SINGLE CELLS ARE BIOLOGICAL TOKENS: TOWARDS CELL LANGUAGE MODELS

By

Hongzhi Wen

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Doctor of Philosophy

2024

ABSTRACT

The rapid advancement of single-cell technologies allows for simultaneous measurement of multiple molecular features within individual cells, providing unprecedented multimodal data through single-cell multi-omics and spatial omics technologies. This dissertation addresses the complex challenges of modeling these multimodal interactions using deep learning techniques. We propose two series of studies: the first, is the application of graph neural networks and graph transformers to model relations between multimodal features, incorporating external domain knowledge. We propose Single-cell Multi-Omics GNN (scMoGNN) and Single-cell Multi-Omics Transformer (scMoFormer), the latter extends the former one and demonstrates the prospect of transformers in single-cell multi-omics representation learning. The second is the application of transformers in spatial omics representation learning. We propose Spatial Transformer (SpaFormer), a transformer-based masked autoencoder learning framework for extracting cell context information and imputing spatial transcriptomics data. Despite the effectiveness of these models, their knowledge transferability across tasks and datasets remains limited. To overcome this, we introduce a new transformer-based foundation model, Cell Pre-trained Language Model (CellPLM), that encodes inter-cellular relations and multimodal features, demonstrating the significant potential of foundation models for future research in single-cell biology.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Jiliang Tang, for his invaluable guidance, inspiration, and unwavering support throughout my Ph.D. journey. Dr. Tang has been more than just an academic mentor; he has also been a true friend, offering wisdom and advice in many aspects of life. Joining Dr. Tang's research lab was one of the best decisions I made four years ago, as it has provided me with an enriching experience. Under his mentorship, I have acquired extensive knowledge, from effective research methodologies to leadership skills. Dr. Tang is among the finest advisors one could hope for, and I am profoundly grateful for his consistent encouragement and responsiveness in nurturing the growth of his students. I owe much of my success to his steadfast support over the years.

I would like to extend my deepest appreciation to my dissertation committee members, Dr. Guan-Hua Tu, Dr. Hui Liu, and Dr. Yuying Xie, for their insightful comments and invaluable suggestions. Their expertise and feedback have been instrumental in shaping both this dissertation and my future career path.

I am grateful to have had the pleasure and fortune of having supportive and encouraging colleagues during my Ph.D. I am thankful to all my colleagues from the Data Science and Engineering Lab: Dr. Jamell Dacon, Dr. Wei Jin, Jiayuan Ding, Wenzhuo Tang, Xinnan Dai, Dr. Remy Liu, Kaiqi Yang, Hang Li, Zhikai Chen, Haitao Mao, Yingqian Cui, Jie Ren, Dr. Wenqi Fan, Dr. Hamid Karimi, Dr. Tyler Derr, Dr. Yao Ma, Dr. Haochen Liu, Hua Liu, Dr. Xiaorui Liu, Dr. Wentao Wang, Dr. Yiqi Wang, Dr. Zhiwei Wang, Dr. Han Xu, Dr. Xiangyu Zhao, Haoyu Han, Harry Shomer, Kai Guo, Juanhui Li, Yaxin Li and Yuxuan Wan.

I would also like to extend my heartfelt appreciation to the DANCE team, my dear friends, and collaborators outside of MSU for their unwavering support and encouragement during the challenging moments of my Ph.D. journey. Their kindness and steadfast support helped me persevere through times of frustration. Finally, I would like to thank my family, for their unconditional love and support.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation

The rapid advance of single-cell technologies makes it possible to simultaneously measure multiple molecular features at multiple modalities in a cell, such as gene expressions, protein abundance, and chromatin accessibility. This is known as **single-cell multi-omics technologies**. For instance, CITE-seq (cellular indexing of transcriptomes and epitopes by sequencing) (Stoeckius et al., 2017) enables simultaneous quantification of mRNA expression and surface proteins abundance; methods like sci-CAR (Cao et al., 2018), Paired-seq (Zhu et al., 2019), and SNARE-seq (Chen et al., 2019) enable joint profiling of mRNA expression and chromatin accessibility (i.e. genome-wide DNA accessibility). The joint measurements from these methods provide unprecedented multimodal data for single cells, which has given rise to valuable insights into not only the relationship between different modalities but, more importantly, a holistic understanding of the cellular system. In addition to the aforementioned multi-omic technologies, recently, **single-cell spatial omics technologies** have emerged as another next-generation tool for biomedical research. For instance, in-situ hybridization (ISH) based technology (Lubeck et al., 2014) produces detailed single-cell transcriptomic profiles along with the location of cells within a tissue, yielding deeper insights into cell identity and functionality than ever.

The proliferation of these advanced single-cell data also introduces tremendous challenges in modeling the complex interactions among different modalities. As a result, there is an emerging trend to leverage deep learning techniques to handle multimodal single-cell data (Molho et al., 2024). For example, BABEL (Wu et al., 2021b) translated between the transcriptome (mRNA) and chromatin (DNA) profiles of a single cell based on an encoder-decoder architecture; scMM (Minoura et al., 2021) implemented a mixture-of-experts deep generative model for joint embedding learning and modality prediction. Cobolt (Gong et al., 2021) acquired joint embedding via a variant of Multimodal Variational Autoencoder (MVAE) (Yao et al., 2021). While there are a growing number of deep learning frameworks for modeling multimodal single-cell data, most of them treat each cell

as a separate input without considering possible high-order interactions among cells or different modalities. This makes them not optimal for multi-omics technology and spatial omics technology. In light of this, we propose two series of studies to fill up the gaps in terms of two perspectives. **First**, we study how to model the complex relations between multi-modal features. Specifically, we explore the application of graph neural networks and graph transformers on cell-feature heterogeneous graphs, which can readily incorporate external domain knowledge and model the interactions within each modality and cross-modalities. **Second**, we study how to model the complex interactions between cells. Specifically, we propose a transformer-based framework for spatial transcriptomic data and leverage positional encodings to help extract cell context information. Through these two studies, we demonstrate the effectiveness and advantages of deep learning models, particularly, transformers, in modeling single-cell multimodal data.

Despite that the proposed models are effective in handling multimodal data, the knowledge learned by models is not transferable across tasks and datasets. To address this issue, there is an emerging effort (Yang et al., 2022; Gong et al., 2023; Shen et al., 2023; Cui et al., 2023; Theodoris et al., 2023) from the research community to explore the potential of **a foundation model** that first extracts latent knowledge from unlabeled data and subsequently generalizes this knowledge to a variety of tasks. Existing foundation models all regard genes as tokens and focus solely on modeling gene relationships within individual cells, neglecting the inter-cellular information in an organism, as well as the abundant multi-modal resources. Therefore, a need has arisen to extend the foundation model to multimodal applications beyond single cells. Based on our previous studies, we propose a new transformer-based foundation model to encode inter-cellular relations and multimodal features. Our research on various downstream tasks demonstrates the power of this foundation model, which has great potential to facilitate future research in single-cell biology.

1.2 Contributions

This dissertation addresses the complex challenges of modeling single-cell multi-omics and spatial omics data using deep learning techniques. First, we study the representation learning for multi-omics data, explore the application of graph neural networks and transformers, and

highlight the prospect of transformers in single-cell analysis. Second, we study the representation learning of spatial omics with transformer architecture and demonstrate the effectiveness of masked autoencoders in spatial omics learning. Based on these findings, we further develop a foundation model for single-cell data analysis. The major contributions of this dissertation can be summarized as follows:

- We conduct research on designing transformer-based frameworks for single-cell spatial and multimodal data. Based on the studies, we further propose a multimodal foundation model for single-cell analysis.
- In Chapter 2, we explore the application of graph neural networks on cell-feature heterogeneous graphs for multi-modal single-cell data, named *scMoGNN* (Wen et al., 2022a). *scMoGNN* can readily incorporate external domain knowledge and model the interactions within each modality and cross-modalities.
- In Chapter 3, we extend our studies from graph neural networks (i.e., *scMoGNN*) to graph transformers (i.e., *scMoFormer* (Tang et al., 2023)) for multi-omics data analysis, which not only enhances performance but also further highlights the potential of transformers in multi-omics representation learning.
- In Chapter 4, we propose a transformer-based model for spatial transcriptomic data, *SpaFormer* (Wen et al., 2023), and leverage positional encodings to facilitate cell-cell relation identification and data imputation. Our results demonstrate the high efficiency and performance of a masked autoencoder backbone for spatial omics representation learning.
- In Chapter 5, by unifying the explored architectures, we propose a new transformer-based foundation model for multimodal single-cell data, *CellPLM* (Wen et al., 2024), which can encode spatial and transcriptomics features as well as cell-cell relations.

CHAPTER 2

GRAPH NEURAL NETWORKS FOR SINGLE-CELL MULTI-OMICS REPRESENTATION LEARNING

Recent advances in multimodal single-cell technologies have enabled simultaneous acquisitions of multiple omics data from the same cell, providing deeper insights into cellular states and dynamics. However, it is challenging to learn the joint representations from the multimodal data, model the relationship between modalities, and, more importantly, incorporate the vast amount of single-modality datasets into the downstream analyses. To address these challenges and correspondingly facilitate multimodal single-cell data analyses, three key tasks have been introduced: *Modality prediction*, *Modality matching*, and *Joint embedding*. In this work, we present a general Graph Neural Network framework *scMoGNN* to tackle these three tasks and show that *scMoGNN* demonstrates superior results in all three tasks compared with the state-of-the-art and conventional approaches. Our method is an official winner in the overall ranking of *Modality prediction* from NeurIPS 2021 Competition[1], and all implementations of our methods have been integrated into DANCE package[2].

2.1 Chapter Introduction

The rapid advance of single-cell technologies makes it possible to simultaneously measure multiple molecular features at multiple modalities in a cell, such as gene expressions, protein abundance and chromatin accessibility. For instance, CITE-seq (cellular indexing of transcriptomes and epitopes by sequencing) (Stoeckius et al., 2017) enables simultaneous quantification of mRNA expression and surface proteins abundance; methods like sci-CAR (Cao et al., 2018), Paired-seq (Zhu et al., 2019), and SNARE-seq (Chen et al., 2019) enable joint profiling of mRNA expression and chromatin accessibility (i.e. genome-wide DNA accessibility). The joint measurements from these methods provide unprecedented multimodal data for single cells, which has given rise to valuable insights for not only the relationship between different modalities but, more importantly, a holistic understanding of the cellular system.

Despite the emergence of joint platforms, single-modality datasets are still far more prevalent.

[1]The competition official website is https://openproblems.bio/neurips_2021.

[2]Our DANCE package is released at https://github.com/OmicsML/dance.

How to effectively utilize complementary information from multimodal data to investigate cellular states and dynamics and to incorporate the vast amount of single-modality data while leveraging the multimodal data pose great challenges in single-cell genomics. To address these challenges, Luecken (Luecken et al., 2021) summarized three major tasks: (1) *Modality prediction* aims at predicting the features of one modality from the features of another modality (Wu et al., 2021b); (2) *Modality matching* focuses on identifying the correspondence of cells between different modalities (Welch et al., 2017); and (3) *Joint embedding* requires embedding the features of two modalities into the same low-dimensional space (Stoeckius et al., 2017). The motivation of modality prediction and modality matching is to better integrate existing single-modality datasets, while joint embedding can provide more meaningful representations of cellular states from different types of measurements. In light of these benefits, computational biologists recently organized a competition for multimodal single-cell data integration at NeurIPS 2021 (Luecken et al., 2021) to benchmark these three tasks and facilitate the computational biology communities.

There is an emerging trend to leverage deep learning techniques to tackle the tasks mentioned above for multimodal single-cell data (Molho et al., 2024). BABEL (Wu et al., 2021b) translated between the transcriptome (mRNA) and chromatin (DNA) profiles of a single cell based on an encoder-decoder architecture; scMM (Minoura et al., 2021) implemented a mixture-of-experts deep generative model for joint embedding learning and modality prediction. Cobolt (Gong et al., 2021) acquired joint embedding via a variant of Multimodal Variational Autoencoder (MVAE) (Yao et al., 2021). MOFA2 (Argelaguet et al., 2020) used Bayesian group factor analysis to reduce dimensions of multi-modality data and generate a low-dimensional joint representation. However, most of these approaches treat each cell as a separate input without considering possible high-order interactions among cells or different modalities. Such higher-order information can be essential for learning with high-dimensional and sparse cell features, which are common in single-cell data. Take the joint embedding task for example, the feature dimensions for GEX (mRNA) and ATAC (DNA) data are as high as 13,431 and 116,490, respectively; however, only 9.75% of GEX and 2.9% of ATAC features are nonzero on average over $42,492$ training samples (cells). Furthermore, integrated

measuring often requires additional processing to cells, which can lead to extra noise and drop-out in the resulting data (Lee et al., 2020; Mimitou et al., 2021). Therefore, it is a desired technique that can mitigate the negative impact of such noise.

Recently, the advances in graph neural networks (GNNs) (Kipf and Welling, 2017; Wu et al., 2020; Battaglia et al., 2018; Gilmer et al., 2017; Liu et al., 2021b) pave the way for addressing the aforementioned issues in single-cell data integration. Specifically, GNNs aggregate information from neighborhoods to update node embeddings iteratively (Gilmer et al., 2017). Thus, the node embedding can eventually encode high-order structural information through multiple aggregation layers. In addition, GNNs smooth the features by aggregating neighbors' embedding and also filter the eigen-values of graph Laplacian, which provides an extra denoising mechanism (Ma et al., 2021b). Hence, by modeling the interactions between cells and their features as a graph, we can adopt GNNs to exploit the structural information and tackle the limitations of previous techniques for single-cell data integration. With the constructed graph, we can readily incorporate external knowledge (e.g., interactions between genes) into the graph to serve as additional structural information. Moreover, it enables a transductive learning paradigm with GNNs to gain additional semi-supervised signals to enhance representation learning.

Given those advantages, we aim to design a GNN framework for multimodal data integration. While several existing works attempted to introduce graph neural networks to single cell analysis (Song et al., 2021; Wang et al., 2021; Ciortan and Defrance, 2022; Shao et al., 2021), none of them tackle the challenging problem of multimodal data integration which requires handling different modalities simultaneously. Therefore, we aim to develop GNN methods for the tasks in multimodal data integration, especially for modality prediction, modality matching and joint embedding. Specifically, we propose a general framework *scMoGNN* for modeling interactions of modalities and leveraging GNNs in single-cell analysis[3]. Our framework is highly versatile: we demonstrate its use cases in the three different multimodal tasks. To the best of our knowledge, we are the first to develop a GNN framework in this emerging research topic, i.e., multimodal single-cell data integration. Our

[3]Our solution won the first place of the modality prediction task in the Multimodal Single-Cell Data Integration competition at NeurIPS 2021.

proposed framework achieves the best results in all of these three tasks on the benchmark datasets, providing a very strong baseline for follow-up research. Our contributions can be summarized as follows:

1. We study the problem of multimodal single-cell data integration and propose a general GNN-based framework *scMoGNN* to capture the high-order structural information between cells and modalities.
2. The proposed general framework is highly flexible as it can be adopted in different multimodal single-cell tasks.
3. Our framework achieves remarkable performance across tasks. It has won the first place of the modality prediction task in the Multimodal Single-Cell Data Integration competition, and currently outperforms all models for all three tasks on the leaderboard[4]. All of our results are based on publicly available data and are reproducible.

2.2 Problem Statement

Before we present the problem statement, we first introduce the notations used in this paper. There are three modalities spanning through each task. They are GEX as mRNA data, ATAC as DNA data and ADT as protein data. Each modality is initially represented by a matrix $\mathbf{M} \in \mathbb{R}^{N \times k}$ where N indicates the number of cells, and k denotes the feature dimension for each cell. In our work, we later construct a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ based on each modality $\mathbf{M}$, where $\mathcal{U}$ is the set of N cell nodes $\{u_1, u_2, ..., u_N\}$ and $\mathcal{V}$ is the set of k feature nodes $\{v_1, v_2, ..., v_k\}$.

With the aforementioned notations, the problem of learning GNNs for single-cell data integration is formally defined as, *Given a modality* $\mathbf{M} \in \mathbb{R}^{N \times k}$*, we aim at learning a mapping function* f_θ *which maps* $\mathbf{M}$ *to the space of downstream tasks.*

In the following, we formally define these three key tasks of single-cell data integration: modality prediction, modality matching and joint embedding. We will also define the corresponding evaluation metrics for each task. Note that these metrics are also adopted by the competition to decide the top winners.

[4]The leaderboard can be accessed at https://eval.ai/web/challenges/challenge-page/1111/leaderboard/2860.

2.2.1 Task 1: Modality Prediction

In this task, given one modality (like GEX), the goal is to predict the other (like ATAC) for all feature values in each cell. It can be formally defined as,

Given a source modality $\mathbf{M_1} \in \mathbb{R}^{N \times k1}$*, the goal is to predict a target modality* $\mathbf{M_2} \in \mathbb{R}^{N \times k2}$ *via learning a mapping function* f_θ *parameterized by* θ *such that* $\mathbf{M_2} = f_\theta(\mathbf{M_1})$.

Possible modality pairs of ($\mathbf{M_1}$, $\mathbf{M_2}$) are (GEX, ATAC), (ATAC, GEX), (GEX, ADT) and (ADT, GEX), which correspond to four sub-tasks in Task 1. Root Mean Square Error (RMSE) is used to quantify performance between observed and predicted feature values.

2.2.2 Task 2: Modality Matching

The goal of this task is to identify the correspondence between two single-cell profiles and provide the probability distribution of these predictions. It can be formally defined as,

Given modality $\mathbf{M_1} \in \mathbb{R}^{N \times k1}$ *and modality* $\mathbf{M_2} \in \mathbb{R}^{N \times k2}$*, we aim to learn two mapping functions* f_{θ_1} *parameterized by* θ_1 *and* f_{θ_2} *parameterized by* θ_2 *to map them into the same space such that*

$$\mathbf{S} = g(f_{\theta_1}(\mathbf{M_1}), f_{\theta_2}(\mathbf{M_2})) \tag{2.1}$$

where g *is a score function to calculate probability distribution of correspondence predictions.* $\mathbf{S} \in \mathbb{R}^{N \times N}$ *is an output score matrix with each row summing to 1.* $\mathbf{S_{ij}}$ *is the correspondence probability between* i*-th cell from modality* $\mathbf{M_1}$ *and* j*-th cell from modality* $\mathbf{M_2}$.

Possible modality pairs of ($\mathbf{M_1}$, $\mathbf{M_2}$) are (GEX, ATAC), (ATAC, GEX), (GEX, ADT) and (ADT, GEX), which correspond to four sub-tasks in Task 2. The sum of weights in the correct correspondences of $\mathbf{S}$ is used as final score to quantify prediction performance using $score = \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{S}_{i,j}$ if $i = j$.

2.2.3 Task 3: Joint Embedding

In this task, the goal is to learn an embedded representation that leverages the information of two modalities. The quality of the embedding will be evaluated using a variety of criteria generated from expert annotation. It can be formally defined as,

Given modality $\mathbf{M_1} \in \mathbb{R}^{N \times k1}$ *and modality* $\mathbf{M_2} \in \mathbb{R}^{N \times k2}$*, we aim to learn three mapping functions*

f_{θ_1}, f_{θ_2} and f_{θ_3} parameterized by θ_1, θ_2 and θ_3 accordingly to project them into downstream tasks,

$$\mathbf{H} = f_{\theta_3}\big(CONCAT(f_{\theta_1}(\mathbf{M_1}), f_{\theta_2}(\mathbf{M_2}))\big) \tag{2.2}$$

where $f_{\theta_1}(\mathbf{M_1}) \in \mathbb{R}^{N\times k1'}$ and $f_{\theta_2}(\mathbf{M_2}) \in \mathbb{R}^{N\times k2'}$ correspond to new representations learned from modality $\mathbf{M_1}$ and $\mathbf{M_2}$ separately, $\mathbf{H} \in \mathbb{R}^{N\times k3}$ is a final output embedding learned through f_{θ_3} on concatenation of $f_{\theta_1}(\mathbf{M_1})$ and $f_{\theta_2}(\mathbf{M_2})$.

H will be measured using six different metrics broken into two classes: biology conservation and batch removal. Biology conservation metrics include "NMI cluster/label", "Cell type ASW", "Cell cycle conservation" and "Trajectory conservation" which aim to measure how well an embedding conserves expert-annotated biology. Batch removal metrics include "Batch ASW" and "Graph connectivity" which are to evaluate how well an embedding removes batch variation. Please refer to the Appendix A.1 for more details about these metrics description. Possible modality pairs of ($\mathbf{M_1}$, $\mathbf{M_2}$) are (GEX, ATAC) and (ADT, GEX), which correspond to two sub-tasks in Task 3.

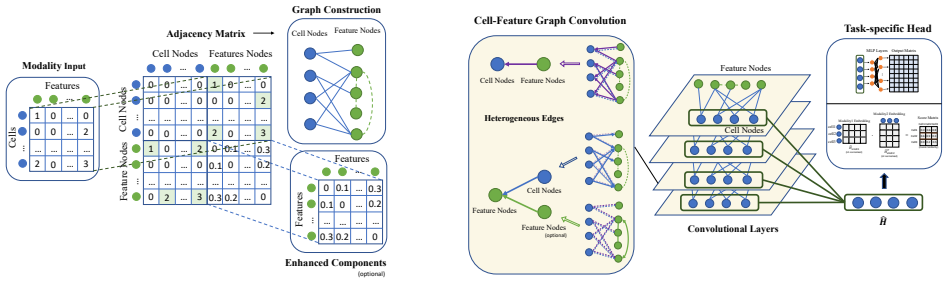
In this work, we instantiate f_θ as a graph neural network model by first constructing a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ based on modality **M**, and then learning better cell node representation through message passing on graphs.

2.3 Method

In this section, we first introduce the proposed general framework *scMoGNN* for multimodal data integration. Then we introduce how to adapt *scMoGNN* to advance three tasks: modality prediction, modality matching and joint embedding. An illustration of our framework is shown in Figure 2.1. Specifically, our framework can be divided into three stages: graph construction, cell-feature graph convolution, and task-specific head.

2.3.1 A General GNN Framework

To develop a GNN-based framework for single-cell data integration, we are essentially faced with the following challenges: (1) how to construct the graph for cells and its features (or modalities); (2) how to effectively extract meaningful patterns from the graph; and (3) how to adapt the framework to different multimodal tasks.



(a) Graph Construction

(b) Cell-feature Graph Convolution with Task-Specific Head

Figure 2.1 An overview of *scMoGNN*. We first construct the cell-feature graph from a given modality and then perform cell-feature graph convolution to obtain latent embeddings of cells, which are sent to a task-specific head to perform the downstream task.

2.3.1.1 Graph Construction

With the single-cell data, our first step is to construct a cell-feature graph that GNNs can be applied to. We construct a cell-feature bipartite graph where the cells and their biological features (e.g. GEX, ADT or ATAC features) are treated as different nodes, which we term as cell nodes and feature nodes, respectively. A cell node is connected with the feature nodes that represent its features. With such graph structure, the advantage is that cell nodes can propagate features to their neighboring feature nodes, and vice versa.

Formally, we denote the bipartite graph as $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$. In this graph, $\mathcal{U}$ is the set of N cell nodes $\{u_1, u_2, ..., u_N\}$ and $\mathcal{V}$ is the set of k feature nodes $\{v_1, v_2, ..., v_k\}$, where each feature node refers to one feature dimension of the input data. $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{V}$ represents the set of edges between $\mathcal{U}$ and $\mathcal{V}$, which describe the relations between cells and features. The graph can be denoted as a weighted adjacency matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{O} & \mathbf{M} \\ \mathbf{M}^T & \mathbf{O} \end{pmatrix} \in \mathbb{R}^{(N+k)\times(N+k)}, \tag{2.3}$$

where $\mathbf{O}$ is a matrix filled with constant 0 and $\mathbf{M} \in \mathbb{R}^{N\times k}$ is the input feature matrix of cells. $\mathbf{M}$ can be also viewed as the features of one modality such as GEX. Note that $\mathbf{A}$ is a bipartite graph where two nodes within the same set (either feature nodes or cell nodes) are not adjacent. Based on the aforementioned process of graph construction, we can adjust it for specific tasks, e.g., incorporating

a prior knowledge graph of genes, which can change the bipartite characteristic of $\mathbf{A}$.

Furthermore, since GNNs are mostly dealing with attributed graphs, we need to assign initial embeddings for feature and cell nodes. Specifically, we use $\mathbf{X}_{\text{cell}}$ and $\mathbf{X}_{\text{feat}}$ to denote the initial embeddings for cell and feature nodes, respectively. We have $\mathbf{X}_{\text{cell}} \in \mathbb{R}^{N \times d'}$ and $\mathbf{X}_{\text{feat}} \in \mathbb{R}^{k \times d''}$ where d' and d'' are determined by the task-specific settings. As an illustrative example, the initial embeddings of feature nodes $\{v_1, ..., v_k\}$ could be the one-hot index of each feature dimension; thus, $\mathbf{X}_{\text{feat}} \in \mathbb{R}^{k \times k}$ is an identity matrix, i.e., $\mathbf{X}_{\text{feat}} = \mathbf{I}_k$. Meanwhile, we do not have any prior knowledge for each cell, thus, $\mathbf{X}_{\text{cell}} = \mathbf{O}_{N \times 1}$. Together with the node features, the constructed cell-feature graph can be denoted as $\mathcal{G} = (\mathbf{A}, \mathbf{X}_{\text{feat}}, \mathbf{X}_{\text{cell}})$.

2.3.1.2 Cell-Feature Graph Convolution

After we obtain the constructed cell-feature graph, the next challenge is how to extract high-order structural information from the graph. In this work, we utilize GNNs to effectively learn cell/feature representations over the constructed graph. Note that there are two types of nodes in the graph and we need to deal with them differently. For the ease of illustration, we first only consider one type of nodes (e.g., feature nodes $\mathcal{V}$), and later we extend it to the whole graph. Typically, GNNs follow the message-passing paradigm (Gilmer et al., 2017) and in each layer of GNNs, the embedding of each node is updated according to the messages passed from its neighbors. Let $\mathbf{H}^l = \{\mathbf{h}_1^l, ..., \mathbf{h}_N^l\}$, $\mathbf{h}_i^l \in \mathbb{R}_l^d$ be the input node embeddings in the l-th layer, where $\mathbf{h}_i^l$ corresponds to node v_i. Hence, the output embeddings of the l-th layer can be expressed as follows:

$$\mathbf{h}_i^{l+1} = \text{Update}(\mathbf{h}_i^l, \text{Agg}(\mathbf{h}_j^l | j \in \mathcal{N}_i)), \tag{2.4}$$

where $\mathcal{N}_i$ is the set of first-order neighbors of node v_i, $\text{Agg}(\cdot)$ indicates an aggregation function on neighbor nodes' embedding, and $\text{Update}(\cdot)$ is an update function that generates a new node embedding vector from the previous one and aggregation results from neighbors. Notably, for the input node embedding in the first layer, we have

$$\mathbf{H}^1 = \sigma(\mathbf{X}_{\text{feat}}\mathbf{W}_{\text{feat}} + \mathbf{b}_{\text{feat}}), \mathbf{W}_{\text{feat}} \in \mathbb{R}^{d' \times d} \tag{2.5}$$

where $\mathbf{W}_{\text{feat}}$ is a transformation matrix, d is the dimension of the hidden space and σ is an activation function.

Though there exist a number of different GNN models, in this work, we focus on the most representative one, Graph Convolution Network (GCN) (Kipf and Welling, 2017). Note that it is straightforward to extend the proposed framework to other GNN models. Considering that we have two different types of nodes in the graph (i.e., cells and features nodes), we make some modifications on the vanilla GCN to deal with different types of nodes/edges. We name it as *cell-feature graph convolution*, where we separately perform the aggregation function on different types of edges to capture interactions between cell and feature nodes. Moreover, we use different parameters for aggregation on different edges, thus allowing the embedding of each node type to have very different distributions. Specifically, from a message passing perspective, the operation in a cell-feature graph convolution layer can be expressed as two steps, i.e., aggregation and updating. In order to generate a message $\mathbf{m}$ for different types of nodes, there are at least two basic aggregation functions, one is:

$$\mathbf{m}_{\mathcal{U}\to\mathcal{V}}^{i,l} = \sigma(\mathbf{b}_{\mathcal{U}\to\mathcal{V}}^{l} + \sum_{j\in\mathcal{N}_i, v_i\in\mathcal{V}} \frac{e_{ji}}{c_{ji}} \mathbf{h}_j^l \mathbf{W}_{\mathcal{U}\to\mathcal{V}}^{l}) \tag{2.6}$$

where i corresponds to node $v_i \in \mathcal{V}$, j corresponds to node $u_j \in \mathcal{U}$; e_{ji} denotes the edge weight between v_j and u_i, $\mathbf{W}_{\mathcal{U}\to\mathcal{V}}^{l}$ and $\mathbf{b}_{\mathcal{U}\to\mathcal{V}}^{l}$ are trainable parameters, $\sigma(\cdot)$ is an activation function such as ReLU, and c_{ji} is a normalization term defined as follows:

$$c_{ji} = \sqrt{\sum_{k\in\mathcal{N}_j} e_{jk}} \sqrt{\sum_{k\in\mathcal{N}_i} e_{ki}} \tag{2.7}$$

Obviously, Eq. (2.6) is the aggregation function from cell nodes $\mathcal{U}$ to feature nodes $\mathcal{V}$. Thus the other aggregation function from $\mathcal{V}$ to $\mathcal{U}$ can be written as:

$$\mathbf{m}_{\mathcal{V}\to\mathcal{U}}^{i,l} = \sigma(\mathbf{b}_{\mathcal{V}\to\mathcal{U}}^{l} + \sum_{j\in\mathcal{N}_i, u_i\in\mathcal{U}} \frac{e_{ji}}{c_{ji}} \mathbf{h}_j^l \mathbf{W}_{\mathcal{V}\to\mathcal{U}}^{l}) \tag{2.8}$$

The transformation matrices $\mathbf{W}_{\mathcal{U}\to\mathcal{V}}^{l}$ and $\mathbf{W}_{\mathcal{V}\to\mathcal{U}}^{l}$ project the node embeddings from one hidden space to another vice versa. After generating the messages from neighborhoods, we then update the embedding for nodes in $\mathcal{V}$ and $\mathcal{U}$ accordingly:

$$\mathbf{h}_i^{l+1} = \mathbf{h}_i^l + \mathbf{m}_{\mathcal{U}\to\mathcal{V}}^{i,l}, \quad \mathbf{h}_j^{l+1} = \mathbf{h}_j^l + \mathbf{m}_{\mathcal{V}\to\mathcal{U}}^{j,l}, \tag{2.9}$$

where $v_i \in \mathcal{V}$ and $u_j \in \mathcal{U}$. In Eq. (2.9), we adopt a simple residual mechanism in order to enhance self information. As mentioned earlier, we can have more than two types of edges depending on the downstream task and the graph structure can be much more complex than a bipartite graph. Despite such complexity, our proposed framework and cell-feature graph convolution have the capacity to handle more types of edges/nodes. We will introduce these details in Section 2.3.2.

2.3.1.3 Task-specific Head

After we learn node embeddings for feature and cell nodes, we need to project the embedding to the space of the specific downstream task. Hence, we design a task-specific head, which depends on the downstream task. Specifically, we first take the node embeddings of cell nodes from each convolution layer, aggregate them and project them into the space of downstream task $\hat{\mathbf{Y}}$:

$$\hat{\mathbf{Y}} = \text{Head}\left(\text{Readout}_\theta(\mathbf{H}^1_{\mathcal{U}}, ..., \mathbf{H}^L_{\mathcal{U}})\right) \tag{2.10}$$

where $\mathbf{H}^i_{\mathcal{U}}$ refers to embeddings of all cell nodes in i-th layer, $\text{Readout}(\cdot)_\theta$ is a trainable aggregation function, $\text{Head}(\cdot)$ is a linear transformation that projects the latent embedding to the downstream task space. With the obtained output, we can then optimize the framework through minimizing the task-specific loss functions. In the following subsection, we give the details of training the general framework for different tasks.

2.3.2 Model Specifications

With the proposed general framework *scMoGNN*, we are now able to perform different tasks by adjusting some of the components. In this subsection, we show how *scMoGNN* is applied in the three important tasks in single-cell data integration: modality prediction, modality matching and joint embedding.

2.3.2.1 Modality Prediction

In the modality prediction task, our objective is to translate the data from one modality to another. Given the flexibility of graph construction and GNNs in our framework, we can readily incorporate external knowledge into our method. In this task, we adjust the graph construction to include such domain knowledge to enhance the feature information. Specifically, in GEX-to-ADT and

GEX-to-ATAC subtasks, we introduce hallmark gene sets(Liberzon et al., 2015) (i.e., pathway data) from the Molecular Signatures Database (MSigDB)(Subramanian et al., 2005). The pathway data describe the correlations between gene features (i.e., features of GEX data) and the dataset consists of 50 so-called gene sets. In each gene set, a group of correlated genes are collected. However, there is no numerical information in these gene sets to help us quantify the relations between those genes, resulting in homogeneous relations. Intuitively, we can construct a fully-connected inter-gene graph for each gene set, and incorporate those edges into our original graph $\mathcal{G}$. Hence, the new adjacency matrix for $\mathcal{G}$ is:

$$\mathbf{A} = \begin{pmatrix} \mathbf{O} & \mathbf{M} \\ \mathbf{M}^T & \mathbf{P} \end{pmatrix} \tag{2.11}$$

where $\mathbf{P} \in \mathbb{R}^{k \times k}$ is a symmetric matrix, which refers to the links between gene features, generated from gene sets data. Furthermore, we manually add some quantitative information, i.e., cosine similarity between gene features based on their distributions in GEX input data.

Due to the existence of extra type of edges in the graph (i.e., edges among feature nodes), we need to make corresponding adjustment on our cell-feature graph convolution. Taking an arbitrary feature node v_i as example, we have $\mathcal{N}_i = \mathcal{N}_i^{\mathrm{u}} \cup \mathcal{N}_i^{\mathrm{v}}$, where $\mathcal{N}_i$ denotes the set of neighbors of node v_i, $\mathcal{N}_i^{\mathrm{u}} \subseteq \mathcal{U}$ is the set of cell node neighbors of v_i, $\mathcal{N}_i^{\mathrm{v}} \subseteq \mathcal{V}$ is the set of feature node neighbors of v_i. Since cell and feature nodes have very different characteristics, when we aggregate the embedding from $\mathcal{N}_i^{\mathrm{u}}$ and $\mathcal{N}_i^{\mathrm{v}}$ respectively, we expect to get very different results. Thus, when we update the embedding of center node, we have different strategies to combine messages from different channels. As a starting point, we decide to enable a scalar weight. Formally speaking, similar to Eq. (2.6) and Eq. (2.8), we first generate two messages $\mathbf{m}_{\mathcal{U}\to\mathcal{V}}^{i,l}$ and $\mathbf{m}_{\mathcal{V}\to\mathcal{V}}^{i,l}$ for each node $v_i \in \mathcal{V}$, then we update the node embedding of v_i following the formulation below:

$$\mathbf{h}_i^{l+1} = \mathbf{h}_i^l + \alpha \cdot \mathbf{m}_{\mathcal{V}\to\mathcal{V}}^{i,l} + (1-\alpha) \cdot \mathbf{m}_{\mathcal{U}\to\mathcal{V}}^{i,l} \tag{2.12}$$

where α is either a hyper-parameter or a learnable scaler to control the ratio between inter-feature aggregation and cell-feature aggregation.

Next we elaborate on the modality prediction head and loss function for the task. The head structure for modality prediction is relatively simple. Note that we deliberately keep the same hidden dimension throughout the cell-feature graph convolution; thus we can simply use a trainable weight vector $\mathbf{w}$ to sum up cell node embeddings from different layers, as follows:

$$\hat{\mathbf{H}} = \sum_{i=1}^{L} \mathbf{w}_i \cdot \mathbf{H}_{\mathcal{U}}^{i} \tag{2.13}$$

where $\hat{\mathbf{H}}, \mathbf{H}_{\mathcal{U}}^{i} \in \mathbb{R}^{N \times d}$, and d is the dimension of hidden layer. After that, a simple fully connected layer is performed to transform it to the target space:

$$\hat{\mathbf{Y}} = \hat{\mathbf{H}}\mathbf{W} + \mathbf{b}. \tag{2.14}$$

A rooted mean squared error (RMSE) loss is then calculated as:

$$\mathcal{L} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\mathbf{Y}_i - \hat{\mathbf{Y}}_{\mathbf{i}})^2}, \tag{2.15}$$

which is a typical loss function for regression tasks.

