scPaLM: Pre-training of Single-Cell Language Models through Genetic Pathway Learning

Anonymous ACL submission

Abstract

As scRNA-seq expands dataset richness and advances cellular biology, transformer-based single-cell foundation models have emerged. Despite their efficacy, they face two key challenges: (1) an **efficiency challenge**—the high dimensionality of gene expression ($N_g \approx$ 20,000–60,000) leads to prohibitive $O(N_g^2)$ attention cost, and gene filtering discards housekeeping genes with essential regulatory functions; (2) an **interpretability challenge**—existing models produce dense gene-level embeddings without exposing the pathway-level functional units in which genes actually operate. Driven by these challenges, we propose a novel **S**ingle-**c**ell Pre-trained **L**anguage **M**odel via Genetic **Pa**thway Learning, named `scPaLM`, that addresses both challenges through three key innovations: ❶ **Permutation-Invariant Embedding** that condenses all genes into a reduced set of $N \ll N_g$ tokens via order-agnostic pooling; ❷ a **Genetic Pathway Learning** module that learns discrete representations through vector quantization, enabling the modeling of collective gene behaviors in a data-driven way; ❸ **Cell-level Information Aggregation** that progressively aggregates cell representations into a designated cell token $\boldsymbol{h}_C$ during training, with a tailored masking strategy and a token-level contrastive regularizer. Comprehensive evaluations across four biological benchmarks demonstrate `scPaLM`'s superiority: it achieves average 10.1% improvement in cell type annotation compared to scGPT across all datasets, 5.15% increase in drug response prediction correlation compared to scFoundation, zero-shot imputation results that beat all baselines, and the highest Pearson Δ on both Adamson and Norman perturbation benchmarks.

1 Introduction

Single-cell RNA sequencing has emerged as the state-of-the-art method for elucidating the intricacies and diversity inherent in RNA transcripts at the individual cell level and providing insights into the composition of distinct cell types and their respective functions within tissues, organs, and organisms (Jovic et al., 2022). The massive amount of data generated by scRNA-seq techniques has provided massive information on various cells, enabling a better understanding of them, and hence benefit diverse research areas such as development (Semrau et al., 2017), auto-immune diseases (Gaublomme et al., 2015) and cancer diagnosis or prognosis (Patel et al., 2014).

To effectively model the scRNA-seq data, various computational methods (Cui et al., 2024; Hao et al., 2023) with different architecture designs have been proposed. Recent progress in deep learning has inspired the usage of advanced machine learning techniques, such as transformers (Vaswani, 2017), to scRNA-seq data analysis. These models, particularly those inspired by the success of BERT (Devlin et al., 2018) GPT (Radford et al., 2019), adopt a token-based approach, treating the expression count of each gene as a "token", a concept widely embraced in natural language processing (NLP). These tokens are then assembled into a "sentence" representing the genetic expression profile of a cell. Such models have demonstrated exceptional performance in capturing the underlying structures inherent to scRNA-seq data and have consistently outperformed conventional algorithms across various downstream tasks, including cell type annotation and bulk drug response prediction (Cui et al., 2024; Hao et al., 2023; Theodoris et al., 2023; Yang et al., 2022). Despite its effectiveness, major bottlenecks remain:

Efficiency Challenge. The high dimensionality of gene expression data poses significant computational challenges. Each cell is expressed with tens of thousands of genes ($N_g \approx 20,000$–$60,000$), and treating each gene as a distinct token leads to prohibitive $O(N_g^2)$ attention complexity (Cui et al., 2024). Existing approaches address this through

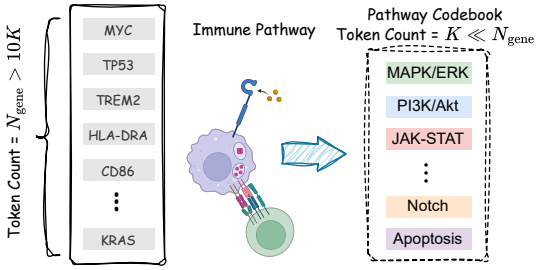

Figure 1: **Pathway codebook enables token-efficient learning.** Existing single-cell foundation models treat each of $N_g \approx$ 20–60k genes as a distinct token, incurring $O(N_g^2)$ attention. Biologically, however, related genes (e.g., *TREM2*, *HLA-DRA*, *CD86* in immune response) act jointly as functional *pathways*. `scPaLM` replaces gene-token tokenization with $N = 256$ permutation-invariant patch tokens, then quantizes them against a learned codebook $\mathcal{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K\}$ of $K = 128$ entries, producing pathway-aligned discrete codes that are both compute-efficient (99.6% token reduction) and biologically interpretable.

gene filtering, selecting only highly variable genes (e.g., top 2,000), which substantially reduces computational cost but inevitably discards valuable biological information (Joshi et al., 2022). Essential genes like housekeeping genes may not exhibit high variability while maintaining pivotal regulatory functions, making their exclusion problematic for comprehensive cellular modeling.

Interpretability Challenge. Beyond computational efficiency, current foundation models lack mechanisms to provide biologically meaningful intermediate representations. Existing approaches produce dense, gene-level embeddings that are difficult to interpret in terms of known biological processes (Cui et al., 2024; Hao et al., 2023). However, genes do not function in isolation—they operate in coordinated functional units known as *pathways* to execute cellular processes (Alexa et al., 2006; Mi et al., 2013). This pathway-level organization provides a natural abstraction that could simultaneously reduce dimensionality and enhance interpretability, yet remains largely unexplored in current single-cell foundation models.

In this paper, we propose `scPaLM`, a transformer-based foundation model that addresses both efficiency and interpretability challenges through a unified pathway-centric architecture. Our contributions are summarized as follows:

- We propose `scPaLM`, a foundation model that addresses the efficiency-interpretability trade-off in single-cell modeling through pathway-centric tokenization, achieving 99.6% token reduction while maintaining biological interpretability.
- We introduce three key innovations: (1) permutation-invariant embedding with symmetric pooling for scalable gene representation, (2) genetic pathway learning via vector quantization that discovers interpretable pathway-like structures, and (3) cell-level aggregation with contrastive learning for discriminative cell representations.
- Comprehensive experiments evaluate `scPaLM` across cell annotation, drug response prediction, zero-shot imputation, and perturbation prediction, demonstrating efficient transferable representations across diverse biological settings.

2 Methodology

We use "language model" here in the masked-reconstruction sense: `scPaLM` is a transformer pre-trained with self-supervised masked reconstruction over a learned discrete vocabulary of pathway tokens, rather than an autoregressive next-token model.

Overview. A scRNA-seq dataset is usually stored as a matrix $X_{\text{raw}} \in \mathbf{N}^{N_c \times N_g}$, where N_c is the number of cells and N_g is the number of genes. In X, each row represents the expression values of genes in a cell. Following common single-cell preprocessing practice (Luecken et al., 2022), a per-cell library-size normalization followed by `log1p` is applied on X_{raw} to obtain X, and we follow this processing paradigm as well. For clarity and ease of reference, we summarize the key notations used throughout this work in Table 9.

The main components of `scPaLM` are transformers (Vaswani, 2017). To facilitate the learning process, we propose a novel embedding process (§2.1), referred to as $\texttt{embed}(\cdot)$, that maps each row of X into N tokens. Note that our embedding process can produce a reduced amount of tokens, which is more memory- and computation-efficient compared to existing works which usually need to construct a large number of tokens (typically equals to N_g). The tokens are then fed into the transformers in `scPaLM`, which leverage the attention mechanism to capture the interaction between tokens and learn the biological implications behind the scRNA-seq data. The transformers have multiple layers and process the token representations sequentially with the following formula: $\boldsymbol{h}_i = \text{Layer}_i(\boldsymbol{h}_{i-1})$, where $\boldsymbol{h}_i$ indicates the token representations from i-th

