A LLM USAGE DISCLAIMER

We used large language models (ChatGPT and Claude) to assist with manuscript polishing, including grammar and clarity improvements, and to verify technical definitions and terminology. All scientific content, analysis, and conclusions are the original work of the authors.

B DEFINITIONS

Self-attention Attention mechanism is defined as $\mathrm{Att}(\mathbf{Q},\mathbf{K},\mathbf{V}) \overset{df}{=} \mathrm{softmax}(\mathbf{Q}\mathbf{K}^\top)\mathbf{V}$, where $\mathbf{Q}\in\mathbb{R}^{m\times D}, \mathbf{K}\in\mathbb{R}^{M\times D}, \mathbf{V}\in\mathbb{R}^{M\times D}$, and $\mathrm{softmax}(\cdot)$ is the row-wise softmax function.[4] Typically, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are results of linear transformations of given inputs. If all matrices are calculated based on the same input $\mathbf{X}$, we refer to it as *self-attention*.

Cross-attention However, if $\mathbf{Q}$ is calculated using $\mathbf{X}$, and another input $\mathbf{Y}$ is used to obtain $\mathbf{K}$ and $\mathbf{V}$, then we refer to it as *cross-attention*.

Multi-head attention Att_K denotes a multi-head attention with K heads, i.e., a concatenation of K attention layers.

Multi-head Attention Block (MAB) A multi-head attention block (MAB) is defined as follows Lee et al. (2019); Zhang et al. (2022):

$$\begin{aligned}\mathrm{MAB}(\mathbf{X},\mathbf{Y}) &\overset{df}{=} f(\mathbf{X},\mathbf{Y}) + \mathrm{relu}(f(\mathbf{X},\mathbf{Y})\mathbf{W}), \\ f(\mathbf{X},\mathbf{Y}) &\overset{df}{=} \mathbf{X}\mathbf{W}_f + \mathrm{Att}_K(\mathbf{Q}(\mathbf{X}),\mathbf{K}(\mathbf{Y}),\mathbf{V}(\mathbf{Y})),\end{aligned} \tag{9}$$

where $f(\mathbf{X},\mathbf{Y}) \overset{df}{=} \mathbf{X}\mathbf{W}_f + \mathrm{Att}_K(\mathbf{Q}(\mathbf{X}),\mathbf{K}(\mathbf{Y}),\mathbf{V}(\mathbf{Y}))$, $\mathbf{W}$ and $\mathbf{W}_f$ are learnable weight matrices.

Set Attention Block (SAB) Set Attention Block (SAB) is defined as follows Lee et al. (2019):

$$\mathrm{SAB}(\mathbf{X}) \overset{df}{=} \mathrm{MAB}(\mathbf{X},\mathbf{X}). \tag{10}$$

Induced Set Attention Block (ISAB) Induced Set Attention Block (ISAB) is defined as follows Lee et al. (2019):

$$\mathrm{ISAB}(\mathbf{X}) \overset{df}{=} \mathrm{MAB}\left(\mathbf{X}, \mathrm{MAB}(\mathbf{U},\mathbf{X})\right), \tag{11}$$

where $\mathbf{U}\in\mathbb{R}^{m\times D}$ are *inducing points*, i.e., a global weight matrix learnable by backpropagation.

Pooling by Multi-Head Attention Pooling by Multi-head Attention (PMA) is defined as follows:

$$\mathrm{PMA}(\mathbf{X}) = \mathrm{MAB}(\mathbf{S}, \mathrm{rFF}(\mathbf{X})), \tag{12}$$

where $\mathbf{S}\in\mathbb{R}^{m\times D}$ is a matrix of learnable *inducing points* (or *pseudoinputs*), and $\mathrm{rFF}:\mathbb{R}^{M\times D}\to\mathbb{R}^{M\times D}$ is a row-wise linear layer. For fixed inducing points $\mathbf{S}$ in PMA, this layer is permutation-invariant (this is due to applying the attention layer, see Property 3 in Appendix C).

C PERMUTATION-EQUIVARIANCE AND PERMUTATION-INVARIANCE OF ATTENTION MECHANISM

The row-wise self-attention function fulfills the following property:

Property 1. *For a given matrix $\mathbf{X}\in\mathbb{R}^{M\times D}$, and two permutation matrices $\mathbf{P}_M\in\{0,1\}^{M\times M}$ and $\mathbf{P}_D\in\{0,1\}^{D\times D}$, the following statements hold true for the row-wise softmax function: (i)* $\mathrm{softmax}(\mathbf{X}\mathbf{P}_D^\top) = \mathrm{softmax}(\mathbf{X})\mathbf{P}_D^\top$, *(ii)* $\mathrm{softmax}(\mathbf{P}_M\mathbf{X}) = \mathbf{P}_M\mathrm{softmax}(\mathbf{X})$.

[4] We skip scaling $\mathbf{Q}\mathbf{K}^\top$ by $1/\sqrt{D}$ to avoid unnecessary clutter.

Property 1 tells us that applying the permutation to the softmax function just reorders its columns/rows, hence, applying softmax before or after the reordering gives the same vector, merely shuffled.

Before we move to next properties, we recall that any permutation matrix is orthogonal, hence, $\mathbf{P}\mathbf{P}^\top = \mathbf{P}^\top\mathbf{P} = \mathbf{I}$.

It is a well-known fact that the (self-)attention mechanism is permutation-equivariant, namely, the following property holds true:

Property 2. *For a given permutation matrix* $\mathbf{P} \in \{0,1\}^{M\times M}$*, the attention mechanism is permutation-equivariant, i.e.,* $\mathrm{Att}\left(\mathbf{PQ}(\mathbf{X}), \mathbf{PK}(\mathbf{X}), \mathbf{PV}(\mathbf{X})\right) = \mathbf{P}\,\mathrm{Att}\left(\mathbf{Q}(\mathbf{X}), \mathbf{K}(\mathbf{X}), \mathbf{V}(\mathbf{X})\right)$.

Proof. First, we notice that: $\mathbf{Q}(\mathbf{PX}) = \mathbf{PXW}_Q = \mathbf{PQ}(\mathbf{X})$, $\mathbf{K}(\mathbf{PX}) = \mathbf{PXW}_K = \mathbf{PK}(\mathbf{X})$ and $\mathbf{V}(\mathbf{PX}) = \mathbf{PXW}_V = \mathbf{PV}(\mathbf{X})$. Next, to avoid unnecessary clutter, let us skip the dependency on $\mathbf{X}$. Then:

$$\begin{aligned}
\mathrm{Att}\left(\mathbf{PQ}, \mathbf{PK}, \mathbf{PV}\right) &= \mathrm{softmax}(\mathbf{PQ}(\mathbf{PK})^\top)\mathbf{PV} \\
&= \mathrm{softmax}(\mathbf{PQK}^\top\mathbf{P}^\top)\mathbf{PV} \\
&= \mathbf{P}\,\mathrm{softmax}(\mathbf{QK}^\top)\mathbf{P}^\top\mathbf{PV} \\
&= \mathbf{P}\,\mathrm{softmax}(\mathbf{QK}^\top)\mathbf{V} \\
&= \mathbf{P}\,\mathrm{Att}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right).
\end{aligned}$$

□

The attention mechanism becomes permutation-invariant for $\mathbf{Q}$ being a global parameter matrix only if the following property holds true:

Property 3. *For a given permutation matrix* $\mathbf{P} \in \{0,1\}^{M\times M}$*, the attention mechanism with inducing points* $\mathbf{Q} \in \mathbb{R}^{m\times D}$ *is permutation-invariant, i.e.,* $\mathrm{Att}\left(\mathbf{Q}, \mathbf{PK}(\mathbf{X}), \mathbf{PV}(\mathbf{X})\right) = \mathrm{Att}\left(\mathbf{Q}, \mathbf{K}(\mathbf{X}), \mathbf{V}(\mathbf{X})\right)$.

