-linearities.

4.3 TRAINING

In practice, we train each element of Eq. (8) separately. Given the data as described in Section 4.2, we derive the parameterizations for $p(\mathbf{y})$ and $p(l|\mathbf{y})$ trivially by fitting the distributions to the empirical category proportions and (conditional) size factors. To train OT-CFM and learn to generate $\mathbf{z}$ from Gaussian noise, we first need to find a proper representation $\mathbf{z}$ that equates gene proportions before performing softmax and scaling by l. In our case, we learn such a representation from the data by means of a dimensionality-preserving encoder f_ν that maps a discrete data point $\mathbf{x}$ to $\mathbf{z}$ and is trained via maximum likelihood optimization following:

$$\arg\max_{\nu, \boldsymbol{\theta}} p(\mathbf{x}|\mathbf{z}, l; \boldsymbol{\theta}) = \arg\max_{\nu, \boldsymbol{\theta}} \mathrm{NB}(l \cdot \mathrm{softmax}(f_\nu(\mathbf{x})), \boldsymbol{\theta}) \,, \tag{9}$$

where $\mathbf{z} = f_\nu(\mathbf{x})$, p follows a Negative Binomial density, and $\boldsymbol{\theta}$ is learnt as a per-gene inverse dispersion parameter to optimize the likelihood. In other words, f_ν encodes the data into a continuous space corresponding to the mean of a cell's distribution up to a softmax and a scaling operation. The reason behind such a choice is that $\mathbf{z}$ lives in an unconstrained domain, making training a NeuralODE easier than forcing it to directly generate the strictly positive mean of the Negative Binomial.

Table 1: Evaluation metrics of single-cell data generated by different models. Output type indicates whether a model is designed to generate *counts* or *normalized* expression vectors. Bold and underlined entries represent the best and second-best models.

	Output type	2-WD(↓)			MMD(↓)			Recall(↑)		
		PBMC	Dentate gyrus	Lung	PBMC	Dentate gyrus	Lung	PBMC	Dentate gyrus	Lung
Splatter	counts	**2.17**	1.79	2.63	1.45	<u>0.55</u>	3.11	0.14	0.14	0.05
scGAN	normalized	**2.17**	**1.72**	<u>2.02</u>	<u>1.19</u>	0.76	<u>2.89</u>	0.29	**0.63**	**0.12**
ACTIVA	normalized	2.21	2.50	2.34	**0.85**	3.84	4.70	<u>0.53</u>	0.08	<u>0.18</u>
scDiffusion	normalized	2.49	<u>1.77</u>	**1.90**	1.76	**0.32**	**2.05**	0.06	0.08	0.01
cellFlow	counts	<u>2.18</u>	<u>1.77</u>	2.04	1.41	0.88	2.93	**0.90**	<u>0.60</u>	**0.41**

Given an optimized encoder f_ν, we train OT-CFM by encoding each batch as $\mathbf{z} = f_\nu(\mathbf{x})$ and training the vector field v_ξ on the derived representation as described in Section 3.3.

4.4 Sampling

In order to sample from a trained model, we first draw a conditioning vector $\mathbf{y} \sim p(\mathbf{y})$ and use it to index the log-normal distribution from which the library size is sampled. A noise vector $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_G)$ is then sampled and used as the initial condition for the following simulation:

$$\mathbf{z}_1 = \mathbf{z}_0 + \int_0^1 v_\xi(t, \mathbf{z}_t, l, \mathbf{y})\mathrm{d}t \tag{10}$$

We then map $\mathbf{z}_1$ to the mean vector $\boldsymbol{\mu} = l\,\mathrm{softmax}(\mathbf{z}_1)$ and generate RNA-seq counts by sampling $\mathbf{x} \sim \mathrm{NB}(\boldsymbol{\mu}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ has been pre-trained via likelihood optimization (see Section 4.3).

5 Experiments

5.1 Datasets

We evaluate cellFlow on three single-cell RNA-sequencing datasets; **(i)** We first employ a PBMC (Peripheral Blood Mononuclear Cells) dataset freely available for download from 10X Genomics and considered in the Seurat package Satija et al. (2015) tutorials [1]. The dataset consists of 2700 single cells from a healthy donor clustering into 8 cell types; **(ii)** The dentate gyrus dataset La Manno et al. (2018) follows the expression of 18,213 cells from a developing mouse hippocampus. It counts 14 cell types denoting the different stages of cellular differentiation; **(iii)** We also use a dataset of lung cells Vieira Braga et al. (2019) consisting of 32,472 cells with 17 cell types. Such datasets contain substantial batch effects being cellular samples collected from three separate sources. For all datasets, cellFlow is conditioned on cell type.

