GENERATING MULTI-MODAL AND MULTI-ATTRIBUTE SINGLE-CELL COUNTS WITH CFGEN

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative modeling of single-cell RNA-seq data has proven instrumental for tasks like trajectory inference, batch effect removal, and gene expression generation. However, the most recent deep generative models simulating synthetic single cells from noise operate on pre-processed continuous gene expression approximations, overlooking the discrete nature of single-cell data, which limits their effectiveness and hinders the incorporation of robust noise models. Additionally, aspects like controllable multi-modal and multi-label generation of cellular data are underexplored. This work introduces Cell Flow for Generation (CFGen), a flow-based conditional generative model that accounts for the discrete nature of single-cell data. CFGen generates whole-genome multimodal single-cell counts reliably, improving the recovery of crucial biological data characteristics while tackling relevant generative tasks such as rare cell type augmentation and batch correction. We also introduce a novel framework for compositional data generation using Flow Matching. By showcasing CFGen on a diverse set of biological datasets and settings, we provide evidence of its value to the fields of computational biology and deep generative models.

1 INTRODUCTION

Single-cell transcriptomics has revolutionized the exploration of cellular heterogeneity, revealing critical biological processes and cellular states (Hwang et al., 2018; Rozenblatt-Rosen et al., 2017). Advances in single-cell RNA-seq (scRNA-seq) enable parallel gene expression profiling in millions of cells, resulting in large datasets that investigate cellular differentiation (Gulati et al., 2020), disease progression (Zeng & Dai, 2019), and responses to drug perturbation (Ji et al., 2021). Aware of the multi-faceted complexity of the molecular state of a cell, modern studies often complement the quantification of single-cell gene expression with additional measurements, such as information on the accessibility state of DNA regions encoding for molecular functions like gene expression and regulation (Lowe et al., 2019; Baysoy et al., 2023). Accessibility is measured using the Transposase-Accessible Chromatin assay (ATAC-seq) (Grandi et al., 2022), with open regions termed *peaks*. Recently, multi-modal single-cell datasets have emerged, providing a more comprehensive view of the biological states of cellular populations. Yet, technical bias and high experimental costs still hinder the homogeneous profiling of all possible cell states within the inspected biological process. In light of this, generative modeling offers a promising avenue to highlight underexplored biological states, drive scientific hypotheses, and enhance downstream applications (Lopez et al., 2018).

Generative models for single-cell data, in particular Variational Autoencoders (VAEs), have been extensively employed in representation learning (Lopez et al., 2018; Palma et al., 2023), perturbation prediction (Lotfollahi et al., 2019; 2023; Hetzel et al., 2022) and trajectory inference (Gayoso et al., 2023; Chen et al., 2022; Farrell et al., 2023). Recently, more complex approaches leveraging diffusion-based models (Luo et al., 2024) or Generative Adversarial Networks (GAN) (Marouf et al., 2020) have paved the way for the task of synthetic data generation, demonstrating promising performance on realistic single-cell data modeling. Single-cell data is inherently discrete, as gene expression is collected as the number of transcribed gene copies found experimentally. Due to the incompatibility of discrete data with continuous models such as Gaussian diffusion (Yang et al., 2023a), most approaches generate data that has been pre-processed through normalization and scaling. This limits their flexibility to support downstream tasks centered around raw counts, such as batch correction (Lopez et al., 2018) and analyses where the total number of transcripts per cell is important (Gulati et al., 2020). Additionally, technical and biological effects in single-cell counts have been formalized

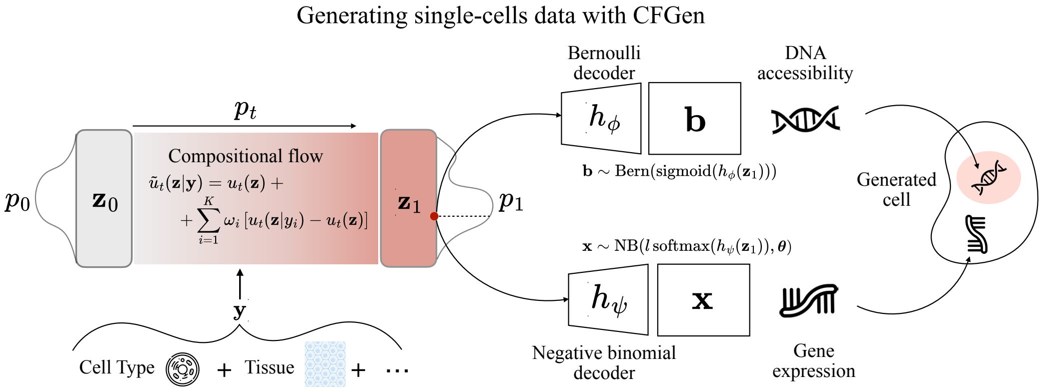
Generating single-cells data with CFGen
Bernoulli decoder
DNA accessibility
p_t
Compositional flow
$\tilde{u}_t(\mathbf{z}|\mathbf{y}) = u_t(\mathbf{z}) + \sum_{i=1}^{K} \omega_i \left[u_t(\mathbf{z}|y_i) - u_t(\mathbf{z})\right]$
p_0
$\mathbf{z}_0$
$\mathbf{z}_1$
p_1
h_ϕ
$\mathbf{b}$
$\mathbf{b} \sim \mathrm{Bern}(\mathrm{sigmoid}(h_\phi(\mathbf{z}_1)))$
Generated cell
$\mathbf{x} \sim \mathrm{NB}(l\,\mathrm{softmax}(h_\psi(\mathbf{z}_1)), \boldsymbol{\theta})$
h_ψ
$\mathbf{x}$
$\mathbf{y}$
Cell Type + Tissue + ···
Negative binomial decoder
Gene expression

Figure 1: The CFGen generative model. Gaussian noise from a prior p_0 is converted into a latent variable $\mathbf{z}_1$ sampled from a target density p_1 by a compositional flow conditioned on multiple biological and technical attributes. Decoders for gene expression and DNA accessibility map $\mathbf{z}_1$ to the parameters of negative binomial and Bernoulli likelihood models from which single-cell gene expression and DNA accessibility peaks are sampled.

under effective discrete noise models (Hafemeister & Satija, 2019), which should be incorporated into generative models for single-cell data to approximate the underlying data generation process better.

In this work, we present Cell Flow for Generation (CFGen) (Fig. 1), a conditional flow-based generative model designed to reproduce multi-modal single-cell discrete counts realistically. Our approach combines the expressiveness of recent Flow Matching techniques (Albergo & Vanden-Eijnden, 2022; Liu et al., 2022b; Lipman et al., 2023; Dao et al., 2023; Tong et al., 2024) with modeling the statistical properties of single-cell data across multiple modalities, each following a distinct discrete likelihood model. Moreover, we extend the current literature on Flow Matching by introducing the concept of *compositionality*, enabling the generation of cells conditioned on single attributes or combinations thereof in a controlled setting.

We demonstrate the value of our approach in terms of generative performance improvement and the enhancement of downstream tasks. In summary, we propose the following contributions:

- We introduce CFGen, a generative model for discrete multi-modal single-cell counts, which explicitly accounts for key statistical properties of single-cell data under a specified noise model.
- We extend the Flow Matching framework to incorporate guidance for compositional generation under multiple attributes.
- We demonstrate that our model's full-genome generative performance consistently outperforms existing single-cell generative models qualitatively and quantitatively on a diverse set of biological datasets.
- We showcase the application of CFGen in enhancing downstream tasks, including robust data augmentation for improved classification of rare cell types and batch correction.

2 RELATED WORK

The synthetic generation of single-cell datasets is a well-established research direction pioneered by models using standard probabilistic methods to estimate gene-wise parameters in a single modality (Zappia et al., 2017; Li & Li, 2019) or multiple modality setting (Song et al., 2024). With the advent of deep generative models, VAE-based approaches have demonstrated an extremely flexible performance, offering popular tools for batch correction (Lopez et al., 2018), modality integration (Gayoso et al., 2021), trajectory inference (Gayoso et al., 2023) and perturbation prediction (Lotfollahi et al., 2019; 2023; Hetzel et al., 2022). Despite their relevance, most of the mentioned approaches orient towards learning meaningful cellular representations or counterfactual predictions rather than generating synthetic datasets from noise. Such a task has instead been extensively explored by other works leveraging the expressive potential of diffusion models (Luo et al., 2024), Generative Adversarial Networks (GANs) (Marouf et al., 2020) and Large Language Models (LLMs) (Levine et al., 2023) to produce realistic cells approximating the observed data distribution. Our technical contribution builds upon Flow Matching (Albergo & Vanden-Eijnden, 2022; Liu et al., 2022b; Lipman et al., 2023), an efficient formulation of continuous normalizing flows for generative modeling. Since its release, Flow Matching has been successfully applied to optimal transport (Tong et al., 2024; Eyring et al., 2023; Pooladian et al., 2023), protein generation (Jing et al., 2024; Yim et al., 2023), interpolation on general geometries (Chen & Lipman, 2023; Kapusniak et al., 2024) and guided conditional generation (Zheng et al., 2023). Finally, Flow Matching showed promising performance in tasks involving scRNA-seq, such as learning cellular evolution across time (Tong et al., 2024; Kapusniak et al., 2024) and responses to drugs (Klein et al., 2023).

3 BACKGROUND

3.1 DEEP GENERATIVE MODELING OF SINGLE-CELL DATA

Single cells are represented as high-dimensional vectors of discrete counts, where each feature corresponds to a gene and its measurement reflects the number of transcripts detected in a cell. Technical bias and biological variation lead to unique statistical characteristics in cells, including *sparsity* and *over-dispersion*. Sparsity arises from genes not being expressed in specific cellular states (biological cause) or measurement dropouts in scRNA-seq (technical cause). Over-dispersion occurs when the gene-wise variance exceeds the mean. Over-dispersed counts are typically modeled using a Negative Binomial (NB) distribution, parameterized by a mean μ and an inverse dispersion parameter θ.

Formally, given a nonnegative count expression matrix $\mathbf{X} \in \mathbb{N}_0^{N \times G}$ with N cells and G genes, entries x_{ng} of the expression matrix are assumed to follow the negative binomial model:

$$x_{ng} \sim \mathrm{NB}(\mu_{ng}, \theta_g) \;, \tag{1}$$

where $\mu_{ng} \in \mathbb{R}_{\geq 0}$ is a cell-gene-specific mean and $\theta_g \in \mathbb{R}_{\geq 0}$ is the gene-specific inverse dispersion. Thus, we assume each cell has an individual mean, while over-dispersion is modeled gene-wise.

When scRNA-seq is coupled with information on DNA accessibility, transcription measurements are complemented by a binary matrix $\mathbf{B} \in \{0,1\}^{N \times P}$, where P is the number of DNA regions profiled for accessibility. Here, each dimension independently follows the Bernoulli model $b_{np} \sim \mathrm{Bern}(\pi_{np})$, with π_{np} indicating a cell-gene-specific success probability.

In most single-cell representation learning settings, a deep latent variable model is trained to map a latent space $\mathcal{Z}$ to the parameter space $\mathcal{H}$ of the noise model via a decoder maximizing the log-likelihood of the data. Given a latent cell state $\mathbf{z}$, the likelihood parameters of each modality are inferred as

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

where h_ψ and h_ϕ are modality-specific decoders and l is the size factor, defined as the total number of counts of the generated cell. The vector $\boldsymbol{\rho}$ represents gene expression proportions. One can interpret the parameterization of the scRNA-seq likelihood based on the connection between the negative binomial and Poisson-gamma distributions (see Appendix A.1).

3.2 CONTINUOUS NORMALIZING FLOWS AND FLOW MATCHING

Continuous Normalizing Flows (CNF). Chen et al. (2018) introduced CNFs as a generative model to approximate complex data distributions. Given data in a continuous domain $\mathcal{Z} \subset \mathbb{R}^d$, one can define a time-dependent probability path $p : [0,1] \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, transforming a tractable prior density p_0 into a more complex one p_1, where we indicate the probability path at time t as $p_t : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ such that $\int p_t(\mathbf{z})\, \mathrm{d}\mathbf{z} = 1$. The probability path is formally *generated* by a time-dependent vector field, $u_t : \mathbb{R}^d \to \mathbb{R}^d$ with $t \in [0,1]$ satisfying $\frac{\mathrm{d}p}{\mathrm{d}t} = -\nabla \cdot (p_t u_t)$. The field u_t is the time-derivative of an invertible *flow* $\phi_t : \mathbb{R}^d \to \mathbb{R}^d$ following the Ordinary Differential Equation (ODE) $\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(\mathbf{z}) = u_t(\phi_t(\mathbf{z}))$, where $\phi_0(\mathbf{z}) = \mathbf{z}$. Learning a CNF means defining a push-forward operator $p_t = [\phi_t]_* p_0$, transforming the prior p_0 into the data density p_1. In other words, learning the *target vector field* u_t generating p_t allows transporting samples from the prior distribution to realistic data samples by simulating the ODE in time.

Flow Matching (Lipman et al., 2023). Assume the goal is to model a complex data distribution q by learning the target vector field. One can marginalize the probability path p_t as $p_t(\mathbf{z}) = \int p_t(\mathbf{z}|\mathbf{z}_1) q(\mathbf{z}_1) \mathrm{d}\mathbf{z}_1$, where $\mathbf{z}_1$ indicates a sample from the data distribution q and

$p_t(\cdot|\mathbf{z}_1)$ is the *conditional probability path* transporting noise to $\mathbf{z}_1$ under the boundary conditions $p_0(\mathbf{z}|\mathbf{z}_1) = p_0(\mathbf{z})$ and $p_1(\mathbf{z}|\mathbf{z}_1) \approx \delta(\mathbf{z} - \mathbf{z}_1)$. Here, δ represents a delta probability with mass concentrated at the point $\mathbf{z}_1$. Note that, at $t = 1$, the marginal distribution p_1 approximates the data distribution q. Moreover, $p_t(\mathbf{z})$ is generated by a *marginal velocity field* $u_t(\mathbf{z})$ satisfying

$$u_t(\mathbf{z}) = \int u_t(\mathbf{z}|\mathbf{z}_1) \, \frac{p_t(\mathbf{z}|\mathbf{z}_1) q(\mathbf{z}_1)}{p_t(\mathbf{z})} \, \mathrm{d}\mathbf{z}_1, \tag{3}$$

where $u_t(\mathbf{z}|\mathbf{z}_1)$ is called *conditional vector field*. While directly regressing $u_t(\mathbf{z})$ is intractable, Lipman et al. (2023) show that minimizing the Flow Matching objective

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t \sim \mathcal{U}[0,1], q(\mathbf{z}_1), p_t(\mathbf{z}|\mathbf{z}_1)} \left[||v_{t,\xi}(\mathbf{z}) - u_t(\mathbf{z}|\mathbf{z}_1)||^2 \right] \tag{4}$$

corresponds to learning to approximate the marginal vector field $u_t(\mathbf{z})$ with the time-conditioned neural network $v_{t,\xi}$. Defining $p_t(\mathbf{z}|\mathbf{z}_1) = \mathcal{N}(\alpha_t \mathbf{z}_1, \sigma_t^2 \mathbf{I})$ with the functions α_t, σ_t controlling the noise schedule, $u_t(\mathbf{z}|\mathbf{z}_1)$ has a closed form, and Eq. (4) is tractable (see Appendix A.2 for more details). We define such a formulation *Gaussian marginal paths*.

Classifier-free guidance (Zheng et al., 2023). One can *guide* data generation on a condition y by learning the conditional marginal field $u_t(\mathbf{z}|y)$ via a neural network $v_{t,\xi}(\mathbf{z}|y)$. Given a guidance strength hyperparameter $\omega \in \mathbb{R}$, Zheng et al. (2023) show that generating data points following the vector field $\tilde{u}_t(\cdot|y) = (1 - \omega) u_t(\cdot) + \omega u_t(\cdot|y)$ approximates sampling from the distribution $\tilde{q}(\mathbf{z}|y) \propto q(\mathbf{z})^{1-\omega} q(\mathbf{z}|y)^{\omega}$, where $q(\mathbf{z})$ and $q(\mathbf{z}|y)$ are, respectively, the unconditional and conditional data distributions. Guidance can thus be achieved by combining conditional and unconditional vector fields learned simultaneously during training.

4 CFGEN

Our objective is to define a latent Flow-Matching-based generative model for discrete single-cell data, where each cell is measured through gene expression and DNA accessibility. Our model, CFGen, is flexible: It can handle single and multiple modalities. Moreover, it supports guiding generation conditioned on single or combinations of attributes without needing to train a different model for each. In what follows, we present the assumptions and generative process formulation in the uni-modal and multi-modal settings. We additionally illustrate our novel approach to compositional guidance.

4.1 UNI-MODAL AND SINGLE-ATTRIBUTE GENERATION

Let $\mathbf{X} \in \mathbb{N}_0^{N \times G}$ be a single-cell matrix where an observed single-cell vector is $\mathbf{x} \in \mathbb{N}_0^G$, with G being the number of genes. Additionally, let $\mathbf{y} \in \mathbb{N}_0^N$ be a categorical attribute (for example, cell type) associated with each observation. We also define $l = \sum_{g=1}^{G} x_g$ as the size factor of an individual cell $\mathbf{x}$.

The generative process. When the technical bias is negligible, we define the standard CFGen setting as the following generative model:

$$p(\mathbf{x}, \mathbf{z}, l, y) = p(\mathbf{x}|\mathbf{z}, l) p(\mathbf{z}|y, l) p(l) p(y) \ , \tag{5}$$

where $\mathbf{z}$ is a continuous latent variable modeling the cell state, and we assumed that (1) $\mathbf{x}$ is independent of y conditionally on $\mathbf{z}$ and l, and (2) l is independent of y. Crucially, Eq. (5) provides a standard formulation for the generative process, but the factorization choice is flexible and can be adjusted based on the properties of the data. Although partially related to existing VAE-based single-cell generative models, our proposed factorization is novel. We detail the relationship between Eq. (5) and existing generative models in Appendix A.5.

Modeling the distributions in Eq. (5). Each factor of Eq. (5) is modeled separately: $p(y)$ is a categorical distribution $\mathrm{Cat}(N_y, \boldsymbol{\pi}_y)$ where N_y is the number of categories and $\boldsymbol{\pi}_y$ a vector of N_y class probabilities, and $p(l) = \mathrm{LogNormal}(\mu_l, \sigma_l^2)$. The parameters of $p(y)$ and $p(l)$ can be learned as maximum likelihood estimates over the dataset (see Appendix B.4). Given an attribute class y and size factor l sampled from the respective distributions, $p(\mathbf{z}|y, l)$ is given by a conditional normalizing flow $\phi_t(\cdot\,|y, l)$ learned via Flow Matching with Gaussian marginal paths (see Section 3.2, Appendix A.2 and Appendix B.2 for the architecture details) transporting samples from $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to latent cell representations $\mathbf{z} = \mathbf{z}_1 = \phi_1(\mathbf{z}_0|y, l)$. Finally, given a latent variable $\mathbf{z}$ and a size factor l, $p(\mathbf{x}|\mathbf{z}, l)$ is a negative binomial distribution with mean parameterized by a decoder h_ψ as in Eq. (2) and inverse dispersion modeled by a global parameter $\boldsymbol{\theta}$. In practice, h_ψ and $\boldsymbol{\theta}$ are optimized before training the flow, together with an encoder f_η that maps the data to a latent space (more details in Appendix B.1). We outline the reasons for training the encoder and the flow separately in Appendix B.5.