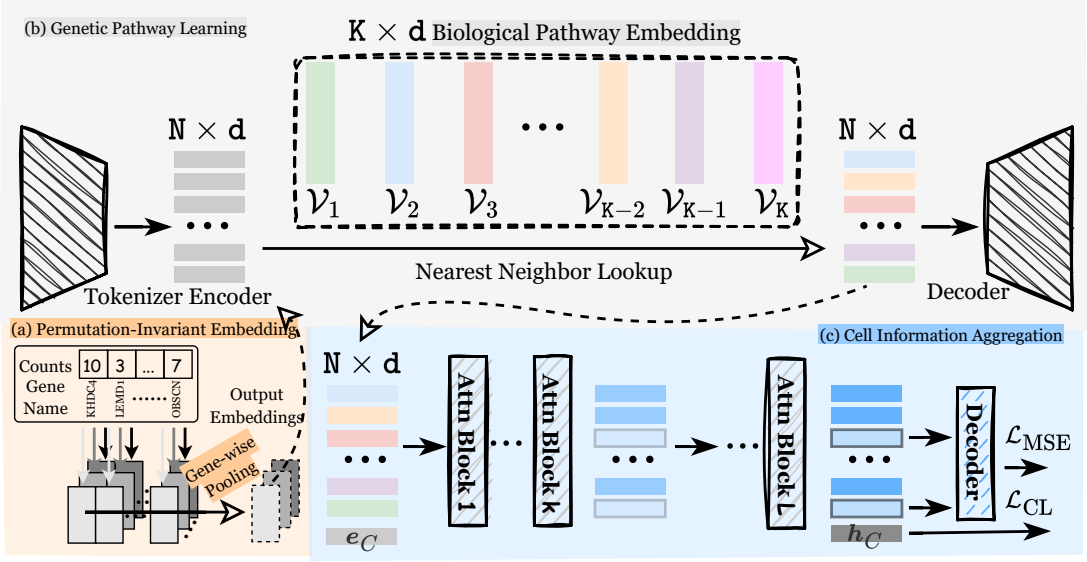

Figure 2: Overview of **scPaLM**. Three components are introduced: (i) an order-agnostic embedding process that condenses N_g gene expression values into $N \ll N_g$ patch tokens; (ii) a genetic pathway learning module that discretizes patch tokens against a learned codebook $\mathcal{V}$; (iii) a cell-information aggregation stage that summarizes pathway tokens into a designated cell token $\boldsymbol{h}_C$ via masked reconstruction and a token-level contrastive regularizer.

transformer layer.

The training process for **scPaLM** consists of two distinct stages. *During the initial stage*, we train both an encoder and a decoder using a reconstruction loss. Specifically, the encoder is trained to map the raw gene tokens to tokens that represent genetic pathways (as discussed in §2.2), while the decoder's role is to reverse this mapping, converting genetic pathway tokens back to the original expression levels. *In the second stage*, we train another encoder with an additional token designed to capture cell-specific information (as discussed in §2.3). The two encoders we have trained in these two stages collectively empower various downstream tasks, exhibiting superior performance.

2.1 Permutation-Invariant Embedding

Pathway-Inspired Patch Construction Drawing inspiration from Vision Transformers (Dosovitskiy et al., 2020), we reconceptualize gene patches as functional units analogous to biological pathways. However, unlike image pixels with spatial coherence, genes lack inherent positional relationships. Direct application of ViT-style patching introduces order sensitivity—permuting gene order alters patch composition, contradicting biological reality where expression vectors are permutation-invariant. To resolve this, we develop a symmetric embedding architecture that dynamically clusters genes into N pathway-aligned patches ($N \ll N_g$) through order-agnostic operations.

Symmetric Embedding Pipeline As detailed in Algorithm 1, our pipeline enforces permutation invariance through three key stages. First, each gene's scalar expression is lifted into a d-dimensional latent via a per-gene projection $\boldsymbol{P}_1 \in \mathbb{R}^{N_g \times 1 \times d}$, then refined by an intra-gene MLP weight $\boldsymbol{P}_2 \in \mathbb{R}^{d \times d}$ and rescaled by the gene-specific 2D coefficients $\boldsymbol{\alpha} \in \mathbb{R}^{N_g \times d}$. A hierarchical pooling strategy then aggregates these embeddings: zero-expression genes are replaced by a learnable embedding and pooled separately, then combined with active genes through concatenation and linear fusion. This two-stage process ensures balanced representation of both silent and active genomic regions. The pooled N-dimensional vector is finally interpolated against the basis lookup $\boldsymbol{P}' \in \mathbb{R}^{N \times h}$ to produce the patch embeddings $\bar{\boldsymbol{E}} \in \mathbb{R}^{N \times h}$, eliminating positional dependencies. This architecture reduces token count by 99.6% (from $N_g = 60,664$ to $N = 256$) while achieving 20.4% higher ARI (0.201 vs. 0.167) than the shared-first-layer baseline; the per-gene baseline is out-of-memory at batch size 1 (Table 7).

2.2 Genetic Pathway Learning

Building on the pathway motivation in Sec 1, we now instantiate it as a discrete latent vocabulary learned end-to-end from data.

Discrete Pathway Modeling.
We propose to learn distinct "*pathway*" tokens, represented as *discrete* codes, by training an encoder

and a vector quantizer. Specifically, the encoder, implemented as a transformer, maps $\bar{\boldsymbol{E}}$ into hidden representations, denoted as $\mathcal{E} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_N\}$. Subsequently, the quantizer learns a codebook $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_K\}$, and associates each $\boldsymbol{e}_i$ with the closest entry in $\mathcal{V}$ in terms of distance. More precisely, for each embedding with index i, we derive the corresponding embedding from the codebook with the following formula: $\boldsymbol{z}_i = \arg\min_{v \in \mathcal{V}} \|\boldsymbol{v} - \boldsymbol{e}_i\|_2$. After acquiring $\mathcal{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N\}$, we input it into an additional decoder, which is implemented as another transformer model, and obtain the output vector $\boldsymbol{o} \in \mathbb{R}^{N_g}$. These modules are trained using reconstruction tasks, where a mean-squared-error (MSE) loss is employed to minimize the dissimilarity between $\boldsymbol{x}$ and $\boldsymbol{o}$. We follow the EMA-updated VQ formulation of Huh et al. (2023), where the codebook is maintained by an exponential moving average rather than gradient descent, so only the encoder-commitment term is needed:

$$\mathcal{L}_{\text{cmt}} = \beta \sum_{i=1}^{N} \left\| \boldsymbol{e}_i - \texttt{sg}(\boldsymbol{z}_i) \right\|_2^2, \tag{1}$$

where sg is the stop-gradient operation and $\beta = 1$ is the commitment weight. To mitigate index collapse, we additionally adopt their unused-token reinitialization and affine reparameterization (details in Appendix G).

2.3 Cell Information Aggregation

While the pathway encoder encodes gene expression values into pathway tokens, it does not provide a cellular-level representation. One possible approach to constructing cell representations involves concatenating the representation of genetic pathway tokens. However, this results in a prohibitively high-dimensional representation, leading to increased computational costs in downstream tasks. Alternatively, using the average representation of pathway tokens, while simpler, yields inferior performance as shown in §4.

Additional tokens to aggregate cell information. To aggregate gene representations effectively and efficiently at the cellular level, we introduce a learnable token, $\boldsymbol{e}_C$, designed to encapsulate cell-specific information. Throughout this section, $\boldsymbol{e}_C$ denotes the input learnable cell-token embedding and $\boldsymbol{h}_C$ denotes its encoder output. This token is associated with a subset of $N' = \lfloor (1-r)\, N \rfloor$ pathway tokens denoted as $\{\boldsymbol{z}_{i_1}, \boldsymbol{z}_{i_2}, \ldots, \boldsymbol{z}_{i_{N'}}\}$, drawn with monotonically increasing indices from the set $\mathcal{Z}$ at mask ratio r (we use $r = 0.15$ for second-stage pretraining, i.e., $N'/N = 85\%$; the same keep ratio appears as $p\%$ in Algorithm 2), so cell-level information is transferred into $\boldsymbol{e}_C$. We employ an encoder to convert these tokens into representations, which we denote as $\mathcal{H} = \{\boldsymbol{h}_C, \boldsymbol{h}_{i_1}, \boldsymbol{h}_{i_2}, \ldots, \boldsymbol{h}_{i_{N'}}\}$. Subsequently, we exclude $\boldsymbol{h}_C$, replace the positions of the previously omitted pathway tokens with a common learnable token $\boldsymbol{e}_M$, and proceed to decode this modified embedding using a decoder.

Token Scaling and Reconstruction Despite pathway tokens being fewer than genes, our scaling algorithm (Algorithm 3) effectively aligns their dimensions through linear interpolation. This approach preserves biological fidelity while reducing computational complexity, as evidenced by the superior clustering performance of $\boldsymbol{h}_C$ over gene-averaged baselines in Table 7. The architecture freezes the encoder and quantizer introduced in §2.2, minimizing reconstruction loss between original and decoded expressions, ensuring stable training while maintaining pathway-aware representations.

