Pre-training of Single-cell Language Models through Genetic Pathway Learning

Xuxi Chen [1] **Zhangyang Wang** [1] **Marinka Zitnik** [2] **Manolis Kellis** [2] **Tianlong Chen** [3 2]

Abstract

The utilization of state-of-the-art single-cell RNA sequencing (scRNA-seq) techniques has significantly enhanced the depth and richness of scRNA-seq datasets, contributing to a more comprehensive comprehension of cellular biology and facilitating advancements across a spectrum of research domains. In this work, we propose a novel **S**ingle-**c**ell Pre-trained **L**anguage **M**odel via Genetic **Pa**thway Learning, named **scPaLM**, that effectively harnesses scRNA-seq data and enables various downstream applications. **scPaLM** integrates several innovative designs: (1) an embedding process that adeptly represents gene information with a reduced token count, enhancing computational efficiency; (2) a genetic pathway learning module that is designed to learn discrete representations, enabling the modeling of collective gene behaviors in a data-driven way; (3) an innovative training methodology that progressively aggregates cell representations into a designated token during the training phase, with a tailored masking strategy and a token-level contrastive regularizer. **scPaLM** demonstrates superior performance on various downstream tasks, including cell type annotations, imputation, and cancer drug response prediction, by clear margins compared to baselines.

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 (Patel et al., 2014) diagnosis or prognosis.

To effectively model scRNA-seq data, various computational methods with different architectural designs have been proposed. Recent advances in deep learning have inspired the application of advanced machine learning techniques, such as transformers (Vaswani et al., 2017), to scRNA-seq data analysis. These models, especially those inspired by the success of BERT (Devlin et al., 2018) and GPT (Radford et al., 2019), use a token-based approach where each gene's expression count is treated as a "token", a concept widely adopted in natural language processing (NLP). These tokens are then compiled into a "sentence" that represents the genetic expression profile of a cell. While transformer-based algorithms have proven to be effective, they are not without their limitations. First, treating each gene as a distinct token significantly increases the total token count, leading to considerable computational demands. One solution is to select a subset of highly variable genes, such as the top 2000, which can significantly reduce the gene count. However, this approach inevitably results in the loss of valuable biological information. Alternatively, the use of memory-efficient transformers can help reduce computational costs, though possibly at the expense of performance. Additionally, these transformer-based models overlook biological priors, which could prevent the models from attaining a more thorough understanding of scRNA-seq data. For example, current methods often treat each cell as a sequence of genes, even though the order of genes does not inherently have biological significance.

In this paper, we propose **scPaLM**, a transformer-based model that effectively harnesses massive scRNA-seq data. **scPaLM** incorporates multiple innovative elements: (1) We introduce an efficient embedding process that condenses information from all genes into a reduced number of tokens by leveraging a symmetric transformation. This ensures the embedding process is permutation-invariant, significantly reduces computational costs, and facilitates rapid training and inference; (2) Recognizing the collective nature of gene functionality, we introduce a genetic pathway encoder. This encoder translates gene tokens into genetically related pathway tokens and obtains discrete representations

[1]University of Texas at Austin [2]MIT [3]University of North Carolina at Chapel Hill. Correspondence to: Tianlong Chen <tianlong@cs.unc.edu>.

Proceedings of the ICML 2024 Workshop on Accessible and Efficient Foundation Models for Biological Discovery, Vienna, Austria.

to capture the unique yet collective functionality of genes; (3) To aggregate cell-specific information, we establish a tailored training framework that learns a designated token to represent cells, incorporating a specific masking strategy and a token-level contrastive regularizer. In several downstream tasks including cell type annotation, drug response prediction and imputation, **scPaLM** achieves greater performance compared to baseline methods including Geneformer (Theodoris et al., 2023), scGPT (Cui et al., 2023) and scFoundation (Hao et al., 2023).

2. Related Works

Several methods have been proposed to model transcriptome measurements at the single-cell level that reflect biological diversity (Lopez et al., 2018). MAGIC (Dijk et al., 2017) aims to predict the missing measurements, often referred to as "dropouts", by propagating in a graph constructed based on cell-cell similarity. scImpute (Li & Li, 2018) learns to accurately and robustly identify dropouts and perform imputation on these identified positions. SAVER (Huang et al., 2018) leverages gene-to-gene relationships to recover the expression level of each gene individual cell. Lopez et al. (2018) developed a scalable framework called scVI for probabilistic representation and analysis of gene expression in single cells. More recently, there has been a notable utilization of pretrained transformers in the context of modeling single-cell RNA sequencing (scRNA-seq) data. In the realm of encoder-only transformers, scBERT (Yang et al., 2022) embeds each gene into a token and leverages an efficient transformer to model over 16000 genes for each individual cell. Subsequently, scFoundation (Hao et al., 2023) has made advancements in the embedding process introduced by scBERT and resulted in enhanced performance. Geneformer (Theodoris et al., 2023) discards the original measurement of transcriptome and constructs input sequences that account for the ranking of measurements across the entirety of the dataset, thereby creating a representation that encapsulates the relative expression levels of all genes within each cell. For decoder-based models, scGPT (Cui et al., 2023) leverages the concept of next token prediction in NLP to iteratively predict the masked genes, creating a novel path for scRNA-seq data modeling. A concurrent work CellPLM (Wen et al., 2023) encodes cell-cell relations by leveraging spatially-resolved transcriptomic data in pretraining. In this work, our objective is to deploy a vector quantization technique to learn discrete genetic pathway representation.