Proof. First, we notice that: $\mathbf{K}(\mathbf{PX}) = \mathbf{PXW}_K = \mathbf{PK}(\mathbf{X})$ and $\mathbf{V}(\mathbf{PX}) = \mathbf{PXW}_V = \mathbf{PV}(\mathbf{X})$. Next, to avoid unnecessary clutter, let us skip the dependency on $\mathbf{X}$. Then:

$$\begin{aligned}
\mathrm{Att}\left(\mathbf{Q}, \mathbf{PK}, \mathbf{PV}\right) &= \mathrm{softmax}(\mathbf{Q}(\mathbf{PK})^\top)\mathbf{PV} \\
&= \mathrm{softmax}(\mathbf{QK}^\top\mathbf{P}^\top)\mathbf{PV} \\
&= \mathrm{softmax}(\mathbf{QK}^\top)\mathbf{P}^\top\mathbf{PV} \\
&= \mathrm{softmax}(\mathbf{QK}^\top)\mathbf{V} \\
&= \mathrm{Att}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right).
\end{aligned}$$

□

However, for a latent matrix $\mathbf{Z} \in \mathbb{R}^{m\times D}$ obtained by transforming $\mathbf{X}$ in the permutation-invariant manner, an embedding matrix $\mathbf{E} \in \mathbb{R}^{V\times D}$, the inducing points determined by a set of indices $\mathcal{I}$, i.e., $\mathbf{Q} = \mathbf{E}_{\mathcal{I}}$, is permutation-equivariant if the indices $\mathcal{I}$ are permuted, i.e., a permutation of indices $\pi(\mathcal{I})$ induces the matrix $\mathbf{P}$, thus, $\mathbf{PQ} = \mathbf{E}_{\pi(\mathcal{I})}$. Then, the following property holds true:

Property 4. *For a given permutation of indices* $\mathcal{I}$*,* $\pi(\mathcal{I})$*, or, equivalently, a matrix permutation* $\mathbf{P} \in \{0,1\}^{|\mathcal{I}|\times|\mathcal{I}|}$*, and latents* $\mathbf{Z}$ *calculated by a permutation-invariant function* f*, i.e.,* $\mathbf{Z} = f(\mathbf{PX})$*, the attention mechanism with inducing points* $\mathbf{Q} \in \mathbb{R}^{|\mathcal{I}|\times D}$ *is permutation-equivariant, i.e.,* $\mathrm{Att}\left(\mathbf{PQ}, \mathbf{K}(f(\mathbf{PX})), \mathbf{V}(f(\mathbf{PX}))\right) = \mathbf{P}\mathrm{Att}\left(\mathbf{Q}, \mathbf{K}(\mathbf{Z}), \mathbf{V}(\mathbf{Z})\right)$.

Proof. First, since f is permutation-invariance, we get $\mathbf{Z} = f(\mathbf{PX}) = f(\mathbf{X})$. Second, we note that $\mathbf{K}(\mathbf{PZ}) = \mathbf{PZW}_K = \mathbf{PK}(\mathbf{Z})$ and $\mathbf{V}(\mathbf{PZ}) = \mathbf{PZW}_V = \mathbf{PV}(\mathbf{Z})$. Then:

$$\begin{aligned}
\mathrm{Att}\left(\mathbf{PQ}, \mathbf{K}(f(\mathbf{PX})), \mathbf{V}(f(\mathbf{PX}))\right) &= \mathrm{softmax}(\mathbf{PQK}(f(\mathbf{PX}))^\top)\mathbf{V}(f(\mathbf{PX})) \\
&= \mathbf{P}\,\mathrm{softmax}(\mathbf{QK}(f(\mathbf{X}))^\top)\mathbf{V}(f(\mathbf{X})) \\
&= \mathbf{P}\,\mathrm{softmax}(\mathbf{QK}(\mathbf{Z})^\top)\mathbf{V}(\mathbf{Z}) \\
&= \mathbf{P}\,\mathrm{Att}\left(\mathbf{Q}, \mathbf{K}(\mathbf{Z}), \mathbf{V}(\mathbf{Z})\right).
\end{aligned}$$

□

D Related Work (extended)

Modeling a probability distribution over order-agnostic objects like sets is challenging for at least two reasons. First, a model must be permutation-equivariant, meaning, changing the order of variables changes the order of parameters as well. Second, the model must also be exchangeable. Additionally, in the case of gene expression, for various tissues, we get different subsets of genes, thus, ideally, we would like to learn a single model to *transfer* hidden dependencies (correlations) among cells from distinct tissues.

Autoregressive Models A varying-size objects are typically modeled by autoregressive models (ARMs) like transformer-based LLMs for text (Vaswani et al., 2017) or WaveNet for audio (Van Den Oord et al., 2016). However, ARMs assume a fixed order of variables, otherwise, like in the case of sets, their performance can drop significantly. Recently, it has been shown that misspecifying the order in ARMs can result in a huge drop in their performance (Kim et al., 2025). There are ways of dealing with the order in ARMs (Pannatier et al., 2024), but they are not well-suited for processing objects without an explicitly defined order.

Masked Diffusion Models Recently, a masked version of diffusion-based models (Ho et al., 2020) are used to generate text quite successfully (Nie et al., 2025) since they can alleviate the need of specifying the order of generation. However, as proven in (Kim et al., 2025), masked diffusion-based models are order-agnostic but at the price of learning an extremely complex task of predicting a variable value conditioned on a set of unmasked variables in arbitrary positions.

Variational Auto-Encoders Another modeling approach is to define a Variational Auto-Encoder (Kingma & Welling, 2014; Rezende et al., 2014) since this framework allows defining its components in a flexible manner. In (Kim et al., 2021), a SetVAE was formulated by introducing two separate latent variables to deal with varying size of sets, namely, $\mathbf{z}_{\mathcal{I}}$ of the same dimensionality as $\mathbf{x}_{\mathcal{I}}$ such that $\mathbf{z}_i$ corresponds to $\mathbf{x}_i$, $i \in \mathcal{I}$, and an additional vector of latents $\mathbf{c} \in \mathbb{R}^{d_1}$ of a constant size d_1. In general, $\mathbf{x}_{\mathcal{I}}$ can be generated given $\mathbf{z}_{\mathcal{I}}$ and each $\mathbf{z}_i$ is generated given $\mathbf{c}$, i.e., $p(\mathbf{x}_{\mathcal{I}}|\mathcal{I}, \mathbf{z}_{\mathcal{I}})\, p(\mathbf{z}_{\mathcal{I}}|\mathbf{c})\, p(\mathbf{c})$, where $p(\mathbf{x}_{\mathcal{I}}|\mathcal{I}, \mathbf{z}_{\mathcal{I}}) = \prod_{i=1}^{|\mathcal{I}|} p(\mathbf{x}_i|\mathbf{z}_i)$ and $p(\mathbf{z}_{\mathcal{I}}|\mathbf{c}) = p(|\mathcal{I}|) \prod_{i=1}^{|\mathcal{I}|} p(\mathbf{z}_i|\mathbf{c})$. Then, the variational posteriors can take the following form: $q(\mathbf{z}_{\mathcal{I}}|\mathbf{x}_{\mathcal{I}}) = \delta(|\mathcal{I}|) \prod_{i=1}^{|\mathcal{I}|} q(\mathbf{z}_i|\mathbf{x}_i)$, where $\delta(\cdot)$ is Dirac's delta, and additionally we have $q(\mathbf{c}|\mathbf{x}_{\mathcal{I}})$. In (Kim et al., 2021), a few simplifications were made such that the model fits well modeling point clouds (sets of 3-D points), namely, for all $i = 1, \ldots, |\mathcal{I}|$, $p(\mathbf{z}_i|\mathbf{c}) = p(\mathbf{z}_i)$, and $q(\mathbf{z}_i|\mathbf{c}) = p(\mathbf{z}_i)$. Further, the authors of (Kim et al., 2021) suggested to define a hierarchical VAE with multiple layers of $\mathbf{z}_{\mathcal{I}}$'s and $\mathbf{c}$'s since a single layer did not result in good performance, and they replaced the conditional likelihood with Chamfer Distance as a well-suited distance for point clouds. In this paper, we find a great appeal of the VAE framework and its flexibility; however, we claim using two distinct latents and a hierarchical latent structure to be unnecessary. Instead, we suggest picking a careful parameterization to be *crucial* in obtaining high performance.

Permutation-equivariant/invariant Parameterizations Deep Neural Networks are widely used as transformations of raw data and parameterizations of probability distributions. It is advocated (but also observed empirically) that modeling probability distributions requires utilizing symmetries in data (Bronstein et al., 2021). For instance, for objects whose dimensions can be shuffled without changing the underlying latent structure, we need either *permutation-invariant* or *permutation-equivariant* transformations. For a given permutation matrix $\mathbf{P}$, a function $f : \mathbb{X} \to \mathbb{Y}$ is *permutation-invariant* if $f(\mathbf{Px}) = \mathbf{y}$; on the other hand, a function $f : \mathbb{X} \to \mathbb{Y}$ is *permutation-equivariant* if $f(\mathbf{Px}) = \mathbf{Py}$.

A general *blueprint* for composing geometric deep neural networks is a composition of permutation-invariant layers and/pr permutation-equivariant layers, with nonlinearity activation functions in between (Bronstein et al., 2021). An example of such a blueprint is an architecture called DeepSets (Zaheer et al., 2017). It formulates a general permutation-equivariance layer treating all variables consistently regardless their positions. Then it applies a symmetric aggregation like averaging or other pooling operators (Kimura et al., 2024; Ilse et al., 2018; Xie & Tong, 2025) to combine these

equivariant features in a permutation-invariant fashion, ensuring the order of inputs does not affect the output. The drawback of this approach is that all elements are processed separately before being aggregated with a non-learnable pooling operator. This manner of constructing permutation-invariant transformations might be highly limiting.

