R-SCGNN: ENHANCING GRAPH AUTOENCODERS FOR IMPROVED CLUSTERING IN SCRNA-SEQ ANALYSIS

Shyaman Jayasundara
sjayasun@purdue.edu
PUID: 34770287

1 INTRODUCTION

Single-cell RNA-sequencing (scRNA-seq) enables transcriptome-wide gene expression measurement in individual cells, which has revolutionized our understanding of cellular heterogeneity in complex diseases. However, scRNA-seq analysis remains challenging due to the complexity and undetermined nature of the data distribution, which has a large volume and a high rate of dropout events. To address this challenge, the emerging field of graph neural networks (GNNs) has provided a solution by deconvoluting node (i.e., cell) relationships in a graph through neighbor information propagation in a deep learning architecture.

scGNN (Wang et al., 2021), is a multi-modal iterative framework for modeling heterogeneous cell-cell relationships and cell clustering on scRNA-Seq data. The scGNN architecture consists of three main components: the feature autoencoder, the graph autoencoder, and the cluster autoencoder. The graph autoencoder is used to learn the low-dimensional representation of the graph topology and train node relationships from a global view of the whole graph.

scGNN uses a vanilla Graph Autoencoder (GAE) (Kipf & Welling, 2016b) to learn a low-dimensional representation of the scRNA-Seq graph topology and train node relationships. However, this approach, which is designed for general-purpose graph tasks, may not be optimal for clustering tasks. To improve scGNN's performance, it is necessary to modify the objective function of GAE to focus more on clustering-specific graph construction objectives. This modification will enable the effective modeling of complex relationships between different cells and facilitate accurate cell clustering based on gene expression patterns. To improve the scGNN model's performance, a joint clustering and embedded learning objective can be incorporated into the graph autoencoder architecture. By doing so, the model can optimize the low-dimensional embeddings while also enhancing the clustering accuracy. This is achieved by minimizing the difference between predicted and true clustering assignments through the joint objective function.

However, incorporating a joint objective can still result in two significant problems: Feature Randomness (FR) and Feature Drift (FD) (Mrabah et al., 2020). FR arises when the learned embeddings capture non-representative features, which can negatively impact the latent structure of the data. FD, on the other hand, occurs when the learned embeddings become less relevant to the clustering objective over time due to the optimization process. To overcome these issues, a recent study (Mrabah et al., 2022) proposes two solutions: 1) The use of a denoising mechanism to remove irrelevant features, and 2) The introduction of a regularization term that encourages the learned embeddings to remain aligned with the clustering objective during the optimization process. These modifications ensure that the learned embeddings are both representative and pertinent to the clustering task at hand.

In this study, we present a variant of the scGNN framework that employs a clustering-focused graph construction objective instead of the standard GNN. To tackle the problem of FD and FR, we utilize two approaches introduced by Mrabah et al. (2022). Specifically, we integrate a denoising mechanism to remove irrelevant features and introduce a regularization term to ensure that the learned embeddings remain consistent with the clustering objective throughout the optimization process.

2 DATASET

scRNA-seq analysis is challenging due to its complex data distribution, high dropout rate, and large volume. Some methodologies use KNN graphs to model cell relationships, but this oversimplifies complex cell and gene relationships. Hence, we assessed the capability of GNNs in capturing complex relations in comparison to existing methods by evaluating them on four benchmark datasets. The used datasets are namely Chung (Chung et al., 2017), Klein (Klein et al., 2015), Zeisel (Zeisel et al., 2015), and Kolodziejczyk (Kolodziejczyk et al., 2015). Each dataset D_i has a gene expression matrix $X_i \in \mathbb{R}^{N_i \times M_i}$, where N_i is the number of cells (samples) and M_i is the number of genes (variables). Thus, each row of X_i represents the gene expression profile of a single cell, and each column represents the expression level of a single gene across all cells. Additionally, each dataset has a vector of cluster labels $Y_i \in 1, 2, \ldots, C_i$, where C_i is the number of cell types in the dataset.