3. Methodology

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 Section 3.2), 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 Section 3.3). The two encoders we have trained in these two stages collectively empower various downstream tasks, exhibiting superior performance.

3.1. Permutation-Invariant Embedding

A crucial aspect of implementing the Transformer involves the construction of "tokens". One commonly employed method for tokenization is to transform each gene's expression value into a token (Hao et al., 2023; Cui et al., 2023), which yields a sequence with a length of N_g. However, it is worth noting that N_g can sometimes grow to reach values in the tens of thousands, posing significant challenges due to the substantial memory and computational resources required to handle such a long sequence. To alleviate the necessity for an excessively large token count, an alternative approach can be employed known as "patching", a technique inspired by the Vision Transformer (Dosovitskiy et al., 2020), which groups nearby genes together, thereby reducing computational complexity. While this strategy seems to be effective, it does exhibit sensitivity to the *order* of genes within different patches. In contrast to natural languages, the relative positions of genes lack biological significance; in other words, permuting the vector of gene expression values results in the same cellular expression, which is not the case for the patching technique.

We introduce a tokenization scheme that helps reduce computational costs and enhances the representation of gene expression values while ensuring permutation invariance. Our approach initiates by mapping the expression values of individual genes into high-dimensional vectors through gene-specific projection matrices, represented as $\boldsymbol{P} \in \mathbb{R}^{N_g \times d_1}$. A trainable gene-specific coefficient $\boldsymbol{\alpha}$ is adopted to scale these vectors, concurrently pass them through a linear layer, and finally the outputs of these two operations by summation are combined to obtain $\boldsymbol{E}_{\text{value}}$. Subsequently, we set up learnable embeddings for individual gene, denoted as $\boldsymbol{E}_{\text{gene}}$ and concatenate it with $\boldsymbol{E}_{\text{value}}$. We feed the concatenated embeddings into a linear layer and obtain $\boldsymbol{E} \in \mathbb{R}^{N_g \times N}$. To save computational cost, we introduce a symmetric pooling function denoted as $f(\cdot)$, that reduces the matrix $\boldsymbol{E}$ to a single N dimensional vector. Widely adopted options to achieve this include max or mean pooling. However, due to the high sparsity of the vector $\boldsymbol{x}$, where only a small subset of genes has non-zero expression values, employing these conventional pooling techniques results in suboptimal outcomes. As a remedy, we design a hierarchical pooling strategy to relieve the issue brought by excessive zero-count genes, where the function f first processes embeddings corresponding to genes with zero-count expressions. Subsequently, f pools the embeddings of the remaining genes, while incorporating the pooled embeddings of the zero-count genes. After

obtaining the pooled embeddings, we use them as the coefficients to interpolate N learnable vectors, denoted as $\boldsymbol{P}' \in \mathbb{R}^{N \times d}$. This can enhance the representation ability of our embedding process. We summarize the whole process in Algorithm 2.

3.2. Genetic Pathway Learning

Most genes do not function in isolation; instead, they function in concert to perform biological functions. Groups of biologically related genes that demonstrate substantial associations with specific biological processes are commonly referred to as pathways. Recognizing the activated pathways within a cell holds paramount importance in comprehending its characteristics (Wang & Sherwood, 2011).

We propose to learn distinct "*pathway*" tokens, represented as *discrete* codes, by training an encoder and a vector quantizer. To be more specific, 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 $\boldsymbol{o}$. Furthermore, we introduce a commitment loss (Huh et al., 2023) to minimize the distance between each pair of $\boldsymbol{e}_i$ and $\boldsymbol{z}_i$. Additionally, we follow Huh et al. (2023) to regularly replace the unused tokens in codebooks with randomly re-initialized tokens, and leverage an affine parameterization to minimize interval covariate shifts. More details are provided in Section A.2.

3.3. 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 (Section B.7).

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. This token is associated with a subset, denoted as $\boldsymbol{z}_{i_1}, \boldsymbol{z}_{i_2}, \ldots, \boldsymbol{z}_{i_{N'}}$, randomly selected with monotonically increasing indices from the set $\mathcal{Z}$. This subset operation is analogous to the masking concept, where information is transferred from the masked tokens to $\boldsymbol{e}_C$. We employ an encoder to convert these tokens into representations, which we denote as $\mathcal{H} = \{\boldsymbol{h}_C, \boldsymbol{h}_{i_1}, \boldsymbol{h}_{i_2}, \ldots, \boldsymbol{h}_{iN'}\}$. Subsequently, we exclude $\boldsymbol{h}_C$, replace the positions of the previously omitted pathway tokens (*i.e.,* those not selected in the subset) with a common token $\boldsymbol{e}_M$, and proceed to decode this modified embedding using a decoder. To achieve the aggregation, we train the two introduced tokens, namely $\boldsymbol{e}_C$ and $\boldsymbol{e}_M$, as well as two neural networks, namely the encoder and the decoder. Note that the encoder and the quantizer introduced in Section 3.2 are frozen. The optimization is achieved by minimizing the reconstruction loss between the original gene expression and the decoded representation within the masked region. Despite the fact that the number of pathway tokens differs from the number of genes, we have found that a straightforward scaling algorithm is effective (see Algorithm 1). The experimental results confirm that the stronger capability of $\boldsymbol{h}_e$ in representing cells compared to other variants such as the average of $\boldsymbol{z}_i$ after training. To further encourage $\boldsymbol{h}_C$ to learn cell-specific information, we devise a contrastive learning framework, as presented in Section A.1.