Sampling in practice. To generate a cell using CFGen as illustrated in Eq. (5), we first sample a size factor l and a condition y (the latter to specify a class). We then integrate the parameterized vector field $v_{t,\xi}(\mathbf{z}_t|y,l)$ with $t \in [0,1]$, starting from $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0},\mathbf{I})$. Here, we imply $\mathbf{z}_t = \phi_t(\mathbf{z}_0|y,l)$, where ϕ_t is a flow as described in Section 3.2. We then take the simulated $\mathbf{z}_1 = \phi_1(\mathbf{z}_0|y,l)$ at $t=1$ as our latent $\mathbf{z}$ in Eq. (5). Finally, we sample $\mathbf{x} \sim \mathrm{NB}(l\,\mathrm{softmax}(h_\psi(\mathbf{z}_1)),\boldsymbol{\theta})$.

Size factor as a technical effect. When l is influenced by technical effect under a categorical covariate $c \in \{1,\dots,C\}$, we reformulate Eq. (5) as $p(\mathbf{x},\mathbf{z},l,y) = \frac{1}{C}\sum_c p(\mathbf{x}|\mathbf{z},l)p(\mathbf{z}|y,l)p(l|c)p(y)p(c)$, where we assume that $\mathbf{z}$ is independent of c given l (i.e., l contains all necessary technical effect information to guide the flow), and y is independent of c. The last assumption derives from our choice of y as an attribute encoding biological identity preserved across technical batches.

4.2 MULTI-MODAL AND SINGLE-ATTRIBUTE GENERATION

Let $\mathbf{X}$ and $\mathbf{y}$ be defined as in Section 4.1. In the multi-modal setting, we have additional access to a binary matrix $\mathbf{B} \in \{0,1\}^{N\times P}$ representing DNA region accessibility, with P being the number of measured peaks. Each sample is, therefore, a tuple ($\mathbf{x}$, $\mathbf{b}$, y), where $\mathbf{x}$ and $\mathbf{b}$ are realizations of different discrete noise models (negative binomial and Bernoulli). Following Eq. (2), both parameters of the negative binomial and Bernoulli likelihoods are functions of the same latent variable $\mathbf{z}$, encoding a continuous cell state shared across modalities. We write the likelihood term in Eq. (5) as

$$p(\mathbf{x},\mathbf{b}|\mathbf{z},l) \overset{(1)}{=} p(\mathbf{x}|\mathbf{z},l)p(\mathbf{b}|\mathbf{z},l) \overset{(2)}{=} p(\mathbf{x}|\mathbf{z},l)p(\mathbf{b}|\mathbf{z})\ , \tag{6}$$

where in (1) we use the fact that the likelihood of $\mathbf{x}$ and $\mathbf{b}$ are optimized disjointedly given $\mathbf{z}$, and in (2) that $\mathbf{b}$ is independent of the size factor l (see Eq. (2)). In simple terms, all the modalities are encoded to the same latent space $\mathcal{Z}$ used to train the conditional flow approximating $p(\mathbf{z}|y,l)$ in Eq. (5) (more details in Appendix B.1). During generation, separate decoders h_ψ and h_ϕ map a sampled latent state $\mathbf{z}$ to the parameter spaces of the negative binomial and Bernoulli distributions, representing expression counts and binary DNA accessibility information, respectively (Fig. 1).

4.3 GUIDED COMPOSITIONAL GENERATION WITH MULTIPLE ATTRIBUTES

We extend classifier-free guidance (CFG) for Flow Matching (Zheng et al., 2023) to handle multiple attributes, enhancing control over the generative process in targeted data regions. This is especially relevant in scRNA-seq, where datasets are defined by several biological and technical covariates. Here, $\mathbf{Y} \in \mathbb{N}_0^{N\times K}$ represents a matrix of K categorical attributes. Instead of training a conditional flow model for every attribute combination, we propose a flexible approach that generates cells by composing multiple flow models, each conditioned on a single attribute.

Let $q(\mathbf{z}|\mathbf{y})$ be the conditional data distribution, with $\mathbf{y} = (y_1,\dots,y_K)$ being a collection of observed categorical attributes. In analogy with CFG in diffusion models (Ho & Salimans, 2022), we aim to implement a generative model to sample from $\tilde{q}(\mathbf{z}|\mathbf{y}) \propto q(\mathbf{z})\prod_{i=1}^{K}\left[\frac{q(\mathbf{z}|y_i)}{q(\mathbf{z})}\right]^{\omega_i}$, where ω_i is the guidance strength for attribute i (see Appendix A.4). Diffusion models generate data by learning to approximate the score of the time-dependent distribution, $\nabla_{\mathbf{z}}\log p_t(\mathbf{z}|\mathbf{y})$, with a neural network and using it to simulate a reverse diffusion Stochastic Differential Equation (SDE) transporting noise samples to generated data observations (Song et al., 2021). Importantly, the reverse diffusion SDE corresponds to a deterministic *probability flow ODE* with the same time-marginal distributions (Yang et al., 2023b).

Based on the relation between score-based diffusion and energy models, Liu et al. (2022a) demonstrated that compositional classifier-free guidance is achievable through expressing the drift of the generating reverse SDE with the compositional score:

$$\nabla_{\mathbf{z}}\log\tilde{p}_t(\mathbf{z}|\mathbf{y}) = \nabla_{\mathbf{z}}\log p_t(\mathbf{z}) + \sum_{i=1}^{K}\omega_i[\nabla_{\mathbf{z}}\log p_t(\mathbf{z}|y_i) - \nabla_{\mathbf{z}}\log p_t(\mathbf{z})]\ . \tag{7}$$

Following the direct relationship between Flow Matching and CFG provided in Ho & Salimans (2022), we build the Flow Matching counterpart to Eq. (7).

Provided that we have access to Flow Matching models for the unconditional marginal vector field $u_t(\mathbf{z})$ and the single-attribute conditional fields $\{u_t(\mathbf{z}|y_i)\}_{i=0}^{K}$, the following holds:

Table 1: Quantitative performance comparison between conditional and unconditional single-cell generative models. Evaluation is performed based on distribution matching metrics (RBF-kernel MMD and 2-Wasserstein distance) and the generated cell type classification F1 score achieved by a kNN classifier trained on real data. Results are averaged across generations performed using three different seeds.

	PBMC3K			Dentate			T. Muris			HLCA		
	MMD (↓)	WD (↓)	KNNc (↑)	MMD (↓)	WD (↓)	KNNc (↑)	MMD (↓)	WD (↓)	KNNc (↑)	MMD (↓)	WD (↓)	KNNc (↑)
Cond.												
c-CFGen	**0.85 ± 0.05**	**16.94 ± 0.44**	0.36 ± 0.15	**1.12 ± 0.04**	**21.55 ± 0.17**	0.21 ± 0.03	**0.19 ± 0.02**	**7.39 ± 0.20**	**0.15 ± 0.03**	**0.54 ± 0.02**	**10.72 ± 0.08**	**0.18 ± 0.02**
scDiff.	1.27 ± 0.20	22.41 ± 1.21	0.23 ± 0.06	1.22 ± 0.05	22.56 ± 0.10	**0.22 ± 0.03**	0.24 ± 0.04	7.89 ± 0.45	0.12 ± 0.01	0.96 ± 0.04	15.82 ± 0.45	0.08 ± 0.01
scVI	0.94 ± 0.05	17.66 ± 0.29	**0.43 ± 0.15**	1.15 ± 0.04	22.61 ± 0.23	0.19 ± 0.03	0.26 ± 0.02	9.76 ± 0.53	0.10 ± 0.02	0.58 ± 0.02	11.78 ± 0.19	0.09 ± 0.01
Uncond.												
u-CFGen	0.44 ± 0.01	16.81 ± 0.06	-	0.42 ± 0.01	**21.20 ± 0.02**	-	**0.08 ± 0.00**	**8.54 ± 0.06**	-	**0.15 ± 0.01**	**10.63 ± 0.01**	-
scGAN	**0.36 ± 0.01**	**15.54 ± 0.06**	-	0.42 ± 0.01	22.52 ± 0.03	-	0.25 ± 0.00	12.85 ± 0.04	-	0.18 ± 0.01	10.81 ± 0.01	-

Proposition 1 *If the attributes $y_1, ..., y_K$ are conditionally independent given $\mathbf{z}$, the vector field*

$$\tilde{u}_t(\mathbf{z}|\mathbf{y}) = u_t(\mathbf{z}) + \sum\nolimits_{i=1}^{K} \omega_i[u_t(\mathbf{z}|y_i) - u_t(\mathbf{z})] \tag{8}$$

coincides with the velocity of the probability-flow ODE associated with the generative SDE of a diffusion model with a compositional score as in Eq. (7).

We provide a proof for Proposition 1 in Appendix A.4. In other words, the reversed diffusion SDE from compositional CFG admits a deterministic probability flow ODE with velocity as in Eq. (8). Consequently, classifier-free sampling from compositions of attributes is achievable by simulating the probability path $\tilde{p}_t(\mathbf{z}|\mathbf{y}) \propto p_t(\mathbf{z}) \prod_{i=1}^{K} \left[\frac{p_t(\mathbf{z}|y_i)}{p_t(\mathbf{z})}\right]^{\omega_i}$ from a prior Gaussian density p_0 integrating the parameterized field $\tilde{v}_{t,\xi}(\mathbf{z}|\mathbf{y}) = v_{t,\xi}(\mathbf{z}) + \sum_{i=1}^{K} \omega_i[v_{t,\xi}(\mathbf{z}|y_i) - v_{t,\xi}(\mathbf{z})]$. Note that both conditional and unconditional fields are parameterized by the same model, which is learned by providing single-attribute conditioning with a certain probability during training (see Algorithms 1 and 2).

5 EXPERIMENTS

In this section, we compare CFGen with existing models in uni-modal (Section 5.1) and multi-modal (Section 5.2) generation across five datasets. We evaluate quantitatively by measuring distributional proximity between real and generated cells, and qualitatively by assessing how well models capture real data properties. In Section 5.3, we show the effectiveness of multi-attribute generation in guiding synthetic samples toward specific biological labels and donors. Lastly, in Sections 5.4 and 5.5, we demonstrate that CFGen enhances rare cell type classification through targeted data augmentation and performs batch correction on par with widely used VAE-based models.

5.1 COMPARISON WITH EXISTING METHODS

Baselines. We choose scVI (Gayoso et al., 2021) and scDiffusion (Luo et al., 2024) as conditional models and scGAN (Marouf et al., 2020) as unconditional baseline. The scVI model is based on a VAE architecture with a negative binomial decoder and performs generation by decoding low-dimensional Gaussian noise into parameters of the likelihood model. Conversely, scDiffusion and scGAN operate on a continuous-space domain, performing generation using standard latent diffusion (Rombach et al., 2022) and GAN (Goodfellow et al., 2020) models. Thus, we train them using normalized counts (more in Appendix D).

Datasets. We assess model performance on four datasets of varying size: **(i)** PBMC3K[1] (2,700 cells from a healthy donor, clustering into 8 cell types), **(ii)** Dentate gyrus (La Manno et al., 2018) (18,213 cells from a developing mouse hippocampus), **(iii)** Tabula Muris (tab, 2018) (245,389 cells from mouse hippocampus across multiple tissues), and **(iv)** Human Lung Cell Atlas (HLCA) (Sikkema et al., 2023) (584,944 cells from a size-controlled version of 2.4 million cells from 486 individuals across 49 datasets). Conditioning is performed on *cell type* for all datasets except Tabula Muris, where we use the *tissue* label. Dataset descriptions and pre-processing details are in Appendix F and Table 4.

Quantitative evaluation. As evaluation metrics, we use distribution distances (RBF-kernel MMD (Borgwardt et al., 2006) and Wasserstein-2 distance) computed between the Principal Component (PC) projections of real and generated data in 30 dimensions. The generated data is embedded

---

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

using the PC loadings of the real data for comparability. For conditional models, we additionally evaluate the cell-type classification F1 score of a k-Nearest-Neighbors (kNN) classifier trained on real data and evaluated on generated cells. All evaluations are performed on a held-out set of cells, considering the whole genome, with a filtering step for low expression genes (see Appendix F).

Quantitative results. In Table 1, we evaluate the generative performance of CFGen conditionally (c-CFGen) and unconditionally (u-CFGen) against three baselines on the scRNA-seq generation task. CFGen always reaches either the top or second-best performance on the conditional generation task and overcomes scGAN on three out of four datasets.

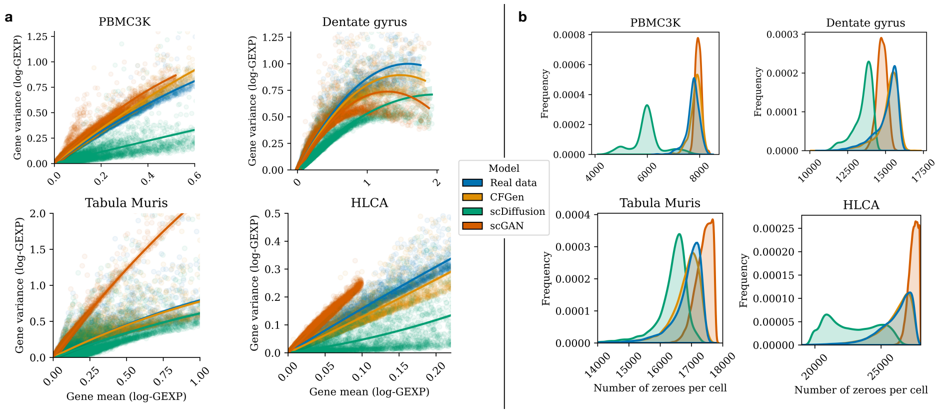

Figure 2: **(a)** Comparison between the gene-wise empirical mean-variance trend in real data and samples from generative models. **(b)** Frequency of the number of zeroes per cell in real and generated data.

Qualitative evaluation. Evaluating realistic data generation in biology requires more than distribution matching metrics. CFGen surpasses diffusion and GAN-based single-cell generators by explicitly modeling the probabilistic properties of the data. By learning a rigorous likelihood-based model for discrete data (Eq. (5)), our generated cells align closely with real observations in key aspects: **(1) Sparsity:** caused by technical biases in gene transcript detection or gene inactivity in specific contexts. **(2) Over-dispersion:** scRNA-seq data exhibits a nonlinear mean-variance relationship, modeled through the inverse dispersion parameter of a negative binomial distribution. **(3) Discreteness and skewness:** scRNA-seq data is discrete and highly skewed toward zero.

Qualitative results. In Fig. 2, we provide qualitative evidence that CFGen is more effective in recovering properties (1) and (2) compared to scDiffusion and scGAN, which assume a continuous data space. Property (3) naturally follows when modeling discrete counts with CFGen. Specifically, Fig. 2a shows that explicitly modeling counts with gene-specific inverse dispersion leads to better alignment of the generated gene-wise mean-variance relationship with real data. Additionally, Fig. 2b demonstrates near-perfect recovery of the actual distribution of zero counts per cell. In contrast, scDiffusion often shifts towards actively expressed genes, while scGAN tends to either under or overestimate data sparsity. Furthermore, CFGen is the only conditional model capable of generating plausible synthetic cells in terms of overlap with the real data distribution for large datasets such as the HLCA and Tabula Muris (see Appendix H.4, Appendix H.2 and Fig. A7).

5.2 MULTI-MODAL GENERATION

We evaluate the qualitative and quantitative performance of CFGen at generating multi-modal data comprising gene expression and binary DNA-region accessibility.

Baselines. We compare CFGen with a VAE-based multi-modal generative model (MultiVI) (Ashuach et al., 2023). For completeness, we include as baselines two single-modality generative models: PeakVI (Ashuach et al., 2022) (DNA accessibility) and scVI (Lopez et al., 2018) (gene expression). Furthermore, we consider scDiffusion as a baseline for scRNA-seq generation.

Datasets. We use the multiome PBMC10K dataset, made available by 10X Genomics [2]. Here, each cell is measured both in gene expression and peak accessibility. The dataset consists of 10,000 cells

---

[2] `https://www.10xgenomics.com/support/single-cell-multiome-atac-plus-gene-expression`

across 25,604 genes and 40,086 peaks and was annotated with 15 cell types, with their respective marker peaks (enriched in accessible or inaccessible points) and genes.

Evaluation. We use the RBF-kernel MMD and Wasserstein-2 distances in the same setting described in Section 5.1. Before comparison, we normalize both real and generated binary measurements of DNA accessibility via TF-IDF (Aizawa, 2003) (in analogy to text mining). The metrics are computed in a 30-dimensional PC projection of the data. RNA counts are treated as in Section 5.1. In Appendix H.5 we compare CFGen and MultiVI more biologically. Specifically, we assess how well they approximate per-cell-type marker peak accessibility and gene expression (see Appendix G.5). For each cell type, we compute the accessibility fraction and mean expression of literature-derived marker peaks and genes in both real and generated cells and report their correlation per cell type in Fig. A8b.

Table 2: Comparison between CFGen, scDiffusion and VAE-based models on generating multiple single-cell modalities. We report distribution distance performance (RBF-kernel MMD and Wasserstein-2 distance) between real and generated cells across three seeds. Underlined values indicate the second-best performance.

	RNA		ATAC	
	MMD (↓)	WD (↓)	MMD (↓)	WD (↓)
CFGen multi.	0.89± 0.02	**13.90 ± 0.07**	**0.92 ± 0.02**	**18.86 ± 0.37**
CFGen RNA	**0.86 ± 0.02**	14.30 ± 0.08	-	-
scDiff.	1.02 ± 0.02	14.82 ± 0.11	-	-
MultiVI	**0.86 ± 0.03**	15.92 ± 0.25	0.96 ± 0.03	21.09 ± 0.34
PeakVI	-	-	1.49 ± 0.02	20.84 ± 0.45
scVI	0.96 ± 0.03	14.38 ± 0.11	-	-

Results. CFGen outperforms both MultiVI and PeakVI in modeling accessibility data based on distribution matching metrics (see Table 2). When considering the RNA modality, our model surpasses scVI and scDiffusion in all metrics and MultiVI in terms of Wasserstein-2 distance. Qualitatively, Fig. A8a shows substantial overlap between real and generated modalities. Finally, Fig. A8b demonstrates that CFGen better approximates average marker peak accessibility and gene expression, outperforming MultiVI across all cell type categories.

5.3 MULTI-ATTRIBUTE GENERATION AND GUIDANCE

We qualitatively assess our approach to compositional guidance, as outlined in Section 4.3.

Datasets. We showcase guidance on datasets with extensive technical variation, as one could combine different levels of biological and technical annotations to either augment rare cell type and batch combinations or control for the amount of technical effect added in simulation settings. Specifically, we consider **(i)** The NeurIPS 2021 dataset (Luecken et al., 2021a) - 90,261 bone marrow cells from 12 healthy human donors. We use donor as a batch attribute and cell type as a biological covariate. We also consider **(ii)** the Tabular Muris dataset described in Section 5.1, using tissue and Mouse ID as covariates.

Evaluation. The power of our guidance model is to generate data conditionally on an arbitrary subset of attributes—including unconditional generation—using a *single trained model*. For each pair of covariates (y_i, y_j), we evaluate generation on 500 generated cells varying the parameter ω_j, keeping ω_i fixed. The expected result is conditional generation on ω_i when $\omega_j = 0$ and generation from the intersection between the two attributes as ω_j increases. We additionally test the unconditional model, given by $\omega_i = \omega_j = 0$, expected to recover the whole single-cell dataset. In the unconditional case, we generate as many cells as there are in the dataset to better evaluate the coverage.

Results. Visual guidance results are shown in Fig. 3, with examples of double-attribute guidance between CD14+ monocytes and donor 1 for the NeurIPS 2021 dataset and tongue and mouse 18-M-52 for Tabula Muris. Unconditional generation recreates the original data (left-hand side) for both datasets. Setting guidance weights to zero for mouse ID and donor attributes leads to single-attribute conditional generation. Increasing the guidance weight steers the generation to the intersection of the two attributes. In Section 5.5, we show how multi-attribute guidance applies to batch correction. Quantitative results on attribute intersection generation quality are in Appendix H.8.

