# Graph Neural Networks for Multimodal Single-Cell Data Integration

**Anonymous Author(s)**
Affiliation
Address
email

## 1 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) [27] enables simultaneous quantification of mRNA expression and surface proteins abundance; methods like sci-CAR [4], Paired-seq [34], and SNARE-seq [6] 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. 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 et al. [9] summarized three major tasks: (1) *Modality prediction* aims at predicting the features of one modality from the features of another modality [31]; (2) *Modality matching* focuses on identifying the correspondence of cells between different modalities [30]; and (3) *Joint embedding* requires embedding the features of two modalities into the same low-dimensional space [27]. 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 [9] 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. BABEL[31] translated between the transcriptome (mRNA) and chromatin (DNA) profiles of a single cell based on an encoder-decoder architecture; scMM[19] implemented a mixture-of-experts deep generative model for joint embedding learning and modality prediction. Cobolt[11] acquired joint embedding via a variant of Multimodal Variational Autoencoder (MVAE[33]). MOFA2[1] 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

Submitted to NeurIPS 2021 AI for Science Workshop.

requires additional processing to cells, which can lead to extra noise and drop-out in the resulting data [14, 18]. Therefore, it is a desired technique that can mitigate the negative impact of such noise.

Recently, the advances in graph neural networks (GNNs) [13, 32, 3, 10, 16] pave the way for addressing the aforementioned issues in single-cell data integration. Specifically, GNNs aggregate information from neighborhoods to update node embeddings iteratively [10]. 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 [17]. 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.

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 [25, 29, 7, 22], 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[1]. 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 multimodal single-cell data integration. Our 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 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. Our *scMoGNN* 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 the modality prediction task in a 2021 NeurIPS competition, and currently outperforms other models for all three tasks.

## 2 Related Work

**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. Song et al. [25] propose scGCN model for knowledge transfer in single-cell omics (mRNA or DNA) based on Graph Convolutional Networks [13]. scGNN [29] 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 [22] 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 [24] 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 [8, 26, 12] or statistical based methods [28, 23, 30] and 2) autoencoder based methods [31, 11]. Specifically, BABEl [31] 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[11] acquires joint embedding via a variant of Multimodal Variational Autoencoder (MVAE[33]). 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

[1] Our solution won the first place of the modality prediction task in a NeurIPS competition.

single-cell data integration and build a GNNs-based general framework to broadly work on these three key tasks from one NeurIPS 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.

# 3 Problem Statement

There are three modalities spanning through each task, i.e., 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 $N$ is the number of cells, and $k$ is the feature dimension for each cell. In our work, we 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* $f_\theta$ *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.

**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* $f_\theta$ *parameterized by* $\theta$ *such that* $\mathbf{M_2} = f_\theta(\mathbf{M_1})$.

**Task 2: Modality Matching.** The goal 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* $f_{\theta_1}$ *parameterized by* $\theta_1$ *and* $f_{\theta_2}$ *parameterized by* $\theta_2$ *to map them into the same space such that*

$$\mathbf{S} = g(f_{\theta_1}(\mathbf{M_1}), f_{\theta_2}(\mathbf{M_2})) \tag{1}$$

*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* $\mathbf{M_2}$.

**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* $f_{\theta_1}$*,* $f_{\theta_2}$ *and* $f_{\theta_3}$ *parameterized by* $\theta_1$*,* $\theta_2$ *and* $\theta_3$ *accordingly to project them into downstream tasks,*

$$\mathbf{H} = f_{\theta_3}\big(CONCAT(f_{\theta_1}(\mathbf{M_1}), f_{\theta_2}(\mathbf{M_2}))\big) \tag{2}$$

*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* $f_{\theta_3}$ *on concatenation of* $f_{\theta_1}(\mathbf{M_1})$ *and* $f_{\theta_2}(\mathbf{M_2})$. $\mathbf{H}$ will be measured using six different metrics broken into two classes: biology conservation and batch removal (see Appendix A for more details).