2.3.2.2 Modality Matching

Our goal in the modality matching task is to predict the probability that a pair of different modality data is actually from the same cell. Modality matching is very different from the modality prediction task in two regards: (1) it requires interactions between two modalities; and (2) it does not demand the model to give detailed predictions but it emphasizes pairings between two modalities. Therefore, we need to adjust the framework in graph construction and task-specific head correspondingly.

Since two different modalities are presented as input in this task, we construct two cell-feature graphs. Cell-feature graph convolutions are then performed on these two graphs separately to obtain two embedding matrices $\hat{\mathbf{H}}_{\mathrm{m1}} \in \mathbb{R}^{N \times d_1}$ and $\hat{\mathbf{H}}_{\mathrm{m2}} \in \mathbb{R}^{N \times d_2}$. We set $d_1 = d_2$ so they can be directly multiplied together. Thus, we calculate the cosine similarity between each cell pair as follows:

$$\mathbf{S} = \hat{\mathbf{H}'}_{\mathrm{m1}} \cdot \hat{\mathbf{H}'}_{\mathrm{m2}}^{T} \tag{2.16}$$

where $\mathbf{S} \in \mathbb{R}^{N\times N}$ denotes the symmetric score matrix; $\hat{\mathbf{H}}'_{\text{m1}}$ and $\hat{\mathbf{H}}'_{\text{m2}}$ indicate that we perform L2 normalization for each row (i.e., each cell) in $\hat{\mathbf{H}}_{\text{m1}}$ and $\hat{\mathbf{H}}_{\text{m2}}$ before we perform matrix multiplication. We further calculate the softmax function for each row and each column of $\mathbf{S}$ to convert scores to probabilities. Then we can express the loss function as follows:

$$\begin{aligned} \mathcal{L}_{\text{match}} &= -\sum_{c_1=1}^{N}\sum_{c_2=1}^{N} \mathbf{Y}_{c_1,c_2}\log(\mathbf{P}^r_{c_1,c_2}) + \mathbf{Y}_{c_1,c_2}\log(\mathbf{P}^c_{c_1,c_2}) \\ \text{with}\quad \mathbf{P}^r_{i,j} &= \frac{e^{\mathbf{S}_{i,j}}}{\sum_{k=1}^{N} e^{\mathbf{S}_{i,k}}}, \quad \mathbf{P}^c_{i,j} = \frac{e^{\mathbf{S}_{i,j}}}{\sum_{k=1}^{N} e^{\mathbf{S}_{k,j}}}, \end{aligned} \tag{2.17}$$

where $\mathbf{Y} \in \mathbb{R}^{N\times N}$ denotes a binarized matching matrix that indicates the perfect matching (i.e., the ground truth label), and $\mathbf{P}_{i,j} \in \mathbb{R}^{N\times N}$ represents the probability that i-th data in modality 1 and j-th data in modality 2 are actually referring to the same cell.

In addition to the matching loss $\mathcal{L}_{\text{match}}$, we include a set of auxiliary losses to boost the performance, i.e., prediction losses and reconstruction losses:

$$\begin{aligned} \mathcal{L}_{\text{aux}} &= \mathcal{L}_{\text{pred12}} + \mathcal{L}_{\text{pred21}} + \mathcal{L}_{\text{recon11}} + \mathcal{L}_{\text{recon22}} \\ &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{\text{m2}} - f_{\theta_2}(\hat{\mathbf{H}}_{\text{m1}}))^2 + \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{\text{m1}} - f_{\theta_1}(\hat{\mathbf{H}}_{\text{m2}}))^2 \\ &+ \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{\text{m1}} - f_{\theta_1}(\hat{\mathbf{H}}_{\text{m1}}))^2 + \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{\text{m2}} - f_{\theta_2}(\hat{\mathbf{H}}_{\text{m2}}))^2 \end{aligned} \tag{2.18}$$

where $\mathbf{X}_{\text{m1}}$ and $\mathbf{X}_{\text{m2}}$ refer to the preprocessing results of two modalities respectively, f_{θ_1} and f_{θ_2} each refers to a fully connected network with one hidden layer, which project the node embeddings $\hat{\mathbf{H}}$ to the particular target modality space. These auxiliary losses provide extra supervision for our model to encode the necessary information within the hidden space $\hat{\mathbf{H}}$. Hence they have the potential to improve the robustness of the model and reduce the risk of overfitting. In the training phase, we jointly optimize $\mathcal{L}_{\text{match}}$ and $\mathcal{L}_{\text{aux}}$.

Lastly, in the inference phase, we introduce bipartite matching as an extra post-processing method to further augment our matching result. Specifically, we first use percentile threshold to filter the score matrix $\mathbf{S}$ in order to reduce the subsequent calculations, resulting in a symmetric sparse matrix $\mathbf{S}'$. We consider $\mathbf{S}'$ as an adjacency matrix of a bipartite graph, which helps us model

the two different modality data and their inter-class relations. Thus, our goal (i.e., matching between two modalities) becomes a rectangular linear assignment problem, where we try to find a perfect matching that maximizes the sum of the weights of the edges included in the matching. We can effectively solve this problem through the Hungarian algorithm, also known as the Munkres or Kuhn-Munkres algorithm.

2.3.2.3 Joint Embedding

The target of the joint embedding task is to learn cell embeddings from multiple modalities and thus better describe cellular heterogeneity. Several complex metrics are enabled in this task, there often exist trade-offs between metrics and metrics (e.g. to remove the batch effect while retaining the batch information). To provide more training signals, we utilize both supervised and self-supervised losses to train our graph neural networks. Specifically, we first use the LSI for preprocessing to generate the input node features for two modalities and concatenate them as one joint modality, which allows us to construct a cell-feature graph. Based on the graph, we perform the proposed cell-feature graph convolution and generate the output cell node embedding in the same way as in Eq. (2.13). As suggested by the work (Liu et al., 2021a), cell type information plays a key role in the metrics of joint embedding evaluation and it is beneficial to extract T dimensions from the hidden space to serve as supervision signals. Following this idea, we calculate the softmax function for $t \in \{1, ..., T\}$, $i \in \{1, ..., N\}$ with N being the number of cells:

$$\hat{\mathbf{Y}}_{i,t} = \frac{e^{\hat{\mathbf{H}}_{i,t}}}{\sum_{k=1}^{T} e^{\hat{\mathbf{H}}_{i,k}}} \tag{2.19}$$

where $\hat{\mathbf{Y}}$ is the probability that cell i belongs to cell type t and T is set to be exactly equal to the total number of cell types. Then we introduce the loss functions:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{cell type}} + \mathcal{L}_{\text{regular}} \\ &= \frac{1}{N} \sum_{i=1}^{N} (\mathbf{X}_{\text{LSI}} - f_\theta(\hat{\mathbf{H}}))^2 + \sum_{t=1}^{T} \mathbf{Y}_t \log(\hat{\mathbf{Y}}_t) + \beta * \|\hat{\mathbf{H}}_{\tilde{\mathcal{J}}}\|_2 \end{aligned} \tag{2.20}$$

where f_θ is a two-layer perceptron, $\mathbf{Y} \in \mathbb{R}^{N \times T}$ is the sparse labels of cell types, and $\hat{\mathbf{H}}_{\tilde{\mathcal{J}}}$ refers to the other hidden dimensions aside from the T dimensions that have been exclusive to cell type

information. Eventually, the output hidden space $\hat{\mathbf{H}}$ would contain cellular information required by the task, although the loss functions do not directly optimize the final metric.

2.4 Experiment

In this section, we evaluate the effectiveness of our framework *scMoGNN* against three tasks and show how *scMoGNN* outperforms representative baselines over all three tasks by combining our general framework with task-specific design. Note that in this experiment, we follow the official settings and datasets in the multimodal single-cell data integration competition at NeurIPS 2021 (Luecken et al., 2021) and we compare the performance of the proposed framework with that from the top winners in the competition. All source codes of our model have been integrated into the DANCE package (Ding et al., 2024).

2.4.1 Modality Prediction

Datasets. In the modality prediction dataset, each modality is provided with source data, preprocessed data (with default methods), and batch labels. Data statistics are shown in Table A.1 in Appendix A.2. Note that the GEX-ATAC and ATAC-GEX subtasks are not entirely symmetric, because in the GEX-ATAC task, the output dimension is deliberately reduced, where 10,000 ATAC features are randomly selected out of the original 116,490 dimensions.

Settings and Parameters. For our own model, we report the average performance of 10 runs. In each run, we reserved 15% of training cells for validation and early stopping. In practice, several tricks help boost the performance: (1) utilizing initial residual instead of the skip connections in Eq.2.9, (2) using group normalization for aggregation results and dismissing the edge weight normalization stated in Eq. 2.7. Besides, we empirically set our parameter α in Eq. 2.12 to 0.5. Additionally, in order to fit the high-dimensional ATAC features, we enabled node sampling.

Baselines. In Table 2.1, we only show the teams that acquired top results in one or more subtasks in the competition, because they are officially required to declare their model details. For further comparison, we briefly detail each method: (1) Baseline, a truncated-SVD dimensionality reduction followed by linear regression. (2) Dengkw, a well-designed model based on kernel ridge regression. (3) Novel, an encoder-decoder structure with LSI preprocessing. (4) Living System

Lab, an ensemble model composed of random forest models, catboost(Prokhorenkova et al., 2018) models, and k-nearest neighbors regression models. (5) Cajal, a feed forward neural network with heavy feature selection guided by prior knowledge. (6) ScJoint, an ensemble neural network model incorporated various strategies of preprocessing such as extracting TF-IDF features and filtering highly variable genes/features.

The source codes for all the methods above can be found in the official github of the competition[5]. It can be seen that the existing models are relatively simple, mainly based on traditional machine learning algorithms and autoencoders. In contrast, our framework has a more advanced architecture, which provides the flexibility to different structural data (e.g. graph data) and different tasks. This makes our framework a very suitable backbone in the field of single-cell multimodal data integration.

Results. As shown in Table 2.1, our method achieved the lowest overall loss in the competition (the lower, the better). An interesting observation is that there is no team lead consistently across all subtasks, resulting in individual s for each category, which is very different from the other two tasks in the competition. However, as far as we know, there is no team that worked only on one subtask. Such a phenomenon may be caused by three reasons: (1) the modality prediction task is the most competitive task in the competition, and many participating teams participated only in this task (including our own team). As a result, over 40 individual teams appeared on the final leaderboard. (2) the modality prediction task has only 1,000 cells in the private test dataset, therefore, certain variance exists in the evaluation results. (3) the diverse feature dimensionality and sparsity in different modalities raised additional challenges to the model's generalization ability. Compared to the other models, our GNN model presented consistently better performance across these four subtasks and became the overall winner in the competition.

Furthermore, we even improved our models after the competition, with modifications including: a learning-rate decay training strategy, more hidden units along with stronger regularization of dropout and weight decay. Eventually, we've effectively strengthened our graph neural network model hence significantly improved our results, especially in the toughest subtask GEX-to-ADT,

[5]The official github link is https://github.com/openproblems-bio/neurips2021_multimodal_topmethods.

Table 2.1 RMSE for Modality Prediction (Task 1)↓ . '*' indicates ensemble models.

	GEXADT	ADT2GEX	GEX2ATAC	ATAC2GEX	Overall
Baseline	0.4395	0.3344	0.1785	0.2524	0.3012
Dengkw*	0.3854	0.3242	0.1833	0.2449	0.2836
Novel	0.4153	**0.3177**	0.1781	0.2531	0.2911
LS. Lab*	0.4065	0.3228	**0.1774**	0.2393	0.2865
Cajal	0.4393	0.3311	0.1777	**0.2169**	0.2891
ScJoint*	0.3954	0.3247	0.1785	0.2377	0.2840
*scMoGNN**	0.3898	0.3221	0.1776	0.2403	0.2824
scMoGNN (Single)	0.3885	0.3242	0.1778	0.2315	0.2805
*scMoGNN** (Ensemble)	**0.3809**	0.3223	0.1777	0.2310	**0.2780**

where the output for each cell is a 134-dimensional dense vector. We now achieved an RMSE loss of 0.3809 which is lower than the previous best score of 0.3854 in the competition. Overall, the results prove the effectiveness of our GNN framework, and in some specific cases, *scMoGNN* has tremendous advantage in view of performance.

2.4.2 Modality Matching

Datasets. The majority of the modality matching dataset is the same as the modality prediction dataset (as shown in Table A.2 in Appendix A.2, while several differences exist, including: (1) the number of testing cells; (2) the dimensionality of ATAC features; and (3) the inconsistent cell order among modalities. In the training data, samples' correspondence between different modalities is given, while for the test data, our goal is to find the correspondence.

Settings and Parameters. The experimental settings are similar to the previous task, while some adjustments were made. For calculation convenience, we decoupled the propagation and transformation. Besides, batch labels are given in company with the test data, which provides a very strong prior knowledge for matching. We thus divided the test data into different batches, then matched samples that belong to the same batch. This trick dramatically reduced the search space, resulting in a significant performance boost. To be fair, we have confirmed that the winning solution also used the same strategy.

Baselines. In Table 2.2, we only compare our models with the winning team and runner-up

Table 2.2 Performances for Modality Matching (Task 2)↑.

	GEX2ADT	ADT2GEX	GEX2ATAC	ATAC2GEX	Overall
Baseline	0.0000	0.0000	0.0001	0.0001	0.0001
GLUE (CLUE)	0.0495	0.0516	0.0560	0.0583	0.0539
Novel	0.0373	0.0373	0.0412	0.0412	0.0392
scMoGNN	**0.0810**	**0.0810**	**0.0630**	**0.0630**	**0.0720**

team in the competition. Next we briefly introduce those models: (1) Baseline, first projecting one modality to the other with linear regression, and then searching for the nearest neighbors in the same modality. (2) GLUE (Cao and Gao, 2022) (CLUE), the official winner, a variational auto-encoder (VAE) model supervised by three auxiliary losses. (3) Novel, a feed-forward neural network directly supervised by matching loss.

Results. As shown in Table 2.2, *scMoGNN* outperforms the winning team and the runner-up team with a very large margin. Note that we didn't create any models for this task during the competition since we focused on the modality prediction task. This is the reason why we don't have any results on the official leaderboard.

The score of the metric can be roughly seen as the accuracy of predicting a right correspondence for each piece of data. Meanwhile the search space grows with the total number of cells in the test data. For example, in the test phase of the ADT-to-GEX subtask, we have 15,066 cells to match, thus for each piece of ADT data, we have 15,066 candidates in GEX data. Although we separated those cells into three batches, the rough expectation of the accuracy of randomly guessing is still as low as 1/5000, which can indicate the difficulty of this task. Thus, *scMoGNN* has already achieved very high performance (e.g. 0.08 in ADT-GEX subtask). Note that both team Novel's model and *scMoGNN* utilizes a symmetric matching algorithm, thus we have exactly the same performance for dual subtasks (e.g. GEX2ADT and ADT2GEX). Another interesting observation is that our proposed graph neural network model is especially good at GEX-ADT dual subtasks, where we improved the previous winning performance from 0.05 to 0.08.

Table 2.3 Performances for GEX-ADT Joint Embedding (Task 3)↑.

	NMI	Cell type ASW	Cc_con	Traj_con	Batch ASW	Graph Conn	Average
Baseline	0.6408	0.5266	**0.9270**	0.8325	0.7982	0.8945	0.7699
Amateur (JAE)	0.7608	0.6043	0.7817	**0.8631**	0.8432	0.9700	0.8039
GLUE	0.8022	0.5759	0.6058	0.8591	**0.8800**	0.9506	0.7789
scMoGNN	**0.8499**	**0.6496**	0.7084	0.8532	0.8691	**0.9708**	**0.8168**

2.4.3 Joint Embedding

Datasets. The training data of this task are basically the same as the modality prediction task. Moreover, data from different modalities are already aligned. Regarding the complementary data, there are two settings for this task. In the 'online' setting, the only data is features of two modalities. Meanwhile, in the 'pre-trained' setting, any external knowledge is acceptable. In this paper, we follow the second setting (i.e. pre-trained setting) and we only compare our results with other pre-trained models. Generally speaking, pre-trained models obtain better performance than online models. In our experiments, cell type labels are provided in the training phase, while test data consist of all the train cells and unseen test cells but are not along with cell type labels.

Settings and Parameters. Considering that the input in this task is similar to modality matching, we followed settings in Section 2.4.2.

Baselines. We briefly describe the other three models in Table 2.3: (1) Baseline, a concatenation of PCA results of two modalities. (2) Amateur (JAE), the official winner, an auto-encoder model that incorporates extra supervision (cell annotations). The model is adapted from scDEC (Liu et al., 2021a). (3) GLUE (Cao and Gao, 2022), an auto-encoder model guided by an external knowledge graph.

Results. As shown in Table 2.3, our *scMoGNN* significantly outperforms the other two models in GEX-ADT joint embedding task, with an improvement over 0.1 according to the average metric.

2.4.4 Ablation Study

Throughout the previous sections, we have examined that our graph neural network general framework is suitable for all these tasks in single-cell multimodal data integration. In this subsection, we investigate if we really benefit from graph structural information. We take the modality matching

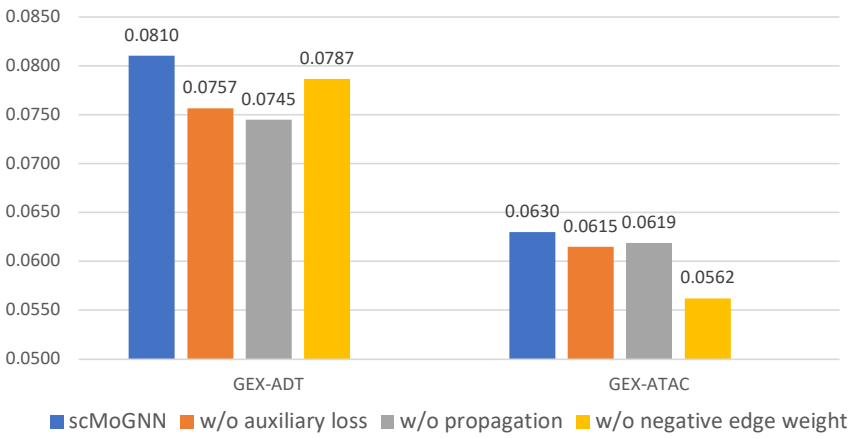


Figure 2.2 Ablation study for the modality matching task.

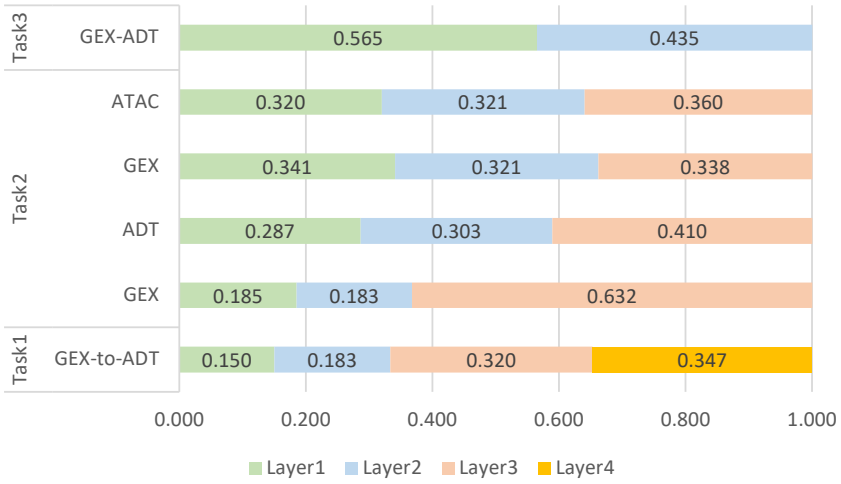


Figure 2.3 Parameter analysis of layer weight $\mathbf{w}$.

task as an example. In the modality matching task we use decoupled GNNs, thus, we can easily remove the graph structural information by eliminating the propagation layers. The result is referred to as "w/o propagation" in Figure 2.2. The performance significantly drops from 0.0810 to 0.0745 in the GEX-ADT subtask and from 0.0630 to 0.0562 in the GEX-ATAC subtask, respectively. These observations indicate that the graph structural information extracted by the propagation layers indeed helped the performance of our method significantly. We also examined the importance of our auxiliary loss, shown in Figure 2.2. Without the supervision of auxiliary losses, *scMoGNN* lost a lot of generalization ability, behaving as poorly as without graph structural information.

2.4.5 Parameter Analysis

We analyzed an important parameter in our framework, i.e. $\mathbf{w}$ in Eq. 2.13, in order to gain a deeper insight on *scMoGNN*. Specifically, $\mathbf{w}$ is a learnable parameter that controls the weight between each propagation layer. Intuitively, the value of $\mathbf{w}$ can prove to us the effectiveness of graph structural information and help us understand how much higher-order structural information is valued by models in different tasks. Therefore we show values of $\mathbf{w}$ learned by *scMoGNN* in different tasks in Figure 2.3. Note that in different tasks we have different numbers of layers, in modality prediction we have 4 layers and in modality matching and joint embedding we have 3 layers and 2 layers, respectively. The results consistently show that *scMoGNN* tends to synthesize the information in each layer, not just limited to the shallow layer, which suggests that the information of the higher-order graph structure is indeed effective. As for more details, joint embedding depends more on local information, which exists in source input data. While in modality prediction, more higher-order information is referenced, indicating that the model needs to enrich more information from similar cells or similar features. This can be explained from the need for more detailed information in modality prediction.

2.5 Related Work

In this section, we briefly introduce works related to our work including GNNs on single-modality data and multimodal data integration.

GNNs on Single-Modality Data. Graphs occur as a natural representation of single-cell data both as feature-centric (RNAs, DNAs, or proteins) and cell-centric. Thus, a few recent works have applied GNNs to the single-cell data. scGCN (Song et al., 2021) proposes a GNN model for knowledge transfer in single-cell omics (mRNA or DNA) based on Graph Convolutional Networks (Kipf and Welling, 2017). scGNN (Wang et al., 2021) formulates and aggregates cell-cell relationships with Graph Neural Networks for missing-data imputation and cell clustering using single-cell RNA sequencing (scRNA-seq) data. scDeepSort (Shao et al., 2021) is a pre-trained cell-type annotation tool for scRNA-seq data that utilizes a deep learning model with a weighted GNN. Similar to our proposed model, scDeepSort also relies on feature-cell graphs. However, it

does not incorporate any prior knowledge into GNNs. Using spatial transcriptomics (mRNA) data, DSTG (Song and Su, 2021) utilizes semi-supervised GCN to deconvolute the relative abundance of different cell types at each spatial spot. Despite its success on single-modality data, there are few efforts on applying GNNs to multimodal single-cell data.

Multimodal Data Integration. Most of the prior works in multimodal data integration can be divided into 1) matrix factorization (Duren et al., 2018; Stein-O'Brien et al., 2018; Jin et al., 2020) or statistical based methods (Stuart et al., 2019; Shen et al., 2009; Welch et al., 2017) and 2) autoencoder based methods (Wu et al., 2021b; Gong et al., 2021). Specifically, BABEl (Wu et al., 2021b) leverages autoencoder frameworks with two encoders and two decoders to take only one of these modalities and infer the other by constructing reconstruction loss and cross-modality loss. Cobolt(Gong et al., 2021) acquires joint embedding via a variant of Multimodal Variational Autoencoder (MVAE(Yao et al., 2021)). Unlike our proposed models, these aforementioned methods are unable to incorporate high-order interactions among cells or different modalities. To the best of our knowledge, we are the first to apply GNNs in the field of multimodal single-cell data integration and build a GNNs-based general framework to broadly work on these three key tasks from NeurIPS 2021 Competition. Our framework officially won first place in the overall ranking of the modality prediction task. After the competition, we extended our framework to the other two tasks and achieved superior performance compared with the top winning methods.

2.6 Chapter Conclusion

In this chapter, we proposed a general framework *scMoGNN* based on GNNs for multimodal single-cell data integration and multi-omics representation learning. It can be broadly applied to all three key tasks, modality prediction, modality matching and joint embedding, from the NeurIPS 2021 Competition. Our framework *scMoGNN* is able to capture high-order structural information between cells and features. To the best of our knowledge, we are the first to apply GNNs in this field. Our method officially won first place in the overall ranking of the modality prediction task and now outperforms all models from three tasks on the leaderboard with remarkable advantage.

CHAPTER 3

TRANSFORMERS FOR SINGLE-CELL MULTI-OMICS REPRESENTATION LEARNING

The recent development of multimodal single-cell technology has made the possibility of acquiring multiple omics data from individual cells, thereby enabling a deeper understanding of cellular states and dynamics. Nevertheless, the proliferation of multimodal single-cell data also introduces tremendous challenges in modeling the complex interactions among different modalities. The recently advanced methods focus on constructing static interaction graphs and applying graph neural networks (GNNs) to learn from multimodal data, such as *scMoGNN* in Chapter 2. However, such static graphs can be suboptimal as they do not take advantage of the downstream task information; meanwhile, GNNs also have some inherent limitations when deeply stacking GNN layers. To tackle these issues, in this work, we investigate how to leverage transformers for multimodal single-cell data in an end-to-end manner while exploiting downstream task information. In particular, we propose a *scMoFormer* framework that can readily incorporate external domain knowledge and model the interactions within each modality and cross-modalities.

3.1 Chapter Introduction

Advancements in multimodal single-cell technologies provide the capability to simultaneously profile multiple data types in the same cell, including chromatin accessibility (Cao et al., 2018; Chen et al., 2019), DNA methylation (Gaiti et al., 2019), nucleosome occupancy (Pott, 2017). For example, CITE-seq (Stoeckius et al., 2017) utilizing oligonucleotide-conjugated antibodies can quantify RNA and surface protein abundance in the same cells. Here, protein abundance is measured via the antibody-derived tags (ADTs) read counts. The Single Cell Multiome ATAC + Gene Expression technology (Belhocine et al., 2021) concurrently profiles assay of transposase-accessible chromatin (ATAC-seq) (Buenrostro et al., 2013) and RNA expression from the same cell. These technologies offer an exciting opportunity to characterize cell identity and state at an unprecedented resolution, enabling a better understanding of gene regulatory networks in multicellular organisms and tissues (Zhu et al., 2020).

Despite the rapid accumulation of multimodal single-cell data, the analyses of such data are still faced with numerous challenges. First, single-cell measurements frequently exhibit high sparsity levels, making it difficult to draw meaningful insights from the data. The measurement process is impacted by various environmental factors, such as amplification bias, cell cycle effects, variations in library size, and RNA capture rate, all of which contribute to substantial noise in single-cell data (Eraslan et al., 2019). Furthermore, samples are often measured under different conditions, including batches, times, locations, or using different instruments, which leads to systematic variations in the measured values, which further complicates the interpretation of single-cell data. These imperfections can result in biased estimates of cell-cell interactions and pose tremendous challenges for computational models to exploit such interactions.

To effectively capture the intricate interactions within cells and genes, current research focuses on constructing static graphs based on heuristic criteria and then employing graph neural networks (GNNs) (Kipf and Welling, 2017; Velickovic et al., 2018; Ma and Tang, 2021) to extract information from the built graph. For example, Hao et al. (Hao et al., 2021) and Van et al. (Van Dijk et al., 2018) have built the k-nearest neighbor (k-NN) graph for cells, which assesses the connections between cells by measuring the similarity in gene expression of cells. However, the quality of k-NN graphs is contingent upon the selected heuristic similarity measure and it does not incorporate downstream task information, which can introduce noise for GNNs to perform well on the downstream task. An alternative way for building the interaction graph is to construct the graph based on prior domain knowledge, e.g., utilizing publicly accessible databases (Wen et al., 2022a). For instance, Wen et al. (Wen et al., 2022a) construct the graph based on the pathway data (Liberzon et al., 2015) extracted from the Molecular Signatures Database (Subramanian et al., 2005), which describes the correlations between gene features. However, similar to k-NN graphs, such graph construction process does not leverage downstream task information and the knowledge base may not include all relevant genes/proteins. To make things worse, GNNs have some inherent limitations that hinder their success in applications: the over-smoothing (Kreuzer et al., 2021) and over-squashing (Alon and Yahav, 2021) issues where GNNs produce poor results when we deeply stack GNN layers. In

view of these, one question naturally arises: *can we have a better approach to construct interaction graphs (among cells, genes, and proteins) that utilize downstream information while avoiding the aforementioned issues?*

In light of the recent advances of transformers (Devlin et al., 2019; Kitaev et al., 2020; Liu et al., 2021c, 2022) in capturing pairwise relations among objects, we seek to utilize transformers for learning the interaction graph for cells, genes, and proteins in an end-to-end manner. Transformers are well-suited to address the limitations of static graphs: they learn the interaction between objects through the self-attention mechanism, where all objects are attended to each other with learnable attention scores indicating their interaction strength. Thus, the attention matrix provides an advanced approach to characterize the interaction between objects in a data-driven way and has demonstrated success in reducing unwanted variance and noise across batches (Yang et al., 2022). Moreover, multi-head and multi-layer transformers have the capability of capturing more complex and nuanced relationships during the training process (Huang et al., 2021; Ieremie et al., 2022). However, these traditional transformers do not account for the available graph structure and are therefore unable to leverage prior information present in graph data, such as biological knowledge graphs. In this context, graph transformers (Rampasek et al., 2022; Ying et al., 2021; Dwivedi and Bresson, 2020) offer a solution by combining the strengths of GNNs and transformers to make use of graph data. These approaches allow for the incorporation of prior insights from structural information learned from GNNs, while still allowing for data-specific interactions to be learned through the attention mechanism. On top of that, graph transformers also alleviate the over-smoothing and over-squashing problems in GNNs by enabling individual objects to attend to unconnected objects (Chen et al., 2022; Dwivedi et al., 2021; Kreuzer et al., 2021; Rampášek et al., 2022). Therefore, it is of great importance to investigate the potential of (graph) transformers in single-cell analysis.

In this work, we aim to design a transformer framework for multimodal single-cell data. In essence, to utilize the strengths of (graph) transformers, we are faced with two non-trivial challenges. **First**, since multimodal data contains diverse information from various sources, e.g., genes, proteins, and cells, it can be difficult for a single transformer to capture all aspects. **Second**, traditional

transformers suffer a quadratic computation complexity w.r.t. the number of objects, which poses a challenge for single-cell analysis where the number of cells can be large. To address the first challenge, we introduce the Single-Cell Multimodal Transformer *scMoFormer*, which employs multiple transformers to model the multimodal data, allowing each transformer to deal with a specific data modality. The core of *scMoFormer* is the cross-modality aggregation component which builds a bridge between these transformers and aggregates the necessary information from individual ones. For the second challenge, *scMoFormer* employs linearized transformers (Choromanski et al., 2021) to the cells which greatly reduces the computational complexity.

To the best of our knowledge, we are the first to employ transformers to advance the analysis of multimodal single-cell data. Our proposed framework achieves promising results on the benchmark datasets, providing a very strong baseline for follow-up research. Our contributions can be summarized as follows:

- We study the problem of multimodal single-cell data analysis and propose a transformer framework *scMoFormer* to capture the intricate relations within modalities and between modalities.
- The proposed *scMoFormer* is versatile and can flexibly incorporate domain knowledge of biological databases regarding genes and proteins.
- The proposed *scMoFormer* achieves superior performance on various benchmark datasets. Remarkably, we won a **Kaggle silver medal** with the rank of $24/1221$ (**Top 2%**) without multi-model ensembling in a NeurIPS 2022 competition[1].

3.2 Problem Statement

Before we state the problem, we first introduce the notations used in the following sections. For clarity and simplicity, we use the subscripts "g", "p" and "c" for gene, protein, and cell, respectively. For instance, we use $\mathbf{h}_g$, $\mathbf{h}_p$, $\mathbf{h}_c$ to denote the embeddings of genes, proteins, and cells, respectively.

Although we aim to develop a general framework for multi-omics, a great entry point is to focus on a specific standardized problem that is easy to evaluate. In this study, we follow the benchmark

[1]The competition official website is https://nips.cc/virtual/2022/competition/50092.

setting in the NeurIPS 2022 competition, i.e., Multimodal Single-Cell Integration Across Time, Individuals, and Batches[2]. The specific problem in this competition is to predict a paired modality with a given modality and to infer how DNA, RNA, and protein measurements co-vary in single cells. For simplicity, in this work, we focus on using gene expression (RNA) to predict surface protein levels. We denote $\mathbf{X}_\mathrm{g}$ and $\mathbf{X}_\mathrm{p}$ as the measurement counts of gene and protein, respectively. With $\mathbf{X}_\mathrm{g}$, we try to learn a mapping function that can best describe the relationship between two modalities. We denote $\mathcal{L}(\cdot,\cdot)$ as the objective function that measures the dissimilarity between the predicted and the true protein level. Formally, we describe our target as an optimization problem:

Given $\mathbf{X}_g$ and the objective function $\mathcal{L}(\cdot,\cdot)$, we aim to find a mapping function f_θ^* (parameterized by θ) that minimize the objective loss:

$$f_\theta^* = \arg\min_{f_\theta} \mathcal{L}\left(f_\theta\left(\mathbf{X}_g\right), \mathbf{X}_p\right). \tag{3.1}$$

In the subsequent sections, we formulate the mapping f_θ^* using transformers and GNNs and employ Root Mean Square Error, Mean Absolute Error, and Pearson correlation coefficient as evaluation metrics for our predictions.

3.3 Proposed Multimodal Transformer Framework

In this section, we introduce the proposed multimodal framework *scMoFormer*. An overview of *scMoFormer* is shown in Figure 3.1. It consists of multimodal graph construction, a multimodal transformer, and a prediction layer. In brief, we first construct a heterogeneous graph that contains cell, gene, and protein nodes together with their interactions. We then utilize multiple (graph) transformers on top of this heterogeneous graph to extract rich cell representations and predict each cell's surface protein abundance levels. In the following subsections, we will detail these key components.

3.3.1 Multimodal Graph Construction

In this subsection, we introduce how we construct the graph for multimodal single-cell data. Specifically, we construct a heterogeneous graph containing three different types of nodes to denote

[2]The competition information can be found at https://www.kaggle.com/competitions/open-problems-multimodal.

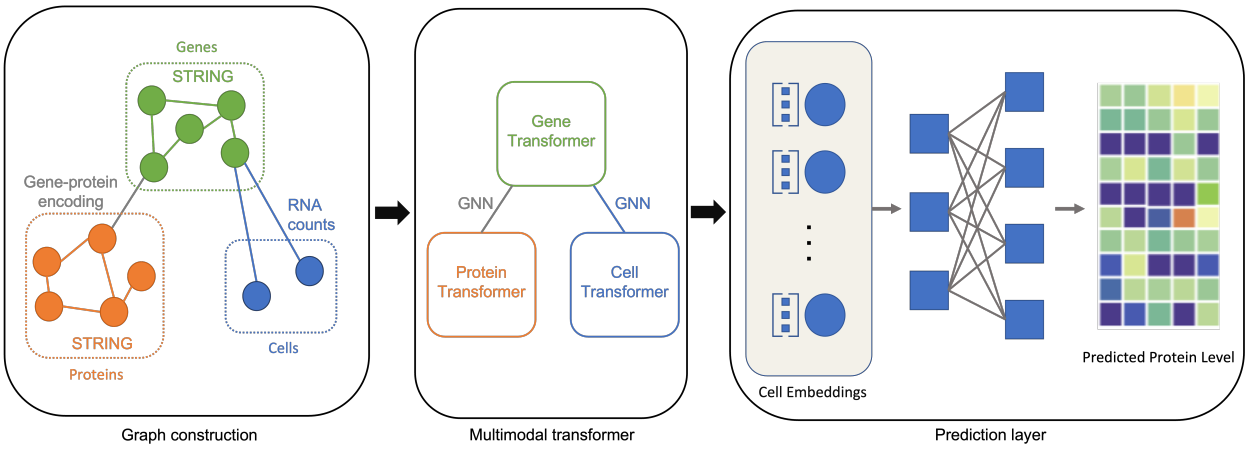


Figure 3.1 An illustration of *scMoFormer*. In this framework, three important components are included: graph construction, multimodal transformer, and prediction layer.

the entire data. It has four subgraphs as shown in Figure 3.1, i.e., a protein-protein graph, a gene-gene graph, a gene-protein graph, and a cell-gene graph. Next, we detail how to conduct these subgraphs.