5.4 APPLICATION: DATA AUGMENTATION TO IMPROVE CLASSIFICATION OF RARE CELL TYPES

We propose using CFGen to improve cell-type classifier generalization by augmenting rare cell types in datasets. Previous work has shown that large auxiliary datasets enhance performance (Richter et al., 2024). Here, we use scGPT (Cui et al., 2024), a transformer pre-trained on 33 million cells.

Datasets. We leverage two large datasets: **(i)** PBMC COVID (Yoshida et al., 2022) - 422,220 blood cells from 93 patients ranging across paediatric and adult. **(ii)** The HLCA dataset described in Section 5.1. Both datasets are processed by selecting 2000 highly variable features and holding out cells from 20% of the donors.

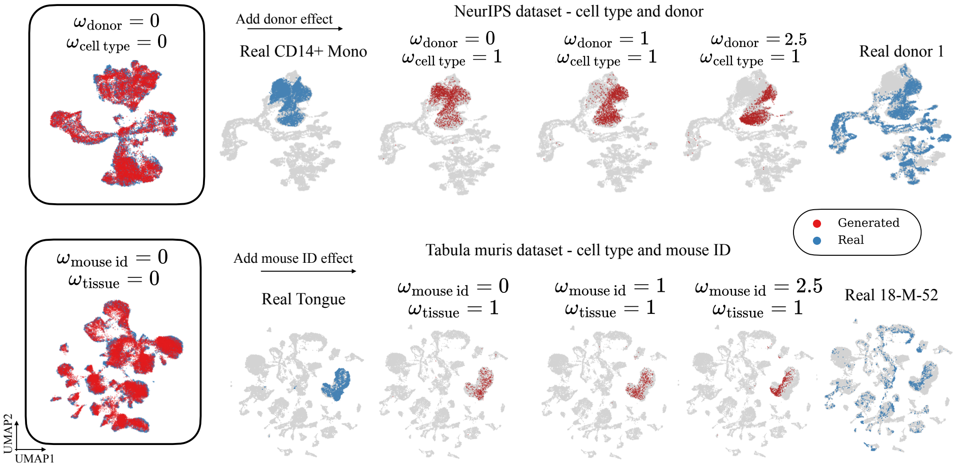

Figure 3: Qualitative evaluation of guidance performance on attribute pairs in the NeurIPS 2021 and Tabula Muris datasets. Left: unconditional performance with guidance weights at 0. Moving right: simulate 500 cells, progressively increasing the guidance strength of one attribute while keeping the counterpart unchanged.

Evaluation. We train CFGen on the PBMC COVID and HLCA training sets and successively augment both to 800,000 samples by upsampling rare cell types. For each cell type, we compute $\frac{1}{N_{\text{ct}}}$, where N_{ct} is the total number of cells from a cell type `ct`. We then generate observations to fill the gap between the dataset size and 800,000 cells, sampling cells proportional to the inverse of their cell type frequency. This process yields significantly more observations for rare cell types. However, we still do not reach uniformity, as class imbalance may be biologically meaningful. Following the original publication, we train kNN cell-type classifiers on scGPT's embeddings from the original and augmented training sets, evaluating generalization performance on held-out donors. For each cell type, we assess if performance increases upon augmentation as a function of its frequency in the dataset.

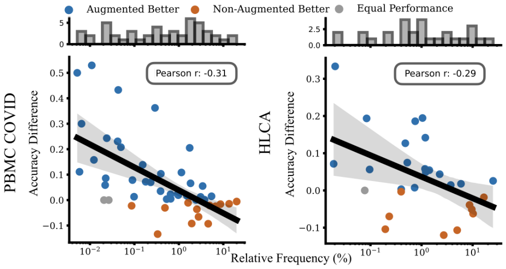

Figure 4: Cell-type classification difference as a function of cell type frequency before and after augmentation on PBMC COVID and HLCA datasets. A 10-neighbor kNN classifier is trained on the scGPT model's feature space. The held-out set includes cells from 20% of donors.

Results. Our results are displayed in Fig. 4 for the two datasets. Remarkably, most cell types in the held-out dataset are better classified after augmentation, suggesting that CFGen not only generates reliable cell samples but can be a valuable supplement to relevant downstream tasks. Moreover, the performance difference between before and after augmentation is inversely proportional to the frequency of the cell type in the dataset. Therefore, the improvement in generalization is more accentuated for rare cell types. Additionally, Fig. A11 in the Appendix shows that augmentation via CFGen outperforms the competing methods at improving the generalization performance on rare cell types in unseen donors (raw cell type accuracies are in Table 9 and Table 10).

5.5 APPLICATION: BATCH CORRECTION

We apply multi-attribute CFGen to batch correction (see Fig. 5), a common use case for generative models in scRNA-seq (Tran et al., 2020; Luecken et al., 2021b). Given a dataset with batch labels, we choose a reference batch $y^{\text{ref}}_{\text{batch}}$. For a latent cell $\mathbf{z}_j$ with attributes $y^{(j)}_{\text{batch}}$ and $y^{(j)}_{\text{cell type}}$, we invert the generative flow to remove the attribute structure. Next, we simulate the forward flow from the obtained representations while fixing the cell type and assigning $y^{\text{ref}}_{\text{batch}}$ to all observations. Guidance weights regulate the preservation of biological versus batch labels.

Datasets. We evaluate CFGen as a batch correction method on two datasets: **(i)** The NeurIPS dataset described in Section 4.3, using *cell type* as a biological variable to preserve and *acquisition site* as batch variable. **(ii)** The C. Elegans molecular atlas (Packer et al., 2019), which profiles 89,701 cells across 7 sources (batches). Similarly to (i), we use cell type as a biological annotation.

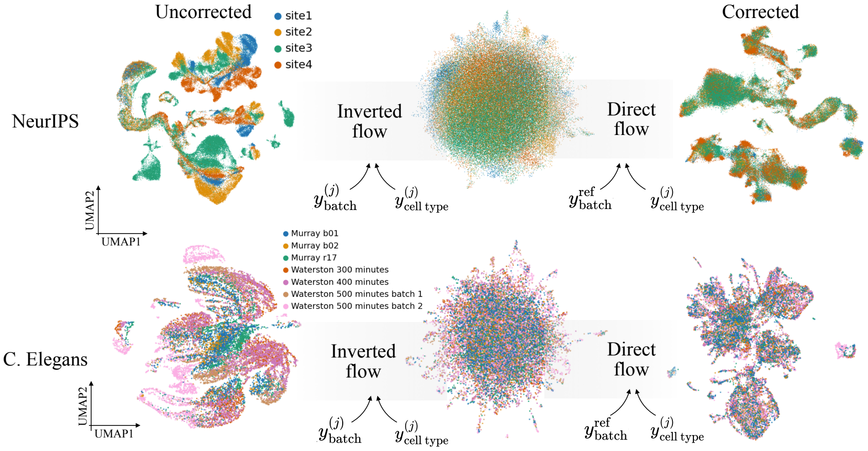

Figure 5: To perform batch correction, the scRNA-seq data distribution is mapped to the prior distribution by inverting the flow model. The resulting points are then transported back to the data domain based on a common reference batch label and the original cell type label to preserve the biological structure.

Evaluation. We compare our model with established VAE-based integration methods: scanVI (Xu et al., 2021), scVI (Lopez et al., 2018), and scPoli (De Donno et al., 2023). Using scIB metrics (Luecken et al., 2022), we assess batch correction and biological conservation based on neighborhood composition in the embedding space (see Appendix G.8). All methods are evaluated on a 50-dimensional latent space, with scores from the PC representation of uncorrected data included for comparison. We find that setting $\omega_{\text{batch}} = 1$ and $\omega_{\text{cell type}} = 2$ for C.Elegans and $\omega_{\text{batch}} = 2$ and $\omega_{\text{cell type}} = 1$ for NeurIPS preserves cell type variation while correcting for technical variation (see Appendix G.8 for more details on the selection).

Table 3: Average batch correction and biological conservation metrics from the scIB package comparing CFGen with VAE-based batch correction models in a 50-dimensional representation space. PC projections of the data are used to evaluate the uncorrected data.

	NeurIPS		C. Elegans	
	Batch (↑)	Bio (↑)	Batch (↑)	Bio (↑)
CFGen	**0.61**	0.64	**0.68**	**0.63**
scPoli	0.52	0.64	0.61	0.56
scanVI	0.48	**0.68**	0.61	0.59
scVI	0.44	0.63	0.58	0.55
Uncorrected	0.33	0.62	0.40	0.53

Results. The conceptualization of our technical effect correction approach together with qualitative results are shown in Fig. 5, highlighting batch mixing performance on the two datasets. In Table 3, we compare CFGen with baseline models for batch correction. Our model outperforms the others, achieving 9% and 6% overall improvements over the second-best models on the two datasets. Additionally, the biological conservation score is on par with scVI and scPoli. Despite failing to properly correct batch effects in the NeurIPS dataset, scanVI reaches a higher cell type preservation across datasets since it incorporates explicit cell type label information via a latent classifier (Appendix D.2)

6 CONCLUSION

We presented CFGen, a conditional latent flow-based generative model for single-cell discrete data that combines state-of-the-art generative models with rigorous probabilistic considerations. CFGen incorporates established noise models to sample realistic gene expression and DNA accessibility states, with promising applications in data augmentation and batch correction tasks. Furthermore, our model demonstrates improved performance over existing generative frameworks, reproducing data more faithfully across modalities. Our core machine learning contribution extends classifier-free guidance in Flow Matching with compositional generation of multiple attributes. Overall, CFGen represents a significant advancement in the simulation and augmentation of single-cell data, offering the research community powerful tools to support biological analysis.

Limitations. Our framework relies on multiple assumptions, including independence in the data, which may not hold in all biological contexts. Thus, exploring data characteristics is essential before using CFGen for generation. Furthermore, we currently train the autoencoder-based representation framework separately from the generative flow, which can be inefficient and memory-intensive.

7 ETHICS STATEMENT

This work explores the core features of scRNA-seq data and examines how capturing complex, high-dimensional cellular information can assist in answering biological questions. We aim to release CFGen as a user-friendly, open-source tool to facilitate its adoption in single-cell analysis. Given its application in biological research, CFGen may be utilized in sensitive environments that involve clinical data and patient information.

8 REPRODUCIBILITY STATEMENT

Reproduction details are reported in the Appendix and main text. The proof for Proposition 1 is extensively described in Appendix A.4, while prior knowledge on Flow Matching and classifier-free guidance is provided in Appendix A.2 and Appendix A.3. Algorithms for training and sampling with CFGen are reported in Appendix A.6. We introduce a thorough model description of both autoencoder and flow components together with modeling choices in Appendix B. Baselines and their characteristics are reported in Appendix D. All datasets are publicly available and their source publications are referenced in the main text. We additionally synthesize dataset characteristics in Table 4. Metrics and experimental setups are detailed in Appendix G. Finally, we report our computational infrastructures in Appendix E.

REFERENCES

Single-cell transcriptomics of 20 mouse organs creates a tabula muris. *Nature*, 562(7727):367–372, October 2018. ISSN 1476-4687. doi: 10.1038/s41586-018-0590-4. URL `http://dx.doi.org/10.1038/s41586-018-0590-4`.

Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing and; Management*, 39(1):45–65, January 2003. ISSN 0306-4573. doi: 10.1016/s0306-4573(02)00021-3. URL `http://dx.doi.org/10.1016/S0306-4573(02)00021-3`.

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Tal Ashuach, Daniel A. Reidenbach, Adam Gayoso, and Nir Yosef. Peakvi: A deep generative model for single-cell chromatin accessibility analysis. *Cell Reports Methods*, 2(3):100182, 2022. ISSN 2667-2375. doi: https://doi.org/10.1016/j.crmeth.2022.100182. URL `https://www.sciencedirect.com/science/article/pii/S2667237522000376`.

Tal Ashuach, Mariano I Gabitto, Rohan V Koodli, Giuseppe-Antonio Saldi, Michael I Jordan, and Nir Yosef. Multivi: deep generative model for the integration of multimodal data. *Nature Methods*, 20(8):1222–1231, 2023. ISSN 1548-7105. doi: https://doi.org/10.1038/s41592-023-01909-9. URL `https://doi.org/10.1038/s41592-023-01909-9`.

Alev Baysoy, Zhiliang Bai, Rahul Satija, and Rong Fan. The technological landscape and applications of single-cell multi-omics. *Nature Reviews Molecular Cell Biology*, 24(10):695–713, 2023. ISSN 1471-0080. doi: https://doi.org/10.1038/s41580-023-00615-w. URL `https://doi.org/10.1038/s41580-023-00615-w`.

Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola. Integrating structured biological data by Kernel Maximum Mean Discrepancy. *Bioinformatics*, 22(14):e49–e57, 07 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl242. URL `https://doi.org/10.1093/bioinformatics/btl242`.

Danila Bredikhin, Ilia Kats, and Oliver Stegle. Muon: multimodal omics analysis framework. *Genome Biology*, 23(1):42, 2022.

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 (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf`.

Ricky TQ Chen and Yaron Lipman. Riemannian flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023.

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. *Science Advances*, 8(48), December 2022. ISSN 2375-2548. doi: 10.1126/sciadv.abq3745. URL `http://dx.doi.org/10.1126/sciadv.abq3745`.

Pietro E Cippà and Thomas F Mueller. A first step toward a cross-tissue atlas of immune cells in humans. *Transplantation*, 107(1):8–9, 2023.

Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, Nan Duan, and Bo Wang. scgpt: toward building a foundation model for single-cell multi-omics using generative ai. *Nature Methods*, pp. 1–11, 2024. ISSN 1548-7105. doi: https://doi.org/10.1038/s41592-024-02201-0. URL `https://doi.org/10.1038/s41592-024-02201-0`.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

Carlo De Donno, Soroor Hediyeh-Zadeh, Amir Ali Moinfar, Marco Wagenstetter, Luke Zappia, Mohammad Lotfollahi, and Fabian J. Theis. Population-level integration of single-cell datasets enables multi-scale analysis across samples. *Nature Methods*, 20(11):1683–1692, October 2023. ISSN 1548-7105. doi: 10.1038/s41592-023-02035-2. URL `http://dx.doi.org/10.1038/s41592-023-02035-2`.

Luca Eyring, Dominik Klein, Théo Uscidda, Giovanni Palla, Niki Kilbertus, Zeynep Akata, and Fabian Theis. Unbalancedness in neural monge maps improves unpaired domain translation. *arXiv preprint arXiv:2311.15100*, 2023.

Spencer Farrell, Madhav Mani, and Sidhartha Goyal. Inferring single-cell transcriptomic dynamics with structured latent gene expression dynamics. *Cell Reports Methods*, 3(9):100581, September 2023. ISSN 2667-2375. doi: 10.1016/j.crmeth.2023.100581. URL `http://dx.doi.org/10.1016/j.crmeth.2023.100581`.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL `http://jmlr.org/papers/v22/20-451.html`.

Adam Gayoso, Zoë Steier, Romain Lopez, Jeffrey Regier, Kristopher L Nazor, Aaron Streets, and Nir Yosef. Joint probabilistic modeling of single-cell multi-omic data with totalvi. *Nature methods*, 18 (3):272–282, 2021. ISSN 1548-7105. doi: https://doi.org/10.1038/s41592-020-01050-x. URL `https://doi.org/10.1038/s41592-020-01050-x`.

Adam Gayoso, Philipp Weiler, Mohammad Lotfollahi, Dominik Klein, Justin Hong, Aaron Streets, Fabian J. Theis, and Nir Yosef. Deep generative modeling of transcriptional dynamics for rna velocity analysis in single cells. *Nature Methods*, 21(1):50–59, September 2023. ISSN 1548-7105. doi: 10.1038/s41592-023-01994-w. URL `http://dx.doi.org/10.1038/s41592-023-01994-w`.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11): 139–144, oct 2020. ISSN 0001-0782. doi: 10.1145/3422622. URL `https://doi.org/10.1145/3422622`.

Fiorella C Grandi, Hailey Modi, Lucas Kampman, and M Ryan Corces. Chromatin accessibility profiling by atac-seq. *Nature protocols*, 17(6):1518–1552, 2022. ISSN 1750-2799. doi: https://doi.org/10.1038/s41596-022-00692-9. URL `https://doi.org/10.1038/s41596-022-00692-9`.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL `http://jmlr.org/papers/v13/gretton12a.html`.

Gunsagar S. Gulati, Shaheen S. Sikandar, Daniel J. Wesche, Anoop Manjunath, Anjan Bharadwaj, Mark J. Berger, Francisco Ilagan, Angera H. Kuo, Robert W. Hsieh, Shang Cai, Maider Zabala, Ferenc A. Scheeren, Neethan A. Lobo, Dalong Qian, Feiqiao B. Yu, Frederick M. Dirbas, Michael F. Clarke, and Aaron M. Newman. Single-cell transcriptional diversity is a hallmark of developmental potential. *Science*, 367(6476):405–411, January 2020. ISSN 1095-9203. doi: 10.1126/science.aax0249. URL `http://dx.doi.org/10.1126/science.aax0249`.

Christoph Hafemeister and Rahul Satija. Normalization and variance stabilization of single-cell rna-seq data using regularized negative binomial regression. *Genome Biology*, 20(1), December 2019. ISSN 1474-760X. doi: 10.1186/s13059-019-1874-1. URL `http://dx.doi.org/10.1186/s13059-019-1874-1`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Leon Hetzel, Simon Boehm, Niki Kilbertus, Stephan Günnemann, Mohammad Lotfollahi, and Fabian Theis. Predicting cellular responses to novel drug perturbations at a single-cell resolution. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 26711–26722. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/aa933b5abc1be30baece1d230ec575a7-Paper-Conference.pdf`.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Byungjin Hwang, Ji Hyun Lee, and Duhee Bang. Single-cell rna sequencing technologies and bioinformatics pipelines. *Experimental & molecular medicine*, 50(8):1–14, 2018. ISSN 2092-6413. doi: https://doi.org/10.1038/s12276-018-0071-8. URL `https://doi.org/10.1038/s12276-018-0071-8`.

Yuge Ji, Mohammad Lotfollahi, F. Alexander Wolf, and Fabian J. Theis. Machine learning for perturbational single-cell omics. *Cell Systems*, 12(6):522–537, 2021. ISSN 2405-4712. doi: https://doi.org/10.1016/j.cels.2021.05.016. URL `https://www.sciencedirect.com/science/article/pii/S2405471221002027`.

Bowen Jing, Bonnie Berger, and Tommi Jaakkola. Alphafold meets flow matching for generating protein ensembles. *arXiv preprint arXiv:2402.04845*, 2024.

Kacper Kapusniak, Peter Potaptchik, Teodora Reu, Leo Zhang, Alexander Tong, Michael Bronstein, Avishek Joey Bose, and Francesco Di Giovanni. Metric flow matching for smooth interpolations on the data manifold, 2024.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 21696–21707. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/b578f2a52a0229873fefc2a4b06377fa-Paper.pdf`.

Dominik Klein, Théo Uscidda, Fabian Theis, and Marco Cuturi. Entropic (gromov) wasserstein flow matching with genot, 2023. URL `https://arxiv.org/abs/2310.09254`.

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. ISSN 1476-4687. doi: https://doi.org/10.1038/s41586-018-0414-6. URL `https://doi.org/10.1038/s41586-018-0414-6`.

