NAR Genomics and Bioinformatics, 2024, **6**, 1–11
https://doi.org/10.1093/nargab/lqae004
Advance access publication date: 29 January 2024
Methods Article

scBiG for representation learning of single-cell gene expression data based on bipartite graph embedding

Ting Li[1,†], Kun Qian[1,†], Xiang Wang[1], Wei Vivian Li[2,*] and Hongwei Li[1,*]

[1]School of Mathematics and Physics, China University of Geosciences, Wuhan 430074, China
[2]Department of Statistics, University of California, Riverside, Riverside, CA 92507, USA

[*]To whom correspondence should be addressed. Tel: +86 18627905633; Fax: +86 18627905633; Email: hwli@cug.edu.cn
Correspondence may also be addressed to Wei Vivian Li. Tel: +1 951 827 0233; Fax: +1 951 827 0233; Email: weil@ucr.edu
[†]The first two authors should be regarded as Joint First Authors.

Abstract

Analyzing single-cell RNA sequencing (scRNA-seq) data remains a challenge due to its high dimensionality, sparsity and technical noise. Recognizing the benefits of dimensionality reduction in simplifying complexity and enhancing the signal-to-noise ratio, we introduce scBiG, a novel graph node embedding method designed for representation learning in scRNA-seq data. scBiG establishes a bipartite graph connecting cells and expressed genes, and then constructs a multilayer graph convolutional network to learn cell and gene embeddings. Through a series of extensive experiments, we demonstrate that scBiG surpasses commonly used dimensionality reduction techniques in various analytical tasks. Downstream tasks encompass unsupervised cell clustering, cell trajectory inference, gene expression reconstruction and gene co-expression analysis. Additionally, scBiG exhibits notable computational efficiency and scalability. In summary, scBiG offers a useful graph neural network framework for representation learning in scRNA-seq data, empowering a diverse array of downstream analyses.

Introduction

The continuous advancement of single-cell RNA sequencing (scRNA-seq) technologies has opened up new avenues for transcriptomic research. The scRNA-seq techniques offer great potential for exploring gene expression at the individual cell level. They have propelled scientific exploration by revealing previously unknown cell types and scrutinizing cellular heterogeneity (1). However, analyzing high-dimensional and sparse single-cell gene expression data is not a trivial task (2,3). Dimensionality reduction has emerged as a pivotal step in scRNA-seq analysis, enabling the reduction of complexity and enhancement of signal to support downstream analyses.

A plethora of dimensionality reduction methods have been adopted or proposed for scRNA-seq data to obtain embeddings of single cells. Earlier methods include principal component analysis (PCA) (4), independent component analysis (ICA) (5) and zero-inflated factor analysis (ZIFA) (6). Both PCA and ICA are generic dimensionality reduction methods based on linear projections. ZIFA incorporates an additional zero-inflation layer to specifically account for the excess zeros in scRNA-seq data. More recently, deep learning-based methods have been developed, including autoencoders like DCA (7) and SAUCIE (8), variational autoencoders like scVI (9), and graph autoencoders like scGAE (10). Using autoencoder networks and zero-inflated negative binomial (ZINB) distributions, scVI and DCA represent each cell as a point in a low-dimensional latent space. SAUCIE uses multiple layers of encoders and decoders and an embedding layer to achieve multiple tasks such as imputation and batch effect removal in addition to dimensionality reduction. scGAE constructs a nearest neighbor graph for cells and trains a graph neural network (GNN) to produce cell embeddings.

In addition to representation learning methods that treat each cell independently, methods based on GNNs (11,12) aim to capture higher-order relationships between nodes by iterative information passing, making them well suited for modeling complex scRNA-seq data. For example, scGNN (13) uses a Gaussian mixture model and a GNN with multimodal autoencoders to learn cell–cell relationships. Graph-sc (14) constructs a cell–gene graph, obtains initial low-dimensional cell embeddings by PCA and trains a graph autoencoder to update cell embeddings. There have been a few methods that extended the representation learning from cells to other features. For example, SIMBA (15) utilizes graph structures to co-embed cells and various features, such as genes and open chromatin regions, into a shared latent space and uses these embeddings for downstream analysis. siVAE (16) is a deep generative model that uses pairs of encoders and decoders to learn representations of genes and cells in the latent space based on the variational autoencoder framework.

Inspired by the success of graph-based representation learning in scRNA-seq data analysis (17), we propose the scBiG (representation learning of single-cell gene expression data based on Bipartite Graph embedding) method, which leverages higher-order information in the cell–gene bipartite graph to simultaneously obtain cell and gene embeddings. scBiG's key innovation lies in modeling cell–gene relationships by transforming a gene expression matrix into a cell–gene bipartite graph. Using a model-driven graph reconstruction strategy, it conducts self-supervised training of a GNN on this sparse graph, effectively capturing higher-order topological

Received: October 19, 2023. **Revised:** December 19, 2023. **Editorial Decision:** January 3, 2024. **Accepted:** January 9, 2024

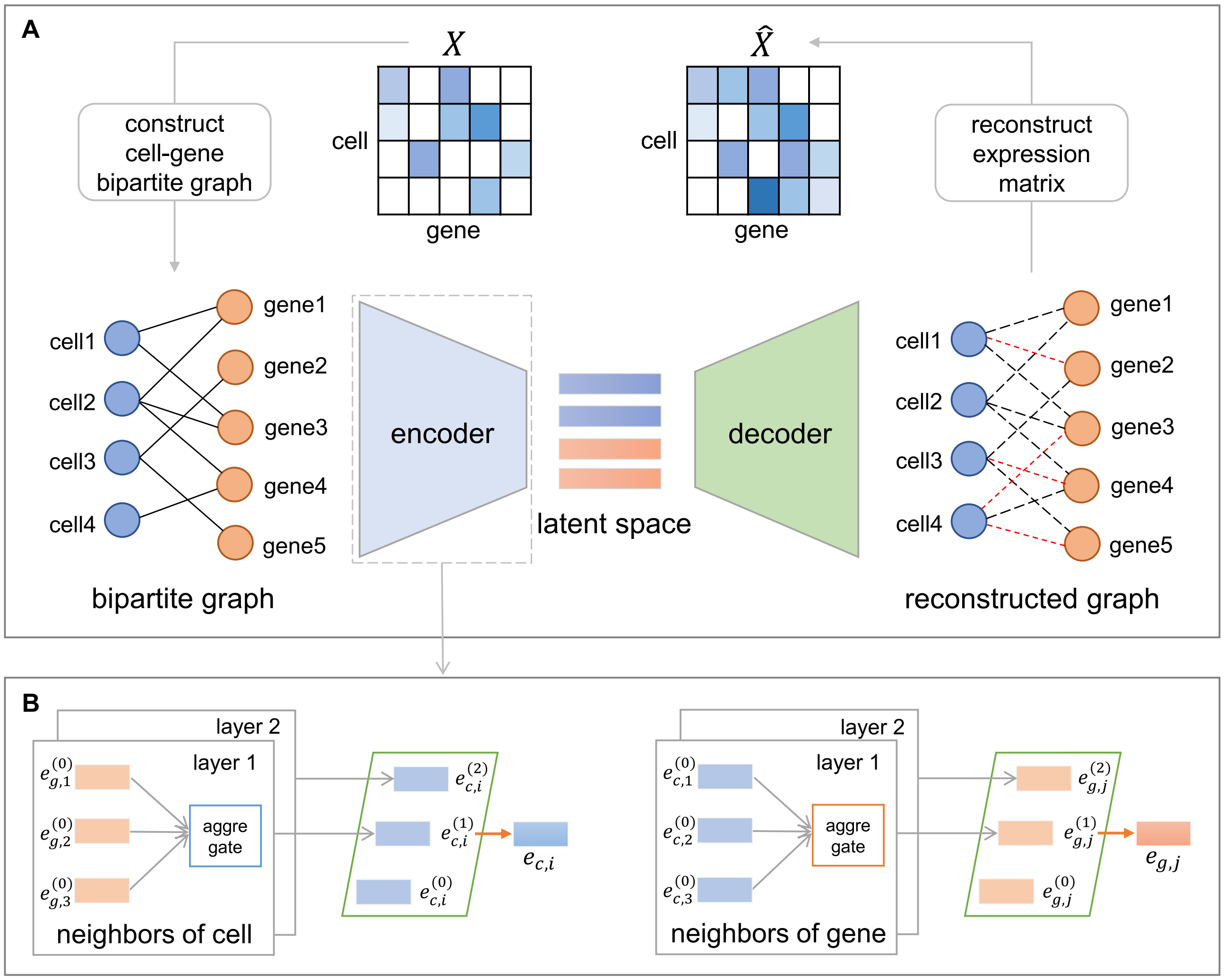

Figure 1. Overview of the scBiG method. (**A**) scBiG takes the count matrix as input, constructs a cell–gene bipartite graph and then initializes node embeddings. The GCN-based encoder updates the node embeddings and the ZINB model-based decoder reconstructs the gene expression profiles. (**B**) Network structure of the GCN-based encoder. Initial cell and gene embeddings ($\boldsymbol{e}_{c,i}^{(0)}$ and $\boldsymbol{e}_{g,j}^{(0)}$) are generated by random initialization, and the network passes, aggregates and updates the neighborhood information for cell and gene nodes in the bipartite graph, respectively. The final node embeddings ($\boldsymbol{e}_{c,i}$ and $\boldsymbol{e}_{g,j}$) are obtained as the weighted average of the embeddings in previous layers.

features. As we will demonstrate in the results, the method facilitates the reconstruction of scRNA-seq data while extracting biologically meaningful representations of both cells and genes. Specifically, we evaluated the performance of scBiG's cell embeddings in both clustering analysis and trajectory inference. Additionally, the learned gene embeddings have been shown to capture cell-type-specific gene co-expression patterns. Overall, we expect scBiG to be a useful tool for learning low-dimensional representations (of both cells and genes) and extracting biological signals from scRNA-seq data.