3.3.1.1 Subgraph Construction.

Before we present the construction of the multimodal heterogeneous graph, we first describe how we build the subgraphs within modalities and between modalities.

Protein-protein graph. To integrate prior biological information into protein-protein graph, we refer to the STRING (Szklarczyk et al., 2023) database. STRING provides a comprehensive resource of protein-protein interactions and the functional relationships between different proteins. It contains seven different channels covering varied aspects of sources, including genomic context, experiment results, and text mining efforts. We use the combined confidence score of all channels as edge weights to comprehensively enhance the graph. The proteins in STRING are labeled in Ensemble (Cunningham et al., 2022) protein (ENSP), and the mapping from ENSP to the protein preferred name is provided in additional information resource. Notice that regularly one protein may have multiple aliases. To match the protein display names with ENSP, we utilize GeneCards (Stelzer et al., 2016) to find all possible aliases of our target proteins. In a nutshell, the mapping from STRING nodes to our target proteins is given by ENSP $\rightarrow$ possible aliases of proteins $\rightarrow$ the display names of proteins within the dataset.

Gene-gene graph. In order to maintain consistency and reduce the impact of potential noise

from multiple sources of prior information, we also utilized the STRING database to construct the gene-gene graph and combined all seven channels of the STRING database to enhance the graph as much as possible. It is worth noting that although the gene and protein nodes are biologically equivalent in the sense of prior information, we process them separately to more specifically handle data-specific information related to the target labels. The genes are labeled in Ensemble gene (ENSG) and additional matching efforts are needed to form the gene-gene graph. We utilize the MyGeneInfo (Wu et al., 2014) gene query service to align the STRING protein nodes, encoded by ENSP, with the input gene nodes, encoded by ENSG.

Gene-protein graph. Now we have two separate graphs among proteins and genes respectively, and we would like to form a general frame by adding the connections between genes and proteins. Following the central dogma of molecular biology, information flows from RNA to proteins via translation. This encoding relationship is recorded within the gene and protein nomenclature. Specifically, the gene names from existing single-cell multimodal datasets contain two parts: the ENSG and the symbol or name of corresponding proteins. That is, if a gene and a protein share the same symbol, it means the target protein is encoded by the gene. Utilizing biological information, we link the proteins to genes by matching their symbols.

Cell-gene graph. The constructed gene-protein graph is fully based on prior knowledge, and we integrate data-based information into the multimodal heterogeneous graph by involving cell nodes. The gene expression counts of multimodality data imply the data-specific relationships between genes and cells. Naturally, a cell and a gene are connected if the gene expresses within that cell. Note that the number of genes detected within each cell varies significantly, which depicts that the raw counts of RNA abundance show substantial heterogeneity among cells. In addition, the raw data includes extremely large counts, making it impractical to straightforwardly apply counts data as the edge weights between cells and genes. Therefore, we normalize each cell by total counts over all genes and taking the logarithm of the resulting data matrix, i.e., library size normalization and centered log-transformation. The processed data is then fed into the multimodal graph to describe the cell-gene links.

Remark. In the above discussion, we have introduced four types of subgraphs, i.e., a protein-protein graph, a gene-gene graph, a gene-protein graph, and a cell-gene graph. It is worth noting that we do not construct the cell-cell graph. Instead, we use the transformer to learn the cell-cell relationships via the attention mechanism. This is in contrast with some previous studies that utilize a static cell-cell similarity graph generated from the input features, which might be prone to inaccurate cell relationships due to the noisy nature of single-cell data. On the other hand, we do not explicitly establish connections between the cells and the target protein nodes, as it might cause the model to easily overfit.

3.3.1.2 Heterogeneous Graph Construction.

With the aforementioned subgraphs, we are now ready to explain how we construct the multimodal heterogeneous graph.

Formal Graph Definition. We now formally define the multimodal heterogeneous graph. Let $\mathbf{A}$ be the adjacency matrix of the multimodal heterogeneous graph with $\mathcal{V} = \left(\mathbf{v}_{\mathrm{p}}, \mathbf{v}_{\mathrm{g}}, \mathbf{v}_{\mathrm{c}}\right)$ as the node set, where we have

$$\begin{aligned} \mathbf{v}_{\mathrm{p}} &= \left(v_{\mathrm{p}}^{1}, v_{\mathrm{p}}^{2}, \ldots, v_{\mathrm{p}}^{N_{\mathrm{p}}}\right), \\ \mathbf{v}_{\mathrm{g}} &= \left(v_{\mathrm{g}}^{1}, v_{\mathrm{g}}^{2}, \ldots, v_{\mathrm{g}}^{N_{\mathrm{g}}}\right), \\ \mathbf{v}_{\mathrm{c}} &= \left(v_{\mathrm{c}}^{1}, v_{\mathrm{c}}^{2}, \ldots, v_{\mathrm{c}}^{N_{\mathrm{c}}}\right), \end{aligned} \tag{3.2}$$

and N_{p}, N_{g}, N_{c} are the number of proteins, genes, cells, respectively. Denote $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ as the edge set, we write the multimodal heterogeneous graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Let $N = (N_{\mathrm{p}} + N_{\mathrm{g}} + N_{\mathrm{c}})$ be the total number of nodes, the adjacency matrix $\mathbf{A}$ can be written as follows:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{\mathrm{p}} & \mathbf{S}^{T} & \mathbf{0} \\ \mathbf{S} & \mathbf{A}_{\mathrm{g}} & \mathbf{A}_{\mathrm{RNA}}^{T} \\ \mathbf{0} & \mathbf{A}_{\mathrm{RNA}} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{N \times N}, \tag{3.3}$$

where $\mathbf{A}_{\mathrm{p}} \in \mathbb{R}^{N_{\mathrm{p}} \times N_{\mathrm{p}}}$ is the protein-protein interaction graph; $\mathbf{A}_{\mathrm{g}} \in \mathbb{R}^{N_{\mathrm{g}} \times N_{\mathrm{g}}}$ denotes the gene-gene graphs mapped from STRING via MyGene.info (Lelong et al., 2022); $\mathbf{S} \in \mathbb{R}^{N_{\mathrm{g}} \times N_{\mathrm{p}}}$ is the encoding relationship between genes and proteins; and $\mathbf{A}_{\mathrm{RNA}} \in \mathbb{R}^{N_{\mathrm{c}} \times N_{\mathrm{g}}}$ represents the gene expression.

Node Feature initialization. With the graph structure, next we discuss how to initilize the node features. Since the RNA counts $\mathbf{X}_\mathrm{g}$ show drastic sparsity along with high dimension, it is not practical to be directly used as cell features. Therefore, we first denoise the data and reduce the dimension by Singular Value Decomposition (SVD). To alleviate the effects of cell-to-cell heterogeneity and extreme large counts, we conduct library size normalization and centered log-transformation. The preprocessed data $\bar{\mathbf{X}}$ is then passed through the SVD algorithm. Cell features are initialized with the reduced features $\mathbf{h}_\mathrm{c}^0 \in \mathbb{R}^{N_\mathrm{c} \times d_0}$. We then initialize gene features by the weighted sum of the reduced features $\mathbf{h}_\mathrm{g}^0 = \bar{\mathbf{X}}_\mathrm{g}^T \cdot \mathbf{h}_\mathrm{c}^0 \in \mathbb{R}^{N_\mathrm{g} \times d_0}$ with the normalized counts $\bar{\mathbf{X}}_\mathrm{g}$ as the weights. In the studied problem, proteins are the target modality for prediction; thus they are initialized randomly based on their indices.

3.3.2 Multimodal Transformers

In the proposed *scMoFormer*, we address the challenge of heterogeneity in the multimodal heterogeneous graph consisting of three distinct modalities: cells, genes, and proteins. To effectively handle this heterogeneity, we employ multiple transformers, each designed to process a specific data modality. The information obtained from these individual transformers is then aggregated through a cross-modality aggregation mechanism. Subsequently, we will present the transformers assigned to each modality and elaborate on how to coherently integrate them.

3.3.2.1 Cell Transformer

Transformer (Vaswani et al., 2017) has made significant achievements in the field of Natural Language Processing (NLP) in recent years. The attention mechanism can capture high-order and non-Euclidean connections between nodes, which is desired to explore the cell-cell relationships within single-cell multimodal data. Following the notations of the original transformer, we denote the queries, keys and input cell embeddings as $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{h}_\mathrm{c} \in \mathbb{R}^{N_\mathrm{c} \times d}$ with input dimension d, the scaled dot-product attention can be formulated as

$$\mathrm{Attn}(\mathbf{h}_\mathrm{c}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{h}_\mathrm{c}, \tag{3.4}$$

where $\mathbf{A}_{\text{attn}} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)$ is the attention matrix of cell-cell interactions. Despite being effective in various NLP tasks, the original transformer has limitations in scalability due to its relatively high space complexity of $O(N_c^2 + N_c d)$ and time complexity of $O(N_c^2 d)$. This issue considerably limits the application of transformers in single-cell multimodal analysis. Since typically multimodal data includes tens of thousands of cells, it is not applicable to implement the original transformer directly on cells.

To address the scalability issue, we employ generalized kernelizable attention (Choromanski et al., 2021) as a computationally efficient approximation of traditional attention. The attention blocks is kernelized in the form $\mathbf{A}(i, j) = \mathrm{K}\left(\mathbf{q}_i^\top, \mathbf{k}_j^\top\right)$, where $\mathbf{q}_i$ stands for the i^{th} row of query $\mathbf{Q}$ and $\mathbf{k}_j$ denotes the j^{th} row of key $\mathbf{K}$. The kernel $\mathrm{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ is specified as:

$$\mathrm{K}(\mathbf{x}, \mathbf{y}) = \mathbb{E}\left[\phi(\mathbf{x})^\top \phi(\mathbf{y})\right], \tag{3.5}$$

where $\phi : \mathbb{R}^d \to \mathbb{R}_+^r$ denotes the feature mapping. Let $\mathbf{Q}', \mathbf{K}' \in \mathbb{R}^{N_c \times r}$ be the approximate query and key with rows given as $\phi\left(\mathbf{q}_i^\top\right)^\top$ and $\phi\left(\mathbf{k}_i^\top\right)^\top$ respectively, the kernel approximation of cell attention is formulated as

$$\begin{aligned} &\text{Attn}(\mathbf{h}_c) = \widehat{\mathbf{D}}^{-1}\left(\mathbf{Q}'\left((\mathbf{K}')^\top \mathbf{h}_c\right)\right), \\ &\text{where} \quad \widehat{\mathbf{D}} = \text{diag}\left(\mathbf{Q}'\left((\mathbf{K}')^\top \mathbf{1}_{N_c}\right)\right). \end{aligned} \tag{3.6}$$

With kernel of dimension r, the space complexity and time complexity is reduced to $O(N_c r + N_c d + r d)$ and $O(N_c r d)$, respectively.

The generalized kernelizable attention boosts our cell transformer to linear space and time complexity, while still delivering results comparable to regular transformers (Choromanski et al., 2021). The attention mechanism in the model captures the intricate relationships between cells, resulting in an improved representation of the individual cells.

3.3.2.2 Gene Transformer and Protein Transformer

To leverage biological insights, we include STRING (Szklarczyk et al., 2023) as an addition to provide local information of genes and proteins. Although STRING provides solid prior networks, there may still exist data-related concerns when applied to sequencing data. For instance,

the NeurIPS 2022 competition dataset[3] contains $22,050$ genes, but only $13,101$ of these genes have interactions recorded in the STRING database. This issue also occurs in proteins, as the NeurIPS 2022 competition dataset collects 140 proteins and only 120 of them are found within the STRING (Szklarczyk et al., 2023) networks. This means that information about interactions within the remaining molecules is not available unless additional global information is included. To address the concerns, we utilized graph transformers to encode both the local and global information about genes and proteins. Specifically, following GraphGPS (Rampasek et al., 2022), we adapt the graph transformers for gene-gene and protein-protein graphs separately, i.e., gene transformer and protein transformer. Since the gene transformer and protein transformer have the same architecture, we will use the gene transformer to illustrate the details.

A gene transformer layer consists of two parallel components: a message-passing GNN block and a global attention block. The GNN block subtracts the gene interaction information from the prior local network, while the attention block learns global data-specific relationships by allowing each gene to attend to all other genes. The results from two blocks are summed together and then processed by fully connected layers to update the gene embeddings. Recall that the adjacency matrix of the gene-gene graph is denoted as $\mathbf{A}_{\mathrm{g}} \in \mathbb{R}^{N_{\mathrm{g}} \times N_{\mathrm{g}}}$, let $\mathbf{h}_g^{\ell} \in \mathbb{R}^{N_{\mathrm{g}} \times d_\ell}$ with dimension d_ℓ be the gene embedding of ℓ-th layer, the update functions are as follows:

$$\begin{aligned} \mathbf{h}_{\mathrm{g},M}^{\ell+1} &= \mathrm{GNN}^{\ell}\left(\mathbf{h}_{\mathrm{g}}^{\ell}, \mathbf{A}_{\mathrm{g}}\right), \\ \mathbf{h}_{\mathrm{g},T}^{\ell+1} &= \mathrm{Attn}^{\ell}\left(\mathbf{h}_{\mathrm{g}}^{\ell}\right), \\ \mathbf{h}_{\mathrm{g}}^{\ell+1} &= \mathrm{MLP}^{\ell}\left(\mathbf{h}_{\mathrm{g},M}^{\ell+1} + \mathbf{h}_{\mathrm{g},T}^{\ell+1}\right), \end{aligned} \tag{3.7}$$

where GNN^{ℓ} and Attn^{ℓ} represent the message passing GNN block and the global attention mechanism and MLP^{ℓ} is a 2-layer MLP block. The GNN block brings in prior biological insights, while the attention block allows resolving the expressivity bottlenecks caused by over-smoothing (Kreuzer et al., 2021) and over-squashing (Alon and Yahav, 2021).

Positional encoding is applied to the gene transformer layer to provide another solution to the expressivity bottlenecks among gene-gene graphs. Message-passing GNNs update gene node

[3]The dataset can be accessed at https://www.kaggle.com/competitions/open-problems-multimodal.

embeddings by aggregating local neighborhood representation given by gene knowledge graphs. By incorporating additional positional information, positional encoding helps to differentiate nodes that have the same local surroundings but distinct positions. Accompanied by prior gene-gene interaction networks, positional encoding distinguishes genes by the absolute position of each gene within the STRING network. This is important from a biological perspective as each gene functions differently. Here, we implement Laplacian positional encoding (Dwivedi and Bresson, 2020) and random walk positional encoding (Dwivedi et al., 2022). The Laplacian positional encoding captures the spectral information of graph Laplacian by its eigenvectors. Denote the graph Laplacian of input gene graph as $\mathbf{L}_{\mathrm{g}}$, the matrix factorization of $\mathbf{L}_{\mathrm{g}}$ is formulated as

$$\mathbf{L}_{\mathrm{g}} = \mathbf{I} - \mathbf{D}_{\mathrm{g}}^{-1/2}\mathbf{A}_{\mathrm{g}}\mathbf{D}_{\mathrm{g}}^{-1/2} = \mathbf{U}_{\mathrm{g}}^{T}\mathbf{\Lambda}_{\mathrm{g}}\mathbf{U}_{\mathrm{g}}, \tag{3.8}$$

where $\mathbf{D}_{\mathrm{g}}$ is the degree matrix of gene graph, $\mathbf{\Lambda}_{\mathrm{g}}$ and $\mathbf{U}_{\mathrm{g}}$ correspond to the eigenvalues and eigenvectors respectively. The gene node Laplacian positional encoding of dimension k is defined as the k smallest non-trivial eigenvectors. While Laplacian positional encoding embeds the positional information from the graph Laplacian, the random walk positional encoding tends to grasp the positional information given by the graph clusters. The random walk positional encoding of dimension k is defined with k-steps of the random walk as:

$$\mathbf{p}_i = \left[\mathrm{RW}_{ii}, \mathrm{RW}_{ii}^{2}, \cdots, \mathrm{RW}_{ii}^{k}\right] \in \mathbb{R}^{k}, \tag{3.9}$$

where $\mathrm{RW} = \mathbf{A}_{\mathrm{g}}\mathbf{D}_{\mathrm{g}}^{-1}$ is the random walk operator. The term RW_{ii}^{k} represents the landing probability of a gene node i to itself after k steps. The processed positional encoding is then combined to gene features through a fully connected layer.

3.3.2.3 Cross-Modality Aggregation

Transformers are constructed separately for each modality. To build a bridge among those transformers, we implement message passing GNNs among the links which connect nodes of distinct types. The information from cell transformer will denoise the prior knowledge by adding data-specific details into gene embeddings and protein embeddings. Meanwhile, information from gene transformer and protein transformer will bring in biological insights to cell transformer to

predict the target proteins. Particularly, we take the advantage of GraphSAGE (Hamilton et al., 2017) to transfer the information. Denote the i-th destination node as v_i and j-th source node as u_j. The information from the source nodes to the destination nodes is updated by:

$$\begin{aligned} \mathbf{h}_{\mathcal{N}(v_i)}^{(\ell+1)} &= \text{Aggregate}\left(\left\{\mathbf{h}_{u_j}^{\ell}, \forall\, u_j \in \mathcal{N}(v_i)\right\}\right) \\ \mathbf{h}_{v_i}^{(\ell+1)} &= \text{Update}\left(\mathbf{h}_{v_i}, \mathbf{h}_{\mathcal{N}(v_i)}\right) \end{aligned} \tag{3.10}$$

For node v_i, the message passing GNN aggregates information from its neighbors through aggregator function (Aggregate). The neighborhood information $\mathbf{h}_{\mathcal{N}(v_i)}^{(\ell+1)}$ is then combined to the embeddings $\mathbf{h}_{v_i}^{\ell}$ and processed by an updating procedure. The newly generated embeddings are normalized before next iteration. The message passing GNN modules facilitate communication between the transformers, enabling the transformers to leverage various forms of information during the training process.

Now we summarize the workflow of *scMoFormer*. As shown in Figure3.1, we apply transformers within each type of node and utilize message passing GNNs to form the bridges between transformers. In a formal way, for ℓ-th layer, denote the cell transformer as $\text{Trans}_{\text{c}}^{\ell}$, gene graph transformer and protein graph transformer as $\text{GT}_{\text{g}}^{\ell}$ and $\text{GT}_{\text{p}}^{\ell}$ respectively. Let $\text{MPG}_{\text{g}\to\text{p}}^{\ell}$ and $\text{MPG}_{\text{p}\to\text{g}}^{\ell}$ be the message passing GNN modules between genes and proteins, and $\text{MPG}_{\text{g}\to\text{c}}^{\ell}$ and $\text{MPG}_{\text{c}\to\text{g}}^{\ell}$ are for the links between genes and cells. For the purpose of prediction, we process the cell embeddings with MLP as cell readouts, where FC^{ℓ} represents the ℓ-th fully connected layer. The updates of node embeddings are achieved by

$$\begin{aligned} \mathbf{h}_{\text{g}}^{\ell+1} &= \text{GT}_{\text{g}}^{\ell}\left(\mathbf{h}_{\text{g}}^{\ell}, \mathbf{A}_{\text{g}}\right) + \text{MPG}_{\text{p}\to\text{g}}^{\ell}\left(\mathbf{h}_{\text{p}}^{\ell}, \mathbf{S}^{T}\right) + \text{MPG}_{\text{c}\to\text{g}}^{\ell}\left(\mathbf{h}_{\text{c}}^{\ell}, \mathbf{A}_{\text{RNA}}\right), \\ \mathbf{h}_{\text{c}}^{\ell+1} &= \text{Trans}_{\text{c}}^{\ell}\left(\mathbf{h}_{\text{c}}^{\ell}\right) + \text{MPG}_{\text{g}\to\text{c}}^{\ell}\left(\mathbf{h}_{\text{g}}^{\ell}, \mathbf{A}_{\text{RNA}}^{T}\right) + \text{FC}^{\ell}\left(\mathbf{h}_{\text{c}}^{\ell}\right), \\ \mathbf{h}_{\text{p}}^{\ell+1} &= \text{GT}_{\text{p}}^{\ell}\left(\mathbf{h}_{\text{p}}^{\ell}, \mathbf{A}_{\text{p}}\right) + \text{MPG}_{\text{g}\to\text{p}}^{\ell}\left(\mathbf{h}_{\text{g}}^{\ell}, \mathbf{S}\right), \end{aligned} \tag{3.11}$$

where $\mathbf{S}$, $\mathbf{A}_{\text{p}}$, $\mathbf{A}_{\text{g}}$, $\mathbf{A}_{\text{RNA}}$ are adjacency blocks defined in Section 3.3.1.

3.3.3 Prediction Layer

With the number of layers be L, the predictions are given by one extra full connected layer FC^{L+1} as

$$\begin{aligned} \mathbf{h}_{\mathrm{c}}^{L+1} &= \mathrm{Concat}\left(\mathbf{h}_{\mathrm{c}}^{\ell},\ \ell \text{ in } \{1, 2, \ldots, L\}\right), \\ \widehat{\mathbf{X}}_{\mathrm{p}} &= \mathrm{FC}^{L+1}\left(\mathbf{h}_{\mathrm{c}}^{L+1}\right), \end{aligned} \tag{3.12}$$

where $\widehat{\mathbf{X}}_{\mathrm{p}} \in \mathbb{R}^{N_{\mathrm{c}} \times N_{\mathrm{p}}}$ is the final prediction. To optimize the whole framework, we adapt a Mean Square Error (MSE) loss to measure the difference between the predictions and the ground-truth values:

$$\mathcal{L}\left(\widehat{\mathbf{X}}_{\mathrm{p}}, \mathbf{X}_{\mathrm{p}}\right) = \frac{1}{N_p N_c} \sum_{i=1}^{N_p} \sum_{i=1}^{N_c} \left(\mathbf{X}_{\mathrm{p}}^{ij} - \widehat{\mathbf{X}}_{\mathrm{p}}^{ij}\right)^2. \tag{3.13}$$

3.4 Experiment

In this section, we present the experimental results of *scMoFormer* against baselines on benchmark datasets. In particular, we aim to answer the following questions:

- **RQ1:** How does *scMoFormer* perform compare against baselines based on various evaluation metrics?
- **RQ2:** Given various choices of the positional encodings, how do they affect the performance of *scMoFormer*?
- **RQ3:** How does each of the model component impact the performance of *scMoFormer*?

Before presenting our experimental results and observations, we first introduce the experimental settings.

3.4.1 Experimental Settings

3.4.1.1 Datasets

We follow the setting of the NeurIPS multimodal single-cell integration competition of the year 2021 (Luecken et al., 2021) and 2022 and collect the joint measurements of gene expression and surface protein levels datasets from the competitions. Both datasets contain the raw counts, which represent the number of reads per gene per cell, as well as the normalized counts. For the NeurIPS

Table 3.1 Dataset Statistics.

	CITE	**GEX2ADT**
Number of RNA	22,050	13,953
Number of Proteins	140	134
Train Cells	42,843	66,175
Test Cells	28,145	1,000
RNA Zero Rate	0.780	0.904

2021 competition, we pick the data corresponding to the task of protein abundance prediction via gene expression and refer to it as "GEX2ADT". The processed data is centered and log-transformed for denoising purposes. For the competition in 2022, which we refer to as "CITE", the objective is to utilize CITE-seq (Stoeckius et al., 2017) data measured from days 2, 3, and 4 to predict the protein level on day 7 from different individuals. It is worth mentioning that the protein level testing data is not available during the completion of this work. Therefore, we simulate the competition scenario by treating the training data from day 4 as our testing set. The processed RNA data is centered and log-transformed, while the normalized protein levels are denoised and scaled by background (Kotliarov et al., 2020). We summarize the statistics of both datasets in Table 3.1.

3.4.1.2 Baselines

We evaluate the performance of *scMoFormer* against state-of-art multimodal prediction models among the task of using gene expression to predict surface protein levels. The selected baselines are as follows:

- **Cross-modal Autoencoders** (Yang et al., 2021), short for CMAE, incorporated multiple autoencoders to integrate multimodal data and utilized domain knowledge by adding discriminative loss to the training process to align shared markers or clusters among datasets.
- **BABEL** (Wu et al., 2021b) proposed a general framework for multimodal translation with modality-specific encoders and decoders. Note that initially, BABEL focused on RNA and ATAC-seq (Buenrostro et al., 2013) data. In this evaluation, we repurpose BABEL to the RNA to protein setting.
- **scMM** (Minoura et al., 2021) modeled the multimodal data with generative setting. We note

Table 3.2 Prediction evaluations based on different metrics (score ± std).

Dataset	CITE			GEX2ADT		
Metric	RMSE ↓	MAE ↓	Corr ↑	RMSE ↓	MAE ↓	Corr ↑
BABEL	1.67388 ± 0.00765	1.07777 ± 0.00602	0.87475 ± 0.00119	0.45387 ± 0.00738	0.30720 ± 0.00618	0.86144 ± 0.00274
CMAE	2.00874 ± 0.02088	1.21897 ± 0.01042	0.82502 ± 0.00843	0.51549 ± 0.00857	0.34855 ± 0.00488	0.81565 ± 0.00463
scMM*	-	-	-	0.64067 ± 0.00722	0.43407 ± 0.00307	0.68287 ± 0.00981
ScMoGNN	1.66634 ± 0.00741	1.07577 ± 0.00372	0.87788 ± 0.00113	0.42576 ± 0.01180	0.28819 ± 0.00976	0.87051 ± 0.00524
scMoFormer	**1.62720 ± 0.00731**	**1.05639 ± 0.00221**	**0.88552 ± 0.00080**	**0.41987 ± 0.00234**	**0.28289 ± 0.00223**	**0.87698 ± 0.00121**

*The scale of scMM predictions is not compatible with that of normalized protein levels.

that the input of scMM is restricted to raw counts by design, and the output predictions are scaled as centered log-transformed data.

- **ScMoGNN** (Wen et al., 2022a) involved domain knowledge like biological pathways to enhance the GNNs. The original ScMoGNN followed a transductive setting. In this work, we implement an inductive setting of ScMoGNN for a fair comparison with the baselines.

3.4.1.3 Parameter Setting

To benchmark the performance of baselines and *scMoFormer*, we uniformly employ inductive settings among both datasets. On the CITE dataset, we use the data measured on day 4 for testing and randomly split 80/20% of the data prior to day 4 for training and validation. On the GEX2ADT dataset, we randomly pick 15% of the training data for validation and evaluate the predictions on the testing set. For BABEL, the hidden dimension is tuned from $\{16, 32, 64, 128\}$. For CMAE, the weights of adversarial loss and reconstruction loss are chosen from $\{0.1, 1, 2.5, 5, 10\}$. For scMM, the hidden dimensions are tuned from $\{16, 32, 64, 128\}$. For ScMoGNN, the weight decay parameter of the optimizer is tuned from $\{5 \times 10^{-6}, 1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$.

3.4.2 Evaluation of Predictions

We evaluate the final protein-level prediction performance using Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). Meanwhile, because multimodal data usually suffers from the influence of batch effects and unbalanced measuring depth, the count's scale of each cell may vary significantly, which will substantially affect the RMSE and MAE metrics. Therefore, we also include the Pearson correlation coefficient (Corr), which is normalized by the mean and variance of the input on a per-cell basis, as a robust and scale-free metric to evaluate the predictions.

A lower RMSE or MAE score indicates a geometrically closer estimation of the protein levels, while a higher Corr score suggests a statistically more similar match to the actual value. We report the mean and the standard deviation of each metric across five different runs, and the results are illustrated in Table 3.2. The best performance is highlighted in bold.

To answer the first question, we note that our *scMoFormer* consistently outperforms all other baselines according to all three metrics on both datasets, indicating that *scMoFormer* successfully captures the quantitative characteristics of target protein levels given the input gene expression measurements. Particularly, for the CITE dataset, *scMoFormer* achieved significantly lower RMSE compared to the second-best model ScMoGNN, by 0.04. More importantly, *scMoFormer* achieved a significant improvement in terms of the Pearson correlation metrics over all other baselines, with a noticeably lower performance variation across runs, indicating the stability of our model.

We further analyze the performance of different models on proteins that are least well captured by any models. Specifically, for each model, we compute the RMSE for each protein separately and identify ten proteins that resulted in the highest average RMSE across all models. As shown in Figure 3.2, *scMoFormer* and ScMoGNN achieved relatively stable results and are consistently better compared to BABEl and CMAE.

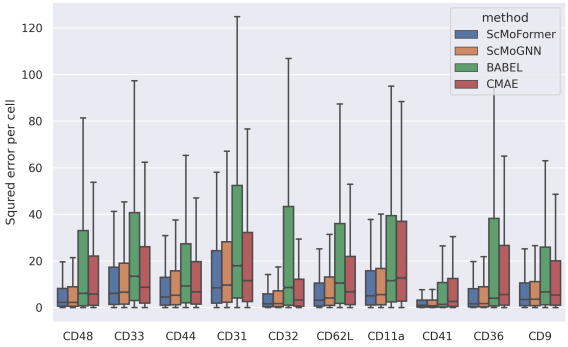


Figure 3.2 Least well-predicted protein performance comparison across models.

3.4.3 Positional Encoding

As mentioned in Section 3.3.2.2, we implement Laplacian positional encoding (Dwivedi and Bresson, 2020) and random walk positional encoding (Dwivedi et al., 2022) to capture positional information of prior knowledge graph. For ease of notation, we use the abbreviation PE to refer to

Table 3.3 Prediction RMSE results of different positional encoding (score ± std).

	CITE	**GEX2ADT**
Laplacian PE	1.63161 ± 0.01082	0.42025 ± 0.00243
Random Walk PE	1.63014 ± 0.01129	**0.41987 ± 0.00234**
w/o PE	**1.62720 ± 0.00731**	0.42202 ± 0.00399

the positional encoding. To benchmark the impact of the two types of PE among two datasets and answer the second question, we show the performance of *scMoFormer* with each PE and compare them with the scenario without any PE. The mean and standard deviation of RMSE scores of five runs are shown in Table 3.3.

According to the results, the influence of PE varies among datasets. The setting without PE reaches the best RMSE score on CITE dataset, while two types of PE both improve the performance on GEX2ADT dataset. Notice that in Table 3.1, the RNA zero rate of CITE dataset is significantly lower compared to the GEX2ADT dataset, providing the model with greater access to data-specific information. If the data contains sufficient information, then the neighborhood information from the GNNs alone is adequate and there is no need for the extra prior knowledge from the PEs. This is further supported by the observation that random walk PE performs better than Laplacian PE in both datasets. The Laplacian PE models global information by using the spectral information of the graph Laplacian, while the random walk PE encodes local information by accessing the landing probability of a k-step random walk. In cases where the prior knowledge may be noisy for downstream tasks, the local information alone is enough for predictions and the global structure becomes redundant.

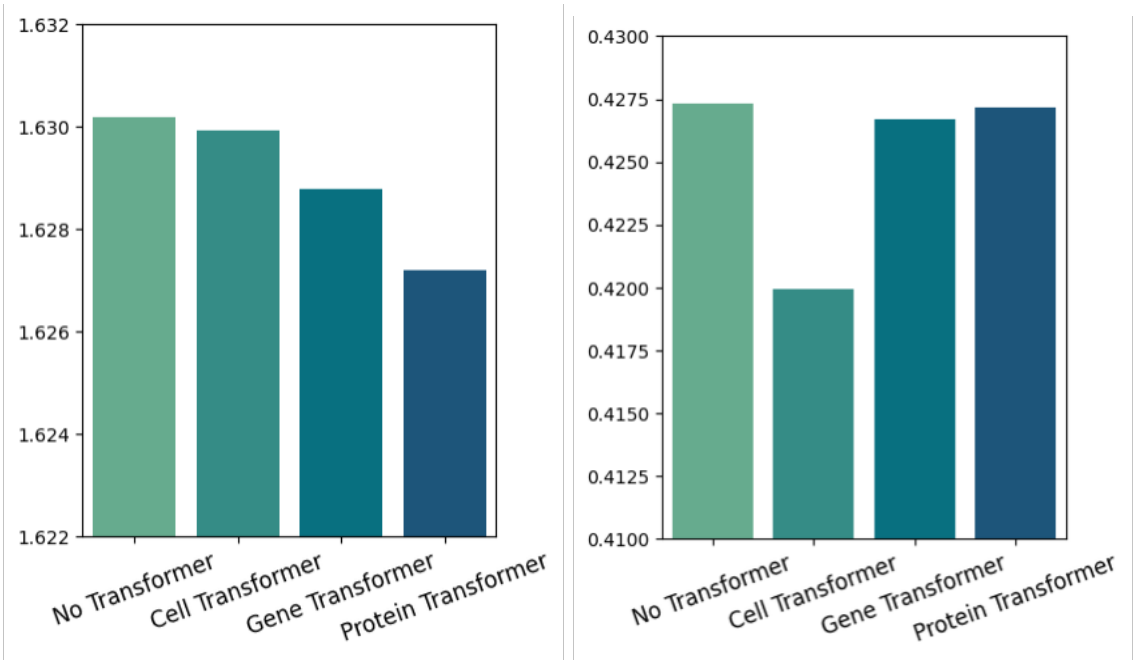
3.4.4 Ablation Study

Table 3.2 demonstrates that models that incorporate domain knowledge perform better in modality prediction compared to those that do not. BABEL, ScMoGNN, and *scMoFormer* are the three models that make use of domain knowledge, and they show improved performance compared to the other two models. Among these three models, ScMoGNN, which is based on GNNs, performs better than BABEL, while *scMoFormer* outperforms all other models with its combination of transformers and GNNs framework. Given that *scMoFormer* includes three transformers, this raises the questions: *Which transformer has the biggest impact on performance? How much do the*

transformers contribute to the improvement in performance?

3.4.4.1 Influence of Every Transformer

The propose multimodal transformers consist of three different transformers, namely the cell transformer, gene transformer and protein transformers. As our predictions are based on the cell readout, it is expected that each of the three transformers will have different levels of impact on the performance. To quantify the specific impact of a single transformer, we conduct an experiment by removing the other two transformers and measuring the prediction RMSE scores. The results of the evaluation, including the scores of three partial models and the model with no transformers, are summarized in Figure 3.3.



(a) CITE (b) GEX2ADT

Figure 3.3 Prediction RMSE↓ results of keeping only one Transformer.

The performance of the gene transformer and protein transformer is better than that of the cell transformer in the CITE dataset, while it is the opposite in the GEX2ADT dataset. This can be explained by the difference in RNA zero rate between the two datasets, as shown in Table 3.1. For the GEX2ADT dataset, the high RNA zero rate means less information, making the cell transformer crucial in increasing performance by drawing more information from the data. On the other hand,

Table 3.4 Prediction RMSE results of scMoFormer over GNNs (score ± std).

	CITE	GEX2ADT
GNN	1.63071 ± 0.01081	0.43070 ± 0.00237
GNN-prior	1.63020 ± 0.00672	0.42731 ± 0.00138
scMoFormer	**1.62720 ± 0.00731**	**0.41987 ± 0.00234**

the CITE dataset has a lower zero rate, meaning it provides more information, allowing the gene transformer and protein transformer to enhance the model by adding external biological knowledge.

3.4.4.2 How to Utilize Prior Knowledge

To answer the second question, we compare *scMoFormer* with two GNN-based models in Table 3.4. The model "GNN-prior" refers to the GNNs that built on the same graph in Section 3.3.1, while the "GNN" model is constructed using only the cell-gene graph without incorporating any prior information. The results show that the incorporation of prior knowledge into the graph results in a slight performance boost in both datasets. However, when multimodal transformers are included, the performance improvement is much more pronounced. This highlights the usefulness of prior knowledge and the importance of using transformers to effectively incorporate this information into the model.

3.5 Related Work

In this section, we go through some related works of our proposed framework, including GNNs, transformers, and other deep learning methods in single-cell analysis.