Daniel Levine, Sacha Lévy, Syed Asad Rizvi, Nazreen Pallikkavaliyaveetil, Xingyu Chen, David Zhang, Sina Ghadermarzi, Ruiming Wu, Zihe Zheng, Ivan Vrkic, Anna Zhong, Daphne Raskin, Insu Han, Antonio Henrique de Oliveira Fonseca, Josue Ortega Caro, Amin Karbasi, Rahul M. Dhodapkar, and David van Dijk. Cell2sentence: Teaching large language models the language of biology. September 2023. doi: 10.1101/2023.09.11.557287. URL `http://dx.doi.org/10.1101/2023.09.11.557287`.

Wei Vivian Li and Jingyi Jessica Li. A statistical simulator scDesign for rational scRNA-seq experimental design. *Bioinformatics*, 35(14):i41–i50, 07 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz321. URL `https://doi.org/10.1093/bioinformatics/btz321`.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=PqvMRDCJT9t`.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pp. 423–439. Springer, 2022a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022b.

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, November 2018. ISSN 1548-7105. doi: 10.1038/s41592-018-0229-2. URL `http://dx.doi.org/10.1038/s41592-018-0229-2`.

Mohammad Lotfollahi, F Alexander Wolf, and Fabian J Theis. scgen predicts single-cell perturbation responses. *Nature methods*, 16(8):715–721, 2019. doi: 10.1038/s41592-019-0494-8. URL `https://doi.org/10.1038/s41592-019-0494-8`.

Mohammad Lotfollahi, Anna Klimovskaia Susmelj, Carlo De Donno, Leon Hetzel, Yuge Ji, Ignacio L Ibarra, Sanjay R Srivatsan, Mohsen Naghipourfar, Riza M Daza, Beth Martin, Jay Shendure, Jose L McFaline-Figueroa, Pierre Boyeau, F Alexander Wolf, Nafissa Yakubova, Stephan Günnemann, Cole Trapnell, David Lopez-Paz, and Fabian J Theis. Predicting cellular responses to complex perturbations in high-throughput screens. *Molecular Systems Biology*, 19(6), May 2023. ISSN 1744-4292. doi: 10.15252/msb.202211517. URL `http://dx.doi.org/10.15252/msb.202211517`.

Elijah K. Lowe, Claudia Cuomo, Danila Voronov, and Maria I. Arnone. *Using ATAC-seq and RNA-seq to increase resolution in GRN connectivity*, pp. 115–126. Elsevier, 2019. doi: 10.1016/bs.mcb.2018.11.001. URL `http://dx.doi.org/10.1016/bs.mcb.2018.11.001`.

Malte Luecken, Daniel Burkhardt, Robrecht Cannoodt, Christopher Lance, Aditi Agrawal, Hananeh Aliee, Ann Chen, Louise Deconinck, Angela Detweiler, Alejandro Granados, Shelly Huynh, Laura Isacco, Yang Kim, Dominik Klein, BONY DE KUMAR, Sunil Kuppasani, Heiko Lickert, Aaron McGeever, Joaquin Melgarejo, Honey Mekonen, Maurizio Morri, Michaela Müller, Norma Neff, Sheryl Paul, Bastian Rieck, Kaylie Schneider, Scott Steelman, Michael Sterr, Daniel Treacy, Alexander Tong, Alexandra-Chloe Villani, Guilin Wang, Jia Yan, Ce Zhang, Angela Pisco, Smita Krishnaswamy, Fabian Theis, and Jonathan M Bloom. A sandbox for prediction and integration of dna, rna, and proteins in single cells. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021a. URL `https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/158f3069a435b314a80bdcb024f8e422-Paper-round2.pdf`.

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. *Nature Methods*, 19(1):41–50, December 2021b. ISSN 1548-7105. doi: 10.1038/s41592-021-01336-8. URL `http://dx.doi.org/10.1038/s41592-021-01336-8`.

Malte D Luecken, Maren Büttner, Kridsadakorn Chaichoompu, Anna Danese, Marta Interlandi, Michaela F Müller, Daniel C Strobl, Luke Zappia, Martin Dugas, Maria Colomé-Tatché, et al. Benchmarking atlas-level data integration in single-cell genomics. *Nature methods*, 19(1):41–50, 2022.

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. ISSN 2041-1723. doi: 10.1038/s41467-019-14018-z. URL `https://doi.org/10.1038/s41467-019-14018-z`.

Jonathan S. Packer, Qin Zhu, Chau Huynh, Priya Sivaramakrishnan, Elicia Preston, Hannah Dueck, Derek Stefanik, Kai Tan, Cole Trapnell, Junhyong Kim, Robert H. Waterston, and John I. Murray. A lineage-resolved molecular atlas of c. elegans embryogenesis at single-cell resolution. *Science*, 365(6459), September 2019. ISSN 1095-9203. doi: 10.1126/science.aax1971. URL `http://dx.doi.org/10.1126/science.aax1971`.

Alessandro Palma, Sergei Rybakov, Leon Hetzel, and Fabian J Theis. Modelling single-cell rna-seq trajectories on a flat statistical manifold. In *NeurIPS 2023 AI for Science Workshop*, 2023.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Michael Poli, Stefano Massaroli, Atsushi Yamashita, Hajime Asama, Jinkyoo Park, and Stefano Ermon. Torchdyn: Implicit models and neural numerical methods in pytorch.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky TQ Chen. Multisample flow matching: Straightening flows with minibatch couplings. *arXiv preprint arXiv:2304.14772*, 2023.

Till Richter, Mojtaba Bahrami, Yufan Xia, David S Fischer, and Fabian J Theis. Delineating the effective use of self-supervised learning in single-cell genomics. *bioRxiv*, 2024.

Robin Rombach, Patrick Esser, and Bjorn Ommer. Network-to-network translation with conditional invertible neural networks. *Advances in Neural Information Processing Systems*, 33:2784–2797, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Orit Rozenblatt-Rosen, Michael J. T. Stubbington, Aviv Regev, and Sarah A. Teichmann. The human cell atlas: from vision to reality. *Nature*, 550(7677):451–453, October 2017. ISSN 1476-4687. doi: 10.1038/550451a. URL `http://dx.doi.org/10.1038/550451a`.

Lisa Sikkema, Ciro Ramírez-Suástegui, Daniel C Strobl, Tessa E Gillett, Luke Zappia, Elo Madissoon, Nikolay S Markov, Laure-Emmanuelle Zaragosi, Yuge Ji, Meshal Ansari, et al. An integrated cell atlas of the lung in health and disease. *Nature Medicine*, 29(6):1563–1577, 2023. ISSN 1546-170X. doi: https://doi.org/10.1038/s41591-023-02327-2. URL `https://doi.org/10.1038/s41591-023-02327-2`.

Dongyuan Song, Qingyang Wang, Guanao Yan, Tianyang Liu, Tianyi Sun, and Jingyi Jessica Li. scdesign3 generates realistic in silico data for multimodal single-cell and spatial omics. *Nature Biotechnology*, 42(2):247–252, 2024. ISSN 1546-1696. doi: https://doi.org/10.1038/s41587-023-01772-1. URL `https://doi.org/10.1038/s41587-023-01772-1`.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. URL `https://arxiv.org/abs/2011.13456`.

Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=CD9Snc73AW`. Expert Certification.

Hoa Thi Nhu Tran, Kok Siong Ang, Marion Chevrier, Xiaomeng Zhang, Nicole Yee Shin Lee, Michelle Goh, and Jinmiao Chen. A benchmark of batch-effect correction methods for single-cell rna sequencing data. *Genome Biology*, 21(1), January 2020. ISSN 1474-760X. doi: 10.1186/s13059-019-1850-9. URL `http://dx.doi.org/10.1186/s13059-019-1850-9`.

Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

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. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Zhizhong Wang, Lei Zhao, and Wei Xing. Stylediffusion: Controllable disentangled style transfer via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7677–7689, 2023.

F Alexander Wolf, Philipp Angerer, and Fabian J Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19:1–5, 2018. ISSN 1474-760X. doi: https://doi.org/10.1186/s13059-017-1382-0. URL `https://doi.org/10.1186/s13059-017-1382-0`.

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. *Molecular Systems Biology*, 17(1), January 2021. ISSN 1744-4292. doi: 10.15252/msb.20209620. URL `http://dx.doi.org/10.15252/msb.20209620`.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Comput. Surv.*, 56(4), nov 2023a. ISSN 0360-0300. doi: 10.1145/3626235. URL `https://doi.org/10.1145/3626235`.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023b.

Jason Yim, Andrew Campbell, Andrew YK Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S Veeling, Regina Barzilay, Tommi Jaakkola, et al. Fast protein backbone generation with se (3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023.

Masahiro Yoshida, Kaylee B Worlock, Ni Huang, Rik GH Lindeboom, Colin R Butler, Natsuhiko Kumasaka, Cecilia Dominguez Conde, Lira Mamanova, Liam Bolt, Laura Richardson, et al. Local and systemic responses to sars-cov-2 infection in children and adults. *Nature*, 602(7896): 321–327, 2022. ISSN 1476-4687. doi: https://doi.org/10.1038/s41586-021-04345-x. URL `https://doi.org/10.1038/s41586-021-04345-x`.

Luke Zappia, Belinda Phipson, and Alicia Oshlack. Splatter: simulation of single-cell rna sequencing data. *Genome biology*, 18(1):174, 2017. ISSN https://doi.org/10.1186/s13059-017-1305-0. doi: https://doi.org/10.1186/s13059-017-1305-0. URL `https://doi.org/10.1186/s13059-017-1305-0`.

Tao Zeng and Hao Dai. Single-cell rna sequencing-based computational analysis to describe disease heterogeneity. *Frontiers in Genetics*, 10, July 2019. ISSN 1664-8021. doi: 10.3389/fgene.2019.00629. URL `http://dx.doi.org/10.3389/fgene.2019.00629`.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023.

A THEORETICAL SUPPLEMENT

A.1 POISSON-GAMMA AND NEGATIVE BINOMIAL DISTRIBUTION

A possible parameterization of the negative binomial distribution is via a mean μ and an inverse dispersion parameter θ following the Probability Mass Function (PMF):

$$p_{\text{NB}}(x \mid \mu, \theta) = \frac{\gamma(\theta + x)}{x!\gamma(\theta)}\Big(\frac{\theta}{\theta+\mu}\Big)^{\theta}\Big(\frac{\mu}{\theta+\mu}\Big)^{x} . \tag{9}$$

One can show that the negative binomial distribution is obtained by a *continuous mixture* of Poisson distributions with gamma-distributed rate. More formally, define a Poisson model $x \sim \text{Poisson}(\lambda)$ with $\lambda \geq 0$. The parameter λ represents both the mean and the variance of the distribution. Since the mean and the variance of the distribution are equal, a Poisson model is not suited for modeling over-dispersed counts (i.e., where the variance exceeds the mean). A way around such a shortcoming is to model the rate of a Poisson distribution as a random variable following a gamma distribution.

$$x \sim \text{Poisson}(\lambda) \ , \tag{10}$$

$$\lambda \sim \text{gamma}\left(\theta, \frac{\mu}{\theta}\right) \ , \tag{11}$$

where $\theta \geq 0$ is the shape parameter and $\mu \geq 0$ the scale parameter. Marginalizing out λ in the PMF of the Poisson distribution, one retrieves the PMF of a negative binomial with mean μ and inverse dispersion θ. Notably, the variance of such a negative binomial parameterization is $\mu + \frac{\mu^2}{\theta}$. As one can see, the variance always exceeds the mean as long as θ is finite, making the negative binomial distribution a suitable tool to model over-dispersed counts.

A.2 FLOW MATCHING WITH GAUSSIAN PATHS

Flow Matching (Lipman et al., 2023) learns a time-dependent vector field $u_t(\mathbf{z})$, with $t \in [0, 1]$, generating the probability path $p_t(\mathbf{z})$, such that $p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a standard Gaussian prior and p_1 is a complex distribution. A common way to formulate the marginal p_t is via the mixture

$$p_t(\mathbf{z}) = \int p_t(\mathbf{z}|\mathbf{z}_1)q(\mathbf{z}_1)\mathrm{d}\mathbf{z}_1,$$

where q is the target data distribution. The following marginal vector field generates such a mixture of paths (Lipman et al., 2023):

$$u_t(\mathbf{z}) = \int u_t(\mathbf{z}|\mathbf{z}_1)\frac{p(\mathbf{z}|\mathbf{z}_1)q(\mathbf{z}_1)}{p_t(\mathbf{z})}\mathrm{d}\mathbf{z}_1 \ .$$

While u_t is intractable, the conditional field $u_t(\mathbf{z}|\mathbf{z}_1)$ has a closed-form expression given an observed data point $\mathbf{z}_1$ and a pre-defined choice of the probability path $p_t(\mathbf{z}|\mathbf{z}_1)$ satisfying the boundary conditions $p_0(\cdot|\mathbf{z}_1) = p_0$ and $p_1(\cdot|\mathbf{z}_1) = \delta(\mathbf{z} - \mathbf{z}_1)$. Notably, $u_t(\mathbf{z}|\mathbf{z}_1)$ admits the same minimizer as $u_t(\mathbf{z})$ and can be used as a regression target during training. Following Lipman et al. (2023), one can assume Gaussian probability paths $p_t(\mathbf{z}|\mathbf{z}_1) = \mathcal{N}(\mathbf{z}|\alpha_t\mathbf{z}_1, \sigma_t^2\mathbf{I})$, where the tuple (α_t, σ_t) is called scheduler and satisfies $\alpha_0 = 0 = \sigma_1$ and $\alpha_1 = 1 = \sigma_0$. In this work, we use standard linear scheduling, where $\alpha = t$ and $\sigma = 1 - t$.

A.3 THE RELATIONSHIP BETWEEN FLOW MATCHING AND CLASSIFIER-FREE GUIDANCE

Zheng et al. (2023) draw a relationship between classifier-free guidance in score-based models (Ho & Salimans, 2022) and the Flow Matching vector field u_t. Specifically, the authors show that the following relationship between the score $\nabla_{\mathbf{z}} \log p_t(\mathbf{z}|y)$ and the marginal vector field $u_t(\mathbf{z}|y)$ holds:

$$u_t(\mathbf{z}|y) = a_t\mathbf{z} + b_t\nabla_{\mathbf{z}} \log p_t(\mathbf{z}|y) \ , \tag{12}$$

both in the conditional case and when $y = \varnothing$, with $a_t = \frac{\dot{\alpha}_t}{\alpha_t}$ and $b_t = (\dot{\alpha}_t\sigma_t - \alpha_t\dot{\sigma}_t)\frac{\sigma_t}{\alpha_t}$.

Let the equation

$$\nabla_{\mathbf{z}} \log \tilde{p}_t(\mathbf{z}|y) = (1-\omega)\nabla_{\mathbf{z}} \log p_t(\mathbf{z}) + \omega\nabla_{\mathbf{z}} \log p_t(\mathbf{z}|y)$$

be the classifier-free guidance score as formulated by Ho & Salimans (2022) with guidance strength ω. Zheng et al. (2023) define the vector field of classifier-free Flow Matching as

$$\tilde{u}_t(\mathbf{z}|y) = (1-\omega)u_t(\mathbf{z}) + \omega\, u_t(\mathbf{z}|y) \tag{13}$$

which is related to the classifier-free guidance score by

$$\tilde{u}_t(\mathbf{z}|y) = a_t\mathbf{z} + b_t\nabla_{\mathbf{z}}\log\tilde{p}_t(\mathbf{z}|y)\ ,$$

derived by substituting Eq. (12) into Eq. (13).

A.4 PROOF OF PROPOSITION 1

Proposition 1 *If the attributes $y_1, ..., y_K$ are conditionally independent given $\mathbf{z}$, the vector field*

$$\tilde{u}_t(\mathbf{z}|\mathbf{y}) = u_t(\mathbf{z}) + \sum_{i=1}^{K}\omega_i[u_t(\mathbf{z}|y_i) - u_t(\mathbf{z})]$$

coincides with the velocity of the probability-flow ODE associated with the generative SDE of a diffusion model with the compositional score as in Eq. (7)*.*

Proof. (Proposition 1) We first justify the conditional independence assumption and successfully show the described equality.

Conditional independence assumption. Given a variable $\mathbf{z}$ and a set of attributes $\mathbf{y} = y_1, ..., y_K$, our aim is to sample conditionally from the marginal distribution $p_t(\mathbf{z}|y_1, ..., y_K)$. To obtain the compositional score formulation in Eq. (7), one must first assume that the attributes are conditionally independent given $\mathbf{z}$. Then:

$$p_t(\mathbf{z}|y_1, ..., y_K) \propto p_t(\mathbf{z}, y_1, ..., y_K) = p_t(\mathbf{z})\prod_{i=1}^{K}p_t(y_i|\mathbf{z}) \propto p_t(\mathbf{z})\prod_{i=1}^{K}\frac{p_t(\mathbf{z}|y_i)}{p_t(\mathbf{z})}\ . \tag{14}$$

Taking the logarithm and then the gradient with respect to $\mathbf{z}$ on both sides in Eq. (14), we obtain:

$$\nabla_{\mathbf{z}}\log p_t(\mathbf{z}|y_1, ..., y_K) = \nabla_{\mathbf{z}}\log p_t(\mathbf{z}) + \sum_{i=1}^{K}\left[\nabla_{\mathbf{z}}\log p_t(\mathbf{z}|y_i) - \nabla_{\mathbf{z}}\log p_t(\mathbf{z})\right]\ . \tag{15}$$

Additionally, if the goal is to sample with attribute-specific guidance strengths $\{\omega_i\}_{i=1}^{K}$ according to a modified conditional distribution

$$\tilde{p}(\mathbf{z}|y_1, ..., y_K) \propto p(\mathbf{z})\prod_{i=1}^{K}\left[\frac{p(\mathbf{z}|y_i)}{p(\mathbf{z})}\right]^{\omega_i}\ , \tag{16}$$

the score in Eq. (15) becomes:

$$\nabla_{\mathbf{z}}\log\tilde{p}_t(\mathbf{z}|y_1, ..., y_K) = \nabla_{\mathbf{z}}\log p_t(\mathbf{z}) + \sum_{i=1}^{K}\omega_i\left[\nabla_{\mathbf{z}}\log p_t(\mathbf{z}|y_i) - \nabla_{\mathbf{z}}\log p_t(\mathbf{z})\right]\ . \tag{17}$$

The formulation in Eq. (17) is used to parameterize the drift of the reverse-time SDE that generates data points conditionally on the attributes $y_1, ..., y_K$ with guidance strengths $\{\omega_i\}_{i=1}^{K}$.