Materials and methods

An overview of the scBiG method

By constructing a cell–gene bipartite graph from the single-cell gene expression matrix, scBiG converts the dimensionality reduction problem to a node embedding problem given the bipartite graph, allowing it to simultaneously learn low-dimensional representations of both cells and genes. Specifically, scBiG employs a graph autoencoder network to extract higher-order representations of cells and genes from the bipartite graph. First, the graph encoder takes the initial cell and gene embeddings, as well as the cell–gene bipartite graph, to calculate the final embeddings of both cells and genes. Subsequently, these final embeddings, along with the bipartite graph, are utilized by the graph decoder to reconstruct the gene expression matrix (Figure 1A) by estimating ZINB parameters for each cell–gene pair.

The encoder in scBiG uses multiple layers of graph convolutional network (GCN) (18,19), in which each layer accepts a cell–gene bipartite graph of interactions and node embeddings from the previous layer. The node embeddings are updated iteratively to capture higher-order features by aggregating cell-to-gene and gene-to-cell interaction information. The final cell and gene embeddings are obtained through weighted averaging across layers (Figure 1B).

In order to train the GCN, scBiG constructs a loss function based on the negative log-likelihood function of ZINB distributions along with a regularization term. Through the self-supervised encoding–decoding reconstruction strategy, scBiG effectively learns embeddings of cells and genes, which are useful for various downstream analytical tasks. Finally, scBiG outputs the cell and gene embeddings, as well as the reconstructed gene expression matrix (optional) obtained through the reconstructed complete cell–gene bipartite graph.

Data pre-processing

We assume that scRNA-seq data are summarized as a count matrix where rows represent cells and columns represent genes. The pre-processing steps of scBiG are as follows. First, genes expressed in <3 cells and cells with <200 expressed genes are filtered out. The remaining count matrix is denoted as $\boldsymbol{X} = [x_{ij}] \in \mathbb{R}^{m\times n}$, where x_{ij} represents the expression of gene j in cell i, m represents the number of cells and n represents the number of genes. Next, for cell i, we calculate the library size factor l_i as

$$l_i = \frac{\sum_{j=1}^{n} x_{ij}}{\mathrm{median}_{i'}\{\sum_{j=1}^{n} x_{i'j}\}}. \tag{1}$$

Then, the counts are normalized by the library size factors and log-transformed to get the normalized expression values:

$$\widetilde{x}_{ij} = \log\left(\frac{x_{ij}}{l_i} + 1\right). \tag{2}$$

In addition, we define the gene factor s_j as the the maximum expression value of gene j across all cells:

$$s_j = \max_i \left\{\widetilde{x}_{ij}\right\}. \tag{3}$$

The library size factor l_i and gene factor s_j would be used in a subsequent step of expression matrix reconstruction.

Construction of the cell–gene bipartite graph

To represent the relationship between cells and genes, we convert the gene expression matrix into a cell–gene bipartite graph denoted as $G = (U, V, \boldsymbol{A})$. In this graph, U denotes the set of cells, V denotes the set of genes and $\boldsymbol{A}$ denotes the adjacency matrix:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{O}_1 & \boldsymbol{R} \\ \boldsymbol{R}^{\mathrm{T}} & \boldsymbol{O}_2 \end{pmatrix}, \tag{4}$$

where $\boldsymbol{O}_1 \in \mathbb{R}^{m\times m}$ and $\boldsymbol{O}_2 \in \mathbb{R}^{n\times n}$ are matrices of zeros, and $\boldsymbol{R} = [r_{ij}] \in \mathbb{R}^{m\times n}$ is the cell–gene adjacency matrix:

$$r_{ij} = \begin{cases} 1, & \text{if gene } j \text{ is expressed in cell } i, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

We define the edge between gene j and cell i as a positive edge if $r_{ij} = 1$ and a negative edge if $r_{ij} = 0$.

Graph encoder based on GCN

Given the cell–gene bipartite graph G, we use a GCN to encode representations of cells and genes. Specifically, the initial cell embeddings $\boldsymbol{e}_{\mathrm{c},i}^{(0)} \in \mathbb{R}^h$ and gene embeddings $\boldsymbol{e}_{\mathrm{g},j}^{(0)} \in \mathbb{R}^h$ are randomly initialized and treated as model parameters, where h is the embedding dimension. Then, starting from the first graph convolutional layer, the cell embeddings and gene embeddings are updated in the kth layer as

$$\begin{cases} \boldsymbol{e}_{\mathrm{c},i}^{(k)} = \sum_{g\in N_{\mathrm{c},i}} \frac{1}{\sqrt{|N_{\mathrm{c},i}|}\sqrt{|N_{\mathrm{g},j}|}} \boldsymbol{e}_{\mathrm{g},j}^{(k-1)}, \\ \boldsymbol{e}_{\mathrm{g},j}^{(k)} = \sum_{c\in N_{\mathrm{g},j}} \frac{1}{\sqrt{|N_{\mathrm{c},i}|}\sqrt{|N_{\mathrm{g},j}|}} \boldsymbol{e}_{\mathrm{c},i}^{(k-1)}, \end{cases} \tag{6}$$

where k ($1 \le k \le K$) is the layer index and K is the number of graph convolution layers; $N_{\mathrm{c},i}$ and $N_{\mathrm{g},j}$ represent the neighbor sets of cell i and gene j in the bipartite graph; and $|N_{\mathrm{c},i}|$ and $|N_{\mathrm{g},j}|$ are the number of neighbors of cell i and gene j (i.e., the degrees of cell i and gene j). $1/\sqrt{|N_{\mathrm{c},i}|}\sqrt{|N_{\mathrm{g},j}|}$ serves as a degree normalization term.

After K layers of graph convolution, scBiG obtains (K + 1) sets of embeddings for cell i ($\boldsymbol{e}_{\mathrm{c},i}^{(0)}, \ldots, \boldsymbol{e}_{\mathrm{c},i}^{(K)}$) and gene j ($\boldsymbol{e}_{\mathrm{g},j}^{(0)}, \ldots, \boldsymbol{e}_{\mathrm{g},j}^{(K)}$), respectively. Since the embeddings from the last layer may suffer from oversmoothing as the layer number increases (20), we do not directly use them as the inputs of the decoder. Instead, we calculate the weighted average of these embeddings to obtain the final embeddings, which are denoted as $\boldsymbol{e}_{\mathrm{c},i}$ for cell i and $\boldsymbol{e}_{\mathrm{g},j}$ for gene j:

$$\begin{cases} \boldsymbol{e}_{\mathrm{c},i} = \sum_{k=0}^{K} w^{(k)} \boldsymbol{e}_{\mathrm{c},i}^{(k)}, \\ \boldsymbol{e}_{\mathrm{g},j} = \sum_{k=0}^{K} w^{(k)} \boldsymbol{e}_{\mathrm{g},j}^{(k)}, \end{cases} \tag{7}$$

where $\{w^{(k)}\}_{k=0}^{K}$ are layer weights and take values between 0 and 1.

Model-guided graph decoder

Given the embeddings of all cells and genes, along with a provided bipartite graph G' whose edges $\{(i, j)\}$ represent a list of cell–gene pairs whose expression values are to be predicted, a model-guided graph decoder is constructed. We use the ZINB distribution to model the count of gene j in cell i:

$$\begin{aligned} \mathrm{ZINB}(x_{ij}|\pi_{ij}, \mu_{ij}, \theta_{ij}) = (1-\pi_{ij}) \cdot \mathrm{NB}(x_{ij}|\mu_{ij}, \theta_{ij}) \\ + \pi_{ij} \cdot \delta_0(x_{ij}), \end{aligned} \tag{8}$$

where x_{ij} is the count value of gene j in cell i, π_{ij} is the probability of zero inflation, NB represents the negative binomial distribution with mean μ_{ij} and dispersion θ_{ij}, and $\delta_0(\cdot)$ is the Dirac delta function. The decoder processes the above-mentioned inputs and generates the parameters of the ZINB distribution for each pair of cell and gene within the graph G', thereby reconstructing the bipartite graph.