There is a growing number of deep learning-based methods for multimodal single-cell analysis in the community. For instance, scMDC (Lin et al., 2022) is an end-to-end autoencoder-based model with one encoder and two decoders. The encoder takes the concatenation of two modalities as an input and then reconstructs two modalities separately via two individual decoders. After training, the learned latent embedding would be used for clustering analysis. DCCA (Zuo et al., 2021) learns a coordinated but distinct representation for each omics data by mutually supervising each other on the basis of semantic similarity across embeddings, and then reconstructs back to the original dimension as output via a decoder for each omics data. Cross-modal Autoencoders (Yang

et al., 2021) utilize multiple autoencoders to map different modalities onto the same latent space, and incorporate prior knowledge through the use of adversarial loss and paired anchor loss in the training process. BABEL (Wu et al., 2021b) consists of two neural-network-based encoders and two decoders for translation and reconstruction. Both Cross-modal Autoencoders and BABEL focus on multimodal translation by adding interoperability constraints to train multiple encoders and decoders. Another approach, scMM (Minoura et al., 2021), captures nonlinear latent structures with variational autoencoders. It exploits a mixture-of-expert framework with a deep generative model and attains end-to-end learning by modeling raw counts of each modality. While these models have made significant advancements in multimodal integration, most of them are based on autoencoders and tend to overlook the underlying biological interactions of molecules and cells.

To capture the biological interactions of molecules and cells, there has been an increasing number of GNNs and transformer frameworks published in the field of single-cell analysis. One benefit of transformers applied in single-cell data is to capture long-range dependency in a global view. Another benefit is to interpret biological phenomena via the attention mechanism in transformers. From GNNs' perspective, graphs are natural to represent all kinds of data in single-cell data, like gene-to-gene graphs, cell-two-cell graphs, and cell-to-gene graphs. Another benefit of GNNs is to easily add domain knowledge or prior knowledge into graphs, like pathways between genes or overlaps between genes and peaks. For example, scGNN (Wang et al., 2021) models cell-cell interaction by incorporating GNN with multi-modal autoencoders. Specifically, scGNN builds a cell graph by capturing cell-type-specific regulatory signals and utilizes a Left-Truncated Mixture Gaussian model for scRNA-Seq data analysis. GLUE (Cao and Gao, 2022) pre-trains modality-specific variational autoencoders to get cell embeddings and then encodes a knowledge-based graph with GNNs. The next step involves performing an adversarial multimodal alignment of the cells through an iterative optimization process. In addition, ScMoGNN (Wen et al., 2022a) models the cell similarity and feature similarity by building a cell-feature graph and extracts information from data with a graph encoder. ScMoGNN takes advantage of gene pathway data as prior knowledge to enhance the graph and denoise the data. Moreover, scBERT (Yang et al., 2022) follows the

pre-training and fine-tuning paradigm of bidirectional encoder representations from transformers (BERT) for cell annotation of scRNA-seq data. The process of annotation involves extracting high-level patterns of cell types from the reference dataset. Different from these approaches which focus on single-modality data, we are the first to introduce transformers and GNNs to single-cell multimodal prediction.

3.6 Chapter Conclusion

In this chapter, we enhance our *scMoGNN* with a transformer model architecture and present *scMoFormer*, a multimodal transformer model for single-cell surface protein abundance from gene expression measurements. We combined the data with prior biological interaction knowledge from the STRING database into a richly connected heterogeneous graph and leveraged the transformer architectures to learn an accurate mapping between gene expression and surface protein abundance. Remarkably, *scMoFormer* achieves superior and more stable performance than other baselines on both 2021 and 2022 NeurIPS single-cell competition datasets.

CHAPTER 4

TRANSFORMERS FOR SINGLE-CELL SPATIAL OMICS REPRESENTATION LEARNING

Spatially resolved transcriptomics brings exciting breakthroughs to single-cell analysis by providing physical locations along with gene expression. However, as a cost of the extremely high spatial resolution, the cellular level spatial transcriptomic data suffer significantly from missing values. While a standard solution is to perform imputation on the missing values, most existing methods either overlook spatial information or only incorporate localized spatial context without the ability to capture long-range spatial information. Using multi-head self-attention mechanisms and positional encoding, transformer models can readily grasp the relationship between tokens and encode location information. In this paper, by treating single cells as spatial tokens, we study how to leverage transformers to impute spatial transcriptomic data. In particular, investigate the following two key questions: (1) *how to encode spatial information of cells in transformers*, and (2) *how to train a transformer for spatial transcriptomic imputation*. By answering these two questions, we present a transformer-based imputation framework, *SpaFormer*, for cellular-level spatial transcriptomic data. Extensive experiments demonstrate that *SpaFormer* outperforms existing state-of-the-art imputation algorithms on three large-scale datasets while maintaining superior computational efficiency.

4.1 Chapter Introduction

Spatial transcriptomic technologies have rapidly developed in recent years and emerged as next-generation tools for biomedical research. For instance, in-situ hybridization (ISH) based technology (Lubeck et al., 2014) produces detailed single-cell transcriptomic profiles along with the location of cells within a tissue, yielding deeper insights into cell identity and functionality than ever. However, as the number of profiled genes increases, the requirement for additional rounds of hybridization also increases, which elevates the potential for cumulative errors. As a result, ISH-based spatial transcriptomic data generally suffer from low mRNA capture efficiency and may miss a significant number of expressed genes, resulting in many false zero counts in observed gene expression data.

An effective approach to mitigate this problem is applying imputation methods to rectify the false zeros. There are numerous types of imputation methods for *conventional transcriptomic data* (i.e., scRNA-seq data). Nevertheless, those methods tend to yield suboptimal performance when imputing spatial transcriptomic data, as they do not leverage the presented spatial information. Notably, the spatial locations of cells provide important information about cell-cell interactions as well as cell similarities (Wen et al., 2022b), which have the great potential to advance the imputation process. For example, tumor cells usually show strong aggregation, and neighboring tumor cells with similar micro-environment in terms of ligands tend to have higher gene expression similarity than distant cells (Browaeys et al., 2020).

To effectively utilize spatial information, graph neural networks have been recently utilized for spatial transcriptomic analysis (Li et al., 2022; Wang et al., 2022). Concretely, graph neural networks (GNNs) are applied to the cell-cell neighboring graphs built on the spatial positions. However, these methods are limited to modeling a localized spatial context, which can be unfavorable for identifying long-range correlated cells. For example, Treg cells are scarce spatially in many tissues but still tend to be homologous and share similar gene expression profiles (Rudensky, 2011). Hence, it is desirable to capture the cell interactions from broader contexts. To achieve this goal, we employ the transformer model (Vaswani et al., 2017) for the studied problem. The transformer model was originally designed for textual data. It uses multi-head self-attention mechanisms to model relationships between input tokens and utilizes positional encoding to model the locations of tokens. Transformers are able to weigh the importance of each input token relative to all other tokens, rather than adjacent ones as in GNNs. In our studied problem, by treating cells as tokens, we can readily apply the transformer model to capture long-range correlations between cells.

In this chapter, we investigate two key questions when applying transformers to spatial transcriptomic imputation: (1) *how to encode spatial information of cells in transformers*, and (2) *how to train a transformer for transcriptomic imputation*. To answer the first question, one natural idea is to adopt the positional encodings from common transformers to encode spatial cellular coordinates. However, the spatial coordinates of cells are continuous and irregular, which are essentially different from the

discrete coordinates in the common practice of transformers. Therefore, efforts are still desired to design positional encodings for spatial transcriptomics. To address this issue, we investigate existing positional encodings, compare their advantages and disadvantages, and conduct comprehensive experiments to obtain a best practice for spatial transcriptomics. To answer the second question, we generalize the well-studied imputation models for conventional transcriptomic data into a flexible autoencoder framework, where we adopt a transformer as the encoder. Furthermore, we propose a new bi-level masking technique that can be plugged into the general autoencoder framework. With the solutions to two questions, our proposed framework, *SpaFormer*, consistently achieves outstanding imputation performance on three large-scale cellular-level spatial transcriptomic datasets.

4.2 Problem Statement

Before we introduce the notations and basic concepts, we first introduce the data we use. It is important to note that the focus of this chapter is on **high-resolution cellular-level datasets** (typically generated by ISH-based techniques), as opposed to the commonly studied spot-level datasets (e.g., 10X Visium, Seq-based data). In this work, we use two published and one unpublished dataset produced by the Nanostring CosMx (He et al., 2022b) platform. In order to obtain the cellular level gene expression, CellPose (Stringer et al., 2021) software is applied to conduct cell segmentation. Figure 4.1 gives an example about the raw image data and how cells are segmented.

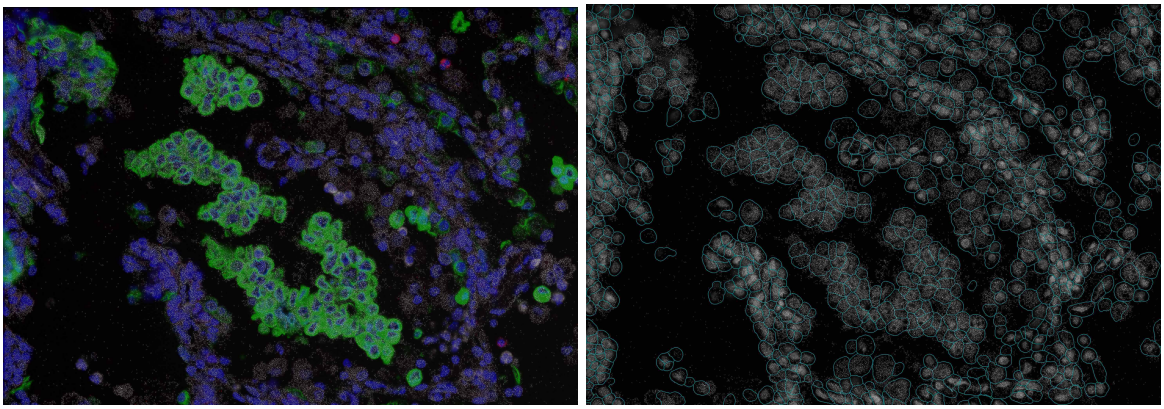
After pre-processing, a typical single-cell spatial transcriptomic dataset is comprised of two essential components, i.e., the gene expression of cells and the corresponding spatial locations. We denote the gene expression data as a matrix $\mathbf{X} \in \mathcal{R}^{N \times k}$, where N is the number of cells, and k is the number of genes. Hereby, $\mathbf{X}_{i,j}$ denotes the count of the j-th gene captured in the i-th cell. We use $\mathbf{C} \in \mathcal{R}^{N \times 2}$ to denote the two-dimensional coordinates of each cell, where those coordinates are based on the center position of each cell. Note that each dataset is composed of multiple field-of-views (FOVs). Each FOV contains thousands of cells and might not be adjacent to each other. Thus, we focus on units of FOVs by default.

In the spatial transcriptomic imputation problem, we suppose that a part of the input values in $\mathbf{X}$ are missing, denoted as a mask matrix $\mathbf{M} \in \{0, 1\}^{N \times k}$, where the value of $\mathbf{X}_{i,j}$ can only be observed

when $\mathbf{M}_{i,j} = 1$. A partially observed data matrix $\widetilde{\mathbf{X}}$ is defined as:

$$\widetilde{\mathbf{X}}_{i,j} = \begin{cases} 0 & \mathbf{M}_{i,j} = 0 \\ \mathbf{X}_{i,j} & \mathbf{M}_{i,j} = 1 \end{cases} \tag{4.1}$$

Our objective is to predict the missing values $\mathbf{X}_{i,j}$ at $\mathbf{M}_{i,j} = 0$, given the partially observed data $\widetilde{\mathbf{X}}_{i,j}$ and the spatial positions $\mathbf{C}$. *Note that in this work, we treat cells as tokens thus we use these two terms exchangeably in the remaining of the chapter.*



(a) Molecular image. (b) Cell segmentation.

Figure 4.1 A sample image of protein, RNA molecules, and segmented cells. Colors in sub-figure (a) indicate the protein molecules that are stained. These proteins contribute to the cell segmentation process, which results in the sub-figure (b). The final output from the pipeline consists of the position of each cell and a cell-by-gene count matrix.

4.3 Encoding Spatial Information in Transformers

An essential component of our *SpaFormer* framework is transformer (Vaswani et al., 2017) encoders, which were introduced in Section 3.3.2.1. When employing transformer models to spatial transcriptomic data, a critical challenge is how to encode spatial information in transformers. In a standard transformer, positional encodings (PEs) are added to the token embeddings to make use of the positional information of tokens. Different from chapter 3, in this chapter, positional encodings in the transformer mainly encode the spatial positions for every single cell. The coordinates of spatial transcriptomics are continuous and irregular, which are essentially different from the sequential data, image data, or graph data (as in Chapter 3). To tackle this issue, we investigate three general groups of positional encodings and demonstrate how we apply them to spatial transcriptomics. On top of

that, we introduce two advanced model-based positional encodings that address the limitations of previous positional encodings.

4.3.1 Patch-based Positional Encodings

We define patch-based positional encodings as encodings derived from discrete and regular patches. For example, ViT (Dosovitskiy et al., 2020) separates an image into 16×16 patches, so that patches lie in a regular grid. Positional encodings generated from these patch coordinates are considered patch-based positional encodings. To apply this approach to spatial transcriptomic data, we segment a whole input region, a.k.a, a field-of-view (FOV), into regular patches. In this chapter, we segment each FOV into 100×100 patches, where each patch contains 0.3-0.55 cell on average due to the distinct cell size of different tissues. Based on the patch coordinates, there are generally two ways to produce positional encodings (PE) for each patch, i.e., ***learnable PE*** and ***sinusoid PE***. *Learnable PE* is introduced in ViT (Dosovitskiy et al., 2020), where it learns a positional encoding for each individual position. *Sinusoid PE* is proposed in vanilla transformer (Vaswani et al., 2017), originally designed for sequential data, while it can be extended to 2-dimensional space (Carion et al., 2020).

4.3.2 Coordinate-based Positional Encodings

Coordinate-based positional encodings aim to generate positional encodings directly from the continuous spatial coordinates. In order to improve the generalizability, we normalize the spatial coordinates $\mathbf{C} \in \mathcal{R}^{n \times 2}$ to $[0, 1]$ by applying min-max normalization in each FOV. In this subsection, we discuss two specific types of positional encodings: ***naive PE*** and ***sinusoid PE***. *naive PE* projects normalized spatial coordinates to desired dimension d via an MLP or other transformation. Compared to patch-based *learnable PE*, *naive PE* imposes the ordinal relation between coordinates. On the other hand, coordinate-based *sinusoid PE* replaces discrete patch coordinates in the patch-based version with continuous spatial coordinates.

4.3.3 Graph-based Positional Encodings

Graph-based positional encodings are derived from the spatial adjacency graph that connects adjacent tokens. The motivation of graph-based positional encodings is that the spatial adjacency

graph conveys the relative position relations between tokens so that we can capture the positional information by encoding the spatial adjacency graph. To this end, we construct a spatial graph in which cell pairs are connected when the euclidean distance is smaller than 15-25μm. As a result, each cell is connected to 4-6 cells on average, depending on the specific tissue type. The resulting adjacency matrix is denoted as $\mathbf{A}$. Next, we present two ways to encode positional information from spatial adjacency graphs: ***random walk PE*** (RWPE) , which derives from landing probabilities of random walks, is proposed in LSPE (Dwivedi et al., 2021). This positional encoding is effective in encoding graph structures, however, it hardly encodes distance information. Furthermore, it is limited to structural information within the k-th order neighborhood and thus overlooks the global context. ***Laplacian PE*** (LapPE), which uses Laplacian eigenvectors as positional encodings (Kreuzer et al., 2021). Specifically, graph Laplacian $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{I}_n$ is an identity matrix, and $\mathbf{\Lambda} \in \mathcal{R}^{k\times k}$ and $\mathbf{U} \in \mathcal{R}^{n\times k}$ correspond to the eigenvalues and eigenvectors respectively, n is the number of nodes, and k is the number of top eigenvalues we select. The Laplacian eigenvectors constitute a local coordinate system that retains the overall structure of the graph. Thus, we can use the i-th row of eigenvector matrix $\mathbf{U}$ as the positional encoding for node i (a.k.a, cell i) in the graph. LapPE is generally a good choice despite being limited by sign ambiguity (Kreuzer et al., 2021). A straightforward approach to address this problem is to randomly flip the sign of eigenvectors during training, to force the model to be sign-invariant.

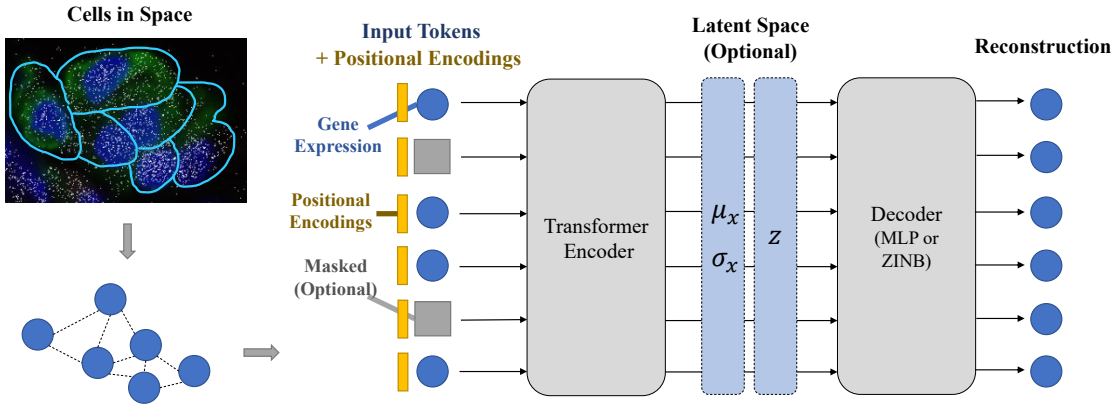


Figure 4.2 An illustration of our transformer-based autoencoder framework for spatial transcriptomics data imputation.

4.3.4 Model-based Positional Encodings

Despite that the aforementioned positional encodings can encode spatial information, each of them has certain limitations. Specifically, patch-based and coordinate-based positional encodings are dependent on the absolute location. However, studies have shown that relative positional encodings (Wu et al., 2021a; Luo et al., 2021), which consider the pair-wise relationships, generally perform better in various domains. One key advantage of relative positional encodings is their translation-invariant property. Translation-invariance refers to the property that positional encodings remain unchanged upon global translation of the coordinates, thereby enhancing the generalizability of transformers. Fortunately, graph-based positional encodings also achieve translation invariance since they are derived from the spatial adjacency graph, which is naturally based on invariant pair-wise distances. Nevertheless, RWPE cannot accurately encode pair-wise distance, while LapPE suffers from sign ambiguity.

To overcome the aforementioned limitations, we introduce two advanced models to generate positional encodings based on spatial adjacency graphs: ***SignNet*** (Lim et al., 2022) proposes a sign invariant and permutation invariant network to learn positional encodings from the Laplacian eigenvectors of a target graph. The resulting positional encodings inherit the advantages of LapPE while not suffering from sign ambiguity. ***Cond PE***, which stands for conditional positional encodings, originally proposed in CPVT (Chu et al., 2021). In the original version, CPVT added a simple convolution layer before vision transformers, to provide positions conditioned on the local neighborhood of each input token. We adopt this idea while substituting the visual convolution with a graph convolution, as it is more feasible to implement the graph convolution on a spatial graph compared to directly convolute the sparsely positioned cell tokens in 2-dimensional space. Consequently, we apply a graph attention network (Velickovic et al., 2018) to the spatial adjacency graph to generate initial embedding for each cell token. This embedding is considered a conditional positional encoding (Cond PE) that encodes spatial neighborhoods while achieving translation invariance.

Table 4.1 Comparison between positional encodings regarding the desired properties for spatial transcriptomics.

	Sources	Distance Aware.	Global Effec.	Translation Invari.	Structure Aware.	Other Issues
Sinusoid PE	Patch / Coordinate	✓	✓	✗	✗	
Learnable PE	Patch	✓	✓	✗	✗	
Naive PE	Coordinate	✓	✓	✗	✗	
RWPE	Graph	✗	✗	✓	✓	
LapPE	Graph	✓	✓	✓	✓	Sign Ambiguity
Relative PE	Distance	✓	✓	✓	✗	Scalability
SignNet (Lim et al., 2022)	Model	✓	✓	✓	✓	
Cond PE	Model	✓	✗	✓	✓	

4.3.5 Summary

We summarize all aforementioned positional encodings in Table 4.1. We consider four main properties when we compare these methods, namely distance awareness, global effectiveness, translation invariance, and spatial structure awareness. These properties have already been mentioned in the previous introduction. In conclusion, we investigate four types of position encoding to capture spatial information in transformers, which possess different properties and mostly demonstrate good potential as position encodings for spatial transcriptomics. We will apply distinct positional encodings in our *SpaFormer* framework in order to gain the best practices for position encoding in spatial transcriptomes.

4.4 Proposed Transformer Framework

In this section, we introduce our *SpaFormer* framework. An overview of *SpaFormer* is illustrated in Figure 4.2. In *SpaFormer*, we first extract the spatial positional encoding for each cell, using different methods as discussed in Section 4.3. Then cell embeddings are initialized with gene expressions and positional encodings, while some cells are selectively masked depending on the specific setting. Next, a transformer encoder is applied to encode both cellular profiles and intercellular contexts into the latent space. Finally, a decoder reconstructs the input (or masked) information based on the latent variables. In the following, we first introduce our general framework that generalizes popular transcriptomic imputation models in Section 4.4.1. Then based on the general framework, we propose a new bi-level masking strategy in Section 4.4.2, which is particularly suitable for spatial transcriptomic data imputation.

4.4.1 Generalized Autoencoder Framework

The most popular architecture for deep-learning-based transcriptomic data imputation methods (Molho et al., 2024) is autoencoder, due to its prevalence in data denoising and missing data applications. Existing methods introduced a few variants of autoencoders, including variational autoencoders (VAE) (Kingma and Welling, 2013; Lopez et al., 2018) and ZINB-based (zero-inflated negative binomial) autoencoders (Eraslan et al., 2019), while they often lack a systematic comparison for these autoencoder variants. In order to compare the performances of different autoencoders on the spatial transcriptomic imputation task, our *SpaFormer* framework generalizes all these variants of autoencoders.

Our general framework takes a batch of cells as input. To be consistent with Section 4.2, we denote the input as $\widetilde{\mathbf{X}} \in \mathcal{R}^{n \times k}$, where n is the number of cells in the input field-of-view (FOV). Note that here we omit the positional encodings, which should also be included in the input matrix. An encoder q_θ projects the input data to latent space, resulting in $\mathbf{Z} \in \mathcal{R}^{n \times d}$, where d is the latent dimension. A decoder p_ϕ then generates an output $\widehat{\mathbf{X}} \in \mathcal{R}^{n \times k}$ from the latent space, where we expect $\widehat{\mathbf{X}}$ to be identical with the input matrix $\widetilde{\mathbf{X}}$. The overall framework can be simply written as:

$$\mathbf{Z} = q_\theta\left(\widetilde{\mathbf{X}}\right) \tag{4.2}$$

$$\widehat{\mathbf{X}} = p_\phi\left(\mathbf{Z}\right) \tag{4.3}$$

Next, we proceed to introduce how we implement various autoencoder variants under the general framework.

Vanilla autoencoders. In the case of vanilla autoencoder, we implement q_θ with a transformer and p_ϕ with a multi-layer perception (MLP). Note that we have selected an asymmetrical encoder-decoder architecture, as we believe that the model requires a greater ability to utilize spatial information for denoising during the encoding process. While during decoding, the contextual information of each cell has already been included in the latent space, making decoding easier. The

loss function for vanilla autoencoders can be written as:

$$\mathcal{L}_{\text{MSE}} = \left\| \widehat{\mathbf{X}} - \widetilde{\mathbf{X}} \right\|_F^2 \tag{4.4}$$

ZINB-based autoencoders. Previous studies (Eraslan et al., 2019; Lopez et al., 2018) pointed out that the data distribution of the transcriptomic data can be approximated by zero-inflated negative binomial (ZINB) distribution because the data are discrete, overdispersed and contain many zero values. Therefore, we can adopt a ZINB-based autoencoder to leverage this prior information. A ZINB distribution is defined as:

$$\text{NB}(x \mid \mu, \theta) = \frac{\Gamma(x+\theta)}{x!\Gamma(\theta)} \left(\frac{\theta}{\theta+\mu}\right)^{\theta} \left(\frac{\mu}{\theta+\mu}\right)^{x} \tag{4.5}$$

$$\text{ZINB}\,(x \mid \pi, \mu, \theta) = \pi\delta_0(x) + (1-\pi)\,\text{NB}(x) \tag{4.6}$$

where μ and π denote mean and dispersion, π is the weight of the point mass at zero, and δ_0 generates constant 0.

In our *SpaFormer* framework, to adapt a vanilla autoencoder to a ZINB-based autoencoder, the encoder p_ϕ remains unchanged, while a ZINB decoder implements the decoder q_θ instead of an MLP decoder. A ZINB decoder takes the latent code $\mathbf{Z}$ as input and generates an intermediate result $\widehat{\mathbf{H}} \in \mathcal{R}^{n \times d'}$ via MLP. On top of that, ZINB decoders have three fully-connected output layers to estimate the parameters of ZINB distribution. Let $\mathbf{\Pi}$, $\mathbf{M}$ and $\mathbf{\Theta}$ be the three parameters π, μ, θ of ZINB distribution estimated by the three output layers respectively, then the overall loss function of ZINB-based autoencoder changes to the negative log-likelihood of the ZINB distribution, formulated as:

$$\mathcal{L}_{\text{ZINB}} = -\log\Big(\text{ZINB}(\widetilde{\mathbf{X}} \mid \mathbf{\Pi}, \mathbf{M}, \mathbf{\Theta})\Big) \tag{4.7}$$

In the inference stage, the estimated mean matrix $\mathbf{M}$ is selected as the imputation output.

Variational autoencoders. VAEs have been wildly applied in single-cell transcriptomic analysis (Lopez et al., 2018, 2019; Molho et al., 2024) since they are robust to technical noise and bias. In our *SpaFormer* framework, we optionally transform an autoencoder framework to a VAE by making modifications to the latent space and loss function. Specifically, two additional

fully-connected layers are appended to the encoder to estimate the mean and variance of latent variables:

$$\boldsymbol{\mu}_z = q_\theta\left(\widetilde{\mathbf{X}}\right) \cdot \mathbf{W}_{\mu_z}, \quad \boldsymbol{\sigma}_z = \exp\left(q_\theta\left(\widetilde{\mathbf{X}}\right) \cdot \mathbf{W}_{\sigma_z}\right) \tag{4.8}$$

where $\mu_{\mathbf{z}}$ and $\sigma_{\mathbf{z}}$ are mean and variance respectively. The latent variables $\mathbf{Z}$ are then sampled from the estimated Gaussian distribution $\mathcal{N}\left(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z\right)$, where we apply reparametrization trick (Kingma and Welling, 2013) to keep the gradient descend possible. Furthermore, an additional Kullback-Leibler (KL) divergence loss term is added to regularize the distribution of latent variables, which is:

$$\mathcal{L}_{\mathrm{KL}} = D_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z\right) \| \mathcal{N}\left(0, 1\right)\right) \tag{4.9}$$

The loss functions of vanilla autoencoder and ZINB-based autoencoder then turn to: $\mathcal{L} = \mathcal{L}_{\mathrm{MSE}} + \beta\mathcal{L}_{\mathrm{KL}}$, or $\mathcal{L} = \mathcal{L}_{\mathrm{ZINB}} + \beta\mathcal{L}_{\mathrm{KL}}$, respectively, where β is the KL loss weight. In the inference stage, we directly employ the estimated mean $\mu_{\mathbf{z}}$ as latent variables, instead of sampling.

4.4.2 Bi-level Masked Autoencoders

Inspired by the tremendous success of the masked autoencoding paradigm in NLP (Devlin et al., 2019) and Computer Vision (He et al., 2022a), we propose to incorporate a new masked autoencoder model into our generalized framework. Specifically, masked autoencoders are a form of more general denoising autoencoders (Vincent et al., 2008), which adopt a simple concept to remove a proportion of the data and then learn to recover the removed parts. The idea of masked autoencoders is also natural and applicable in spatial transcriptomics since we expect a powerful model to be able to recover missing cellular profiles from neighboring cells sharing the same microenvironment and homologous cells with the same cell state. Note that the objective of the masked autoencoder is highly consistent with the process of data imputation. Hence, it is very promising that the model will be trained with the specific capability for imputing missing values.

Despite that masked autoencoders are highly suitable for training an imputation model, the characteristic of spatial transcriptomics adds difficulties to the utilization of masked autoencoders. Compared to NLP and vision, it is much more difficult to recover cell profiles solely based on contexts. One reason is that the external signals a cell receives depend not only on the concentration

of ligands in the microenvironment but also on the amount of receptors on its membrane. Therefore, it's hardly possible to identify intercellular correlations when the cell profile is completely masked out. To address this issue, we made a modification to the standard masked autoencoder, which we call it a bi-level masking strategy.

In a bi-level masking strategy, for each input batch of tokens, we first determine which cells are to be masked. Then, instead of thoroughly masking the tokens, we selectively mask out a certain proportion of features in those tokens, leaving the potential for the model to recover underlying intercellular correlations. Specifically, we first sample a mask vector $\widetilde{\mathbf{m}}^{\text{token}} \in \{0,1\}^n$ from a Bernoulli distribution with $p = \theta$, where n is the number of input tokens, θ is the probability that a token is selected to be masked. Next, $\widetilde{\mathbf{M}}^{\text{feat}} \in \{0,1\}^{n \times k}$ is sampled from another Bernoulli distribution with $p = \gamma$, where γ is the probability that a feature is masked in a selected token. Lastly, we combined $\widetilde{\mathbf{m}}^{\text{token}}$ and $\widetilde{\mathbf{M}}^{\text{feat}}$ as:

$$\widetilde{\mathbf{M}}_{i,j} = 1 - \widetilde{\mathbf{m}}_i^{\text{token}} \cdot \widetilde{\mathbf{M}}_{i,j}^{\text{feat}} \tag{4.10}$$

where $\widetilde{\mathbf{M}}$ is the finalized mask matrix. The input features $\widetilde{\mathbf{X}}_{i,j}$ are then masked when $\widetilde{\mathbf{M}}_{i,j} = 0$, resulting in a new feature matrix $\widetilde{\mathbf{X}}' \in \mathcal{R}^{n \times k}$ written as:

$$\widetilde{\mathbf{X}}' = \phi\left(\widetilde{\mathbf{M}} \odot \widetilde{\mathbf{X}}\right) \tag{4.11}$$

where $\odot$ indicates element-wise multiplication, and ϕ refers to a rescaling technique that maintains the mean of the input unchanged for each cell. Notably, when $\gamma = 1$, our masked autoencoder is equivalent to MAE for vision (He et al., 2022a). When $\theta = 1$, our masked autoencoder is equivalent to a denoising autoencoder (Vincent et al., 2008). Such a bi-level masking strategy provides our framework with greater flexibility.

During training, masked features $\widetilde{\mathbf{X}}'$ are used as initial token embeddings. Therefore, it should be plugged into Eq. 4.2 to replace $\widetilde{\mathbf{X}}$. Note that our bi-level masking strategy perfectly fits the overall autoencoder framework, which means we can combine the masked autoencoder with other components, i.e., VAE and ZINB decoders. When enabling the masked autoencoder, we need to

change the reconstruction loss in Eq. 4.4 to:

$$\mathcal{L}_{\text{MSE}} = \left\| (1 - \mathbf{M}') \odot \left(\widehat{\mathbf{X}} - \widetilde{\mathbf{X}} \right) \right\|_F^2 \tag{4.12}$$

where we only calculate mean squared error (MSE) for the prediction of masked values. For inference, we simply dismiss the bi-level masking strategy. Instead, the unmasked input $\widetilde{\mathbf{X}}$ is enabled, which provides extra information for imputation.

Table 4.2 Imputation performance on three CosMx datasets.

	Methods	**Lung 5**			**Kidney 1139**			**Liver Normal**		
		RMSE↓	Pearson↑	Cosine↑	RMSE↓	Pearson↑	Cosine↑	RMSE↓	Pearson↑	Cosine↑
Baseline	Raw	0.3758	-	-	0.3747	-	-	0.3507	-	-
scRNA-seq Methods	scImpute	0.3245	0.444	0.5214	0.311	0.4824	0.5714	0.3048	0.4437	0.5074
	SAVER	0.3213	0.4564	0.5269	0.3106	0.4887	0.5689	0.2909	0.5462	0.5864
	scVI	0.2861	<u>0.6231</u>	<u>0.6661</u>	0.2901	0.5834	0.648	0.2797	0.5749	0.6224
	DCA	<u>0.2858</u>	0.6223	0.6648	<u>0.2852</u>	<u>0.5985</u>	<u>0.6597</u>	0.2542	0.657	0.688
	GraphSCI	0.3957	0.1334	0.3081	0.3624	0.2403	0.4128	0.3347	0.3707	0.443
	scGNN		OOM *			OOM *			OOM *	
Reference-based Methods	gimVI	0.3170	0.5320	0.5917	0.4387	-0.0104	0.1967	0.4542	-0.0015	0.1170
	Tangram	0.3905	0.3161	0.3883	0.3639	0.4037	0.4798	0.3373	0.4830	0.5236
	SpaGE	0.4420	0.3942	0.4572	0.5096	0.4311	0.5065	0.6433	0.5483	0.5899
Spatial Methods	Sprod		OOT **			OOT**			OOT**	
	SEDR	0.3245	0.4949	0.5499	0.3116	0.5101	0.5826		OOM*	
Ours	SpaGAT	0.2865	0.6047	0.6518	0.2859	0.5852	0.6506	<u>0.2241</u>	<u>0.7624</u>	<u>0.7829</u>
	SpaFormer	**0.2785**	**0.6363**	**0.6786**	**0.2794**	**0.6108**	**0.671**	**0.2117**	**0.7793**	**0.7973**

*Run out of 300G CPU Memory. **Run more than 48 hours on 128 CPUs.

4.5 Experiment

4.5.1 Experimental settings

Datasets. We validate the proposed approach on three spatial transcriptomic datasets generated by the CosMX platform (He et al., 2022b) from Nanostring, and can be accessed on their official website[1]. These datasets differ in their scales and sources of tissues, which highlights the comprehensiveness of our experiments. The data statistics are presented in Table 4.3.

Baselines. To evaluate the effectiveness of *SpaFormer*, we compare it with the state-of-the-art spatial and non-spatial transcriptomic imputation models: scImpute (Li and Li, 2018), SAVER (Huang et al., 2018), scVI (Lopez et al., 2018), DCA (Eraslan et al., 2019), GraphSCI (Rao

[1]Dataset can be downloaded from https://nanostring.com/products/cosmx-spatial-molecular-imager/nsclc-ffpe-dataset.

Table 4.3 Dataset statistics.

Dataset	Cell Num.	FOV Num.	Gene Num.	Zero Ratio
Lung 5	99,656	30	960	86.74%
Kidney 1139	61,283	18	960	83.49%
Liver Normal	305,730	244	1000	86.39%

et al., 2021), scGNN (Wang et al., 2021), gimVI (Lopez et al., 2019), Tangram (Biancalani et al., 2021), SpaGE (Abdelaal et al., 2020), Sprod (Li et al., 2022), and SEDR (Xu et al., 2024). The first six methods are designed for scRNA-seq data imputation. gimVI, Tangram, and SpaGE aim to impute spatial transcriptomic data via external reference scRNA-seq data. Sprod and SEDR build cell-cell graphs among spatial neighbors to help imputation. In addition to these published baselines, we create a new baseline model SpaGAT which uses the same bi-level masking autoencoder framework as *SpaFormer*, based on a graph neural network encoder with spatial graphs. Specifically, we implement a graph attention network (Velickovic et al., 2018) as the encoder. Since the graph attention network is a localized version of transformers, SpaGAT can be considered an ablation study for our *SpaFormer* model.

Implementation Settings. Before we conduct the experiment, we first randomly mask 30% of the data to create the partially observed data, while the original masked data are considered as ground truth. All training and inference processes are then conducted on the partially observed data, and those dropped values are kept for evaluation. Based on the partially observed data, we further conduct preprocessing methods, according to the recommended settings of each specific model. For our own SpaGAT and *SpaFormer*, we first normalize the total RNA counts of each cell, and then apply log1p transform. In addition, considering that the output format of the baseline models varies between raw counts and log-transformed values, we uniformly conduct postprocessing to make sure all imputed data and ground-truth data are normalized and log-transformed. By default, SpaGAT and *SpaFormer* adopt a bi-level masked autoencoder, while *SpaFormer* chooses a Cond PE as positional encoding. Reproducibility details and codes for all methods can be found on our GitHub repository[2] as well as Appendix B.1.