Proof of equality. Following the standard theory of score-based models (Yang et al., 2023b) and their compositional version (Liu et al., 2022a), we first note that one can use the compositional classifier-free guidance score

$$\nabla_{\mathbf{z}}\log\tilde{p}_t(\mathbf{z}|\mathbf{y}) = \nabla_{\mathbf{z}}\log p_t(\mathbf{z}) + \sum_{i=1}^{K}\omega_i\left[\nabla_{\mathbf{z}}\log p_t(\mathbf{z}|y_i) - \nabla_{\mathbf{z}}\log p_t(\mathbf{z})\right]$$

to simulate the probability-flow ODE

$$\dot{\mathbf{z}} = f_t\mathbf{z} - \frac{1}{2}g_t^2\nabla_{\mathbf{z}}\log\tilde{p}_t(\mathbf{z}|\mathbf{y})\ . \tag{18}$$

Given a scheduling pair (α_t, σ_t), Kingma et al. (2021) show that

$$f_t = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}, \quad g_t^2 = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t \;, \tag{19}$$

which yield

$$\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t} = \frac{\dot{\alpha}}{\alpha_t} = a_t, \quad -\frac{1}{2}\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} + \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t = (\dot{\alpha}\sigma_t - \alpha_t\dot{\sigma}_t)\frac{\sigma_t}{\alpha_t} = b_t \;. \tag{20}$$

Hence, the probability-flow ODE is written as

$$\dot{\mathbf{z}} = a_t\mathbf{z} + b_t\nabla_{\mathbf{z}}\log\tilde{p}_t(\mathbf{z}|\mathbf{y}) \;. \tag{21}$$

Given the score in Eq. (8), we first use results from (Zheng et al., 2023) explained in Appendix A.3 to justify the following equalities:

$$u_t(\mathbf{z}) = a_t\mathbf{z} + b_t\nabla_{\mathbf{z}}\log p_t(\mathbf{z}) \tag{22}$$

$$u_t(\mathbf{z}|y_i) = a_t\mathbf{z} + b_t\nabla_{\mathbf{z}}\log p_t(\mathbf{z}|y_i) \tag{23}$$

Plugging Eq. (23) and Eq. (22) into Eq. (8) yields:

$$\begin{aligned}\tilde{u}_t(\mathbf{z}|\mathbf{y}) &= a_t\mathbf{z} + b_t\left[\nabla_{\mathbf{z}}\log p_t(\mathbf{z}) + \sum_{i=1}^{K}\omega_i(\nabla_{\mathbf{z}}\log p_t(\mathbf{z}|y_i) - \nabla_{\mathbf{z}}\log p_t(\mathbf{z}))\right] && (24)\\ &= a_t\mathbf{z} + b_t\nabla_{\mathbf{z}}\log\tilde{p}_t(\mathbf{z}|\mathbf{y}) \;. && (25)\end{aligned}$$

completing the proof.

A.5 RELATIONSHIP WITH EXISTING SINGLE-CELL GENERATIVE MODELS

Although likelihood models are standard in the single-cell literature, CFGen leverages a novel factorization scheme as depicted in Eq. (5). We delineate the difference between our approach and standard single-cell VAEs:

- In scVI (Lopez et al., 2018) the conditioning does not happen at the level of the prior on the latent variable $p(\mathbf{z})$ but in the decoding phase. Conversely, CFGen performs conditioning ahead of drawing the latent variable $\mathbf{z}$, so it can perform sampling from multiple modes. Generating from noise leads to a more informative cell state than scVI. Thus, given a conditioner y, CFGen models a conditional prior $p(\mathbf{z}|y)$.
- As a result, the likelihood component $p(\mathbf{x}|\mathbf{z}, l)$ is also not the same as in most single-cell VAEs, since in our case it is represented by an unconditional decoder, as the conditioning of y is already incorporated in the flow-based generation of $\mathbf{z}$, while conditional VAE models need to feed the label both to the encoder and the decoder.
- VAEs considering a conditional prior exist (Xu et al., 2021). However, they are usually employed as representation learning rather than generation frameworks. As a result, they tend to under-regularize the latent space to the prior distribution to favor structure and reconstruction. Conversely, setting a *strong* conditional flow-based prior onto a latent cell representation, CFGen is not compromised by the KL-likelihood trade-off, and can thus generate arbitrarily well on unregularized latent representations.
- While other work may have explored defining a distribution on the library size, our way of using it to distinctively condition the generative process is unique, and so is our factorization in Eq. (5). The library size is normally used just for the likelihood optimization process multiplied by the post-softmax output of the decoder. We, instead, employ it to inform the conditional generation from the flow-based conditional prior. We do this in a formally sound manner by first defining it as a conditioning attribute and then factorizing our latent variable model to generate $\mathbf{z}$ from $p(\mathbf{z}|y, l)$. To the best of our knowledge, this formulation was previously unexplored but it significantly boosts the generation of single-cell counts. Generating from noise in single-cell VAEs has no notion of cell size and may be incompatible with settings where such information is biologically relevant.

A.6 GUIDANCE ALGORITHM

Algorithm 1 and Algorithm 2 depict our training strategies. In what follows, for notational simplicity, we indicate $\phi_t(\mathbf{z})$ with $\mathbf{z}_t$, with $t \in [0, 1]$.

Algorithm 1 Train CFGen with multiple attributes on scRNA-seq

```
Require: Probability of unconditional generation $p_{\text{uncond}}$, trained encoder $f_\eta$, scheduling $(\alpha_t, \sigma_t)$.
Initialize $v_{t,\xi}$
while not converged do
    Sample $(\mathbf{x}_1, y_1, ..., y_K)$ from the data
    $\mathbf{z}_1 \leftarrow f_\eta(\mathbf{x}_1)$
    Sample $t$ from $\mathcal{U}[0, 1]$
    $l \leftarrow$ Sum of entries of $\mathbf{x}_1$
    Sample $b$ from Bernoulli($p_{\text{uncond}}$)
    if $b = 1$ then
        $y \leftarrow \emptyset$
    else
        $y \leftarrow$ random sample among $y_1, ..., y_K$
    end if
    $\mathbf{z}_0 \sim p(\mathbf{z}_0)$ {sample noise}
    $\mathbf{z}_t \leftarrow \alpha_t \mathbf{z}_1 + \sigma_t \mathbf{z}_0$ {noisy data point}
    $\dot{\mathbf{z}}_t \leftarrow \dot{\alpha}_t \mathbf{z}_1 + \dot{\sigma}_t \mathbf{z}_0$
    Take gradient step on $\nabla_\xi ||v_{t,\xi}(\mathbf{z}_t|y, l) - \dot{\mathbf{z}}_t||^2$
end while
Output: $v_{t,\xi}$
```

Algorithm 2 Sampling from multi-attribute guided CFGen for scRNA-seq

```
Require: Trained velocity field $v_{t,\xi}$, conditions $y_1, ..., y_K$, guidance parameters $\omega_1, ..., \omega_K$,
    size factor distribution parameters $(\mu_l, \sigma_l)$, number of ODE steps $n_{\text{ode}}$,
    trained decoder $h_\psi$, trained inverse dispersion parameter $\boldsymbol{\theta}$.
Sample size factor $l$ from $\text{LogNormal}(\mu_l, \sigma_l)$
$\mathbf{z}_0 \sim p(\mathbf{z}_0)$ {sample noise}
$n \leftarrow 1/n_{\text{ode}}$ {step size}
$\tilde{u}_t(\cdot) \leftarrow v_{t,\xi}(\cdot|\emptyset, l) + \sum_{i=1}^{K} \omega_i \left[v_{t,\xi}(\cdot|y_i, l) - v_{t,\xi}(\cdot|\emptyset, l)\right]$ {guided velocity}
for $t = 0, n, \ldots, 1 - n$ do
    $\mathbf{z}_{t+n} \leftarrow$ ODEStep($\tilde{u}_t, \mathbf{z}_t$) {ODE solver step}
end for
$\mathbf{x}_1 \leftarrow$ Sample from NB($l\,\text{softmax}(h_\psi(\mathbf{z}_1)), \boldsymbol{\theta}$)
Output: $\mathbf{x}_1$
```

B MODEL DETAILS

The CFGen model is implemented in `PyTorch` (Paszke et al., 2017), version `2.1.2`.

B.1 THE CFGEN AUTOENCODER

Before training the flow generating noise from data, we embed the data with an autoencoder model trained with maximum likelihood optimization.

Encoder. The encoder is a multi-layer perceptron (MLP) with two hidden layers of dimension `[512, 256]`. The last layer maps to a latent space of variable size. In our experiment, we use 50 latent dimensions for all datasets excluding the Human Lung Cell Atlas (HLCA) and the Tabula Muris datasets, where we set the latent dimension to 100 for higher expressivity. In the multi-modal setting, the different modalities are embedded into the same latent space. Both RNA and ATAC inputs initially pass through a modality-specific MLP encoder. For ATAC, we fix the dimension of the hidden layers to `[1024, 512]` due to its higher dimensionality. The outputs of the two modality-specific encoders are then concatenated and further encoded by a shared encoder layer, mapping to a 100-dimensional latent space.

Decoder. The decoder maps the latent space to the parameter space of a likelihood model. When dealing with multi-modal data, one decoder per modality is used to sample from the noise model corresponding to the specific modality.

- For scRNA-seq, we map the latent representation to the mean of the Negative Binomial likelihood, $\boldsymbol{\mu}$, with one dimension per gene. Following (Lopez et al., 2018), we compute the `softmax` transformation of the output across the gene dimension. The resulting probabilities are multiplied by the *size factor*, which is the total number of transcripts per cell. The inverse dispersion is a learned model parameter implemented via `torch.nn.Parameter`. Crucially, we offer the option to model the inverse dispersion per gene and attribute, in case this reflects the properties of the datasets.
- When the DNA accessibility modality is present, the decoder is used to map the latent space to continuous logit values. These are processed with a `sigmoid` function for each dimension, with no need for a size factor.

Additional training details. We train the encoder and decoder networks jointly via likelihood optimization. Therefore, the weights of the networks are tweaked (together with the inverse dispersion parameter) to produce the parameters that maximize the likelihood of the data under a pre-defined noise model. For scRNA-seq, we employ the negative binomial distribution. For ATAC, we use Bernoulli likelihood. The losses from different modalities are summed before applying backpropagation. Notice that scRNA-seq data are provided by the encoder in their $\log$-transformed version for training stability. Intuitively, the loss is evaluated on the original counts.

For all settings, we keep the learning rate to 0.001, while all couples of layers are interleaved with 1-dimensional batch normalization layers. We use a standard `AdamW` optimizer and the `ELU` activation function as a non-linearity.

B.2 THE FLOW MODEL

The flow architecture. The flow model inputs the *latent representation* computed by the encoder and produces a vector field used to simulate paths generating data from noise. The architecture is essentially a deep dimensionality-preserving ResNet (He et al., 2016). The flow architecture is made of the following modules:

- A linear projection layer from the input dimension to the hidden dimension of the flow model.
- Three middle ResNet blocks stacked on top of each other performing representation learning and conditioning.
- An output layer with one level of non-linearity, given by a `SiLU` activation function.
- A time embedder, inputting sinusoidal multi-dimensional encoding (Vaswani et al., 2017) of the time with `1e4` frequency value. It is represented by an MLP with two layers and `SiLU` non-linearity.
- A size factor embedder, inputting a sinusoidal embedding (Vaswani et al., 2017) of the size factor to approximately guide generation to a pre-defined number of transcripts. This is used only if the size factor is a conditioner for the model, that is, we do not assume $p(\mathbf{z}|y, l) = p(\mathbf{z}|y)$. Before being passed through sinusoidal embeddings, the $\log$ size factor is normalized to a value lying approximately between 0 and 1 using the maximum and minimum $\log$ size factors from the dataset.
- A covariate embeddings for all different conditioning attributes.

Additional technical details. During training the covariate embeddings are elementwise summed to the time embedding and the size factor embedding (if applicable). Hence, all embeddings are either designed to share the same size or are transformed to a common dimensionality. The sum of such covariate representations is passed to the ResNet middle blocks as a single vector.

The summed conditioning embedding and the down-projected input are provided to the ResNet blocks. The ResNet blocks consist of:

- A non-linear input transformation of the state embedding.
- A linear encoder for the covariate embedding.
- A non-linear output transformation.
- A skip connection.

The results of the non-linear input transformation and the covariate encoder are summed and passed to the output transformation. The result is summed to the input of the ResNet via the skip connection as in traditional residual blocks (He et al., 2016). All non-linear transformations are simple `[siLU, Linear]` stacks.

In the standard setting, we train the flow for $1,000$ epochs, using `AdamW` as an optimizer, a learning rate of 0.001 and batch size 256.

B.3 COVARIATE EMBEDDINGS

Covariate embeddings are trainable `torch.nn.Embdding` layers of pre-defined size. In our experiments, we use size 100 in most of the settings.

B.4 SAMPLING FROM NOISE

To generate discrete observations from noise, we first draw a covariate from the associated categorical distribution with proportions obtained from the observed data and a size factor from the $\mathrm{LogNormal}$ distribution (with mean and standard deviation as the Maximum Likelihood Estimates (MLE) from the whole dataset or conditional on a technical effect covariate). Then, we sample Gaussian noise. We simulate a latent observation from the real datasets conditionally by integrating the vector field computed by the neural network in Appendix B.2 starting from Gaussian noise and using the `dopri5` solver with `adjoint` sensitivity and `1e-5` tolerance from the `torchdyn` (Poli et al.) package in Python3 (Van Rossum & Drake, 2009). We integrate over the $(0, 1)$ time interval. The generated latent vector is decoded to the parameter space of the data likelihood (negative binomial for scRNA-seq or Bernoulli for ATAC-seq) and single cells are sampled from the parameterized noise model.

B.5 SEPARATE TRAINING

In CFGen we train the encoder f_η separately from the flow model. Initially, when modeling the AE and the flow jointly, we found that training the flow was unstable. Specifically, Flow Matching performs better on a fixed state space. Alternating AE and flow updates causes continuous changes in data representation since the AE evolves with the flow, hindering accurate velocity field estimation, especially during the VAE's early updates. One could initially train the AE with a higher learning rate than the flow, periodically decreasing the former and increasing the latter. However, this approach is similar to training them separately, which we ultimately adopted to avoid retraining the AE repeatedly.

C SCHEDULING

We use linear scheduling, following the original formulation advanced by Lipman et al. (2023). For more details, we refer to Appendix A.2.

D BASELINE DESCRIPTION

D.1 SCVI, MULTIVI, PEAKVI

scVI (Lopez et al., 2018), MultiVI (Ashuach et al., 2023) and PeakVI (Ashuach et al., 2022) are all VAE-based generative models for single-cell discrete data. Following the standard VAE setting, such models learn a Gaussian latent space which is decoded to the parameters of the discrete likelihood models describing different single-cell modalities. While PeakVI and scVI generate single modalities (respectively, ATAC and scRNA-seq data), MultiVI learns a common latent space between modalities, while sampling from different discrete decoders.

D.2 SCANVI AND SCPOLI

In the batch correction experiment described in Section 5.5 we compare CFGen with two additional VAE-based models: scanVI (Xu et al., 2021) and scPoli (De Donno et al., 2023). scanVI is similar to scVI, with the addition of a latent cell type classifier to enforce biological preservation in the representation space and a conditional prior on the latent space. scPoli utilizes continuous embeddings rather than the one-hot encodings as conditioners to the VAE. Moreover, scPoli differs from scanVI in that it enforces biological coherence between cell-type-specific representations using latent cell-type prototypes. In simple terms, the model pulls cellular representations close to the average embedding vector of their associated cell type.

D.3 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. Notably, scGAN has been explored for conditional generation as well. However, we found the results conditioning the model on cell type to be way worse than when not providing the label. In the latter case, the model is trained conditionally on data-dependent Leiden cluster labels. We name such version *unconditional* as it does not exploit real labels but data-driven attributes.

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. During inference, the diffusion backbone denoises embeddings to produce new ones for gene expression data generation. The autoencoder addresses high-dimensional data and a 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.

D.5 DISCUSSION: WHAT SEPARATES SCDIFFUSION FROM CFGEN

Data Properties: scDiffusion does not account for key properties of single-cell data, such as sparsity, overdispersion, and discreteness. Although normalization can ensure continuity, most methods preserve zeros and introduce non-linear mean-variance trends. Continuous decoders, such as in scDiffusion, typically require centered and non-sparse input, making its design sub-optimal for scRNA-seq.

Conditional Sampling: scDiffusion relies on classifier-based guidance, making conditional sampling dependent on classifier performance for individual labels. This limits its application to rare cell type generation or attributes that are challenging to classify.

Training Stability: Training SDE-based diffusion models like scDiffusion is empirically complex and unstable for small datasets, such as PBMC3k.

In contrast, our framework overcomes these limitations by training a latent Flow Matching model with a discrete likelihood scheme and classifier-free guidance.

Table 4: List of datasets considered in this work with the associated number of genes, cells and cell types.

Dataset name	Number of cells	Number of genes	Number of cell types
PMBC3K	2,638	8,573	8
Dentate gyrus	18,213	17,002	14
Tabula Muris	245,389	19,734	123
HLCA	584,944	27,997	50
PBMC10k	10,025	25,604	14
NeurIPS	90,261	14,087	45
PBMC COVID	344,820	2,000	29
C.Elegans	89,701	17,747	35 (plus unknown)

CFGen also provides significantly faster sampling—two to three orders of magnitude faster than scDiffusion—due to:

Efficient Sampling: Flow Matching deterministically maps noise to data via approximately straight trajectories, requiring far fewer simulation steps (5–10 for CFGen vs. >1000 for scDiffusion) while achieving superior results.

Lower Dimensionality: CFGen operates in a reduced latent space (50–100 dimensions) compared to scDiffusion (1000 dimensions).

Guidance Independence: Unlike scDiffusion's reliance on classifier-based guidance, CFGen uses classifier-free guidance, avoiding performance dependence on a classifier's gradient.

These features enable CFGen to address key scRNA-seq challenges while being both computationally efficient and robust across datasets.

E COMPUTATIONAL RESOURCES

For the implementation of our model, we utilized Python 3.10 (Van Rossum & Drake, 2009) for the deep learning components. The experiments were executed on a variety of GPU servers, each possessing unique specifications:

- GPU: 16 Tesla V100 GPUs, each with 32GB of RAM
- GPU: 2 Tesla V100 GPUs, each with 16GB of RAM
- GPU: 8 A100-SXM4 GPUs, each with 40GB of RAM

F 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` and log-transformed. Since CFGen, MultiVI, PeakVI and scVI work in discrete spaces, we did not normalize the data to train them. Additionally, we filter out genes that are expressed in less than 20 cells in all the datasets. In each dataset, we allocate 80% of the observations to training and 20% to testing.

G EXPERIMENT DESCRIPTION AND EVALUATION METRICS

G.1 WASSERSTEIN-2 DISTANCE AND MMD

We use the Wasserstein-2 distance and the RBF-kernel Mean Maximum Discrepancy (MMD) with scales $\{0.01, 0.1, 1, 10, 100\}$ (Gretton et al., 2012) to measure the overlap between real and generated data. To implement the former we use the Python Optimal Transport (POT) (Flamary et al., 2021) package. For the linear MMD metric, we resort to the implementation proposed in [3].

---

[3] `https://github.com/atong01/conditional-flow-matching`

G.2 KNNC

Similarly to (Levine et al., 2023), we train a 10-NN classifier for cell type on the real test data and evaluate its F1 performance on the generated set. This indicates whether generated cell types are proximal to the real ones in the real dataset. Notice that a low score is not necessarily a symbol of poor generation performance, as it may be caused by the F1 scores being low on the real data. Therefore, such a metric should be considered comparatively.

G.3 DISTRIBUTION METRICS COMPARISONS

To compute the metrics in Table 1, we use all competing models to generate three datasets of the same size as the original. In the conditional case, distribution metrics are computed per cell type, namely, each cell type in the real and generated datasets is a subset and used for comparison. In the unconditional case, we compare batches of $5,000$ cells sampled from the whole data distribution. The MMD and Wasserstein-2 distances are computed in the Principal Component (PC) projection of the data onto 30 dimensions. This is done to face the curse of dimensionality, which hinders the reliability of distances in high dimensions. To make PC embeddings comparable between real and generated cells, we project generated cells using the PC loadings of the real cells. Crucially, scDiffusion and scGAN generate normalized data, while CFGen and scVI produce discrete count data. To make the obtained numbers comparable, we first normalize the output of CFGen and scVI to ensure that gene counts sum to `1e4`, then we $\log$-transform the results of all generative models to obtain a better range for the distances. The same is performed on the real data. After pre-processing, all generative models and the real data represent the same quantity. All metric measures are reported on the test set.

G.4 VARIANCE-MEAN TREND PLOT AND SPARSITY HISTOGRAMS

We select scDiffusion as a continuous baseline for demonstrative purposes. Given generated cells across datasets (after pre-processing as explained in Appendix G.3), we compute the mean and variance expression across cells per gene and the frequency of unexpressed genes per cell. These values are compared to each other. Note that the mean-variance relationship is expected to be quadratic when considering raw counts (see Appendix A.1). However, after the normalization and $\log$-transformation needed to compare with scDiffusion, the trend is no longer quadratic. Nevertheless, the relationship between mean and variance gene expression is still worth investigating, as it should follow the empirical behavior of real data.