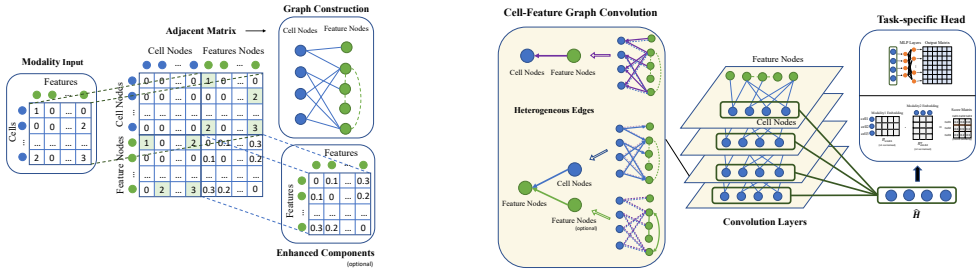
# 4 Method

## 4.1 A General GNN Framework

To develop a GNN-based framework for single cell data integration, there are three 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 tasks.

**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




(a) Graph Construction (b) Cell-Feature Graph Convolution with Task Head

Figure 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.

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 $\mathbf{A} = \begin{pmatrix} \mathbf{O} & \mathbf{M} \\ \mathbf{M}^T & \mathbf{O} \end{pmatrix} \in \mathbb{R}^{(N+k)\times(N+k)}$, 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 $\mathcal{G} = (\mathbf{A}, \mathbf{X}_{\text{feat}}, \mathbf{X}_{\text{cell}})$.

**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 [10] 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 $\mathbf{h}_i^{l+1} = \text{Update}(\mathbf{h}_i^l, \text{Agg}(\mathbf{h}_j^l | j \in \mathcal{N}_i))$, 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 $\mathbf{H}^1 = \sigma(\mathbf{X}_{\text{feat}}\mathbf{W}_{\text{feat}} + \mathbf{b}_{\text{feat}})$, where $\mathbf{W}_{\text{feat}} \in \mathbb{R}^{d'\times d}$ is a transformation matrix, $d$ is the dimension of the hidden space and $\sigma$ 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) [13]. 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:

$$\mathbf{m}_{\mathcal{U}\to\mathcal{V}}^{i,l} = \sigma(\mathbf{b}_{\mathcal{U}\to\mathcal{V}}^{l} + \sum\nolimits_{j\in\mathcal{N}_i, v_i\in\mathcal{V}} \frac{e_{ji}}{c_{ji}} \mathbf{h}_j^l \mathbf{W}_{\mathcal{U}\to\mathcal{V}}^{l}) \tag{3}$$

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 $c_{ji} = \sqrt{\sum_{k\in\mathcal{N}_j} e_{jk}}\sqrt{\sum_{k\in\mathcal{N}_i} e_{ki}}$. Obviously, Eq. (3) 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:

$$\mathbf{m}_{\mathcal{V}\to\mathcal{U}}^{i,l} = \sigma(\mathbf{b}_{\mathcal{V}\to\mathcal{U}}^{l} + \sum\nolimits_{j\in\mathcal{N}_i, u_i\in\mathcal{U}} \frac{e_{ji}}{c_{ji}} \mathbf{h}_j^l \mathbf{W}_{\mathcal{V}\to\mathcal{U}}^{l}) \tag{4}$$

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:

$$\mathbf{h}_i^{l+1} = \mathbf{h}_i^l + \mathbf{m}_{\mathcal{U}\to\mathcal{V}}^{i,l}, \quad \mathbf{h}_j^{l+1} = \mathbf{h}_j^l + \mathbf{m}_{\mathcal{V}\to\mathcal{U}}^{j,l}, \tag{5}$$

where $v_i \in \mathcal{V}$ and $u_j \in \mathcal{U}$. In Eq. (5), 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 4.2.

**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 $\hat{\mathbf{Y}}$: $\hat{\mathbf{Y}} = \mathrm{Head}\left(\mathrm{Readout}_\theta(\mathbf{H}_{\mathcal{U}}^1, ..., \mathbf{H}_{\mathcal{U}}^L)\right)$, where $\mathbf{H}_{\mathcal{U}}^i$ refers to embeddings of all cell nodes in $i$-th layer, $\mathrm{Readout}(\cdot)_\theta$ is a trainable aggregation function, $\mathrm{Head}(\cdot)$ 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.