4. Experiments

4.1. Implementation Details

We provide details on benchmark datasets below, and defer other details to Section B.1. We compare against baseline methods on various benchmark datasets that are not included in the pretraining data: (1) CLL (GEO: GSE111014) (Rendeiro et al., 2020), which originally contains 48016 cells with 33694 genes and 6 types of cells. We further filter out cells without type annotations, resulting in 30K cells; (2) COVID (GEO: GSE150861) (Guo et al., 2020), which contains 11931 cells; (3) Jurkat from 10x Genomics, which contains 3258 cells. We filter out zero-count genes and retain 17753 genes. (4) PBMC-5k that also comes from 10x Genomics which contains around 5K cells; and finally the Cancer Cell Line Encyclopedia (CCLE) (Barretina et al., 2012) and Genomics of Drug Sensitivity in Cancer (GDSC) (Iorio et al., 2016) datasets which are leveraged in Section 4.3.

4.2. 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 three baselines, namely PCA, Geneformer, and scGPT. These experiments are conducted on the CLL and the COVID dataset. Figure 1 and 3 display UMAP visualizations created from the cell representations, *i.e.,* $\boldsymbol{h}_C$. We use the Leiden (Traag et al., 2019) algorithm with a resolution of 1.0 to cluster the embeddings, and assess the clustering performance with the adjusted rand index (ARI) and normalized mutual information (NMI) scores. Qualitatively speaking, PCA exhibits the

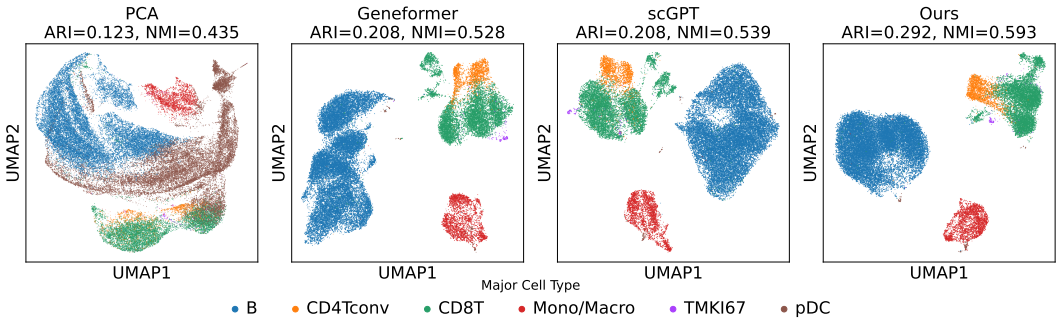

Figure 1. Unsupervised clustering performance on the CLL dataset.

poorest results, whereas the UMAPs generated by the other three models demonstrate significantly superior clustering quality. We can also observe that **scPaLM** possesses a smoother and more clustered latent space with respect to the ground-truth cell type labels. Quantitative results also confirm **scPaLM**'s ability to annotate types of cells. Notably, our method achieves higher ARI and NMI scores compared to the baselines by clear margins on both datasets. On CLL, **scPaLM** outperforms the baselines by $0.084 \sim 0.169$ in terms of the ARI score and $0.054 \sim 0.158$ in terms of the NMI score. Similarly, on COVID, it outperforms the baselines by $0.030 \sim 0.356$ in terms of the ARI score and $0.002 \sim 0.247$ in terms of the NMI score. These improvements indicate the high quality of produced embeddings.

4.3. 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. From Table 1, we can observe that both scFoundation and our method outperform the baseline framework DeepCDR significantly and achieve a stronger correlation between the prediction and the IC50 values. Notably, when using no additional information from the mutation and methylation, our method significantly outperforms scFoundation by 5% in terms of the Pearson Correlation Coefficient (PCC). Having additional information, all the methods demonstrate higher PCCs, yet our method remains to be the top performer among all the methods. To have a better understanding of the performance gain, we provide pairwise visualization of the correlation achieved by our method and scFoundation in Figure 4. In these experiments, 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. From those figures, we can observe that **scPaLM** achieves better PCCs on all but one cancer type, and improves the metrics on a majority of cell lines. Following the analysis, we further visualize the best prediction case of the cancer type, namely the low-grade gliomas (LGG) in Figure 5, where we observe both methods achieve high PCC values despite that the IC50 values have a large range from -6 to 6. **scPaLM** outperforms scFoundation by 2% and 4% in terms of the PCC and the Spearman correlation coefficient. These results showcase the effectiveness of **scPaLM**. It is also noteworthy that the embeddings generated by **scPaLM** are smaller in dimension compared to those of scFoundation, which implies that **scPaLM** are more efficient in modeling scRNA-seq data.

