CELLUMM: A UNIFIED MULTIMODAL MODEL FOR CELLULAR SEMANTIC UNDERSTANDING AND QUANTITATIVE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Single-cell measurements describe cell state as high-dimensional quantitative expression, whereas biological knowledge is organized as semantics such as gene function, cell type, and experimental condition. Prior work optimizes these two views separately within a specific modality and task space, so semantic understanding, quantitative generation, and state transition have not been placed in one framework. We present CellUMM, a unified cell–language multimodal model that casts cellular modeling as four conditional mappings between a semantic space and a quantitative cell-state space. CellUMM connects biological semantics with cellular expression through a shared semantic–cellular representation space and continuous latent cell programs, and it performs conditional generation and perturbation state transition by latent flow matching in the continuous cell-state space. Joint training on gene function annotation, cell type annotation, conditional cell generation, and perturbation response prediction makes the learned semantic representation both describe cell states and drive executable cell-distribution generation and perturbation response prediction. On the benchmarks, CellUMM outperforms task-specific baselines, leading on the majority of metrics in four tasks.

1 INTRODUCTION

Single-cell transcriptomic measurements characterize cell state as a high-dimensional expression profile, which is a quantitative basis for studying cellular heterogeneity, disease mechanisms, and intervention responses (Regev et al., 2017; Lähnemann et al., 2020; Tabula Sapiens Consortium, 2022; Srivatsan et al., 2020). Biological knowledge such as cell type, gene function, disease state, tissue context, and experimental condition is organized instead as text, labels, and structured semantics. Together they form a semantic space and a quantitative cell-state space for cellular modeling.

Real biological questions require inference in both directions between these two spaces; a generic cell embedding is not sufficient. We group the inferences into four conditional mappings: semantics to structured semantics (gene function annotation) (Ashburner et al., 2000; Aleksander et al., 2025), quantitative to semantics (cell type annotation) (Aran et al., 2019; Domínguez Conde et al., 2022), semantics to quantitative (conditional cell generation) (Lopez et al., 2018; Palma et al., 2025), and conditional quantitative to quantitative (perturbation response prediction) (Lotfollahi et al., 2019; Roohani et al., 2024; Bunne et al., 2023; Norman et al., 2019). The mappings make coupled demands: semantic tasks must stay interpretable, quantitative tasks must preserve the high-dimensional expression structure and its distributional statistics, conditional generation must ground semantic conditions in samplable cell states, and perturbation modeling must learn the distributional shift from control to intervention beyond the mean change alone. A common constraint follows on the generation side: outputs must be native expression matrices, with cells as rows and genes as columns, so that samples can be evaluated directly at the distribution level (Gretton et al., 2012). Text, discrete tokens, or a single embedding can only serve as intermediate representations (Jaegle et al., 2022; Zhou et al., 2025).

Existing work advances these directions separately. Biomedical text models provide knowledge-level semantic representations (Lee et al., 2020). Cell–language models connect expression profiles to language models for annotation, retrieval, and instruction-driven pseudo-cell generation (Fang

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

et al., 2025; Schaefer et al., 2025; Rizvi et al., 2025; Xiao et al., 2025). Single-cell foundation models learn transferable cell-state representations through large-scale self-supervised pretraining (Theodoris et al., 2023; Cui et al., 2024; Hao et al., 2024; Rosen et al., 2026). Probabilistic and flow-matching models characterize expression distributions and the transition from control to intervention (Palma et al., 2025; Klein et al., 2025; Li et al., 2026; Lopez et al., 2018; Adduri et al., 2025). Most of these methods, however, are optimized around a single modality, a single condition type, and a single output space, and do not couple semantics with a samplable expression distribution.

This separation has concrete consequences. Supervision that lies on labels or text contains no expression likelihood. Semantic models return category-level judgments and cannot retain graded responses or heterogeneous state spectra. Fixed-dimensional embeddings offer no samplable distribution interface. As a result, distribution-level counterfactual prediction requires a separately trained decoder. Generative models conditioned on structured variables such as batch, cell type, drug, or gene perturbation cannot accept open-vocabulary natural-language conditions and usually do not perform the two annotation tasks. Nothing in their condition space corresponds to a semantic representation. Semantic understanding and quantitative modeling have therefore developed along two independent tracks that transfer poorly to each other, which raises the question of whether one model can connect semantic understanding, cell-state interpretation, native cell-distribution generation, and perturbation state transition.

We present CellUMM, a unified cell–language multimodal model that casts cellular modeling as the four conditional mappings above. CellUMM connects biological semantics with expression states through a shared semantic–cellular representation space and continuous latent cell programs: a semantic decoder performs gene function annotation and cell type annotation, while latent flow matching performs conditional cell generation and perturbation state transition and directly emits native expression samples that can be evaluated at the distribution level. The four tasks form a mutually reinforcing training process: the annotation tasks force latent cell programs to be semantically interpretable, conditional generation requires semantic conditions to be realizable as real cell distributions, and perturbation response prediction requires the quantitative transition from a control state to an intervened state.

The main contributions are as follows.

- **A unified cell–language multimodal architecture.** A shared semantic–cellular representation space and continuous latent cell programs connect biological semantics with quantitative expression states, and latent flow matching models native cell distributions and state transitions.
- **A single model that both interprets cell states and generates their distributions.** Joint training on the four tasks gives one shared parameter set cell-state interpretation, conditional generation, and perturbation response prediction.
- **A unified formulation of four cellular inference tasks.** The four tasks are formalized as conditional mappings between the semantic and quantitative spaces, and one model is compared with task-specific baselines on all four.

2 Related Work

2.1 Semantic Models for Gene- and Cell-Level Knowledge

Biomedical text models provide mature representations of gene-level knowledge: BioBERT continues domain-adaptive pretraining on PubMed and PMC, and PubMedBERT is pretrained with a domain vocabulary, both outperforming domain-fine-tuned general models on entity recognition, relation extraction, and question answering (Lee et al., 2020; Gu et al., 2021). Cell–language models extend this interface to expression data, with Cell2Sentence writing an expression profile as a gene-token sequence for a general-purpose language model (Levine et al., 2023), CellWhisperer learning a joint transcriptome–text embedding for retrieval and feature annotation (Schaefer et al., 2025), CellTok adding discrete cell tokens to the vocabulary (Xiao et al., 2025), LangCell aligning expression with text for cell annotation (Zhao et al., 2024), InstructCell performing annotation and pseudo-cell generation from instructions (Fang et al., 2025), and CellHermes harmonizing multimodal omics data (Gao et al., 2025). Reference-based annotators instead label cells by similarity to annotated references (Aran et al., 2019; Domínguez Conde et al., 2022). Their inputs and outputs, however, remain in text or discrete-token space, so they neither emit native cell distributions

Table 1: Formal definitions of the four tasks. x is expression, c is a condition, g is gene text, C is a candidate set, y is a semantic label, and x_0 and x_1 are the control and perturbed states.

Task	Conditional prediction	Output object
Gene function annotation	$p_\theta(y \mid g, C, c)$	Candidate GO association vector, or a single association probability
Conditional cell generation	$p_\theta(x \mid c)$	Expression distribution given organ and cell type
Perturbation response prediction	$p_\theta(x_1 \mid x_0, c_p)$	Treated expression distribution given control and intervention
Cell type annotation	$p_\theta(y \mid x, C, c)$	Answer probability over candidate type names

nor jointly constrain the four tasks of this paper. Per-model details are in Appendix F.1 and Appendix F.2.

2.2 Quantitative Models for Cellular Representation and Generation

Single-cell foundation models learn transferable representations by large-scale self-supervised pretraining, from rank-ordered gene sequences (Theodoris et al., 2023) and generative masked modeling (Cui et al., 2024) to larger pretraining scales (Hao et al., 2024), expression tokenization for annotation (Yang et al., 2022), zero-shot transfer across tissues, species, and modalities (Rosen et al., 2026), and embeddings of cell state across tissues and conditions (Adduri et al., 2025). Probabilistic and flow-matching models supply the statistical basis for generation: scVI provides hierarchical modeling and variational inference (Lopez et al., 2018), CFGen and SAVE perform multi-condition generation (Palma et al., 2025; Li et al., 2026), CellFlow and STATE model the transition from control to perturbed distributions (Klein et al., 2025; Adduri et al., 2025), scGen, CPA, and GEARS predict perturbed states from control cells (Lotfollahi et al., 2019; 2023; Roohani et al., 2024), CellOT casts the problem as neural optimal transport (Bunne et al., 2023), and biolord disentangles known and unknown perturbation attributes (Piran et al., 2024). Perturbation atlases and evaluation frameworks provide the data and metrics (Peidli et al., 2024; Wenkel et al., 2026; Viñas Torné et al., 2025). Both families take structured covariates such as batch, cell type, drug, or gene perturbation and return embeddings or distributions whose interface is purely quantitative, so they strengthen the quantitative side without closing a semantic–quantitative loop. Table 10 summarizes their capability coverage, and per-model details are in Appendix F.3 and Appendix F.4.

3 Preliminaries

Handling cell states and biological knowledge with one model requires making their mathematical forms explicit and specifying which mappings the model should establish between them.

Definition 1 (Two representation spaces)**.** *Fix a gene panel of G genes. The quantitative cell-state space is $\mathcal{X} = \mathbb{R}_+^G$: a cell state is a non-negative vector $x \in \mathbb{R}_+^G$ whose g-th entry is the expression of gene g in a unified coordinate, and cell states under one biological condition follow a conditional distribution $P(x \mid c)$ approximated in practice by a finite sample. The semantic space $\mathcal{S}$ consists of natural language, ontology terms, and task instructions, so a semantic unit $s \in \mathcal{S}$ is a discrete symbol sequence whose value set is not fixed in advance and can therefore name tissues, genes, or interventions outside the training data.*

Definition 2 (Multimodal cell model)**.** *A multimodal cell model is a family of mappings f_θ with shared parameters θ that takes a context of semantic descriptions and observable cell states and returns either a semantic label $y \in \mathcal{Y}$ or a conditional distribution of cell states $P_\theta(x \mid c)$, so it performs bidirectional inference between $\mathcal{S}$ and $\mathcal{X}$.*

Definition 3 (Four conditional mappings)**.** *Let g be gene text, C a candidate set, x_0 a control cell state, and c_p an intervention description. This paper selects four representative mappings, $(g, C) \to y$, $(x, C) \to y$, $c \to P(x \mid c)$, and $(x_0, c_p) \to P(x_1 \mid x_0, c_p)$, whose task names and output objects are summarized in Table 1.*

The condition c decomposes into biological attributes b and a semantic description s (Appendix E.1). Semantic tasks output discrete symbols over an open vocabulary and need a separable representation, while quantitative tasks output numerical distributions over fixed gene coordinates and need a reconstructable one. All four mappings share one quantitative coordinate: for a cell-state matrix $X \in \mathbb{R}_+^{N \times G}$ (cells as rows, genes as columns) with library size ℓ_i, the unified evaluation representa-

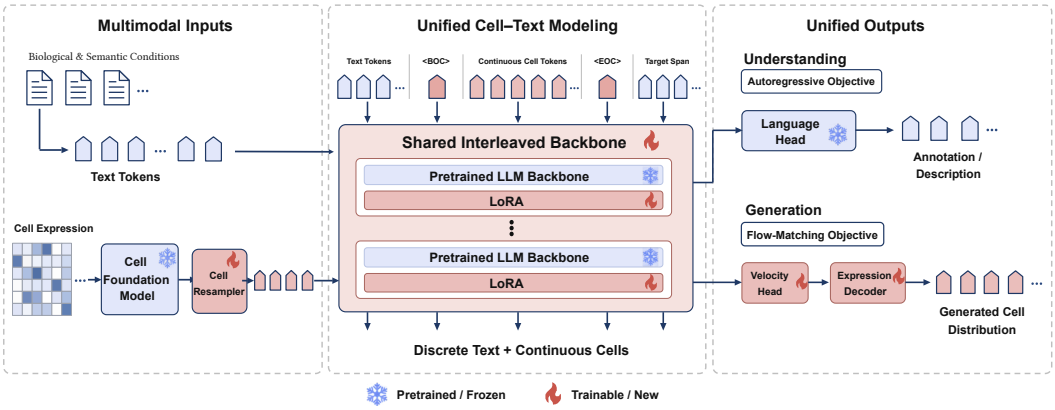


Figure 1: Overall architecture of CellUMM. Left: text tokens and an expression profile compressed into continuous cell programs by a frozen cell foundation model and a resampler. Middle: a shared interleaved backbone that processes both token types in one context under one set of low-rank adaptation parameters. Right: a language head with an autoregressive objective for semantic understanding, and a velocity head with a flow-matching objective whose samples are mapped back to native cell distributions by the frozen decoder.

Table 2: Input and output paths on the shared backbone.