To learn the ZINB parameters, π_{ij}, μ_{ij} and θ_{ij} for edge (i, j) in G', we integrate a generalized matrix factorization (GMF) model (21) into the decoder, which takes embeddings $\boldsymbol{e}_{\mathrm{c},i}$ of cell i and $\boldsymbol{e}_{\mathrm{g},j}$ of gene j as input and implements the following forward propagation process:

$$\begin{cases} \boldsymbol{e} = \boldsymbol{e}_{\mathrm{c},i} \odot \boldsymbol{e}_{\mathrm{g},j}, \\ \mu_{ij} = \exp\left(\mathrm{sigmoid}(\boldsymbol{W}_\mu \boldsymbol{e}) \times s_j\right) \times l_i, \\ \theta_{ij} = \mathrm{softplus}\left(\boldsymbol{W}_\theta \boldsymbol{e} \times s_j\right), \\ \pi_{ij} = \mathrm{sigmoid}(\boldsymbol{W}_\pi \boldsymbol{e}), \end{cases} \tag{9}$$

where $\odot$ denotes the Hadamard product, $\boldsymbol{W}_\mu \in \mathbb{R}^{1\times h}$, $\boldsymbol{W}_\theta \in \mathbb{R}^{1\times h}$ and $\boldsymbol{W}_\pi \in \mathbb{R}^{1\times h}$ are weight parameters, and l_i and s_j, respectively, are the library size factor and gene factor calculated in the pre-processing step. For mean μ_{ij}, we apply the sigmoid activation function to constrain the output between 0 and 1, and then recover the expression level using gene factor s_j and library size factor l_i. For dispersion θ_{ij}, we use the softplus activation function to ensure nonnegativity.

Self-supervised learning

scBiG utilizes the encoder to extract cell and gene embeddings from the cell–gene bipartite graph G and subsequently feeds these embeddings, along with a graph to be reconstructed, into the decoder to predict ZINB parameters for each cell–gene pair. To improve efficiency and enhance generalization, during each iteration, scBiG randomly samples 10% positive edges from G to construct a positive subgraph G_{pos}, and samples the same number of negative edges to construct a negative subgraph G_{neg}. The model then employs both G_{pos} and G_{neg} as inputs to the decoder and computes the loss function based on the negative log-likelihood (NLL) function:

$$L_{\text{NLL}} = -\frac{1}{N}\sum_{(i,j)\in G_{\text{pos}}} \log(\text{ZINB}(x_{ij}; \pi_{ij}, \mu_{ij}, \theta_{ij})) - \frac{1}{N}\sum_{(i',j')\in G_{\text{neg}}} \log(\text{ZINB}(0; \pi_{i'j'}, \mu_{i'j'}, \theta_{i'j'})), \quad (10)$$

where N is the number of edges in G_{pos} or G_{neg}. To prevent overfitting, we also add regularization terms of the initial embeddings and define the overall loss function of scBiG as

$$L = L_{\text{NLL}} + \lambda \frac{\|\boldsymbol{E}_{\text{c}}^{(0)}\|_{\text{F}}^2 + \|\boldsymbol{E}_{\text{g}}^{(0)}\|_{\text{F}}^2}{2(m+n)}, \quad (11)$$

where $\lambda > 0$ is the regularization coefficient, $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm, and $\boldsymbol{E}_{\text{c}}^{(0)} = \left(\boldsymbol{e}_{\text{c},1}^{(0)}, \ldots, \boldsymbol{e}_{\text{c},m}^{(0)}\right) \in \mathbb{R}^{h\times m}$ and $\boldsymbol{E}_{\text{g}}^{(0)} = \left(\boldsymbol{e}_{\text{g},1}^{(0)}, \ldots, \boldsymbol{e}_{\text{g},n}^{(0)}\right) \in \mathbb{R}^{h\times n}$ are the initial embedding matrices of cells and genes, respectively.

After training the model parameters, scBiG utilizes the encoder to extract all cell and gene embeddings. Subsequently, scBiG constructs a complete bipartite graph G_{comp} in which each pair of cell and gene is connected by an edge. The gene/cell embeddings and this complete bipartite graph are then provided to the graph decoder, which reconstructs the graph by estimating the ZINB parameters, $\hat{\mu}_{ij}$, $\hat{\theta}_{ij}$, $\hat{\pi}_{ij}$, for each edge in G_{comp}. Finally, scBiG takes $\hat{\mu}_{ij}$ as the reconstructed expression value $\hat{x}_{ij}$ for gene j in cell i to obtain the reconstructed expression matrix $\hat{\boldsymbol{X}}$.

Implementation

The scBiG package is implemented in Python 3.8, using Scanpy version 1.9.1 (22) for data pre-processing and PyTorch version 1.10.0 (23) and Deep Graph Library version 0.8.2 (https://doi.org/10.48550/arXiv.1909.01315) for implementing the GNN. In our model, the number of graph convolution layers K is set to 2, and the embedding dimension h is set to 64. The weights of GCN layers are set as $w^{(0)} = 1/2$, $w^{(1)} = 1/4$, and $w^{(2)} = 1/4$. We use Adam (24) for optimization with a learning rate of 0.1 and train scBiG for a total of 200 epochs. The parameter λ is set to 0.0001 in the loss function.

In the numerical evaluation, we used two-dimensional visualization of the embedded data. Therefore, we also performed a comparison by directly setting the embedding dimension h to 2 in the clustering analysis (Supplementary Table S1). The results indicate that, due to the complexity of scRNA-seq data, two-dimensional embeddings are not sufficient to fully capture their intrinsic features.

Alternative methods for comparison

In our study, we evaluated the performance of scBiG in three analytical tasks: dimensionality reduction, reconstruction of gene expression and gene co-expression analysis. For dimensionality reduction, we compared scBiG with nine alternative methods, including Seurat (25), ICA (5), ZIFA (6), graph-sc (14), scGAE (10), scGNN (13), DCA (7), scVI (9) and SIMBA (15). When implementing these methods, their own parameters for the dimensions were used if default values were given in the software; otherwise, we set the embedding dimension to 64, the same value used in scBiG. For reconstruction of gene expression, we compared it with five alternative methods, including MAGIC (26), scImpute (27), ALRA (28), DCA (7) and scVI (9). For gene co-expression analysis, we compared it with siVAE (16) and SIMBA (15). Supplementary Table S2 provides a brief summary of the alternative methods and corresponding software tools.

Data description

For clustering analysis, we downloaded seven real scRNA-seq datasets with cell type annotations (Supplementary Table S3). These seven datasets included three human datasets and four mouse datasets. For simplicity, these datasets are referred to as human pancreas (1724 cells) (29), mouse ES (2717 embryonic stem cells) (30), mouse bladder (2746 cells) (31), mouse kidney (3660 cells) (32), human PBMC (4271 cells) (33), human kidney (5685 cells) (34) and mouse retina (14 653 cells) (35). For cell trajectory analysis, we downloaded two real datasets (Supplementary Table S3), including a human embryo dataset (90 cells from 6 developmental stages) (36) and a mouse embryo dataset (268 cells from 10 developmental stages) (37). In addition, we also used PROSSTT (38) to generate a simulated dataset (500 cells and 10 000 genes) with ground truth and pseudotime orders (Supplementary Methods). For gene co-expression analysis, we downloaded two real datasets (Supplementary Table S3), including a human liver dataset (500 cells) (39) and a mouse trachea dataset (7193 cells) (40). For reconstruction of gene expression, we used Splatter (41) to generate simulated single-cell count matrices of 5000 cells and 10 000 genes (Supplementary Methods). By adjusting the parameter 'dropout.mid' in the software, we generated four simulated datasets with different sparsity levels (0.8, 0.85, 0.9 and 0.95). In order to evaluate runtime and memory usage, we also used Splatter to generate six datasets with 10 000 genes and 2000, 4000, 8000, 16 000, 32 000 and 64 000 cells, respectively.