Contrastive Regularization. To enhance cell-specific discriminability, we design a contrastive framework leveraging dynamic K-means-derived pseudo-labels. During training, two augmented views ($\mathcal{H}_i$, $\mathcal{H}'_i$) of each cell are generated by masking different pathway token subsets. Embeddings $\boldsymbol{h}_{i,C}$ and $\boldsymbol{h}'_{i,C}$ from the same cell cluster form positive pairs, while cross-cluster cells serve as negatives (limited to K_{neg} pairs per batch, distinct from the codebook size K). Letting $s(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / (\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|)$ denote cosine similarity and τ a temperature (set to $\tau = 0.5$ in our implementation, following Chen et al. 2020), the contrastive loss is

$$\mathcal{L}_{CL} = -\sum_i \log \frac{\exp(s_i^+/\tau)}{\exp(s_i^+/\tau) + \sum_{j \in \text{neg}(i)} \exp(s_{ij}^-/\tau)}, \tag{2}$$

where $s_i^+ := s(\boldsymbol{h}_{i,C}, \boldsymbol{h}'_{i,C})$ and $s_{ij}^- := s(\boldsymbol{h}_{i,C}, \boldsymbol{h}_{j,C})$. A fixed-length queue Q stores recent $\boldsymbol{h}_C$ embeddings, enabling periodic K-means updates to adapt pseudo-labels to evolving representations. This implicit modeling of cell-type relationships enforces topological consistency in the latent space (visualizations in Appendix C). The detailed pipeline is in Algorithm 2.

3 Experiments

For readers unfamiliar with single-cell biology, the evaluation metrics used below are: *ARI* (Adjusted Rand Index, chance-corrected clustering agreement; higher is better, $\in [-1, 1]$); *NMI* (Normalized Mutual Information; higher is better, $\in [0, 1]$); *Leiden* (a graph-based community-detection algorithm); *Pearson* Δ (Pearson correlation between

predicted and observed post-perturbation expression on the top differentially expressed genes); *PCC* (Pearson correlation coefficient); *IC50* (drug concentration inhibiting 50% of cell viability); *Reactome / GSEA* (curated biological pathway databases queried via Gene Set Enrichment Analysis). At a high level, our VQ codebook can be viewed as a *discrete pathway vocabulary*, analogous to learning a BPE/WordPiece subword vocabulary but over biological function units rather than text.

In this section, we detail our experimental setting and demonstrate the superior performance of `scPaLM`. A series of experiments are conducted on various downstream tasks, and we have provided ablation studies in §4 to validate the importance of our proposed techniques.

3.1 Implementation Details

`scPaLM` is pretrained on 43.3M single-cell RNA-seq profiles with $60,664$ genes from CELLxGENE (Megill et al., 2021). The default model uses $N = 256$ patch tokens, a codebook size of $K = 128$, hidden size 768, 12 transformer layers, and 8 attention heads. We evaluate on CLL, COVID-19, Jurkat, PBMC-5k, CCLE/GDSC, Adamson, and Norman benchmarks, with downstream datasets manually excluded from pretraining. Baselines include PCA, scFoundation (Hao et al., 2023), scGPT (Cui et al., 2024), Geneformer (Theodoris et al., 2023), CellPLM (Wen et al., 2023), SAVER (Huang et al., 2018), scImpute (Li and Li, 2018), DCA (Eraslan et al., 2019), MAGIC (Dijk et al., 2017), DeepCDR (Liu et al., 2020), and GEARS (Roohani et al., 2024). Full optimizer, dataset, checkpoint, and hyperparameter details are provided in Appendix B.

3.2 Unsupervised Cell Type Annotation

Our first set of experiments involves applying computational methods on unseen scRNA-seq data and providing type annotations to those unseen cells in an unsupervised manner. We compare the performance of `scPaLM` with baselines including PCA, scFoundation, scGPT, Geneformer, and CellPLM. These experiments are conducted on the CLL and the COVID dataset. UMAP visualizations are shown in Figure 3. The results with variance are presented in Table 1, showing that our model consistently achieves better and more stable performance compared to all baselines.

Performance Comparison. We cluster each model's embeddings with the Leiden (Traag et al., 2019) algorithm. To compare methods fairly and eliminate resolution as a confounding hyperparameter, we apply the *same* sweep protocol to every model (all baselines and `scPaLM`): sweep Leiden resolution over $[0.1, 2.0]$, select the resolution that maximizes the model's own ARI per dataset, and report ARI/NMI at that operating point averaged over 4 random seeds. Every method receives identical sweep budget and selection rule. Qualitatively, PCA exhibits the poorest UMAPs while the foundation models produce noticeably better cluster separation; `scPaLM` yields a smoother and more clearly separated latent space relative to ground-truth labels. Quantitatively, `scPaLM` is the top-performing method on three of the four reported metrics in Table 1 (CLL NMI, COVID-19 ARI, and COVID-19 NMI), and is competitive on CLL ARI where CellPLM is slightly higher. Beyond the per-method optimum, `scPaLM` also sustains a stable plateau of high ARI across the entire $[0.1, 2.0]$ sweep, while several baselines fluctuate sharply with the chosen resolution.

Table 1 reports four-seed ARI/NMI means and variances for CLL and COVID-19.

3.3 Cancer Drug Response Prediction

Cancer Drug Responses is an important task that can help guide the design of anti-cancer drugs and also understand the cancer biology (Unger et al., 2015). Following the setting in scFoundation (Hao et al., 2023), we combine `scPaLM` with a CDR prediction framework, DeepCDR (Liu et al., 2020), to provide prediction of the IC50 values (*i.e.*, half-maximal inhibitory concentrations) of drugs across different cells. We adopt the settings from scFoundation (Hao et al., 2023) to fuse the extracted representations from gene expression values with the representations of drugs and fit a graph convolution network (GCN) to learn representations that encompass information from multiple sources and modalities. We follow the settings of DeepCDR and experiment with different options: (1) *Use Mut*, which indicates the usage of genomic mutation information; and (2) *Use Methy*, which indicates the usage of DNA methylation data.

Figure 4 shows both scFoundation and `scPaLM` improve over the baseline DeepCDR setting. Under the shared scFoundation/DeepCDR protocol with identical mutation and methylation features, `scPaLM` reaches 0.9106 ± 0.004 PCC against scFoundation's 0.8839 ± 0.006, a $+0.027$ PCC gap

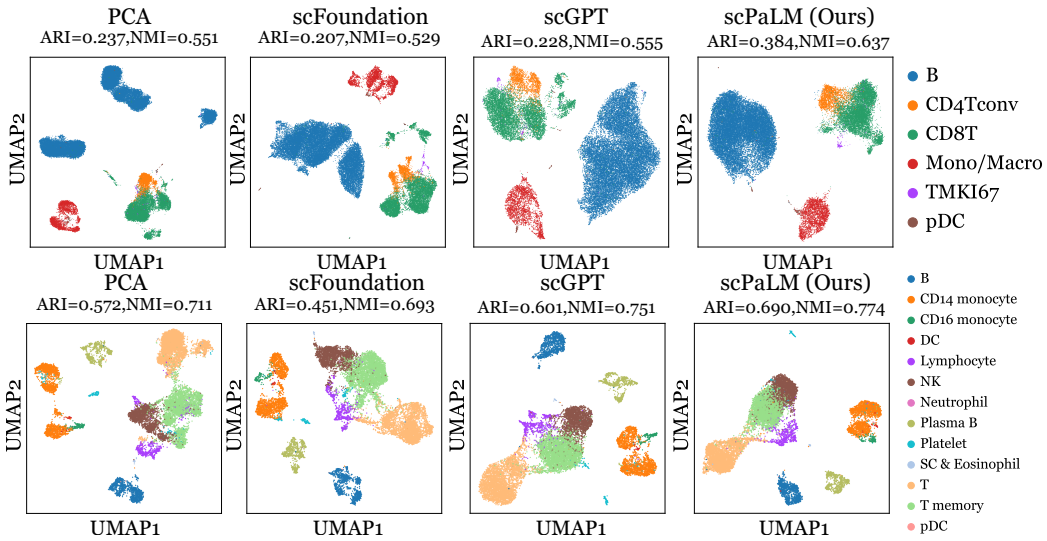

Figure 3: Unsupervised clustering performance on the CLL (top) and COVID-19 (bottom) datasets.

Method	CLL		COVID-19	
	ARI ↑	NMI ↑	ARI ↑	NMI ↑
PCA	0.227 ± 0.008	0.543 ± 0.007	0.527 ± 0.049	0.701 ± 0.011
scFoundation	0.194 ± 0.010	0.515 ± 0.007	0.436 ± 0.016	0.693 ± 0.004
scGPT	0.210 ± 0.024	0.540 ± 0.018	0.210 ± 0.024	0.747 ± 0.004
Geneformer	0.314 ± 0.002	0.537 ± 0.002	0.401 ± 0.011	0.589 ± 0.012
CellPLM	$\mathbf{0.406 \pm 0.016}$	0.571 ± 0.003	0.346 ± 0.003	0.616 ± 0.004
`scPaLM`	0.327 ± 0.052	$\mathbf{0.604 \pm 0.029}$	$\mathbf{0.645 \pm 0.052}$	$\mathbf{0.756 \pm 0.017}$

Table 1: Clustering performance with variance on CLL (Rendeiro et al., 2020) and COVID-19 (Guo et al., 2020). We report ARI and NMI; **bold** marks the best value in each column.