Task	Inputs visible to the backbone	Supervised output
Cell type annotation	Instruction, candidate names, observed cell block	Autoregressive answer over type names; candidate ranking
Gene function annotation	Gene text, GO candidates, task instruction	32 candidate association probabilities, or one binary probability
Conditional cell generation	Organ and cell type text, cell block to generate	24×512 latent velocity
Perturbation response prediction	Intervention text, observed control block, treated block to generate	24×512 latent velocity

tion is $x_{ig} = \log(1 + 10^4 X_{ig}/\ell_i)$, i.e., $\log1\mathrm{p}(\mathrm{CP10K})$. Gene panel alignment, duplicate-identifier merging, and the zero mask are given in Appendix B.1.

4 Method

CellUMM couples semantic understanding and quantitative generation in one attention context (Figure 1). It consists of a frozen cell encoder, a trainable cell-program resampler, an interleaved language backbone, and a frozen expression decoder, with one set of adaptation parameters shared by the four tasks. A training sample is one interleaved sequence: text conditions and instructions enter it as tokens, and each cell state enters as a fixed-length block, `<text condition> <BOC>` $[z_1 \cdots z_{24}]$ `<EOC> <text answer>`, where $z_j \in \mathbb{R}^{512}$ is a continuous cell program. The two semantic tasks are supervised on the text output and the two generation tasks on the velocity of the generated block, where perturbation response prediction additionally provides an observed control block in the same sequence (Table 2). Sharing comes from shared parameters and a shared condition representation; no per-task condition encoder or decoder is added.

4.1 Cell Programs and the Interleaved Backbone

Cell Encoder and Resampler. Gene expression is variable-length and sparse, so a cell is encoded in two stages. The frozen encoder ranks genes by the ratio between a gene count and the median of that gene over the full panel, and maps the ranked gene tokens with the frozen Geneformer-12L-30M encoder (Theodoris et al., 2023) to contextual features $H \in \mathbb{R}^{L_{\mathrm{seq}} \times 512}$; this key is order-equivalent to the original Geneformer sorting rule, which only adds a within-cell library-size factor. A trainable resampler then compresses the variable-length features into a fixed number of units with two structurally identical blocks, following query-based latent compression (Jaegle et al., 2022; Li et al., 2023); each block applies cross-attention to the memory $M = W_m H$, self-attention among the queries, and a feed-forward transformation:

$$Q^{(l+1)} = \mathrm{Block}_l\left(Q^{(l)}, M\right), \qquad z = \mathrm{LN}\left(Q^{(2)}\right) \in \mathbb{R}^{24 \times 512}. \tag{1}$$

The output z is the continuous cell program: the 24 units trade representation capacity against sequence cost, and they share the token width of 512, so no modality-alignment loss is needed. The remaining resampler settings are given in Appendix E.3.

Interleaved Backbone. Each cell program is projected to the hidden space of the language backbone by $W_p \in \mathbb{R}^{3584\times512}$ and added to a time embedding $e_{\text{time}}(t_j)$ and a slot embedding $e_{\text{slot}}(j)$:

$$u_j = W_p z_j + e_{\text{time}}(t_j) + e_{\text{slot}}(j), \tag{2}$$

where e_{time} embeds the flow-matching time t_j of slot j and e_{slot} distinguishes the 24 positions within one cell block. The backbone is Qwen2.5-7B-Instruct (Yang et al., 2024), whose hidden size of 3,584 sets the output width of W_p and the embedding width of the boundary markers; its base weights are frozen and only low-rank adaptation parameters (Hu et al., 2022) are updated. Because low-rank adaptation updates neither the word embedding matrix nor the output head, the boundary markers <BOC> and <EOC> receive dedicated parameters $E_b, W_b \in \mathbb{R}^{2\times3584}$ in place of their vocabulary embeddings and logits (Appendix E.5).

Fusion is controlled by the attention mask rather than by output concatenation: text positions follow a causal mask, the 24 slots of one cell block attend to each other bidirectionally and thus form a single random variable, padding positions serve neither as queries nor as keys, and the whole block shares one rotary position (Su et al., 2024) while slots are separated by $e_{\text{slot}}(j)$ (Appendix E.4).

4.2 Expression Decoder

Decoder Structure. Programs must be mapped back to gene expression when samples are produced. A frozen decoder pools the 24 units with learned attention weights and passes the pooled vector through a low-rank factor model:

$$\begin{aligned} &\alpha = \operatorname{softmax}\left(w_s^\top z\right), \qquad h = \sum_j \alpha_j z_j, \qquad \kappa = W_2\,\mathrm{SiLU}(W_1\,\mathrm{LN}(h)) \in \mathbb{R}^{256}, \\ &\hat{x} = \operatorname{softplus}\left(\kappa F^\top + b\right), \qquad F \in \mathbb{R}^{21523\times256}. \end{aligned} \tag{3}$$

Here $\operatorname{softplus}$ guarantees a non-negative output, and the 256-dimensional factor subspace makes the decoder grow linearly in the number of genes while restricting the representable covariance structure. The decoder is pretrained on expression reconstruction, then calibrated on the training data and frozen: it takes no gradient during joint training and is used only when a cell program is sampled. It emits values on the $\log1\mathrm{p}(\mathrm{CPM})$ scale, which are converted per cell to $\log1\mathrm{p}(\mathrm{CP10K})$ at evaluation time, and the outputs of the two semantic tasks never pass through it. Conditional cell generation and perturbation response prediction share one decoder, so the metrics of the two generation tasks fall on the same quantitative coordinates (Appendix E.6).

4.3 Conditional Flow Matching

Objective. CellUMM models cell programs by a conditional velocity field rather than by a deterministic regression target. Let z_1 be the program of the true target cell, and let $\varepsilon \sim \mathcal{N}(0, I)$ and $t \sim \mathcal{U}(0, 1)$ be independently sampled noise and time. Following the flow-matching framework (Lipman et al., 2023), an intermediate state is constructed along a straight path and the displacement is used as the regression target:

$$z_t = (1-t)\,\varepsilon + t\,z_1, \qquad v^* = z_1 - \varepsilon, \tag{4}$$

and the shared backbone predicts a velocity at that position, trained by the mean squared error between the prediction and the target displacement:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}\,\|v_\theta(z_t, t, c) - (z_1 - \varepsilon)\|^2\,. \tag{5}$$

Role-Conditioned Velocity Field. Perturbation response prediction conditions on a real control cell, so one sequence carries an observed block and a generated block. A single velocity field with an explicit control argument covers both generation tasks:

$$v_\theta\left(z_t,\, t,\, c,\, z^{\text{ctrl}}\right), \qquad z^{\text{ctrl}} = \begin{cases} \varnothing, & \text{conditional generation,} \\ z_{\text{ctrl}}, & \text{perturbation response prediction,} \end{cases} \tag{6}$$

so the two tasks share one backbone, one velocity head, and one condition interface, and differ only in whether a control cell is present in the context. The observed block keeps clean program values and acts as context in attention, and the loss is computed on the generated slots only:

$$t_i = \begin{cases} t \sim \mathcal{U}(0,1), & i \in \mathcal{T}, \\ 1, & i \in \mathcal{O}, \end{cases} \qquad \mathcal{L}_{\text{flow}}^{\mathcal{T}} = \frac{1}{|\mathcal{T}|} \Big\| \big(v_\theta(z_t, t, c, z^{\text{ctrl}}) - (z_1 - \varepsilon)\big)_{\mathcal{T}} \Big\|^2, \tag{7}$$

where $\mathcal{T}$ is the target (generated) block and $\mathcal{O}$ the observed control block, so gradients do not flow through the context block. Because time embeddings are given per position, different blocks within one sequence can carry different times.

Sampling and Generation. Training is defined in the continuous program space, whereas sampling and the production of expression profiles happen in the expression space; the decoder takes no part in training. At inference the velocity field is integrated from standard normal noise with the Heun method for neural ordinary differential equations (Chen et al., 2018), and the endpoint is mapped back to gene expression by the frozen decoder D:

$$\begin{aligned} &z^{(0)} \sim \mathcal{N}(0, I), \quad \Delta = \tfrac{1}{N_{\text{step}}}, \quad t_k = k\Delta, \quad a^{(k)} = v_\theta\big(z^{(k)}, t_k, c, z^{\text{ctrl}}\big), \\ &b^{(k)} = v_\theta\big(z^{(k)} + \Delta a^{(k)}, t_k + \Delta, c, z^{\text{ctrl}}\big), \quad z^{(k+1)} = z^{(k)} + \tfrac{\Delta}{2}\big(a^{(k)} + b^{(k)}\big), \\ &\hat{x} = D\big(z^{(N_{\text{step}})}\big), \quad N_{\text{step}} = 8, \quad k = 0, \ldots, N_{\text{step}} - 1. \end{aligned} \tag{8}$$

Repeated sampling under one condition yields an empirical approximation of the conditional distribution. Heun is a second-order method that needs two velocity evaluations per step, i.e., 16 evaluations for $N_{\text{step}} = 8$; the choice of N_{step} is discussed in Appendix E.7.

4.4 Joint Training Objective

Task Losses. Cell type annotation applies autoregressive cross-entropy to the answer suffix and adds a candidate ranking loss as auxiliary supervision,

$$\mathcal{L}_{\text{anno}} = -\sum_{i \in \mathcal{A}} \log p_\theta\big(y_i \mid y_{<i}, x\big) + \lambda_r \, \mathcal{L}_{\text{rank}}, \qquad q_k = \frac{\exp\big(\cos(u, e_k)/\tau\big)}{\sum_{j=1}^{K} \exp\big(\cos(u, e_j)/\tau\big)}, \tag{9}$$

where $\mathcal{A}$ is the set of valid answer-suffix positions, u is the mean-pooled cell-program representation, e_k the frozen word embedding of candidate k, both projected to 256 dimensions and L_2-normalized, and $\tau = 0.07$, $\lambda_r = 0.5$; the cross-entropy on q uses label smoothing of 0.05. Predictions are ranked by the average answer-suffix log-likelihood (Appendix A.4). Gene function annotation takes the shared hidden state at the last non-padding text position and produces candidate logits through a linear head, with 32-dimensional and 1-dimensional outputs for the multi-label and binary forms:

$$\begin{aligned} \mathcal{L}_{\text{gene}} &= -\frac{1}{K} \sum_{k=1}^{K} \Big[w^+ y_k \log p_k + (1 - y_k) \log(1 - p_k) \Big], \\ p_k &= \sigma(a_k), \qquad w^+ = \begin{cases} 3, & \text{multi-label}, \\ 1, & \text{binary}. \end{cases} \end{aligned} \tag{10}$$

Joint Objective. Both generation tasks supervise the velocity field on the generated block (equation 7), $\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{pert}} = \mathcal{L}_{\text{flow}}^{\mathcal{T}}$, and the joint objective averages the four losses:

$$\mathcal{L} = \frac{1}{4} \left(\mathcal{L}_{\text{anno}} + \mathcal{L}_{\text{gene}} + \mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{pert}} \right). \tag{11}$$

Each step draws one batch per task and back-propagates the sum of the four losses scaled by $1/4$, without rescaling them to the loss magnitude. Training and inference use the same backbone and the same frozen decoder, with no per-task parameter switching; the batch composition and optimization settings are given in Section 5.1 and Appendix E.9. Training uses no loss in the expression space, so a small velocity-regression error does not by itself guarantee good distribution metrics; quantitative evaluation therefore also reports distribution distances and high-variable-gene consistency in addition to mean-level errors (Appendix E.8, Appendix D.2).

Table 3: Results on cell type annotation. All methods are compared on our in-house fine-grained test split (200 test cells, 31 candidate classes). Rank is the position of a model by its mean rank over the metrics of the task; lower is better.

	STATE SE	LangCell	GPT-OSS 120B	Cell Whisperer	C2S-27B	DeepSeek V4-Pro	Geneformer	Qwen3.5-27B Thinking	Qwen3.5-9B Thinking	CellTok	Instruct Cell	CellUMM
Accuracy (%) ↑	<u>51.00</u>	49.00	45.50	44.00	44.00	41.50	37.50	29.50	23.50	10.50	4.00	**60.50**
Macro-Precision (%) ↑	45.18	47.81	<u>50.00</u>	45.83	46.67	41.82	35.26	27.89	35.86	9.91	0.42	**56.15**
Macro-Recall (%) ↑	<u>52.07</u>	49.69	45.70	44.39	44.39	42.01	37.40	29.72	23.12	10.68	3.76	**61.75**
Macro-F1 (%) ↑	<u>45.79</u>	43.60	43.02	41.22	40.83	38.89	31.63	24.22	21.57	6.95	0.70	**54.78**
Rank	2	2	4	5	5	7	8	9	10	11	12	**1**

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

All RNA data are aligned to 21,523 gene coordinates on the $\log1p(\mathrm{CP10K})$ coordinate and masked at unmeasured genes. Training uses human GOA annotations and the GO ontology (Ashburner et al., 2000; Aleksander et al., 2025) for gene function annotation, a frozen CELLxGENE source list (Abdulla et al., 2025) for cell type annotation, seven-organ Tabula Sapiens (Tabula Sapiens Consortium, 2022) for conditional cell generation, and Tahoe-100M chemical (Zhang et al., 2025) with Replogle–Nadig genetic (Replogle et al., 2022; Nadig et al., 2025) interventions for perturbation response prediction; splits are made by identity (gene, donor, or condition), so no unit appears on both sides. Evaluation uses gene-disjoint GOA questions, donor-disjoint generation conditions, the in-house fine-grained cell type annotation split, the H1 hESC test set of the Virtual Cell Challenge (Roohani et al., 2025), and SciPlex3 (Srivatsan et al., 2020); corpus statistics and split details are given in Appendix B.