### 4.2 Model Specifications

We show how *scMoGNN* is applied in the three important tasks in single-cell data integration.

**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. 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 $\mathbf{A} = \begin{pmatrix} \mathbf{O} & \mathbf{M} \\ \mathbf{M}^T & \mathbf{P} \end{pmatrix}$, 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.

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. (3) and Eq. (4), 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 $\mathbf{h}_i^{l+1} = \mathbf{h}_i^l + \alpha \cdot \mathbf{m}_{\mathcal{V}\to\mathcal{V}}^{i,l} + (1-\alpha)\cdot \mathbf{m}_{\mathcal{U}\to\mathcal{V}}^{i,l}$, where $\alpha$ is either a hyper-parameter or a learnable scaler to control the ratio between inter-feature aggregation and cell-feature aggregation. 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: $\hat{\mathbf{H}} = \sum_{i=1}^{L} \mathbf{w}_i \cdot \mathbf{H}_{\mathcal{U}}^i$, 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: $\hat{\mathbf{Y}} = \hat{\mathbf{H}}\mathbf{W} + \mathbf{b}$. A rooted mean squared error (RMSE) loss is then calculated as $\mathcal{L} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{Y}_i - \hat{\mathbf{Y}}_i)^2}$.

**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. 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 $\hat{\mathbf{H}}_{\text{m1}} \in \mathbb{R}^{N \times d_1}$ and $\hat{\mathbf{H}}_{\text{m2}} \in \mathbb{R}^{N \times d_2}$. We set $d_1 = d_2$ so they can be directly multiplied together. Thus, we calculate the cosine similarity between each cell pair as $\mathbf{S} = \hat{\mathbf{H}}'_{\text{m1}} \cdot \hat{\mathbf{H}}'^{T}_{\text{m2}}$, 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. Then we can express the loss function as follows:

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

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

$$\begin{aligned}\mathcal{L}_{\text{aux}} &= \mathcal{L}_{\text{pred12}} + \mathcal{L}_{\text{pred21}} + \mathcal{L}_{\text{recon11}} + \mathcal{L}_{\text{recon22}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{\text{m2}} - f_{\theta_2}(\hat{\mathbf{H}}_{\text{m1}}))^2 \\ &+ \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{\text{m1}} - f_{\theta_1}(\hat{\mathbf{H}}_{\text{m2}}))^2 + \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{\text{m1}} - f_{\theta_1}(\hat{\mathbf{H}}_{\text{m1}}))^2 + \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{\text{m2}} - f_{\theta_2}(\hat{\mathbf{H}}_{\text{m2}}))^2 \quad (6)\end{aligned}$$

where $\mathbf{X}_{\text{m1}}$ and $\mathbf{X}_{\text{m2}}$ refer to the preprocessing results of two modalities respectively, $f_{\theta_1}$ and $f_{\theta_2}$ 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.

**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 modality prediction. As suggested by the work [15], 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: $\hat{\mathbf{Y}}_{i,t} = \frac{e^{\hat{\mathbf{H}}_{i,t}}}{\sum_{k=1}^{T} e^{\hat{\mathbf{H}}_{i,k}}}$, 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:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{cell type}} + \mathcal{L}_{\text{regular}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X}_{\text{LSI}} - f_\theta(\hat{\mathbf{H}}))^2 + \sum_{t=1}^{T} \mathbf{Y}_t \log(\hat{\mathbf{Y}}_t) + \beta * \|\hat{\mathbf{H}}_{\tilde{\mathcal{J}}}\|_2 \quad (7)$$

where $f_\theta$ 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.

Table 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** |

# 5 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. We defer the experimental setup to Appendix.

## 5.1 Main Results

**Modality Prediction.** As shown in Table 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, 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.