Table 1. Comparison of Pearson correlation coefficient (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.

Settings		Method		
Use Mut	Use Methy	DeepCDR	scFoundation	Ours
✗	✗	83.79	86.11	91.26
✓	✗	92.11	92.15	92.28
✓	✓	92.38	92.45	92.46

5. Conclusion

This work presents **scPaLM**, a foundation model pre-trained on single-cell RNA-seq data. We devise several novel techniques that efficiently represent gene expression values into tokens, model the collective function of genes, and effectively aggregate cell-specific information into a single token. We evaluate **scPaLM** on a wide range of downstream tasks, and demonstrate it reaches SoTA.

References

Barretina, J., Caponigro, G., Stransky, N., Venkatesan, K., Margolin, A. A., Kim, S., Wilson, C. J., Lehár, J., Kryukov, G. V., Sonkin, D., et al. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, 2012.

Consortium*, T. T. S., Jones, R. C., Karkanias, J., Krasnow, M. A., Pisco, A. O., Quake, S. R., Salzman, J., Yosef, N., Bulthaup, B., Brown, P., et al. The tabula sapiens: A multiple-organ, single-cell transcriptomic atlas of humans. *Science*, 376(6594):eabl4896, 2022.

Cui, H., Wang, C., Maan, H., Pang, K., Luo, F., and Wang, B. scgpt: Towards building a foundation model for single-cell multi-omics using generative ai. *bioRxiv*, pp. 2023–04, 2023.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dijk, D. v., Nainys, J., Sharma, R., Kaithail, P., Carr, A. J., Moon, K. R., Mazutis, L., Wolf, G., Krishnaswamy, S., and Pe'er, D. Magic: A diffusion-based imputation method reveals gene-gene interactions in single-cell rna-sequencing data. *BioRxiv*, pp. 111591, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Eraslan, G., Simon, L. M., Mircea, M., Mueller, N. S., and Theis, F. J. Single-cell rna-seq denoising using a deep count autoencoder. *Nature communications*, 10(1):390, 2019.

Fabregat, A., Jupe, S., Matthews, L., Sidiropoulos, K., Gillespie, M., Garapati, P., Haw, R., Jassal, B., Korninger, F., May, B., et al. The reactome pathway knowledgebase. *Nucleic acids research*, 46(D1):D649–D655, 2018.

Gaublomme, J. T., Yosef, N., Lee, Y., Gertner, R. S., Yang, L. V., Wu, C., Pandolfi, P. P., Mak, T., Satija, R., Shalek, A. K., et al. Single-cell genomics unveils critical regulators of th17 cell pathogenicity. *Cell*, 163(6):1400–1412, 2015.

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

Guo, C., Li, B., Ma, H., Wang, X., Cai, P., Yu, Q., Zhu, L., Jin, L., Jiang, C., Fang, J., et al. Single-cell analysis of two severe covid-19 patients reveals a monocyte-associated and tocilizumab-responding cytokine storm. *Nature communications*, 11(1):3924, 2020.

Hao, M., Gong, J., Zeng, X., Liu, C., Guo, Y., Cheng, X., Wang, T., Ma, J., Song, L., and Zhang, X. Large scale foundation model on single-cell transcriptomics. *bioRxiv*, pp. 2023–05, 2023.

Huang, M., Wang, J., Torre, E., Dueck, H., Shaffer, S., Bonasio, R., Murray, J. I., Raj, A., Li, M., and Zhang, N. R. Saver: gene expression recovery for single-cell rna sequencing. *Nature methods*, 15(7):539–542, 2018.

Huh, M., Cheung, B., Agrawal, P., and Isola, P. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. *arXiv preprint arXiv:2305.08842*, 2023.

Iorio, F., Knijnenburg, T. A., Vis, D. J., Bignell, G. R., Menden, M. P., Schubert, M., Aben, N., Gonçalves, E., Barthorpe, S., Lightfoot, H., et al. A landscape of pharmacogenomic interactions in cancer. *Cell*, 166(3):740–754, 2016.

Jovic, D., Liang, X., Zeng, H., Lin, L., Xu, F., and Luo, Y. Single-cell rna sequencing technologies and applications: A brief overview. *Clinical and Translational Medicine*, 12(3):e694, 2022.

Li, W. V. and Li, J. J. An accurate and robust imputation method scimpute for single-cell rna-seq data. *Nature communications*, 9(1):997, 2018.

Liu, Q., Hu, Z., Jiang, R., and Zhou, M. Deepcdr: a hybrid graph convolutional network for predicting cancer drug response. *Bioinformatics*, 36(Supplement_2):i911–i918, 2020.

Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., and Yosef, N. Deep generative modeling for single-cell transcriptomics. *Nature methods*, 15(12):1053–1058, 2018.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Patel, A. P., Tirosh, I., Trombetta, J. J., Shalek, A. K., Gillespie, S. M., Wakimoto, H., Cahill, D. P., Nahed, B. V., Curry, W. T., Martuza, R. L., et al. Single-cell rna-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science*, 344(6190):1396–1401, 2014.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rendeiro, A. F., Krausgruber, T., Fortelny, N., Zhao, F., Penz, T., Farlik, M., Schuster, L. C., Nemc, A., Tasnády, S., Réti, M., et al. Chromatin mapping and single-cell immune profiling define the temporal dynamics of ibrutinib response in cll. *Nature Communications*, 11(1):577, 2020.