[2]Codes and hyperparameter settings are available at https://github.com/wehos/CellT.

Evalutaion. For imputation, the rooted mean squared error (RMSE), Pearson correlation coefficients (Pearson), and cosine similarity (Cosine) metrics are calculated based on the predictions of the masked values. For clustering, we first reduce the dimension of imputed data with PCA and then construct kNN graph based on the first 10 PCs. The clustering result is obtained from Leiden algorithm on the kNN graph, where we conduct a grid search to find the optimal resolution for Leiden. Clustering metrics normalized mutual information (NMI) and adjusted Rand index (ARI) are then calculated based on the clustering results and predefined cell type labels. Lastly, all deep learning models are evaluated with 5 random seeds, and the average performance is reported, while statistical models are evaluated only once. The standard deviation is presented in Appendix B.2.

4.5.2 Imputation Performance

In Table 4.2, we present the experimental results. It is shown that our *SpaFormer* consistently outperforms other baselines by showing a better imputation performance. Aside from that, there are several interesting observations. (1) scVI and DCA consistently present suboptimal performance, outperforming some advanced methods, e.g., GraphSCI. Since both DCA and scVI adopt ZINB-based autoencoders, this demonstrates the effectiveness of ZINB-based autoencoders. (2) Reference-based methods, i.e., gimVI, Tangram, SpaGE, generally do not work well on our highly noisy spatial transcriptomic dataset. These methods map spatial transcriptomic data to scRNA-seq reference datasets and impute the missing values based on reference data. However, the single-cell spatial transcriptomics data has a substantially low cover rate of RNA molecules, which causes distinct distributions between spatial transcriptomic data and reference scRNA-seq data. These results indicate the limitation of these reference-based imputation approaches. (3) SpaGAT, the non-transformer version of *SpaFormer*, presents fairly good performance. This demonstrates the advantage of our bi-level mask-autoencoder framework. (4) Recent methods specifically designed for spatial transcriptomics either suffer from scalability issues or demonstrate suboptimal performance. Therefore, there remains significant space for exploration in the field of single-cell spatial transcriptomic imputation.

4.5.3 Scalability Analysis

Scalability is critical when deploying methods on single-cell spatial transcriptomic data since the datasets often contain tens of thousands of cells. We considered scalability as an important factor when conducting the evaluation. Specifically, we set limits on the runtime and memory consumption of the model, i.e., 48 hours of running time, 300G CPU memory, and 45G GPU memory. Notably, we encountered scalability issues in scGNN and Sprod methods even with the smallest Kidney dataset. scGNN and SEDR train neural networks on GPUs, however, both of them may run out of 300G CPU memory when building the cell-cell graph. Sprod separates data into numerous batches and purely runs on CPUs. We allocate 128 CPU processors for it, yet it fails to finish running in 48 hours. In contrast, our proposed *SpaFormer* exhibits **linear complexity w.r.t the number of cells**, thanks to our efficient autoencoder architecture and the Performer (Choromanski et al., 2021) encoder.

In addition, we present the empirical computational overhead of all methods on the largest Liver dataset in Table 4.4,. Specifically, we record the running time, peak CPU memory, and peak GPU memory. Peak CPU memory refers to the physical CPU memory consumption. Peak GPU Memory refers to the peak consumption of GPU memory resources. Running time refers to the training and inference for the imputation, excluding preprocessing time. The experiments are conducted on an RTX 8000 GPU card and 128 AMD EPYC CPU cores with 300G CPU memory. Each setting is repeated 4 times and the average running time is reported. Our *SpaFormer* method demonstrates superior computational efficiency, especially running time.

4.5.4 Clustering Performance

In addition to imputation performance, we conduct unsupervised clustering on the imputed data to validate whether the imputation can help recover cell type information. For evaluation, cell type labels accompanied by the dataset are considered as ground truth. It is expected that well-imputed data can better recover the cell clustering structures. As shown in Figure 4.3, although all models enhance the clustering performance as compared to the unimputed raw data, clustering based on *SpaFormer* achieves the best performance.

Table 4.4 Computational overhead on Liver dataset. Methods are ranked by running time.

Methods	Running Time (hour)	Peak CPU Memory (Mb)	Peak GPU Memory (Mb)
SpaFormer	**0.068**	6,612	1,678
scVI	0.119	**3,511**	254
Spage	0.179	12,493	N/A
DCA	0.236	7,508	41,850
Tangram	0.551	4,507	17012
gimVI	0.971	6,600	250
SAVER	1.975	245,760	N/A
scImpute	2.917	269,490	N/A
GraphSCI	9.592	18,329	N/A
scGNN	N/A	> 307,200	N/A
SPROD	>48	N/A	N/A
SEDR	N/A	>307,200	N/A

In addition, an interesting observation is that despite reference-based methods (i.e., SpaGE and gimVI) do not achieve impressive performance on imputation metrics, their clustering performance is relatively better. This indicates that they may have succeeded in discovering reference cells from scRNA-seq data, however, they still suffer from the distribution gap between the reference and query data.

4.5.5 Ablation Study

4.5.5.1 Positional Encodings

As we mentioned in Section 3.4.3, we conduct experiments on three datasets to compare the performance of different positional encodings. According to our experiments in FIgure 4.4, Cond PE and LapPE outperform other positional encodings w.r.t overall performance on three datasets, indicating the effectiveness of graph-based PEs. Meanwhile, Cond PE outperforms SignNet, suggesting that global effectiveness might not be a desired property for spatial transcriptomics, as we discussed in Section 3.4.3. In conclusion, we select Cond PE as a default setting in *SpaFormer*. Besides, a 2-dimensional Sinusoid PE can also be a good choice for spatial transcriptomics since

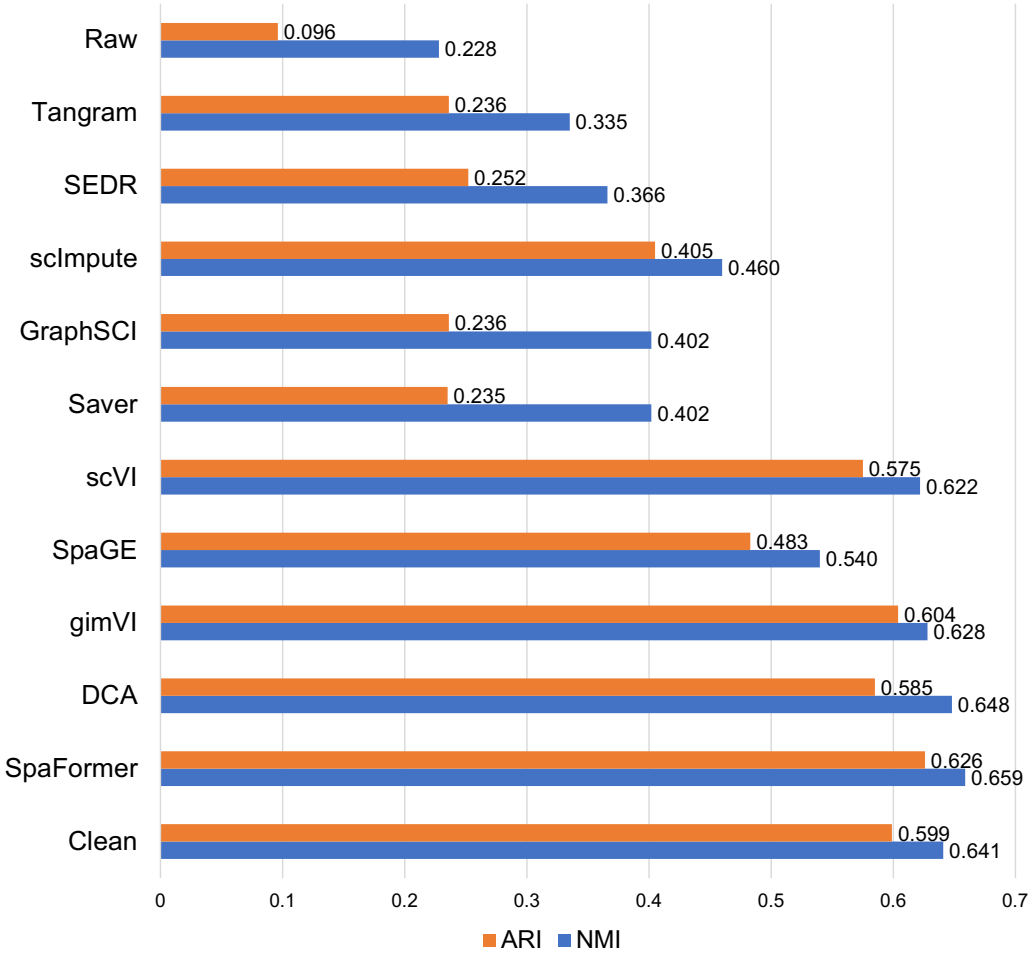


Figure 4.3 Clustering performance on imputed data of Lung dataset.

it has fewer parameters than others. Meanwhile, the gap between transformers with and without positional encodings is less significant than we expected. Therefore, there is still large room for exploration in positional encodings for spatial transcriptomic data.

4.5.5.2 Autoencoder Frameworks

Another highlight of our *SpaFormer* framework is that we generalize different popular autoencoder-based models. Based on our general framework, we conduct a thorough ablation study on the Liver dataset. The experimental results in Figure 4.5 indicate that various variations of autoencoders achieved optimal performance when combined with masking. Additionally, the simple bi-level mask autoencoder obtained the best results. Therefore, we recommend the masked autoencoder as the default setting for our *SpaFormer* framework.

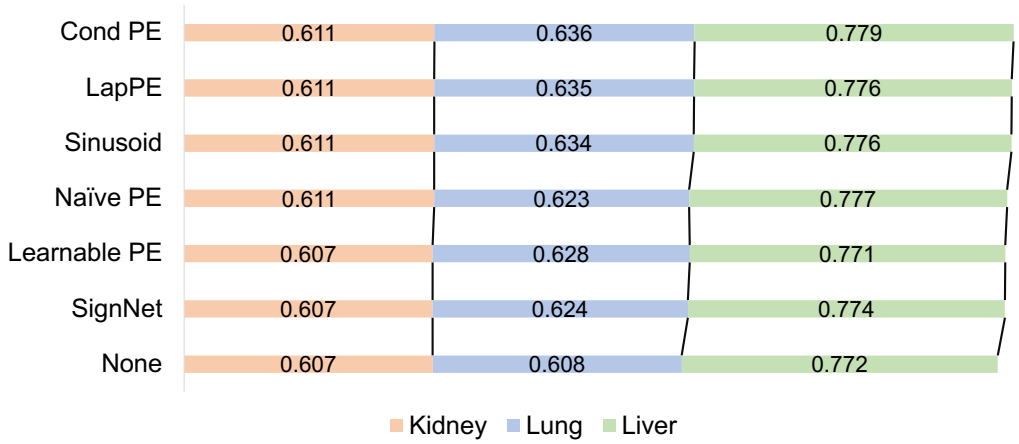


Figure 4.4 Comparison between different positional encodings on three datasets. Values indicate Pearson correlation coefficient.

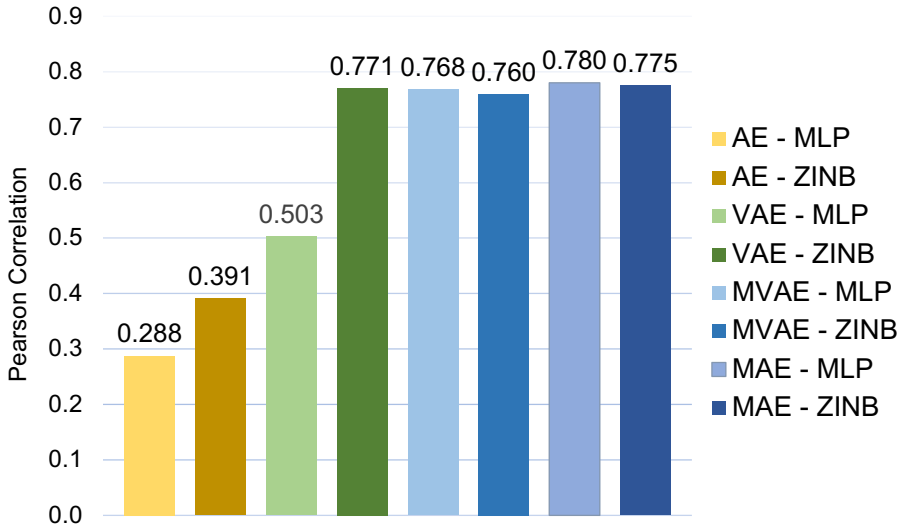


Figure 4.5 Ablation study on different autoencoder variants, i.e., vanilla autoencoders (AE), variational autoencoders (VAE), bi-level masked autoencoder (MVAE) and masked autoencoder (MAE). For each variant, we implement both MLP and ZINB decoders. Values indicate the Pearson correlation coefficient on the Liver dataset.

4.5.5.3 Mask Ratio

Lastly, we conduct a parameter analysis on token masked ratio θ and feature masked ratio γ in Section 4.4.2 to demonstrate the importance of our bi-level masking strategy. As shown in Figure 4.6, joint tuning θ and γ results in an optimal Pearson correlation score. In our main experiment, we select $\theta = 0.5$, $\gamma = 0.5$ as default parameters, since this setting consistently achieves optimal performance on three datasets.

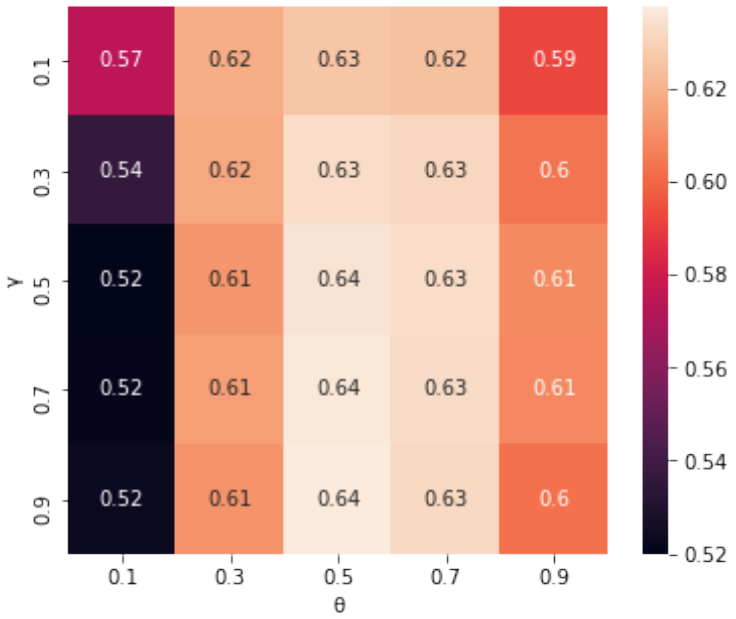


Figure 4.6 Parameter analysis on θ and γ. Values indicate Pearson correlation coefficient on Lung dataset.

4.6 Related Work

The increased resolution of transcriptomics profiling methods comes at a cost of increasing data sparsity. The profiling technology may fail to capture a number of the expressed genes of an individual cell due to low amounts of mRNA in individual cells and a low capture rate. A popular way to address this issue is to perform imputation, which aims to correct false zeros by estimating realistic values for those gene-cell pairs. A large number of methods have been developed for the task of scRNA-seq data imputation, mainly focusing on generative probability models or matrix factorization (Gong et al., 2018; Huang et al., 2018; Ronen and Akalin, 2018; van Dijk et al., 2017). Aside from these methods, deep learning models have gained immense popularity over recent years. A natural deep learning architecture for the imputation task is the autoencoder, due to its prevalence in data denoising and missing data applications (Beaulieu-Jones et al., 2017; Boquet et al., 2019; Gondara and Wang, 2017, 2018; Pereira et al., 2020).

To leverage the spatial information of spatial transcriptomic data, the latest method, Sprod (Wang et al., 2022) constructs a graph by connecting cells with similar transcriptomic profiles and prunes it with physical distance. Then a denoised matrix is learned by minimizing the reconstruction error

and a graph Laplacian smoothing term. However, this graph only connects cells that are spatially close to each other. As a result, it is limited in a localized spatial context and difficult to identify cells with long-range correlations. Different from the aforementioned methods, we propose to utilize transformers for spatial transcriptomic imputation to globally model the cell-cell interactions while exploiting the spatial information.

4.7 Chapter Conclusion

In this chapter, we comprehensively investigate two key questions related to implementing transformer models for spatial transcriptomic imputation at the cellular level. By answering these two questions, we proposed a transformer-based general imputation framework *SpaFormer* for spatial transcriptomics data. In addition, we propose a new bi-level masking technique, which can be incorporated into general autoencoder frameworks. Our method outperforms all existing methods for recovering missing values and clustering structures. This result strongly suggests the advantage of transformers in spatial omics data analysis.

CHAPTER 5

BUILDING SINGLE-CELL FOUNDATION MODEL BEYOND SINGLE CELLS

Despite that the proposed methods in chapter 3 and 4 are effective in handling multimodal data, the knowledge learned by models is not transferable across tasks and datasets. To address this kind of issue, there is an emerging effort (Yang et al., 2022; Gong et al., 2023; Shen et al., 2023; Cui et al., 2023; Theodoris et al., 2023) from the research community to explore the potential of a foundation model that first extracts latent knowledge from unlabeled data and subsequently generalizes this knowledge to a variety of tasks. The current state-of-the-art single-cell pre-trained models are greatly inspired by the success of large language models. They trained transformers by treating genes as tokens and cells as sentences. However, three fundamental differences between single-cell data and natural language data are overlooked: (1) scRNA-seq data are presented as bag-of-genes instead of sequences of RNAs; (2) Cell-cell relations are more intricate and important than inter-sentence relations; and (3) The quantity of single-cell data is considerably inferior to text data, and they are very noisy. In light of these characteristics, we propose a new pre-trained model *CellPLM*, which takes cells as tokens and tissues as sentences. In addition, we leverage spatially-resolved transcriptomic data in pre-training to facilitate learning cell-cell relationships and introduce a Gaussian mixture prior distribution as an additional inductive bias to overcome data limitation. *CellPLM* is the first single-cell foundation model that encodes cell-cell relations and it consistently outperforms existing foundation models in diverse downstream tasks, with 500x times higher inference speed on generating cell embeddings than existing pre-trained models.

5.1 Chapter Introduction

Next-generation sequencing technologies such as single-cell RNA sequencing (scRNA-seq) have produced vast amounts of data, sparking a surge of interest in developing large-scale pre-trained models for single-cell analysis (Yang et al., 2022; Gong et al., 2023; Shen et al., 2023; Cui et al., 2023; Theodoris et al., 2023). These models seek to capture underlying structures and patterns from unlabeled scRNA-seq data, and can be fine-tuned on specific downstream datasets to deliver accurate predictions and nuanced insights into cellular mechanisms. Particularly, these pre-trained

models have been inspired by the success of large language models, such as BERT and GPT (Devlin et al., 2019; Bubeck et al., 2023), and treat genes as words (tokens) and cells as sentences to train transformers (Vaswani et al., 2017). However, we argue that these approaches may have limitations due to the fundamental differences between single-cell data and natural language data, which have been largely overlooked in existing literature:

First, unlike sentences, the scRNA-seq data utilized by existing pre-trained models are not sequential. Before the training stage, RNA sequences have been identified as functional units, i.e., genes. Instead of original sequences, data is denoted as a cell-by-gene count matrix that measures the abundance of individual genes within each cell. This is analogous to the bag-of-words model in natural languages, where the set of genes is fixed, and there is no sequential relationship among them.

Second, the relationship between cells is remarkably more intricate and important than that of sentences, since cell-cell communications play an essential role in determining cell states and cell development (Armingol et al., 2021). Additionally, within tissues, there are numerous cells from the same or similar cell lineage, which grants them similar gene expression profiles and hence provides valuable supplementary information for denoising and identifying cell states (Cannoodt et al., 2016; Molho et al., 2024; Street et al., 2018). As a result, many recent methods (Wang et al., 2021; Shao et al., 2022; Xu et al., 2023; Wen et al., 2023) have constructed cell-cell graphs to advance representation learning for single-cell data. Such evidence demonstrates the importance of the cell-cell relationship, which is neglected by existing pre-trained models.

Third, the quantity and quality of single-cell datasets are significantly lower than those of natural language data. For comparison, the high-quality filtered English dataset extracted from Common Crawl corpora (Wenzek et al., 2020) consists of 32 billion sentences, whereas the largest collection of single-cell datasets, namely the Human Cell Atlas (Regev et al., 2017), includes less than 50 million cells. To make things worse, single-cell data often suffer from technical artifacts and dropout events (Svensson et al., 2017; Qiu, 2020), as well as significant batch effects between sequencing platforms and experiments (Tran et al., 2020; Argelaguet et al., 2021).

The aforementioned differences introduce distinct challenges that call for new pre-training strategies tailored for single-cell data. To bridge this gap, we propose a novel single-**Cell P**re-trained **L**anguage **M**odel (*CellPLM*), which addresses these challenges from the following perspective: **First**, As shown in Figure 5.1, *CellPLM* proposes a cell language model to account for cell-cell relations. The cell embeddings are initialized by aggregating gene embeddings since gene expressions are bag-of-word features. **Second**, *CellPLM* leverages a new type of data, spatially-resolved transcriptomic (SRT) data, to gain an additional reference for uncovering cell-cell interactions. Compared to scRNA-seq data, SRT data provide additional positional information for cells. Both types of data are jointly modeled by transformers. **Third**, *CellPLM* introduces inductive bias to overcome the limitation of data quantity and quality by utilizing a Gaussian mixture model as the prior distribution in the latent space. This design can lead to smoother and better cell latent representations (Grønbech et al., 2020; Xu et al., 2023; Jiang et al., 2023). To the best of our knowledge, the proposed *CellPLM* is the first pre-trained transformer framework that encodes inter-cell relations, leverages spatially-resolved transcriptomic data, and adopts a reasonable prior distribution. It is evident from our experiments that *CellPLM* consistently outperforms both pre-trained and non-pre-trained methods across five distinct downstream tasks, with **500x times higher inference speed** on generating cell embeddings compared to existing pre-trained models.

5.2 Cell Language Model Beyond Single Cells

In this section, we introduce the concept of the cell language models and detailed implementation of the proposed *CellPLM*. As illustrated in Figure 5.2, *CellPLM* consists of four modules: a gene expression embedder, an encoder, latent space, and a decoder, which we will demonstrate in Section 5.2.2. At a higher level, there are two stages in our framework: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data with a masked language modeling objective. For fine-tuning, the model is first initialized with the pre-trained parameters, and then all of the parameters are fine-tuned using data and labels (if available) from the downstream datasets. We demonstrate the pre-training and fine-tuning framework in Section 5.2.3 and 5.2.3.2, respectively.

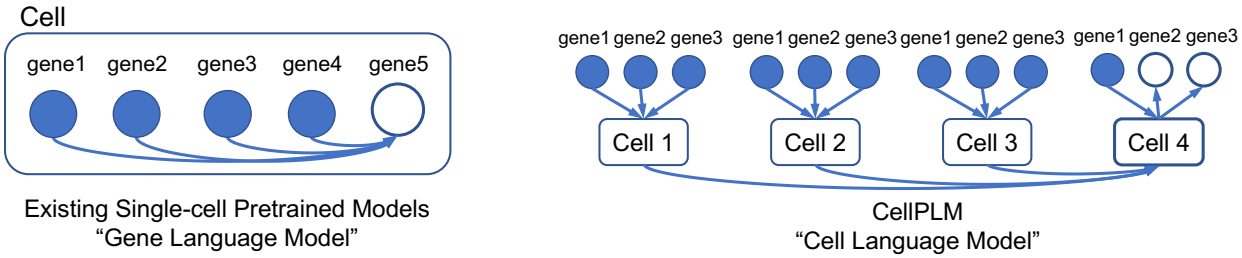


Figure 5.1 An illustration of the difference in the language models between existing single-cell pre-trained models and *CellPLM*. Existing pre-trained models only consider conditional probability between gene expressions within the same cell, while in *CellPLM*, gene expression distribution is also conditioned on other cells. See details in Section 5.2.

5.2.1 Cell Language Model

Due to the recent achievements of large language models (Bubeck et al., 2023), several studies have drawn inspiration from natural language processing in an attempt to establish a foundation model for single-cell analysis. These studies consider genes as tokens and train transformers on them, aiming to model the conditional probability between gene expressions. Concretely, previous pre-trained models are trained on scRNA-seq data, which are stored in the format of a cell-by-gene matrix $\mathbf{X} \in \mathcal{R}^{N \times k}$, where N is the number of cells, and k is the number of distinct gene types. The value of $\mathbf{X}_{i,j}$ denotes the count of gene j observed in cell i, also known as gene expression. The pre-training goal of these models is to estimate a conditional probability distribution, which can be formulated as:

$$p\left(\mathbf{X}_{i,j} | \{\mathbf{X}_{i,o}\}_{o \in O(i)}\right), j \in \mathcal{U}(i), \tag{5.1}$$

where i refers to the i-th cell and $O(i)$ is the set of observed genes in cell i whose expressions are known; $\mathcal{U}(i)$ denotes the set of unobserved genes in cell i whose expression will be predicted by the model, typically referring as masked genes. If we consider genes as words, this objective is analogous to the language model in computational linguistics (Bengio et al., 2000), and thus can be named a "gene language model". In this way, the model is trained to capture the intrinsic relations between genes, which can provide prior knowledge for downstream analysis.

However, in Eq. (5.1), the distribution of unobserved gene expressions only depends on genes within the same cell, while disregarding the information of other cells within the same tissue, which

does not align with the inherent nature of biology. Therefore, in *CellPLM*, we provide a different perspective to model scRNA-seq data by treating cells as tokens:

$$p\left(\mathbf{X}_{i,j}|\{\mathbf{X}_{u,v}\}_{(u,v)\in\mathcal{M}^{\mathrm{C}}}\right),(i,j)\in\mathcal{M}, \tag{5.2}$$

where we denote $\mathcal{M}$ as the set of masked gene expressions in $\mathbf{X}$, and $\mathcal{M}^C$ is the complement, i.e., the set of unmasked expressions. The distribution of a masked entry $\mathbf{X}_{i,j}$ depends on both the observed genes in cell i and genes from other cells that are not masked. We hereby name it as "cell language model", which models the distribution of cellular features beyond single cells. By estimating the conditional probability distribution in Eq. (5.2), *CellPLM* is trained to capture the intricate relationships that exist between not only genes but also cells.

From a biology perspective, there are particularly two types of inter-cell relations that can be beneficial to *CellPLM*. First, within tissues, there are numerous cells from the same or similar cell lineage, which mutually provide valuable supplementary information for denoising and identifying cell states (Cannoodt et al., 2016; Molho et al., 2024; Street et al., 2018). The other type of relations, cell-cell communications, plays an essential role in determining cell development and cell states (Armingol et al., 2021). Existing analysis methods (Hou et al., 2020; Jin et al., 2021; Raredon et al., 2019) have already explored the cell-cell communications on the cell type or cluster levels, while *CellPLM* aims to capture the intricate "language" of cell-cell communications between single cells. Overall, *CellPLM* presents a novel cell language model that aligns well with biological principles and holds great potential to enhance downstream tasks by extracting valuable cellular knowledge from unlabeled single-cell data.

5.2.2 Model Architecture

5.2.2.1 Gene Expression Embedder

The first module in *CellPLM* model is a gene expression embedder, which projects input gene expressions into a low-dimensional cellular feature space. In light of the nature that scRNA-seq is profiled as bag-of-genes features, *CellPLM* learns an embedding vector for each type of gene and then aggregates these gene embeddings according to their expression levels in each cell. Formally

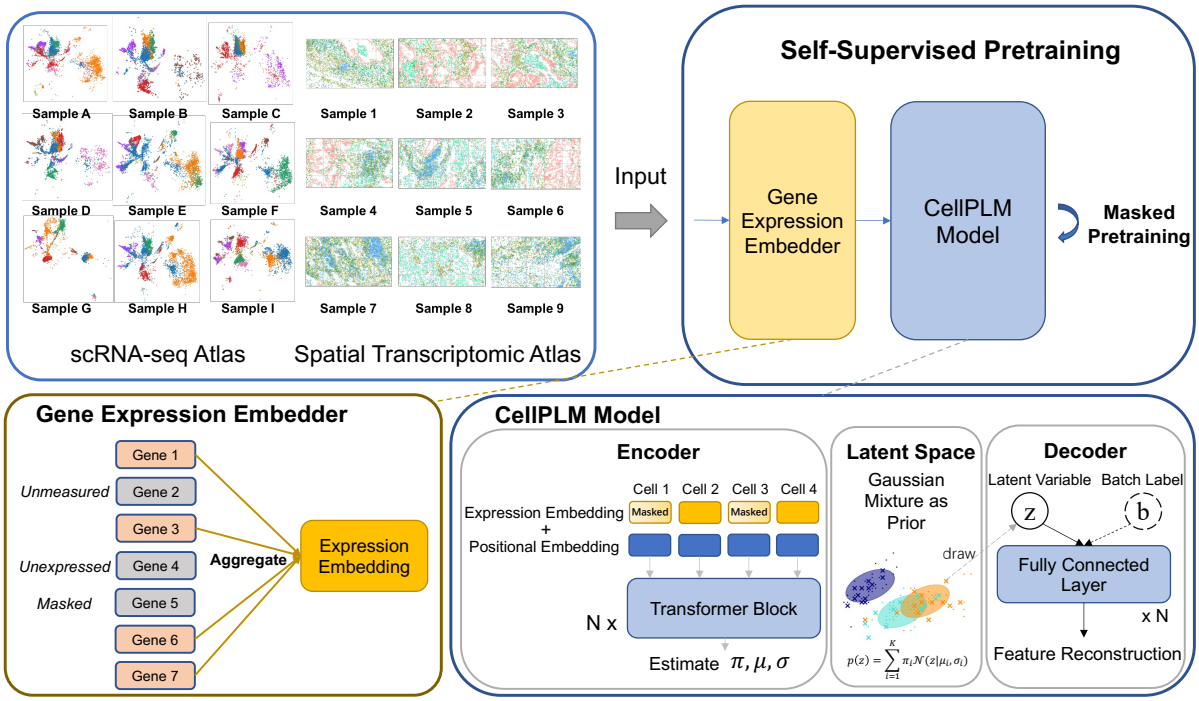


Figure 5.2 An illustration of the pre-training framework of *CellPLM*. *CellPLM* is pre-trained with cell-level masked language modeling task. The model consists of four modules: a gene expression embedder, a transformer encoder, a Gaussian mixture latent space, and a batch-aware decoder.

speaking, for gene $j \in \{1, ..., k\}$, a randomly-initialized learnable embedding vector $\mathbf{h}_j \in \mathcal{R}^d$ is assigned, where d is the hidden dimension of the encoder layers. The gene expression embedding matrix $\mathbf{E} \in \mathcal{R}^{N \times d}$ is then generated by aggregating gene embeddings according to their expressions:

$$\mathbf{E}_i = \sum_{j=1}^{k} \mathbf{X}_{i,j} \mathbf{h}_j, \tag{5.3}$$

where $\mathbf{E}_i$ is the i-th row vector of $\mathbf{E}$, corresponding to the gene expression embedding for cell i. Note that the gene expression matrix $\mathbf{X}$ is a sparse matrix since the zero-rate of scRNA-seq can be up to 90% (Jiang et al., 2022). In addition, unmeasured genes (per sequencing platforms) also lead to zero entries in $\mathbf{X}$. Therefore, when implementing Eq. (5.3), *CellPLM* leverages sparse operations, which significantly improves memory and computational efficiency. In addition, following the convention (Stuart et al., 2019), we preprocessed $\mathbf{X}$ with library size normalization and log1p transformation before inputting the model.

5.2.2.2 Transformer Encoder

The proposed *CellPLM* follows an encoder-decoder structure, where the encoder is based on transformers (Vaswani et al., 2017). The transformer model was originally developed for processing textual data. It leverages multi-head self-attention mechanisms to capture relationships between input tokens and incorporates positional encoding to represent the token positions. In *CellPLM*, by considering cells as tokens, we can readily apply the transformer model to capture intercellular relationships. When applying the transformer, we consider the embedding at l-th layer $\mathbf{H}^{(l)} \in \mathcal{R}^{N\times d}$ as a set of N tokens, where N is the total number of cells in a tissue sample, and d is the hidden dimension. By stacking L transformer layers, *CellPLM* gradually encodes cellular and inter-cellular information into cell embeddings, formulated as:

$$\mathbf{H}^{(l)} = \text{TransformerLayer}^{(l)}(\mathbf{H}^{(l-1)}). \tag{5.4}$$

In practice, N can scale up to ten thousand, which is out of the capacity of an ordinary transformer. Therefore, we adopt an efficient variant of transformers with linear complexity (i.e., Flowformer (Wu et al., 2022)) for the implementation of transformer layers.

To further inform inter-cellular relations, we incorporate spatial positional information of individual cells from a novel type of data, spatially-resolved transcriptomic (SRT) data. Specifically, SRT data consists of two parts. One is a gene expression matrix $\mathbf{X} \in \mathcal{R}^{N\times k}$ same as scRNA-seq data, and the other part is a 2D coordinate matrix $\mathbf{C} \in \mathcal{R}^{N\times 2}$. The coordinates denote the center position of each cell within a field-of-view (FOV) where the cells are located (an illustration can be found in Appendix C.1). This feature helps locate the microenvironment surrounding each cell, providing an additional reference for identifying cell lineage and cell communications, which were introduced in Section 5.2.1. To encode this extra positional information, we leverage the idea of positional encodings (PE) in transformers.

5.2.2.3 Gaussian Mixture Latent Space

One of the highlights of *CellPLM* is the design of probabilistic latent space. Prior studies have employed variational autoencoders for single-cell analysis, which typically assumes an isotropic Gaussian distribution as the prior distribution of the latent space (Lopez et al., 2018; Xu et al., 2021).

While this approach can effectively remove batch effects, it may also result in a loss of information regarding the underlying biological structure of cell groups. To address this limitation, *CellPLM* incorporates the concept of Gaussian mixture variational encoder (Dilokthanakul et al., 2016; Yang et al., 2019; Xu et al., 2023), which utilizes a mixture of Gaussians to capture the information of distinct functional groups of cells. Formally, for $i \in \{1, \ldots, N\}$, the generative model of cell i can be formulated as:

$$\begin{aligned} p(\mathbf{y}_i; \boldsymbol{\pi}) &= \text{Multinomial}(\boldsymbol{\pi}), \\ p\left(\mathbf{z}_i \mid \mathbf{y}_i\right) &= \prod_{i=1}^{L} \mathcal{N}\left(\boldsymbol{\mu}_{y_{i,l}}, \text{diag}\left(\boldsymbol{\sigma}^2_{y_{i,l}}\right)\right), \\ p_{\theta_{dec}}\left(\mathbf{x}_i \mid \mathbf{z}_i\right) &= \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{z}_i}, \sigma^2 \mathbf{I}\right), \end{aligned} \tag{5.5}$$

where $\mathbf{y}_i \in \mathcal{R}^L$ represents the one-hot latent cluster variable and $\boldsymbol{\pi}$ is its prior; $y_{i,l}$ denotes the l-th entry of $\mathbf{y}_i$; $\boldsymbol{\mu}_{y_l} \in \mathcal{R}^{d_z}$ and $\boldsymbol{\sigma}^2_{y_l} \in \mathcal{R}^{d_z \times d_z}$ denote the mean and variance of the l-th Gaussian component, respectively; and $\boldsymbol{\mu}_{z_i} \in \mathcal{R}^k$ and $\sigma^2 \mathbf{I} \in \mathcal{R}^{k \times k}$ denote the posterior mean and variance of expression $\mathbf{x}_i$, respectively. In this work, we assume that σ^2 is a constant and the posterior mean is parameterized by $\boldsymbol{\mu}_{z_i} = f_{dec}(\mathbf{z}_i; \theta_{dec})$.