**Modality Matching.** As shown in Table 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 $0.08$.

**Joint Embedding.** In Table 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.

Table 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** |

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

| | NMI cluster | Cell type ASW | Cc_con | Traj_con | Batch ASW | Graph connect. | Avg. metric |
|---|---|---|---|---|---|---|---|
| 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** |

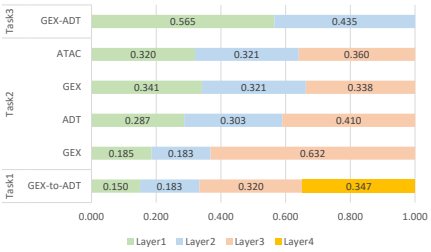



Figure 2: Parameter analysis of layer weight $\mathbf{w}$.

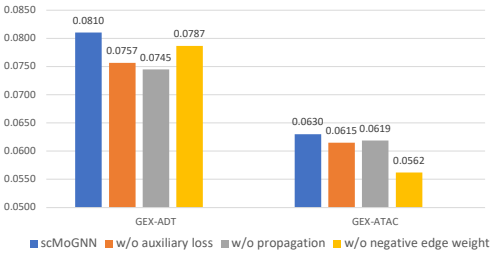



Figure 3: Ablation study for modality matching.

### 5.2 Further Analysis

**Ablation Study.** We investigate if we really benefit from graph structural information. We take the modality matching 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 3. 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 3. Without the supervision of auxiliary losses, *scMoGNN* lost generalization ability, behaving as poorly as without graph structural information.

**Parameter Analysis.** We analyzed an important parameter in our framework, i.e. $\mathbf{w}$ in $\hat{\mathbf{H}} = \sum_{i=1}^{L} \mathbf{w}_i \cdot \mathbf{H}_{\mathcal{U}}^{i}$ in modality prediction, to gain a deeper insight on *scMoGNN*. We show values of $\mathbf{w}$ learned by *scMoGNN* in different tasks in Figure 2. 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. In modality prediction, more higher-order information is referenced, indicating that the model needs to enrich more information from similar cells or similar features.

## 6 Conclusion

In this work, we proposed a general framework *scMoGNN* based on GNNs for multimodal single-cell data integration. It can be broadly applied on all three key tasks, modality prediction, modality matching and joint embedding. 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 wins 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.

# References


[1] Ricard Argelaguet, Damien Arnol, Danila Bredikhin, Yonatan Deloro, Britta Velten, John C Marioni, and Oliver Stegle. 2020. MOFA+: a statistical framework for comprehensive integration of multi-modal single-cell data. *Genome biology* 21, 1 (2020), 1–17.

[2] Fatima Batool and Christian Hennig. 2021. Clustering with the average silhouette width. *Computational Statistics & Data Analysis* 158 (2021), 107190.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *ArXiv preprint* (2018).

[4] Junyue Cao, Darren A Cusanovich, Vijay Ramani, Delasa Aghamirzaie, Hannah A Pliner, Andrew J Hill, Riza M Daza, Jose L McFaline-Figueroa, Jonathan S Packer, Lena Christiansen, et al. 2018. Joint profiling of chromatin accessibility and gene expression in thousands of single cells. *Science* 361, 6409 (2018), 1380–1385.

[5] Zhi-Jie Cao and Ge Gao. 2022. Multi-omics single-cell data integration and regulatory inference with graph-linked embedding. *Nature Biotechnology* (2022), 1–9.

[6] Song Chen, Blue B Lake, and Kun Zhang. 2019. High-throughput sequencing of the transcriptome and chromatin accessibility in the same cell. *Nature biotechnology* 37, 12 (2019), 1452–1457.

[7] Madalina Ciortan and Matthieu Defrance. 2022. GNN-based embedding for clustering scRNA-seq data. *Bioinformatics* 38, 4 (2022), 1037–1044.

[8] Zhana Duren, Xi Chen, Mahdi Zamanighomi, Wanwen Zeng, Ansuman T Satpathy, Howard Y Chang, Yong Wang, and Wing Hung Wong. 2018. Integrative analysis of single-cell genomics data by coupled nonnegative matrix factorizations. *Proceedings of the National Academy of Sciences* 115, 30 (2018), 7723–7728.

[9] Luecken et al. 2021. A sandbox for prediction and integration of DNA, RNA, and proteins in single cells. In *NeurIPS Datasets and Benchmarks Track (Round 2)*.

[10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 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)*.