All four tasks are compared with representative methods under the same data splits, gene coordinates, and candidate construction; the baseline families, input conditions, and comparison boundaries are given in Appendix C.

CellUMM is trained jointly on the four tasks; the optimization configuration, the sampling protocol, and the metric definitions are given in Appendix E.9 and Appendix D.

5.2 CELL TYPE ANNOTATION

Cell type annotation compares CellUMM with cell–language models, single-cell foundation models, and general-purpose large language models under a unified candidate set. CellUMM obtains the best value on all four metrics, with an accuracy 9.5 percentage points above the second-best method, and the scale of the general-purpose models does not transfer: GPT-OSS-120B and the Qwen3.5 series remain below the domain models, so candidate discrimination here depends on a quantitative representation of cell state that a language prior cannot provide. Results are listed in Table 3.

5.3 GENE FUNCTION ANNOTATION

Gene function annotation uses a multi-label form with 32 candidate GO terms and a binary form with one candidate term; both share the candidate construction, the gene split, and the threshold. Figure 2 reports the results as percentages, with AUROC rescaled from its 0–1 range to the same axis. In the multi-label form, positives are sparse within each candidate set and term granularity differs across GO namespaces. CellUMM obtains the best exact match and macro-F1, about 16 percentage points above BioBERT in exact match, and its micro-recall is clearly higher than that of TF-IDF at a lower micro-precision, so it returns a more complete association set. In the binary form the gaps narrow: the term-based baseline already exceeds 80% accuracy, so this form mainly tests shallow matching between a candidate and a gene description. CellUMM still obtains the best value on all four metrics, 2.1 percentage points above BioBERT and 1.0 point above PubMedBERT

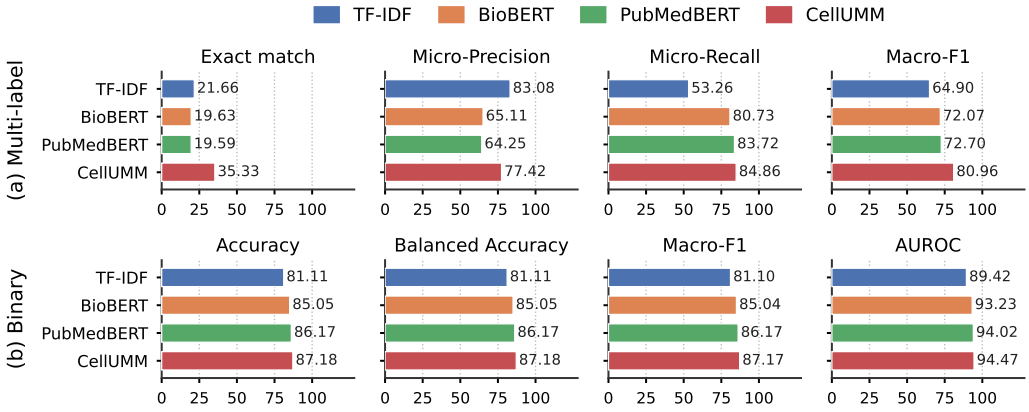


Figure 2: Gene function annotation in the (a) multi-label (32 candidate GO terms) and (b) binary (one candidate term) forms. Each panel reports one metric for TF-IDF (linear), BioBERT, PubMedBERT, and CellUMM, and the value of every bar is annotated at its end.

Table 4: Results on conditional cell generation. The average rank is the mean rank of a model over the six metrics; lower is better.

Model	MMD^2 ↓	Mean-expr. MSE ↓	HVG1000 Jaccard ↑	HVG2000 Jaccard ↑	HVG1000 MSE ↓	HVG2000 MSE ↓	Avg. rank
Organ/cell-type mean	0.148	**0.0251**	0	0	**0.0489**	**0.0456**	2.83
SAVE	0.157	0.0293	0.063	0.113	0.0628	0.0595	2.67
scVI	0.162	0.0452	0.044	0.090	0.0956	0.0906	3.83
CFGen	0.221	0.0653	0.033	0.066	0.1163	0.1171	5.00
InstructCell	0.364	0.1054	0.046	0.090	0.1955	0.1962	5.00
CellUMM	**0.136**	0.0253	**0.069**	**0.114**	0.0538	0.0499	**1.50**

in accuracy. The two forms together show that a shared semantic representation helps most when the task depends on relations among candidates.

5.4 Conditional Cell Generation

Conditional cell generation receives a natural-language condition of organ and cell type and must produce a single-cell expression distribution that satisfies it (Table 4). We include the training mean of the same organ and cell type as a statistical baseline next to scVI, CFGen, SAVE, and InstructCell, so that the predictability of the conditional mean is separated from the generation capability of the conditional distribution. CellUMM achieves the best distribution distance and high-variable-gene consistency and ranks first on average. The training mean shows why distribution modeling is necessary: it is best on the three MSE metrics, which only measure the conditional mean, but it is behind CellUMM on MMD^2 and its high-variable-gene consistency is zero, because one vector cannot express the heterogeneity within a condition. This is why CellUMM generates a conditional distribution that a mean cannot represent.

5.5 Perturbation Response Prediction

Perturbation response prediction is evaluated in zero-shot and fine-tuned settings for genetic and chemical interventions under the same conditions and metrics (Table 5).

Under genetic interventions CellUMM obtains the best value on all four metrics in both settings, and its advantage is largest on the direction-based metrics: after fine-tuning its directional reach is 0.5556 against 0.2920 for CellFlow and 0.2277 for STATE. Discrimination is close to saturation in the zero-shot setting, where the three methods are within 0.013 of each other on PDS, so we also report direction-based and differential-expression-based metrics, which show whether a model captures intervention-specific responses.

Table 5: Results on perturbation response prediction. Best in bold, second best underlined.

Model	PDS ↑	Directional fidelity ↑	Directional reach ↑	DE Jaccard ↑
Gene perturbation				
CellFlow (zero-shot)	0.5000	0.5430	0.0851	0.00619
STATE (zero-shot)	0.4958	0.5139	0.0559	0.01850
CellUMM (zero-shot)	**0.5083**	**0.5463**	**0.1207**	**0.01983**
CellFlow (fine-tuned)	0.4984	0.5467	0.2920	0.00710
STATE (fine-tuned)	0.4943	0.5482	0.2277	0.01971
CellUMM (fine-tuned)	**0.5000**	**0.7448**	**0.5556**	**0.02176**
Chemical perturbation				
CellFlow (zero-shot)	0.4961	0.5374	0.0796	0.00583
STATE (zero-shot)	0.4917	0.5092	0.0518	0.01726
CellUMM (zero-shot)	**0.5032**	**0.5408**	**0.1142**	**0.01847**
CellFlow (fine-tuned)	0.4952	0.5321	0.2634	0.00662
STATE (fine-tuned)	0.4916	0.5347	0.2051	0.01842
CellUMM (fine-tuned)	**0.4971**	**0.7264**	**0.5108**	**0.02039**

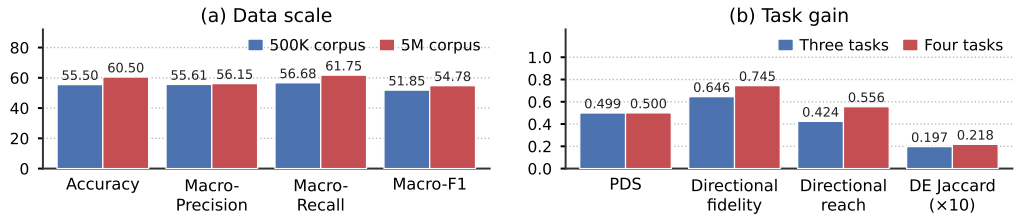


Figure 3: Data scale and task gain. (a) Annotation corpus size (500K vs. 5M cells) on the in-house fine-grained cell type annotation task. (b) Removing the annotation task, with the three-task and four-task models fine-tuned identically on gene perturbation response prediction; DE Jaccard is plotted at ten times its value.

Chemical interventions follow the same pattern: CellUMM leads on all four metrics in both settings, and fine-tuning raises its directional reach from 0.1142 to 0.5108 and its fidelity from 0.5408 to 0.7264, while both baselines stay below 0.27 on reach.

5.6 Data Scale and Task-Gain Analysis

Besides the four main tasks, we study how the size of the cell type annotation corpus affects the semantic task and whether annotation supervision benefits perturbation response prediction (Figure 3), which reports both analyses in two panels.

The first analysis trains the same model on annotation corpora of 500K and 5M cells and evaluates both checkpoints on the in-house fine-grained cell type annotation task. The larger corpus improves all four metrics, most on accuracy (55.50 to 60.50) and macro-recall (56.68 to 61.75), so this task has not saturated at 500K cells.

The second analysis removes the cell type annotation task and compares the three-task model with the four-task CellUMM after the same fine-tuning on gene perturbation response prediction. Directional fidelity drops from 0.7448 to 0.6458, directional reach from 0.5556 to 0.4238, and DE Jaccard from 0.02176 to 0.01971, whereas discrimination is unchanged (0.5000 versus 0.4992). As a result, annotation supervision mainly improves the direction-based metrics of perturbation prediction.

6 Conclusion

CellUMM casts cellular modeling as four conditional mappings between a semantic space and a quantitative cell-state space, and realizes them in one model with a shared interleaved backbone and continuous cell programs. The same latent programs support semantic prediction, conditional generation, and perturbation state transition, and the joint objective lets the semantic representation carry quantitative information that generation can use. On the benchmarks CellUMM outperforms task-specific baselines, leading on the majority of metrics in four tasks, and the distribution metrics show that it reproduces the conditional distribution as a whole. Future work includes dispersion calibration of the decoder and extension to further modalities.

AI USE STATEMENT

We employed generative AI tools in two capacities while writing this manuscript: to help with writing and to polish the language, and to search for and identify relevant literature. All outputs produced with AI assistance were individually reviewed by the authors: papers recommended or retrieved by AI were confirmed to exist and to be relevant by reading the originals, and no technical statement or finding was changed by the language editing. Responsibility for the entire content of this paper rests with the authors.

ETHICS STATEMENT

The experiments in this paper rely exclusively on public datasets; their sources and the way they were processed are documented in the appendix. We do not foresee any concerns regarding fairness or bias, privacy and security, regulatory compliance, competing interests, or the integrity of the research.

REPRODUCIBILITY STATEMENT

The main text together with the appendix provides a complete account of our experiments: how the datasets were built and preprocessed, the model architecture and essential implementation choices, the hyperparameters used for training, and the protocol followed for evaluation.

REFERENCES

Shibla Abdulla, Brian Aevermann, Pedro Assis, et al. CZ CELLxGENE Discover: a single-cell data platform for scalable exploration, analysis and modeling of aggregated data. *Nucleic Acids Research*, 53:D886–D900, 2025. doi: 10.1093/nar/gkae1142.

Abhinav Adduri et al. Predicting cellular responses to perturbation across diverse contexts with State. bioRxiv preprint, 2025.

Suzi A. Aleksander, James P. Balhoff, Seth Carbon, et al. The gene ontology knowledgebase in 2026. *Nucleic Acids Research*, 54:D1779–D1792, 2025. doi: 10.1093/nar/gkaf1292.

Dvir Aran, Agnieszka P. Looney, Leqian Liu, et al. Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. *Nature Immunology*, 20:163–172, 2019. doi: 10.1038/s41590-018-0276-y.

Michael Ashburner, Catherine A. Ball, Judith A. Blake, et al. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000. doi: 10.1038/75556.

Charlotte Bunne, Stefan G. Stark, Gabriele Gut, et al. Learning single-cell perturbation responses using neural optimal transport. *Nature Methods*, 20:1759–1768, 2023. doi: 10.1038/s41592-023-01969-x.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. arXiv:1806.07366.

Haotian Cui, Chloe Wang, Hassaan Maan, et al. scGPT: toward building a foundation model for single-cell multi-omics using generative AI. *Nature Methods*, 21:1470–1480, 2024. doi: 10.1038/s41592-024-02201-0.

César Domínguez Conde, Canran Xu, Lorna B. Jarvis, et al. Cross-tissue immune cell analysis reveals tissue-specific features in humans. *Science*, 376:eabl5197, 2022. doi: 10.1126/science.abl5197.

Yin Fang, Xinle Deng, Kangwei Liu, et al. A multi-modal AI copilot for single-cell analysis with instruction following. arXiv preprint, 2025. arXiv:2501.08187.

Yuxuan Gao et al. Language may be all omics needs: Harmonizing multimodal data for omics understanding with CellHermes. bioRxiv preprint, 2025.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

Yu Gu, Robert Tinn, Hao Cheng, et al. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare*, 3(1):1–23, 2021. doi: 10.1145/3458754.