To estimate the posterior of $\mathbf{z}_i$ and $\mathbf{y}_i$, we parameterize the inference process with neural networks, which is detailed in Appendix C.3. On top of that, a log-evidence lower bound (ELBO) can be derived from this generative model for the optimization purpose (Dilokthanakul et al., 2016). However, as mentioned in Section 5.2.1, our pre-training framework incorporates a cell language model, where parts of the input gene expression matrix $\mathbf{X}$ are masked. This will result in a modified objective. To formalize the problem, recall that previously we defined the masked set as $\mathcal{M}$. On top of that, we denote $\mathbf{M} \in \mathcal{R}^{N \times k}$ as a mask indicator matrix such that

$$\mathbf{M}_{i,j} = \begin{cases} 1 & \text{if } (i,j) \notin \mathcal{M}, \\ 0 & \text{if } (i,j) \in \mathcal{M}. \end{cases}$$

Let $\tilde{\mathbf{X}} \in \mathcal{R}^{N \times k}$ be the masked gene expression matrix given by the element-wise multiplication $\tilde{\mathbf{X}} = \mathbf{M} \odot \mathbf{X}$. The objective of cell language model with Gaussian mixture prior, i.e., a denoising

variational lower bound (Im Im et al., 2017), can be formulated as:

$$\begin{aligned}\mathcal{L}_{\text{CellLM}} =& \mathbb{E}_{q(\mathbf{Z},\mathbf{Y}|\tilde{\mathbf{X}})}\mathbb{E}_{p(\tilde{\mathbf{X}}|\mathbf{X})}\left[\ln \frac{p_\theta(\mathbf{X},\mathbf{Z},\mathbf{Y})}{q_\eta(\mathbf{Z},\mathbf{Y}\mid\tilde{\mathbf{X}})}\right] \\ =& \underbrace{\mathbb{E}_{q_{\eta_{enc}}(\mathbf{Z}|\tilde{\mathbf{X}})}\mathbb{E}_{p(\tilde{\mathbf{X}}|\mathbf{X})}\left[\log p_{\theta_{dec}}(\mathbf{X}\mid\mathbf{Z})\right]}_{\mathcal{L}_{\text{recon}}} - \underbrace{\mathbb{E}_{q_{\eta_\pi}(\mathbf{Y}|\mathbf{Z})}\left[\text{KL}\left(q_{\eta_{enc}}(\mathbf{Z}\mid\tilde{\mathbf{X}})\|p(\mathbf{Z}\mid\mathbf{Y})\right)\right]}_{\mathcal{L}_{\text{cond}}} \\ & - \underbrace{\mathbb{E}_{q_{\eta_{enc}}(\mathbf{Z}|\tilde{\mathbf{X}})}\left[\text{KL}\left(q_{\eta_\pi}(\mathbf{Y}\mid\mathbf{Z})\|p(\mathbf{Y})\right)\right]}_{\mathcal{L}_{\text{Y}}}.\end{aligned} \tag{5.6}$$

Similar to previous works (Dilokthanakul et al., 2016), we refer to the three terms in Eq. (5.6) as reconstruction term $\mathcal{L}_{\text{recon}}$, conditional prior term $\mathcal{L}_{\text{cond}}$ and $\mathbf{Y}$ prior term $\mathcal{L}_{\text{Y}}$. The approximation and estimation of the denoising variational lower bound are specified in Section 5.2.3.

5.2.2.4 Batch-aware Decoder

The decoder in *CellPLM* operates by decoding each cell individually, given that the tissue context has already been encoded into the latent space by the encoder. The decoder's purpose is twofold: to reconstruct masked features and to help remove batch effects from the latent space. In order to accomplish this goal, the decoder stacks several feed-forward layers (FFLayers) atop the input of latent variables $\mathbf{z}$, and a batch embedding, denoted as $\mathbf{b} \in \mathcal{R}^{d_z}$. Specifically, for each cell, the batch embedding is loaded from a learnable lookup table as $\mathbf{b} = \text{LookUp}(b)$, where b is the label indicating the specific tissue sample (or FOV for SRT data) from which the cell has been drawn. By feeding the batch label to the decoder, a batch-effect-free latent space can be achieved, as empirically evidenced in scVI (Lopez et al., 2018). The decoder can thus be formulated as:

$$\mathbf{h}^{(0)} = \mathbf{z} + \mathbf{b}, \qquad \mathbf{h}^{(l)} = \text{FFLayer}^{(l)}(\mathbf{h}^{(l-1)}),$$

where l indicates the number of the layer, $\mathbf{h}^{(l)}$ is the hidden vector of layer $l \in (1..L-1)$, and L is the total number of fully connected layers. The dimension of the last layer is different from the previous layers because the last layer is considered as an output layer, with $\mathbf{h}^L \in \mathcal{R}^k$, where k is the size of gene sets in the gene expression matrix $\mathbf{X} \in \mathcal{R}^{N\times k}$.

5.2.3 Model Pre-training and Fine-tuning

5.2.3.1 Pre-training

The pre-training of *CellPLM* follows a cell language modeling objective, as demonstrated in Eq. (5.6). Specifically, given a batch of cell tokens as input, we first decide which cells should be masked. Instead of completely masking these cell tokens, we selectively mask a certain percentage of the gene expressions within them. This allows the model to recover underlying correlations between cells, as proposed in a recent preprint, SpaFormer (Wen et al., 2023). A significant concern in *CellPLM* is the disparity in the number of genes measured by different sequencing platforms. Notably, the gap between scRNA-seq and SRT can be substantial, ranging from 1,000 to 30,000. Taking this into consideration, ***CellPLM* only masks the expression of genes that are measured in each dataset**, implying that the reconstruction loss is calculated exclusively on these measured genes. When optimizing the denoising variational lower bound in Eq. (5.6), we apply reparameterization trick and Monte Calo sampling, as proposed in VAE (Kingma and Welling, 2014). Furthermore, under the independent Gaussian assumption, we reformulate and estimate the reconstruction term $\mathcal{L}_{\text{recon}}$ in Eq. (5.6) with a mean squared error (MSE). Therefore, the pre-training loss function of *CellPLM* can be formulated as:

$$\mathcal{L}_{\text{MSE}} = \left\| \mathbf{M} \odot \left(\mathbf{H}^{(L)} - (1 - \mathbf{M}) \odot \mathbf{X} \right) \right\|_F^2, \mathcal{L}_{\text{pretrain}} = \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{cond}} + \mathcal{L}_{\text{Y}}, \tag{5.7}$$

where $\odot$ signifies element-wise multiplication, $\mathbf{H}^{(L)} \in \mathcal{R}^{N \times k}$ is the output from the decoder, $\mathbf{X}$ and $\mathbf{M}$ are the ground-truth gene expression matrix and the mask indicator matrix respectively, as defined above. $\mathcal{L}_{\text{cond}}$ and $\mathcal{L}_{\text{Y}}$ are derived from Eq. (5.6).

5.2.3.2 Task-specific Fine-tuning

When fine-tuning *CellPLM*, the model is first initialized with the pre-trained parameters. In downstream tasks that require gene expressions as output, the pre-trained decoder can be directly leveraged, and the batch embedding is set to the mixture of all pre-training batches. Otherwise, the decoder will be replaced with a task-specific head. The entire model is then fine-tuned with task-specific loss functions, which helps align the general knowledge of the model to the specific

downstream task. For example, in the spatial transcriptomic imputation task, the entire pre-trained model can do zero-shot inference. It can also be fine-tuned on a query SRT dataset and a reference scRNA-seq dataset, where two datasets are sampled from the same type of tissue. In this case, the loss function remains the same as Eq.(5.7). After fine-tuning on these datasets, *CellPLM* fits the data distribution of the target tissue and can readily perform imputation. The design and implementation of heads and loss functions for other downstream tasks are elucidated in Appendix C.5.

5.3 Experiment

CellPLM is first pre-trained on more than 9 Million scRNA-seq cells and 2 Million SRT cells, with the masked language modeling objective, demonstrated in Section 5.2.3. The model consists of over 80 million parameters and the pre-training was finished in less than 24 hours on a GPU server with 8 Nvidia Tesla v100 16GB cards. The hyperparameters, datasets, and reproducibility information for pre-trained models are detailed in Appendix C.4.

In the following sections, we evaluate the performance of *CellPLM* on various downstream tasks, including zero-shot clustering, scRNA-seq denoising, spatial transcriptomic imputation, cell type annotation, and perturbation prediction. With the selected tasks, we aim to answer the following research questions:

RQ1: Is *CellPLM* capable of transferring pre-train knowledge to a brand new dataset?

RQ2: Does *CellPLM* provide better cell representations than other pre-trained and non-pre-trained models?

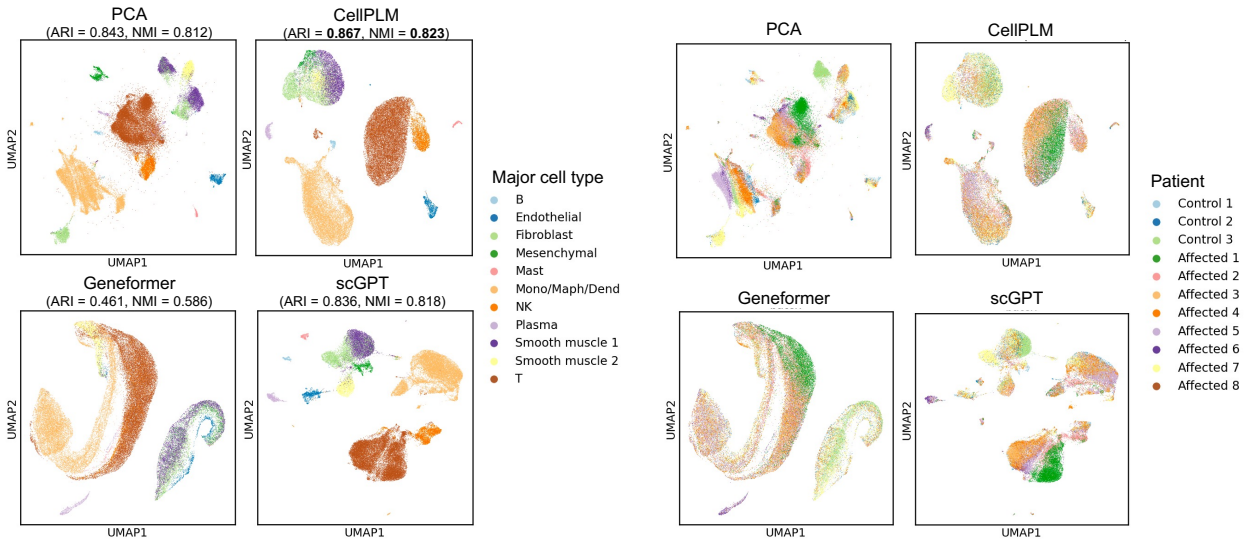
RQ3: Does *CellPLM* succeed in jointly modeling scRNA-seq and SRT data?

5.3.1 Preliminary study: zero-shot clustering

Table 5.1 Inference time(s) for querying 48,082 cells on a Tesla V100 GPU. Due to GPU memory capacity, the batch size of Geneformer and scGPT is set to 8, while the batch size of *CellPLM* is 48,082.

Geneformer	scGPT	*CellPLM*
1183.20	1540.65	**2.54**

To evaluate the transferability of the pre-trained model, we extract cell embeddings from the



(a) Clustering Performance

(b) Batch Correction

Figure 5.3 *CellPLM* readily removes patient batch effect and provides accurate cell clustering results without fine-tuning.

pre-trained encoder on a public dataset from Li, etc.(Li et al., 2020), which is not included in pre-train data. In addition to *CellPLM*, we include three baselines, i.e., PCA, Geneformer (Theodoris et al., 2023) and scGPT(Cui et al., 2023). PCA refers to the first 512 PCs of log-normalized expressions from 4500 highly variable genes (the number of PCs equals the embedding size of scGPT and *CellPLM*). This is a common embedding method on scRNA-seq data. Geneformer and scGPT are two recently published pre-trained models that are capable of generating cell embeddings. Figure 5.3a illustrates how well the embeddings are aligned with curated cell type labels, and Figure 5.3b demonstrates models' ability to remove technical artifacts and mix biological signals from different experiments. Notably, the clustering result of *CellPLM*'s embedding achieves the highest ARI and NMI with respect to the ground-truth cell type labels. From the visualization, it is also clear that *CellPLM* possesses smoother latent space than others, which is attributed to our Gaussian mixture prior distribution for pre-training. We also notice that *CellPLM* is over 500 times faster than other pre-trained models that conduct self-attention among gene tokens, as shown in Table 5.1. Overall, this preliminary study addresses **RQ1** and **RQ2**, and indicates that *CellPLM* can readily transfer pre-trained knowledge to new datasets in removing batch effect and generating high-quality cell embeddings, at extraordinarily high inference speed.

Table 5.2 (*Task 1*) The scRNA-seq denoising performance on the PBMC 5K and Jurkat datasets.

Model	PBMC 5K		Jurkat	
	RMSE (↓)	MAE (↓)	RMSE (↓)	MAE (↓)
DeepImpute	1.168 ± 0.018	1.051 ± 0.025	0.786 ± 0.006	0.557 ± 0.003
scGNN 2.0	1.376 ± 0.015	1.237 ± 0.019	1.001 ± 0.016	0.917 ± 0.021
GraphSCI	1.068 ± 0.007	0.924 ± 0.009	0.659 ± 0.030	0.481 ± 0.024
SAVER	0.884 ± 0.001	0.748 ± 0.001	0.569 ± 0.001	0.472 ± 0.001
DCA	0.775 ± 0.002	0.621 ± 0.002	0.423 ± 0.001	0.351 ± 0.001
scVI	0.777 ± 0.005	0.623 ± 0.004	0.416 ± 0.001	0.344 ± 0.002
MAGIC	0.793 ± 0.001	0.639 ± 0.001	0.424 ± 0.001	0.351 ± 0.002
scImpute	1.170 ± 0.003	1.002 ± 0.001	0.624 ± 0.002	0.529 ± 0.001
scGPT (fine-tuned)	0.901 ± 0.001	0.565 ± 0.001	0.711 ± 0.001	0.498 ± 0.001
CellPLM (zero-shot)	0.854 ± 0.001	0.692 ± 0.000	0.517 ± 0.001	0.426 ± 0.000
CellPLM (from scratch)	0.761 ± 0.009	0.571 ± 0.011	0.395 ± 0.003	0.320 ± 0.003
CellPLM (fine-tuned)	**0.725 ± 0.001**	**0.551 ± 0.001**	**0.391 ± 0.001**	**0.320 ± 0.001**

5.3.2 Task 1: scRNA-seq Denoising

Given that single-cell RNA-Seq protocols capture only a subset of the mRNA molecules within individual cells, the resulting measurements exhibit substantial technical noise (Grün et al., 2014). Therefore, we consider denoising power as the most desired and essential property for a single-cell foundation model. The goal of the denoising task is to estimate the true expression level of each gene in each cell from a noisy observation. To assess the denoising efficacy of *CellPLM*, we conduct an evaluation on two single-cell RNA-Seq datasets, i.e., PBMC 5K and Jurkat from 10x Genomics[1]. These two datasets were excluded from pre-training. Following the setting of scGNN (Wang et al., 2021) and scGNN2.0 (Gu et al., 2022), we apply a random flipping process to a subset of non-zero entries, transforming them into zeros in order to simulate the effects of dropout. We compare *CellPLM* against a broad range of contemporary approaches, including DeepImpute (Arisdakessian et al., 2019), scGNN2.0 (Gu et al., 2022), SAVER (Huang et al., 2018), DCA (Eraslan et al., 2019), scVI (Lopez et al., 2018), MAGIC (Van Dijk et al., 2018), scImpute (Li and Li, 2018) and scGPT (Cui et al., 2023). We evaluate scRNA-seq denoising performance based on two popular regression metrics, i.e., Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), to measure the degree of similarity between predicted gene expression and the actual ones. More details pertaining to these methods, the fine-tuning of *CellPLM*, and the evaluation metrics under

[1] 10x Genomics datasets are available at https://support.10xgenomics.com/single-cellgene-expression/datasets.

the task of scRNA-seq denoising can be found in Appendix C.5.1.

It is evident that the fine-tuned *CellPLM* consistently exhibits superior performance compared to all baseline models on both datasets. Note that even under the **zero-shot setting**, *CellPLM* shows satisfactory results that surpass the majority of baselines (6 out of 8) on each dataset. These observations answer the question of **RQ1** and **RQ2**. As a powerful denoising model, *CellPLM* can serve as a foundation for other downstream analyses.

5.3.3 Task 2: Spatial Transcriptomic Imputation

Spatially resolved transcriptomics has revolutionized single-cell analysis by incorporating physical locations along with gene expression, leading to exciting breakthroughs. However, as a tradeoff for the highly detailed spatial resolution, spatial transcriptomic data at the cellular level typically cover less than $1,000$ genes, which poses challenges in data analysis. To assess the potential benefits of the pre-trained model in the given task, we evaluate *CellPLM* on two spatial transcriptomic datasets at single-cell resolution, i.e., Lung2 and Liver2 from Merscope datasets dataset[2] (the whole study is not included in our pre-train data). Following the setting of baselines including SpaGE (Abdelaal et al., 2020), stPlus (Shengquan et al., 2021), gimVI (Lopez et al., 2019) and Tangram (Biancalani et al., 2021), we impute the unmeasured genes of the SRT dataset utilizing a scRNA-seq dataset as reference. We identify the testing gene set in SRT data by stratified sampling according to gene sparsity (Avşar and Pir, 2023) and hold out those genes in the fine-tuning stage. To evaluate the accuracy of spatial transcriptomic imputation, we employ the Pearson correlation coefficient (Corr) and cosine similarity (Cosine) to measure the degree of similarity between the predicted spatial gene expressions and the corresponding ground-truth expression values.

Remarkably, the fine-tuned *CellPLM* takes the lead on both datasets, effectively addressing the research question **RQ1** and **RQ3**. However, when training from scratch on these datasets, *CellPLM* hardly converges. This indicates the pre-training information is necessary for *CellPLM* to impute the SRT data. In addition, we conduct a preliminary study by visualizing the attention between cells in space, which might hint at cell-cell communication. For the visualization and additional

[2]Merscope ffpe human immuno-oncology datasets are available at https://info.vizgen.com/ffpe-showcase.

Table 5.3 (*Task 2*) The results of spatial tanscriptomic imputation on the Lung2 and Liver2 datasets.

Model	Lung2		Liver2	
	Corr (↑)	Cosine (↑)	Corr (↑)	Cosine (↑)
SpaGE	0.227 ± 0.011	0.352 ± 0.015	0.253 ± 0.014	0.376 ± 0.005
stPlus	0.177 ± 0.021	0.360 ± 0.014	0.224 ± 0.010	0.399 ± 0.012
gimVI	0.130 ± 0.010	0.325 ± 0.010	0.163 ± 0.019	0.338 ± 0.010
Tangram	0.123 ± 0.005	0.285 ± 0.008	0.168 ± 0.024	0.309 ± 0.008
CellPLM (zero-shot)	0.119 ± 0.024	0.327 ± 0.011	0.141 ± 0.013	0.322 ± 0.145
CellPLM (from scratch)	0.058 ± 0.020	0.370 ± 0.013	0.024 ± 0.039	0.352 ± 0.011
CellPLM (fine-tuned)	**0.318 ± 0.015**	**0.481 ± 0.011**	**0.328 ± 0.011**	**0.481 ± 0.010**

Table 5.4 (*Task 3*) The results of cell type annotation on MS and hPancreas dataset. "*" indicates results directly taken from scGPT (Cui et al., 2023).

	MS		hPancreas	
	F1 (↑)	Precision (↑)	F1 (↑)	Precision (↑)
CellTypist	0.667 ± 0.002	0.693 ± 0.001	0.708 ± 0.023	0.736 ± 0.025,
ACTINN	0.628 ± 0.012	0.634 ± 0.009	0.705 ± 0.005	0.709 ± 0.006
SingleCellNet	0.637 ± 0.001	0.700 ± 0.001	0.739 ± 0.006	**0.761 ± 0.004**
TOSICA*	0.578	0.664	0.656	0.661
scBERT* (fine-tuned)	0.599	0.604	0.685	0.699
scGPT* (fine-tuned)	0.703	0.729	0.718	0.735
CellPLM (from scratch)	0.709 ± 0.007	0.732 ± 0.015	0.689 ± 0.034	0.682 ± 0.037
CellPLM (fine-tuned)	**0.766 ± 0.007**	**0.803 ± 0.008**	**0.749 ± 0.010**	0.753 ± 0.010

information regarding baselines, the fine-tuning of the *CellPLM*, and the evaluation metrics under this task, please refer to Appendix C.5.2.

5.3.4 Task 3: Cell Type Annotation

Cell type annotation is another important task in single-cell analysis as it enables the identification and characterization of distinct cell populations within a tissue or organism. The objective of this task is to classify the type of cells from query datasets according to the annotations in reference datasets. Here we follow the suggestion of scGPT (Cui et al., 2023) to include hPancreas (Chen et al., 2023) and Multiple Sclerosis (MS) (Schirmer et al., 2019) datasets. More details about the datasets, baseline methods, the fine-tuning of *CellPLM* can be found in Appendix C.5.4.

The empirical results presented in Table 5.4 indicate that *CellPLM* learns a well-represented and generalizable cellular embedding, achieving considerably large improvement on the cell type annotation task. This again confirms our positive answer to the research questions **RQ1** and **RQ2**.

In addition to the aforementioned cell-level tasks, we also explore the potential of *CellPLM* in gene-level tasks. We conduct an experiment on **perturbation prediction** following the setting of GEARS (Roohani et al., 2022). The results show that *CellPLM* can also benefit gene perturbation prediction, and successfully outperform previous SOTA. Due to space limitations, we leave the detailed results in Appendix C.5.3.

5.4 Related Work

Deep learning methods for single-cell data have garnered significant research interest in recent years (Molho et al., 2024). However, due to the distinct model architectures, the knowledge learned by models is not transferable across tasks. To address this issue, there is an emerging effort (Yang et al., 2022; Gong et al., 2023; Shen et al., 2023; Cui et al., 2023; Theodoris et al., 2023) from the research community to explore the potential of a foundation model that first extracts latent knowledge from unlabeled scRNA-seq data and subsequently generalizes this knowledge to a variety of tasks.

The first such pre-trained model for single-cell data, scBERT (Yang et al., 2022), takes genes as tokens and leverages an efficient transformer (Choromanski et al., 2020) to encode over 16,000 gene tokens for each cell. By randomly masking a fraction of non-zero gene expression values and predicting them based on the remaining data, scBERT effectively learns intricate relationships between genes, leading to improved cellular representation. Later, xTrimoGene (Gong et al., 2023) made two key enhancements to scBERT: pruning zero-expressed genes and improving expression binning strategies by an auto-discretization strategy. These modifications notably enhance scalability and feature resolutions. Another latest preprint, scGPT (Cui et al., 2023), introduces a variant of masked language modeling that mimics the auto-regressive generation in natural language processing, where the masked genes are iteratively predicted according to model's confidence. Unlike the aforementioned models, Geneformer (Theodoris et al., 2023) and tGPT (Shen et al., 2023) completely abandon precise expression levels of genes. Instead, they model the rank of gene expressions and construct sequences of genes according to their relative expression levels within each cell.

The aforementioned models all regard genes as tokens and focus solely on modeling gene relationships within individual cells, neglecting the intercellular information in an organism. In contrast, *CellPLM* overcomes this limitation by introducing a cell language model that extends beyond single cells. Furthermore, by leveraging the spatial information of cells acquired from SRT data, along with a prior Gaussian mixture distribution, the model achieves unparalleled performance on a range of downstream tasks.

5.5 Chapter Conclusion

In this chapter, we propose Cell Language Model, a novel paradigm of single-cell pre-trained model, which aligns well with the fundamental characteristics of single-cell data. This has led to *CellPLM*, the first pre-trained transformer framework that encodes inter-cell relations, leverages spatially-resolved transcriptomic data and adopts a reasonable prior distribution. Our experiments on various downstream tasks demonstrate the power of *CellPLM*, which has great potential to facilitate future research in single-cell biology.

CHAPTER 6

CONCLUSION

In this chapter, we summarize the research results of this dissertation, discuss their broader impact and highlight promising research direction.

6.1 Summary

In this dissertation, we have explored various deep learning approaches for advancing single-cell multi-omics and spatial omics analysis. By leveraging graph neural networks (GNNs) and transformers, we introduced several innovative models tailored for the unique challenges presented by single-cell data. These models not only demonstrate significant improvements over existing methodologies but also lay the groundwork for future developments in the field.

One of the central contributions is the development of *CellPLM*, a novel pre-trained model that effectively captures cell-cell relationships and integrates spatial transcriptomic data, outperforming existing models in various tasks. *CellPLM* is built upon our findings from the first chapters, and effectively transfers knowledge from cross-domain unlabeled data, to various downstream tasks and datasets of interest. It can serve as a universal backbone for single-cell research, which exactly fits the concept of a foundation model. Therefore, the implications of this dissertation extend beyond the specific models and tasks discussed. The methodologies proposed in this work highlight the importance of considering complex inter-cellular interactions and multi-modal data integration in single-cell foundation model development. As single-cell technologies continue to advance, the need for more sophisticated foundation models that can handle the intricacies of these data types will only grow.

6.2 Future Work

In the future, we plan to further unleash the power of foundation models, i.e., *CellPLM*, in single-cell analysis. Currently, *CellPLM* only considers two modalities: scRNA-seq and spatial transcriptomics. While these modalities have already demonstrated significant improvements in various tasks, there is substantial room for growth and refinement.

One potential direction for future work is the extension of the *CellPLM* framework to incorporate

multi-omics data, just like what we have explored in *scMoGNN* and *scMoFormer*. The integration of additional omics layers, such as epigenomics, proteomics, and metabolomics, would enable a more comprehensive understanding of cellular processes. By incorporating these diverse data types, *CellPLM* could capture a more holistic view of the molecular landscape, leading to improved accuracy in cell classification, state prediction, and trajectory inference.

Another promising area for future exploration is the enhancement of the embedder module within *CellPLM* to better handle cross-modality integration. The introduction of advanced techniques, such as multi-view learning or contrastive learning, could improve the model's ability to learn meaningful representations across different omics modalities. Furthermore, the development of novel attention mechanisms tailored specifically for multi-modal data could allow the model to more effectively prioritize and fuse information from each modality, thus enhancing its overall performance.

Additionally, there is a growing interest in the application of foundation models like *CellPLM* to the study of rare and heterogeneous cell populations. As single-cell technologies continue to advance, the ability to analyze these rare cell types with high precision becomes increasingly important. Future work could involve the adaptation of *CellPLM* to focus on these challenging cell populations, possibly through the incorporation of few-shot learning techniques or the development of specialized pre-training strategies designed to handle imbalanced datasets.

In conclusion, the potential future directions for *CellPLM* are vast and varied. By extending the framework to incorporate more diverse data types, refining the model's architecture, and exploring new application domains, we can continue to push the boundaries of single-cell analysis and unlock new biological insights. As single-cell technologies continue to evolve, the development of sophisticated foundation models like *CellPLM* will be crucial in addressing the increasing complexity and diversity of the data being generated.

BIBLIOGRAPHY

Abdelaal, T., Mourragui, S., Mahfouz, A., and Reinders, M. J. (2020). Spage: spatial gene enhancement using scrna-seq. Nucleic Acids Research, 48(18):e107–e107.

Adamson, B., Norman, T. M., Jost, M., Cho, M. Y., Nuñez, J. K., Chen, Y., Villalta, J. E., Gilbert, L. A., Horlbeck, M. A., Hein, M. Y., et al. (2016). A multiplexed single-cell crispr screening platform enables systematic dissection of the unfolded protein response. Cell, 167(7):1867–1882.

Alon, U. and Yahav, E. (2021). On the bottleneck of graph neural networks and its practical implications. In International Conference on Learning Representations.

Argelaguet, R., Arnol, D., Bredikhin, D., Deloro, Y., Velten, B., Marioni, J. C., and Stegle, O. (2020). Mofa+: a statistical framework for comprehensive integration of multi-modal single-cell data. Genome Biology, 21(1):1–17.

Argelaguet, R., Cuomo, A. S., Stegle, O., and Marioni, J. C. (2021). Computational principles and challenges in single-cell data integration. Nature biotechnology, 39(10):1202–1215.

Arisdakessian, C., Poirion, O., Yunits, B., Zhu, X., and Garmire, L. X. (2019). Deepimpute: an accurate, fast, and scalable deep neural network method to impute single-cell rna-seq data. Genome Biology, 20(1):1–14.

Armingol, E., Officer, A., Harismendy, O., and Lewis, N. E. (2021). Deciphering cell–cell interactions and communication from gene expression. Nature Reviews Genetics, 22(2):71–88.

Avşar, G. and Pir, P. (2023). A comparative performancencodere evaluation of imputation methods in spatially resolved transcriptomics data. Molecular Omics.

Batool, F. and Hennig, C. (2021). Clustering with the average silhouette width. Computational Statistics & Data Analysis, 158:107190.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261.

Beaulieu-Jones, B. K., Moore, J. H., and CONSORTIUM, P. R. O.-A. A. C. T. (2017). Missing data imputation in the electronic health record using deeply learned autoencoders. In Pacific symposium on biocomputing 2017, pages 207–218. World Scientific.

Belhocine, K., DeMare, L., and Habern, O. (2021). Single-cell multiomics: Simultaneous epigenetic and transcriptional profiling: 10x genomics shares experimental planning and sample preparation tips for the chromium single cell multiome atac+ gene expression system. Genetic Engineering & Biotechnology News, 41(1):66–68.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. Advances in Neural Information Processing Systems, 13.

Biancalani, T., Scalia, G., Buffoni, L., Avasthi, R., Lu, Z., Sanger, A., Tokcan, N., Vanderburg, C. R., Segerstolpe, Å., Zhang, M., et al. (2021). Deep learning and alignment of spatially resolved single-cell transcriptomes with tangram. Nature Methods, 18(11):1352–1362.

Boquet, G., Vicario, J. L., Morell, A., and Serrano, J. (2019). Missing data in traffic estimation: A variational autoencoder imputation method. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2882–2886. IEEE.

Browaeys, R., Saelens, W., and Saeys, Y. (2020). Nichenet: modeling intercellular communication by linking ligands to target genes. Nature Methods, 17(2):159–162.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. arXiv preprint arXiv:2303.12712.

Buenrostro, J. D., Giresi, P. G., Zaba, L. C., Chang, H. Y., and Greenleaf, W. J. (2013). Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, dna-binding proteins and nucleosome position. Nature Methods, 10(12):1213–1218.

Burgess, D. J. (2019). Spatial transcriptomics coming of age. Nature Reviews Genetics, 20(6):317–317.

Cang, Z., Zhao, Y., Almet, A. A., Stabell, A., Ramos, R., Plikus, M. V., Atwood, S. X., and Nie, Q. (2023). Screening cell–cell communication in spatial transcriptomics via collective optimal transport. Nature Methods, 20(2):218–228.

Cannoodt, R., Saelens, W., and Saeys, Y. (2016). Computational methods for trajectory inference from single-cell transcriptomics. European journal of immunology, 46(11):2496–2506.

Cao, J., Cusanovich, D. A., Ramani, V., Aghamirzaie, D., Pliner, H. A., Hill, A. J., Daza, R. M., McFaline-Figueroa, J. L., Packer, J. S., Christiansen, L., et al. (2018). Joint profiling of chromatin accessibility and gene expression in thousands of single cells. Science, 361(6409):1380–1385.

Cao, Z.-J. and Gao, G. (2022). Multi-omics single-cell data integration and regulatory inference with graph-linked embedding. Nature Biotechnology, 40(10):1–9.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16, pages 213–229. Springer.

Chen, D., O'Bray, L., and Borgwardt, K. (2022). Structure-aware transformer for graph representation learning. In International Conference on Machine Learning, pages 3469–3489. PMLR.

Chen, J., Xu, H., Tao, W., Chen, Z., Zhao, Y., and Han, J.-D. J. (2023). Transformer for one stop interpretable cell type annotation. Nature Communications, 14(1):223.

Chen, S., Lake, B. B., and Zhang, K. (2019). High-throughput sequencing of the transcriptome and chromatin accessibility in the same cell. Nature Biotechnology, 37(12):1452–1457.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. (2020). Rethinking attention with performers. arXiv preprint arXiv:2009.14794.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. (2021). Rethinking attention with performers. In International Conference on Learning Representations.

Chu, X., Tian, Z., Zhang, B., Wang, X., Wei, X., Xia, H., and Shen, C. (2021). Conditional positional encodings for vision transformers. arXiv preprint arXiv:2102.10882.

Ciortan, M. and Defrance, M. (2022). Gnn-based embedding for clustering scrna-seq data. Bioinformatics, 38(4):1037–1044.

Cui, H., Wang, C., Maan, H., and Wang, B. (2023). scgpt: Towards building a foundation model for single-cell multi-omics using generative ai. bioRxiv, pages 2023–04.

Cunningham, F., Allen, J. E., Allen, J., Alvarez-Jarreta, J., Amode, M. R., Armean, I. M., Austine-Orimoloye, O., Azov, A. G., Barnes, I., Bennett, R., et al. (2022). Ensembl 2022. Nucleic Acids Research, 50(D1):D988–D995.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186.

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., and Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. arXiv preprint arXiv:1611.02648.

Ding, J., Liu, R., Wen, H., Tang, W., Li, Z., Venegas, J., Su, R., Molho, D., Jin, W., Wang, Y., et al. (2024). Dance: A deep learning library and benchmark platform for single-cell analysis. Genome Biology, 25(1):72.

Dixit, A., Parnas, O., Li, B., Chen, J., Fulco, C. P., Jerby-Arnon, L., Marjanovic, N. D., Dionne, D., Burks, T., Raychowdhury, R., et al. (2016). Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. Cell, 167(7):1853–1866.

Domínguez Conde, C., Xu, C., Jarvis, L., Rainbow, D., Wells, S., Gomes, T., Howlett, S., Suchanek,

O., Polanski, K., King, H., et al. (2022). Cross-tissue immune cell analysis reveals tissue-specific features in humans. Science, 376(6594):eabl5197.

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

Duren, Z., Chen, X., Zamanighomi, M., Zeng, W., Satpathy, A. T., Chang, H. Y., Wang, Y., and Wong, W. H. (2018). Integrative analysis of single-cell genomics data by coupled nonnegative matrix factorizations. Proceedings of the National Academy of Sciences, 115(30):7723–7728.

Dwivedi, V. P. and Bresson, X. (2020). A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. (2021). Graph neural networks with learnable structural and positional representations. arXiv preprint arXiv:2110.07875.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. (2022). Graph neural networks with learnable structural and positional representations. In International Conference on Learning Representations.

Eraslan, G., Simon, L. M., Mircea, M., Mueller, N. S., and Theis, F. J. (2019). Single-cell rna-seq denoising using a deep count autoencoder. Nature Communications, 10(1):1–14.

Gaiti, F., Chaligne, R., Gu, H., Brand, R. M., Kothen-Hill, S., Schulman, R. C., Grigorev, K., Risso, D., Kim, K.-T., Pastore, A., et al. (2019). Epigenetic evolution and lineage histories of chronic lymphocytic leukaemia. Nature, 569(7757):576–580.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, Proceedings of Machine Learning Research.

Gondara, L. and Wang, K. (2017). Multiple imputation using deep denoising autoencoders. arXiv preprint arXiv:1705.02737, 280.

Gondara, L. and Wang, K. (2018). Mida: Multiple imputation using denoising autoencoders. In Pacific-Asia conference on knowledge discovery and data mining, pages 260–272. Springer.

Gong, B., Zhou, Y., and Purdom, E. (2021). Cobolt: integrative analysis of multimodal single-cell sequencing data. Genome Biology, 22(1):1–21.

Gong, J., Hao, M., Zeng, X., Liu, C., Ma, J., Cheng, X., Wang, T., Zhang, X., and Song, L. (2023). xtrimogene: An efficient and scalable representation learner for single-cell rna-seq data. bioRxiv, pages 2023–03.

Gong, W., Kwak, I.-Y., Pota, P., Koyano-Nakagawa, N., and Garry, D. J. (2018). Drimpute: imputing dropout events in single cell rna sequencing data. BMC bioinformatics, 19(1):1–10.