5.2 Baselines

As baselines, we include three recent deep generative models for single-cell data generation: scGAN Marouf et al. (2020), scDiffusion Luo et al. (2024) and ACTIVA Heydari et al. (2022). Although similar in scope to cellFlow, all methods are designed to generate real-valued normalized gene expression, where cell counts are normalized to sum to a pre-defined value. We train them in their original settings and compare them to cellFlow on a subset of 2000 highly variable genes. As a baseline, we also considered Splatter Zappia et al. (2017), a simulation tool of single-cell counts based on gene expression parameter estimation.

5.3 Evaluation metrics

We evaluated the generated samples both qualitatively and quantitatively. Qualitatively, we assess the joint UMAP McInnes et al. (2018) and Principal Component Analysis (PCA) visualization of real and generated data. Quantitatively, our focus lies on two key aspects: data characteristics and distributional similarities. For the former, we study if the sparsity patterns of generated count samples preserve those observed in the real data. To quantify the closeness of generated data to the original

[1] https://satijalab.org/seurat/articles/pbmc3k_tutorial.html

distribution, we evaluate **(i)** the squared Wasserstein-distance (2-WD) and **(ii)** the linear Maximum Mean Discrepancy (MMD) measures between real and generated datasets (see Appendix B). Additionally, we include **(iii)** the Recall measure (Sajjadi et al., 2018), which gets closer to 1 the more real samples lie on the generated data, estimated via 10-nearest-neighbors (kNN) balls around fake cells. All metrics were computed in a reduced feature space of 50 PCs.

5.4 Results

We present our quantitative results in Table 1, where we compare cellFlow with the considered simulation methods. Notably, cellFlow performs competitively with other models, reaching the best or second-best Recall score in two out of three datasets and the second-best squared Wasserstein distance in both the PBMC3K and Dentate gyrus datasets. A better Recall means that the generated data mixes with real data neighborhoods more effectively, as exemplified by Fig. 2a via the PBMC3k dataset, where the largest score discrepancy between our and competing models arises.

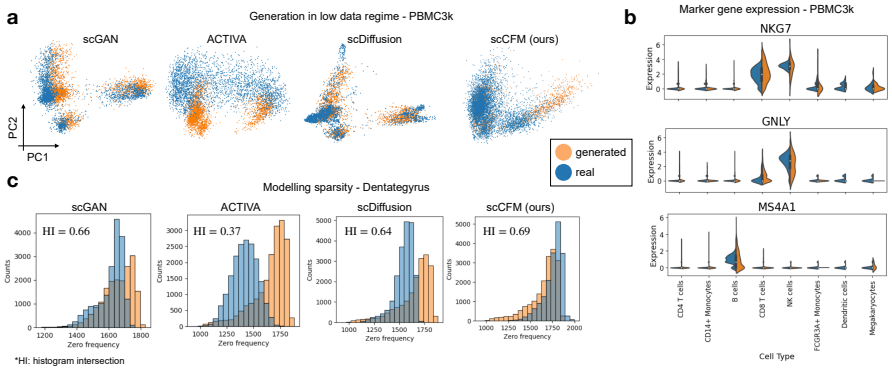

Figure 2: Evaluation of the generated single-cell samples with real data in blue and generated data in orange. **a.** PCA visualization of generated compared to the real dataset. **b.** Gene expression of known markers in the PBMC3k dataset. **c.** Frequency of zeros in real and generated cells. HI stands for Histogram Intersection.

We remark that our task of generating counts from noise is complicated and unexplored, as we preserve the nature of the data and allow a more flexible downstream usage such as datasets where a homogeneous normalization is not optimal at preserving the biological signal. Despite the complex task, our conditional model reproduces faithful lineage-specific marker expression (see Fig. 2b for an example on the PBMC3k dataset) and, therefore, shows promising performance as a tool for realistic gene expression generation. Finally, in Fig. 2c we demonstrate that considering a discrete likelihood model, like in our case, better captures key properties such as data sparsity, as the frequency of per-cell zero counts better approximates the one of observed data. This aspect is inherent in our model formulation. While other generative models assume continuous single-cell data, samples from cellFlow incorporate realistic single-cell distributional properties.