3 METHODOLOGY

3.1 PRELIMINARIES

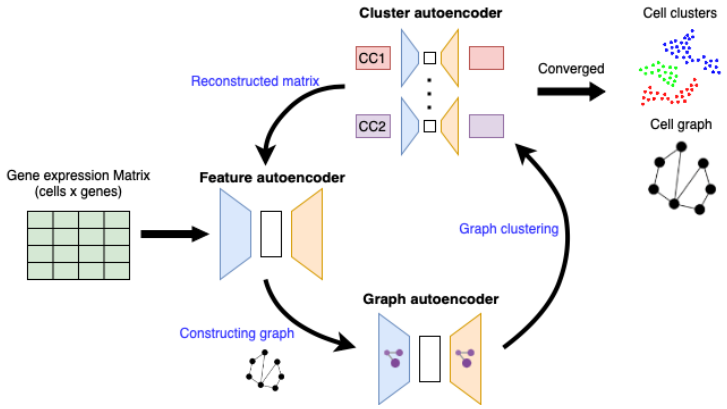


Figure 1: The scGNN architecture

The scGNN framework comprises three key components: feature autoencoder, graph autoencoder, and cluster encoder (Fig. 1). The feature autoencoder is used to learn the representative embedding of the scRNA expression through two stacked dense layers in both the encoder and decoder. The encoder constructs the low-dimensional embedding of X' from the input gene expression X, and the decoder reconstructs the expression $\hat{X}$ from the embedding; thus, $X, \hat{X} \in \mathbb{R}^{N \times M}$ and $X' \in \mathbb{R}^{N \times M'}$, where M is the number of input genes, M' is the dimension of the learned embedding, and $M' < M$. The objective of training the feature autoencoder is to achieve a maximum similarity between the original and reconstructed through minimizing the loss function (MSE). The cluster autoencoder in scGNN has a similar architecture to the feature autoencoder. However, in the cluster autoencoder, each cluster has its own individual autoencoder that is trained to reconstruct gene expression values specifically for that cluster.

In this approach, a graph is created to represent cell-to-cell relationships by utilizing the embedding obtained from the feature autoencoder. For this purpose, a KNN graph is utilized, where each node corresponds to a single cell and edges represent the relationships between them. The weights of the edges are calculated based on the Euclidean distance of the learned embedding vectors. Isolation Forest (Liu et al., 2008) is then used to prune the graph by identifying and disconnecting outliers.

The GAE is designed to learn and represent the topological information of the pruned graph. Given an input pruned graph $G = (V, E)$ with $N = |V|$ nodes representing cells and E representing edges, A is its adjacency matrix, and D is its degree matrix. The node feature matrix of the GAE is the learned embedding X' from the feature autoencoder. The graph convolution network (GCN) is defined as $GCN(X_0, A) = \mathrm{ReLU}(\tilde{A}X_0W)$, where W is a weight matrix learned during training, and $\tilde{A} = D^{-1/2}AD^{-1/2}$ is the symmetrically normalized adjacency matrix with the activation

function $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. The encoder of the GAE consists of two layers of GCN, and the graph embedding Z is learned through the encoder as $Z = \mathrm{ReLU}(\tilde{A}\mathrm{ReLU}(\tilde{A}X_0W_1)W_2)$. The learned weight matrices W_1 and W_2 are the weights in the first and second layers, respectively. The GAE's decoder calculates the inner product of the learned embedding and passed it through a sigmoid activation function to obtain the reconstructed adjacency matrix $\hat{A} = \mathrm{sigmoid}(ZZ^T)$.

The objective of the GAE is to minimize the cross-entropy loss function, which measures the dissimilarity between the input adjacency matrix A and the reconstructed matrix $\hat{A}$:

$$\begin{aligned} L_{\mathrm{GAE}} &= L_{bce}(\hat{A}(Z(\theta)), A) \\ &= -\frac{1}{N^2}\sum_{i,j=1}^{N} a_{ij}\log\left(\hat{a}_{ij}\right) + (1-a_{ij})\log\left(1-\hat{a}_{ij}\right) \end{aligned} \tag{1}$$

Here, a_{ij} and $\hat{a}_{ij}$ are the elements of the adjacency matrix A and $\hat{A}$ in the ith row and jth column. θ refers to the parameters of the GAE model.

Single-cell RNA sequencing data is challenging to analyze due to the high levels of technical noise and dropout events, resulting in sparsity and distortion of the underlying data structure. Therefore, the objective of the iterative process in scGNN is to effectively recover and refine the underlying structure of the data. The iterative process constructs the single-cell graph step by step and runs the above components repeatedly until convergence is achieved. The cell graph can be expressed as follows:

$$\tilde{A} = \lambda L_0 + (1-\lambda)\frac{A_{ij}}{\sum_j A_{ij}},$$

where L_0 is the normalized adjacency matrix of the initial pruned graph, and $L_0 = D_0^{-1/2}A_0D_0^{-1/2}$, where D_0 is the degree matrix. The parameter λ controls the convergence rate, with $\lambda \in [0, 1]$. At each iteration t, two criteria are evaluated to decide whether to terminate the iteration: (1) whether the adjacency matrix has converged, i.e., $\tilde{A}_t - \tilde{A}_{t-1} < \gamma_1\tilde{A}_0$; or (2) whether the inferred cell types are sufficiently similar, i.e., ARI $< \gamma_2$. The final cell-type results are determined by the clustering outcome in the last iteration.