A potential solution to that issue is replacing static pooling with a learned attention mechanism, allowing utilizing transformer-based architectures to transform all elements jointly in an equivariant and invariant fashion. An example of a fully-transformer-based model is SetTransformer (Lee et al., 2019) that builds on the following idea. Attention mechanism is defined as $\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \stackrel{df}{=} \mathrm{softmax}(\mathbf{Q}\mathbf{K}^\top)\mathbf{V}$, where $\mathbf{Q} \in \mathbb{R}^{m \times D}$, $\mathbf{K} \in \mathbb{R}^{M \times D}$, $\mathbf{V} \in \mathbb{R}^{M \times D}$, and $\mathrm{softmax}(\cdot)$ is the row-wise softmax function.[5] Typically, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are results of linear transformations of some inputs. If all matrices are calculated based on the same input $\mathbf{X}$, we refer to it as *self-attention*. However, if $\mathbf{Q}$ is calculated using $\mathbf{X}$, and another input $\mathbf{Y}$ is used to obtain $\mathbf{K}$ and $\mathbf{V}$, then we refer to it as *cross-attention*. Let Att_K denote a multi-head attention with K heads, i.e., a concatenation of K attention layers. SetTransformer introduces a multi-head attention block (MAB) Lee et al. (2019); Zhang et al. (2022):

$$\begin{aligned} \mathrm{MAB}(\mathbf{X}, \mathbf{Y}) &\stackrel{df}{=} f(\mathbf{X}, \mathbf{Y}) + \mathrm{relu}(f(\mathbf{X}, \mathbf{Y})\mathbf{W}), \\ f(\mathbf{X}, \mathbf{Y}) &\stackrel{df}{=} \mathbf{X}\mathbf{W}_f + \mathrm{Att}_K(\mathbf{Q}(\mathbf{X}), \mathbf{K}(\mathbf{Y}), \mathbf{V}(\mathbf{Y})), \end{aligned} \tag{13}$$

where $f(\mathbf{X}, \mathbf{Y}) \stackrel{df}{=} \mathbf{X}\mathbf{W}_f + \mathrm{Att}_K(\mathbf{Q}(\mathbf{X}), \mathbf{K}(\mathbf{Y}), \mathbf{V}(\mathbf{Y}))$, $\mathbf{W}$ and $\mathbf{W}_f$ are learnable weight matrices. Given MAB, SetTransformer further defines the following two blocks, namely, the Set Attention Block (SAB) and Induced Set Attention Block (ISAB): $\mathrm{SAB}(\mathbf{X}) \stackrel{df}{=} \mathrm{MAB}(\mathbf{X}, \mathbf{X})$, and $\mathrm{ISAB}(\mathbf{X}) \stackrel{df}{=} \mathrm{MAB}\left(\mathbf{X}, \mathrm{MAB}(\mathbf{U}, \mathbf{X})\right)$, where $\mathbf{U} \in \mathbb{R}^{m \times D}$ are *inducing points*, i.e., a global weight matrix learnable by backpropagation. ISAB allows to change the size of the input, and similarly to SAB, it is permutation-equivariant (Lee et al., 2019). To obtain a permutation-invariant transformation, SetTransformer proposes to use another layer called Pooling by Multi-head Attention (PMA):

$$\mathrm{PMA}(\mathbf{X}) = \mathrm{MAB}(\mathbf{S}, \mathrm{rFF}(\mathbf{X})), \tag{14}$$

where $\mathbf{S} \in \mathbb{R}^{m \times D}$ is a matrix of learnable *inducing points* (or *pseudoinputs*), and $\mathrm{rFF} : \mathbb{R}^{M \times D} \to \mathbb{R}^{M \times D}$ is a row-wise linear layer. For fixed inducing points $\mathbf{S}$ in PMA, this layer is permutation-invariant (this is due to applying the attention layer, see Property 3 in Appendix C). These building blocks can be used to formulate a deep neural network for parameterizing a probabilistic model. However, we advocate for a different parameterization that applies a single multi-head attention layer in a transformer block to obtain a fixed-size output, and then a series of transformer blocks.

Latent Diffusion Models Latent Diffusion Models (LDMs) perform diffusion processes in learned latent spaces rather than directly in high-dimensional data spaces. Stable Diffusion (Rombach et al., 2022) pioneered this approach for text-to-image synthesis by training diffusion models in the latent space of a pre-trained VAE, dramatically reducing computational costs while maintaining generation quality. This paradigm has proven effective across diverse scientific domains: all-atom diffusion transformers (Joshi et al., 2025) generate molecules and materials with atomic-level precision, similary LaM-SLidE (Sestak et al., 2025) utilizes transformer-based LMD for molecular dynamics (among others), while La-proteina (Geffner et al., 2025) employs transformer-based partially latent flow matching for atomistic protein generation. These advances demonstrate the versatility of latent diffusion approaches for complex, high-dimensional scientific data across multiple modalities. Here, we extend this framework to single-cell transcriptomics by proposing a transformer-based LDM for this biological data type.

Generative Models for scRNA-seq In the context of single-cell genomics, numerous generative models have been developed for (conditional) sampling of gene expression profiles. scVI (Lopez et al., 2018) represents an early VAE-based generative model, while more recent approaches include GAN-based and diffusion-based architectures such as scGAN (Marouf et al., 2020a) and scDiffusion (Luo et al., 2024). These models operate in continuous space and therefore transform discrete gene expression data into log-normalized counts. Recently, latent diffusion frameworks have emerged with models like SCLD (Wang et al., 2023) and CFGen (Palma et al., 2025a), which leverage latent diffusion frameworks. Additionally, application-specific generative models have been

[5] We skip scaling $\mathbf{Q}\mathbf{K}^\top$ by $1/\sqrt{D}$ to avoid unnecessary clutter.

developed for perturbational single-cell genomics, including CPA Lotfollahi et al. (2023a), SquiDiff (He et al., 2024), CellFlow (Klein et al., 2025), and CellOT (Bunne et al., 2023), which are tailored to capture the effects of genetic and chemical perturbations on cellular states. Our approach is similar in vein to CFGen and SCLD, but leverages transformer-based architectures for both our newly proposed VAE as well as the latent diffusion model.

E Our Approach: Additional information

E.1 Gene expression data embedding: Replacing *dropouts*

We present a method for processing sparse gene expression data that focuses computational resources on biologically relevant signals. Given a set of D genes with their corresponding expression counts, our approach addresses the inherent sparsity in single-cell RNA sequencing data, where typically 70% or more of gene-cell entries are zero.

Let $\mathcal{I} = \{1, 2, \ldots, D\}$ denote the complete set of gene IDs represented as integers, and let $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ represent the corresponding gene expression counts for a given cell, where $x_i \in \mathbb{N}_0$ is the count for gene g_i, then an n-th single cell is defined as a tuple $(\mathbf{x}_{\mathcal{I}_n}, \mathcal{I}_n)$.

Our method proceeds as follows:

1. **Context length constraint**: We define a maximum context length $d < D$ to limit the computational complexity of downstream processing.
2. **Expression-based filtering**: For each cell, we identify the set of expressed genes:
$$\mathcal{E} = \{i \in \mathcal{I} : x_i > 0\} \tag{15}$$
with corresponding expression values $\mathbf{x}_{\mathcal{E}} = \{x_i : i \in \mathcal{E}\}$.
3. **Context construction**: We construct a fixed-length input representation of dimension d. When $|\mathcal{E}| < d$ (which is typically the case due to high sparsity), we pad the input with artificial tokens to maintain consistent dimensionality:
$$\text{Input} = \begin{cases} \{(x_i, i)\}_{i \in \mathcal{E}} \cup \{(0, \text{PAD})\}^{d - |\mathcal{E}|} & \text{if } |\mathcal{E}| < d \\ \{(x_i, i)\}_{i \in \mathcal{E}} & \text{if } |\mathcal{E}| = d \end{cases} \tag{16}$$
where PAD is a special token for zero expression count.

This approach offers both computational and biological advantages. By excluding zero-expression genes (dropouts) from the input representation, we enable the model to focus exclusively on expressed genes, which carry the meaningful biological signal. The padding tokens serve purely as placeholders for implementation consistency and do not introduce spurious biological information, as they are explicitly marked with zero counts. This design choice aligns with the biological understanding that in single-cell data, the absence of detected expression often represents technical dropouts rather than meaningful biological zeros, making it advantageous to direct the model's attention solely to the detected expression events.

E.2 Conditional likelihood: The parameterization of Negative Binomial

We model the gene expression counts using a Negative Binomial distribution, which effectively captures the overdispersion commonly observed in single-cell RNA-seq data. The conditional likelihood for our model is specified as follows.