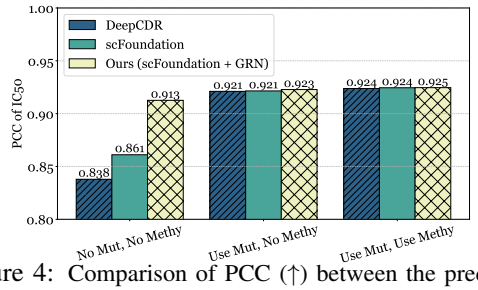

Figure 4: Comparison of PCC (↑) between the predicted and the ground-truth IC50 values using different settings of feature. We compare `scPaLM`'s performance with two baseline algorithms, DeepCDR and scFoundation.

(roughly $4.5\times$ the per-seed standard deviation). The per-cell-line / per-cancer-type / per-drug-type pairwise breakdown in Figure 5 shows that `scPaLM` improves over scFoundation on the majority of cell lines and cancer types, and Figure 6 zooms into low-grade gliomas, where both methods achieve high PCC despite IC50 spanning a wide $[-5, 5]$ range.

3.4 Imputation

We mask 10% of non-zero entries per cell (sampling probability $\propto \exp(-x)$) and report RMSE/MAE on the log-normalized masked entries (Table 2). Even under the zero-shot setting, `scPaLM` outperforms all baselines—which are themselves trained on the downstream datasets—confirming the model's ability to denoise expression and capture cell–gene interactions.

3.5 Gene Perturbation Prediction

Gene perturbation prediction evaluates transcriptional responses to genetic interventions, testing whether representations generalize to unseen perturbations (Roohani et al., 2024).

Experimental Setup. We follow the GEARS benchmark (Roohani et al., 2024) and evaluate on Adamson (Adamson et al., 2016) and Norman (Norman et al., 2019). We use Pearson Δ on the top 20 differentially expressed genes as the primary metric. For perturbation prediction, `scPaLM` uses the pre-pooling per-gene embedding matrix as static gene

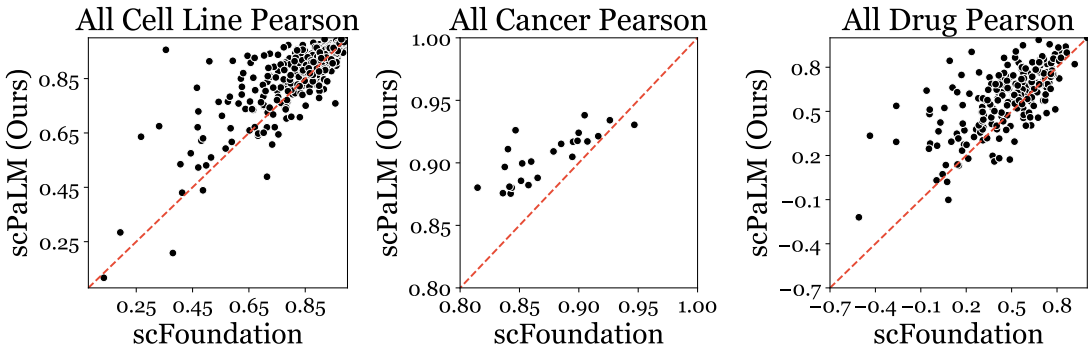

Figure 5: Pairwise comparison of Pearson correlation coefficients for scFoundation and `scPaLM`, grouped by cell lines, cancer types, and drug types. The red lines indicate $y = x$.

Method	Jurkat		PBMC	
	RMSE ↓	MAE ↓	RMSE ↓	MAE ↓
SAVER	0.766±0.001	0.652±0.001	0.939±0.001	0.839±0.001
scImpute	0.997±0.003	0.771±0.002	1.023±0.001	0.838±0.001
DCA	0.381±0.001	0.318±0.001	0.586±0.001	0.488±0.001
MAGIC	0.417±0.001	0.354±0.001	0.659±0.001	0.563±0.001
`scPaLM` **(Zero Shot)**	**0.361**±0.003	**0.279**±0.002	**0.546**±0.001	**0.453**±0.001

Table 2: Imputation performance of various methods on the Jurkat and the PBMC dataset. We use the rooted mean square error (RMSE) and the mean absolute error (MAE) as the measurements. Results are averaged over 4 random seeds.

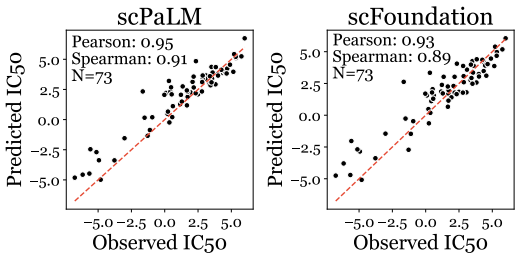

Figure 6: Predicted and observed IC50 values on low-grade gliomas (LGG). Both `scPaLM` and scFoundation achieve high PCC and Spearman correlations despite the wide $[-5, 5]$ range of IC50 values.

Method	Adamson Pearson Δ ↑	Norman Pearson Δ ↑
GEARS (Roohani et al., 2024)	0.618±0.033	0.497±0.013
scGPT (Cui et al., 2024)	0.600±0.042	0.507±0.030
scFoundation (Hao et al., 2023)	0.221±0.019	0.140±0.015
`scPaLM`	**0.692**±0.046	**0.532**±0.021

Table 3: Gene perturbation prediction performance (Pearson Δ on top 20 DE genes, higher is better). Results are averaged over 4 random seeds.

embeddings in the GEARS decoder; the pooled cell token is not used as the perturbation carrier.

Results. As shown in Table 3, `scPaLM` improves over GEARS on Adamson and Norman, indicating that the learned gene and pathway representations transfer to perturbation prediction.

3.6 Biological Structure of Quantized Tokens

Quantitative Validation. We further inspect whether the learned VQ codes align with known biological programs. For each code, we aggregate gene expression from cells where the code is activated, rank protein-coding genes after gene-wise normalization, and query Reactome enrichment for the top associated genes. We compare against a permuted-token null that preserves code frequencies while breaking the code-expression alignment. As shown in Table 4, the learned codebook is non-collapsed and substantially enriched for curated pathways.

Representative Pathways. Representative enriched codes cover immune signaling, GPCR/complement signaling, cell-cell communication, and lipid remodeling; the per-code top-gene lists and Reactome terms are reported in Appendix Table 8. These results support using the VQ bottleneck as a pathway-enriched discrete interface for post-hoc biological inspection.

Analysis	Learned	Permuted null
Active codes	106/128	–
Perplexity	61.8	–
Max code frequency	3.1%	–
Reactome hits (FDR<0.05)	117/128	6/128

Table 4: Codebook utilization and pathway enrichment. Active codes are measured on a held-out 500-cell batch. Reactome hits are computed over *all* 128 codes by querying the top-20 protein-coding genes whose weighted expression most strongly associates with each code (gene-wise z-scored); a code counts as a hit if at least one Reactome term reaches FDR<0.05. The 117/128 count is therefore over the full codebook (including codes that were inactive on the held-out batch but still have well-defined per-code gene associations across pretraining). The permuted-null baseline shuffles code-cell assignments and recomputes the same statistic. Caveat: many enriched codes share overlapping "best terms" (e.g., Lewis blood group biosynthesis); we therefore do not claim one-to-one code-to-pathway correspondence (Sec. 3.6).

4 Ablation Summary

We conduct three ablations to isolate the roles of cell aggregation, the pathway encoder, and permutation-invariant embedding (full tables in Appendix D). The learned cell token reaches ARI 0.384 on CLL versus 0.015 for mean pooling and 0.181 for concatenation (Table 5). The pathway encoder lifts ARI from 0.050 to 0.191 on COVID-19, with VQ adding a further $+0.010$ to reach 0.201 (Table 6). Direct per-gene tokenization out-of-memories at batch size 1, while a shared-first-layer reaches ARI 0.167 versus our 0.201 (Table 7).

Additional controls. We further summarize robustness checks that are detailed in Appendix E. The contrastive regularizer is designed to make $\boldsymbol{h}_C$ discriminative across cell types (Table 1, 5) and is not intended to drive regression-style tasks whose downstream heads consume per-gene embeddings. Consistent with this scoping, removing contrastive learning has negligible effect on Drug Response (0.9106 vs. 0.9102 PCC) and Adamson perturbation (0.6916 vs. 0.6887 Pearson Δ), whereas removing VQ or encoder masking—both of which alter the upstream pathway representation—produces substantially larger drops. Efficiency measurements also reflect the intended token compression: on an NVIDIA RTX A6000, `scPaLM` processes 247 cells/s, compared with 30 cells/s for scGPT and 8.5 cells/s for scFoundation under the measured embedding pipelines.