Minsheng Hao, Jing Gong, Xin Zeng, et al. Large-scale foundation model on single-cell transcriptomics. *Nature Methods*, 21:1481–1491, 2024. doi: 10.1038/s41592-024-02305-7.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, et al. Perceiver IO: A general architecture for structured inputs and outputs. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2107.14795.

Dominik Klein, Jonas Simon Fleck, et al. CellFlow enables generative single-cell phenotype modeling with flow matching. bioRxiv preprint, 2025.

David Lähnemann, Johannes Köster, Ewa Szczurek, et al. Eleven grand challenges in single-cell data science. *Genome Biology*, 21:31, 2020. doi: 10.1186/s13059-020-1926-6.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, et al. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020. doi: 10.1093/bioinformatics/btz682.

Daniel Levine, Syed A. Rizvi, Sacha Lévy, et al. Cell2Sentence: Teaching large language models the language of biology. bioRxiv preprint, 2023.

Jiahao Li, Jiayi Dong, Peng Ye, Xiaochi Zhou, et al. SAVE: A generalizable framework for multi-condition single-cell generation with gene block attention. arXiv preprint, 2026. arXiv:2604.16776.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2301.12597.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.02747.

Romain Lopez, Jeffrey Regier, Michael B. Cole, Michael I. Jordan, and Nir Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15:1053–1058, 2018. doi: 10.1038/s41592-018-0229-2.

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

Mohammad Lotfollahi, Anna Klimovskaia Susmelj, Carlo De Donno, et al. Predicting cellular responses to complex perturbations in high-throughput screens. *Molecular Systems Biology*, 19: e11517, 2023. doi: 10.15252/msb.202211517.

Ajay Nadig, Joseph M. Replogle, Angela N. Pogson, et al. Transcriptome-wide analysis of differential expression in perturbation atlases. *Nature Genetics*, 57:1228–1237, 2025. doi: 10.1038/s41588-025-02169-3.

Thomas M. Norman, Max A. Horlbeck, Joseph M. Replogle, et al. Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. *Science*, 365:786–793, 2019. doi: 10.1126/science.aax4438.

Alessandro Palma, Till Richter, Hanyi Zhang, et al. Multi-modal and multi-attribute generation of single cells with CFGen. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2407.11734.

Stefan Peidli, Tessa D. Green, Ciyue Shen, et al. scPerturb: harmonized single-cell perturbation data. *Nature Methods*, 21:531–540, 2024. doi: 10.1038/s41592-023-02144-y.

Zoe Piran, Niv Cohen, Yedid Hoshen, and Mor Nitzan. Disentanglement of single-cell data with biolord. *Nature Biotechnology*, 42:1678–1683, 2024. doi: 10.1038/s41587-023-02079-x.

Aviv Regev, Sarah A. Teichmann, Eric S. Lander, et al. The human cell atlas. *eLife*, 6:e27041, 2017. doi: 10.7554/eLife.27041.

Joseph M. Replogle, Reuben A. Saunders, Angela N. Pogson, et al. Mapping information-rich genotype-phenotype landscapes with genome-scale Perturb-seq. *Cell*, 185:2559–2575, 2022. doi: 10.1016/j.cell.2022.05.013.

Syed A. Rizvi, Daniel Levine, Aakash Patel, et al. Scaling large language models for next-generation single-cell analysis. bioRxiv preprint, 2025.

Yusuf Roohani, Kexin Huang, and Jure Leskovec. Predicting transcriptional outcomes of novel multigene perturbations with GEARS. *Nature Biotechnology*, 42:927–935, 2024. doi: 10.1038/s41587-023-01905-6.

Yusuf H. Roohani, Tony J. Hua, Po-Yuan Tung, et al. Virtual cell challenge: Toward a turing test for the virtual cell. *Cell*, 188:3370–3374, 2025. doi: 10.1016/j.cell.2025.06.008.

Yanay Rosen, Yusuf Roohani, Ayush Agrawal, et al. Universal cell embedding provides a foundation model for cell biology. *Nature*, 656:183–191, 2026. doi: 10.1038/s41586-026-10689-z.

Moritz Schaefer, Giovanni Palla, Bryan Ho, et al. Multimodal learning enables chat-based exploration of single-cell data. *Nature Biotechnology*, 2025. doi: 10.1038/s41587-025-02857-9.

Sanjay R. Srivatsan, José L. McFaline-Figueroa, Vijay Ramani, et al. Massively multiplex chemical transcriptomics at single-cell resolution. *Science*, 367:45–51, 2020. doi: 10.1126/science.aax6234.

Jianlin Su, Murtadha Ahmed, Yu Lu, et al. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Tabula Sapiens Consortium. The tabula sapiens: A multiple-organ, single-cell transcriptomic atlas of humans. *Science*, 376:eabl4896, 2022. doi: 10.1126/science.abl4896.

Christina V. Theodoris, Ling Xiao, Anant Chopra, et al. Transfer learning enables predictions in network biology. *Nature*, 618:616–624, 2023. doi: 10.1038/s41586-023-06139-9.

Ramon Viñas Torné et al. Systema: a framework for evaluating genetic perturbation response prediction beyond systematic variation. *Nature Biotechnology*, 44:1050–1059, 2025. doi: 10.1038/s41587-025-02777-8.

Fabian Wenkel, William Tu, Bram Masschelein, et al. TxPert: using multiple knowledge graphs for prediction of transcriptomic perturbation effects. *Nature Biotechnology*, 2026. doi: 10.1038/s41587-026-03113-4.

Tianxiang Xiao et al. Tokenizing single-cell transcriptomes as a native language for large language models. bioRxiv preprint, 2025.

An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. arXiv preprint, 2024. arXiv:2412.15115.

Fan Yang, Wenchuan Wang, Fang Wang, et al. scBERT as a large-scale pretrained deep language model for cell type annotation of single-cell RNA-seq data. *Nature Machine Intelligence*, 4:852–866, 2022. doi: 10.1038/s42256-022-00534-z.

Jiaqi Zhang et al. Tahoe-100M: A giga-scale single-cell perturbation atlas for context-dependent gene function and cellular modeling. bioRxiv preprint, 2025.

Suyuan Zhao, Jiahuan Zhang, Yushuai Wu, et al. LangCell: Language-cell pre-training for cell identity understanding. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2405.06708.

Chunting Zhou, Lili Yu, Arun Babu, et al. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2408.11039.

CONTENTS OF THE APPENDIX

A Task

A.1 Gene Function Annotation: Semantics to Structured Semantics

A.2 Conditional Cell Generation: Semantics to Quantitative States

A.3 Perturbation Response Prediction: Conditional Quantitative State Transition

A.4 Cell Type Annotation: Quantitative States to Semantics

B Data

B.1 Unified Expression Coordinates and Data Boundaries

B.2 Gene Function Annotation Data

B.3 Conditional Cell Generation Data

B.4 Perturbation Response Prediction Data

B.5 Cell Type Annotation Data

B.6 Downstream Data

C Baseline

C.1 Gene Function Annotation

C.2 Conditional Cell Generation

C.3 Perturbation Response Prediction

C.4 Cell Type Annotation

D Metric

D.1 Semantic Discrimination Metrics

D.2 Expression Distribution Metrics

D.3 Perturbation Response Prediction Metrics

E Model

E.1 Notation and Tensor Shapes

E.2 Expression Encoding and Tokenization

E.3 Continuous Cell Program Resampler

E.4 Interleaved Attention and Position Numbering

E.5 Parameterization of the Boundary Markers

E.6 Expression Decoder

E.7 Flow-Matching Objective and Sampler

E.8 Output Heads and Loss Functions of the Four Tasks

E.9 Parameter Groups and Training Configuration

F Related Work

F.1 Biomedical Language Models for Gene-Level Semantics

F.2 Cell–Language Models for Multimodal Cellular Understanding

F.3 Single-Cell Foundation Models for Cellular Representation

F.4 Probabilistic and Flow-Based Models for Cellular Generation

A TASK

A.1 GENE FUNCTION ANNOTATION: SEMANTICS TO STRUCTURED SEMANTICS

Gene function annotation takes a gene identity and the text representation of functional candidates as input and predicts the association between the gene and candidate Gene Ontology (GO) terms. Let the input gene be g, the candidate set be $C = (c_1, \ldots, c_K)$, and the label vector be $y \in \{0,1\}^K$. The multi-label form fixes $K = 32$ and lets the model output an independent probability for each candidate position; the binary form takes a single candidate term as input and outputs one association probability. The biological process, molecular function, and cellular component branches of GO form three kinds of functional semantics:

$$p_k = \sigma(a_k), \qquad \hat{y}_k = \mathbf{1}\left[p_k \geq 0.5\right], \qquad k = 1, \ldots, K. \tag{12}$$

The multi-label input contains the gene identifier, the task instruction, and the index and name of each candidate within the same GO namespace; the binary input additionally contains the candidate term definition. The candidate order corresponds item by item to the label vector, so the k-th dimension of the output head corresponds to the k-th candidate of the current input rather than to a fixed k-th term in the GO vocabulary. Training reads numerical labels directly and uses a classification loss; answer text in the samples is used only for data inspection. The semantic output of this task is a set of GO associations. The discrimination metrics are given in Appendix D.1.

A multi-label sample contains 1–8 positive candidates, and the remaining candidates are drawn from terms that are not recorded as positive associations for that gene. These negative candidates represent cases without an annotation record in the knowledge base, so evaluation measures annotation recovery with respect to a fixed snapshot. Splitting by gene prevents one gene from overlapping across different candidate combinations and provides a controlled evaluation setting for cross-gene generalization.

A.2 CONDITIONAL CELL GENERATION: SEMANTICS TO QUANTITATIVE STATES

Conditional cell generation takes a natural-language condition c of organ and cell type as input and generates single-cell expression states that satisfy it. The training data are the real cell sets corresponding to the conditions, and each optimization step uses one real cell as the target of the continuous latent variable, so that the supervision signal retains the cell heterogeneity within a condition. At inference time, Gaussian initial noise is drawn repeatedly and passed through the same conditional velocity field and expression decoder to obtain several single-cell samples:

$$\varepsilon^{(j)} \sim \mathcal{N}(0, I), \qquad \hat{x}^{(j)} = D\Big(\Phi_\theta\Big(\varepsilon^{(j)}; c\Big)\Big), \qquad j = 1, \ldots, n, \tag{13}$$

where Φ_θ denotes the ordinary differential equation integration from $t = 0$ to 1 and D is the fixed expression decoder. The n independently generated samples form an empirical approximation of the conditional distribution. The donor identifier is used for data grouping and generalization evaluation but does not enter the condition input, so cross-donor evaluation tests distribution transfer under the same organ and cell type condition.

A.3 PERTURBATION RESPONSE PREDICTION: CONDITIONAL QUANTITATIVE STATE TRANSITION

Perturbation response prediction receives a control cell expression x_0 and an intervention description c_p and generates the perturbed expression x_1. The condition of a chemical intervention is the drug name, dose, and dose unit; the condition of a genetic intervention is the target gene and the intervention task text. The cell line is used to match control and treated samples. Let the cell background be s; the training data are then characterized by

$$x_0 \sim P_{\text{control}}(\cdot \mid s), \qquad x_1 \sim P_{\text{treated}}(\cdot \mid s, c_p), \qquad \hat{x}_1 \sim p_\theta(\cdot \mid x_0, c_p). \tag{14}$$

Control cells and treated cells in training are drawn independently under a matched background rather than measured longitudinally from the same cell before and after intervention, so the supervision target is a state-distribution transition under background matching. At inference time the control latent variable serves as a fixed observed block, and the block to generate is initialized from independent Gaussian noise and takes part in the flow integration.

Table 6: Statistical units and split boundaries of the four datasets.

Task	Data and scale	Split unit
Gene function annotation	19,627 genes; 289,053 positive associations	Gene identity
Conditional cell generation	405,058 cells; 7 organs; 566 raw conditions	Donor; the main training development split is globally isolated by raw donor
Perturbation response prediction	62,285 upstream conditions; the main training run uses 221,752 real target cells	Drug / target-gene identity grouping within a source
Cell type annotation	5,000,000 training cells; 548 canonical types	Training construction is separate from external evaluation

A.4 Cell Type Annotation: Quantitative States to Semantics

Cell type annotation takes single-cell RNA expression, an annotation instruction, and candidate cell types as input and outputs one type name from the candidate set. Expression is first converted into continuous cell programs by the cell encoder and the resampler, and then passed to the shared backbone together with the text. Training applies autoregressive cross-entropy to the answer tokens of the target type name and uses a candidate ranking loss as auxiliary supervision. Inference ranks candidates by the average log-likelihood of their answer suffixes:

$$\hat{y} = \operatorname*{argmax}_{y \in C} \; |a(y)|^{-1} \sum_{j} \log p_\theta(a_j(y) \mid x, c, a_{<j}(y)) \,, \tag{15}$$

where $a(y)$ denotes the complete answer suffix corresponding to candidate name y, including the end-of-answer marker, and $a_{<j}(y)$ is its first $j-1$ tokens. Ranking by the average log-likelihood of the suffix removes the bias caused by different candidate text lengths. The output uses language tokens, and the evaluation range is determined by the given candidate set.