Let $\mathbf{h}(\mathbf{Z}) \in \mathbb{R}^D$ denote the output of our neural network for a given cell embeddibg $\mathbf{Z}$, where D is the number of genes. We apply a softmax transformation to obtain normalized ratios:

$$p_i(\mathbf{Z}) = \frac{\exp(\mathbf{h}_i(\mathbf{Z}))}{\sum_{j=1}^{D} \exp(\mathbf{h}_j(\mathbf{Z}))} \tag{17}$$

where $i = 1, 2, \ldots, D$, and $\sum_{i=1}^{D} p_i = 1$.

To obtain the expected expression counts, we scale these probabilities by the cell-specific library size L, namely:

$$\eta_i(\mathbf{Z}) = L \cdot p_i(\mathbf{Z}) \tag{18}$$

where μ_i represents the mean parameter for gene i in the Negative Binomial distribution.

The gene expression count x_i for gene i is then modeled as:

$$x_i \sim \text{NB}(\eta_i(\mathbf{Z}), \alpha_i) \tag{19}$$

where NB denotes the Negative Binomial distribution parameterized by mean μ_i and dispersion α_i. The probability mass function is given by:

$$p(x_i|\eta_i(\mathbf{Z}), \alpha_i) = \frac{\Gamma(x_i + \alpha_i^{-1})}{\Gamma(x_i + 1)\Gamma(\alpha_i^{-1})} \left(\frac{\alpha_i^{-1}}{\alpha_i^{-1} + \eta_i(\mathbf{Z})}\right)^{\alpha_i^{-1}} \left(\frac{\eta_i(\mathbf{Z})}{\alpha_i^{-1} + \eta_i(\mathbf{Z})}\right)^{x_i} \tag{20}$$

We consider two parameterizations for the dispersion:

1. **Shared dispersion**: A single parameter α is used for all genes, i.e., $\alpha_i = \alpha$ for all $i \in \{1, 2, \ldots, D\}$. This reduces the number of parameters and assumes homogeneous overdispersion across genes.
2. **Gene-specific dispersion**: Each gene has its own dispersion parameter, resulting in a vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_D)$. This allows for heterogeneous overdispersion patterns across genes, providing greater flexibility at the cost of additional parameters.

This formulation ensures that the predicted expression values respect the constraint that total counts sum to the observed library size, while the Negative Binomial distribution appropriately models the count nature and overdispersion of the data. The softmax transformation guarantees that the neural network learns a proper distribution over genes, making the model interpretable as learning the relative expression probabilities for each cell.

F DATASETS

General In our experiments, we used the following datasets: Dentate gyrus, Tabula Muris, Human Lung Census Atlas (HLCA), Parse1M and Replogle-Nadig; see Table 6 for details.

Experiment 1: Cell generation (benchmarks) In the cell generation experiment, we used three widely used datasets, namely, Dentate gyrus, Tabula Muris, and HLCA. Dentate gyrus is the smallest dataset (only 18k cell and 17k genes). Tabula Muris is a small dataset with over 245k cells and almost 20k genes. Human Lung Cell Atlas (HLCA) is the largest, having about 585k cells and almost 28k genes.

Experiment 2: Parse1M & Replogle In the second experiment, we used a curated subset of 10 Million Human PBMC dataset. We carried out experiments on 2k highly variable genes (HVGs). In this data, we focused on a single donor who had 18 cell types undergone 90 cytokine perturbations as well as a control treatment.

Next, we used the well-known Replogle-Nadig dataset which consists of four cell lines and 2024 gene-edits. We carried out experiments on 2k highly variable genes (HVGs). All 2024 gene-edits in three cell-lines ('jurkat', 'k562', 'rpe1'), along with a subset of edits from the 'hepg2' cell line were used for taining. The remaining 'hepg2' gene-edits were held out for testing.

Experiment 3: COVID-19 and Tabula Sapiens 2.0 In the fourth experiment, we used two datasets for embedding evaluation. First, we used the scRNA-seq experimental dataset of four healthy donors' lung sections infected with SARS-CoV-2 (Wu et al., 2024). Data were downloaded from CZ CELLxGENE[6]. Second, we used the Tabula Sapiens 2.0 dataset (Consortium & Quake, 2025), a comprehensive single-cell atlas of human tissues. We focused on 6 tissues: blood, spleen, lymph node, small intestine, thymus, and liver. We filtered out cell types with fewer than 100 cells to ensure robust classification performance and used the resulting filtered dataset for multinomial logistic regression-based cell type prediction tasks.

[6] https://cellxgene.cziscience.com/collections/2a9a17c9-1f61-4877-b384-b8cd5ffa4085

Table 6: Summary of datasets used in the experiments.

Experiment	Dataset name	No. of cells	No. of genes	No. of cell types/lines
1	Dentate gyrus	18,213	17,002	14 cell types
1	Tabula Muris	245,389	19,734	123 cell types
1	HLCA	584,944	27,997	50 cell types
2	Parse1M	1,267,690	2,000 (HVGs)	18 cell types
2	Replogle-Nadig	624,158	2,000 (HVGs)	4 cell lines
3	COVID-19	354,026	27,998	55 cell types
3	Tabula Sapiens 2.0	1,482,026	$\sim$ 25k	22 cell types

G BASELINES

scVI Single-cell Variational Inference (scVI) (Lopez et al., 2018) is VAE-based generative models designed for single-cell discrete data. Following the standard VAE framework, this model learns a Gaussian latent space that is subsequently decoded into the parameters of a discrete conditional likelihood model.

scDiffusion A version of a latent diffusion model for single-cell gene expression data is scDiffusion (Luo et al., 2024). The scDiffusion model consists of three modules. The first module is an auto-encoderthat transforms gene expression patterns into a compact representation space, allowing dimensionality reduction and identification of complex cellular measurements. In the latent space, a denoising network is trained to reverse a diffusion process applied to the latent embeddings, turning noise into meaningful biological signal encoded in the latent space. To ensure guided generation, a third model is trained, a classifier, for incorporating cell type or other biological attributes.

CFGen CFGen is a current state-of-the-art latent diffusion model that builds upon scVI, training a latent flow matching model in the VAE's latent space (Palma et al., 2025a). Similar to our approach, CFGen employs a two-stage training strategy: first training the autoencoder, then training the flow matching model on the VAE-generated embeddings. While CFGen introduces additive steering through classifier-free guidance, we utilize joint attribute control (see Table 14). Additionally, CFGen models library size within the diffusion framework and samples from the mean and standard deviation of the library size distribution for conditional generation. We adapted this approach for sampling library size in our Negative Binomial conditional likelihood; however, unlike CFGen, we do not condition our Diffusion Transformer model on library size.

CPA Compositional Perturbation Autoencoder (CPA) (Lotfollahi et al., 2023b) is a deep generative model developed to predict gene expression changes under perturbations and their combinations. CPA disentangles latent representations of basal cellular state, perturbation effects, and additional covariates such as cell type. By recombining these factors through its decoder, CPA can reconstruct observed expression profiles and generalize to unseen perturbation–covariate combinations. This compositional structure enables CPA to extrapolate beyond training data, making it particularly well-suited for evaluating out-of-distribution generalization in perturbational single-cell datasets.

H IMPLEMENTATION DETAILS

In this paper, we carried out model selection for various values of hyperparameters. In the following paragraphs, we provide further details for reproducibility.

VAE Table 7 summarizes the hyperparameter configurations used for the VAE encoder architectures in our experiments.

Table 8 summarizes the hyperparameter configurations used for the VAE decoder architectures in our experiments.

Flow Matching Table 9 summarizes the hyperparameter configurations used for the LDM architectures in our experiments.

Table 7: Hyperparameter values of VAE Encoders considered in this paper.

VAE Encoder	
Embedding layer	
Embedding size	256
Cross-Attention Block	
Number of Heads	1
No. pseudoinputs	{64, 256}
Embedding size	256
Transformer Blocks	
Number of Blocks	{2, 4}
Number of Heads	1
Embedding size	256
Gaussian Stochastic Layer	
Latents per token	{8, 16}

Table 8: Hyperparameter values of VAE Decoders considered in this paper.

VAE Decoder	
Transformer Blocks	
Number of Blocks	4
Number of Heads	8
Embedding size	256
Normalization	`LayerNorm`
Cross-Attention Block	
Shared embedding layer	`True`
Number of Heads	1
Embedding size	256
NegativeBinomial Stochastic Layer	
Shared θ	`False`

Table 9: Hyperparameter values of LDMs considered in this paper.

LDM – Flow Matching	
Denoising Transformer	
Number of Blocks	8
Number of Heads	8
Embedding size	256
Normalization	`LayerNorm`
Adaptive Normalization	`True`
Hyperparams	
σ	$1e^{-4}$
v	0
Transport	`linear`

Training details For training, we swept over various configurations of hyperparameters, see Table 10.