Results

Cell embeddings obtained by scBiG improve clustering analysis

To evaluate scBiG's ability to learn informative cell representations, we performed clustering analysis on the cell embeddings obtained by scBiG and nine alternative dimensionality reduction methods (Seurat, ICA, ZIFA, graph-sc, scGAE, scGNN, DCA, scVI and SIMBA). In order to focus on the evaluation of dimensionality reduction, we obtained the cell embeddings from different methods on seven real datasets (Supplementary Table S3), and all embedded data were clustered using the Louvain algorithm (42) and visualized via the UMAP algorithm (43). These processes were implemented in the Scanpy package (22). We calculated the clustering accuracy based on the adjusted Rand index (ARI) and normalized mutual information (NMI) (Supplementary Methods). Our results suggest that scBiG achieved the highest average ARI and NMI scores across the seven datasets, followed by SIMBA and ICA (Figure 2A and Supplementary Figures S1–S6). In addition, scBiG achieved the highest average ARI scores on four datasets: human pancreas, mouse ES, mouse kidney and mouse retina. For example, on the human pancreas dataset, scBiG achieved an ARI value of 0.92 and clearly delineated the major pancreatic cell types (Figure 2B and C). The second best method on this dataset was ICA with an ARI of 0.76, which had difficulty distinguishing the alpha and ductal cells. Furthermore, on larger-scale data such as the mouse retina dataset with 14 653 cells, scBiG also led to the highest ARI of 0.76 (Supplementary Figure S2).

Next, to assess the robustness of the above methods, we created 10 subsamples of each real dataset by randomly selecting 95% of the cells each time. The same clustering and evaluation process was then repeated on each subsample. The summarized clustering accuracy (ARI and NMI) on the subsampled datasets suggests scBiG as the overall most robust method, as it had the highest median accuracy and the smallest standard deviation (SD) when considering the average ARI/NMI/SD values across the seven datasets (Figure 3 and Supplementary Figures S7 and S8). Specifically, scBiG ranked first on four of the seven datasets and ranked among the top three in the other three datasets, in terms of the median ARI scores. In summary, the results showed that scBiG exhibited strong representation learning capability and robustness.

Cell embeddings obtained by scBiG improve cell trajectory inference

In addition to cell clustering analysis, we also investigated the cell embeddings' effectiveness in the task of cell trajectory inference, which aims to construct a continuous path that represents dynamic cellular processes rather than dividing cells into discrete clusters. In this analysis, we still considered the nine alternative dimensionality reduction methods used in the clustering analysis. To perform trajectory inference based on cell embeddings, we selected Slingshot (44), a method constructing minimum spanning trees on clustered cells (obtained by the Louvain method in Scanpy) for lineage reconstruction. In a previous benchmark study, Slingshot achieved a high overall score, performing well across four evaluation criteria (45). In this analysis, we considered one simulated dataset and two real scRNA-seq datasets: human embryo and mouse embryo (see the 'Materials and methods' section). To quantitatively assess the similarity between computationally inferred pseudotime orders and true time points, we used the pseudotime ordering score (POS) (46) and Kendall's rank correlation coefficient (Supplementary Methods), both of which take values between −1 and 1.

On the simulated dataset, five methods achieved a POS above 0.9, including scBiG (0.99), ZIFA (0.99), scVI (0.99), Seurat (0.95) and SIMBA (0.93). As for the rank correlation coefficient, the best five methods were scBiG (0.87), scVI (0.87), ZIFA (0.87), SIMBA (0.81) and Seurat (0.81). The top-performing methods were able to reconstruct the cell trajectory reflecting the correct order of the five cell states (Figure 4A and Supplementary Figure S9). The human embryo dataset consisted of embryonic cells at crucial stages of preimplantation development, including zygote, 2-cell, 4-cell, 8-cell, 16-cell and blastocyst stages (Figure 4B and Supplementary Figure S10). Five methods, including scBiG, achieved a POS close to 1. In addition, the trajectory inferred from scBiG's cell embeddings reflected the continuous path of developmental stages (POS = 0.89, Cor = 1). The mouse embryo dataset consisted of embryonic cells at ten different developmental stages. scBiG (POS = 0.92, Cor = 0.76) and Seurat (POS = 0.92, Cor = 0.75) led to the highest accuracy on this dataset (Supplementary Figure S11). Overall, the cell trajectories inferred by scBiG exhibited good agreements with ground truth labels, suggesting that its cell embeddings could effectively capture continuous cell states. These findings highlight the efficacy of scBiG in inferring developmental trajectories from scRNA-seq data, providing valuable insights into dynamic cellular differentiation processes.

scBiG achieves reliable reconstruction of gene expression data

As scBiG is able to reconstruct gene expression levels via its decoder, we evaluated its performance in this task by applying scBiG and five alternative methods (MAGIC, scImpute, ALRA, DCA and scVI) to four simulated datasets with varying sparsity levels (see the 'Materials and methods' section). We used the root mean square error (RMSE) and Spearman's rank correlation coefficient (Spearman's ρ) as evaluation metrics to compare the ground truth expression levels in the simulation and the reconstructed values inferred by the computational methods (Supplementary Methods).

The numerical results showed that scBiG achieved the lowest RMSE across all datasets. It was also among the top three methods in terms of the Spearman's ρ correlation (Figure 5). Another two methods, MAGIC and DCA, had similar Spearman's ρ compared with scBiG. The DCA method also led to a comparable RMSE when the sparsity level of the data was between 80% and 90%, but had higher RMSE than scBiG when the sparsity level reached 95%. The MAGIC method led to much larger RMSE than DCA and scBiG. Another observation is that even though not all methods led to an RMSE better than the observed data, gene expression reconstruction always improved Spearman's ρ, especially when the sparsity level in the observed data was high. In summary, the above results demonstrated the effectiveness of scBiG in recovering gene expression signals from sparse single-cell data.

Gene embeddings obtained by scBiG capture cell type specificity

Gene co-expression analysis serves as a powerful tool for unraveling gene functions and regulatory mechanisms. Prior research has demonstrated that feature genes extracted through PCA can effectively represent network modules within co-expression networks (47). Building upon this insight, we hypothesize that the gene embeddings learned by the scBiG model not only encapsulate a wide spectrum of features within gene expression data but also encapsulate distinct gene modules, which reflect subpopulations of interconnected genes. To further enhance gene module discovery, we extended the

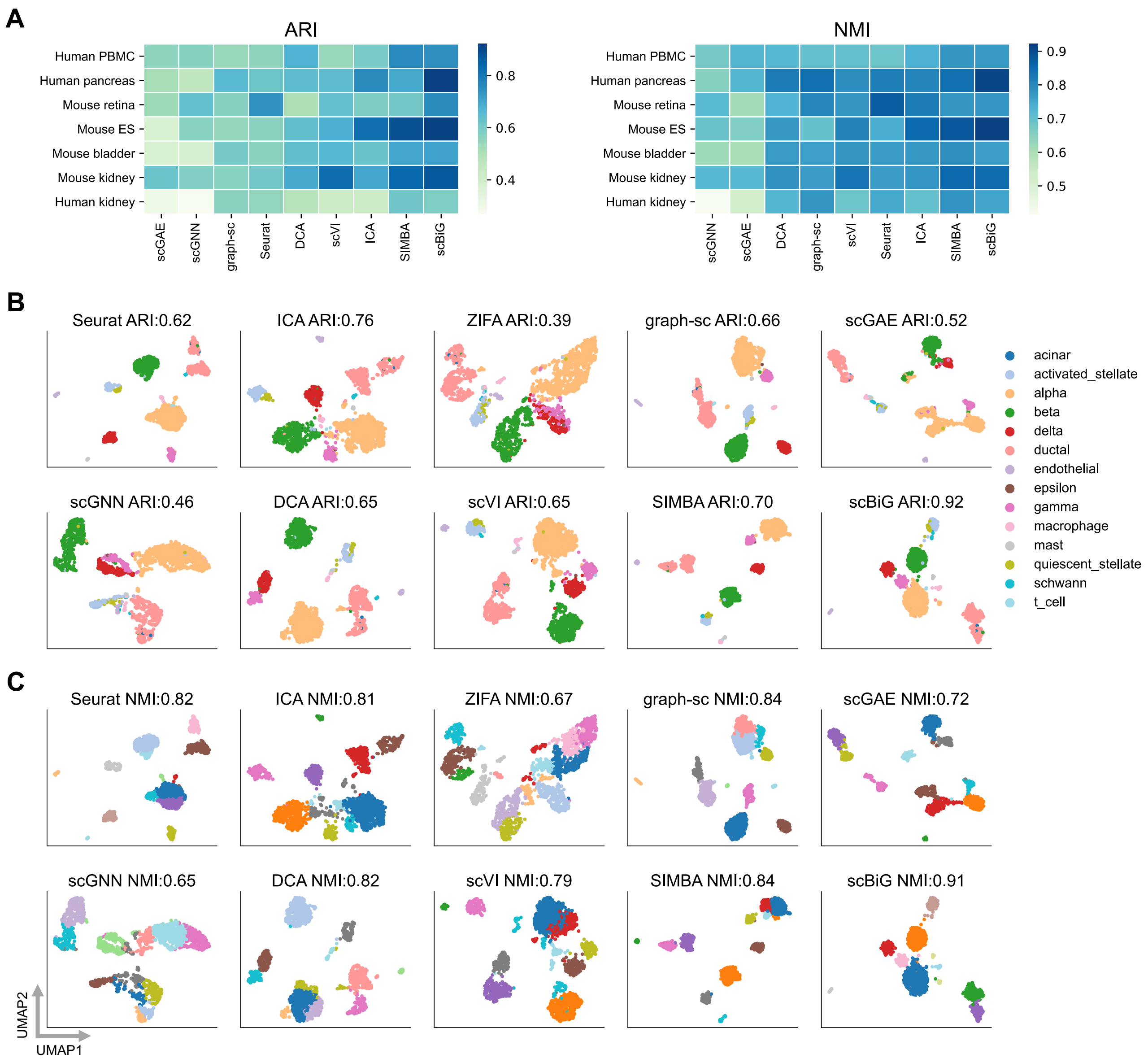

Figure 2. Comparison of different dimensionality reduction methods in clustering analysis. (**A**) ARI and NMI values of nine methods on seven real datasets. The methods are ordered based on the average ARI/NMI values. ZIFA is not included in the heatmaps because it did not finish running within 12 h on the three largest datasets. (**B**) UMAP visualizations of the human pancreas dataset based on the 10 dimensionality reduction methods. Cells are visualized by the annotated cell types. (**C**) Same UMAP visualizations as shown in panel (B) but visualized by the inferred cluster labels.