6 Conclusion

In this work, we presented cellFlow, a deep-learning approach for training flow-based models to generate single-cell discrete counts. To alleviate the issue of applying continuous models to discrete data, we developed a deep dimensionality-preserving encoder that maps single-cell data to the continuous parameter space of a Negative Binomial likelihood. The parameter space allows the training of an OT-CFM model such that, during sampling, noise vectors are mapped to the continuous single-cell parameters used to sample the data likelihood and obtain realistic counts. We tested our model on three single-cell datasets, showing competitive performance with existing generative models for normalized counts and better sparsity-preservation properties of generated scRNA-seq vectors. To the best of our knowledge, this represents a novel and promising research direction that warrants extensive exploration. Future efforts will build on the introduced model for a comprehensive assessment of potential downstream tasks, such as data augmentation and perturbation prediction, and the generation of multi-modal data following different likelihood models.

ACKNOWLEDGMENTS

We thank Luke Zappia for his constructive feedback on the storyline, enhancing its quality. A.P. and T.R. are supported by the Helmholtz Association under the joint research school *Munich School For Data Science*. A.P and F.J.T acknowledge support from the German Federal Ministry of Education and Research (BMBF) (HOPARL, 031L0289A). T.R. and F.J.T. acknowledge support by the Helmholtz Association's Initiative and Networking Fund through CausalCellDynamics (grant # Interlabs-0029), F.J.T. acknowledges support by the European Union (ERC, DeepCell - 101054957). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

Philipp Angerer, Lukas Simon, Sophie Tritschler, F Alexander Wolf, David Fischer, and Fabian J Theis. Single cells make big data: new challenges and opportunities in transcriptomics. *Current opinion in systems biology*, 4:85–91, 2017.

Yue Cao, Pengyi Yang, and Jean Yee Hwa Yang. A benchmark study of simulation methods for single-cell rna sequencing data. *Nature communications*, 12(1):6911, 2021.

Helena L Crowell, Sarah X Morillo Leonardo, Charlotte Soneson, and Mark D Robinson. The shaky foundations of simulating single-cell rna sequencing data. *Genome Biology*, 24(1):1–19, 2023.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

A Ali Heydari, Oscar A Davalos, Lihong Zhao, Katrina K Hoyer, and Suzanne S Sindi. Activa: realistic single-cell rna-seq generation with automatic cell-type identification using introspective variational autoencoders. *Bioinformatics*, 38(8):2194–2201, February 2022. ISSN 1367-4811. doi: 10.1093/bioinformatics/btac095.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pp. 2722–2730. PMLR, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Dominik Klein, Théo Uscidda, Fabian Theis, and Marco Cuturi. Generative entropic neural optimal transport to map within and across spaces, 2023.

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.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023.

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

Mohammad Lotfollahi, F. Alexander Wolf, and Fabian J. Theis. scgen predicts single-cell perturbation responses. *Nature Methods*, 16(8):715–721, July 2019. ISSN 1548-7105. doi: 10.1038/s41592-019-0494-8.

Erpai Luo, Minsheng Hao, Lei Wei, and Xuegong Zhang. scdiffusion: conditional generation of high-quality single-cell data using diffusion model. *arXiv preprint arXiv:2401.03968*, 2024.

Mohamed Marouf, Pierre Machart, Vikas Bansal, Christoph Kilian, Daniel S Magruder, Christian F Krebs, and Stefan Bonn. Realistic in silico generation and augmentation of single-cell rna-seq data using generative adversarial networks. *Nature communications*, 11(1):166, 2020.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Aviv Regev, Sarah A Teichmann, Eric S Lander, Ido Amit, Christophe Benoist, Ewan Birney, Bernd Bodenmiller, Peter Campbell, Piero Carninci, Menna Clatworthy, et al. The human cell atlas. *elife*, 6:e27041, 2017.

Mehrshad Sadria and Anita Layton. The power of two: integrating deep diffusion models and variational autoencoders for single-cell transcriptomics analysis. *bioRxiv*, pp. 2023–04, 2023.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Rahul Satija, Jeffrey A Farrell, David Gennert, Alexander F Schier, and Aviv Regev. Spatial reconstruction of single-cell gene expression data. *Nature Biotechnology*, 33:495–502, 2015. doi: 10.1038/nbt.3192.

Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schrödinger bridges via score and flow matching, 2023a.

Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport, 2023b.

Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Felipe A Vieira Braga, Gozde Kar, Marijn Berg, Orestes A Carpaij, Krzysztof Polanski, Lukas M Simon, Sharon Brouwer, Tomás Gomes, Laura Hesse, Jian Jiang, et al. A cellular census of human lungs identifies novel cell states in health and in asthma. *Nature medicine*, 25(7):1153–1163, 2019.