[11] Boying Gong, Yun Zhou, and Elizabeth Purdom. 2021. Cobolt: integrative analysis of multimodal single-cell sequencing data. *Genome biology* 22, 1 (2021), 1–21.

[12] Suoqin Jin, Lihua Zhang, and Qing Nie. 2020. scAI: an unsupervised approach for the integrative analysis of parallel single-cell transcriptomic and epigenomic profiles. *Genome biology* 21, 1 (2020), 1–19.

[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[14] Jeongwoo Lee, Do Young Hyeon, and Daehee Hwang. 2020. Single-cell multiomics: technologies and data analysis methods. *Experimental & Molecular Medicine* 52, 9 (2020), 1428–1442.

[15] Qiao Liu, Shengquan Chen, Rui Jiang, and Wing Hung Wong. 2021. Simultaneous deep generative modelling and clustering of single-cell genomic data. *Nature machine intelligence* 3, 6 (2021), 536–544.

[16] Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Graph Neural Networks with Adaptive Residual. *Advances in Neural Information Processing Systems* 34 (2021).

[17] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1202–1211.

[18] Eleni P Mimitou, Caleb A Lareau, Kelvin Y Chen, Andre L Zorzetto-Fernandes, Yuhan Hao, Yusuke Takeshima, Wendy Luo, Tse-Shun Huang, Bertrand Z Yeung, Efthymia Papalexi, et al. 2021. Scalable, multimodal profiling of chromatin accessibility, gene expression and protein levels in single cells. *Nature biotechnology* 39, 10 (2021), 1246–1258.

[19] Kodai Minoura, Ko Abe, Hyunha Nam, Hiroyoshi Nishikawa, and Teppei Shimamura. 2021. A mixture-of-experts deep generative model for integrated analysis of single-cell multiomics data. *Cell reports methods* 1, 5 (2021), 100071.

[20] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems* 31 (2018).

[21] Philip Sedgwick. 2014. Spearman's rank correlation coefficient. *Bmj* 349 (2014).

[22] Xin Shao, Haihong Yang, Xiang Zhuang, Jie Liao, Penghui Yang, Junyun Cheng, Xiaoyan Lu, Huajun Chen, and Xiaohui Fan. 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 (2021).

[23] Ronglai Shen, Adam B Olshen, and Marc Ladanyi. 2009. Integrative clustering of multiple genomic data types using a joint latent variable model with application to breast and lung cancer subtype analysis. *Bioinformatics* (2009).

[24] Qianqian Song and Jing Su. 2021. DSTG: deconvoluting spatial transcriptomics data through graph-based artificial intelligence. *Brief Bioinform* 22, 5 (2021), 1–13.

[25] Qianqian Song, Jing Su, and Wei Zhang. 2021. scGCN is a graph convolutional networks algorithm for knowledge transfer in single cell omics. *Nature Communications* 12, 1 (2021), 1–11.

[26] Genevieve L Stein-O'Brien, Raman Arora, Aedin C Culhane, Alexander V Favorov, Lana X Garmire, Casey S Greene, Loyal A Goff, Yifeng Li, Aloune Ngom, Michael F Ochs, et al. 2018. Enter the matrix: factorization uncovers knowledge from omics. *Trends in Genetics* 34, 10 (2018), 790–805.

[27] Marlon Stoeckius, Christoph Hafemeister, William Stephenson, Brian Houck-Loomis, Pratip K Chattopadhyay, Harold Swerdlow, Rahul Satija, and Peter Smibert. 2017. Simultaneous epitope and transcriptome measurement in single cells. *Nature methods* 14, 9 (2017), 865–868.

[28] Tim Stuart, Andrew Butler, Paul Hoffman, Christoph Hafemeister, Efthymia Papalexi, William M Mauck III, Yuhan Hao, Marlon Stoeckius, Peter Smibert, and Rahul Satija. 2019. Comprehensive integration of single-cell data. *Cell* 177, 7 (2019), 1888–1902.

[29] Juexin Wang, Anjun Ma, Yuzhou Chang, Jianting Gong, Yuexu Jiang, Ren Qi, Cankun Wang, Hongjun Fu, Qin Ma, and Dong Xu. 2021. scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses. *Nature communications* 12, 1 (2021), 1–11.