scBiG model by considering gene–gene correlations. Specifically, edges were added between highly correlated genes in the cell–gene bipartite graph before training the network (Supplementary Methods). This extension was only used in the co-expression analysis since our comparative analysis had shown that cell clustering performance was not improved by incorporating the gene–gene edges (Supplementary Table S4).

In this analysis, we compared scBiG with siVAE (16) and SIMBA (15), which can also learn gene embeddings from scRNA-seq data. In addition, we tried a naive approach by directly applying PCA to the normalized gene expression matrices to obtain gene embeddings. For all methods, we only used the top 1000 highly variable genes (HVGs) selected by Scanpy in the modeling process. We investigated the gene embeddings learned by different methods on two real datasets: human liver (39) and mouse trachea (40). For each dataset, we collected known cell-type-specific marker genes. Embeddings of marker genes for the same cell type were expected to be more similar. The human liver dataset included four cell types, and the marker genes were obtained from the cell type signature gene sets in MSigDB (gene sets annotated with 'Aizarani liver') (48). The mouse trachea dataset included six cell types, and the marker genes were obtained from the CellMarker (49) database.

For the human liver dataset, there were 211 marker genes in the top 1000 HVGs. We performed PCA on the learned embeddings of these 211 genes (Figure 6A). Based on the gene embeddings learned from scBiG (with gene–gene edges), marker genes of the same cell type tended to be closer to each other, while marker genes of different cell types were more separated in the PCA plot. In contrast, the grouping patterns of marker genes were less obvious based on gene embeddings

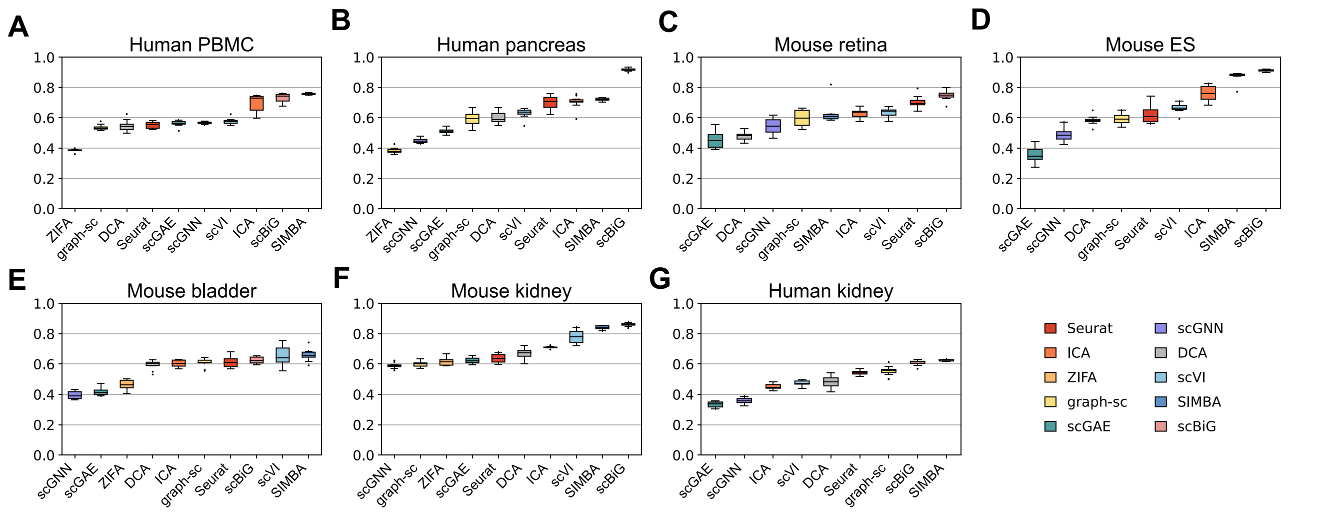

Figure 3. Comparison of different dimensionality reduction methods on subsampled datasets. Boxplots of ARI values were obtained by applying the ten dimensionality reduction methods to subsamples of scRNA-seq datasets: (**A**) human PBMC; (**B**) human pancreas; (**C**) mouse retina; (**D**) mouse ES; (**E**) mouse bladder; (**F**) mouse kidney; and (**G**) human kidney.

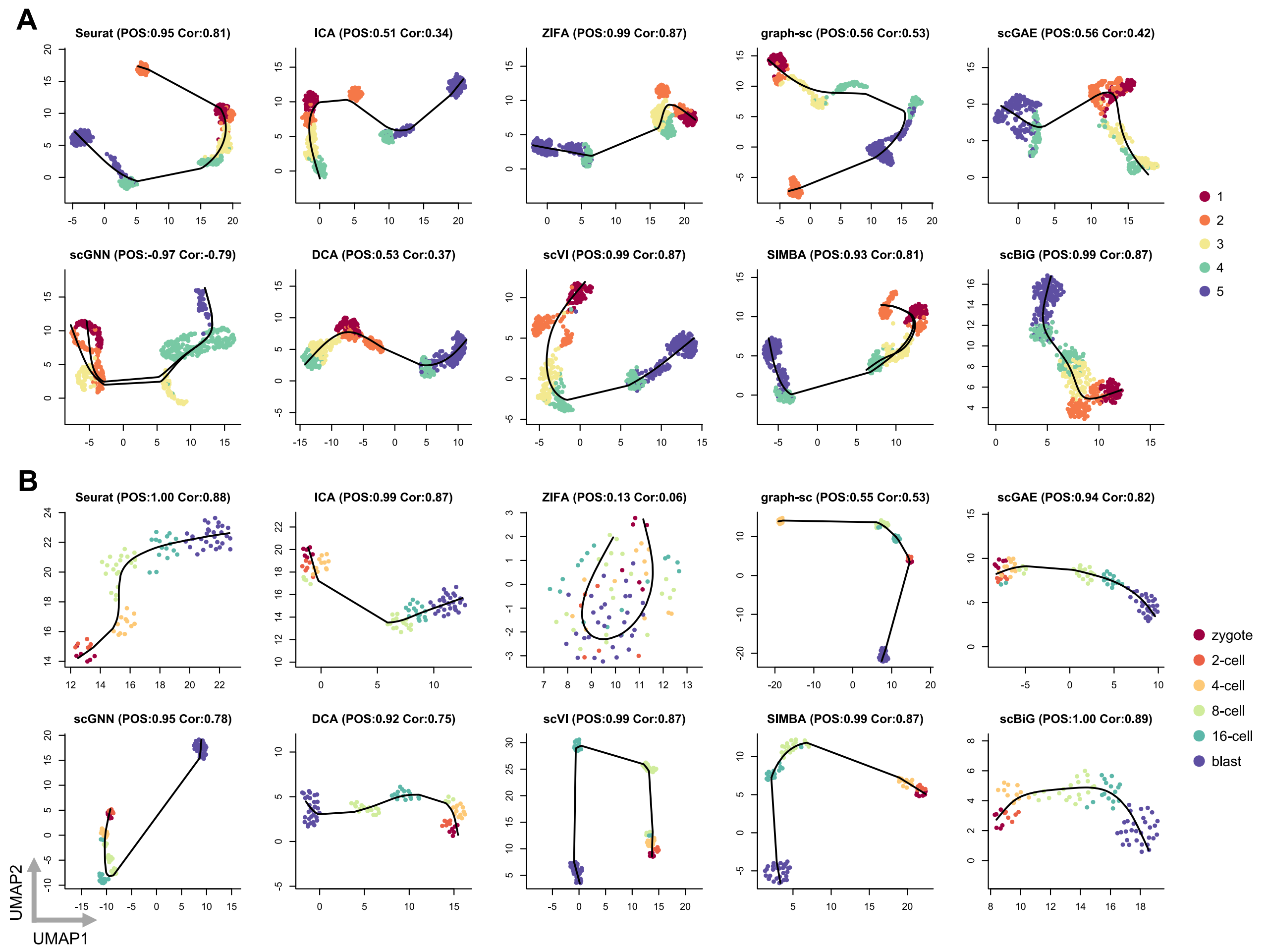

Figure 4. Cell trajectories inferred based on different dimensionality reduction methods. The curves represent the inferred cell trajectories by Slingshot based on the obtained cell embeddings. (**A**) Results on the simulated dataset. The ground truth structure in the simulated data is a curved branching structure with five branches, and each branch represents a developmental state. Cells are visualized by the developmental states. (**B**) Results on the human embryo dataset. Cells are visualized by their developmental stages.