Table 10: Hyperparameter values of training procedures considered in this paper.

Training	
KL-weight	$\{0, 1e^{-5}\}$
Optimizer	`AdamW`
Mini-batch size	256
Learning rate	$1e^{-3}$
(β_1, β_2)	$(0.9, 0.95)$
Weight Decay	$1e^{-7}$
Learning scheduler	`cosine`

I Evaluation

I.1 Maximum Mean Discrepancy (MMD)

We propose to use the Maximum Mean Discrepancy (MMD) (Gretton et al., 2012). MMD is a non-parametric distance measure between probability distributions based on the notion of embedding distributions into a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. Given two distributions P and Q over a domain $\mathcal{X}$, the MMD is defined as:

$$\mathrm{MMD}[\mathcal{F}, P, Q] = \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{y \sim Q}[f(y)] \right), \tag{21}$$

where $\mathcal{F}$ is a class of functions. When $\mathcal{F}$ is the unit ball in an RKHS $\mathcal{H}$ with kernel k, the MMD can be expressed as:

$$\mathrm{MMD}^2[\mathcal{H}, P, Q] = \mathbb{E}_{x,x' \sim P}[k(x, x')] + \mathbb{E}_{y,y' \sim Q}[k(y, y')] - 2\mathbb{E}_{x \sim P, y \sim Q}[k(x, y)]. \tag{22}$$

In practice, given finite samples $X = \{x_1, \ldots, x_m\}$ drawn from P and $Y = \{y_1, \ldots, y_n\}$ drawn from Q, we use the unbiased empirical estimate:

$$\widehat{\mathrm{MMD}}^2[X, Y] = \frac{1}{m(m-1)} \sum_{i \neq j}^{m} k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i \neq j}^{n} k(y_i, y_j) - \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k(x_i, y_j). \tag{23}$$

The choice of kernel k determines the richness of the function class $\mathcal{F}$. Common choices include the Gaussian RBF kernel $k(x, y) = \exp(-\|x - y\|^2 / 2\sigma^2)$ with bandwidth parameter σ. The MMD is zero if and only if $P = Q$ when using a characteristic kernel, making it a powerful tool for two-sample testing and distribution matching applications.

I.2 2-Wasserstein Distance (W2)

The 2-Wasserstein distance provides an alternative metric for comparing probability distributions based on optimal transport theory. For distributions P and Q on $\mathbb{R}^d$, the 2-Wasserstein distance is defined as:

$$W_2(P, Q) = \left(\inf_{\gamma \in \Gamma(P,Q)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|\mathbf{x} - \mathbf{y}\|^2 \, d\gamma(\mathbf{x}, \mathbf{y}) \right)^{1/2}, \tag{24}$$

where $\Gamma(P, Q)$ denotes the set of all joint distributions with marginals P and Q. For empirical distributions with equal sample sizes n, given samples $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$, the discrete 2-Wasserstein distance simplifies to:

$$W_2^2(X,Y) = \frac{1}{n}\min_{\pi\in\Pi_n}\sum_{i=1}^{n}\|\mathbf{x}_i - \mathbf{y}_{\pi(i)}\|^2, \tag{25}$$

where Π_n is the set of all permutations of $\{1,\ldots,n\}$. This optimization problem can be solved efficiently using the Hungarian algorithm or entropic regularization approaches.

When both distributions are Gaussian with means $\boldsymbol{\mu}_P, \boldsymbol{\mu}_Q$ and covariances $\boldsymbol{\Sigma}_P, \boldsymbol{\Sigma}_Q$, the 2-Wasserstein distance has a closed-form expression:

$$W_2^2(P,Q) = \|\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q\|^2 + \text{tr}\left(\boldsymbol{\Sigma}_P + \boldsymbol{\Sigma}_Q - 2(\boldsymbol{\Sigma}_P^{1/2}\boldsymbol{\Sigma}_Q\boldsymbol{\Sigma}_P^{1/2})^{1/2}\right), \tag{26}$$

which coincides with the Frechét Distance.

Unlike MMD, the Wasserstein distance directly captures the geometry of the underlying space and provides interpretable transport plans between distributions.

I.3 Fréchet Distance for Gene Expression Profile Evaluation

We adapt the Fréchet Inception Distance (FID) framework to evaluate the quality of synthetic gene expression profiles by replacing the Inception network's feature extraction with Principal Component Analysis (PCA). This approach provides a computationally efficient and interpretable metric for comparing distributions of real and synthetic gene expression data.

I.3.1 Principal Components Calculation

Let $\mathcal{X}_r = \{\mathbf{x}_1^r, \mathbf{x}_2^r, \ldots, \mathbf{x}_n^r\}$ denote the set of real gene expression profiles and $\mathcal{X}_s = (\mathbf{x}_1^s, \mathbf{x}_2^s, \ldots, \mathbf{x}_m^s\}$ denote the synthetic profiles, where each $\mathbf{x}_i \in \mathbb{R}^D$ represents the expression levels of D genes.

We first apply PCA to the combined dataset to obtain a projection matrix $\mathbf{W} \in \mathbb{R}^{k\times D}$ containing the top k principal components (e.g., $k = 30$). The feature representations are computed as:

$$\mathbf{z}_i^r = \mathbf{W}^\top \mathbf{x}_i^r, \quad \mathbf{z}_j^s = \mathbf{W}^\top \mathbf{x}_j^s \tag{27}$$

where $\mathbf{z}_i^r, \mathbf{z}_j^s \in \mathbb{R}^k$ are the projected representations in the principal component space.

I.3.2 Fréchet Distance Calculation

Assuming the feature representations follow multivariate Gaussian distributions:

- Real data: $\mathcal{N}(\mu_r, \Sigma_r)$;
- Synthetic data: $\mathcal{N}(\mu_s, \Sigma_s)$.

We estimate the parameters:

$$\mu_r = \frac{1}{n}\sum_{i=1}^{n}\mathbf{z}_i^r, \quad \Sigma_r = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{z}_i^r - \mu_r)(\mathbf{z}_i^r - \mu_r)^\top \tag{28}$$

$$\mu_s = \frac{1}{m}\sum_{j=1}^{m}\mathbf{z}_j^s, \quad \Sigma_s = \frac{1}{m-1}\sum_{j=1}^{m}(\mathbf{z}_j^s - \mu_s)(\mathbf{z}_j^s - \mu_s)^\top \tag{29}$$

The Fréchet Distance between these distributions is then computed as:

$$\text{FD} = ||\mu_r - \mu_s||_2^2 + \text{Tr}(\Sigma_r + \Sigma_s - 2(\Sigma_r\Sigma_s)^{1/2}) \tag{30}$$

where $\text{Tr}(\cdot)$ denotes the matrix trace and $(\Sigma_r\Sigma_s)^{1/2}$ is the matrix square root of $\Sigma_r\Sigma_s$.

I.3.3 INTERPRETATION

This metric captures both the difference in means (first term) and the difference in covariance structure (second term) between real and synthetic gene expression profiles in the reduced PCA space. Lower values indicate better agreement between the distributions, suggesting higher quality synthetic data. The use of PCA ensures that the comparison focuses on the most significant sources of variation in the gene expression data while reducing computational complexity from $O(d^2)$ to $O(k^2)$ where typically $k \ll d$.

I.4 PEARSON CORRELATION COEFFICIENT (PCC)

While MMD and Wasserstein distances measure distributional differences, the Pearson correlation coefficient quantifies linear relationships between paired observations. For two random variables X and Y, the population Pearson correlation coefficient is defined as:

$$\rho_{X,Y} = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{\mathbb{E}[(X-\mu_X)^2]}\sqrt{\mathbb{E}[(Y-\mu_Y)^2]}}, \tag{31}$$

where μ_X, μ_Y are the means and σ_X, σ_Y are the standard deviations. Given paired samples $\{(x_i, y_i)\}_{i=1}^n$, the sample correlation coefficient is:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}, \tag{32}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ and $\bar{y} = \frac{1}{n}\sum_{i=1}^n y_i$. The coefficient $r \in [-1, 1]$, with $|r| = 1$ indicating perfect linear relationship and $r = 0$ suggesting no linear correlation. For multivariate data $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{Y} \in \mathbb{R}^{n \times d}$, one can compute the average correlation across dimensions or construct a correlation matrix. While Pearson correlation captures only linear dependencies and is sensitive to outliers, it remains widely used due to its computational efficiency and interpretability in assessing feature-wise relationships between datasets.

I.5 METRICS USED IN EXPERIMENTS

Reconstruction Metrics In our experiments, we use the reconstruction error for the Negative Binomial distribution, PCC and Mean Squared Errors (MSE) as reconstruction metrics.