Method	ARI ↑	NMI ↑	S-Score ↑
Mean	0.015±0.003	0.059±0.005	−0.133±0.012
Concatenated	0.181±0.015	0.478±0.021	0.310±0.018
$\boldsymbol{h}_C$ (No CL)	0.275±0.023	0.573±0.019	0.361±0.025
$\boldsymbol{h}_C$ (Ours)	**0.384**±0.028	**0.637**±0.024	**0.493**±0.031

Table 5: Cell-aggregation ablation on CLL (4 seeds; bold = best).

Configuration		ARI ↑	NMI ↑
Encoder	Quantizer		
✗	✗	0.050±0.008	0.120±0.011
✓	✗	0.191±0.016	0.286±0.019
✓	✓	**0.201**±0.018	**0.291**±0.021

Table 6: Comparison on COVID dataset with different configurations. Models are trained on a small subset of the pretraining data. Results are averaged over 4 random seeds.

5 Conclusion

This work presents `scPaLM`, a single-cell foundation model that tackles the efficiency–interpretability tradeoff via permutation-invariant patch embedding, discrete genetic pathway learning, and contrastive cell-token aggregation. Comprehensive experiments across four downstream tasks confirm its effectiveness. On unsupervised cell type annotation, `scPaLM` attains $+\mathbf{55.7}\%$ relative ARI on CLL and $+\mathbf{207}\%$ on COVID-19 over scGPT (Table 1). For cancer drug response prediction, `scPaLM` reaches **0.9106** PCC against scFoundation's 0.8839 under the shared DeepCDR protocol with identical mutation/methylation features. In zero-shot imputation, `scPaLM` beats the strongest baseline DCA on both Jurkat (0.361 vs. 0.381 RMSE) and PBMC (0.546 vs. 0.586 RMSE) without any task-specific fine-tuning. On gene perturbation prediction, `scPaLM` attains the highest Pearson Δ on *both* Adamson (**0.692**) and Norman (**0.532**). These results, combined with the biological interpretability of our learned pathway tokens, suggest that pathway-centric architectures offer a promising direction for building more efficient and interpretable single-cell foundation models.

Limitations

Implicit Pathway Modeling. While our genetic pathway learning module captures collective gene behaviors through discrete representations, it currently relies on data-driven discovery rather than explicit integration of established pathway databases (e.g., Reactome or KEGG). Future work could enhance biological interpretability by incorporating curated pathway knowledge through hybrid architectures that combine learned codebooks with prior biological constraints. **Human-Centric Data Bias.** Our pre-training dataset primarily focuses on human single-cell transcriptomes from CELLxGENE. While this enables strong performance on human biological tasks, the cross-species generalizability of our pathway representations remains uncertain. The evolutionary divergence of gene regulatory networks across species may require specialized adaptation mechanisms when applying `scPaLM` to model non-human organisms.

Ethical Considerations

This work adheres to ethical research practices in computational biology. All datasets used for pre-training and evaluation are publicly available through GEO/SRA archives, CELLxGENE, or 10x Genomics, with proper ethical approvals obtained in their original studies. Our framework processes only de-identified genomic data, containing no protected health information. While foundation models like `scPaLM` could theoretically accelerate therapeutic development, we emphasize that any clinical application requires rigorous validation through established biomedical research protocols. We release our code and model weights to promote reproducibility and transparency in single-cell computational biology research.

References

Britt Adamson, Thomas M Norman, Marco Jost, Min Y Cho, James K Nuñez, Yuwen Chen, Jacqueline E Villalta, Luke A Gilbert, Max A Horlbeck, Marco Y Hein, and 1 others. 2016. A multiplexed single-cell crispr screening platform enables systematic dissection of the unfolded protein response. *Cell*, 167(7):1867–1882.

Adrian Alexa, Jörg Rahnenführer, and Thomas Lengauer. 2006. Improved scoring of functional groups from gene expression data by decorrelating go graph structure. *Bioinformatics*, 22(13):1600–1607.

Alexei Baevski, Steffen Schneider, and Michael Auli. 2019. vq-wav2vec: Self-supervised learning of discrete speech representations. *arXiv preprint arXiv:1910.05453*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Yiqun Chen and James Zou. 2024. Genept: A simple but effective foundation model for genes and cells built from chatgpt. *bioRxiv*.

Marco Colonna. 2023. The biology of trem receptors. *Nature Reviews Immunology*, 23(9):580–594.

Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, Nan Duan, and Bo Wang. 2024. scgpt: toward building a foundation model for single-cell multi-omics using generative ai. *Nature Methods*, pages 1–11.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

David van Dijk, Juozas Nainys, Roshan Sharma, Pooja Kaithail, Ambrose J Carr, Kevin R Moon, Linas Mazutis, Guy Wolf, Smita Krishnaswamy, and Dana Pe'er. 2017. Magic: A diffusion-based imputation method reveals gene-gene interactions in single-cell rna-sequencing data. *BioRxiv*, page 111591.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Gökcen Eraslan, Lukas M Simon, Maria Mircea, Nikola S Mueller, and Fabian J Theis. 2019. Single-cell rna-seq denoising using a deep count autoencoder. *Nature communications*, 10(1):390.

Patrick Esser, Robin Rombach, and Bjorn Ommer. 2021. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883.

Jellert T Gaublomme, Nir Yosef, Youjin Lee, Rona S Gertner, Li V Yang, Chuan Wu, Pier Paolo Pandolfi, Tak Mak, Rahul Satija, Alex K Shalek, and 1 others. 2015. Single-cell genomics unveils critical regulators of th17 cell pathogenicity. *Cell*, 163(6):1400–1412.

Robert Gray. 1984. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29.

Dominic Grün, Lennart Kester, and Alexander Van Oudenaarden. 2014. Validation of noise models for single-cell transcriptomics. *Nature methods*, 11(6):637–640.

Chuang Guo, Bin Li, Huan Ma, Xiaofang Wang, Pengfei Cai, Qiaoni Yu, Lin Zhu, Liying Jin, Chen Jiang, Jingwen Fang, and 1 others. 2020. Single-cell analysis of two severe covid-19 patients reveals a monocyte-associated and tocilizumab-responding cytokine storm. *Nature communications*, 11(1):3924.

Minsheng Hao, Jing Gong, Xin Zeng, Chiming Liu, Yucheng Guo, Xingyi Cheng, Taifeng Wang, Jianzhu Ma, Le Song, and Xuegong Zhang. 2023. Large scale foundation model on single-cell transcriptomics. *bioRxiv*, pages 2023–05.

Mo Huang, Jingshu Wang, Eduardo Torre, Hannah Dueck, Sydney Shaffer, Roberto Bonasio, John I Murray, Arjun Raj, Mingyao Li, and Nancy R Zhang. 2018. Saver: gene expression recovery for single-cell rna sequencing. *Nature methods*, 15(7):539–542.

Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. 2023. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. *arXiv preprint arXiv:2305.08842*.

Chintan J Joshi, Wenfan Ke, Anna Drangowska-Way, Eyleen J O'Rourke, and Nathan E Lewis. 2022. What are housekeeping genes? *PLoS computational biology*, 18(7):e1010295.

Dragomirka Jovic, Xue Liang, Hua Zeng, Lin Lin, Fengping Xu, and Yonglun Luo. 2022. Single-cell rna sequencing technologies and applications: A brief overview. *Clinical and Translational Medicine*, 12(3):e694.

Peter V Kharchenko, Lev Silberstein, and David T Scadden. 2014. Bayesian approach to single-cell differential expression analysis. *Nature methods*, 11(7):740–742.

Daniel Levine, Syed Asad Rizvi, Sacha Lévy, Nazreen Pallikkavaliyaveetil, David Zhang, Sina Ghadermarzi, Ruiming Wu, Xiang Fu, Bowen Chen, Ryan Cabeen, and 1 others. 2024. Cell2sentence: Teaching large language models the language of biology. In *International Conference on Machine Learning*.

Wei Vivian Li and Jingyi Jessica Li. 2018. An accurate and robust imputation method scimpute for single-cell rna-seq data. *Nature communications*, 9(1):997.

Qiao Liu, Zhiqiang Hu, Rui Jiang, and Mu Zhou. 2020. Deepcdr: a hybrid graph convolutional network for predicting cancer drug response. *Bioinformatics*, 36(Supplement_2):i911–i918.

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

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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

Colin Megill, Bruce Martin, Charlotte Weaver, Sidney Bell, Lia Prins, Seve Badajoz, Brian McCandless, Angela Oliveira Pisco, Marcus Kinsella, Fiona Griffin, and 1 others. 2021. Cellxgene: a performant, scalable exploration platform for high dimensional sparse matrices. *bioRxiv*, pages 2021–04.

Huaiyu Mi, Anushya Muruganujan, John T Casagrande, and Paul D Thomas. 2013. Large-scale gene function analysis with the panther classification system. *Nature protocols*, 8(8):1551–1566.