G.5 MULTI-MODAL EVALUATION

We generate multi-modal data and perform an unconditional comparison with the ground truth as described in Appendix G.3. When dealing with ATAC data, we normalize both real and generated cells using the TF-IDF algorithm implemented in the MUON package (Bredikhin et al., 2022).

To generate Fig. A8, we perform the following steps:

1. Aggregate average gene expression and peak accessibility (i.e, the fraction of accessible regions) per cell type per marker gene/peak as described in [4]. The result is, for both real and all generated datasets, a matrix `cell_type x marker` with, as value, the average expression or accessibility of such marker in such cell type.
2. We correlate the rows of such matrix between real and generated datasets. A high correlation signifies that the generative model correctly captures the mean marker expression and accessibility across markers per cell type.

G.6 GUIDANCE STRENGTH EXPERIMENTS

In Fig. 3, first train CFGen on each dataset following Algorithm 1. Upon successful training, we show the guidance performance qualitatively by sampling 500 cells for an array of guidance strength

---

[4] `https://muon-tutorials.readthedocs.io/en/latest/single-cell-rna-atac/pbmc10k/3-Multimodal-Omics-Data-Integration.html`

combinations between attributes, keeping the guidance weight of an attribute fixed while varying the other as shown in the figure. For the unconditional generation (hence, with guidance strength equal to 0 for both attributes) we generated as many cells as there are in the dataset to better show the overlap between real and synthetic cells. When using guidance we train the guided CFGen model using a probability $p_{\text{uncond}} = 0.2$ (see Algorithm 1).

G.7 SCGPT (CUI ET AL., 2024) GENERALIZATION PERFORMANCE ENHANCEMENT

We split the PBMC COVID and HLCA datasets into a training set and a held-out set. To make the generalization task more challenging, we leave out all cells from 20% of the donors in both datasets. This makes 80 training and 27 test donors for HLCA and 60 training and 15 test donors for PBMC COVID. After augmenting the training set (see Section 5.4) we use a pre-trained scGPT model to embed training and validation sets, for both original and augmented data. Then we fit a cell-type kNN classifier on the training embeddings and evaluate it on the held-out set. The results displayed in Fig. 4 show to what extent the held-out classification performance on a cell type varies as a function of its frequency in the dataset. A performance improvement indicates that a cell type better separates from the rest in the LLM representation space after the model sees additional synthetic examples.

G.8 BATCH CORRECTION EVALUATION

Correction with CFGen. Batch correction consists of finding a representation of the data where the technical effect has been removed while preserving the biological variation. In the standard setting, an observation $\mathbf{x}$ is usually associated with a batch label y_{batch} and a biological annotation $y_{\text{cell type}}$. We first encode $\mathbf{x}$ into a latent variable $\mathbf{z}$. Flow Matching is a generative model mapping observations from a prior distribution to the data distribution using an invertible flow. The invertibility property also allows the data distribution to be transported to noise. Note that it is proven that inverting the flow back to the prior leads to removing the batch information as well as cell type variability from $\mathbf{z}$ (Rombach et al., 2020). Here, we perform flow inversion to strip $\mathbf{z}$ from biological and technical variation. Then we simulate the flow forward again starting from the noisy observation. Given a batch $y^{\text{ref}}_{\text{batch}}$ to equalize all cells to, we simulate the flow forward again conditioned on batch $y^{\text{ref}}_{\text{batch}}$ and the biological label of origin $y_{\text{cell type}}$ to be preserved. When applied to the whole dataset, technical differences are removed by transporting all observations to the same batch.

Remarks. In the context of classifier-free guidance, the weights ω_{batch} and $\omega_{\text{cell type}}$ can be adjusted to tune the level of biological preservation. Note that our correction approach resembles methods used for style transfer in diffusion models Wang et al. (2023).

Evaluation setup. We train CFGen and the competing models using the same cell type and batch covariates. For all comparisons, we use a representation space of dimension 50. For uncorrected data, we use the PC projection of the data as a representation for batch mixing evaluation. All VAE-based models were trained across 100 epochs with default options. Furthermore, scPoli was pre-trained for 40 steps (see (De Donno et al., 2023) for more details).

Metrics. We evaluate the quality of batch correction and biological conservation using the scIB package (Luecken et al., 2022). From scIB, we use five distinct metrics for batch correction and as many metrics for biological conservation. The scores reported in Fig. 5 reflect the average of such metrics, which is computed by default by the scIB package. The metrics are normalized between 0 and 1, with 1 representing perfect correction/conservation. All metrics build k-nearest-neighborhood of cells and use batch and cell type labels to evaluate the technical and biological mixing in the data. Here we provide a brief description of the metrics, however, we refer to (Luecken et al., 2022) for more details about the scores.

Selection of the guidance weights for batch correction. Appendix H.8 provides an intuition for the selection process. In batch correction, cells are transported to noise and back to data guided by biological and batch covariates. The guidance strength parameters ω_{bio} and ω_{batch} determine the emphasis on biological conservation and batch correction. Based on the scIB metrics only, one might select the highest guidance strengths, as these maximize aggregation within cell types and batches. However, as shown in Fig. A16 and Fig. A17, scIB metrics alone can be misleading and should be paired with qualitative evaluation. Excessive guidance collapses variability beyond the batch and biological annotations, leading to artifacts. Parameters near $\omega_{\text{bio}}, \omega_{\text{batch}} \in \{1, 2\}$ generally balance signal preservation and correction effectively. For example, Fig. A16 demonstrates that excessive biological preservation causes unnatural clustering. The extent of batch effect in the data should also guide parameter selection. For C. Elegans, with mild batch effects, $\omega_{\text{bio}} = 2, \omega_{\text{batch}} = 1$ performs better than $\omega_{\text{bio}} = 1, \omega_{\text{batch}} = 2$. Conversely, for NeurIPS, $\omega_{\text{bio}} = 1, \omega_{\text{batch}} = 2$ avoids artifacts observed for $\omega_{\text{bio}} > 1$ (Fig. A17). In summary, we recommend assessing the batch effect severity, sweeping over guidance weights, and selecting parameters that optimize scIB metrics without compromising realistic single-cell representaxtions.

Batch correction.

- Silhouette batch - Represents the Average Silhouette Width (AWS) between batch clusters.
- iLISI - Derived from neighborhood lists for each node in kNN graphs, the Inverse Simpson's index is applied to assess how many cells can be sampled from a neighbor list before encountering the same batch twice.
- KBET - Determines if the label composition of a cell's k-nearest neighborhood matches the expected (global) label distribution.
- Graph Connectivity - Evaluates if the kNN graph representation of the (integrated) data directly links all cells sharing the same label.
- PCR (Principal Component Regression) - Quantifies the amount of variance explained by the batch label before and after correction.

Biological conservation.

- Isolated labels - Define isolated cell labels as those that appear in the fewest number of batches during the integration task. Assesses how effectively these isolated labels are separated from other cell identities. The final score for each metric is the average isolation score of all isolated labels.
- K-means NMI - Normalized Mutual Information between k-mean and batch clusters.
- K-means ARI - Adjusted Ranmd Index between K-means and batch clusters.
- Silhouette label - represents the Average Silhouette Width (AWS) between cell type clusters.
- cLISI - Computes the cell-type-based version of the iLISI score.

H ADDITIONAL RESULTS

H.1 ANALYSIS OF THE RUNTIME

In Fig. A1 we empirically evaluate the impact of different hyperparameters on the runtime of the model. We generate fake data from an untrained CFGen instance initialized with a specific configuration. Each hyperparameter is evaluated for different latent space sizes. Since CFGen is a latent Flow Matching model, the size of the latent space of the representation bottleneck is relevant and is expected to condition generation speed the most. We consider the following hyperparameters:

1. The number of generated genes (default: 20k).
2. The number of generated cells (default: 50k).
3. The dimensions of the denoising model's bottleneck (default: 128).
4. The number of neural network blocks in the denoising model (default: 3).
5. The size of the embedding for the conditions (default: 128).

When evaluating one hyperparameter, the others are set to their default values.

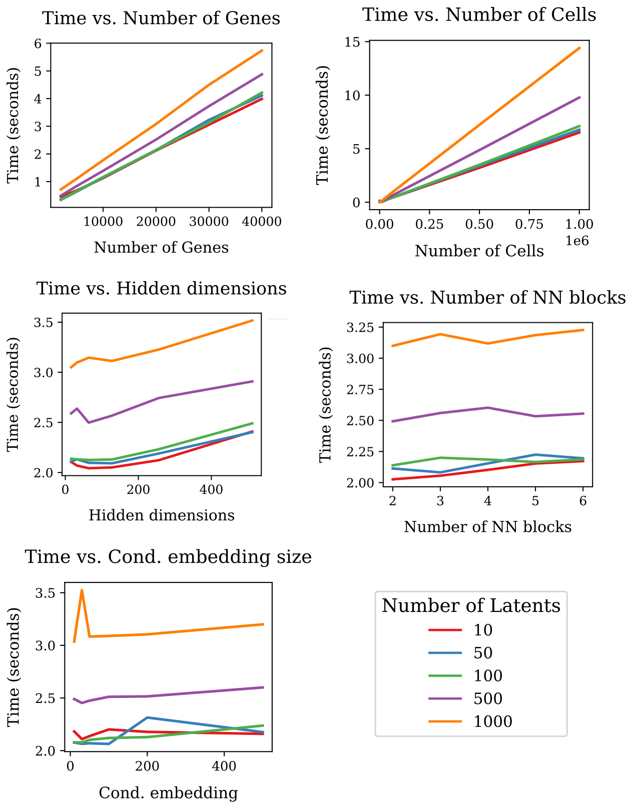

Figure A1: Analysis of the runtime of the CFGen generation process. Each panel represent a different hyperparameter setup. Different lines in every plot correspond to the dimensionality of the latent codes (hence, the dimensionality of the generation space). We test how the generation runtime varies as a function of: the number of genes, the number of cells, the number of block units in the denoising model, the size of the condition embedding and the hidden dimension of the denoising model. Results are reported in seconds. When one hyperparameter is changed dynamically, the others are set to a default value.

In Fig. A1, we notice that the hyperparameters impacting the generation speed the most are the number of cells and genes, while hyperparameter related to the neural network size are not as impactful. Expectedly, the number of latent codes significantly influences how fast the model can sample since it represents the dimensionality of the simulation space.

We additionally provide training and sampling runtimes for CFGen and competing models across datasets (see Table 5 and Table 6).

Table 5: Training runtime table. Each entry corresponds to the time in seconds required to train a model on different datasets. The number of cells and genes composing each dataset are reported at the bottom of the table. CFGen and scDiffusion are broken down in their different components that should be considered additively for an overview of the total runtime. For all the models, the batch size is set to 128.

	PBMC3K	Dentate gyrus	Tabula muris	HLCA	PBMC10K (scRNA-seq)
CFGen FM	1.02	7.13	69.00	192.12	3.23
CFGen AE	1.40	6.31	68.40	253.21	6.30
scVI	0.08	2.11	18.13	65.12	2.15
MultiVI	-	-	-	-	22.12
scDiffusion DM	1.03	2.13	18.02	53.62	4.48
scDiffusion AE	0.98	7.14	165.6	329.02	7.39
scDiffusion classifier	0.01	0.71	9.66	26.32	0.39
scGAN	0.98	5.15	20.41	181.12	2.40
No. of cells	2,638	18,21	245,389	584,944	10,025
No. of genes	8,573	17,00	19,734	27,997	25,604

Table 6: Generation runtime table. Each entry corresponds to the time in seconds required for a model to generate as many cells and genes as in the original dataset. The number of cells and genes composing each dataset are reported at the bottom of the table.

	PBMC3K	Dentate gyrus	Tabula muris	HLCA	PBMC10K (scRNA-seq)
CFGen	0.34	0.26	3.68	8.62	0.43
scVI	0.01	0.02	1.26	3.63	0.03
MultiVI	-	-	-	-	0.03
scDiffusion	48.79	105.08	1255.41	2004.00	113.41
scGAN	0.70	0.94	4.15	12.39	0.68
No. of cells	2,638	18,21		584,944	10,025
No. of genes	8,573	17,00	19,734	27,997	25,604

H.2 ADDITIONAL COMPARISONS ON SINGLE-CELL PROPERTY GENERATION.

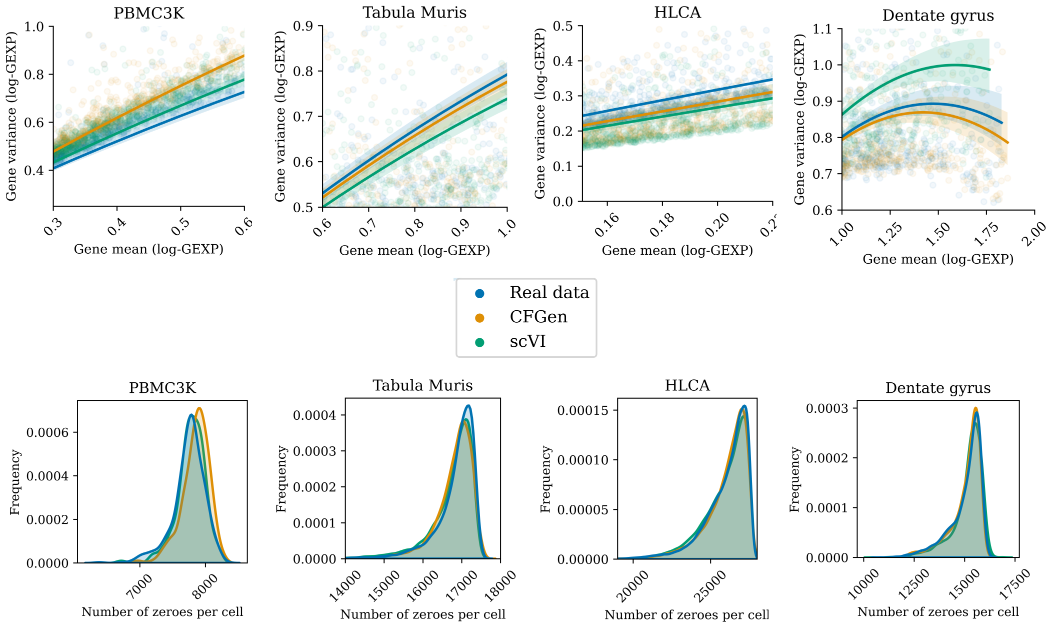

Figure A2: Additional results on modeling the properties of scRNA-seq with CFGen and scVI. Both models explicitly account for key properties in single-cell data, such as sparsity, discreteness and over-dispersion. The top row represents the comparison in terms of the mean-variance trend. We subset the x-axis in the results to the values for which the performance of the models differs the most for demonstrative purposes. In the bottom row, we compare the real and generated distributions of the number of zeroes per cell.

Table 7: Table representing the distribution distance between sparsity and mean-variance trend vectors between real and generated data (the lower, the better). The left part of the table is obtained by computing the vectors representing the number of zeroes per cell for real and generated data and computing the 1D Wasserstein distance between them. The same distance is computed between the mean/variance ratio vectors of real and generated data. Results are reported across three experimental repetitions.

	Sparsity distance real-generated (WD (↓))				Mean-variance trend distance real-generated (WD (↓))			
	PBMC3K	Dentate.	HLCA	T. Muris	PBMC3K	Dentate.	HLCA	T. Muris
CFGen	117.01±5.32	**27.44±3.27**	**37.87±0.49**	**59.63±1.77**	0.14±0.01	**0.03±0.00**	**0.39±0.00**	**0.03±0.00**
scDiff.	1740.15±5.51	1533.40±0.93	3215.39±0.35	466.68±0.03	1.02±0.01	0.74±0.01	1.21±0.02	0.65±0.02
scGAN	200.78±2.91	528.34±0.67	1395.82±0.53	537.48±1.66	0.44±0.00	2.73±0.73	0.81±0.00	35.92±27.98
scVI	**35.64±3.07**	102.03±2.31	106.28±1.06	93.13±1.08	**0.04±0.01**	0.04±0.00	0.52±0.00	0.07±0.00

H.3 EXAMPLE OF SYNTHETIC GENERATION BY CFGEN

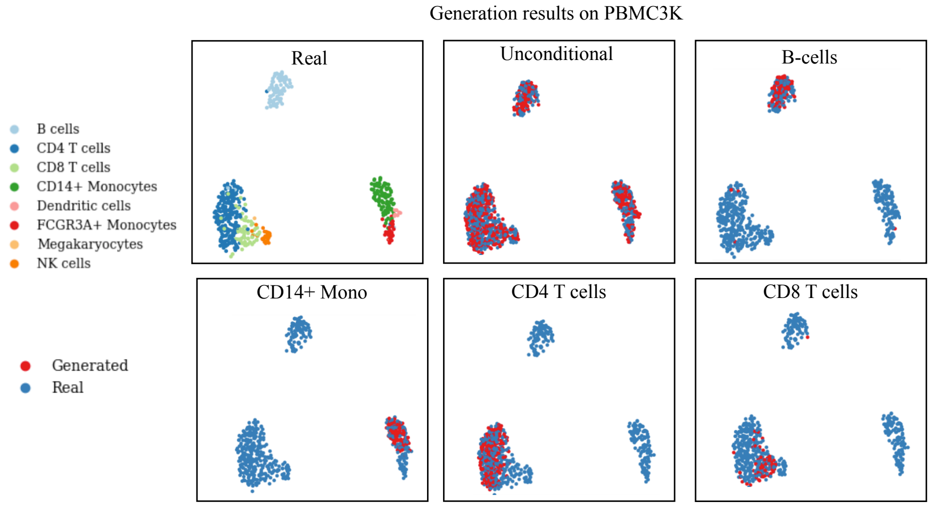

Figure A3: Uni-modal generation of scRNA-seq by CFGen on the PBMC3K dataset. Real and generated cells are embedded together and visualized as 2D UMAP coordinates.

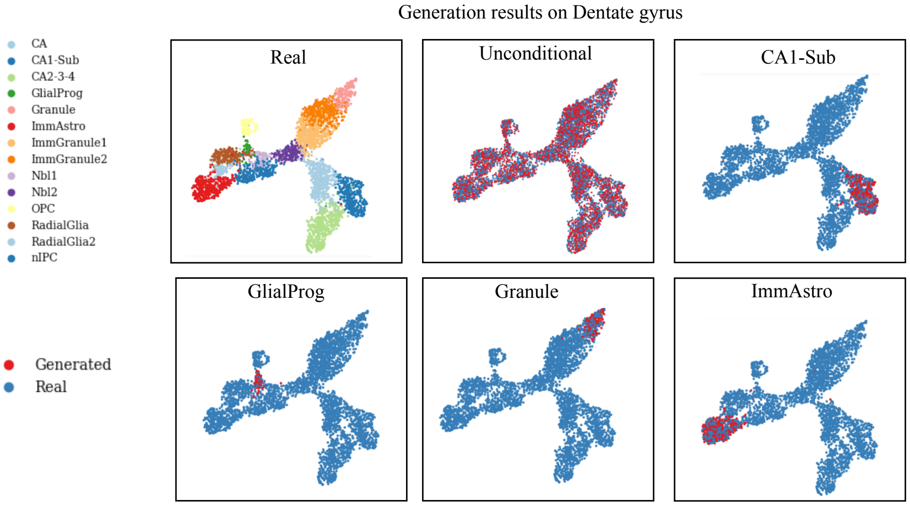

Figure A4: Uni-modal generation of scRNA-seq by CFGen on the Dentate gyrus dataset. Real and generated cells are embedded together and visualized as 2D UMAP coordinates.