Generation Metrics For evaluating generation capabilities of models, we use the MMD with the RBF kernel, the Wasserstein Distance, and the Frechet Distance, all calculated to 30 principal components. We compute the PCA on the true data, and project generated data using the loadings. All evaluations were run using 3 generation seeds.

J ABLATION ON TYPE OF CLASSIFIER-FREE GUIDANCE

Classifier-Free Guidance with Multiple Conditioning Variables. In our setting, as described in section 4.2 the diffusion model is conditioned on multiple attributes simultaneously (e.g., cell type and perturbation). We explore two alternative strategies for applying classifier-free guidance (CFG):

(Type I: Joint conditioning). A single conditioning token is assigned to each unique combination of attributes. The model output under this strategy is given by

$$\tilde{v}_{t,\epsilon}(\mathbf{Z}, y) = v_{t,\epsilon}(\mathbf{Z}; \text{Null}) + \omega\left[v_{t,\epsilon}(\mathbf{Z}; y) - v_{t,\epsilon}(\mathbf{Z}; \text{Null})\right], \tag{33}$$

where y encodes the full joint condition (e.g., "CD4 Naive + IL-9" or "HepG2 + PPP6C").

(Type II: Additive conditioning). Instead of encoding combinations directly, we treat each conditioning variable independently. For M attributes with labels $\{y^{(j)}\}_{j=1}^M$, the guided output is

$$\tilde{v}_{t,\epsilon}\Big(\mathbf{Z}, \{y^{(j)}\}_{j=1}^M\Big) = v_{t,\epsilon}(\mathbf{Z}; \text{Null}) + \sum_{j=1}^M \omega_j \Big[v_{t,\epsilon}(\mathbf{Z}; y^{(j)}) - v_{t,\epsilon}(\mathbf{Z}; \text{Null})\Big], \tag{34}$$

where each attribute contributes an additive adjustment relative to the unconditional prediction.

Empirical Comparison. We evaluate both approaches on **Parse 1M** (conditioning on cell type + cytokine perturbation) and **Replogle** (conditioning on cell type + gene knockout). As shown in Table 14, the joint conditioning strategy (Type I) consistently outperforms the additive conditioning strategy (Type II) across metrics, indicating that learning a single joint embedding for each combination of attributes is more effective in capturing complex context–perturbation interactions than treating them independently.

In all experiments, we set guidance weight ω to 1, and we did not tune this parameter.

K Additional Results

K.1 Experiment 1

In Table 11 we report an ablation study on encoding gene expression following our approach of zero padding the non-expressed genes (see Appendix E.1), or utilizing the full context as input, as typically done in scVI. Our approach is superior in terms of reconstruction performance, as well as more computationally efficient.

Table 11: Reconstruction performance comparison of our scLDM using the zero padding strategy for encoding, or using all genes as input.

Dataset	Model	RE ↓	PCC ↑	MSE ↓
Dentate Gyrus	full context	5458.6	0.097	0.252
	zero padding	**5325.3**	**0.125**	**0.242**

In Table 12, we present the model comparison on benchmark datasets, but on all genes. Please note that in this comparison, scdiffusion performs better, and it is drastically better than on the highly variable genes. The reason for that is that the data is extremely sparse, and the model synthesizes data consisting mostly of zeros. As a result, the match becomes better. This is a clear indication of deficiencies of the currently used evaluation metrics for generative models, which is a long-standing issue in the field (Theis et al., 2016).

In Figure 4, 5 and 6, we present a visualization of gene-wise variance for true and generated data on Dentate Gyrus, Tabula Muris, and HLCA, respectively. CFGen and scLDM properly recover true variance, with a slight tendency of scLDM to overestimate, while scDiffusion completely fails and underestimates the true Variance.

In Figure 7 we present a visualization of true and generated data for all models and all datasets in UMAP coordinates. In Figure **??** we present the conditional generation results in UMAP coordinates colored by the conditional class employed.

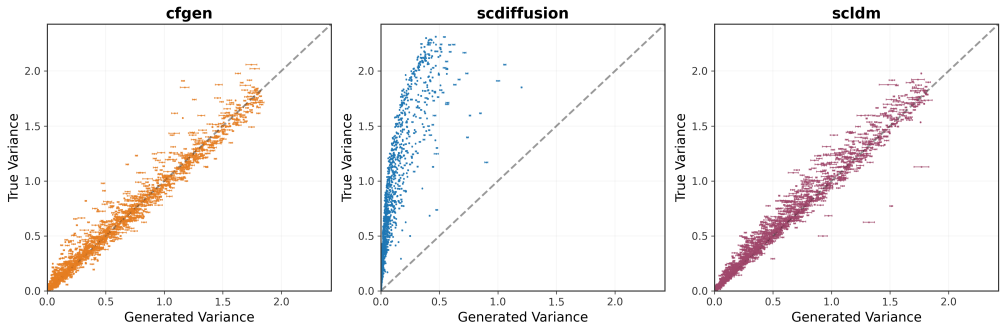

Figure 4: Visualization of the gene-wise variance for true and generated data for CFGen (left), scDiffusion (middle) our model (right), for the conditional generation settings on Dentate Gyrus. The error bars represent the standard errors over 3 seeds.

Table 12: Model performance comparison on unconditional and conditional cell generation benchmarks on all genes. W2, MMD^2 RBF and FD metrics calculated on 30 principal components are reported (lower is better).

Setting	Model	W2 ↓	MMD^2 RBF ↓	FD ↓
		Dentate Gyrus		
Uncond	scDiffusion	8.545 ± 0.061	0.021 ± 0.000	16.303 ± 0.265
	CFGen	11.396 ± 0.025	0.027 ± 0.001	24.942 ± 0.456
	Ours	9.489 ± 0.054	0.027 ± 0.000	28.307 ± 0.357
Cond	scDiffusion	8.906 ± 0.041	0.093 ± 0.001	28.829 ± 1.612
	CFGen	10.580 ± 0.022	0.082 ± 0.001	40.298 ± 0.472
	Ours	9.147 ± 0.024	0.108 ± 0.004	31.045 ± 0.884
		Tabula Muris		
Uncond	scDiffusion	8.616 ± 0.215	0.002 ± 0.000	6.881 ± 0.565
	CFGen	11.331 ± 0.081	0.009 ± 0.000	31.788 ± 1.073
	Ours	10.573 ± 0.092	0.005 ± 0.000	17.641 ± 0.337
Cond	scDiffusion	11.459 ± 0.081	0.035 ± 0.001	43.456 ± 1.678
	CFGen	9.420 ± 0.041	0.026 ± 0.001	22.045 ± 0.389
	Ours	8.530 ± 0.110	0.019 ± 0.001	15.547 ± 0.557
		HLCA		
Uncond	scDiffusion	9.234 ± 0.008	0.002 ± 0.000	5.585 ± 0.180
	CFGen	12.651 ± 0.025	0.008 ± 0.000	24.038 ± 0.492
	Ours	10.816 ± 0.089	0.010 ± 0.000	24.126 ± 0.473
Cond	scDiffusion	9.998 ± 0.048	0.094 ± 0.002	40.093 ± 3.103
	CFGen	10.715 ± 0.039	0.087 ± 0.005	36.178 ± 0.961
	Ours	9.350 ± 0.046	0.084 ± 0.005	28.398 ± 1.358

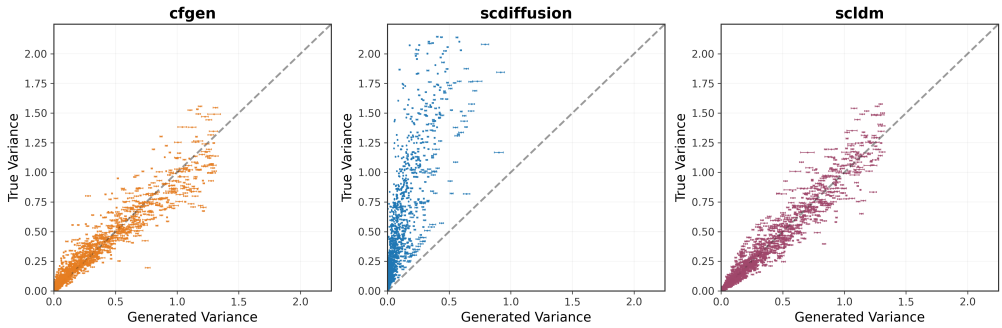

Figure 5: Visualization of the gene-wise variance for true and generated data for CFGen (left), scDiffusion (middle) our model (right), for the conditional generation settings on Tabula Muris. The error bars represent the standard errors over 3 seeds.