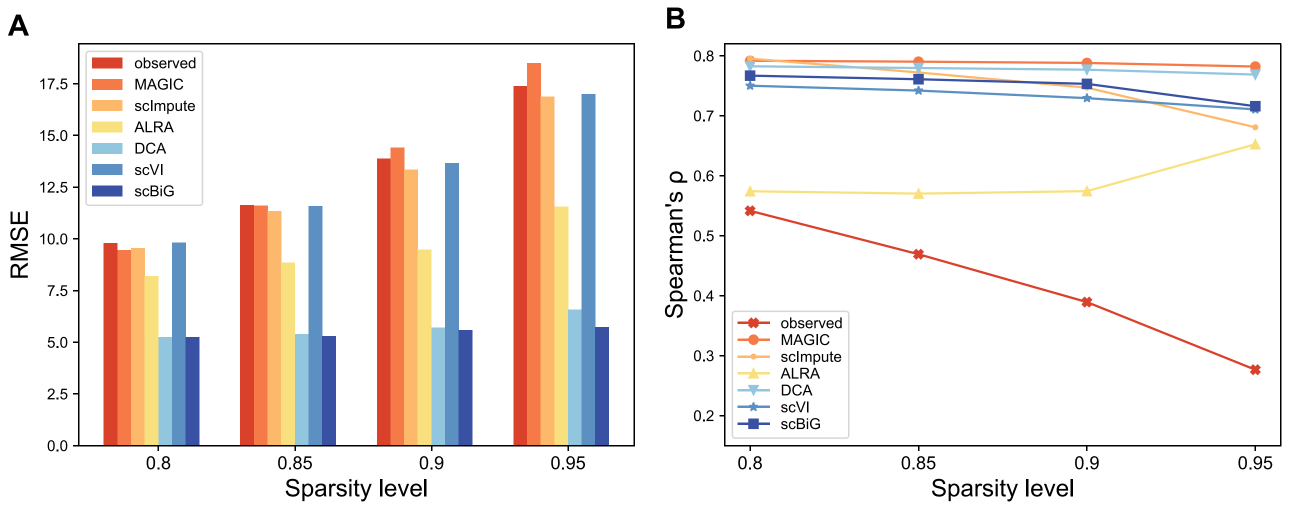

Figure 5. Accuracy of gene expression reconstruction by scBiG and alternative methods on simulated datasets with four sparsity levels. 'Observed' represents the result based on observed counts without reconstruction. (**A**) Accuracy based on the RMSE. (**B**) Accuracy based on the Spearman's rank correlation coefficient (Spearman's ρ).

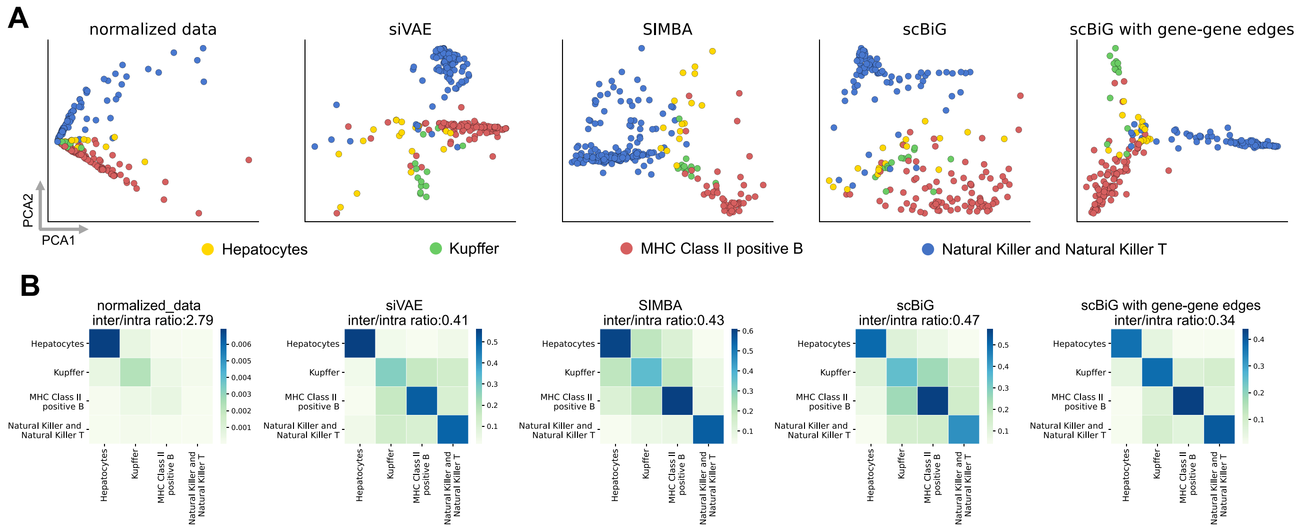

Figure 6. Comparison of marker gene embeddings on the human liver dataset. (**A**) PCA plots based on gene embeddings of normalized data, siVAE, SIMBA, scBiG and scBiG with or without gene–gene edges. Dots represent marker genes and genes are visualized by cell types. (**B**) Similarities between marker gene embeddings. For each method, the diagonal values indicate the average similarity between marker genes of the same cell type (intra-cell-type similarity). The off-diagonal values indicate the average similarity between marker genes of two different cell types (inter-cell-type similarity). The inter/intra ratio is used to measure the relative difference between inter- and intra-cell-type similarities (Supplementary Methods).

obtained by the other methods. We then calculated the similarities between the marker gene embeddings. scBiG (with gene–gene edges) led to relatively large intra-cell-type similarities and low inter-cell-type similarities with the lowest ratio between inter- and intra-cell-type similarities (Figure 6B and Supplementary Methods).

For the mouse trachea dataset, there were 410 marker genes in the top 1000 HVGs, corresponding to the six different cell types in this dataset. The PCA plots of gene embeddings showed that marker genes of the same cell type had stronger clustering patterns in the results of scBiG and SIMBA (Supplementary Figure S12). scBiG with gene–gene edges also achieved more distinct marker gene separation than the version without gene–gene edges. In addition, SIMBA and scBiG (with gene–gene edges) had the lowest ratio between intra- and inter-cell-type similarities. Together, these results suggested scBiG's ability to capture biologically meaningful features of marker genes using its graph convolutional network.

Ablation and sensitivity analyses

The scBiG model consists of two main modules: a GNN-based encoding module and a ZINB-based decoding module (see the 'Materials and methods' section). In the context of cell clustering analysis, we have performed a series of comparative analyses to validate the effectiveness of each module in scBiG. Specifically, we designed analyses to (i) investigate the impact of layer weights in Equation (7), (ii) study the performance of varying combinations of decoder models and (iii) examine the impact of varying numbers of HVGs (Table 1).