B Data

B.1 Unified Expression Coordinates and Data Boundaries

The four datasets provide supervision for functional knowledge, conditional distribution generation, perturbation state transition, and cell identity, respectively. RNA-based tasks are aligned to 21,523 gene coordinates while the observation gene mask of the source data is retained. Column orders of different sources are reordered by gene identifier, and duplicate identifiers are merged in the raw count matrix. Counts are normalized by library size, and evaluation uses $\log 1\mathrm{p}(\mathrm{CP10K})$ uniformly. Unmeasured genes are marked by a mask and do not enter quantitative metrics. Table 6 gives the statistical unit and the split boundary of each dataset.

B.2 Gene Function Annotation Data

Functional supervision comes from human GOA annotations and the GO ontology. The construction snapshot uses the GAF annotation generated on 28 July 2026 and the ontology version of 26 July 2026. Preprocessing keeps the canonical symbols of protein-coding genes, removes NOT negative annotations, obsolete terms, and root terms, and de-duplicates gene–term positive associations. The candidate catalog finally contains 19,627 genes, 289,053 positive associations, and 38,092 valid GO terms, and the evidence codes cover electronic inference evidence such as IEA.

Splits are made by gene identity at roughly 70%/10%/20%, after which questions are constructed within each gene set so that different candidate combinations of one gene do not cross splits. Multi-label candidates are fixed at 32, and negative sampling mixes ontology-adjacent terms with random terms, of which hard negatives account for 0.6; the binary form keeps the numbers of positive and negative samples balanced. All candidate sampling excludes the known positive associations of that gene in the knowledge base. Table 7 lists the number of genes and questions in each split, and Figure 4 shows the split sizes and the distribution of positive associations over the three GO namespaces.

Table 7: Gene function annotation splits. Question counts and gene counts use different statistical units.

Split	Genes	Multi-label questions	Binary questions
Training	13,738	300,000	200,000
Validation	1,962	6,741	20,000
Test	3,927	13,387	40,000

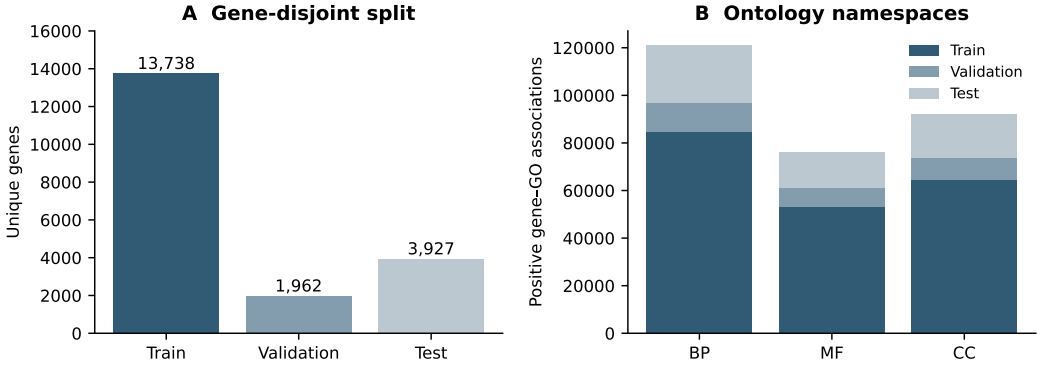


Figure 4: Gene function annotation data distribution. Left: split sizes by gene. Right: number of positive associations in the three GO namespaces; a gene may be associated with several terms, so the number of associations is not the number of genes. Biological process, molecular function, and cellular component contain 121,019, 76,056, and 91,978 de-duplicated positive associations, respectively.

B.3 Conditional Cell Generation Data

The conditional generation data are drawn from the blood, liver, lung, lymph node, spleen, muscle, and heart tissues of Tabula Sapiens, with 405,058 cells in total, and the expression input is built from decontaminated counts. Each condition is determined jointly by organ, donor, and cell type, and the model reads only organ and cell type text. The data cover 41 within-organ donor entries and 106 within-organ cell type entries. Table 8 gives the organ composition of the raw data.

The raw construction contains 452 training conditions and 114 test conditions, corresponding to 323,276 and 81,782 cells. Within the raw training part, joint training performs a global split by donor identity into 346 training conditions and 106 validation conditions, so that the same donor does not enter both sets across organs; the flow-matching supervision of the main training run uses 4,946 real cells covering all 346 conditions. Figure 5 shows the resulting condition coverage.

Validation is fixed during training and covers 106 conditions, with at most 32 reference cells sampled per condition, 2,707 in total. All comparisons fix the condition list, the reference cells, and their normalization.

B.4 Perturbation Response Prediction Data

The perturbation data integrate chemical interventions from Tahoe-100M (Zhang et al., 2025) and genetic interventions from the Replogle–Nadig Perturb-seq series (Replogle et al., 2022; Nadig et al., 2025). The chemical part covers 14 experimental plates and 50 cancer cell lines, and the genetic part covers HepG2, Jurkat, K562, and RPE1. The two sources together provide 54 within-source cell backgrounds, whose intervention designs, measured genes, and control types all differ.

Chemical conditions are organized by cell line, drug, and dose and aggregated across experimental plates, and the control is solvent-treated cells of the same cell line. Genetic conditions are organized by cell line and target gene, and the control is non-targeting cells of the same background. The upstream construction requires at least 32 cells per condition; after filtering, 62,285 conditions remain, associated with 94,263,314 single cells. The number of valid genes is 21,193 for Tahoe and

Table 8: Organ composition of the raw conditional generation data. Condition counts are for the raw training/test split.

Organ	Cells	Within-organ types	Raw conditions
Blood	65,957	8	56 / 16
Liver	21,767	18	54 / 18
Lung	65,759	29	87 / 29
Lymph node	126,598	14	70 / 14
Spleen	66,881	16	80 / 16
Muscle	41,820	16	80 / 16
Heart	16,276	5	25 / 5

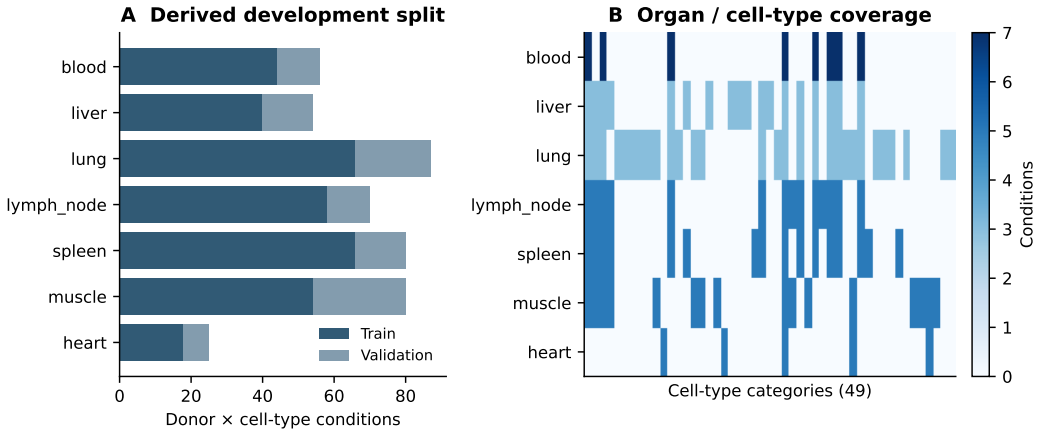


Figure 5: Conditional generation splits used in joint training. Left: number of training and validation conditions per organ, 346 and 106 in total. Right: organ–cell type combinations covered by these 452 conditions, with color indicating the number of conditions; cell type names are de-duplicated across organs, giving 49 classes. The figure does not include the 114 raw test conditions.

7,989–9,064 for the genetic sources, and the observation mask of each source is retained outside the unified output coordinate.

Joint training constructs a new grouping split from the upstream condition table. The grouping key is composed of the data source and the intervention identity, so that different doses of the same drug within one source, cell line conditions, or multiple background conditions of the same target gene fall into one group. After name mapping and the exclusion rules for external evaluation, 3,954 conditions do not enter joint training or validation, and the remaining 55,438 training conditions and 2,893 validation conditions are used for this training round. Table 9 gives the split sizes and the number of training target cells.

For each training condition the main training run deterministically selects 4 distinct real target cells, 221,752 in total; each training background caches at most 32 control cells. Figure 6 summarizes the two sources. The validation subset used for training monitoring contains 132 conditions, of which 100 are chemical and 32 genetic, with 32 reference target cells per condition, 4,224 in total.

The external H1 hESC perturbation evaluation uses a different cell background and a different supervision setting; its 80/20/50 training, validation, and test intervention split is independent of the data records above.

B.5 Cell Type Annotation Data

The cell type annotation corpus is built from a frozen CELLxGENE source list. The 189 source data files contain 44,729,647 observations in total, and 31,056,955 cells remain after filtering for human, primary-sample, and RNA modality. Cell types are mapped to 548 canonical labels, and 5,000,000 training cells are then drawn from the filtered cell set, distributed over 152 data shards.

Table 9: Perturbation condition splits and the actual number of training target cells.

Source	Training conditions	Validation conditions	Excluded conditions	Training target cells
Tahoe	48,122	2,510	3,924	192,488
Replogle–Nadig	7,316	383	30	29,264
Total	55,438	2,893	3,954	221,752

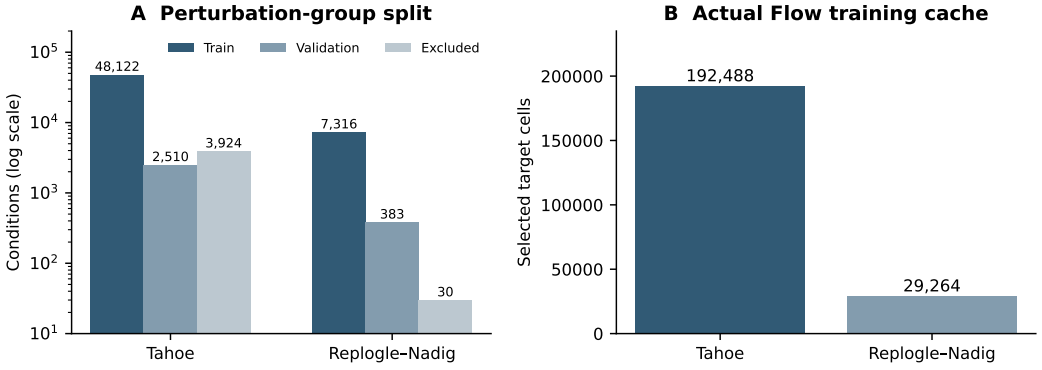


Figure 6: Perturbation data distribution. Left: number of training, validation, and excluded conditions for the two sources, on a logarithmic vertical axis. Right: number of real target cells actually drawn during joint training. The figure shows condition coverage and training sampling and does not count the roughly 94.26 million upstream cells as actual training samples.

To control the dominance of common cell types, sampling first sets a per-type cap of about 17,263–17,264, then allocates source quota in proportion to the square root of source capacity, and uses a fixed random seed to make sampling reproducible. This mechanism preserves usable cells of small classes while constraining the size of large ones.

A nested subset of 500,000 cells is also constructed as a scale reference, with a per-type cap of about 1,036–1,037. For groups that need truncation, representative selection uses a 50-dimensional PCA representation computed on 3,000 highly variable genes and is combined with hash sampling; the subset consists of 349,456 central representatives, 87,364 random supplements, and 63,180 cells kept in full from small groups. The joint main training run uses the complete 5M set.

Both constructed sets are used for training. External cell type annotation uses a separate candidate evaluation split, in which the validation and test splits of the Fine setting contain 200 cells each. Figure 7 shows the corpus construction and the per-type representation of the 548 labels.

B.6 Downstream Data

The downstream evaluation data of the four tasks relate to the training corpora in different ways. Gene function annotation and conditional cell generation are split by identity within the same data source, whereas cell type annotation and perturbation response prediction use external evaluation sets from independent sources. We describe the downstream data of each task and how it is isolated from the training corpus below.

Gene Function Annotation. Downstream evaluation uses the same GOA annotation snapshot and GO ontology version as training and is performed on the split made by gene identity: the multi-label form uses all 6,741 questions and the binary form uses 20,000 questions, with the sigmoid threshold fixed at 0.5. Splits are made by gene, so different candidate combinations of one gene cannot appear on both the training and the evaluation side. Both sides share the same knowledge-base snapshot and candidate construction rules.