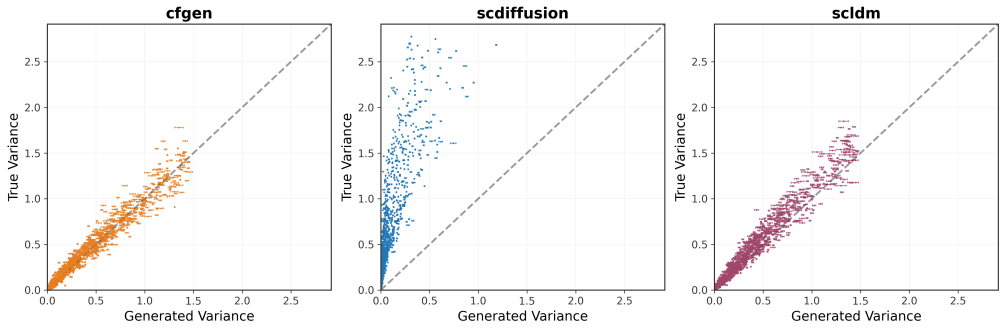

Figure 6: Visualization of the gene-wise variance for true and generated data for CFGen (left), scDiffusion (middle) our model (right), for the conditional generation settings on HLCA. The error bars represent the standard errors over 3 seeds.

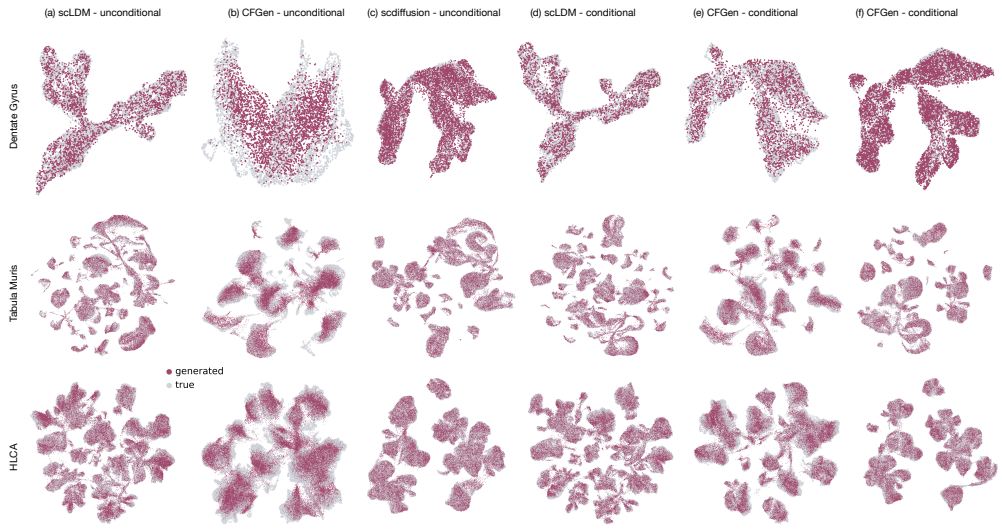

Figure 7: Visualization of the generation results for all datasets and models for conditional and unconditional generations. True and Generated gene expression is embedded in UMAP coordinates jointly, upon normalization, following standard Scanpy pipeline.

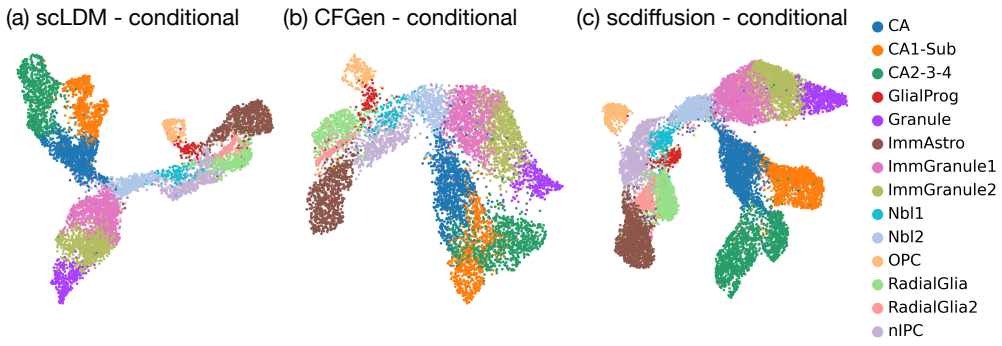

Figure 8: Visualization of the conditional generation results for the dentate gyrus dataset and all models, colored by the conditional label (clusters).

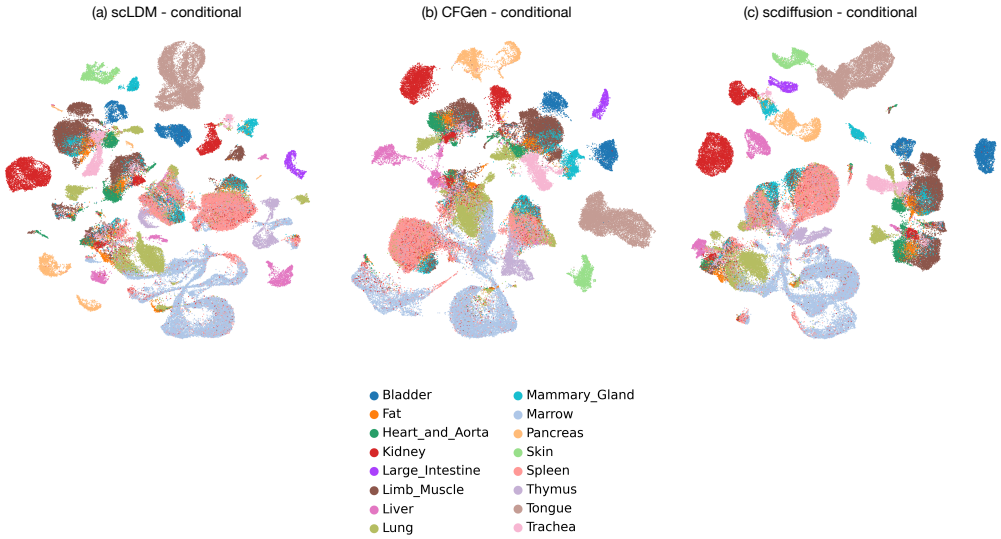

Figure 9: Visualization of the conditional generation results for the tabula muris dataset and all models, colored by the conditional label (tissue).

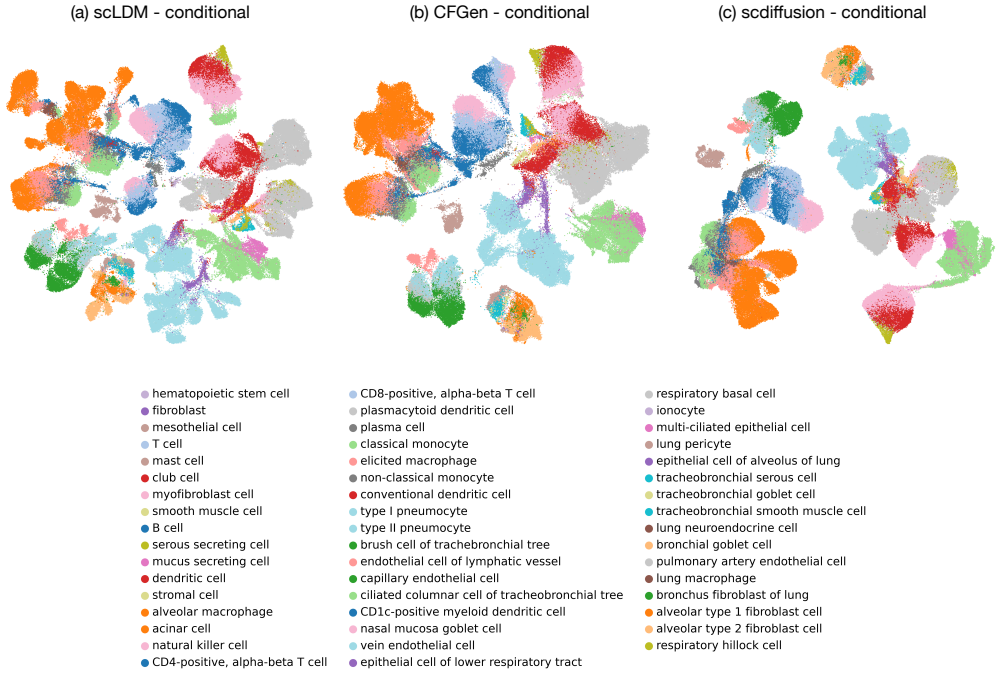

Figure 10: Visualization of the conditional generation results for the hlca dataset and all models, colored by the conditional label (cell type).