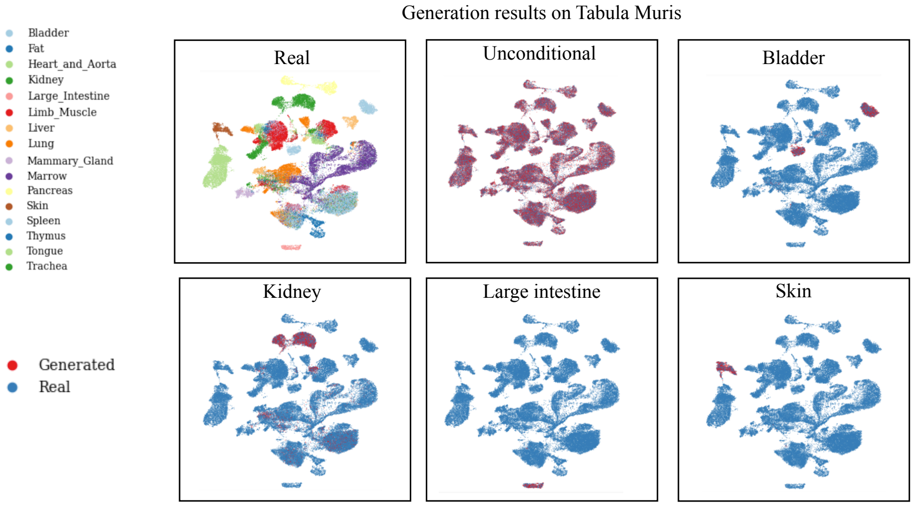

Figure A5: Uni-modal generation of scRNA-seq by CFGen on the Tabula Muris dataset. Real and generated cells are embedded together and visualized as 2D UMAP coordinates.

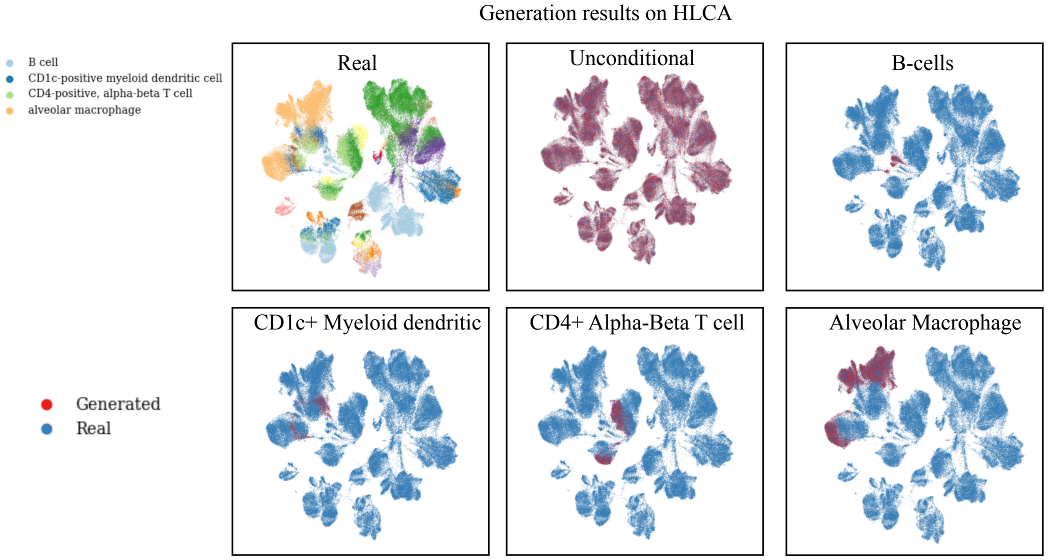

Figure A6: Uni-modal generation of scRNA-seq by CFGen on the HLCA dataset. Real and generated cells are embedded together and visualized as 2D UMAP coordinates.

H.4 COMPARISON BETWEEN CFGEN, SCVI AND SCDIFFUSION ON THE HLCA AND TABULA MURIS DATASETS

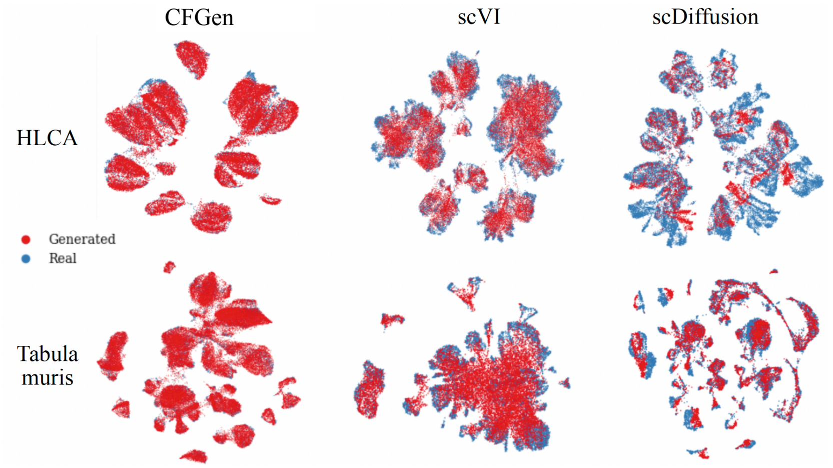

Figure A7: Qualitative comparison of the generation results of CFGen, scVI and scDiffusion on the HLCA and Tabula Muris datasets. Comparison is performed by evaluating the similarity of the generated results to real cells. Real and generated cells for all models are embedded together and visualized as 2D UMAP coordinates.

H.5 ADDITIONAL RESULTS ON MULTIMODAL GENERATION

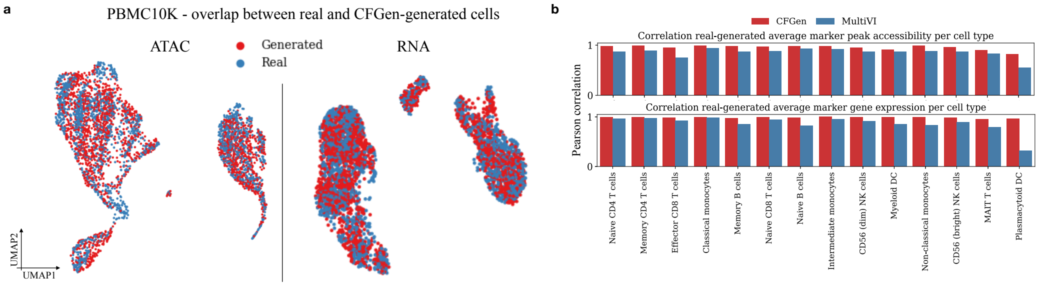

Figure A8: **(a)** 2D UMAP overlap between real and generated cells across modalities on the PBMC10K dataset. **(b)** Pearson correlation between average cell-type-specific marker peak accessibility and marker gene expression between real data and samples from CFGen and MultiVI.

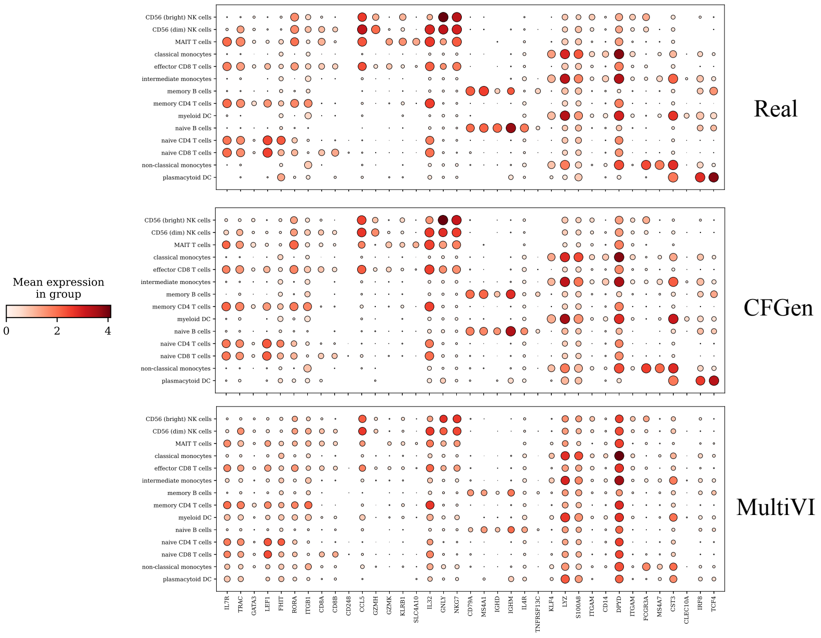

Figure A9: Average marker expression per cell type in real and generated data on the PBMC10k dataset. **x-axis** - marker genes. **y-axis** - cell types.

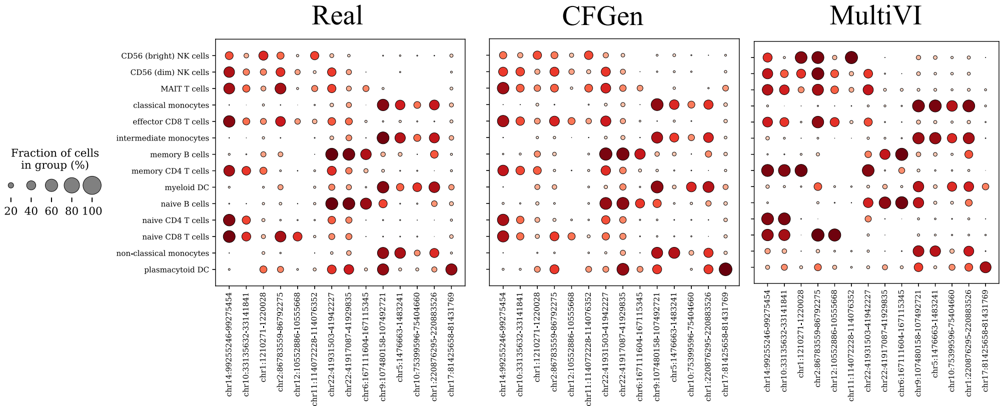

Figure A10: Average number of cells with accessible marker peaks per cell type in real and generated data on the PBMC10k dataset. **x-axis** - marker peaks. **y-axis** - cell types.

H.6 COMPARISON WITH BASELINES ON DATA AUGMENTATION FOR RARE CELL TYPES

Table 8: Average accuracy, precision and recall across cell types on held-out patients with and without prior augmentation. The best (bold) and second-best (underlined) performances are highlighted. CFGen-rare stands for an instance of CFGen where only rare cell types with less than 5000 instances are augmented to 5000 cells.

	PBMC COVID			HLCA		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
No Aug.	0.65	**0.77**	0.65	0.67	0.64	0.73
CFGen	**0.69**	0.67	**0.69**	**0.74**	0.67	**0.75**
CFGen - rare	0.67	0.73	0.67	0.72	0.68	0.71
scDiffusion	0.66	0.70	0.65	0.67	**0.69**	0.67
scVI	0.63	0.61	0.63	0.68	0.63	0.68

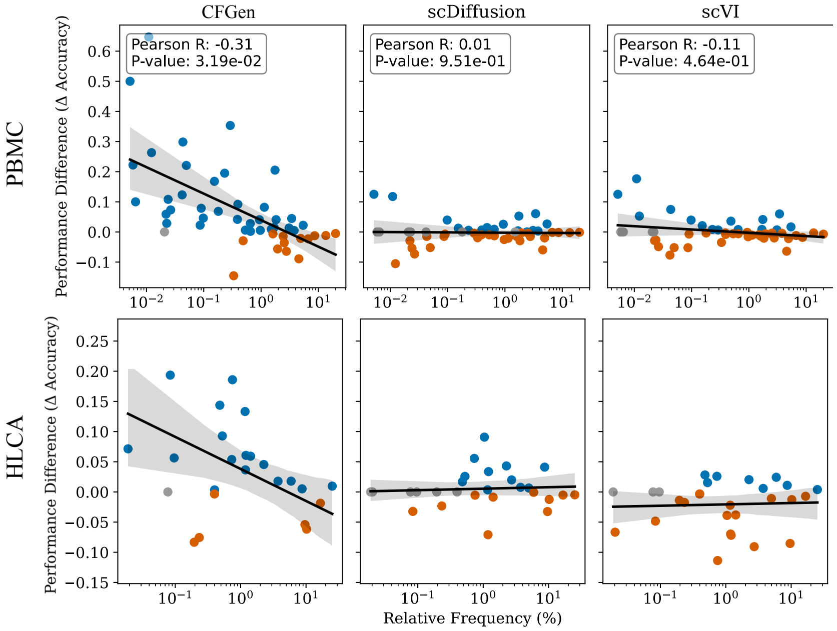

Figure A11: Extension of Fig. 4. Comparison of CFGen with scDiffusion and scVI on boosting the scGPT classifier performance on rare cell types.

Together with scGPT, in Fig. A12 we investigate if using CFGen to augment individual cell types improves the performance of a linear classifier like CellTypist (Cippà & Mueller, 2023). We obtain a similar result as scGPT, with the accuracy performance on real cell types improving after augmentation (hence a negative correlation between the performance improvement and the cell type frequency). For a better appreciation of the classification improvement of single cell type categories, we include Table 9 and Table 10.

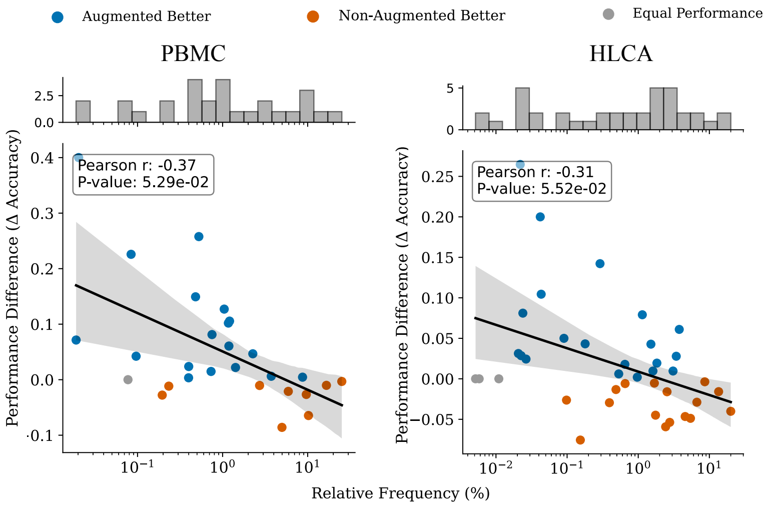

Figure A12: Performance improvement by data augmentation on a linear classifier. The plot displays the cell-type classification difference as a function of cell type frequency before and after augmentation on PBMC COVID and HLCA datasets. As a classifier, we use CellTypist (Cippà & Mueller, 2023), which is based on logistic regression. The held-out set includes cells from 20% of donors for both datasets.

Table 9: Table reporting the cell type classification accuracy of a linear classifier before (Accuracy Base) and after (Accuracy Aug) augmentation on the PBMC covid dataset. The Relative Frequency (%) column reports how rare a certain cell type is in the dataset. For each row, we highlight which setting leads to the highest accuracy.

Cell Type	Relative Frequency (%)	Accuracy Base	Accuracy Aug
naive thymus-derived CD4-positive, alpha-beta ...	25.18	0.87	**0.90**
classical monocyte	16.61	0.98	0.98
natural killer cell	10.22	**0.91**	0.90
CD4-positive helper T cell	9.63	**0.80**	0.75
naive thymus-derived CD8-positive, alpha-beta ...	8.70	**0.83**	0.79
naive B cell	5.92	0.96	**0.98**
CD8-positive, alpha-beta cytotoxic T cell	4.98	**0.78**	0.67
non-classical monocyte	3.73	0.95	**0.96**
central memory CD8-positive, alpha-beta T cell	2.72	**0.45**	0.33
regulatory T cell	2.27	0.28	**0.32**
conventional dendritic cell	1.42	0.79	**0.84**
CD16-negative, CD56-bright natural killer cell ...	1.21	0.66	**0.70**
gamma-delta T cell	1.19	0.49	**0.63**
effector memory CD8-positive, alpha-beta T cell ...	1.17	0.21	**0.27**
class switched memory B cell	1.04	0.51	**0.71**
B cell	0.75	0.21	**0.39**
mucosal invariant T cell	0.73	0.69	**0.70**
CD4-positive, alpha-beta cytotoxic T cell	0.53	0.06	**0.13**
effector memory CD8-positive, alpha-beta T cell	0.48	0.10	**0.23**
plasmacytoid dendritic cell	0.40	1.00	1.00
platelet	0.40	1.00	1.00
plasma cell	0.23	**0.97**	0.90
hematopoietic precursor cell	0.20	0.97	**0.86**
mature NK T cell	0.10	0.00	**0.16**
innate lymphoid cell	0.08	0.21	**0.40**
erythrocyte	0.08	1.00	1.00
dendritic cell	0.02	0.47	**0.80**
plasmablast	0.02	0.93	**1.00**
granulocyte	0.00	1.00	1.00

Table 10: Table reporting the cell type classification accuracy of a linear classifier before (Accuracy Base) and after (Accuracy Aug) augmentation on the HLCA dataset. The Relative Frequency (%) column reports how rare a certain cell type is in the dataset. For each row, we highlight which setting leads to the highest accuracy.

Cell Type	**Relative Frequency (%)**	**Accuracy Base**	**Accuracy Aug**
alveolar macrophage	20.00	0.95	0.95
type II pneumocyte	13.51	0.99	0.99
respiratory basal cell	8.63	0.92	0.90
ciliated columnar cell of tracheobronchial tree	6.67	**0.97**	0.94
nasal mucosa goblet cell	5.43	0.88	**0.89**
CD8-positive, alpha-beta T cell	4.93	**0.89**	0.87
club cell	4.57	**0.62**	0.53
elicited macrophage	3.77	0.70	0.70
CD4-positive, alpha-beta T cell	3.43	0.59	**0.64**
vein endothelial cell	3.09	0.93	**0.94**
capillary endothelial cell	2.77	**0.92**	0.85
alveolar type 2 fibroblast cell	2.54	**0.94**	0.90
classical monocyte	2.43	0.87	0.87
CD1c-positive myeloid dendritic cell	1.95	**0.73**	0.64
pulmonary artery endothelial cell	1.83	0.68	**0.75**
lung macrophage	1.75	0.36	**0.57**
type I pneumocyte	1.69	0.94	**0.95**
non-classical monocyte	1.61	**0.55**	0.54
natural killer cell	1.51	0.81	**0.83**
multi-ciliated epithelial cell	1.14	0.55	**0.66**
endothelial cell of lymphatic vessel	0.97	0.91	**0.93**
epithelial cell of lower respiratory tract	0.94	0.86	**0.88**
mast cell	0.65	0.96	0.96
B cell	0.65	0.88	**0.90**
plasma cell	0.53	0.98	0.98
alveolar type 1 fibroblast cell	0.49	**0.82**	0.79
bronchus fibroblast of lung	0.39	0.67	**0.77**
respiratory hillock cell	0.39	0.78	**0.82**
tracheobronchial smooth muscle cell	0.33	**0.78**	0.65
epithelial cell of alveolus of lung	0.29	0.18	**0.55**
bronchial goblet cell	0.23	0.04	**0.11**
plasmacytoid dendritic cell	0.18	0.87	**0.94**
acinar cell	0.15	0.67	**0.81**
lung pericyte	0.10	0.88	**0.89**
ionocyte	0.09	0.77	**0.85**
T cell	0.09	**0.57**	0.55
tracheobronchial serous cell	0.05	0.43	**0.64**
myofibroblast cell	0.04	0.25	**0.69**
conventional dendritic cell	0.04	0.58	**0.81**
mucus secreting cell	0.03	0.61	0.61
dendritic cell	0.02	0.46	**0.70**
mesothelial cell	0.02	0.91	**1.00**
smooth muscle cell	0.02	0.09	**0.15**
lung neuroendocrine cell	0.02	0.97	0.97
brush cell of tracheobronchial tree	0.01	0.21	**0.37**
stromal cell	0.01	0.35	**0.88**
fibroblast	0.01	0.30	**0.60**
hematopoietic stem cell	0.01	0.78	**0.89**
tracheobronchial goblet cell	0.01	0.00	**0.50**

H.7 MISSING GENE IMPUTATION WITH CFGEN

In the scVI model (Lopez et al., 2018), 10% of data entries are masked and set to zero, with the model trained on this corrupted data. During inference, masked cells are passed through the encoder, and latent codes $\mathbf{z} \sim q_\psi(\cdot|\mathbf{x})$ are sampled from the posterior. The VAE, trained to handle noisy inputs, decodes $\mathbf{z}$ to infer masked counts. Similarly, we propose an imputation strategy using CFGen as follows:

- Train CFGen on noisy data.
- Encode a noisy input $\mathbf{x}$ into the latent representation $\mathbf{z}_1 = f_\psi(\mathbf{x})$.
- Invert the generative flow to compute $\mathbf{z}_0 = \phi_0(\mathbf{z}_1)$, mapping $\mathbf{z}_1$ to its standard normal representation.
- Sample around $\mathbf{z}_0$ as $\mathbf{z}_0' \sim \mathcal{N}(\mathbf{z}_0, \sigma^2 I_d)$.
- Transport $\mathbf{z}_0'$ back to $\mathbf{z}_1' = \phi_1(\mathbf{z}_0')$, then decode to impute gene values for $\mathbf{x}$.