Conditional Cell Generation. Downstream evaluation is drawn from the same seven-organ Tabula Sapiens corpus. Training uses 346 donor-disjoint conditions, the validation set is fixed to 106 conditions, and reference cells are sampled at most 32 per condition; the 114 test conditions of the raw

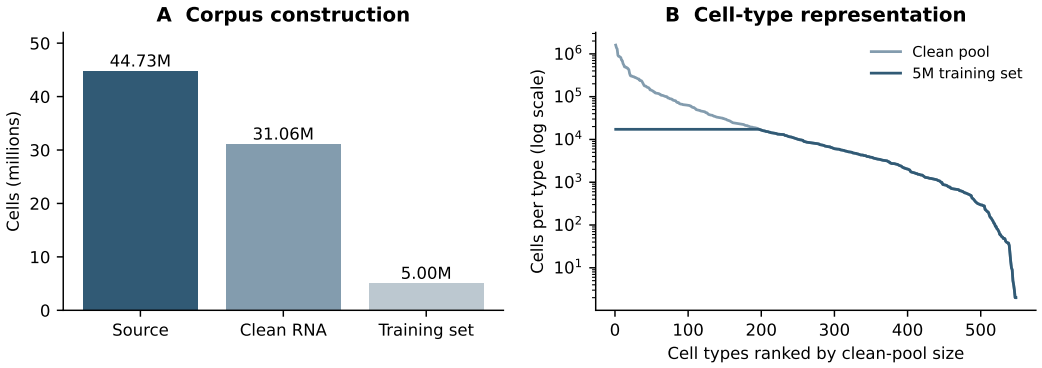


Figure 7: Cell type annotation data distribution. Left: number of cells in the raw sources, the filtered cell set, and the final training set. Right: the 548 labels sorted by available scale, comparing the filtered cell set with the final sampling quota on a logarithmic vertical axis, which shows the truncation of large classes and the retention of small ones.

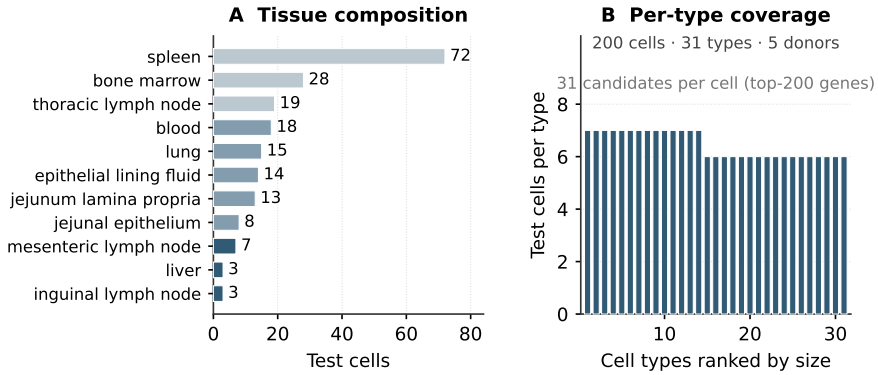


Figure 8: Downstream test set for cell type annotation. Left: tissue composition of the test cells. Right: number of cells of the 31 cell types in the test set.

construction are kept separate and take part in neither training nor the validation split. Evaluation tests distribution transfer across donors under the same organ and cell type condition.

Cell Type Annotation. Downstream evaluation uses our in-house fine-grained test set for the cell type annotation task, which contains 200 test cells, 31 candidate classes, and 5 donors. Each cell is represented by the top 200 gene symbols ranked by expression, and the candidates cover all 31 classes. This test set and the 5-million-cell pretraining corpus come from different CELLxGENE source lists; corpus construction isolates them by study and collection, and a contamination check is performed for this benchmark. Figure 8 shows the tissue composition and the per-type coverage of this test set.

Perturbation Response Prediction. Downstream evaluation uses two independent datasets according to the intervention type, both from sources different from the pretraining corpus. Genetic intervention uses the H1 hESC test set of the Virtual Cell Challenge, which contains 50 interventions and 61,587 real cells that pass quality control, and scores on the intersection of the model output coordinate with the genes measured in this dataset (17,541 genes). Chemical intervention uses the standard condition split of SciPlex3, which provides 1,595 training conditions and 216 validation conditions, while another 442 test conditions remain unopened; evaluation is performed on the validation set, and the three cell lines are scored independently to prevent the perturbation discrimination metric from being inflated by cell line identity. The pretraining corpus consists of the chemical interventions of Tahoe-100M and the genetic interventions of the Replogle–Nadig series and does not overlap in data source with either evaluation set. Figure 9 shows the scale of the two evaluation sets.

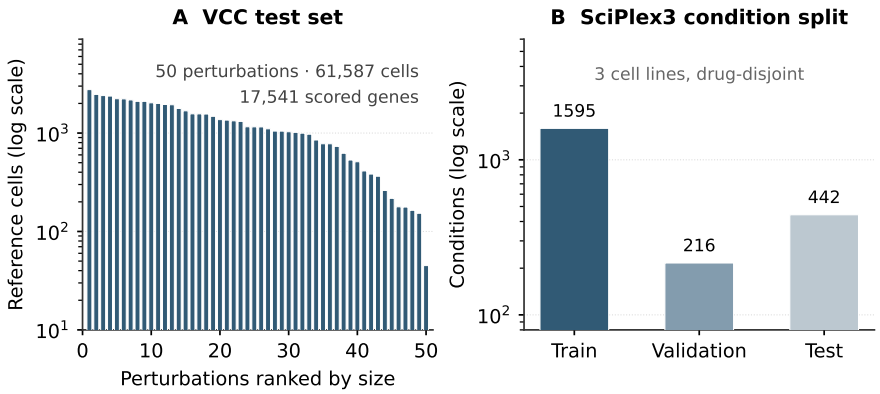


Figure 9: Downstream data for perturbation response prediction. Left: number of reference cells for the 50 interventions of the VCC test set, sorted by scale. Right: the standard condition split of SciPlex3, with the test conditions left unopened.

Both intervention types are reported in zero-shot and fine-tuned settings. The zero-shot setting does not touch the corresponding evaluation set; the fine-tuned setting adapts on the 80 training interventions of VCC and the 1,595 training conditions of SciPlex3, respectively. Both settings use the same test split and the same set of metrics.

C BASELINE

The compared methods are organized by task and correspond one to one with the entries of the result tables in the main text. Candidate labels, gene coordinates, reference cells, and data splits are kept identical across methods, so that the comparison isolates modeling differences from protocol differences. Methods that require an additional annotation head, control cells, or domain priors are noted with their input conditions at the corresponding position. We describe the input form and the comparison boundary of each baseline for the four tasks below. Table 10 summarizes the capability coverage of the model families involved.

Table 10: Capability comparison of representative related work and CellUMM. ✓* marks capabilities that require an added classification head, condition encoder, or downstream adaptation rather than being native model outputs. The rows "cell annotation", "gene annotation", "conditional generation", and "perturbation transition" correspond to cell type annotation, gene function annotation, conditional cell generation, and perturbation response prediction in this paper. The last two rows are restricted to numerical cell distributions that the model can directly output and that can be evaluated, i.e., expression matrices with cells as rows and genes as columns, or samples in the native count space; text, tokens, embeddings, and single classification outputs do not qualify. Column headers are abbreviated: *Text cond.* = text condition, *Native num. output* = native numerical output, *Cell anno.* = cell annotation, *Gene anno.* = gene annotation, *Cond. gen.* = conditional generation, *Pert. transition* = perturbation transition.

Model family	Text cond.	Native num. output	Cell anno.	Gene anno.	Cond. gen.	Pert. transition	Single model
scBERT / Geneformer / scGPT / scFoundation / UCE	×	×	✓*	×	×	×	×
CellTypist	×	×	✓	×	×	×	×
LangCell	✓	×	✓*	×	×	×	×
Cell2Sentence / CellWhisperer	✓	×	✓*	×	×	×	×
C2S-Scale / CellTok	✓	×	✓	×	✓*	✓*	✓
InstructCell	✓	×	✓	×	✓*	×	✓
scVI / scGen	×	✓	×	×	✓*	✓*	×
CPA / biolord / GEARS	×	✓	×	×	✓*	✓	×
CFGen	×	✓	×	×	✓	×	×
CellFlow / STATE	×	✓	×	×	✓	✓	×
CellUMM (ours)	✓	✓	✓	✓	✓	✓	✓

C.1 Gene Function Annotation

Gene function annotation baselines cover term-based methods and biomedical text encoders. TF-IDF builds term features from gene identifiers and GO candidate text and trains a linear classifier, which tests how discriminative the text co-occurrence signal itself is and serves as a lower-bound reference that excludes any contribution of semantic modeling. BioBERT and PubMedBERT are pretrained biomedical text models and are used in this task as paired text encoders: gene text and candidate terms are encoded pairwise to output an association probability, and the 32 pair scores are concatenated in candidate order. The two compare the representations induced by domain pretraining corpora and vocabularies.

All methods are trained and evaluated on the same gene split, uniformly use the multi-label and binary output forms, and fix the sigmoid threshold at 0.5. TF-IDF builds term features on the same gene and GO candidate text; BioBERT and PubMedBERT encode gene and candidate terms pairwise, and the 32 pair scores are concatenated in candidate order. PubMedBERT currently has no result under the same protocol and is left empty in the figures. Generative-explanation metrics and candidate-classification metrics belong to different task forms and are not ranked together.

C.2 Conditional Cell Generation

Conditional cell generation baselines include scVI, CFGen, SAVE, InstructCell, and the same-organ, same-cell-type mean of Table 4. scVI characterizes the expression distribution with a conditional variational autoencoder and provides a statistical reference for conditional latent-variable generation; the way it uses expression references and categorical covariates is described separately from purely text-conditioned inputs. CFGen and SAVE are both based on flow matching: the former generates discrete RNA/ATAC counts and supports multi-attribute conditioning, while the latter models multi-condition generation through a conditional Transformer and gene-block attention. InstructCell provides a reference for instruction-driven cell generation, and its quantitative output is decoded by a conditional variational autoencoder.

The same-organ, same-cell-type mean is the key baseline for separating two capabilities: it is close to optimal on the mean expression error, but its high-variable-gene set consistency is zero, which shows that a single mean vector cannot express the cell heterogeneity within a condition. This baseline therefore serves to determine whether the gain of a model comes from distribution modeling or from fitting the conditional mean.

Comparisons fix the organ and cell type condition and convert the output of each method to the same observed genes and the $\log 1\mathrm{p}(\mathrm{CP10K})$ coordinate.

C.3 Perturbation Response Prediction

The baselines of perturbation response prediction are CellFlow and STATE, both evaluated in zero-shot and fine-tuned supervision settings. CellFlow models the transition from the control distribution to the perturbation distribution with flow matching and optimal transport; the state transition module of STATE takes a control cell set and an intervention label as input and predicts the perturbed cell population.

Both settings use the same evaluation conditions, gene coordinates, and metrics: the zero-shot setting does not touch the target dataset, and the fine-tuned setting adapts on the target data, so the two are summarized separately. Genetic and chemical interventions differ in intervention design, measured genes, and control construction, so their results are reported as separate blocks of Table 5. All comparisons fix the control cells and the intervention split, so that differences come from state-transition modeling itself.

C.4 Cell Type Annotation

Cell type annotation baselines are divided into three groups by method form, and all are compared under the same candidate set and the same evaluation split. The cell–language model group includes LangCell, CellWhisperer, C2S-27B, CellTok, and InstructCell. Given the same candidate labels, each method keeps its native matching or answer scoring: representation-alignment models rank by

cross-modal similarity, and language models produce predictions from candidate answer probabilities or their standard annotation interface. All outputs are mapped to the same canonical labels, so that differences in synonymous names are not counted as classification errors.

The single-cell foundation model group includes Geneformer and STATE SE. Neither provides an annotation interface originally, so they are used in this task in a candidate-matching manner: frozen representations serve as input and discrimination is performed over the same candidate set. The general-purpose large language model group includes GPT-OSS-120B, DeepSeek-V4-Pro, Qwen3.5-9B-Thinking, and Qwen3.5-27B-Thinking, all of which reason directly from a candidate list in the prompt, and serves to test how far general language and reasoning ability transfers to cell type discrimination.

D METRIC

D.1 SEMANTIC DISCRIMINATION METRICS

Cell type annotation and gene function annotation share one set of thresholded discrimination metrics. Let the number of evaluation samples be N and the number of classes be N_c, and let the true positives, false positives, and false negatives of class c be TP_c, FP_c, and FN_c. All metrics are computed on the same candidate set and the same evaluation split and are averaged with equal weight per class, so that frequent classes do not dominate the overall result.