K.2 Experiment 2: Reconstruction capabilities

The results presented in Table 13 demonstrate that our proposed approach significantly outperforms the scVI baseline across all evaluated metrics on the Parse1M dataset. Most notably, our method achieves a substantially lower reconstruction error (RE) of about 310 nats compared to scVI's 432 nats, indicating better reconstructive capabilities. Furthermore, our approach yields a remarkable improvement in Pearson correlation coefficient (PCC), achieving 0.887 versus scVI's mediocre 0.351, which suggests that our model captures the underlying biological relationships much more effectively. The mean squared error (MSE) is also greatly reduced from 0.701 to 0.188, representing an approximately 73% reduction in reconstruction error. These consistent improvements across multiple evaluation criteria provide strong evidence that our method offers substantial advantages over scVI and indicates its great potential in analyzing biological data.

Table 13: Model performance comparison on cell reconstruction task.

Dataset	Model	RE ↓	PCC ↑	MSE ↓
Parse 1M	scVI	432.41 ± 0.08	0.351 ± 0.000	0.701 ± 0.001
	scLDM	$\mathbf{149.70 \pm 0.22}$	$\mathbf{0.874 \pm 0.003}$	$\mathbf{0.165 \pm 0.002}$
Replogle	scVI	2144.86 ± 0.35	0.166 ± 0.000	0.703 ± 0.001
	scLDM	$\mathbf{1590.51 \pm 0.38}$	$\mathbf{0.713 \pm 0.004}$	$\mathbf{0.285 \pm 0.002}$

K.3 Experiment 2: A comparison between *additive* and *joint* conditioning in classifier-free guidance

Table 14 compares the performance of our scLDM model using two different classifier-free guidance approaches for conditional cell generation: the additive steering method proposed by Palma et al. (2025a) and our joint attribute control method. Across all metrics (Wasserstein-2 distance, MMD^2 RBF, and Fréchet Distance) and both datasets (Parse 1M and Replogle), the joint approach consistently outperforms the additive approach, demonstrating substantial improvements in generation quality.

Table 14: Model performance comparison on conditional cell generation on Parse1M and Replogle. For these results scLDM was trained using the classifier-free guidance approach proposed in Palma et al. (2025a) (additive) and ours (joint).

Dataset	Model	W2 ↓	MMD^2 RBF ↓	FD ↓
Parse 1M	scLDM (additive)	15.850 ± 0.073	0.129 ± 0.004	109.196 ± 2.933
	scLDM (joint)	$\mathbf{12.455} \pm 0.001$	$\mathbf{0.027} \pm 0.000$	$\mathbf{18.145} \pm 0.068$
Replogle	scLDM (additive)	18.538 ± 0.058	0.451 ± 0.003	255.510 ± 2.163
	scLDM (joint)	$\mathbf{11.288} \pm 0.011$	$\mathbf{0.200} \pm 0.001$	$\mathbf{53.555} \pm 0.210$

K.4 Experiment 3

For the last experiment, trained three VAEs for our approach (scLDM-VAE): with 20M parameters, 70M parameters, and 270M parameters. Further, we evaluated the resulting models using embeddings on a downstream task (classification) for two out-of-distribution datasets (COVID-19 and Tabula Sapiens 2.0).

First, we evaluated these three versions of our model using reconstruction metrics on the dataset they were trained on, namely, Human Census Data from CellxGene[7]. Looking at Table 15, we can see a clear relationship between model size and reconstruction performance for the scLDM-VAE models on the CellxGene dataset. As the number of parameters increases from 20M to 270M, all three metrics show substantial improvement: reconstruction error (RE) decreases, Pearson correlation coefficient (PCC) increases from, and mean squared error (MSE) drops. These results demonstrate that scaling up the scLDM-VAE architecture yields consistent performance gains across all reconstruction metrics, with the 270M parameter model achieving approximately 17% lower reconstruction error and 18% higher correlation compared to the smallest 20M model.

[7] https://cellxgene.cziscience.com/

Table 15: Reconstruction performance comparison of our scLDM-VAEs with varying number of parameters: 20M, 70M, and 270M.

Dataset	Model	RE ↓	PCC ↑	MSE ↓
CellxGene Census	scLDM-VAE (20M)	1742.7	0.661	0.137
	scLDM-VAE (70M)	1552.7	0.739	0.106
	scLDM-VAE (270M)	**1441.7**	**0.783**	**0.091**

Table 16 presents a comprehensive performance comparison of various models on the COVID-19 dataset, averaged across all donors. Our scLDM model with 270M parameters achieves the best performance across all metrics (ROC AUC, PR AUC, F1 Score, Recall, and Precision), demonstrating consistent improvements over both transformer-based baselines (TranscriptFormer, scGPT, Geneformer, UCE) and traditional VAE approaches (scVI, AIDO.Cell).

Table 16: COVID-19 Model Performance Summary (Averaged Across All Donors). **Bold** indicates the best performing model.

Model	ROC AUC	PR AUC	F1 Score	Recall	Precision
scLDM (270M)	**0.909**± 6e-04	**0.877**± 0.001	**0.820**± 0.001	**0.836**± 0.001	**0.806**± 0.001
TranscriptFormer	0.905± 4e-04	0.874± 9e-04	0.814± 0.002	0.829± 0.003	0.801± 0.001
scLDM (70M)	0.905± 5e-04	0.872 ± 0.001	0.815 ± 0.001	0.83 ± 0.002	0.801 ± 0.001
scLDM (20M)	0.902± 5e-04	0.869 ± 0.001	0.811 ± 0.001	0.827 ± 0.001	0.797 ± 0.002
UCE	0.876± 5e-04	0.834± 0.002	0.775± 8e-04	0.781± 0.001	0.771± 0.002
scGPT	0.876± 4e-04	0.831± 0.001	0.779± 9e-04	0.793± 0.002	0.766± 0.001
Geneformer	0.866± 6e-04	0.815± 0.001	0.768± 0.001	0.781± 0.003	0.757± 0.001
AIDO.Cell	0.821± 7e-04	0.753± 9e-04	0.717± 8e-04	0.729± 0.002	0.708± 0.001
scVI	0.800± 7e-04	0.709± 0.001	0.675± 0.001	0.680± 0.002	0.680± 0.001

Figure 11 visualizes the receiver operating characteristic (ROC) and precision-recall (PR) curves for all models on the COVID-19 classification task. The curves further illustrate the superior discriminative performance of scLDM variants, with the 270M parameter model achieving the highest area under both curves, consistent with the quantitative results in Table 16.

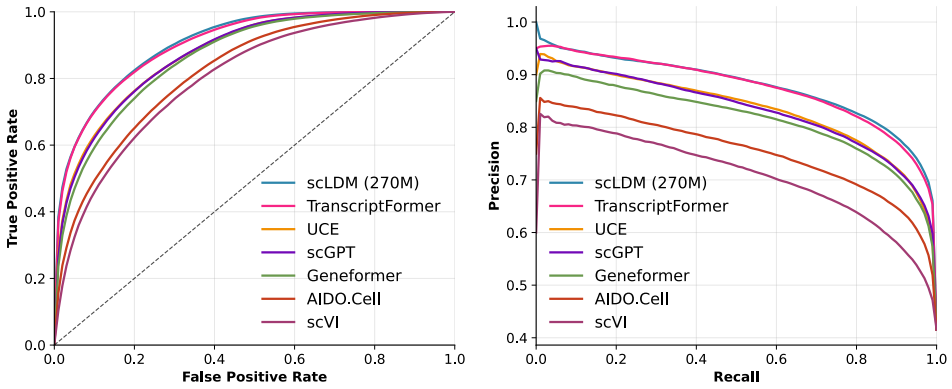

Figure 11: Precision-recall and reciver operator curves for COVID-19 data.

Table 17 summarizes model performance on the Tabula Sapiens 2.0 dataset, averaged across all tissues. Notably, the smallest scLDM variant (20M parameters) achieves the highest F1 score (0.804), slightly outperforming both larger scLDM models and all baseline methods, suggesting that model scale may have diminishing returns on this particular cell type classification task.

Table 17: Tabula Sapiens 2.0 model performance summary (averaged across all tissues)

Model	F1 Score	Recall	Precision
scLDM-20M	**0.804** ± 0.002	**0.805** ± 0.002	**0.812** ± 0.002
scLDM-270M	0.802 ± 0.002	0.803 ± 0.002	0.811 ± 0.002
scLDM-70M	0.802 ± 0.002	0.802 ± 0.002	0.810 ± 0.002
scGPT	0.800 ± 0.002	0.802 ± 0.002	0.806 ± 0.002
scVI	0.799 ± 0.002	0.794 ± 0.002	0.814 ± 0.003
TranscriptFormer	0.799 ± 0.002	0.800 ± 0.002	0.802 ± 0.002
UCE	0.796 ± 0.002	0.797 ± 0.001	0.801 ± 0.003
Geneformer	0.777 ± 0.002	0.776 ± 0.002	0.786 ± 0.003
AIDO.Cell	0.724 ± 0.002	0.715 ± 0.002	0.748 ± 0.003