Semrau, S., Goldmann, J. E., Soumillon, M., Mikkelsen, T. S., Jaenisch, R., and Van Oudenaarden, A. Dynamics of lineage commitment revealed by single-cell transcriptomics of differentiating embryonic stem cells. *Nature communications*, 8(1):1096, 2017.

Theodoris, C. V., Xiao, L., Chopra, A., Chaffin, M. D., Al Sayed, Z. R., Hill, M. C., Mantineo, H., Brydon, E. M., Zeng, Z., Liu, X. S., et al. Transfer learning enables predictions in network biology. *Nature*, pp. 1–9, 2023.

Traag, V. A., Waltman, L., and Van Eck, N. J. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):5233, 2019.

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, Z. and Sherwood, D. R. Dissection of genetic pathways in c. elegans. *Methods in cell biology*, 106:113–157, 2011.

Wen, H., Tang, W., Dai, X., Ding, J., Jin, W., Xie, Y., and Tang, J. Cellplm: Pre-training of cell language model beyond single cells. *bioRxiv*, pp. 2023–10, 2023.

Yang, F., Wang, W., Wang, F., Fang, Y., Tang, D., Huang, J., Lu, H., and Yao, J. 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, 2022.

A. Details on Methodology

A.1. Token-level Contrastive Learning

Our objective is to have diverse cell types manifest distinct representations in line with their unique biological characteristics. However, due to the absence of cell type annotations for scRNA-seq data modeling during training time, we employ K-Means (Lloyd, 1982) to leverage $\boldsymbol{h}_C$ to assign pseudo labels to each cell to enable the construction of positive and negative pairs. In each batch, we generate two distinct sets of representations, denoted as $\mathcal{H}_i$ and $\mathcal{H}'_i$, for every gene expression vector $\boldsymbol{x}_i$ where i indicates the index within a batch. These sets are generated by sampling different subsets of indices, therefore implying different representations of the same cells. We use $\boldsymbol{h}_{i,C}$ to indicate the first representation in $\mathcal{H}_i$, which corresponds to that of $\boldsymbol{e}_C$ summarizing the cell information. Such embeddings from the same cell, *i.e.*, $\boldsymbol{h}_{i,C}$ and $\boldsymbol{h}'_{i,C}$, are considered as positive pairs. Conversely, when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ have different assigned cluster labels, we consider $\boldsymbol{h}_{i,C}$ and $\boldsymbol{h}_{j,C}$ as negative pairs. To ensure training efficiency, we limit the number of sampled negative pairs to K. The regularization based on these positive and negative pairs can be expressed as:

$$\mathcal{L}_{CL} = \sum_i -\log \frac{s(\boldsymbol{h}_{i,C}, \boldsymbol{h}'_{i,C})/\tau}{s(\boldsymbol{h}_{i,C}, \boldsymbol{h}'_{i,C}) + \sum_{j \in \texttt{neg}(i)} s(\boldsymbol{h}_{j,C}, \boldsymbol{h}_{i,C})}, \tag{1}$$

where $s(\cdot, \cdot)$ indicates the cosine similarity and $\texttt{neg}(\cdot)$ provides the indices of negative samples. During training, we maintain a fixed length queue Q to store the derived $\boldsymbol{h}_C$ and periodically update pseudo-label assignments for cells by re-running the K-Means on Q to adapt to changes in their representations. The detailed pipeline is in Algorithm 3.

A.2. 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. (2023). Firstly, they introduce an affine transformation to reparameterize the representation in the codebook with the following formula:

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

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 modification to the codebook update process to enhance the stability.

A.3. Mask Construction and Output Reshaping

In Algorithm 1, we introduce a simple way to make the shape of the output from the decoder introduced in Section 3.3 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.

Algorithm 1 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 $\mathcal{L}_{\text{MSE}} = \|\boldsymbol{o} - \boldsymbol{x}\|_2^2 / \|\boldsymbol{m}\|_1$.

A.4. Association of Genes with Pathway Tokens

In this section, we describe the methodology employed 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.

A.5. Overall Framework

Figure 2 demonstrates the overall framework of **scPaLM**.