We tested this strategy on four datasets, masking 10% of the counts. Fig. A13 shows that our predictions correlate with pre-masking data, and Table 11 demonstrates superior imputation accuracy compared to scVI in three out of four datasets (Pearson correlation, mean absolute distance). Fig. A15 highlights that σ should remain below 0.1 to avoid sampling distant $\mathbf{z}_0'$ values, which generate unrelated cells and disrupt correlations with original gene expressions. These results confirm the effectiveness of our model for community-oriented applications.

Table 11: Mean distance and correlation between real and imputed genes by scVI and CFGen.

	Mean L_1 distance real-imputed counts (↓)				Pearson correlation real-imputed counts (↑)			
	PBMC3K	Dentate.	HLCA	T. Muris	PBMC3K	Dentate.	HLCA	T. Muris
CFGen	**1.21**	0.42	**3.21**	**4.81**	**0.68**	0.56	**0.83**	**0.86**
scVI	1.47	**0.35**	4.43	6.08	0.61	**0.58**	0.75	0.79

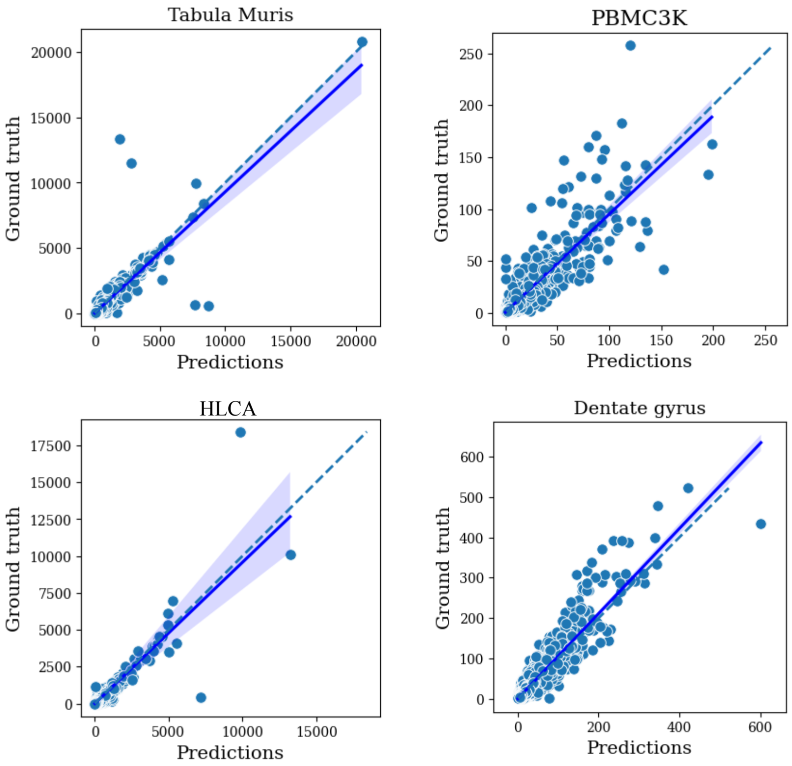

Figure A13: Scatterplot between imputed and real gene expression values before masking across datasets. Correlations can be found in Table 11.

In Fig. A15, we study how the quality of the imputation by CFGen varies as a function of the amount of noise used to sample around an observation. Notably, a higher noise leads to worse imputation results, since the generative modeling aspect takes over and samples a completely new cell which loses the structure of the originally encoded noisy observation.

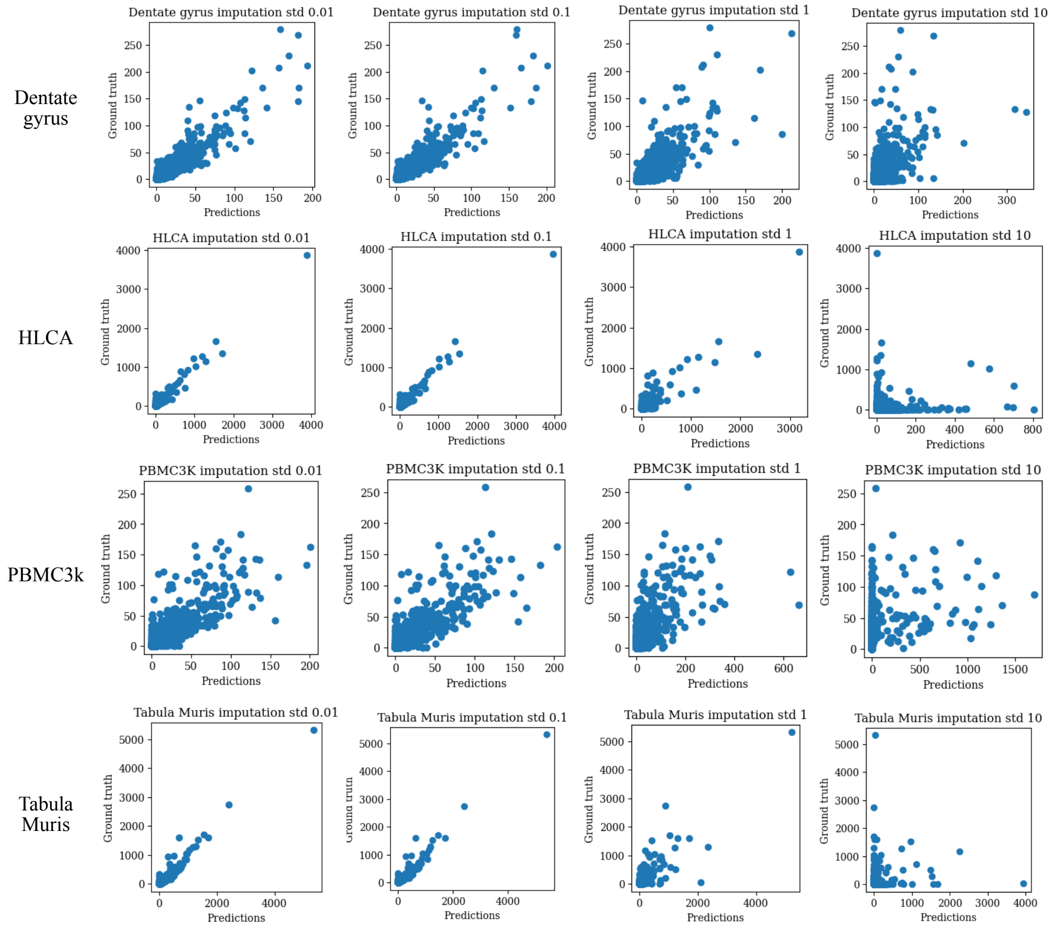

Figure A14: Correlation between the CFGen-imputed and real gene expression before masking as a function of the amount of added noise. Rows represent different datasets, columns stand for the standard deviation of the noise used to sample around the latent representation of the cell. Perfect correlation along the bisector is the best possible imputation result.

H.8 ADDITIONAL RESULTS MULTI-LABEL GENERATION

Table 12: Extension to Fig. 3. We train two 3-layer MLP with softmax head on the real data to predict the classes of the two attributes considered for each dataset. For different levels of the combination of guidance weights, the classifier is applied to predict the average probability that the generated observations are of a certain guidance class. When guided on a single attribute it is expected that generated cells are assigned with high probability only to the class of such an attribute. As guidance strength increases for the counterpart attribute, CFGen models the intersections between attributes increasingly better and, therefore, enables high classification probability for both guiding labels.

	NeurIPS			Tabula Muris	
Weights	$p(\text{CD14} + \text{M.})$	$p(\text{donor 1})$	Weights	$p(\text{Tongue})$	$p(\text{18-M-52})$
$\omega_{\text{donor}} = 0$ $\omega_{\text{cell type}} = 1$	0.98	0.40	$\omega_{\text{mouse ID}} = 0.0$ $\omega_{\text{Tissue}} = 1$	0.98	0.19
$\omega_{\text{donor}} = 1$ $\omega_{\text{cell type}} = 1$	0.96	0.87	$\omega_{\text{mouse ID}} = 1$ $\omega_{\text{Tissue}} = 1$	0.98	0.69
$\omega_{\text{donor}} = 2.5$ $\omega_{\text{cell type}} = 1$	0.96	1.00	$\omega_{\text{mouse ID}} = 2.5$ $\omega_{\text{Tissue}} = 1$	0.96	0.97

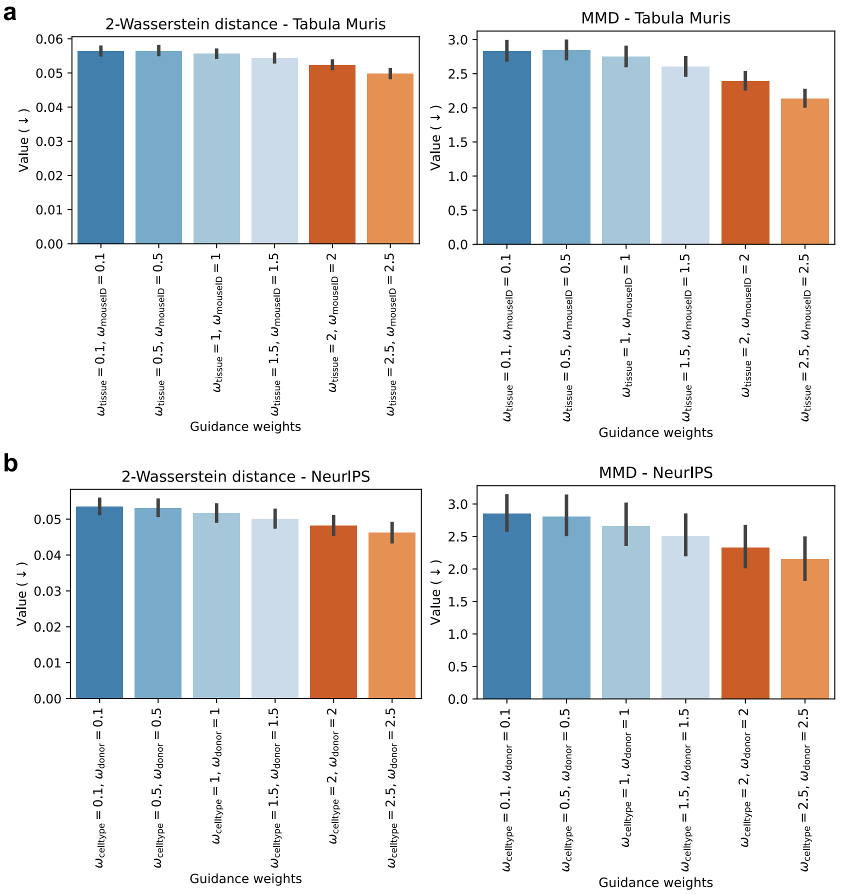

Figure A15: Performance on the generation of intersection of attributes on the Tabula Muris (top) and NeurIPS (bottom) datasets based on distributional metrics. On the x-axis, we increase the guidance parameters for both conditioning attributes.

H.9 ADDITIONAL RESULTS ON MULTI-ATTRIBUTE GUIDANCE RESULTS

Table 13: The average Batch correction and Bio conservation metrics from the scIB package evaluate at different levels of guidance strength.

	C. Elegans		NeurIPS	
Guidance weights	Batch Correction	Bio Conservation	Batch Correction	Bio Conservation
$\omega_{\text{bio}} = 0$ $\omega_{\text{batch}} = 0$	0.48	0.55	0.32	0.63
$\omega_{\text{bio}} = 1$ $\omega_{\text{batch}} = 1$	0.67	0.55	0.61	0.64
$\omega_{\text{bio}} = 1$ $\omega_{\text{batch}} = 2$	0.68	0.54	0.64	0.73
$\omega_{\text{bio}} = 2$ $\omega_{\text{batch}} = 1$	0.68	0.63	0.63	0.61
$\omega_{\text{bio}} = 2$ $\omega_{\text{batch}} = 2$	0.68	0.63	0.64	0.71
$\omega_{\text{bio}} = 2$ $\omega_{\text{batch}} = 3$	0.69	0.64	0.65	0.70
$\omega_{\text{bio}} = 3$ $\omega_{\text{batch}} = 2$	0.70	0.67	0.65	0.77
$\omega_{\text{bio}} = 3$ $\omega_{\text{batch}} = 3$	0.70	0.66	0.66	0.75
$\omega_{\text{bio}} = 3$ $\omega_{\text{batch}} = 4$	0.70	0.67	0.66	0.73
$\omega_{\text{bio}} = 4$ $\omega_{\text{batch}} = 4$	0.70	0.69	0.67	0.77

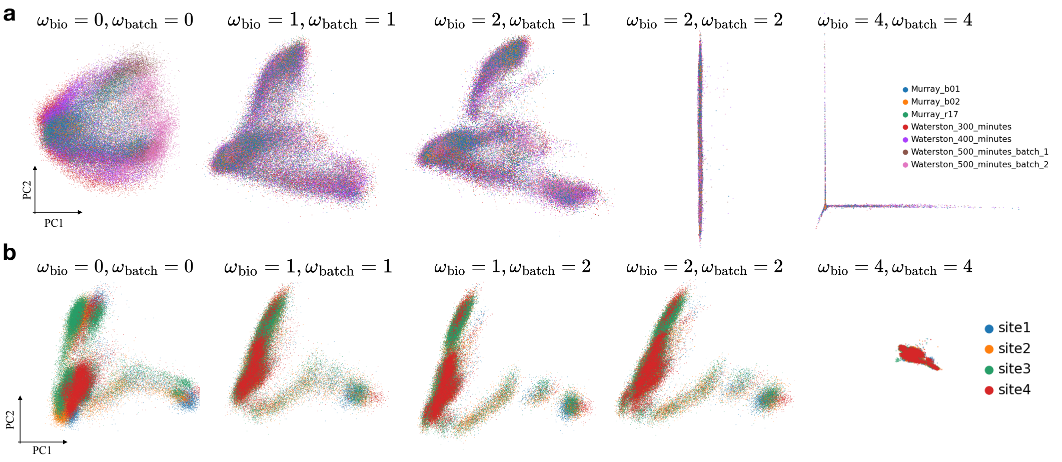

Figure A16: The PCA plot of generated cells colored by batch for the C.Elegans (**a**) and NeurIPS **(b)** datasets. Each column represent a different guidance strength value.

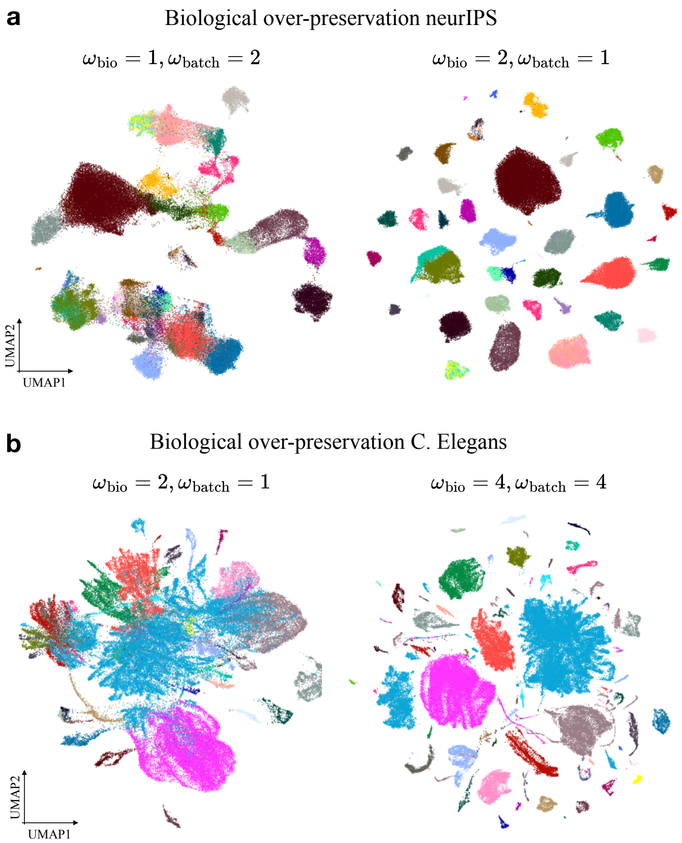

Figure A17: The UMAP plot of generated cells colored by batch for the NeurIPS (**a**) and C.Elegans **(b)** datasets. We show one example of generation with a reasonable guidance scheme (left columns) and one with a guidance scheme causing unrealistic cell type distributions (right).

H.10 ADDITIONAL PLOTS

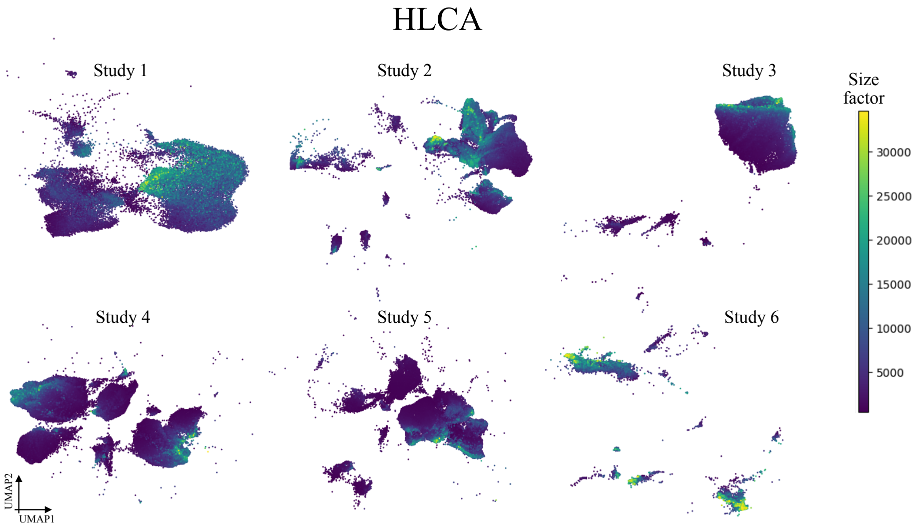

Figure A18: UMAP plots of six studies included in the HLCA dataset colored by size factor.