Table 1. Clustering performance (ARI) of different scBiG variants

Methods \ Datasets	Human PBMC	Mouse ES	Mouse bladder	Human kidney	Mouse kidney	Human pancreas
scBiG	**0.732 (0.033)**	**0.912 (0.006)**	**0.626 (0.023)**	0.609 (0.018)	**0.860 (0.011)**	0.918 (0.010)
scBiG-$\boldsymbol{E}_c^{(0)}$	0.726 (0.027)	0.709 (0.109)	0.583 (0.037)	0.607 (0.025)	0.852 (0.016)	0.908 (0.015)
scBiG-$\boldsymbol{E}_c^{(2)}$	0.235 (0.012)	0.138 (0.015)	0.278 (0.012)	0.183 (0.013)	0.311 (0.020)	0.276 (0.019)
DOT + MSE	0.534 (0.052)	0.675 (0.063)	0.340 (0.048)	0.621 (0.010)	0.537 (0.040)	0.648 (0.041)
GMF + MSE	0.634 (0.017)	0.882 (0.043)	0.531 (0.035)	0.389 (0.213)	0.814 (0.025)	0.874 (0.022)
scBiG (2000 HVGs)	0.710 (0.037)	0.793 (0.041)	0.520 (0.058)	0.664 (0.024)	0.815 (0.026)	**0.928 (0.016)**
scBiG (3000 HVGs)	0.718 (0.027)	0.827 (0.048)	0.475 (0.066)	0.675 (0.016)	0.850 (0.030)	0.919 (0.007)
scBiG (5000 HVGs)	0.716 (0.028)	0.844 (0.033)	0.519 (0.053)	**0.699 (0.019)**	0.846 (0.029)	0.924 (0.006)

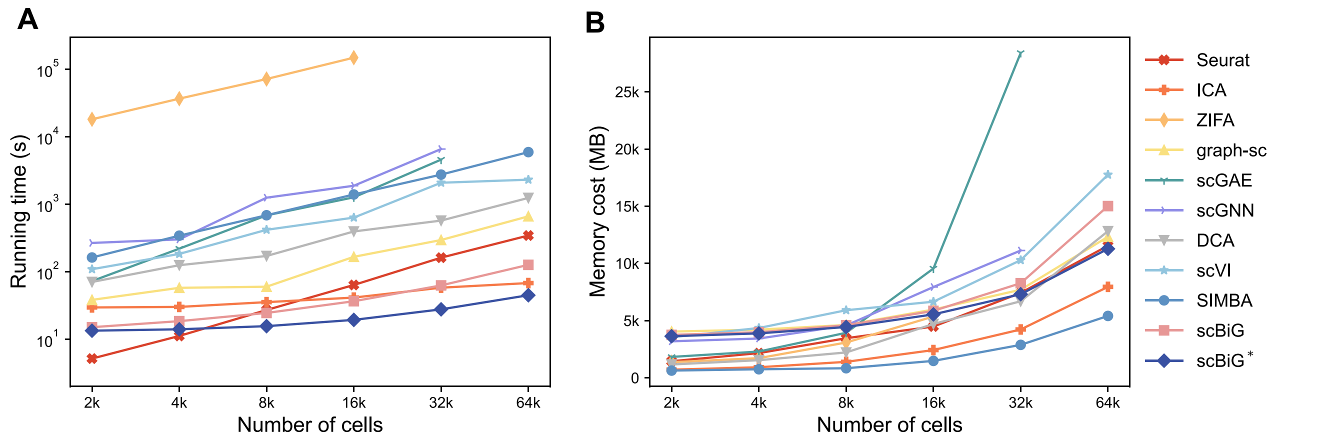

Figure 7. Computational efficiency of scBiG and alternative methods on simulated data with different cell numbers: (**A**) running time and (**B**) maximum memory usage.

The encoder of scBiG consists of two graph convolutional layers, and the final cell embeddings are obtained as a weighted combination of the initial embeddings ($\boldsymbol{E}_c^{(0)}$), the embeddings of the first layer ($\boldsymbol{E}_c^{(1)}$) and the embeddings of the last layer ($\boldsymbol{E}_c^{(2)}$). To examine the effectiveness of the setting in scBiG ($\boldsymbol{E}_c = (1/2)\boldsymbol{E}_c^{(0)} + (1/4)\boldsymbol{E}_c^{(1)} + (1/4)\boldsymbol{E}_c^{(2)}$), we compared it with two variants: directly using initial cell embeddings as final embeddings (denoted as scBiG-$\boldsymbol{E}_c^{(0)}$) and directly using embeddings in the last layer as final embeddings (denoted as scBiG-$\boldsymbol{E}_c^{(2)}$). The results indicated that scBiG achieved the best overall clustering performance (Table 1). Only using the last layer's cell embeddings resulted in poor performance, which is likely due to oversmoothing, while the weighted multilayer combination effectively prevented it. For the decoder layer, scBiG uses a combination of GMF and ZINB models. We considered two different variants of the decoders: one using a dot product decoder with MSE loss (DOT + MSE) and one using a GMF decoder with MSE loss (GMF + MSE) (Supplementary Methods). The results showed that scBiG achieved higher ARI scores than the two variants. With the combination of GMF and ZINB models, scBiG could enhance the representation learning of scRNA-seq data. Lastly, as selecting HVGs is a common optional pre-processing step in scRNA-seq analysis, we conducted an analysis to compare the effect of using 2000, 3000 and 5000 HVGs versus all genes in the cell–gene bipartite graph. Our results indicate that scBiG is not sensitive to HVG selection.

Runtime and memory usage

We compared the computational scalability and efficiency of scBiG and alternative methods by measuring their running time and memory usage (Figure 7). These experiments were conducted on the NVIDIA Tesla V100S-PCIE (32G). Information about the GPU usage of each method is summarized in Supplementary Table S2. scBiG demonstrated a near-linear increase in running time concerning the number of cells. Specifically, scBiG ranked as the second fastest method when dealing with datasets containing 4000 cells or fewer, and it took the second position behind ICA when the cell count exceeds 32 000. Regarding maximum memory usage, scBiG fell within the mid-range and was comparable in magnitude to other methods, including DCA and graph-sc. When utilizing 2000 HVGs, denoted as scBiG*, further reductions in both running time and memory usage could be achieved. However, on the same datasets (without selecting HVGs), ZIFA was not able to process 32 000 cells due to memory overflow issues. scGNN and scGAE were unable to process 64 000 cells for the same reason.

Discussion

In this article, we introduce scBiG, a novel graph representation learning method to simultaneously learn cell and gene embeddings. scBiG's strength lies in its approach of constructing a bipartite graph that connects cells and genes, and utilizing a graph autoencoder for encoding and decoding. This process enables representation learning for both cells and genes, as well as reconstruction of gene expression levels. The encoder within scBiG incorporates multiple graph convolution layers, facilitating the learning and extraction of higher-order features from cell and gene nodes. Meanwhile, the decoder employs a ZINB distribution model to guide training and reconstruct gene expression levels. Through self-supervised learning, scBiG effectively learns lower-dimensional embeddings that prove valuable for various downstream tasks, including cell clustering, cell trajectory inference, gene expression reconstruction and gene co-expression analysis.

Our simulation and real data studies have demonstrated that the cell embeddings learned by scBiG capture meaningful cell–cell relationships, and can improve cell clustering analysis and cell trajectory inference, compared with multiple alternative methods. Furthermore, by taking gene correlations into consideration when constructing the bipartite graph, scBiG enhances representation learning and module identification in gene co-expression analysis. scBiG is also shown to more accurately recover gene expression levels, and it exhibits remarkable robustness to data sparsity. Notably, scBiG serves as a robust deep learning framework for scRNA-seq datasets, and it presents a promising avenue for advancing graph representation learning in the field of single-cell genomics.

In our future work, we will consider improving the design of the network architecture, so that it can consider cell-type-specific and common components in the gene embeddings, making downstream comparison more straightforward. Additionally, we plan to extend the utility of scBiG to more general settings, by including additional information such as batches, gene sequences and protein–protein interactions to expand the bipartite graph. This expansion will further enrich our ability to comprehensively learn representations of both cells and genes, thereby enhancing scBiG's capacity to unveil complex biological processes.

Data availability

scBiG is implemented as a Python package, which is freely available from its GitHub repository (https://github.com/sldyns/scBiG). The accession numbers of all real data used in this work are described in Supplementary Table S3. The data and code used to reproduce the analyses are available at https://github.com/sldyns/scBiG/tree/main/reproducibility and https://doi.org/10.5281/zenodo.10460509.

Supplementary data

Supplementary Data are available at NARGAB Online.

Acknowledgements

We thank members of the Wei Vivian Li and Hongwei Li labs for the helpful discussions and suggestions.

Funding

National Natural Science Foundation of China [42274172 to H.L.]; National Institute of General Medical Sciences [R35GM142702 to W.V.L.].

Conflict of interest statement

None declared.