Thomas M Norman, Max A Horlbeck, Joseph M Replogle, Alex Y Ge, Albert Xu, Marco Jost, Luke A Gilbert, and Jonathan S Weissman. 2019. Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. *Science*, 365(6455):786–793.

Anoop P Patel, Itay Tirosh, John J Trombetta, Alex K Shalek, Shawn M Gillespie, Hiroaki Wakimoto, Daniel P Cahill, Brian V Nahed, William T Curry, Robert L Martuza, and 1 others. 2014. Single-cell rna-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science*, 344(6190):1396–1401.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32.

André F Rendeiro, Thomas Krausgruber, Nikolaus Fortelny, Fangwen Zhao, Thomas Penz, Matthias Farlik, Linda C Schuster, Amelie Nemc, Szabolcs Tasnády, Marienn Réti, and 1 others. 2020. Chromatin mapping and single-cell immune profiling define the temporal dynamics of ibrutinib response in cll. *Nature Communications*, 11(1):577.

Yusuf Roohani, Kexin Huang, and Jure Leskovec. 2024. Predicting transcriptional outcomes of novel multigene perturbations with gears. *Nature Biotechnology*, 42(6):927–935.

Yanay Rosen, Yusuf Roohani, Ayush Agarwal, Leon Samotorcan, Tabula Sapiens Consortium, Stephen R Quake, and Jure Leskovec. 2023. Universal cell embeddings: A foundation model for cell biology. *bioRxiv*.

Anna C Schaar, Alejandro Tejada-Lapuerta, Giovanni Palla, Robert Gutgesell, Lennard Halle, Mariia Minaeva, Larsen Vornholz, Leander Dony, Francesca Drummer, Mojtaba Bahrami, and Fabian J Theis. 2024. Nicheformer: A foundation model for single-cell and spatial omics. *bioRxiv*.

Stefan Semrau, Johanna E Goldmann, Magali Soumillon, Tarjei S Mikkelsen, Rudolf Jaenisch, and Alexander Van Oudenaarden. 2017. Dynamics of lineage commitment revealed by single-cell transcriptomics of differentiating embryonic stem cells. *Nature communications*, 8(1):1096.

Christina V Theodoris, Ling Xiao, Anant Chopra, Mark D Chaffin, Zeina R Al Sayed, Matthew C Hill, Helene Mantineo, Elizabeth M Brydon, Zexian Zeng, X Shirley Liu, and 1 others. 2023. Transfer learning enables predictions in network biology. *Nature*, pages 1–9.

Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):5233.

Florian T Unger, Irene Witte, and Kerstin A David. 2015. Prediction of individual response to anticancer therapy: historical and future perspectives. *Cellular and molecular life sciences*, 72:729–757.

Aaron Van Den Oord, Oriol Vinyals, and 1 others. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Hongzhi Wen, Wenzhuo Tang, Xinnan Dai, Jiayuan Ding, Wei Jin, Yuying Xie, and Jiliang Tang. 2023. Cellplm: Pre-training of cell language model beyond single cells. *bioRxiv*, pages 2023–10.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. 2021. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*.

Fan Yang, Wenchuan Wang, Fang Wang, Yuan Fang, Duyu Tang, Junzhou Huang, Hui Lu, and Jianhua Yao. 2022. scbert as a large-scale pretrained deep language model for cell type annotation of single-cell rna-seq data. *Nature Machine Intelligence*, 4(10):852–866.

A Extended Related Work

Single-cell Data Analysis. Several methods have been proposed to model single-cell transcriptomes while addressing biological variability. MAGIC (Dijk et al., 2017) imputes missing measurements ("dropouts") by propagating values over a cell–cell similarity graph. scImpute (Li and Li, 2018) identifies dropouts and imputes them at selected positions, while SAVER (Huang et al., 2018) exploits gene–gene relationships to recover expression levels. scVI (Lopez et al., 2018) offers a scalable probabilistic framework for representation and analysis of single-cell data. Pretrained transformers have since been applied to scRNA-seq modeling: encoder-only models such as scBERT (Yang et al., 2022) embed genes as tokens to model over 16k genes per cell, scFoundation (Hao et al., 2023) improves the embedding process, Geneformer (Theodoris et al., 2023) encodes rank-based gene expression across datasets, decoder-only scGPT (Cui et al., 2024) predicts masked genes via next-token prediction, and CellPLM (Wen et al., 2023) incorporates spatially resolved data into pretraining. A more recent line of work scales single-cell foundation models further or aligns them with natural language: UCE (Rosen et al., 2023) pretrains a universal cell embedding across species, GenePT (Chen and Zou, 2024) aligns gene tokens with LLM text embeddings, Cell2Sentence (Levine et al., 2024) casts cells as ranked-gene sentences for autoregressive LMs, and Nicheformer (Schaar et al., 2024) extends pretraining to spatial neighborhoods. While prior works primarily focus on imputing dropouts or learning continuous gene-level embeddings, we orthogonally introduce a vector-quantization framework that learns discrete genetic pathway representations, offering an efficient and interpretable basis for downstream biological analysis.

Vector Quantization (VQ). VQ (Gray, 1984) is a classical quantization technique that maintains a *codebook* of representations and associates each input vector with its closest code, providing a form of discrete representation learning in which a single token is activated. VQ has been applied to image generation (Van Den Oord et al., 2017; Razavi et al., 2019; Esser et al., 2021), video generation (Yan et al., 2021), and speech recognition (Baevski et al., 2019), and a small body of recent work also explores discrete codes in single-cell modeling. To our knowledge, `scPaLM` is the first to use a VQ codebook to represent *pathway-level* discrete units in a transformer-based single-cell foundation model, rather than per-gene or per-cell-type discrete tokens.

B Additional Experiment Details

B.1 Implementation Details

Pretraining Data. `scPaLM` is pretrained on single-cell RNA-seq data covering different types of cells, having in total $43,312,189$ cells and $60,664$ genes collected from CELLxGENE platform (Megill et al., 2021). The statistics and description of the pre-training data are in Appendix F.

Architectures. The overall architecture of `scPaLM` can be split into three parts: (1) embedding layers, where we use mainly MLPs as described in Algorithm 1. The N is set to 256 in our experiment; (2) genetic pathway learning, where we introduce an encoder, a decoder, and a quantizer. The encoder and the decoder are developed based on the transformer architecture, which contains 12 layers and a hidden dimension of 768. The quantizer has the same hidden dimension with a codebook size of 128; (3) cell information aggregation, where we introduce another encoder and decoder that have the same architectures and configuration. More details on the architectures can be found in Table 10.

Experiment Settings. The optimizer we use in our experiments is `AdamW` (Loshchilov and Hutter, 2017). In the first stage of training, where we train the encoder for pathway tokens and the vector quantizer, we adopt a learning rate of 0.001 and a batch size of 128, with first-stage mask ratio 0.75 and commitment weight $\beta = 1$. In the second stage, we train other components with a learning rate of 5×10^{-4}, the same batch size of 128, and a decoder-side mask ratio of 0.15 (i.e., 85% of pathway tokens are kept for the cell-token aggregation pathway in Algorithm 2). The contrastive regularizer uses temperature $\tau = 0.5$, queue length $|Q| = 256$, 8 K-means cluster centers for pseudo-label assignment, and a CL-to-MSE loss weight of 0.1. All performance metrics use Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI), and are averaged over 4 random seeds.

B.2 Imputation Protocol

Imputation is an important task where the model is asked to recover the expression value of genes within individual cells. It has real-world implications because the measurement of expression levels often exhibits noise (Grün et al., 2014) and suffer from dropout events (Kharchenko et al., 2014). We conduct a series of simulated experiments on the Jurkat and the PBMC dataset to assess `scPaLM`'s ability to accurately predict the expression levels of the missing genes. We randomly sample 10% of genes from each cell with a probability that is proportional to the exponent of negative gene expression values and mask them as 0.

B.3 Drug Response Visual Analysis

To have a better understanding of the performance gain, we follow the setting of scFoundation and disable the mutation and the methylation features, to focus on the benefit brought by the incorporation of embeddings from gene expression values. Figure 5 shows that `scPaLM` achieves better PCCs on all but one cancer type and improves the metrics on a majority of cell lines. For low-grade gliomas (LGG), both methods achieve high PCC values despite IC50 values spanning a large range from -5 to 5 (Figure 6). `scPaLM` shows favorable PCC and Spearman correlation in this setting.

C Additional Experiment Figures

Figure 5 provides a per-cell-line, per-cancer-type, and per-drug-type pairwise comparison of `scPaLM` and scFoundation under the shared DeepCDR protocol, complementing the aggregate PCC reported in Sec 3.3. Figure 6 zooms into the low-grade glioma (LGG) cancer type, where the IC50 range spans roughly $[-5, 5]$ across drugs; both methods retain high Pearson and Spearman correlation in this setting, indicating that the gain of `scPaLM` is not driven by a narrow dynamic range.