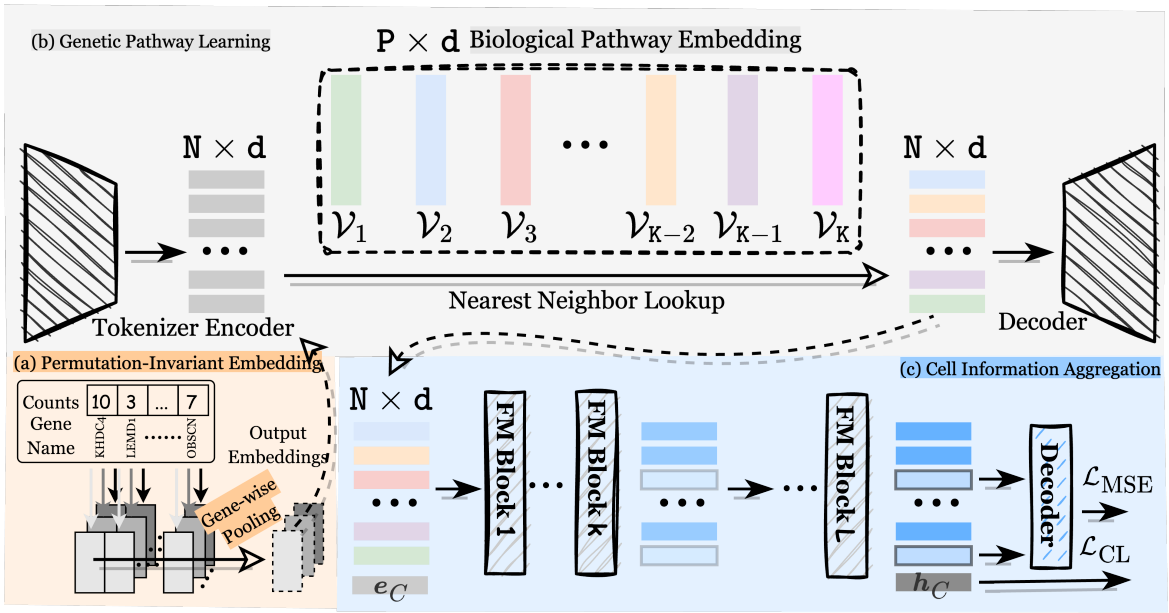

Figure 2. The overview of our framework. Three main innovative components are introduced in our framework: an efficient embedding process that is permutation-invariant; a module aims to capture genetic pathways; and a training framework for cell information aggregation.

Algorithm 2 Permutation-Invariant Embedding

Input: A batch of normalized expression value vectors $\boldsymbol{X} \in \mathbb{R}^{B \times N_g}$
Output: Embedding of $\boldsymbol{X}$
$\boldsymbol{X} \leftarrow \texttt{einsum("bi,ij->bij"}, \boldsymbol{X}, \boldsymbol{P})$
$\boldsymbol{X} \leftarrow \texttt{leaky\_relu}(\boldsymbol{X})$
$\boldsymbol{E}_{\text{value}} \leftarrow \texttt{einsum("ij,bij->bij"}, \boldsymbol{\alpha}, \boldsymbol{X}) + \texttt{MLP}(\boldsymbol{X})$
$\boldsymbol{E} \leftarrow \texttt{concat}(\boldsymbol{E}_{\text{value}}, \boldsymbol{E}_{\text{gene}})\ \{\boldsymbol{E} \in \mathbb{R}^{B \times N_g \times N}\}$
$\bar{\boldsymbol{E}} \leftarrow \texttt{einsum("bj,jl->bjl"}, f(\boldsymbol{E}), \boldsymbol{P}')$
return $\bar{\boldsymbol{E}}$

Algorithm 3 demonstrates the algorithm for training **scPaLM**for one step.

B. More Experimental Settings and Results

B.1. Experimental Settings

Pretraining Data. **scPaLM** is pretrained on single-cell RNA-seq data covering different types of cells, having in total 0.5M cells and around 60K genes. The statistics and description of the pre-training data are in Appendix C.

Baselines. We compare **scPaLM** with various baseline methods on different tasks. For cell-type annotation, we compare with PCA (where we derive the first 256 principal components on the log-normalized expression values), Geneformer (Theodoris et al., 2023), scGPT (Cui et al., 2023). The latter two are current state-of-the-art algorithms on this task. For imputation, we compare with MAGIC (Dijk et al., 2017), SAVER (Huang et al., 2018), scImpute (Li & Li, 2018),

Algorithm 3 Training Pipeline (One Step)

Input: A batch of gene expression vectors $\boldsymbol{X} \in \mathbb{N}^{B \times N_g}$, current time stpe 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 Section 3.1.
Obtain $\mathcal{Z} \in \mathbb{R}^{B \times N \times d}$ according to Section 3.2.
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**
    Re-fit K based on embeddings in Q.
  end if
  Randomly select $p\%$ tokens from $\mathcal{Z}$ for each sample and prepend a token $\boldsymbol{e}_C$. Obtain $\mathcal{H}'$.
  Calculate the contrastive learning loss according to Equation 1.
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.

DCA (Eraslan et al., 2019), which are widely used methods for this task. For drug response prediction, we compare with another transformer-based algorithm, scFoundation (Hao et al., 2023).

Architectures. The overall architecture of **scPaLM** can be split into three parts: (1) embedding layers, where we use mainly MLPs as described in Algorithm 2. 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 of 6 layers and a hidden dimension of 256. 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. Table 2 demonstrates the hyperparameters on the structure of **scPaLM**.

Table 2. Configurations of our **scPaLM**.

Hyperparameters	Value
Hidden Size	256
Intermediate Size	1024
Number of Layers	6
Number of Attention Heads	8
Dropout Probability	0.0
Attention Dropout Probability	0.0

Training Settings. The optimizer we use in our experiments is `AdamW` (Loshchilov & 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. In the second stage, we train other components with a learning rate of 5×10^{-4} and using the same batch size of 128.