Accuracy is the fraction of samples whose prediction agrees with the ground truth. In the binary form, the ground truth y_i and the prediction $\hat{y}_i$ are both scalars:

$$\mathrm{Acc} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\hat{y}_i = y_i\right]. \tag{16}$$

In the multi-label form, the ground truth and the prediction are K-dimensional binary vectors $y_i, \hat{y}_i \in \{0,1\}^K$, and accuracy requires all K candidate positions to be correct at the same time; it is therefore also called exact-match accuracy:

$$\mathrm{EM} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\hat{y}_i = y_i\right], \qquad \hat{y}_{ik} = \mathbf{1}\left[\sigma(a_{ik}) \geq 0.5\right], \tag{17}$$

where a_{ik} is the logit of the k-th candidate of sample i, $\sigma(\cdot)$ is the sigmoid function, and the threshold is fixed at 0.5. Macro-averaged precision and recall are

$$P_{\mathrm{macro}} = \frac{1}{N_c}\sum_{c=1}^{N_c} \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c}, \qquad R_{\mathrm{macro}} = \frac{1}{N_c}\sum_{c=1}^{N_c} \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c}, \tag{18}$$

and macro-averaged F1 is the equal-weight average of the per-class F1:

$$F1_{\mathrm{macro}} = \frac{1}{N_c}\sum_{c=1}^{N_c} \frac{2\,P_c R_c}{P_c + R_c}, \tag{19}$$

where P_c and R_c are the precision and recall of class c, and terms with a zero denominator are treated as 0. When the label space is sparse, as in the multi-label gene function task, we additionally report the micro-averaged precision and recall, which pool the counts over all classes and candidates:

$$P_{\mathrm{micro}} = \frac{\sum_c \mathrm{TP}_c}{\sum_c (\mathrm{TP}_c + \mathrm{FP}_c)}, \qquad R_{\mathrm{micro}} = \frac{\sum_c \mathrm{TP}_c}{\sum_c (\mathrm{TP}_c + \mathrm{FN}_c)}. \tag{20}$$

For the binary gene function task we also report two threshold-free or class-balanced measures. Balanced accuracy averages the recall of the two classes, so that a majority class cannot dominate:

$$\mathrm{BA} = \frac{1}{2}\left(\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} + \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}\right), \tag{21}$$

and AUROC is the area under the receiver operating characteristic curve obtained by sweeping the decision threshold over the continuous association score; it is reported as a percentage, like the other metrics of this task.

D.2 Expression Distribution Metrics

Conditional cell generation compares the generated distribution with the reference distribution at the level of a condition. Let the generated cell matrix under condition c be $\hat{X} \in \mathbb{R}^{n\times G}$, the reference cell matrix be $X \in \mathbb{R}^{m\times G}$, the observed gene set be M, and $G_M = |M|$. All metrics are computed on the per-cell $\log1\mathrm{p}(\mathrm{CP10K})$ coordinate and only within M; genes outside the mask are neither counted in the error nor averaged as zeros. The scale conversion must be performed per cell before means or dispersions are computed: averaging first and converting afterwards is not equivalent to converting first and averaging afterwards.

Let $\hat{\mu}_g$ and μ_g be the cell mean of gene g on the two sides. The mean expression error is the average squared difference between them over the observed genes:

$$\mathrm{MSE}_\mu = \frac{1}{G_M}\sum_{g\in M}\left(\hat{\mu}_g - \mu_g\right)^2. \tag{22}$$

This metric characterizes the location of the conditional mean and does not reflect the dispersion within a condition. To constrain the shape of the distribution as well, we use the squared biased RBF maximum mean discrepancy (Gretton et al., 2012):

$$\begin{aligned}\hat{\mathrm{MMD}}^2 = \frac{1}{n^2}\sum_{i,j=1}^{n} k(\hat{x}_i,\hat{x}_j) + \frac{1}{m^2}\sum_{i,j=1}^{m} k(x_i,x_j) \\ - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k(\hat{x}_i,x_j)\,.\end{aligned} \tag{23}$$

The kernel is $k(u,v) = \exp(-\gamma\|u-v\|^2/G_M)$. The bandwidth is fixed per condition and does not depend on the model: the reference cells of that condition are taken, all pairwise squared Euclidean distances are computed and divided by G_M, and the median of the positive values is used as $\hat{\sigma}^2$, with $\gamma = 1/(2\hat{\sigma}^2)$. If the reference cells of a condition give no positive distance, the bandwidth falls back to the median computed on a fixed set of 128 training cells drawn with a fixed seed.

The high-variable-gene metrics follow the high-variable-gene set of the reference cells. Let H_k be the set of the top k genes ranked by expression variance, with $k \in \{1000, 2000\}$; then

$$J_k = \frac{\left|H_k(\hat{X}) \cap H_k(X)\right|}{\left|H_k(\hat{X}) \cup H_k(X)\right|}, \qquad \mathrm{MSE}_k = \frac{1}{k}\sum_{g\in H_k(X)}\left(\hat{\mu}_g - \mu_g\right)^2, \tag{24}$$

where J_k measures whether the model produces its main variation on the correct genes and MSE_k measures the quantitative deviation on those genes. The average rank is the arithmetic mean of the ranks over the six component metrics:

$$\bar{r} = \frac{1}{6}\sum_{m=1}^{6} r_m. \tag{25}$$

The component metrics do not point in the same direction ($\hat{\mathrm{MMD}}^2$ and the two MSE terms are better when lower, whereas the two J_k terms are better when higher), so the direction is unified before ranking. Aggregation weights conditions equally and does not weight by the number of sampled cells per condition; evaluation fixes 8 generated cells per condition, and reference cells are all rows in the frozen evaluation package, capped at 32 per condition. Different generation counts or reference caps constitute independent protocols and are summarized separately.

D.3 Perturbation Response Prediction Metrics

Perturbation response prediction is carried out under a frozen evaluation interface. The predicted expression, the real treated expression, and the real control are compared on the same gene set, the input is log-normalized expression, and the target total is 10^6.

Perturbation discrimination ranks the predicted effect against the distance to each real intervention effect (Roohani et al., 2025). Let e_p be the real effect of the p-th intervention and $\hat{e}_p$ the predicted

Table 11: Shape conventions of the main tensors.

Symbol	Shape	Meaning
H	$B \times L_{\text{seq}} \times 512$	Per-gene contextual features from the frozen cell encoder
M	$B \times L_{\text{seq}} \times 512$	Memory of the resampler, obtained from H by a linear projection
Q	24×512	Learnable queries of the resampler (batch-independent)
z	$B \times 24 \times 512$	Continuous cell programs, i.e., latent cell programs
u	$B \times 24 \times 3584$	Representation of a cell slot at the backbone input layer
v	$B \times 24 \times 512$	Predicted velocity from the velocity head
$\hat{x}$	$B \times G$	Expression vector from the decoder, with $G = 21{,}523$

effect of the model for that intervention; then

$$\begin{aligned} \text{PDS} &= \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \left(1 - \frac{\text{rank}_p - 1}{|\mathcal{P}| - 1}\right), \\ \text{rank}_p &= 1 + \sum_{q \neq p} \mathbf{1}\left[d(\hat{e}_p, e_q) < d(\hat{e}_p, e_p)\right], \end{aligned} \tag{26}$$

where $d(\cdot, \cdot)$ is the cosine distance, $\mathcal{P}$ is the set of evaluated interventions, the rank denominator is $|\mathcal{P}| - 1$, tied ranks are handled by average ranking, and the target gene itself is excluded within the gene panel of that dataset. This metric measures whether the model identifies the specific intervention, beyond reproducing the overall expression change.

Let $\mathcal{S}_{\text{pred}}$ and $\mathcal{S}_{\text{ref}}$ be the sets of significantly differentially expressed genes of the prediction and the reference, and $\mathcal{D}$ the set of genes with the same direction; then

$$\text{yield} = \frac{|\mathcal{S}_{\text{pred}} \cap \mathcal{S}_{\text{ref}} \cap \mathcal{D}|}{\max\left(|\mathcal{S}_{\text{pred}}|, |\mathcal{S}_{\text{ref}}|\right)}. \tag{27}$$

This metric constrains over-prediction and under-coverage at the same time. Directional reach characterizes the longest ranked prefix of the prediction whose directional agreement rate reaches 0.9:

$$\text{reach} = \frac{1}{|\mathcal{S}_{\text{ref}}|} \max\left\{k \ : \ \frac{|\mathcal{D} \cap \text{top}_k|}{k} \geq 0.9\right\}, \tag{28}$$

where top_k denotes the top k genes ranked by predicted effect strength. The Jaccard similarity of the significant gene sets is

$$J_{\text{DE}} = \frac{|\mathcal{S}_{\text{pred}} \cap \mathcal{S}_{\text{ref}}|}{|\mathcal{S}_{\text{pred}} \cup \mathcal{S}_{\text{ref}}|}. \tag{29}$$

Differential expression uses the Wilcoxon rank-sum test with a corrected p-value threshold of 0.05, FDR correction is performed per intervention, and genes are ranked by absolute $\log_2$ fold change.

E Model

E.1 Notation and Tensor Shapes

This section gives the concrete structure, computation flow, and parameter settings of the modules in Section 4. Table 11 summarizes the shape conventions of the main tensors, where B is the batch size, L_{seq} is the length of the expression gene-token sequence, and G is the dimension of the unified gene coordinate. The condition c of the main text decomposes into biological attributes b, covering cell type, tissue, disease, intervention, dose, time, and modality, and a semantic description s, covering natural-language statements, ontology definitions, functional knowledge, and task instructions; attributes may also be given as text, so the boundary between the two is not strict.

E.2 Expression Encoding and Tokenization

An expression profile is sorted by descending gene expression and converted into a sequence of gene tokens, with a maximum length of 2,048; longer sequences are truncated and shorter ones are padded with 0. The sorting key is the ratio between a gene count and the median of that gene over

the full gene panel. Because the within-cell library size is a constant, this key gives exactly the same ordering as the original Geneformer definition $\mathrm{count}/\mathrm{total} \times 10^4/\mathrm{median}$. Duplicate source gene identifiers are de-duplicated after merging, so that one gene does not occupy several positions in the sequence.

The sequence is passed to Geneformer-12L-30M, and the last-layer hidden state is taken as the per-gene contextual feature with an output dimension of 512. This encoder remains frozen in all training stages and receives no gradient. The training pipeline caches the continuous programs of already-encoded cell blocks, and the cached path and the online encoding path produce exactly the same representation.

E.3 Continuous Cell Program Resampler

The resampler compresses a variable-length feature sequence into 24 continuous units. The memory is obtained by a linear projection $M = W_m H$, the learnable queries Q are initialized from a normal distribution with standard deviation $512^{-1/2}$, the stacking depth is 2, the number of attention heads is 8, and dropout is 0. Each resampler block performs three operations in order: multi-head cross-attention to the memory (with a padding mask applied on the memory side), multi-head self-attention among the queries, and a two-layer position-wise feed-forward transformation. All three use pre-layer normalization and residual connections, and the output is passed through one layer normalization to obtain z.

The dimensions of the resampler serve two goals. The 24 units share a hidden dimension of 512 with language tokens, so a cell block can take part in the shared attention computation without an additional projection. The feed-forward hidden width is 4×512, consistent with a standard Transformer, which makes the capacity comparable with that of the language backbone.

E.4 Interleaved Attention and Position Numbering

The attention visibility of the interleaved sequence is composed of three parts. Text positions keep the standard causal mask, so position i can access only $j \le i$. Slots that belong to the same continuous cell block are released from the causal restriction and can access each other, so that one cell block forms a single random variable. Padding positions are masked in both the query and the key direction. The three parts are merged into an additive mask of shape $B \times 1 \times S \times S$, where S is the sequence length.

Position numbering follows a block-level compression rule. Text positions increase by the count of valid tokens. When a position and its predecessor belong to the same cell block, its position number does not increase, so the whole 24-slot cell block shares one rotary position. This rule guarantees that the first text position after a cell block differs by only 1 from the positions before the cell block, which preserves the relative position structure on the text side. Padding positions are all assigned the number 0 and are excluded by the attention mask.

E.5 Parameterization of the Boundary Markers

Low-rank adaptation does not update the word embedding matrix and the output head of Qwen2.5-7B, so the newly registered boundary markers cannot obtain useful representations from the vocabulary. CellUMM therefore introduces two dedicated modules: an input-side embedding table $E_b \in \mathbb{R}^{2\times 3584}$ and an output-side linear head $W_b \in \mathbb{R}^{2\times 3584}$. In the forward pass, the positions of <BOC> and <EOC> are located by token identifier, and the corresponding rows of the word embedding are replaced by the output of E_b; at the output end, the logit components of these two symbols are replaced by the output of W_b.

The replacement does not change the vocabulary size. The two modules have about 14k parameters in total, which is negligible relative to the 7B backbone; the small size lets the boundary decision converge independently of the vocabulary and avoids training a vocabulary matrix of hundreds of thousands of entries for two new symbols.

E.6 Expression Decoder

The decoder maps the 24 continuous units back to a gene expression vector and consists of attention pooling followed by a low-rank linear mapping. The pooling weights come from a bias-free linear score normalized by $\mathrm{softmax}$; the 24 units are summed with these weights into a single context vector, which is then passed through layer normalization, a $512 \to 1{,}024$ linear mapping, a SiLU activation, and a $1{,}024 \to 256$ linear mapping to obtain the gene factor coefficients κ. The final output multiplies the coefficients with the gene factor matrix $F \in \mathbb{R}^{21523\times 256}$, adds the gene bias b, and applies $\mathrm{softplus}$ to guarantee non-negativity:

$$\hat{x} = \mathrm{softplus}\big(\kappa F^{\top} + b\big)\,. \tag{30}$$

F is initialized from a normal distribution with standard deviation $256^{-1/2}$, and b is initialized to zero. Writing the output as a linear combination of 256 factors makes the number of decoder parameters grow linearly with the number of genes; the factor dimension also imposes an explicit capacity constraint, since the model can represent covariance structure across genes only within a low-dimensional factor subspace. This is also why the distribution dispersion needs to be evaluated separately.