References

1. Papalexi,E. and Satija,R. (2018) Single-cell RNA sequencing to explore immune cell heterogeneity. *Nat. Rev. Immunol.*, **18**, 35–45.
2. Xiang,R., Wang,W., Yang,L., Wang,S., Xu,C. and Chen,X. (2021) A comparison for dimensionality reduction methods of single-cell RNA-seq data. *Front. Genet.*, **12**, 646936.
3. Sun,S., Zhu,J., Ma,Y. and Zhou,X. (2019) Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis. *Genome Biol.*, **20**, 1–21.
4. Bro,R. and Smilde,A.K. (2014) Principal component analysis. *Anal. Methods*, **6**, 2812–2831.
5. Liebermeister,W. (2002) Linear modes of gene expression determined by independent component analysis. *Bioinformatics*, **18**, 51–60.
6. Pierson,E. and Yau,C. (2015) ZIFA: dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biol.*, **16**, 1–10.
7. 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. *Nat. Commun.*, **10**, 390.
8. Amodio,M., Van Dijk,D., Srinivasan,K., Chen,W.S., Mohsen,H., Moon,K.R., Campbell,A., Zhao,Y., Wang,X., Venkataswamy,M., *et al.* (2019) Exploring single-cell data with deep multitasking neural networks. *Nat. Methods*, **16**, 1139–1145.
9. Lopez,R., Regier,J., Cole,M.B., Jordan,M.I. and Yosef,N. (2018) Deep generative modeling for single-cell transcriptomics. *Nat. Methods*, **15**, 1053–1058.
10. Luo,Z., Xu,C., Zhang,Z. and Jin,W. (2021) A topology-preserving dimensionality reduction method for single-cell RNA-seq data using graph autoencoder. *Sci. Rep.*, **11**, 20028.
11. Goyal,P. and Ferrara,E. (2018) Graph embedding techniques, applications, and performance: a survey. *Knowl.-Based Syst.*, **151**, 78–94.
12. Wu,S., Sun,F., Zhang,W., Xie,X. and Cui,B. (2022) Graph neural networks in recommender systems: a survey. *ACM Comput. Surv.*, **55**, 1–37.
13. 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. *Nat. Commun.*, **12**, 1882.
14. Ciortan,M. and Defrance,M. (2022) GNN-based embedding for clustering scRNA-seq data. *Bioinformatics*, **38**, 1037–1044.
15. Chen,H., Ryu,J., Vinyard,M.E., Lerer,A. and Pinello,L. (2023) SIMBA: single-cell embedding along with features. *Nat. Methods*, https://doi.org/10.1038/s41592-023-01899-8.
16. Choi,Y., Li,R. and Quon,G. (2023) siVAE: interpretable deep generative models for single-cell transcriptomes. *Genome Biol.*, **24**, 29.
17. Hetzel,L., Fischer,D.S., Günnemann,S. and Theis,F.J. (2021) Graph representation learning for single-cell biology. *Curr. Opin. Syst. Biol.*, **28**, 100347.
18. Kipf,T.N. and Welling,M. (2017) Semi-supervised classification with graph convolutional networks. In: *International Conference on Learning Representations*, Toulon, France.
19. He,X., Deng,K., Wang,X., Li,Y., Zhang,Y. and Wang,M. (2020) LightGCN: simplifying and powering graph convolution network for recommendation. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xi'an, China*. ACM, New York, NY.
20. Li,Q., Han,Z. and Wu,X. (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA*. AAAI Press, Washington, DC.
21. He,X., Liao,L., Zhang,H., Nie,L., Hu,X. and Chua,T. (2017) Neural collaborative filtering. In: *Proceedings of the 26th International Conference on World Wide Web,Perth, Australia*. International World Wide Web Conferences Steering Committee, Geneva,Switzerland, pp. 173–182.
22. Wolf,F.A., Angerer,P. and Theis,F.J. (2018) SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.*, **19**, 15.

23. Ketkar,N. and Moolayil,J. (2021) Introduction to PyTorch. In: *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch*. Apress, New York,NY, pp. 27–91.
24. Kingma,D.P. and Ba,J. (2015) Adam: a method for stochastic optimization. In: *International Conference on Learning Representations*, San Diego, CA.
25. Satija,R., Farrell,J.A., Gennert,D., Schier,A.F. and Regev,A. (2015) Spatial reconstruction of single-cell gene expression data. *Nat. Biotechnol.*, **33**, 495–502.
26. 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**, 716–729.
27. Li,W.V. and Li,J.J. (2018) An accurate and robust imputation method scImpute for single-cell RNA-seq data. *Nat. Commun.*, **9**, 997.
28. Linderman,G.C., Zhao,J. and Kluger,Y. (2022) Zero-preserving imputation of scRNA-seq data using low-rank approximation. *Nat. Commun.*, **9**, 192.
29. Baron,M., Veres,A., Wolock,S.L., Faust,A.L., Gaujoux,R., Vetere,A., Ryu,J.H., Wagner,B.K., Shen-Orr,S.S., Klein,A.M., *et al.* (2016) A single-cell transcriptomic map of the human and mouse pancreas reveals inter- and intra-cell population structure. *Cell Syst.*, **3**, 346–360.
30. Klein,A.M., Mazutis,L., Akartuna,I., Tallapragada,N., Veres,A., Li,V., Peshkin,L., Weitz,D.A. and Kirschner,M.W. (2015) Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, **161**, 1187–1201.
31. Han,X., Wang,R., Zhou,Y., Fei,L., Sun,H., Lai,S., Saadatpour,A., Zhou,Z., Chen,H., Ye,F., *et al.* (2018) Mapping the mouse cell atlas by Microwell-seq. *Cell*, **172**, 1091–1107.
32. Adam,M., Potter,A.S. and Potter,S.S. (2017) Psychrophilic proteases dramatically reduce single-cell RNA-seq artifacts: a molecular atlas of kidney development. *Development*, **144**, 3625–3632.
33. Zheng,G.X., Terry,J.M., Belgrader,P., Ryvkin,P., Bent,Z.W., Wilson,R., Ziraldo,S.B., Wheeler,T.D., McDermott,G.P., Zhu,J., *et al.* (2017) Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.*, **8**, 14049.
34. Young,M.D., Mitchell,T.J., Vieira Braga,F.A., Tran,M.G., Stewart,B.J., Ferdinand,J.R., Collord,G., Botting,R.A., Popescu,D.-M., Loudon,K.W., *et al.* (2018) Single-cell transcriptomes from human kidneys reveal the cellular identity of renal tumors. *Science*, **361**, 594–599.
35. Macosko,E.Z., Basu,A., Satija,R., Nemesh,J., Shekhar,K., Goldman,M., Tirosh,I., Bialas,A.R., Kamitaki,N., Martersteck,E.M., *et al.* (2015) Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, **161**, 1202–1214.
36. Yan,L., Yang,M., Guo,H., Yang,L., Wu,J., Li,R., Liu,P., Lian,Y., Zheng,X., Yan,J., *et al.* (2013) Single-cell RNA-Seq profiling of human preimplantation embryos and embryonic stem cells. *Nat. Struct. Mol. Biol.*, **20**, 1131–1139.
37. Deng,Q., Ramsköld,D., Reinius,B. and Sandberg,R. (2014) Single-cell RNA-seq reveals dynamic, random monoallelic gene expression in mammalian cells. *Science*, **343**, 193–196.
38. Papadopoulos,N., Gonzalo,P.R. and Söding,J. (2019) PROSSTT: probabilistic simulation of single-cell RNA-seq data for complex differentiation processes. *Bioinformatics*, **35**, 3517–3519.
39. Popescu,D.-M., Botting,R.A., Stephenson,E., Green,K., Webb,S., Jardine,L., Calderbank,E.F., Polanski,K., Goh,I., Efremova,M., *et al.* (2019) Decoding human fetal liver haematopoiesis. *Nature*, **574**, 365–371.
40. Schaum,N., Karkanias,J., Neff,N.F., May,A.P., Quake,S.R., Wyss-Coray,T., Darmanis,S., Batson,J., Botvinnik,O., Chen,M.B., *et al.* (2018) Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris: The Tabula Muris Consortium. *Nature*, **562**, 367–372.
41. Zappia,L., Phipson,B. and Oshlack,A. (2017) Splatter: simulation of single-cell RNA sequencing data. *Genome Biol.*, **18**, 174.
42. Blondel,V.D., Guillaume,J.-L., Lambiotte,R. and Lefebvre,E. (2008) Fast unfolding of communities in large networks. *J. Stat. Mech. Theor. Exp.*, **2008**, P10008.
43. McInnes,L., Healy,J., Saul,N. and Großberger,L. (2018) UMAP: Uniform Manifold Approximation and Projection. *J. Open Source Softw.*, **3**, 861.
44. 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**, 477.
45. Saelens,W., Cannoodt,R., Todorov,H. and Saeys,Y. (2019) A comparison of single-cell trajectory inference methods. *Nat. Biotechnol.*, **37**, 547–554.
46. Ji,Z. and Ji,H. (2016) TSCAN: pseudo-time reconstruction and evaluation in single-cell RNA-seq analysis. *Nucleic Acids Res.*, **44**, e117.
47. Langfelder,P. and Horvath,S. (2007) Eigengene networks for studying the relationships between co-expression modules. *BMC Syst. Biol.*, **1**, 1–17.
48. Liberzon,A., Subramanian,A., Pinchback,R., Thorvaldsdóttir,H., Tamayo,P. and Mesirov,J.P. (2011) Molecular Signatures Database (MSigDB) 3.0. *Bioinformatics*, **27**, 1739–1740.
49. Zhang,X., Lan,Y., Xu,J., Quan,F., Zhao,E., Deng,C., Luo,T., Xu,L., Liao,G., Yan,M., *et al.* (2019) CellMarker: a manually curated resource of cell markers in human and mouse. *Nucleic Acids Res.*, **47**, D721–D728.

Received: October 19, 2023. **Revised:** December 19, 2023. **Editorial Decision:** January 3, 2024. **Accepted:** January 9, 2024