B.2. Unsupervised Cell Type Annotation

Figure 3 provides a UMAP visualization on the embeddings extracted from the COVID dataset.

B.3. Cancer Drug Response

We provide pairwise visualization of the correlation achieved by our method and scFoundation in Figure 4. We further visualize the best prediction case of the cancer type, namely the low-grade gliomas (LGG) in Figure 5.

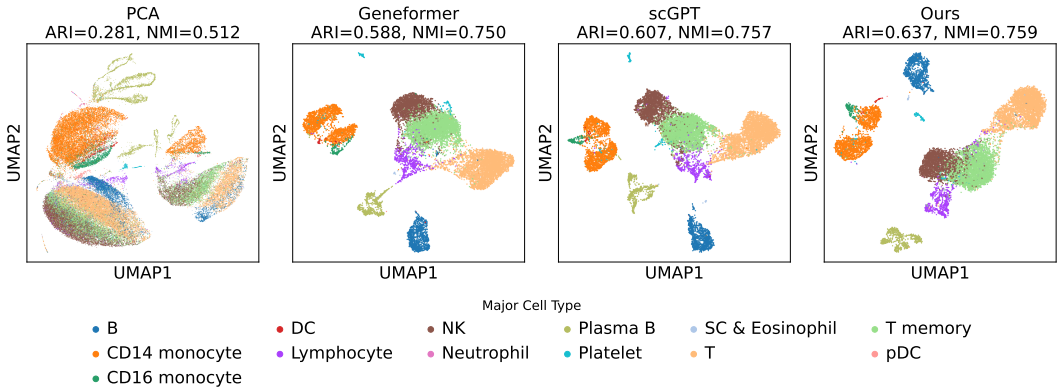

Figure 3. Unsupervised clustering performance on the COVID dataset.

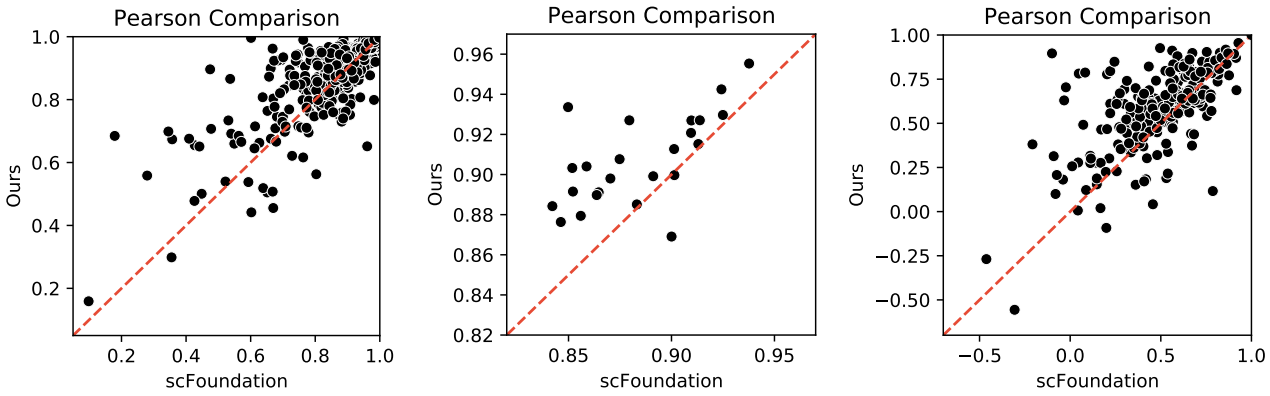

Figure 4. Pairwise visualization of the Pearson correlation coefficient of scFoundation and **scPaLM** based on different grouping strategies. Left: grouping with respect to the cell lines; Middle: grouping with respect to the cancer type; Right: grouping with respect to the drug type. The red lines indicate the relationship of $y = x$.

B.4. Imputation

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). We conduct a series of simulated experiments on the Jurkat and the PBMC dataset to assess **scPaLM**'s ability in accurately predicting the missing genes' expression levels. We randomly sample 10% of genes from each cell with a probability that is proportional to the exponent of negative genes' expression values and mask them as 0.

Table 3. Imputation performance of various methods on the Jurkat and the PBMC dataset. We report the rooted mean square error (RMSE) and the mean absolute error (MAE).

Method	Jurkat		PBMC	
	RMSE	MAE	RMSE	MAE
SAVER	0.841	0.664	0.779	0.594
MAGIC	0.449	0.379	0.656	0.548
scImpute	1.178	0.838	1.528	1.132
DCA	0.937	0.629	0.833	0.638
scPaLM (Zero Shot)	0.494	0.397	0.674	0.539

Table 3 presents the rooted mean square error (RMSE) and the mean absolute error (MAE) between the ground-truth and the predicted expression values on the masked genes across different cells. Note that these metrics are calculated based on the log-normalized expression values. Even under a zero-shot setting, **scPaLM** achieves superior performance compared to most baselines, which estimate their parameters on the downstream datasets. This experiment confirms **scPaLM**'s ability in denoising the expression data and capturing the interactions between cells and genes.