The decoder is pretrained to reconstruct expression on the $\log 1\mathrm{p}(\mathrm{CPM})$ scale, freezing the cell encoder and the language backbone and using a mean squared error. It is then calibrated on the output of the training data and kept frozen during joint training.

E.7 Flow-Matching Objective and Sampler

During training, a time $t \sim \mathcal{U}(0,1)$ and noise $\varepsilon \sim \mathcal{N}(0, I)$ are sampled independently for each generated block, an intermediate state $z_t = (1-t)\,\varepsilon + t\,z_1$ is constructed along a straight path, and the regression target is the displacement $v^* = z_1 - \varepsilon$. The velocity head is a linear mapping $W_f : \mathbb{R}^{3584} \to \mathbb{R}^{512}$ applied to the hidden state of the backbone at the corresponding position. The loss is the mean squared error on the generated block, and the observed block does not take part. The time embedding uses a sinusoidal form and is added to the slot embedding in the input representation of a cell slot, so different blocks within one sequence can carry different times.

Inference starts from standard normal noise and integrates $\mathrm{d}z/\mathrm{d}t = v_\theta(z, t; c)$ over $[0,1]$ with the Heun method. Heun is a second-order method and requires two velocity field evaluations per step, so an integration with N_{step} steps needs $2N_{\mathrm{step}}$ evaluations in total. All experiments in this paper fix the number of steps to 8, i.e., 16 evaluations; this value trades generation fidelity against inference cost and is not tuned per condition. After sampling, the generated continuous programs are mapped back to the gene expression space by the frozen decoder, and repeated sampling of the initial noise yields an empirical approximation of the conditional distribution.

E.8 Output Heads and Loss Functions of the Four Tasks

Semantic Output Heads. The semantic tasks predict from shared hidden states. The candidate ranking head of cell type annotation projects the mean-pooled representation of the cell programs and the frozen word embeddings of the candidate names into 256 dimensions each and applies L_2 normalization, then builds a categorical distribution from their cosine similarity divided by a temperature of 0.07; the cross-entropy on this distribution uses label smoothing of 0.05 and a weight of 0.5 in the total loss. Gene function annotation takes the hidden state at the last non-padding text position of the sequence and outputs candidate logits through a single-layer linear head, with 32-dimensional and 1-dimensional outputs for the multi-label and binary forms, respectively.

Quantitative Output Head. The quantitative tasks share one velocity head, and the supervision of conditional generation and perturbation response prediction both falls on velocity regression. The expression decoder and its gene factor matrix are frozen components and map generated programs back to gene coordinates only at the end of sampling, so the outputs of the two quantitative tasks fall in the same gene space.

Cell Type Annotation Loss. The valid tokens of the answer suffix are supervised by autoregressive cross-entropy, with an additional candidate ranking loss: $\mathcal{L}_{\mathrm{anno}} = \mathcal{L}_{\mathrm{ce}} + 0.5\,\mathcal{L}_{\mathrm{rank}}$. The ranking distribution is obtained from the cosine similarity between candidate names and the cell program representation with a temperature of 0.07, and its cross-entropy uses label smoothing of 0.05.

Table 12: Main hyperparameters of joint training.

Configuration	Value	Note
Number of cell program units	24	Resampler output length
Continuous program dimension	512	Matches the hidden dimension of language tokens
Backbone hidden dimension	3,584	Qwen2.5-7B-Instruct
Unified gene coordinate	21,523	Decoder output dimension
Gene factor rank	256	Dimension of the decoder low-rank factorization
Low-rank adaptation	Rank 16, scaling 32, dropout 0.05	A single set, shared by the four tasks
Number of synchronous replicas	8	Data parallelism
Number of training updates	31,250	Covers the 5M-cell annotation set once
Batch per replica	Annotation 20 / gene 4 / generation 2 / perturbation 4	One batch for each of the four tasks
Learning rate	10^{-5} (adaptation) / 5×10^{-5} (interfaces)	AdamW, weight decay 0.01
Learning rate schedule	Warmup for 200 steps, then cosine decay to 0.1	—
Gradient clipping	1.0	Clipped after averaging across replicas
Random seed	42	—
Sampling steps	8-step Heun	Two velocity field evaluations per step

Gene Function Annotation Loss. $\mathcal{L}_{\text{gene}}$ is the binary cross-entropy on candidate logits, with positive-class weight 3 in the multi-label form and no weighting in the binary form.

Shared Flow-Matching Loss. Conditional cell generation and perturbation response prediction share one velocity field and one loss, $\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{pert}} = \mathcal{L}_{\text{flow}}^{\mathcal{T}}$ (Equation 7); the loss acts only on the generated slots, and the time of the observed control block in perturbation response prediction is fixed to 1 so that it participates in attention as context only.

Latent Objective. Training uses no loss in the expression space. Direct regression on $21{,}523$-dimensional sparse expression would be dominated by zeros, and the decoder already compresses expression into a 256-dimensional factor subspace, so such a loss would add no extra degrees of freedom. The cost is that a nonlinear mapping separates the training objective from the evaluation metrics: a small velocity-regression error does not necessarily imply good distribution metrics, and the capacity constraint of the decoder may narrow the generated distribution. Dispersion calibration of the decoder is left to future work.

Joint Objective. Each optimization step draws one batch for each of the four tasks, and the joint objective is the arithmetic mean of the four losses (Equation 11), whose weights are not adjusted to the loss scale.

E.9 Parameter Groups and Training Configuration

The training parameters are divided into a frozen and a trainable group. The frozen group contains the Qwen2.5-7B base weights, the cell encoder, the resampler, and the calibrated expression decoder. The trainable group contains one shared low-rank adaptation, the cell projection, the time and slot embeddings, the continuous velocity head, the boundary marker interfaces, the candidate ranking head, and the two gene function annotation heads. All trainable parameters are kept in single precision, and the backbone forward pass uses mixed precision. Table 12 lists the main hyperparameters.

Inference uses the same parameter groups and needs no per-task switching. The generation tasks start from standard normal noise, integrate the velocity field over $[0, 1]$ with 8-step Heun, and obtain cell samples through the frozen decoder (Equations 8 and 7); each condition generates 8 cells, and reference cells are all rows of the frozen evaluation package, capped at 32 per condition. Repeated sampling of the initial noise yields an empirical approximation of the conditional distribution.

F Related Work

This appendix gives the detailed version of Sections 2.1–2.2 and lists, for each model, its pretraining objective, output interface, and comparison boundary in the tasks of this paper. Table 13 maps the subsections of the main text to the appendix parts that expand them.

F.1 Biomedical Language Models for Gene-Level Semantics

Biomedical text pretraining provides a mature representation basis for gene-level semantics. BioBERT continues domain-adaptive pretraining on top of BERT using PubMed abstracts and PMC full texts and clearly improves named entity recognition, relation extraction, and question answering

(Lee et al., 2020). PubMedBERT is pretrained from scratch on biomedical corpora with a domain vocabulary, which yields a language representation better suited to biomedical terminology and syntax than "general model plus domain fine-tuning" and improved the best results of the time on several biomedical benchmarks (Gu et al., 2021). Both differ from this work in corpus scale and are therefore used as fixed encoders without further pretraining.

The input and output of these models, however, both lie in the text semantic space: they accept no cell expression matrix and learn neither cell-state distributions, nor cell heterogeneity, nor expression changes after perturbation. They can therefore provide gene-level semantics only and cannot on their own perform the mapping from semantics to a quantitative cell state, infer a semantic interpretation from a cell state, or predict cell state change under perturbation. CellUMM places this knowledge-level semantics and quantitative cell states in one framework, so that gene semantics take part in cell-state modeling and no longer stand apart from it. The concrete use of BioBERT and PubMedBERT in gene function annotation is described in Appendix C.1.

F.2 Cell–Language Models for Multimodal Cellular Understanding

Connecting single-cell data to a language model interface has been a main trend in this direction over the past two years. Cell2Sentence (C2S) rewrites an expression profile as a sequence of gene tokens ordered by expression level and represents a single cell as one sentence, so that a general-purpose large language model can process transcriptomic data without architectural changes (Levine et al., 2023); C2S-Scale extends this paradigm to a larger single-cell corpus and a larger language model and supports perturbation-related sequence prediction (Rizvi et al., 2025). CellWhisperer takes a contrastive alignment route and learns a joint embedding on large-scale transcriptome–text pairs, which supports cell retrieval, cell feature annotation, and natural-language question answering (Schaefer et al., 2025). CellTok uses a VQ-VAE to compress a cell into discrete cell tokens and adds them directly to the language model vocabulary, achieving early fusion of cells, text, and multi-cell context (Xiao et al., 2025). InstructCell introduces multimodal instruction data so that the model can perform cell type annotation, conditional pseudo-cell generation, and drug sensitivity prediction from natural-language instructions (Fang et al., 2025).

These models build an important bridge from cell data to a natural-language interface, but their output interfaces differ: C2S outputs a cell sentence, CellTok outputs discrete cell tokens, CellWhisperer outputs an embedding or a text answer, and InstructCell generates pseudo-cell expression through a dedicated decoder. They cover cell type annotation, semantic interaction, or a specific form of generation, and none of them handles gene function annotation, cell type annotation, native conditional distribution generation, and perturbation distribution transition within one continuous cell-state space. CellUMM keeps their semantic interaction capability and at the same time places the generation target back into the continuous native cell-expression space. In the comparison of this work, each method keeps its native matching or answer scoring, and the input conditions and comparison boundaries are given in Appendix C.4.

F.3 Single-Cell Foundation Models for Cellular Representation

Single-cell foundation models learn transferable cell representations through large-scale self-supervised pretraining. scBERT encodes a cell as a sequence of gene tokens and pretrains an efficient Transformer with a gene-level masked-expression objective, which transfers to cell type annotation (Yang et al., 2022). Geneformer takes rank-ordered gene sequences as input, pretrains context-aware gene and cell representations on about thirty million single-cell transcriptomes, and transfers them to cell type prediction, network biology, and gene regulation tasks (Theodoris et al., 2023). scGPT uses generative masked modeling to pretrain a unified single-cell multi-omics representation on tens of millions of cells (Cui et al., 2024). scFoundation enlarges the pretraining scale further (Hao et al., 2024), and Universal Cell Embedding attempts to learn general cell embeddings across tissues, species, modalities, and even unseen data distributions, with an emphasis on zero-shot transfer (Rosen et al., 2026). For perturbation response, the State Embedding module of STATE learns cell embeddings from large-scale observational single-cell data to characterize cell states across tissues and experimental conditions (Adduri et al., 2025).

These models provide transferable quantitative representations, but their core output is usually an embedding or a task classification result rather than a native cell distribution. Without an addi-

Table 13: Correspondence between Sections 2.1–2.2 and Appendix F.

Section	Content kept in the main text	Content added in Appendix F
2.1	Representation ability of text-space and cell–language models, and their gap on cell-level tasks	Pretraining corpus, vocabulary construction, task coverage, output interface, and native scoring of each model
2.2	Representation and generation capability of quantitative models, and their missing semantic interface	Pretraining scale, transfer objectives, perturbation modeling role, condition type, and comparison boundary of each model

tional generative decoder, an embedding cannot directly answer "which cells should be generated for a given semantic condition" or "how a control state forms a new distribution after perturbation". STATE adds a State Transition module on top of State Embedding to model perturbation response, which shows that representation learning and state transition can be combined, but its main objective remains perturbation response prediction and it does not cover semantic annotation and general conditional generation at the same time. CellUMM uses continuous cell programs as a common interface for representation and generation, so that one latent space supports both cell interpretation and, through flow matching, decoding into native cell distributions.

F.4 Probabilistic and Flow-Based Models for Cellular Generation

Probabilistic generative models provide the statistical basis for quantitative modeling of single-cell data. scVI characterizes single-cell expression distributions with a hierarchical probabilistic model and variational inference, and it incorporates technical noise, batch effects, and sequencing depth into the generative process (Lopez et al., 2018); it provides a statistical reference for conditional latent-variable generation, and the way it uses expression references and categorical covariates is described separately from purely text-conditioned inputs. CFGen uses conditional flow matching to generate discrete RNA/ATAC counts and supports conditioning on multiple attributes such as cell type and batch (Palma et al., 2025). SAVE models multi-condition single-cell generation with a conditional Transformer, gene-block attention, and flow matching (Li et al., 2026). CellFlow uses flow matching and optimal transport to model the transition from control cell distributions to perturbed cell distributions (Klein et al., 2025), and the full STATE model takes a control cell set and a perturbation label as input and predicts the corresponding perturbed cell population (Adduri et al., 2025).

These models have clear advantages in quantitative generation, but their conditions are usually structured covariates such as batch, cell type, drug, or gene perturbation. They do not unify gene function, cell type descriptions, disease semantics, and experimental text in one semantic space, and they usually do not perform gene function annotation or cell type annotation as well. They therefore strengthen the quantitative part of generation and perturbation response prediction without establishing a complete semantic–quantitative loop. In our comparisons the control cells and the intervention split are fixed across methods, so that differences come from state-transition modeling itself. CellUMM keeps their objective of modeling native cell distributions and lets the semantic representation further participate in conditional generation, cell interpretation, and perturbation state transition.