D Ablation Details

D.1 Cell Aggregation, Pathway Encoder, and Embedding Ablations

Table 5 (in the main text) gives the full CLL cell-aggregation ablation; we further provide here the pathway-encoder/quantizer ablation on COVID-19 (Table 6) and the embedding-process comparison (Table 7).

The effectiveness of the cell information aggregation process. We conduct a series of experiments on the CLL dataset to compare two alternatives for building cell representations, where we use the average and the concatenated representations of pathway tokens to represent cells. Table 5 presents the performance on the cell type annotation task, where we can observe that using our cell information aggregation technique yields the best performance among all the variants. We also conducted an experiment where we do not use the

token-level contrastive learning framework to obtain the embedding of $\boldsymbol{h}_C$. The decreased scores of demonstrate the importance of the token-level contrastive learning regularizer.

The effectiveness of the pathway encoder. We conduct experiments excluding the genetic pathway learning module discussed in §2.2. Instead of `scPaLM`, we train the embedding layers and also aggregate information directly from the embeddings of genes on a small subset of training data that have around 100K cells. Table 6 shows the importance of the encoder and the quantizer. We see that the introduced genetic pathway encoder helps improve the clustering performance, by the ARI and NMI gain of 0.14 and 0.16, respectively. The usage of the quantizer further improves the performance by an additional 1%.

The effectiveness of the embedding process. To evaluate the effectiveness of our embedding process, we explore several alternatives and compare them to our proposed embedding process: (1) *per-gene*, where we directly employs gene-specific embeddings $\boldsymbol{E}$. This is a widely adopted option in various methods such as scFoundation (Hao et al., 2023) and Geneformer (Theodoris et al., 2023); (2) *shared-first-layer*, where we employ only a shared $\boldsymbol{P}$ for all the genes. The results are presented in Table 7, where we can observe that these alternatives demonstrate either degraded performance, or suffer from overly high computational cost. The *per-gene* variant results in out-of-memory (OOM) error even using a batch size of 1. Using a shared first layer requires less amount of GPU memory but yields inferior performance compares to our method.

Embedding Method	Memory ↓	ARI ↑	NMI ↑
Per-gene	>80G	-	-
Shared-first-layer	42378MiB	0.167±0.014	0.251±0.017
Perm.-Inv. (Ours)	43678MiB	0.201±0.018	0.291±0.021

Table 7: Comparison between different embedding processes. The models are trained on a subset of the pretraining data and evaluated on the COVID dataset. Results are averaged over 4 random seeds.

E Additional Robustness Analyses

We include additional analyses used to validate robustness and clarify component contributions. Removing contrastive loss changes Drug Response PCC by 0.0004 and Adamson Pearson Δ by 0.0029, indicating that these two tasks are not driven primarily by the contrastive term. Removing encoder masking reduces Drug Response PCC from 0.9106 to 0.8361, while removing VQ reduces it to 0.8887; on Adamson, removing VQ reduces Pearson Δ from 0.6916 to 0.6307. Efficiency measurements on an NVIDIA RTX A6000 show 247 cells/s for `scPaLM`, compared with 30 cells/s for scGPT and 8.5 cells/s for scFoundation.

Compute and Carbon Footprint. End-to-end pretraining (both stages) on the 43.3M-cell CELLxGENE corpus takes approximately 200–300 GPU-hours on NVIDIA RTX A6000 / similar 48GB-class GPUs (12 transformer layers, hidden size 768, $N = 256$ patch tokens, bf16 mixed precision, 6 data-parallel devices). All downstream fine-tuning is short relative to pretraining.

Codebook Size Choice. We fix the codebook size to $K = 128$ throughout this paper. In a preliminary hyperparameter sweep over $K \in \{64, 128, 256\}$, pretraining reconstruction loss was largely insensitive within this range, so we report a single representative K; a fully systematic sweep over codebook size jointly with downstream metrics is left to future work.

F Dataset Description

Our pretraining data comes from CELLxGENE (Megill et al., 2021), which has $43,312,189$ cells from more than 8 tissues.

G More Details on Methodology

Permutation-Invariant Embedding. Algorithm 1 transforms gene expression vectors into permutation-invariant patch embeddings. Each gene's scalar expression is first lifted to a d-dimensional latent via the gene-specific projection $\boldsymbol{P}_1 \in \mathbb{R}^{N_g \times 1 \times d}$, followed by a non-linearity and a small intra-gene MLP weight $\boldsymbol{P}_2 \in \mathbb{R}^{d \times d}$. The result is rescaled by the gene-specific 2D weight matrix $\boldsymbol{\alpha} \in \mathbb{R}^{N_g \times d}$ and combined residually with the MLP output to form $\boldsymbol{E}_{\text{value}}$; zero-expression genes are replaced in place by a dedicated learnable embedding $\boldsymbol{e}_{\text{zero}}$. We then concatenate this value-conditioned channel with the static gene-identity channel $\boldsymbol{E}_{\text{gene}}$ and project through the fuse layer $\boldsymbol{W}_f$ into N output slots. A symmetric pooling function $f(\cdot)$ aggregates across the gene axis (with normalized weights so that the comparatively few non-zero genes are not

Source	Gene list	Enriched program / pathway	Statistic
Original token 25	CD1E, TREM2, ICAM5, CD1B	Immunoregulatory Interactions Between A Lymphoid And A non-Lymphoid Cell	$p = 2.8 \times 10^{-7}$
Original token 25	BTN1A1, MRC1, CD1E, TREM2, ICAM5, CD1B	Adaptive Immune System	$p = 4.4 \times 10^{-7}$
Codebook summary	Top-20 protein-coding genes per token	Reactome hits in learned codebook vs. permuted-token null	117/128 vs. 6/128
Utilization	Held-out 500-cell batch	Active codes / perplexity / max code frequency	106/128; 61.8; 3.1%
Token 5	LCK, LILRB2, LILRA5, MYO1G	Immune System	FDR 0.0057
Token 22	CH25H, CCR1, CD37, C3AR1	GPCR and complement signaling	FDR 0.013
Token 94	DSP, CDH15, LCK, ROBO4	Cell-cell communication	FDR 0.026
Token 9	APOB, DHRS2, BCL2A1, GABRP	Plasma lipoprotein remodeling	FDR 0.043

Table 8: Representative pathway enrichments and codebook diagnostics. The original immune-related examples are retained, and the added rows summarize the learned-vs-null Reactome analysis, codebook utilization, and representative pathway-enriched tokens after protein-coding filtering.

Algorithm 1 Permutation-Invariant Embedding

Input: a batch of log-normalized expression vectors $\boldsymbol{X} \in \mathbb{R}^{B \times N_g}$ with batch size B.
Learnable params: projection $\boldsymbol{P}_1 \in \mathbb{R}^{N_g \times 1 \times d}$, MLP weight $\boldsymbol{P}_2 \in \mathbb{R}^{d \times d}$, gene-specific scale $\boldsymbol{\alpha} \in \mathbb{R}^{N_g \times d}$, gene embeddings $\boldsymbol{E}_{\text{gene}} \in \mathbb{R}^{N_g \times d}$, zero-gene embedding $\boldsymbol{e}_{\text{zero}} \in \mathbb{R}^{N_g \times d}$, fuse layer $\boldsymbol{W}_f \in \mathbb{R}^{2d \times N}$, and basis lookup $\boldsymbol{P}' \in \mathbb{R}^{N \times h}$.
Output: $\bar{\boldsymbol{E}} \in \mathbb{R}^{B \times N \times h}$ via symmetric pooling $f(\cdot)$ over the gene axis.
$\boldsymbol{H} \leftarrow$ `einsum`("bg,gqd $\rightarrow$ bgd", $\boldsymbol{X}$, $\boldsymbol{P}_1$) // project each gene scalar into $\mathbb{R}^d$
$\boldsymbol{H} \leftarrow$ `leaky_relu`($\boldsymbol{H}$)
$\boldsymbol{E}_{\text{value}} \leftarrow \boldsymbol{\alpha} \odot \boldsymbol{H} +$ `einsum`("bgd,dd' $\rightarrow$ bgd'", $\boldsymbol{H}$, $\boldsymbol{P}_2$) // gene-wise scale + MLP residual
$\boldsymbol{Z} \leftarrow (\boldsymbol{1} - \boldsymbol{m}) \odot \boldsymbol{E}_{\text{value}} + \boldsymbol{m} \odot \boldsymbol{e}_{\text{zero}}$ // $\boldsymbol{m}$: mask of zero-expression genes
$\boldsymbol{Z} \leftarrow$ `concat`($\boldsymbol{Z}$, $\boldsymbol{E}_{\text{gene}}$) $\boldsymbol{W}_f$ // fuse static gene id with value-conditioned channel
$\bar{\boldsymbol{e}} \leftarrow f(\boldsymbol{Z})$ over the gene axis (zero positions are normalized so that non-zero genes dominate)
$\bar{\boldsymbol{E}} \leftarrow$ `einsum`("bN,Nh $\rightarrow$ bNh", $\bar{\boldsymbol{e}}$, $\boldsymbol{P}'$) // basis interpolation $\mathbb{R}^N \rightarrow \mathbb{R}^{N \times h}$
return $\bar{\boldsymbol{E}}$