[30] Joshua D Welch, Alexander J Hartemink, and Jan F Prins. 2017. MATCHER: manifold alignment reveals correspondence between single cell transcriptome and epigenome dynamics. *Genome biology* 18, 1 (2017), 1–19.

[31] Kevin E Wu, Kathryn E Yost, Howard Y Chang, and James Zou. 2021. BABEL enables cross-modality translation between multiomic profiles at single-cell resolution. *Proceedings of the National Academy of Sciences* 118, 15 (2021).

[32] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[33] Zizhen Yao, Hanqing Liu, Fangming Xie, Stephan Fischer, Ricky S Adkins, Andrew I Aldridge, Seth A Ament, Anna Bartlett, M Margarita Behrens, Koen Van den Berge, et al. 2021. A transcriptomic and epigenomic cell atlas of the mouse primary motor cortex. *Nature* 598, 7879 (2021), 103–110.

[34] Chenxu Zhu, Miao Yu, Hui Huang, Ivan Juric, Armen Abnousi, Rong Hu, Jacinta Lucero, M Margarita Behrens, Ming Hu, and Bing Ren. 2019. An ultra high-throughput method for single-cell joint analysis of open chromatin and transcriptome. *Nature structural & molecular biology* 26, 11 (2019), 1063–1070.

# A Details of Metrics in Task 3

## A.1 Biology Conservation Metrics

1. **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` [2]). 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.
2. **Cell type ASW:** The silhouette width metric indicates the degree to which observations with identical labels are compact. The average silhouette width (ASW) [2], 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:

$$ASW = (ASW_C + 1)/2 \tag{8}$$

   where C denotes the set of all cell identity labels.
3. **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:

$$CC_{conservation} = 1 - \frac{|Var_{after} - Var_{before}|}{Var_{before}} \tag{9}$$

   where values near to 0 suggest less conservation of variance explained by the cell cycle, while 1 represents complete conservation.
4. **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 [21], $s$, between the pseudotime values before and after integration. The final score is scaled to a value between 0 and 1 using the equation:

$$trajectoryconservation = (s + 1)/2 \tag{10}$$

   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.2 Batch Removal Metrics

1. **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:

$$\text{batch } ASW_j = \frac{1}{|C_j|} \sum_{i \in C_j} 1 - |s(i)| \tag{11}$$

   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:

$$\text{batch } ASW = \frac{1}{|M|} \sum_{j \in M} \text{batch } ASW_j \tag{12}$$

[2] `https://en.wikipedia.org/wiki/Louvain_method`

Table 4: 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 |

Table 5: 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 |

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.

2. **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 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:

$$gc = \frac{1}{|C|} \sum_{c \in C} \frac{|LCC\left(G\left(N_c; E_c\right)\right)|}{|N_c|} \tag{13}$$

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.3 Metric Aggregation

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

$$S_{\text{overall},i} = 0.6 \cdot S_{bio,i} + 0.4 \cdot S_{batch,i} \tag{14}$$

## B Data Statistics

The Table 4 and Table 5 provide statistics about dataset used in modality prediction task and modality matching task respectively.

## C Experimental Setup

Note that in this experiment, we follow the official settings and datasets in the multimodal single-cell data integration competition at NeurIPS 2021 [9] and we compare the performance of the proposed framework with that from the top winners in the competition.

### C.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 4 in Appendix B. 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.5, (2) using group normalization for aggregation results and dismissing the edge weight normalization stated. Besides, we empirically set our parameter $\alpha$ to $0.5$. Additionally, in order to fit the high-dimensional ATAC features, we enabled node sampling.

**Baselines.** In Table 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[20] 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.

### C.2 Modality Matching

**Datasets.** The majority of the modality matching dataset is the same as the modality prediction dataset (as shown in Table 5 in Appendix B, 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, we only compare our models with the winning team and runner-up 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 [5] (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.

### C.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 C.2.

**Baselines.** We briefly describe the other three models in Table 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 [15]. (3) GLUE [5], an auto-encoder model guided by an external knowledge graph.