Grønbech, C. H., Vording, M. F., Timshel, P. N., Sønderby, C. K., Pers, T. H., and Winther, O. (2020). scvae: variational auto-encoders for single-cell gene expression data. Bioinformatics, 36(16):4415–4422.

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

Gu, H., Cheng, H., Ma, A., Li, Y., Wang, J., Xu, D., and Ma, Q. (2022). scgnn 2.0: a graph neural network tool for imputation and clustering of single-cell rna-seq data. Bioinformatics, 38(23):5322–5325.

Hamilton, W. L., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, volume 30.

Hao, Y., Hao, S., Andersen-Nissen, E., Mauck, W. M., Zheng, S., Butler, A., Lee, M. J., Wilk, A. J., Darby, C., Zager, M., et al. (2021). Integrated analysis of multimodal single-cell data. Cell, 184(13):3573–3587.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022a). Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16000–16009.

He, S., Bhatt, R., Brown, C., Brown, E. A., Buhr, D. L., Chantranuvatana, K., Danaher, P., Dunaway, D., Garrison, R. G., Geiss, G., Gregory, M. T., Hoang, M. L., Khafizov, R., Killingbeck, E. E., Kim, D., Kim, T. K., Kim, Y., Klock, A., Korukonda, M., Kutchma, A., Lewis, Z. R., Liang, Y., Nelson, J. S., Ong, G. T., Perillo, E. P., Phan, J. C., Phan-Everson, T., Piazza, E., Rane, T., Reitz, Z., Rhodes, M., Rosenbloom, A., Ross, D., Sato, H., Wardhani, A. W., Williams-Wietzikoski, C. A., Wu, L., and Beechem, J. M. (2022b). High-plex multiomic analysis in ffpe at subcellular level by spatial molecular imaging. bioRxiv.

Hou, R., Denisenko, E., Ong, H. T., Ramilowski, J. A., and Forrest, A. R. (2020). Predicting cell-to-cell communication networks using natmi. Nature Communications, 11(1):5011.

Huang, K., Xiao, C., Glass, L. M., and Sun, J. (2021). Moltrans: molecular interaction transformer for drug–target interaction prediction. Bioinformatics, 37(6):830–836.

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

Ieremie, I., Ewing, R. M., and Niranjan, M. (2022). Transformergo: predicting protein–protein interactions by modelling the attention between sets of gene ontology terms. Bioinformatics, 38(8):2269–2277.

Im Im, D., Ahn, S., Memisevic, R., and Bengio, Y. (2017). Denoising criterion for variational auto-encoding framework. In Proceedings of the AAAI conference on artificial intelligence, volume 31.

Jiang, J., Xu, J., Liu, Y., Song, B., Guo, X., Zeng, X., and Zou, Q. (2023). Dimensionality reduction and visualization of single-cell rna-seq data with an improved deep variational autoencoder. Briefings in Bioinformatics, page bbad152.

Jiang, R., Sun, T., Song, D., and Li, J. J. (2022). Statistics or biology: the zero-inflation controversy about scrna-seq data. Genome Biology, 23(1):1–24.

Jin, S., Guerrero-Juarez, C., Zhang, L., Chang, I., Ramos, R., Kuan, C., Myung, P., Plikus, M., and Nie, Q. (2021). Inference and analysis of cell-cell communication using cellchat. nat. commun. 12, 1088.

Jin, S., Zhang, L., and Nie, Q. (2020). scai: an unsupervised approach for the integrative analysis of parallel single-cell transcriptomic and epigenomic profiles. Genome biology, 21:1–19.

Kim, N., Kim, H. K., Lee, K., Hong, Y., Cho, J. H., Choi, J. W., Lee, J.-I., Suh, Y.-L., Ku, B. M., Eum, H. H., et al. (2020). Single-cell rna sequencing demonstrates the molecular and cellular reprogramming of metastatic lung adenocarcinoma. Nature Communications, 11(1):2285.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations.

Kitaev, N., Kaiser, Ł., and Levskaya, A. (2020). Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451.

Kotliarov, Y., Sparks, R., Martins, A. J., Mulè, M. P., Lu, Y., Goswami, M., Kardava, L., Banchereau, R., Pascual, V., Biancotto, A., et al. (2020). Broad immune activation underlies shared set point signatures for vaccine responsiveness in healthy individuals and disease activity in patients with lupus. Nature Medicine, 26(4):618–629.

Kreuzer, D., Beaini, D., Hamilton, W. L., Létourneau, V., and Tossou, P. (2021). Rethinking graph

transformers with spectral attention. In Advances in Neural Information Processing Systems, volume 34, pages 21618–21629.

Lee, J., Hyeon, D. Y., and Hwang, D. (2020). Single-cell multiomics: technologies and data analysis methods. Experimental & Molecular Medicine, 52(9):1428–1442.

Lelong, S., Zhou, X., Afrasiabi, C., Qian, Z., Cano, M. A., Tsueng, G., Xin, J., Mullen, J., Yao, Y., Avila, R., et al. (2022). Biothings sdk: a toolkit for building high-performance data apis in biomedical research. Bioinformatics, 38(7):2077–2079.

Li, J., Chen, S., Pan, X., Yuan, Y., and Shen, H.-B. (2022). Cell clustering for spatial transcriptomics data with graph neural networks. Nature Computational Science, 2(6):399–408.

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

Li, Y., Ren, P., Dawson, A., Vasquez, H. G., Ageedi, W., Zhang, C., Luo, W., Chen, R., Li, Y., Kim, S., et al. (2020). Single-cell transcriptome analysis reveals dynamic cell populations and differential gene expression patterns in control and aneurysmal human aortic tissue. Circulation, 142(14):1374–1388.

Liberzon, A., Birger, C., Thorvaldsdóttir, H., Ghandi, M., Mesirov, J. P., and Tamayo, P. (2015). The molecular signatures database hallmark gene set collection. Cell systems, 1(6):417–425.

Lim, D., Robinson, J., Zhao, L., Smidt, T., Sra, S., Maron, H., and Jegelka, S. (2022). Sign and basis invariant networks for spectral graph representation learning. arXiv preprint arXiv:2202.13013.

Lin, X., Tian, T., Wei, Z., and Hakonarson, H. (2022). Clustering of single-cell multi-omics data with a multimodal deep learning method. Nature Communications, 13(1):7705.

Liu, Q., Chen, S., Jiang, R., and Wong, W. H. (2021a). Simultaneous deep generative modelling and clustering of single-cell genomic data. Nature machine intelligence, 3(6):536–544.

Liu, X., Ding, J., Jin, W., Xu, H., Ma, Y., Liu, Z., and Tang, J. (2021b). Graph neural networks with adaptive residual. Advances in Neural Information Processing Systems, 34.

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al. (2022). Swin transformer v2: Scaling up capacity and resolution. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 12009–12019.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021c). Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF international conference on computer vision, pages 10012–10022.

Lopez, R., Nazaret, A., Langevin, M., Samaran, J., Regier, J., Jordan, M. I., and Yosef, N. (2019). A

joint model of unpaired data from scrna-seq and spatial transcriptomics for imputing missing gene expression measurements. arXiv preprint arXiv:1905.02269.

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

Lotfollahi, M., Wolf, F. A., and Theis, F. J. (2019). scgen predicts single-cell perturbation responses. Nature Methods, 16(8):715–721.

Lubeck, E., Coskun, A. F., Zhiyentayev, T., Ahmad, M., and Cai, L. (2014). Single-cell in situ rna profiling by sequential hybridization. Nature Methods, 11(4):360–361.

Luecken, M. D., Burkhardt, D. B., Cannoodt, R., Lance, C., Agrawal, A., Aliee, H., Chen, A. T., Deconinck, L., Detweiler, A. M., Granados, A. A., et al. (2021). A sandbox for prediction and integration of dna, rna, and proteins in single cells. In Thirty-fifth conference on neural information processing systems datasets and benchmarks track (Round 2).

Luo, S., Li, S., Cai, T., He, D., Peng, D., Zheng, S., Ke, G., Wang, L., and Liu, T.-Y. (2021). Stable, fast and accurate: Kernelized attention with relative positional encoding. Advances in Neural Information Processing Systems, 34:22795–22807.

Ma, F. and Pellegrini, M. (2020). Actinn: automated identification of cell types in single cell rna sequencing. Bioinformatics, 36(2):533–538.

Ma, L., Wang, L., Khatib, S. A., Chang, C.-W., Heinrich, S., Dominguez, D. A., Forgues, M., Candia, J., Hernandez, M. O., Kelly, M., et al. (2021a). Single-cell atlas of tumor cell evolution in response to therapy in hepatocellular carcinoma and intrahepatic cholangiocarcinoma. Journal of Hepatology, 75(6):1397–1408.

Ma, Y., Liu, X., Zhao, T., Liu, Y., Tang, J., and Shah, N. (2021b). A unified view on graph neural networks as graph signal denoising. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pages 1202–1211.

Ma, Y. and Tang, J. (2021). Deep Learning on Graphs. Cambridge University Press.

Merritt, C. R., Ong, G. T., Church, S. E., Barker, K., Danaher, P., Geiss, G., Hoang, M., Jung, J., Liang, Y., McKay-Fleisch, J., et al. (2020). Multiplex digital spatial profiling of proteins and rna in fixed tissue. Nature Biotechnology.

Mimitou, E. P., Lareau, C. A., Chen, K. Y., Zorzetto-Fernandes, A. L., Hao, Y., Takeshima, Y., Luo, W., Huang, T.-S., Yeung, B. Z., Papalexi, E., et al. (2021). Scalable, multimodal profiling of chromatin accessibility, gene expression and protein levels in single cells. Nature Biotechnology, 39(10):1246–1258.

Minoura, K., Abe, K., Nam, H., Nishikawa, H., and Shimamura, T. (2021). A mixture-of-experts

deep generative model for integrated analysis of single-cell multiomics data. Cell reports methods, 1(5):100071.

Molho, D., Ding, J., Tang, W., Li, Z., Wen, H., Wang, Y., Venegas, J., Jin, W., Liu, R., Su, R., et al. (2024). Deep learning in single-cell analysis. ACM Transactions on Intelligent Systems and Technology, 15(3):1–62.

Norman, T. M., Horlbeck, M. A., Replogle, J. M., Ge, A. Y., Xu, A., Jost, M., Gilbert, L. A., and Weissman, J. S. (2019). Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. Science, 365(6455):786–793.

Pereira, R. C., Santos, M. S., Rodrigues, P. P., and Abreu, P. H. (2020). Reviewing autoencoders for missing data imputation: Technical trends, applications and outcomes. Journal of Artificial Intelligence Research, 69:1255–1285.

Pott, S. (2017). Simultaneous measurement of chromatin accessibility, dna methylation, and nucleosome phasing in single cells. eLife, 6:e23203.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: unbiased boosting with categorical features. Advances in Neural Information Processing Systems, 31.

Qiu, P. (2020). Embracing the dropouts in single-cell rna-seq analysis. Nature Communications, 11(1):1169.

Rampasek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. (2022). Recipe for a general, powerful, scalable graph transformer. In Advances in Neural Information Processing Systems.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. (2022). Recipe for a general, powerful, scalable graph transformer. arXiv preprint arXiv:2205.12454.

Rao, J., Zhou, X., Lu, Y., Zhao, H., and Yang, Y. (2021). Imputing single-cell rna-seq data by combining graph convolution and autoencoder neural networks. iScience, 24(5):102393.

Raredon, M. S. B., Adams, T. S., Suhail, Y., Schupp, J. C., Poli, S., Neumark, N., Leiby, K. L., Greaney, A. M., Yuan, Y., Horien, C., et al. (2019). Single-cell connectomic analysis of adult mammalian lungs. Science advances, 5(12):eaaw3851.

Regev, A., Teichmann, S. A., Lander, E. S., Amit, I., Benoist, C., Birney, E., Bodenmiller, B., Campbell, P., Carninci, P., Clatworthy, M., et al. (2017). The human cell atlas. eLife, 6:e27041.

Ronen, J. and Akalin, A. (2018). netsmooth: Network-smoothing based imputation for single cell rna-seq. F1000Research, 7.

Roohani, Y., Huang, K., and Leskovec, J. (2022). Gears: Predicting transcriptional outcomes of novel multi-gene perturbations. bioRxiv, pages 2022–07.

Rudensky, A. Y. (2011). Regulatory t cells and foxp3. Immunological reviews, 241(1):260–268.

Schirmer, L., Velmeshev, D., Holmqvist, S., Kaufmann, M., Werneburg, S., Jung, D., Vistnes, S., Stockley, J. H., Young, A., Steindel, M., et al. (2019). Neuronal vulnerability and multilineage diversity in multiple sclerosis. Nature, 573(7772):75–82.

Shao, X., Li, C., Yang, H., Lu, X., Liao, J., Qian, J., Wang, K., Cheng, J., Yang, P., Chen, H., et al. (2022). Knowledge-graph-based cell-cell communication inference for spatially resolved transcriptomic data with spatalk. Nature Communications, 13(1):4429.

Shao, X., Yang, H., Zhuang, X., Liao, J., Yang, P., Cheng, J., Lu, X., Chen, H., and Fan, X. (2021). scdeepsort: a pre-trained cell-type annotation method for single-cell transcriptomics using deep learning with a weighted graph neural network. Nucleic Acids Research, 49(21).

Shen, H., Liu, J., Hu, J., Shen, X., Zhang, C., Wu, D., Feng, M., Yang, M., Li, Y., Yang, Y., et al. (2023). Generative pretraining from large-scale transcriptomes for single-cell deciphering. iScience.

Shen, R., Olshen, A. B., and Ladanyi, M. (2009). Integrative clustering of multiple genomic data types using a joint latent variable model with application to breast and lung cancer subtype analysis. Bioinformatics.

Shengquan, C., Boheng, Z., Xiaoyang, C., Xuegong, Z., and Rui, J. (2021). stplus: a reference-based method for the accurate enhancement of spatial transcriptomics. Bioinformatics, 37(Supplement_1):i299–i307.

Song, Q. and Su, J. (2021). Dstg: deconvoluting spatial transcriptomics data through graph-based artificial intelligence. Brief Bioinform, 22(5):1–13.

Song, Q., Su, J., and Zhang, W. (2021). scgcn is a graph convolutional networks algorithm for knowledge transfer in single cell omics. Nature Communications, 12(1):1–11.

Stein-O'Brien, G. L., Arora, R., Culhane, A. C., Favorov, A. V., Garmire, L. X., Greene, C. S., Goff, L. A., Li, Y., Ngom, A., Ochs, M. F., et al. (2018). Enter the matrix: factorization uncovers knowledge from omics. Trends in Genetics, 34(10):790–805.

Stelzer, G., Rosen, N., Plaschkes, I., Zimmerman, S., Twik, M., Fishilevich, S., Stein, T. I., Nudel, R., Lieder, I., Mazor, Y., et al. (2016). The genecards suite: from gene data mining to disease genome sequence analyses. Current protocols in bioinformatics, 54(1):1–30.

Stoeckius, M., Hafemeister, C., Stephenson, W., Houck-Loomis, B., Chattopadhyay, P. K., Swerdlow, H., Satija, R., and Smibert, P. (2017). Simultaneous epitope and transcriptome measurement in

single cells. Nature Methods, 14(9):865–868.

Street, K., Risso, D., Fletcher, R. B., Das, D., Ngai, J., Yosef, N., Purdom, E., and Dudoit, S. (2018). Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. BMC genomics, 19:1–16.

Stringer, C., Wang, T., Michaelos, M., and Pachitariu, M. (2021). Cellpose: a generalist algorithm for cellular segmentation. Nature Methods, 18(1):100–106.

Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck III, W. M., Hao, Y., Stoeckius, M., Smibert, P., and Satija, R. (2019). Comprehensive integration of single-cell data. Cell, 177(7):1888–1902.

Ståhl, P. L., Salmén, F., Vickovic, S., Lundmark, A., Navarro, J. F., Magnusson, J., Giacomello, S., Asp, M., Westholm, J. O., Huss, M., Mollbrink, A., Linnarsson, S., Codeluppi, S., Åke Borg, Pontén, F., Costea, P. I., Sahlén, P., Mulder, J., Bergmann, O., Lundeberg, J., and Frisén, J. (2016). Visualization and analysis of gene expression in tissue sections by spatial transcriptomics. Science, 353(6294):78–82.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., et al. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences, 102(43):15545–15550.

Svensson, V., Natarajan, K. N., Ly, L.-H., Miragaia, R. J., Labalette, C., Macaulay, I. C., Cvejic, A., and Teichmann, S. A. (2017). Power analysis of single-cell rna-sequencing experiments. Nature Methods, 14(4):381–387.

Szklarczyk, D., Kirsch, R., Koutrouli, M., Nastou, K., Mehryary, F., Hachilif, R., Gable, A. L., Fang, T., Doncheva, N. T., Pyysalo, S., et al. (2023). The string database in 2023: protein–protein association networks and functional enrichment analyses for any sequenced genome of interest. Nucleic Acids Research, 51(D1):D638–D646.

Tan, Y. and Cahan, P. (2019). Singlecellnet: a computational tool to classify single cell rna-seq data across platforms and across species. Cell systems, 9(2):207–213.

Tang, W., Wen, H., Liu, R., Ding, J., Jin, W., Xie, Y., Liu, H., and Tang, J. (2023). Single-cell multimodal prediction via transformers. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, pages 2422–2431.

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. (2023). Transfer learning enables predictions in network biology. Nature, pages 1–9.

Tran, H. T. N., Ang, K. S., Chevrier, M., Zhang, X., Lee, N. Y. S., Goh, M., and Chen, J. (2020).

A benchmark of batch-effect correction methods for single-cell rna sequencing data. Genome Biology, 21:1–32.

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

Van Dijk, D., Sharma, R., Nainys, J., Yim, K., Kathail, P., Carr, A. J., Burdziak, C., Moon, K. R., Chaffer, C. L., Pattabiraman, D., et al. (2018). Recovering gene interactions from single-cell data using data diffusion. Cell, 174(3):716–729.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning, pages 1096–1103.

Wang, J., Ma, A., Chang, Y., Gong, J., Jiang, Y., Qi, R., Wang, C., Fu, H., Ma, Q., and Xu, D. (2021). scgnn is a novel graph neural network framework for single-cell rna-seq analyses. Nature Communications, 12(1):1–11.

Wang, Y., Song, B., Wang, S., Chen, M., Xie, Y., Xiao, G., Wang, L., and Wang, T. (2022). Sprod for de-noising spatially resolved transcriptomics data based on position and image information. Nature Methods, 19(8):950–958.

Welch, J. D., Hartemink, A. J., and Prins, J. F. (2017). Matcher: manifold alignment reveals correspondence between single cell transcriptome and epigenome dynamics. Genome Biology, 18(1):1–19.

Wen, H., Ding, J., Jin, W., Wang, Y., Xie, Y., and Tang, J. (2022a). Graph neural networks for multimodal single-cell data integration. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 4153–4163.

Wen, H., Jin, W., Ding, J., Xu, C., Xie, Y., and Tang, J. (2022b). Bi-channel masked graph autoencoders for spatially resolved single-cell transcriptomics data imputation. In NeurIPS 2022 AI for Science: Progress and Promises.

Wen, H., Tang, W., Dai, X., Ding, J., Jin, W., Xie, Y., and Tang, J. (2024). CellPLM: Pre-training of cell language model beyond single cells. In The Twelfth International Conference on Learning

Representations.

Wen, H., Tang, W., Jin, W., Ding, J., Liu, R., Shi, F., Xie, Y., and Tang, J. (2023). Single cells are spatial tokens: Transformers for spatial transcriptomic data imputation. arXiv preprint arXiv:2302.03038.

Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., and Grave, E. (2020). CCNet: Extracting high quality monolingual datasets from web crawl data. In Proceedings of the Twelfth Language Resources and Evaluation Conference, pages 4003–4012, Marseille, France. European Language Resources Association.

Wu, C., Mark, A., and Su, A. I. (2014). Mygene. info: gene annotation query as a service. bioRxiv, page 009332.

Wu, H., Wu, J., Xu, J., Wang, J., and Long, M. (2022). Flowformer: Linearizing transformers with conservation flows. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 24226–24242. PMLR.

Wu, K., Peng, H., Chen, M., Fu, J., and Chao, H. (2021a). Rethinking and improving relative position encoding for vision transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 10033–10041.

Wu, K. E., Yost, K. E., Chang, H. Y., and Zou, J. (2021b). Babel enables cross-modality translation between multiomic profiles at single-cell resolution. Proceedings of the National Academy of Sciences, 118(15):e2023070118.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 32(1):4–24.

Xu, C., Lopez, R., Mehlman, E., Regier, J., Jordan, M. I., and Yosef, N. (2021). Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. Molecular systems biology, 17(1):e9620.

Xu, H., Fu, H., Long, Y., Ang, K. S., Sethi, R., Chong, K., Li, M., Uddamvathanak, R., Lee, H. K., Ling, J., et al. (2024). Unsupervised spatially embedded deep representation of spatial transcriptomics. Genome Medicine, 16(1):12.

Xu, J., Xu, J., Meng, Y., Lu, C., Cai, L., Zeng, X., Nussinov, R., and Cheng, F. (2023). Graph embedding and gaussian mixture variational autoencoder network for end-to-end analysis of single-cell rna sequencing data. Cell Reports Methods, page 100382.

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

Yang, K. D., Belyaeva, A., Venkatachalapathy, S., Damodaran, K., Katcoff, A., Radhakrishnan, A., Shivashankar, G., and Uhler, C. (2021). Multi-domain translation between single-cell imaging and sequencing data using autoencoders. Nature Communications, 12(1):1–10.

Yang, L., Cheung, N.-M., Li, J., and Fang, J. (2019). Deep clustering by gaussian mixture variational autoencoders with graph embedding. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6440–6449.

Yao, Z., Liu, H., Xie, F., Fischer, S., Adkins, R. S., Aldridge, A. I., Ament, S. A., Bartlett, A., Behrens, M. M., Van den Berge, K., et al. (2021). A transcriptomic and epigenomic cell atlas of the mouse primary motor cortex. Nature, 598(7879):103–110.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. (2021). Do transformers really perform badly for graph representation? Advances in Neural Information Processing Systems, 34:28877–28888.

Zhu, C., Preissl, S., and Ren, B. (2020). Single-cell multimodal omics: the power of many. Nature Methods, 17(1):11–14.

Zhu, C., Yu, M., Huang, H., Juric, I., Abnousi, A., Hu, R., Lucero, J., Behrens, M. M., Hu, M., and Ren, B. (2019). An ultra high-throughput method for single-cell joint analysis of open chromatin and transcriptome. Nature Structural & Molecular Biology, 26(11):1063–1070.

Zuo, C., Dai, H., and Chen, L. (2021). Deep cross-omics cycle attention model for joint analysis of single-cell multi-omics data. Bioinformatics, 37(22):4091–4099.

APPENDIX A

GRAPH NEURAL NETWORKS FOR MULTI-OMICS REPRESENTATION LEARNING

A.1 Details of Metrics in Task 3

A.1.1 Biology Conservation Metrics

NMI cluster/label. Normalized mutual information (NMI) compares the overlap of two clusterings. We use NMI to compare the cell type labels with an automated clustering computed on the integrated dataset (based on `Louvain clustering`. NMI scores of 0 or 1 correspond to uncorrelated clustering or a perfect match, respectively. Automated Louvain clustering is performed at resolution ranges from 0.1 to 2 in steps of 0.1, and the clustering output with the highest NMI with the label set is used.

Cell type ASW. The silhouette width metric indicates the degree to which observations with identical labels are compact. The average silhouette width (ASW) (Batool and Hennig, 2021), which ranges between -1 and 1, is calculated by averaging the silhouette widths of all cells in a set. We employ ASW to determine the compactness of the resulting embedding's cell types. The ASW of the cluster is calculated using the cell identity labels and scaled to a value between 0 and 1 using the equation:

$$ASW = (ASW_C + 1)/2 \tag{A.1}$$

where C denotes the set of all cell identity labels.

Cell cycle conservation. The cell cycle conservation score serves as a proxy for the preservation of the signal associated with gene programs during data integration. It determines the amount of variance explained by cell cycle per batch prior to and following integration. The differences in variance before Var_{before} and variance after Var_{after} are aggregated into a final score between 0 and 1, using the equation:

$$CC_{conservation} = 1 - \frac{|Var_{after} - Var_{before}|}{Var_{before}} \tag{A.2}$$

where values near to 0 suggest less conservation of variance explained by the cell cycle, while 1 represents complete conservation.

Trajectory conservation. The conservation score of a trajectory is a proxy for the conservation of a continuous biological signal within a joint embedding. We compare trajectories computed after integration for relevant cell types that depict a continuous cellular differentiation process to trajectories computed per batch and modality using this metric. The conservation of the trajectory is quantified via Spearman's rank correlation coefficient, s, between the pseudotime values before and after integration. The final score is scaled to a value between 0 and 1 using the equation:

$$trajectoryconservation = (s + 1)/2 \tag{A.3}$$

where a value of 1 or 0 indicates that the cells on the trajectory are in the same order before and after integration, or in the reverse order.

A.1.2 Batch Removal Metrics

Batch ASW. The ASW is used to quantify batch mixing by taking into account the incompatibility of batch labels per cell type cluster. We consider the absolute silhouette width, on batch labels per cell, in particular. Here, zero shows that batches are thoroughly mixed, but any variation from zero indicates the presence of a batch effect. We rescale this score so that higher values imply better batch mixing and use the equation below to determine the per-cell type label, j:

$$\text{batch } ASW_j = \frac{1}{|C_j|} \sum_{i \in C_j} 1 - |s(i)| \tag{A.4}$$

where C_j is the set of cells with the cell label j and $|C_j|$ denotes the number of cells in that set. To obtain the final $batchASW$ score, the label-specific $batchASW_j$ scores are averaged:

$$\text{batch } ASW = \frac{1}{|M|} \sum_{j \in M} \text{batch } ASW_j \tag{A.5}$$

where M is the set of unique cell labels. A $batchASW$ value of 1 suggests optimal batch mixing, whereas a value of 0 indicates severely separated batches.

Graph connectivity. The graph connectivity metric determines whether cells of the same kind from various batches are embedded close to one another. This is determined by computing a k-nearest neighbor (kNN) graph using Euclidean distances on the embedding. Then, we determine

Table A.1 Dataset Statistics of modality prediction task. The number of feature dimensions, train/test samples, and batches.

	GEX-ADT	ADT-GEX	GEX-ATAC	ATAC-GEX
Source Dim	13,953	134	13,431	116,490
Target Dim	134	13,953	10,000	13,431
Train Cells	66,175	66,175	42,492	42,492
Test Cells	1,000	1,000	1,000	1,000
Train Batches	9	9	10	10
Test Batches	3	3	3	3

if all cells with the same cell identity label are connected in this kNN graph. For each cell identity label c, we generate the subset kNN graph $G = (N_c; E_c)$, which contains only cells from a given label. Using these subset kNN graphs, we compute the graph connectivity score:

$$gc = \frac{1}{|C|} \sum_{c \in C} \frac{|LCC\left(G\left(N_c; E_c\right)\right)|}{|N_c|} \tag{A.6}$$

where C represents the set of cell identity labels, $LCC()$ denotes the number of nodes in the largest connected component of the graph, and N_c is the number of nodes with cell identity c. The resulting score ranges from 0 to 1, where 1 means that all cells with the same cell identity are connected in the integrated kNN graph, while 0 indicates that no cell is connected in the network.

A.1.3 Metric Aggregation

Due to the differing nature of each metric, each metric would be assigned a weight. An overall weighted average of batch correction and bio-conservation scores will be computed via the equation:

$$S_{\text{overall},i} = 0.6 \cdot S_{bio,i} + 0.4 \cdot S_{batch,i} \tag{A.7}$$

A.2 Data Statistics

The Table A.1 and Table A.2 provide statistics about dataset used in modality prediction task and modality matching task respectively.

A.3 Reproducibility

A.3.1 Source Codes

All the source code of winning solutions can be found at OpenProblems official github. These codes have been officially verified thus reproducibility is ensured.

Table A.2 Dataset Statistics of modality matching task. The number of feature dimensions, train/test samples, and batches.

	GEX-ADT	**ADT-GEX**	**GEX-ATAC**	**ATAC-GEX**
Source Dim	13,953	134	13,431	116,490
Target Dim	134	13,953	116,490	13,431
Train Cells	66,175	66,175	42,492	42,492
Test Cells	15,066	15,066	20,009	20,009
Train Batches	10	10	10	10
Test Batches	3	3	3	3

For the new models we developed after the competitions, all source codes have been integrated into the DANCE package (Ding et al., 2024).

APPENDIX B

TRANSFORMERS FOR SINGLE-CELL SPATIAL OMICS REPRESENTATION LEARNING

B.1 Reproduciblity Details

B.1.1 Data Availability

Two datasets (Lung and Liver) we used are public available on Nanostring official website: https://nanostring.com/products/cosmx-spatial-molecular-imager/nsclc-ffpe-dataset.

B.1.2 Implementation Settings

B.1.2.1 Baselines

To evaluate the effectiveness of *SpaFormer*, we compare it with the state-of-the-art spatial and non-spatial transcriptomic imputation models: (1) scImpute (Li and Li, 2018) employs a probabilistic model to detect dropouts, and implements imputation through non-negative least squares regression. (2) SAVER (Huang et al., 2018) uses negative binomial distribution to model the data and estimates a Gamma prior through Poisson Lasso regression. The posterior mean is used to output expression with uncertainty quantification from the posterior distribution. (3) scVI (Lopez et al., 2018) models dropouts with a ZINB distribution, and estimates the distributional parameters of each gene in each cell with a VAE model. (4) DCA (Eraslan et al., 2019) is an autoencoder that predicts parameters of chosen distributions like ZINB to generate the imputed data. (5) GraphSCI (Rao et al., 2021) employs a graph autoencoder on a gene correlation graph. Meanwhile, it uses another autoencoder to reconstruct the gene expressions, taking graph autoencoder embeddings as additional input. (6) scGNN (Wang et al., 2021) first builds a cell-cell graph based on gene expression similarity and then utilizes a graph autoencoder together with a standard autoencoder to refine graph structures and cell representation. Lastly, an imputation autoencoder is trained with a graph Laplacian smoothing term added to the reconstruction loss. (7) gimVI (Lopez et al., 2019) is a deep generative model for integrating spatial transcriptomics data and scRNA-seq data which can be used to impute spatial transcriptomic data. gimVI is based on scVI (Lopez et al., 2018) and employs alternative conditional distributions to address technology-specific covariate shift more effectively. (8) Sprod (Li et al.,

2022) is the latest published imputation method for spatial transcriptomics data. It first projects gene expressions to a latent space, then connects neighboring cells in the latent space, and prunes it with physical distance. Then a denoised matrix is learned by jointly minimizing the reconstruction error and a graph Laplacian smoothing term. (9) SpaGAT is a baseline model created by ourselves. It is the same bi-level masking autoencoder framework as *SpaFormer*, based on a graph neural network encoder with spatial graphs. Specifically, we implement a graph attention network (Velickovic et al., 2018) as the encoder. Since the graph attention network is a localized version of transformers, SpaGAT can be considered an ablation study for our *SpaFormer* model.

B.1.2.2 Hyperparameter Settings

For our own2 SpaGAT and SpaFormer, we first normalize the total RNA counts of each cell, and then apply log1p transform. We heavily conducted hyperparameters searching on the Lung dataset. However, we noticed that the performance is not sensitive to most hyperparameters, except for masking rate, autoencoder type, and positional encodings as we presented in ablation studies. To reproduce our results, the recommended hyperparameters are n_layer=2, num_heads=8, num_hidden=128, latent_dim=20, learning rate=1e-3, weight_decay=5e-4. We used these default hyperparameters in the other two datasets. The source codes will been released on our github. For our own created baseline model SpaGAT, we used the same set of hyperparameters while replacing the transformer encoder with a graph attention network.

For baseline models, all the implementations are from the authors' repo/software. Optimizers/trainers are provided by original implementations. Preprocessings are also consistent with the original methods. The detailed settings are as below:

ScImpute only involves two parameters. Parameter K denotes the potential number of cell populations, threshold t on dropout probabilities. We set t=0.5 and K=15. This is per the author's instructions in their paper, i.e., a default threshold value of 0.5 is sufficient for most scRNA-seq data, and K should be chosen based on the clustering result of the raw data and the resolution level desired by the users, where K=15 is close to the ground-truth cell-type number.

SAVER is a statistical model and does not expose any hyperparameters in their implementation.

Therefore, we run their default setting.

For scVI, we searched for combinations of n_hidden=[128, 256], n_layer=[1, 2], gene_likelihood=[nb, zinb] on the Lung5 dataset. All settings are repeated five times and the best mean performance is achieved by n_hidden=128, n_layer=1, and gene_likelihood='nb'. Other parameters are per default. We then applied this set of hyperparameters to all three datasets and reported it in the main results.

For DCA, the original hyperparameter optimization mode is broken (there is a relevant unresolved issue on GitHub), and there are no other parameter instructions in the tutorial, therefore we used default parameters.

For GraphSCI, we tried hidden1=[16, 32, 64], hidden2=[32, 64]. Note that GraphSCI does not have other hyperparameters and the default number of hidden sizes is quite small. The performance reported in our paper was obtained from hidden1=32, and hidden2=64.

gimVI software does not expose hyperparameters such as n_hidden and n_layer, so we follow the default settings. Additionally, gimVI requires external scRNA-seq reference data. For the Lung5 dataset, we used data from GSE131907 as a reference. For Kidney, we used data from GSE159115, and for Liver we used data from GSE115469.

scGNN provides two sets of preprocessing in their GitHub repo and we adopt the first one "Cell/Gene filtering without inferring LTMG" for CSV format. Then we follow the corresponding instructions to run their imputation method but get an out-of-memory error during EM algorithm.

Sprod provides batch mode for handling very big spatial datasets. We follow their instructions for dataset without a matching image and ran the batch mode with num_of_batches=30, however, it can not finish running within 48 hours even on the smallest Kidney dataset.

For Spage we follow the official tutorial. For SeDR, we follow their official tutorial for imputation and batch integration, where we consider each FOV as a batch. SeDR does not provide any information on hyperparameters in their tutorial, therefore, we run their default method with 5 random seeds. To adapt SeDR's output to our evaluation protocol, we remove the inherent standardization function from SeDR's '.recon' function.

For Tangram, we found that directly inputting all the data would lead to OOM issue. Therefore,

we follow the official instruction to separate the data into 5 splits and impute them split by split.

B.2 Standard Deviation

For simplicity, we omit the standard deviation in Table 4.2. The detailed experiment statistics are shown below in Table B.1.

Table B.1 The standard deviation of the imputation experiments.

	Lung 5	Lung 5	Lung 5	Kidney 1139	Kidney 1139	Kidney 1139	Liver Norma	Liver Normal	Liver Norma
	RMSE	Pearson	Cosine	RMSE	Pearson	Cosine	RMSE	Pearson	Cosine
scVI	0.2861± 0.0037	0.6231± 0.0136	0.6661± 0.0112	0.2901± 0.0040	0.5834± 0.0152	0.6480± 0.0125	0.2797± 0.0159	0.5749± 0.0641	0.6224± 0.0582
DCA	0.2858± 0.0002	0.6223± 0.0008	0.6648± 0.0007	0.2852± 0.0005	0.5985± 0.0022	0.6597± 0.0017	0.2542± 0.0129	0.657± 0.0461	0.688± 0.0401
GraphSCI	0.3957± 0.0006	0.1334± 0.0016	0.3081± 0.0007	0.3624± 0.0009	0.2403± 0.0048	0.4128± 0.0029	0.3347± 0.0009	0.3707± 0.0065	0.4430± 0.0065
gimVI	0.3170± 0.0018	0.5320± 0.0063	0.5917± 0.0052	0.4387± 0.0015	−0.0104± 0.0004	0.1967± 0.0027	0.4542± 0.0024	−0.0015± 0.0010	0.1167± 0.0050
SpaFormer	0.2785± 0.0005	0.6363± 0.0019	0.6786± 0.0016	0.2794± 0.0042	0.6108± 0.0152	0.671± 0.0121	0.2117± 0.0014	0.7976± 0.0034	0.7797± 0.0032

APPENDIX C

BUILDING SINGLE-CELL FOUNDATION MODEL BEYOND SINGLE CELLS

C.1 Spatially-Resolved Transcriptomic Data

Recently, spatial transcriptomic technologies are developed to spatially resolve transcriptomics profiles (Ståhl et al., 2016; Merritt et al., 2020). With spatial transcriptomics data, researchers can learn the spatial context of cells and cell clusters within a tissue (Burgess, 2019). The major technologies/platforms for spatial transcriptomics are Visium by 10x (Ståhl et al., 2016), GeoMx Digital Spatial Profiler (DSP) (Merritt et al., 2020) by NanoString and CosMx Spatial Molecular Imager (SMI) by NanoString, MERFISH, Vizgen, Resolve, Rebus, and molecular cartography. 10x Visium does not profile at single-cell resolution, and while GeoMx DSP is capable of single-cell resolution through user-drawn profiling regions, the scalability is limited. The most recent platform, CosMx Spatial Molecular Imager (SMI) (He et al., 2022b), can profile consistently at single-cell and even sub-cellular resolution. CosMx SMI follows much of the initial protocol as GeoMx DSP, with barcoding and ISH hybridization. However, the SMI instrument performs 16 cycles of automated cyclic readout, and in each cycle, the set of barcodes (readouts) are UV-cleaved and removed. These cycles of hybridization and imaging yield spatially resolved profiling of RNA and protein at single-cell ($\sim 10\mu m$) and subcellular ($\sim 1\mu m$) resolution. In this work, we use two published and one unpublished dataset produced by the CosMx platform. In order to obtain the cellular level gene expression, CellPose (Stringer et al., 2021) software is applied to conduct cell segmentation.