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

Luke Zappia, Belinda Phipson, and Alicia Oshlack. Splatter: simulation of single-cell rna sequencing data. *Genome Biology*, 18(1), September 2017. ISSN 1474-760X. doi: 10.1186/s13059-017-1305-0.

A MODEL DETAILS

A.1 ENCODER MODEL

The encoder model is a two-layer Multi-Layer Perceptron (MLP) with `ELU` activation function and batch normalization. The first two linear layers count 512 units, projecting the data into a lower dimension. The last two layers map the bottleneck back to the gene expression dimension. We train the model with a learning rate of `0.001` and `AdamW` optimizer.

A.2 VELOCITY MODEL v_ξ

For the velocity model, we develop a deep Residual Neural Network (ResNet) model inspired by the UNET used in Denoising Diffusion Models (Ho et al., 2020) but without the multiscale structure typical of computer vision applications. More in detail, an initial linear layer maps the cell to a 512-dimensional space. Here, three ResNet blocks are stacked on top of each other and transform the data across hidden representations. At every step of the ResNet, embeddings for the library size, cell type label and time are projected into the same dimension of the hidden representations and added to them. Library size and time are embedded with sinusoidal embeddings Vaswani et al. (2017), whereas the one-hot label for cell type is initialized as a `torch.nn.Embedding` object of dimension 256. All non-linearities are `SiLU` activation functions. After the residual blocks, a final upsampling layer maps the hidden output of the ResNet blocks to the gene dimension.

A.3 HYPERPARAMETERS FOR OT-CFM TRAINING

We train OT-CFM with a batch size of 256 and $\sigma = 0.0001$ controlling the deviation around the straight line between source and target described in Eq. (6). The generative model was trained with a learning rate of 0.0001 and `AdamW` optimizer. For sampling, we employed the NeuralODE function of the `torchdyn` Python package. We set the solver type to `dopri5` with `adjoint` sensitivity and parameters `atol` and `rtol` set to `1e-5`. For batch Optimal Transport we use an exact solver from the `pot` package.

B EXPERIMENTAL DETAILS

On each dataset, we explore the following hyperparameters:

Table 2: Hyperparameters for training cellFlow.

Name	Values	Description
Covariate embedding size	[128, 256]	Size of cell type embedding
Embedding dimension	[128, 256]	Dimension of condition embeddings in the ResNet
Number of blocks	[2, 3, 4]	Number of ResNet blocks
Learning rate	[0.0001, 0.001]	Learning rate to train cellFlow
σ	[0, 0.0001, 0.001, 0.01, 0.1]	Variance of Gaussian around the straight line between matched samples
Batch size	[128, 256, 256]	Batch size for Flow Matching

We choose the best configuration based on the OT-CFM loss described in Eq. (7) on a left-out test set.

For metric computation in Table 1, we simulate as many cells as those in the real dataset for all methods. Both real and generated cells are reduced to 2000 highly variable genes and subsequently transoformed to a 50-PC matrix to remove noise. The PC matrices of both real and generated data are standardized and compared with the described metrics.

C METRICS DESCRIPTIONS

C.1 SQUARED WASSERSTEIN DISTANCE

The 2-Wasserstein distance, also known as the Wasserstein-2 or Euclidean Wasserstein distance, measures the dissimilarity between two probability distributions in terms of their optimal transportation cost. It quantifies how much "work" is needed to transform one distribution into the other,

with smaller distances indicating greater similarity between the distributions. The 2-Wasserstein distance is computed as the square root of the minimum transport cost among all possible mappings between the distributions, weighted by the Euclidean distance in the underlying space.

The formula for computing the 2-Wasserstein distance between two distributions P and Q with finite second moments is given by:

$$W_2(P, Q) = \left(\inf_{\gamma \in \Gamma(P,Q)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|^2 \, d\gamma(x, y) \right)^{1/2}$$

where $\Gamma(P, Q)$ represents the set of all joint probability distributions on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals P and Q, and $\| \cdot \|$ denotes the Euclidean norm.

C.2 Mean Maximum Discrepancy

The linear Mean Maximum Discrepancy (MMD) is a metric used to measure the distance between probability distributions. It evaluates dissimilarities between distributions by comparing their empirical means and covariances in a reproducing kernel Hilbert space (RKHS). The linear MMD is particularly suited for capturing differences in the first and second moments of distributions and is often used in tasks such as domain adaptation and two-sample testing.