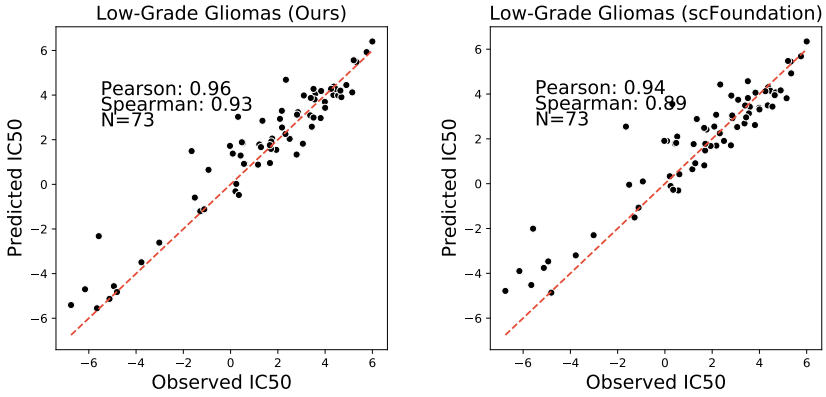

Figure 5. Scatter plots of the predicted and the observed IC50 values on the samples with the cancer type of low-grade gliomas.

Table 4. Significant pathways identified by GSEA (p-value $< 1 \times 10^{-5}$). Two pathways related to the immune system are selected.

Gene Lists (Token ID)	Term	P-value
CD1E, TREM2, ICAM5, CD1B (25)	Immunoregulatory Interactions Between A Lymphoid And A non-Lymphoid Cell	2.8×10^{-7}
BTN1A1, MRC1, CD1E, TREM2, ICAM5, CD1B (25)	Adaptive Immune System	4.4×10^{-7}

B.5. Genetic Pathway Identification

We finally conduct an experiment to understand the obtained pathway tokens from scRNA-seq datasets. We follow the setting from scGPT (Cui et al., 2023) where we aim to identify genetic pathways on the Immune Human dataset. To associate a gene with a certain pathway token, we first derive the $\mathcal{V}$ for each gene, and for every v we calculate the associated gene expression vector weighted by the occurrence percentage of v for each cell. Finally, for every v we obtain the list of associated genes by calculating the relative prevalence of genes. A more detailed algorithm is deferred in Section A.4. We associate 10 genes to each pathway token and run the gene set enrichment analysis (GSEA) algorithm to search for pathways in Reactome Pathway Database (Fabregat et al., 2018). Note that this dataset is not involved in our training set, thereby it constitutes a zero-shot setting. Nevertheless, our method identifies two significant pathways related to the immune system, as shown in Table 4. Particularly, it identifies and clusters the CD1 gene family (CD1E and CD1B), which is involved in antigen presentation that is related to immune reaction.

B.6. Comparison of Different Embedding Processes.

We conduct an experiment to compare the memory usage and the subsequent clustering performance of models with different embedding processes on the COVID dataset. The results are presented in Table 5.

Table 5. Comparison between different embedding process. The models are trained on a subset of the pretraining data, and evaluated on the COVID dataset.

Embedding Algorithm	Memory Usage	ARI	NMI
Per-gene	>80G	-	-
Shared-first-layer	42378MiB	0.167	0.251
Ours	43678MiB	0.201	0.291

B.7. Ablation Studies

The effectiveness of the cell information aggregation process. We conduct a series of experiment 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 6 presents the performance of cell type annotations, where we can observe that

Table 6. Clustering performance of different variants for cell representations on the CLL dataset. We compare the adjusted rand index (ARI), normalized mutual information (NMI), silhouette score (S-score), and clustering time between models.

Method	ARI	NMI	S-score
Mean	0.015	0.059	−0.133
Concatenated	0.181	0.478	0.310
$\boldsymbol{h}_C$ (No CL)	0.275	0.573	0.361
$\boldsymbol{h}_C$	**0.292**	**0.593**	**0.376**

using our cell information aggregation technique yields the best performance. We have also conducted an experiment where we do not use the token-level contrastive learning framework to train the embedding of $\boldsymbol{h}_C$. The decreased scores of these experiments 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 Section 3.2. Instead, 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 7 demonstrates the importance of the encoder and the quantizer. We see that the introduced genetic pathway encoder helps improve the clustering performance, improving the metrics by 0.14 and 0.16, respectively. The usage of the quantizer also further improves the performance by an additional 1%.

Table 7. Comparisons on COVID with different configurations. The models are trained on a small subset of the pre-training data.

Configuration		ARI	NMI
Encoder	Quantizer		
✗	✗	0.050	0.120
✓	✗	0.191	0.286
✓	✓	0.201	0.291

The effectiveness of the embedding process. To evaluate the effectiveness of our embedding process, we explore several alternatives and compare to our deployed embedding process: (1) *per-gene*, where we $\boldsymbol{E}$ as gene embeddings. 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 deploy only a shared $\boldsymbol{P}$ for all the genes. The results are presented in Table 5, 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.

C. Dataset Description

Our pretraining data comes from Tabula Sapiens (Consortium* et al., 2022), which has nearly $500,000$ cells from 24 organs of 15 normal human subjects. The cell-type annotations datasets are from patients with COVID and leukemia.