To give a concrete example, we provide a sample field-of-view (FOV) in Fig. 4.1. Pre-selected types of RNA molecules are captured by the molecular imager, which are denoted as white dots in the figures. Colors in the first sub-figure indicate the protein molecules that are stained. These proteins contribute to the cell segmentation process, which results in the second sub-figure. The final output from the pipeline consists of the position of each cell and a cell-by-gene count matrix, which is produced by counting the number of RNA molecules within each cell. The difference between scRNA-seq and SRT data is further demonstrated in Fig. C.1.

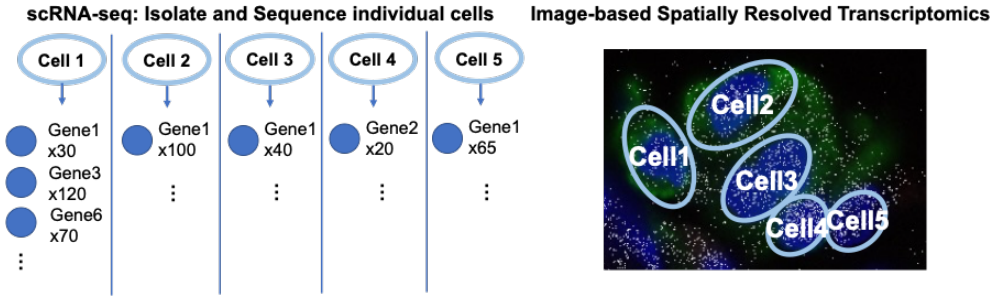


Figure C.1 An illustration of the difference between scRNA-seq and SRT data.

C.2 2D Sinusoid Positional Encodings

Since 2D sinusoidal PE achieves a competitive performance and has a lower complexity on SRT data (Wen et al., 2023), in our transformer encoer, we generate a sinusoidal PE for cells in SRT data, formulated as:

$$\begin{aligned} \mathrm{PE}_{(x,y,2i)} &= \sin\left(x/10000^{4i/d}\right), \mathrm{PE}_{(x,y,2i+1)} = \cos\left(x/10000^{4i/d}\right), \\ \mathrm{PE}_{(x,y,2j+d/2)} &= \sin\left(y/10000^{4j/d}\right), \mathrm{PE}_{(x,y,2j+1+d/2)} = \cos\left(y/10000^{4j/d}\right), \end{aligned} \tag{C.1}$$

where d is the total dimension of positional encoding, $i, j \in [0, d/4)$ specify a specific feature dimension. Let $\tilde{\mathbf{C}} \in \mathcal{R}^{N\times 2}$ be a normalized coordinate matrix, where we normalize and truncate coordinates in $\mathbf{C}$ to integers ranging in $[0, 100)$. x, y then refer to the spatial coordinates from $\tilde{\mathbf{C}}$, e.g., $x = \tilde{\mathbf{C}}_{t,0}$ and $y = \tilde{\mathbf{C}}_{t,1}$ for cell t. In this way, we generate a PE matrix $\mathbf{P} \in \mathcal{R}^{N\times d}$ for every cell in SRT data, where $\mathbf{P}_i$ is the PE vector for cell i. Meanwhile, for scRNA-seq data, a randomly initialized d-dimensional vector p' is shared among all cells, which also results in a placeholder PE matrix $\mathbf{P}$.

C.3 Denoising Variational Lower Bound for Masked Language Modeling

One of the highlights of *CellPLM* is the design of probabilistic latent space. Prior studies have employed variational autoencoders for single-cell analysis, which typically assumes an isotropic Gaussian distribution as the prior distribution of the latent space (Lopez et al., 2018; Xu et al., 2021). While this approach can effectively remove batch effects, it may also result in a loss of information regarding the underlying biological structure of cell groups. To address this limitation, *CellPLM* incorporates the concept of Gaussian mixture variational encoder (Dilokthanakul et al., 2016; Yang et al., 2019; Xu et al., 2023), which utilizes a mixture of Gaussians to capture the information of

distinct functional groups of cells. Formally, for $i \in \{1, \ldots, N\}$, the generative model of cell i can be formulated as:

$$\begin{aligned} p(\mathbf{y}_i; \boldsymbol{\pi}) &= \text{Multinomial}(\boldsymbol{\pi}), \\ p\left(\mathbf{z}_i \mid \mathbf{y}_i\right) &= \prod_{i=1}^{L} \mathcal{N}\left(\boldsymbol{\mu}_{y_{i,l}}, \text{diag}\left(\boldsymbol{\sigma}^2_{y_{i,l}}\right)\right), \\ p_{\theta_{dec}}\left(\mathbf{x}_i \mid \mathbf{z}_i\right) &= \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{z}_i}, \sigma^2 \mathbf{I}\right), \end{aligned} \tag{C.2}$$

where $\mathbf{y}_i \in \mathcal{R}^L$ represents the one-hot latent cluster variable and $\boldsymbol{\pi}$ is its prior; $y_{i,l}$ denotes the l-th entry of $\mathbf{y}_i$; $\boldsymbol{\mu}_{y_l} \in \mathcal{R}^{d_z}$ and $\boldsymbol{\sigma}^2_{y_l} \in \mathcal{R}^{d_z \times d_z}$ denote the mean and variance of the l-th Gaussian component, respectively; and $\boldsymbol{\mu}_{z_i} \in \mathcal{R}^k$ and $\sigma^2 \mathbf{I} \in \mathcal{R}^{k \times k}$ denote the posterior mean and variance of expression $\mathbf{x}_i$, respectively. In this work, we assume that σ^2 is a constant and the posterior mean is parameterized by $\boldsymbol{\mu}_{z_i} = f_{dec}(\mathbf{z}_i; \theta_{dec})$.

To estimate the posterior of $\mathbf{z}_i$ and $\mathbf{y}_i$, we parameterize the inference process with neural networks. Specifically, we assume that the cluster variables $\mathbf{y}$ are independent of the expression $\mathbf{x}_i$ condition on latent variables $\mathbf{z}_i$. The inference model can be formulated as:

$$\begin{aligned} q_{\eta_\mu, \eta_\sigma}(\mathbf{z}_i \mid \mathbf{x}_i) &= \mathcal{N}\left(\hat{\boldsymbol{\mu}}_i, \text{diag}\left(\hat{\boldsymbol{\sigma}}^2_i\right)\right), \\ q_{\eta_\pi}(\mathbf{y}_i \mid \mathbf{z}_i) &= \text{Multinomial}(\hat{\boldsymbol{\pi}}_i), \end{aligned} \tag{C.3}$$

where the estimations are given by

$$\begin{aligned} \mathbf{h}_i &= f_{enc}(\mathbf{x}_i; \eta_{enc}), \\ \hat{\boldsymbol{\mu}}_i &= f_\mu\left(\mathbf{h}_i; \eta_\mu\right), \\ \log\left(\hat{\boldsymbol{\sigma}}^2_i\right) &= f_\sigma\left(\mathbf{h}_i; \eta_\sigma\right), \\ \hat{\boldsymbol{\pi}}_i &= f_\pi\left(\mathbf{z}_i; \eta_\pi\right). \end{aligned} \tag{C.4}$$

Here $f_{enc}(\cdot; \eta_{enc})$ represents the transformer encoder, $f_\mu(\cdot; \eta_\mu)$, $f_\sigma(\cdot; \eta_\sigma)$ and $f_\pi(\cdot; \eta_\pi)$ are neural networks. A log-evidence lower bound (ELBO) can be derived from this generative model for the optimization purpose (Dilokthanakul et al., 2016). However, as mentioned in Section 5.2.1, our pre-training framework incorporates a cell language model, where parts of the input gene expression matrix $\mathbf{X}$ are masked. This will result in a modified objective. To formalize the problem, recall

Table C.1 Hyperparameters for pretraining *CellPLM* model.

	CellPLM
encoder hidden dim	1024
encoder layers	4
latent dimension	512
decoder hidden dim	1024
decoder layers	2
model dropout	0.2
cell mask rate	0.75
gene mask rate	0.25
learning rate	2e-4
weight decay	1e-8
num of cluster (for GMM)	16
total parameter	82,402,543

that previously we defined the masked set as $\mathcal{M}$. On top of that, we denote $\mathbf{M} \in \mathcal{R}^{N\times k}$ as a mask indicator matrix such that

$$\mathbf{M}_{i,j} = \begin{cases} 1 & \text{if } (i,j) \notin \mathcal{M}, \\ 0 & \text{if } (i,j) \in \mathcal{M}. \end{cases}$$

Let $\tilde{\mathbf{X}} \in \mathcal{R}^{N\times k}$ be the masked gene expression matrix given by the element-wise multiplication $\tilde{\mathbf{X}} = \mathbf{M} \odot \mathbf{X}$. The objective of cell language model with Gaussian mixture prior, i.e., a denoising variational lower bound (Im Im et al., 2017), can be formulated as:

$$\begin{aligned} \mathcal{L}_{\text{CellLM}} =& \mathbb{E}_{q(\mathbf{Z},\mathbf{Y}|\tilde{\mathbf{X}})}\mathbb{E}_{p(\tilde{\mathbf{X}}|\mathbf{X})}\left[\ln \frac{p_\theta(\mathbf{X},\mathbf{Z},\mathbf{Y})}{q_\eta(\mathbf{Z},\mathbf{Y}\mid\tilde{\mathbf{X}})}\right] \\ =& \underbrace{\mathbb{E}_{q_{\eta_{enc}}(\mathbf{Z}|\tilde{\mathbf{X}})}\mathbb{E}_{p(\tilde{\mathbf{X}}|\mathbf{X})}\left[\log p_{\theta_{dec}}(\mathbf{X}\mid\mathbf{Z})\right]}_{\mathcal{L}_{\text{recon}}} - \underbrace{\mathbb{E}_{q_{\eta_\pi}(\mathbf{Y}|\mathbf{Z})}\left[\text{KL}\left(q_{\eta_{enc}}(\mathbf{Z}\mid\tilde{\mathbf{X}})\|p(\mathbf{Z}\mid\mathbf{Y})\right)\right]}_{\mathcal{L}_{\text{cond}}} \\ & - \underbrace{\mathbb{E}_{q_{\eta_{enc}}(\mathbf{Z}|\tilde{\mathbf{X}})}\left[\text{KL}\left(q_{\eta_\pi}(\mathbf{Y}\mid\mathbf{Z})\|p(\mathbf{Y})\right)\right]}_{\mathcal{L}_{\text{Y}}}. \end{aligned} \tag{C.5}$$

C.4 Pre-training Settings

C.4.1 Hyperparameter Settings

We pre-trained *CellPLM* model with the hyperparameters specified in Table C.1.

Table C.2 List of dataset and data sources. External links will be included in our github repo.

Source	Datasets
HTCA	HTAN_HTAPP, HTAN_Stanford, HTAN_Vanderbilt, HTAN_BU
HCA	cxg_PBMCs, EGAS00001004571_PBMCs, eQTLAutoimmune, covid19autoimmunityPBMCs, VanDerWijst-Human-10x5pv1, cxg_Airways, COMBAT2022, TabulaSapiens, PAN.A01.v01.raw_count.20210429.PFI.embedding, GTEx_8_tissues_snRNAseq_atlas_071421.public_obs
GEO	GSE139324, GSE136246, GSE179994,GSE131907, GSE171145, GSE139555, GSE156728_CD4, GSE148071, PMID_34663877, Qian_et_al_2020_LC, GSE176021, GSE156728_CD8
Other Atlas (deduplicated)	MalteEtAl_LungAtlas, TICAtlas

C.4.2 Datasets for Pre-training

The dataset for pre-training contains 11.4 million cells from scRNA-seq and SRT data. scRNA-seq data consist of 4.7 million cells from human tumor cell atlas (HTCA, https://humantumoratlas.org), 1.4 million cells from human cell atlas (HCA, https://www.humancellatlas.org), and 2.6 million cells from Gene Expression Omnibus (GEO, https://www.ncbi.nlm.nih.gov/geo). All of them are public available data, elucidated in table C.2. A more detailed list and external links will be disclosed in our GitHub repository. Note that although our *CellPLM* is capable to handle various input feature sets, when we concatenated these scRNA-seq datasets, we used inner join by default of Anndata package. As a result, all scRNA-seq datasets only contain a $13,500$ common gene set. We will address this issue and increase the size of the gene set in future versions of *CellPLM*.

The SRT datasets we used are publicly available on Nanostring official website (nanostring.com), where 2.7 million cells and $1,000$ genes are measured. Both scRNA-seq and SRT data are preprocessed with library size normalization and log1p transformation, following the convention in Stuart et al. (2019).

C.5 Additional Experimental Details

In this section, we provide more experimental details about fine-tuning, baselines, and evaluation metrics under each downstream task.

C.5.1 scRNA-seq Denoising

Downstream Task Datasets. In scRNA-seq denoising task, we evaluate *CellPLM* on two datasets, i.e., PBMC 5K and Jurkat from 10x Genomics[1]. It is worth noting that during the prepossessing stage, we performed sub-setting on both datasets to ensure that all the genes were included in the gene set of pre-training data. Additionally, any genes with zero counts were removed from the analysis. We list the statistics of them in Table C.3.

Table C.3 scRNA-seq denoising datasets.

	5K PBMC	Jurkat
Number of genes	33,538	32,738
Number of cells	5,247	3,258
Num genes picked	7,197	7,618

Evaluation Metrics. Following the setting of scGNN (Wang et al., 2021), scGNN2.0 (Gu et al., 2022) and DeepImpute (Arisdakessian et al., 2019), we performed synthetic dropout simulation with missing at random (MAR) setting. While scGNN only considered a simple scenario, i.e., randomly flipped 10% of the non-zero entries to zeros, DeepImpute applied cell-wise mask with masking probability given by a multinomial distribution. Specifically, we adapted the setting from DeepImpute with exponential kernel. For cell i that contains at least 5 expressed genes, the probability that one non-zero count $x_{i,j}$ is masked during the training process is given by $\mathrm{Exp}(0, 20)$:

$$p_{i,j} = \frac{1}{20} e^{-\frac{x}{20}},$$
$$q_{i,j} = \frac{p_{i,j}}{\sum_{j=0}^{J_i} p_{i,j}},$$

where J_i is the number of non-zero counts within cell i. We masked 10% of the non-zero counts according to $\{q_{i,j}\}_{j=0}^{J_i}$ and evaluate model performance on the masked entries. We calculate the root mean squared error (RMSE) and mean absolute error (MAE) between the predicted values and ground truth.

Baselines. (1) DeepImpute (Arisdakessian et al., 2019) employed a strategy of dividing genes into subsets and constructing deep neural networks to impute scRNA-seq data. We implemented

[1] 10x Genomics datasets are available at https://support.10xgenomics.com/single-cellgene-expression/datasets.

DeepImpute with default settings in DANCE (Ding et al., 2024) package. (2) scGNN2.0 (Gu et al., 2022) incorporated a feature autoencoder, a cluster autoencoder and a graph attention autoencoder for simultaneous imputation and clustering. scGNN2.0 is implemented by DANCE package with default settings. (3) GraphSCI (Rao et al., 2021) combined autoencoders with graph convolution networks among a gene-gene similarity graph. We accommodated the implementation of GraphSCI in DANCE package. (4) SAVER (Huang et al., 2018) leveraged Poisson LASSO regression to model the scRNA-seq counts with Poisson–gamma mixture. We utilized R package SAVER to illustrate the performance of it. (5) DCA (Eraslan et al., 2019) introduced an autoencoder framework based on zero inflated negative binomial (ZINB) distribution. We applied DCA to aforementioned datasets with its Python pacakge. (6) MAGIC (Van Dijk et al., 2018) utilized Markov affinity to capture gene-gene relationship and impute missing gene expression. We adapted its Python package to access the performance of it. (7) scImpute (Li and Li, 2018) developed a Gamma and Gaussian mixture model to identify dropout values. We revealed the performance of scImpute with its R pacakge.

Fine-tuning. Since denoising task requires model to recover the gene expression matrix, we can directly get the zero shot performance of *CellPLM* by specifying the gene set of target dataset. Additionally, we fine-tuned *CellPLM* by replacing the pre-trained decoder with a MLP head and initializing encoder with pre-trained weights. Additionally, for methods require model selection on validation set, we performed another 10% simulation dropout and treat masked entries as validation set. The fine-tuned *CellPLM* was trained on MSE reconstruction loss, while the best model was selected by evaluating MSE on validation set.

C.5.2 Spatial Tanscriptomic Imputation

Downstream Task Datasets. To evaluate spatial tanscriptomic imputation models at single-cell resolution, we collected two samples from MERSCOPE FFPE Human Immuno-oncology Data[2]. Specifically, we chose "Lung cancer 2" and "Liver cancer 2" as our samples, and subsequently referred to them as "Lung2" and "Liver2" respectively. The Lung2 and Liver2 datasets were

[2]Merscope ffpe human immuno-oncology datasets are available at https://info.vizgen.com/ffpe-showcase.

subsetted to align with the gene set of the pre-training data. Additionally, we removed the fields of view (FOVs) that contained fewer than 100 cells and retained only the first 100 FOVs from both datasets. Note that all baselines require reference scRNA-seq datasets to impute the unseen genes of SRT data, we collected GSE131907 (Kim et al., 2020) and GSE151530 (Ma et al., 2021a) for lung cancer and liver cancer, respectively. The statistics of all datasets are illustrated in Table C.4.

Table C.4 Spatial tanscriptomic imputation datasets.

	Lung2	Liver2	GSE131907	GSE151530
Number of genes	500	500	29,634	18,667
Number of cells	836,739	598,141	208,506	56,721
Num genes picked	462	446	All	ALL
Num cells picked	40,114	20,629	All	All

Evaluation Metrics. Following the evaluation pipeline proposed by Avşar et al. (Avşar and Pir, 2023), we selected target genes of SRT data with stratified sampling according to gene sparsity. Specifically, we grouped genes into four categories: low sparse, moderate sparse, high sparse, and very-high sparse. Empirically, the boundaries were defined as $[x < 75, 75 \leq x < 90, 90 \leq x < 95, 95 \leq x]$ to approximate the Gaussian mean and standard deviation slices. Subsequently, we randomly selected 25 genes from each sparsity group and remove them from training data. After training the models, we calculate the evaluation metrics on the target genes. Namely, we compute the root mean squared error (RMSE), Pearson's correlation coefficient (PCC) and cosine similarity (Cosine) between the ground truth values and the corresponding imputed values in a gene-wise approach.

Baselines. (1) SpaGE (Abdelaal et al., 2020) relied on domain adaptation to map scRNA-seq data onto SRT data and utilized a k-nearest-neighbor (k-NN) graph to predict unseen genes. We implemented SpaGE with default settings on both datasets. (2) stPlus (Shengquan et al., 2021) developed an autoencoder framework for learning cell embeddings and imputing SRT genes using a weighted k-NN approach. The performance of stPlus is accessed by its Python package. (3) gimVI (Lopez et al., 2019) introduced a variational autoencoder based model with protocol-specific treatments on scRNA-seq data and SRT data. We applied the algorithm with default settings to

evaluate the performance of gimVI. (4) Tangram (Biancalani et al., 2021) utilized a deep learning approach to learn the spatial alignment of scRNA-seq data based on a reference SRT dataset with consistent spatial maps. We evaluated Tangram with its Python package.

Fine-tuning. Similar to scRNA-seq denoising, the spatial tanscriptomic imputation task requires the ouput of the model to be the gene expression. Thus, we directly fine-tune *CellPLM* on the pre-trained weights while specifying the input genes and target genes. The last two batches were hold out for validation.

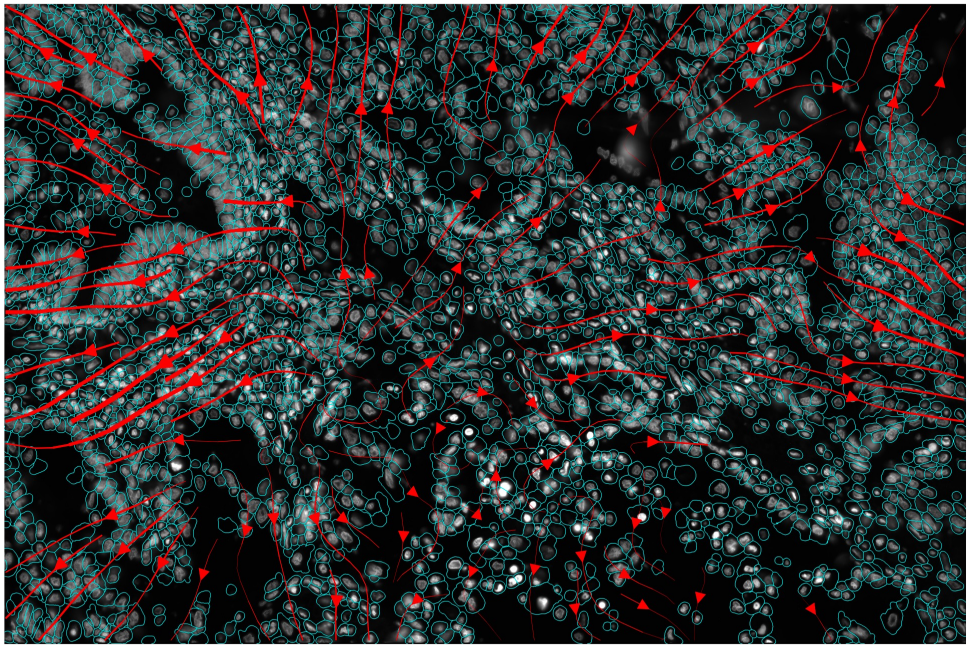


Figure C.2 Visualization of attention matrix demonstrate cell-cell communication.

Visualization of attention. One essential multi-cell task is cell-cell communication (CCC) inference, where CCC mainly represents biochemical signaling through ligand-receptor binding across cells (Cang et al., 2023). Our *CellPLM* applies self-attention mechanism on cell level, from which we can study the interaction strength given by cell attention matrix. As a preliminary study, we extract the attention matrix between cells from a random chosen field of view (FOV) in Cosmx Liver dataset. The attention matrix is treated as CCC scores, and we visualize the results following the stream plot setting in Cang et al. (2023). As shown in the Figure C.2 in our supplementary PDF, there are some strong trends on the left side and right side of the FOV, suggesting further exploration

of specific signaling pathways for the included cells. This case study showcase the potential of our *CellPLM* model in cell-cell communication research. We hope our model can facilitate more insightful biological research in the future.

C.5.3 Perturbation Prediction

The perturb-seq technology has been established to examine the gene expression response at the single-cell level when subjected to pooled perturbations (Dixit et al., 2016). By comparing the gene expression before and after perturbation, downstream analysis of differential expression (DE) enables the identification of genes that play a crucial role in disease progression. To assess the potential benefits of *CellPLM* in gene-level tasks, we conduct experiments to predict the expression value of genes after perturbation. Following the setting of GEARS (Roohani et al., 2022), we partition the perturbations into training, validation, and test sets, ensuring that none of the test perturbations are encountered during the optimization process.

Two perturbation datasets are employed for evaluation: (1) the Adamson Perturb-Seq dataset (Adamson et al., 2016), consisting of 87 one-gene perturbations; and (2) the Norman Perturb-Seq dataset (Norman et al., 2019), containing 131 two-gene perturbations and 105 one-gene perturbations. To evaluate the performance of perturbation prediction, we employ Root Mean Square Error (RMSE) to measure the degree of similarity between the non-zero ground-truth expression values and corresponding predicted gene expressions. In addition, following previous settings in GEARS (Roohani et al., 2022), we also present the RMSE calculated on the top 20 deferentially-expressed genes.

We compare the performance between *CellPLM* and two baselines, i.e., a recent preprint GEARS method (Roohani et al., 2022), and scGen (Lotfollahi et al., 2019). The results in Figure C.3 imply that *CellPLM* achieves the lowest RMSE values across all settings.

Downstream Task Datasets. We included the Adamson Perturb-Seq dataset (Adamson et al., 2016) for one-gene perturbations and the Norman Perturb-Seq dataset (Norman et al., 2019) for two-gene perturbations. We followed the preprocess pipeline of GEARS (Roohani et al., 2022) and both datasets were then gene-wise subsetted to fit in the gene set of pre-training data. The statistics

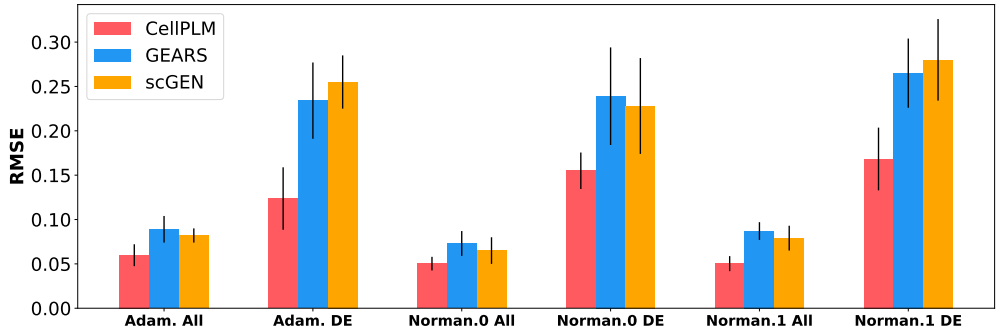


Figure C.3 (*Task 3*) The RMSE performance (↓) on Adamson Perturb-Seq and the Norman Perturb-Seq datasets. The Norman Perturb-seq dataset consists of two settings: one-gene perturbations and two-gene perturbations, denoted as Norm.0 and Norm.1, respectively.

Table C.5 Perturbation prediction datasets.

	Adamson	Norman
Number of genes	5,060	5,045
Number of cells	68,603	91,205
Num genes picked	3,246	2,353
Num one-gene pert.	87	105
Num two-gene pert.	–	131

are summaried in Table C.5.

Evaluation Metrics. Following the setting of GEARS (Roohani et al., 2022), we applied data split such that the testing perturbation are unseen during the training process. Specifically, For Adamson dataset, we randomly hold out 25% of the perturbations for testing and 10% of the perturbations within the training set for validation. For Norman dataset, two settings for two-gene perturbations are implemented for evalutation purpose: $1/2$ unseen and $2/2$ unseen. We excluded all two-gene combinations in which at least one of the individual genes involved in the combination belonged to the unseen set. Finally, we evaluate the performance by calculating the root mean squared error (RMSE) between the predictions and the true values within the testing set.

Baselines. (1) GEARS (Roohani et al., 2022) utilized gene co-expression knowledge graph and Gene Ontology-derived knowledge graph to model the influence of perturbations. We followed the recommended parameter settings within its Python package to access the performance. (2) scGen (Lotfollahi et al., 2019) built a conditional variational autoencoders and incoporated vector arithmetics to model phenomena response. We implemented scGen with its Python package on both

datasets.

Fine-tuning. For one perturbation, we set the input of perturbed genes to be -100 to mimic the gene perturbation action. During the fine-tuning process, we substituted the original batch-aware decoder with a simplified MLP decoder. Additionally, we initialized the remaining components of *CellPLM* with pre-trained weights. The final model was chosen to be the best-performed model on the validation set.

C.5.4 Cell Type Annotation

Cell type annotation is a crucial step in single-cell analysis as it enables the identification and characterization of distinct cell populations within a tissue or organism. This information is crucial for understanding the functional diversity, developmental trajectories, and disease relevance of different cell types, providing insights into biological processes and facilitating targeted therapeutic approaches.

Downstream Task Datasets. We assess the performance of *CellPLM* on the task of cell type annotation on hPancreas (Chen et al., 2023) and Multiple Sclerosis (MS) (Schirmer et al., 2019), which are suggested by (Cui et al., 2023). The hPancreas dataset contains five scRNA-seq datasets of human pancreas cells, divided into reference and query sets with annotations, including 13 cell types and 11 cell types, respectively. The Multiple Sclerosis dataset (M.S.), sourced from EMBL-EBI, includes 9 healthy control and 12 M.S. samples. 3,000 highly variable genes were retained.

Evaluation Metrics. We evaluate cell type annotation performance based on two standard classification metrics, Macro Precision and Macro F1 score.

Baselines. To benchmark the performance of *CellPLM*, we compare it with both pre-trained models including scGPT (Cui et al., 2023), scBERT (Yang et al., 2022), as well as non-pre-trained SOTA models including ACTINN (Ma and Pellegrini, 2020), CellTypist (Domínguez Conde et al., 2022), SingleCellNet (Tan and Cahan, 2019), and TOSICA (Chen et al., 2023). For baseline methods, we adhere to their provided guidelines and utilize the default parameter setting. The performance metrics reported for scBERT, TOSICA and scGPT in this task are directly obtained from scGPT papers.

Fine-tuning. For *CellPLM* model, we attach a feed forward layer to the pre-trained encoder and latent space and tune the downstream model on the downstream dataset with a standard cross entropy loss.

C.6 Comparison between *CellPLM* and scVI

As a supplement to the zero-shot clustering experiments in Section 5.3.1, we add an additional comparison with scVI (Lopez et al., 2018) on the same dataset. As shown in Figure C.4, *CellPLM* successfully outperforms scVI without any training or fine-tuning, while the latter was trained on this specific dataset.

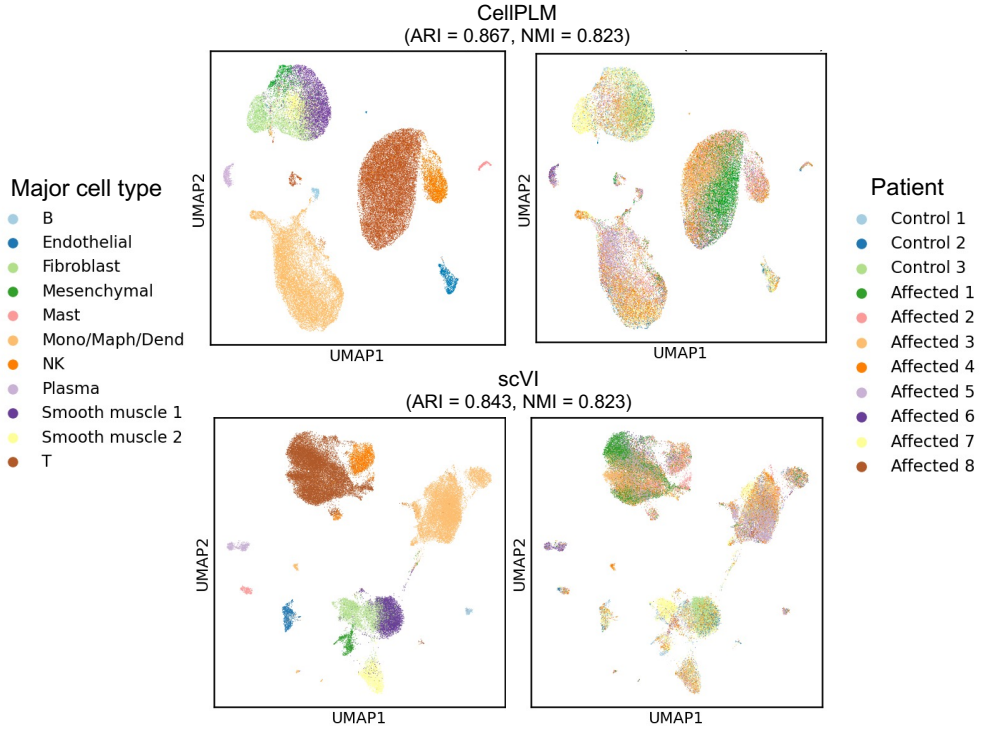


Figure C.4 Visualization and comparison between *CellPLM* (zero-shot) and scVI on the clustering task.

C.7 Visualization of Gene Embeddings

In order to examine whether gene interactions can be encoded in *CellPLM*, we present a visualization of pre-trained gene embeddings from the gene expression embedder in Figure C.5. From the visualization, the gene embeddings maintain some latent structures. To further verify the effectiveness of the latent structure, we highlight a specific family of genes, HLA genes. There are multiple classes of genes in HLA gene family. For example, HLA class I genes (e.g., HLA-A, -B, and

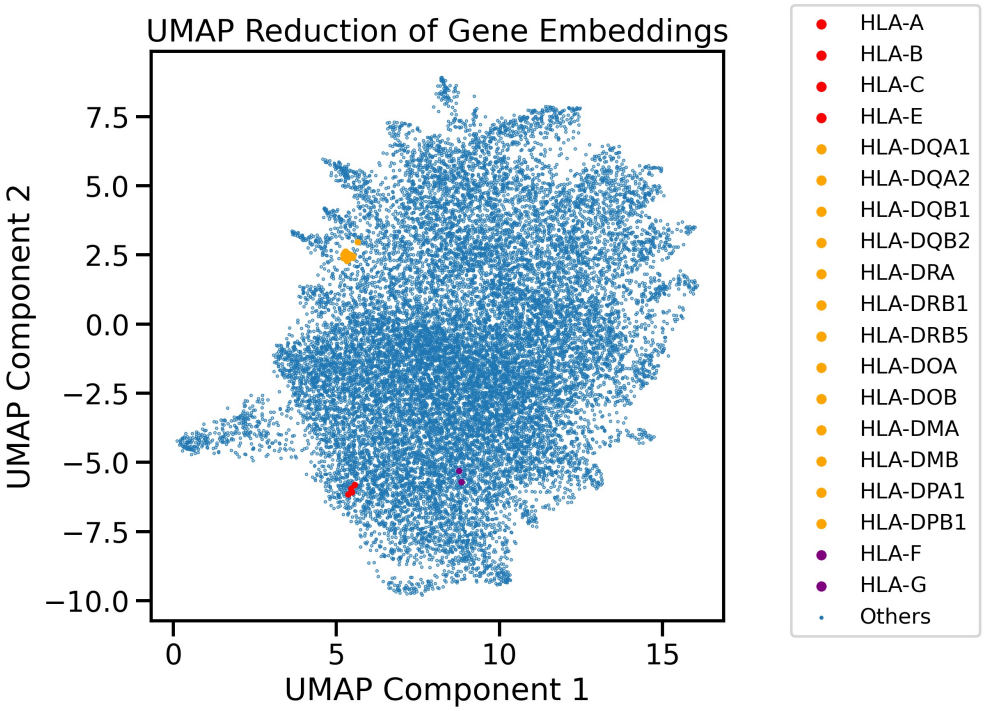


Figure C.5 Visualization of gene embeddings in the pre-trained *CellPLM* demonstrate that *CellPLM* successfully captures gene interactions in the initial gene embeddings. For example, HLA Class I genes and HLA Class II perfectly form two clusters in the gene embedding space.

-C) present endogenous peptides to responding CD8+ T Cells while the class II (e.g., HLA-DR, -DP, and –DQ) process exogenous peptides for presentation to CD4+ helper T Cells. From the UMAP visualization, HLA gene embedding clusters perfectly match the functionality and characteristics of those genes.