The formula for computing the linear MMD between two distributions P and Q with finite second moments is given by:

$$\text{MMD}(P, Q) = \left\| \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{x}_i) - \frac{1}{m} \sum_{j=1}^{m} \phi(\mathbf{y}_j) \right\|_{\mathcal{H}}^2 \tag{11}$$

where $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ and $(\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_m)$ are samples drawn from distributions P and Q respectively, and ϕ represents the feature map that maps samples into a reproducing kernel Hilbert space $\mathcal{H}$.

C.3 Recall

Given a real and a generated dataset with $\mathbf{X}$ and $\mathbf{Y}$ with N and M observations, Recall (Sajjadi et al., 2018) is a metric used to measure the expected likelihood of real samples against a fake manifold. It is computed as:

$$\text{Recall} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{\mathbf{X}_i \in \text{manifold}(\mathbf{Y}_1, ..., \mathbf{Y}_M)} \tag{12}$$

where N is the number of real samples, $\mathbf{1}(\cdot)$ is the indicator function, and $\text{manifold}(\mathbf{y}_1, \ldots, \mathbf{y}_M)$ constructs manifolds using spheres around fake samples. In this context, a sphere centered at each fake sample $\mathbf{y}_i$ with a radius equal to the distance to its k-th nearest neighbor, excluding itself, forms the manifold. This construction enables the comparison of real samples against a neighborhood of fake samples.

D Baseline description

D.1 Splatter Zappia et al. (2017)

Splatter is a versatile tool for generating synthetic single-cell RNA sequencing (scRNA-seq) counts. It offers six different simulation models accessible through a user-friendly interface. Users can simulate various biological scenarios by adjusting parameters such as the number of cells, genes, and levels of biological variation and technical noise. The simulation process involves two steps: estimating parameters from real datasets and using these parameters to generate synthetic scRNA-seq data.

D.2 scGAN (Marouf et al., 2020)

The scGAN model is a Generative Adversarial Network (GAN) Goodfellow et al. (2014) tailored for realistic scRNA-seq data generation. It minimizes the Wasserstein distance between distributions of real and generated cells, utilizing a generator network to produce synthetic samples and a critic network for discrimination. The model employs fully connected layers, incorporates a custom library-size normalization (LSN) layer for stable training, and extends to conditional scGAN (cscGAN) for type-specific cell generation. Evaluation involves metrics like t-SNE and marker gene correlation to gauge generated cell quality.

D.3 ACTIVA (Heydari et al., 2022)

The ACTIVA model incorporates three primary networks, including a self-evaluating VAE at its core and a cell-type classifier as its conditioner. The objective functions of the model are formulated to train the encoder network as an adversary of the generator network, facilitating realistic data generation. The encoder network serves to map scRNA-seq data into an approximate posterior while acting as a discriminator to judge the quality of generated samples against training data. Various loss components, including KL divergence regularization, reconstruction error, and cell-type matching, are employed to train both the encoder and generator networks effectively. Additionally, an automated cell-type conditioning technique is introduced to encourage the generator to produce cells classified similarly to the input data. The model's flexibility allows for the use of any classifier as a conditioner, enabling adaptation to diverse classification tasks.

D.4 scDiffusion (Luo et al., 2024)

The scDiffusion model comprises an autoencoder, a diffusion backbone network, and a conditional classifier. The autoencoder transforms gene expression profiles into latent space embeddings, the diffusion backbone network learns the reverse diffusion process, and the conditional classifier guides cell generation under specific conditions. During training, the autoencoder creates embeddings from real data, followed by diffusion to generate noisy embeddings for backbone training, while the classifier predicts labels. In inference, the diffusion backbone denoises embeddings to produce new ones for gene expression data generation. The autoencoder addresses high-dimensional data and non-Gaussian distribution, while the diffusion backbone network utilizes fully connected layers and a residual structure. In the diffusion process, noise is iteratively added to embeddings, and during inference, noise is iteratively removed to generate new embeddings for final data generation.

E Data preprocessing and description

Single cells were pre-processed via the `Scanpy` (Wolf et al., 2018) software. Count normalization was applied only to baselines requiring real-valued data. In such settings, cells were normalized to sum to `1e4`. To train cellFlow we select 2000 highly variable genes per dataset using the function `scanpy.pp.highly_variable_genes` with flavor `seurat_v3`. Since cellFlow works in count space, the data is not normalized to train it.

F Computational resources

We utilized Python 3.10 for our model's implementation, employing PyTorch 2.0 for deep learning models. Our experiments were conducted separately on GPU servers with the following specifications:

- GPU: 16x Tesla V100 GPUs with 32GB RAM per card
- GPU: 2x Tesla V100 GPUs with 16GB RAM per card
- GPU: 8x A100-SXM4 GPUs with 40GB RAM per card