Symbol	Description
N	Number of patches
N_c	Number of cells
N_g	Number of genes
$X \in \mathbb{R}^{N_c \times N_g}$	Log-normalized gene expression matrix
$\boldsymbol{\alpha} \in \mathbb{R}^{N_g \times d}$	Gene-specific learnable scale weights
$\boldsymbol{P}_1 \in \mathbb{R}^{N_g \times 1 \times d}$	Per-gene linear projection (scalar $\rightarrow \mathbb{R}^d$)
$\boldsymbol{P}_2 \in \mathbb{R}^{d \times d}$	Intra-gene MLP weight
$\boldsymbol{P}' \in \mathbb{R}^{N \times h}$	Basis lookup for N patch slots
$\bar{\boldsymbol{E}} \in \mathbb{R}^{N \times h}$	Final patch-level embeddings after pooling
$\boldsymbol{E} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N\}$	Pathway-encoder token representations
$\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K\}$	Codebook entries (pathway tokens), K=128
$\boldsymbol{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$	Quantized pathway codes
$\boldsymbol{o} \in \mathbb{R}^{N_g}$	Reconstructed gene expression vector
$\boldsymbol{e}_C, \boldsymbol{h}_C$	Cell-token input embedding / encoder output
$\boldsymbol{e}_M$	Learnable mask placeholder for decoder
β, τ	VQ commit weight; CL softmax temperature
$K_{\text{neg}}, \lvert Q \rvert$	#negatives per batch; CL queue length
r	Stage-2 mask ratio ($N' = (1-r)N$)

Table 9: Summary of key notations used in scPaLM.

drowned out by the much larger zero population), and the pooled N-dimensional vector is finally interpolated against the basis matrix $\boldsymbol{P}' \in \mathbb{R}^{N \times h}$ to produce the N-token, h-dimensional patch embeddings $\bar{\boldsymbol{E}} \in \mathbb{R}^{B \times N \times h}$. This decomposition keeps the embedding invariant to permutations of the gene axis while drastically reducing token count.

Genetic Pathway Learning. Algorithm 2 outlines the training pipeline for cell representation learning. It first encodes gene expressions into pathway tokens $\mathcal{Z}$ and aggregates cell-level information via a learnable token $\boldsymbol{e}_C$ appended to masked representation. When contrastive learning is enabled, K-Means periodically updates pseudo-labels using stored embeddings in queue Q, and Equation 2

Algorithm 2 Training Pipeline (One Step)

Input: A batch of gene expression vectors $\boldsymbol{X} \in \mathbb{N}^{B \times N_g}$, current time step T, an interval for re-fit T_r, a K-Means classifier K, a queue Q.
Obtain $\bar{\boldsymbol{E}} \in \mathbb{R}^{B \times N \times d}$ according to §2.1.
Obtain $\mathcal{Z} \in \mathbb{R}^{B \times N \times d}$ according to §2.2.
Randomly select $p\%$ tokens from $\mathcal{Z}$ for each sample and prepend a token $\boldsymbol{e}_C$. Obtain $\mathcal{H}$ and assign clusters.
if use token-level contrastive learning **then**
  if $T\%T_r == 0$ **then**
    Run K-means based on embeddings in Q.
  end if
  Randomly select $p\%$ tokens from $\mathcal{Z}$ for each sample and prepend $\boldsymbol{e}_C$. Obtain $\mathcal{H}'$.
  Calculate the contrastive learning loss according to Equation 2.
end if
Fill $\boldsymbol{e}_M$ into $\mathcal{H}$ at positions previously excluded during sampling.
Train $\boldsymbol{e}_C$, $\boldsymbol{e}_M$, and the two networks with reconstruction loss for excluded tokens (*i.e.*, $\boldsymbol{e}_M$).
Store $\boldsymbol{h}_C$ in Q.

optimizes cluster consistency. The model trains $\boldsymbol{e}_C$, $\boldsymbol{e}_M$, and networks using reconstruction loss for masked tokens while updating $\boldsymbol{h}_C$ in Q.

Mask Construction and Output Reshaping. In Algorithm 3, we introduce a simple way to make the shape of the output from the decoder introduced in §2.3 consistent with the original gene expression vector. Essentially, we flatten the hidden representations, and we assign a region according to the indices of leave-out tokens in which we calculate the MSE loss.

VQ-Techniques For Stable Training. To address the potential index collapse when applying VQ techniques in training neural networks, we follow the pipeline introduced in Huh et al. (Huh et al., 2023). Firstly, they introduce an affine transformation to reparameterize the representation in the codebook with the following formula:

$$\boldsymbol{v}_i = \boldsymbol{c}_{\text{mean}} + \boldsymbol{c}_{\text{std}} \times \boldsymbol{c}_i$$

where the $\boldsymbol{c}_i$ represents the original code vector, and $\boldsymbol{c}_{\text{mean}}$ and $\boldsymbol{c}_{\text{std}}$ indicate the shared affine parameters. Moreover, they introduce several minor modifications to the codebook update process to enhance the stability.

Algorithm 3 Reshaping Masks and Outputs For Loss Calculation.

Input: a gene expression vector $\boldsymbol{x} \in \mathbb{R}^{N_g}$, the hold-out indices $I = \{i_1, \ldots, i_m\}$, number of tokens N.
Calculate the scaling factor $s \leftarrow \lceil N_g / N \rceil$.
Initialize a mask vector $\boldsymbol{m} \leftarrow \mathbf{0}^{N_g}$.
for $j = 1, 2, \ldots, m$ **do**
  $\boldsymbol{m}_{s \times i_j : s \times (i_j + 1)} \leftarrow \mathbf{1}^s$.
end for
Flatten the output from the decoder which also has the shape of $N \times s$ to $1 \times (N \times s)$, and store it as $\boldsymbol{o}$.
Crop both $\boldsymbol{o}$ and $\boldsymbol{m}$ to have the length of N_g. Calculate the MSE loss as $\mathcal{L}_{\text{MSE}} = \|\boldsymbol{o} - \boldsymbol{x}\|_2^2 / \|\boldsymbol{m}\|_1$.

H More Details on Experiments

Model Configurations. Table 10 presents the hyperparameters of our `scPaLM`.

Table 10: Hyperparameter configurations of `scPaLM`.

Hyperparameter	Value
Architecture	
Hidden size	768
Intermediate size	3072
Number of layers	12
Number of attention heads	8
Dropout	0.0
Attention dropout	0.0
Number of patch tokens N	256
Codebook size K	128
Optimization	
Optimizer	AdamW
Stage-1 learning rate	1×10^{-3}
Stage-2 learning rate	5×10^{-4}
Batch size (per device)	128
Mixed precision	bf16
VQ / masking	
Commitment weight β	1.0
Stage-1 mask ratio	0.75
Stage-2 decoder mask ratio r	0.15
Tokens kept for $\boldsymbol{e}_C$ (N'/N)	85%
Contrastive regularizer	
Softmax temperature τ	0.5
K-means queue length $\lvert Q \rvert$	256
K-means #clusters	8
CL-to-MSE loss weight	0.1

Association of Genes with Pathway Tokens. In this section, we describe the methodology em-

ployed for associating genes with specific pathway identifiers through an algorithmic approach. The process involves the utilization of a matrix with a dimension of K by N_g, where K represents the number of tokens and N_g the number of genes. For each vector of gene expression, we obtain the set of tokens that are activated within the codebook. Upon activation of the K_i-th token, the corresponding raw gene expression vector is scaled by the frequency of K_i token occurrences among the activated tokens and subsequently aggregated to the K_i-th row of the matrix. This procedure is iterated across the entire gene dataset. Subsequent to the completion of this iterative process, we perform a normalization step on each column, which correlates to individual genes. Following normalization, for each row, we identify and select the genes that exhibit the most significant values (Colonna, 2023).

I Potential Risks

While `scPaLM` demonstrates promising capabilities for therapeutic discovery and cellular analysis, we acknowledge the dual-use potential inherent in any foundational biomedical AI technology. The model's ability to predict cell-drug interactions could theoretically be misapplied to screen compounds with harmful biological activity. To mitigate this risk, we emphasize that any clinical translation of our method must occur within established regulatory frameworks requiring rigorous safety evaluation and ethical oversight. Researchers utilizing this technology should adhere to institutional bio-safety review processes and existing chemical/biological weapons conventions to prevent misuse.