3.2 Reformulating scGNN for clustering-oriented graph construction objective

The scGNN model has a reconstruction loss (Eq. 1) which is a standard self-supervision method for pretraining GAE models. Let $\mathbb{A}_C$ be a clustering algorithm and $P \in \mathbb{R}^{N\times K}$ be the clustering assignment matrix obtained by applying $\mathbb{A}_C$ to latent representation Z. Here the graph $\mathcal{G}$ to be associated with K clusters defined as $\{C_k\}_{k=1}^K$. Then $L_{\text{clus}}\left(P(Z(\theta))\right)$ is the clustering loss associated with algorithm $\mathbb{A}_C$. The scGNN model optimizes the clustering loss $L_{clus}(P(Z(\theta)))$ by finding the optimal solution P^*, which can be expressed as follows:

$$P^* = \arg\min_P L_{clus}(P(Z(\theta))) \tag{2}$$

As scGNN separates the clustering process from the embedding learning process (Eq. 2), it may face challenges in accurately clustering data due to its limited capacity to learn cluster-oriented features. To address this limitation, a modified objective function that prioritizes clustering-oriented graph construction is essential. This would enable the scGNN model to learn cluster-specific features, rather than relying solely on static objective functions during the clustering process. Some GAE-based clustering methods (Hui et al., 2020) employ joint clustering and embedded learning to overcome this challenge. Accordingly, we can reformulate Eq. 2 to enforce clustering assumptions on the embedded representations, as follows:

$$\theta^*, P^* = \arg\min_{\theta,P} L_{\text{clus}}\left(P(Z(\theta))\right), \tag{3}$$

where P^* and θ^* are the optimal solutions.

Considering the reconstruction loss, we can reformulate Eq. 3 as follows:

$$\theta^*, P^* = \arg\min_{\theta,P} L_{\text{clus}}\left(P(Z(\theta))\right) + \gamma L_{bce}\left(\hat{A}(Z(\theta)), A\right), \tag{4}$$

where P^* and θ^* are the optimal solutions and γ is the balancing hyper-parameter that controls the trade-off between clustering and reconstruction.

Typically, clustering losses aim to minimize intra-cluster variance and maximize inter-cluster variance. By optimizing θ, the embedded points are moved to create a clustering-oriented distribution, which reduces the importance of the choice of clustering cost. However, this formulation presents a challenge as the embedded points may shift in a way that violates their semantic categories while still decreasing the embedded clustering penalty. Therefore, pseudo-supervision is needed to determine the semantic categories of the data by constructing pseudo-labels and training the model with a clustering-oriented task objective (Mrabah et al., 2022). These pseudo-labels are usually predicted using a clustering algorithm. During neural network training with pseudo-labels, a phenomenon known as feature randomness (FR) (Mrabah et al., 2020) can occur. In this case, the network may learn features that capture irrelevant similarities, resulting in little to no correlation between the training samples and their true labels. An additional issue that can arise in embedded clustering is Feature Drift (FD) (Mrabah et al., 2020). This occurs when two competing loss functions are optimized concurrently, namely clustering which aims to decrease intra-cluster variance and increase inter-cluster variance, and the reconstruction objective which seeks to maintain all variances, including clustering-irrelevant similarities. As a result, the features learned through embedded clustering can be compromised by the reconstruction cost.

In order to tackle the FR and FD issues, Mrabah et al. (2022) proposed two solutions. The first solution involves the development of a sampling operator Ξ that gradually identifies nodes with reliable clustering assignments, represented by the set Ω, to act as a protection mechanism against FR. The second solution proposes a graph-specific operator Υ that triggers a correction mechanism against FD. To protect against FD, the self-supervision signal A is transformed into a clustering-oriented signal $\Upsilon(A, P(Z(\theta)), \mathcal{V})$ using this formulation:

$$\theta^*, P^* = \arg\min_{\theta,P} L_{clus}(P(\Xi(Z(\theta)))) + \gamma L_{bce}(\hat{A}(Z(\theta)), \Upsilon(A, P(\Xi(Z(\theta))), \Omega)) \tag{5}$$

To initiate the clustering process, assignments for all samples are obtained, but only those of reliable nodes are used in the clustering loss calculation denoted as $L_{clus}(P(\Xi(Z(\theta))))$. This ensures that the clustering loss is optimized solely on reliable nodes. The graph A is then modified using the Υ operation, which alters the sub-graph defined by the set of reliable nodes Ω.

3.3 Reformulation of scGNN graph autoencoder

We begin the reformulation of the scGNN GAE by replacing the graph representation and adjacency matrix. Let us first define the adjacency matrix $A = (a_{ij}) \in R^{N \times N}$, where:

$$a_{ij} = \begin{cases} \frac{1}{|C_k|} & \text{if } \exists k \text{ such that } i, j \in C_k \\ 0 & \text{otherwise.} \end{cases}$$

The scGNN framework's vanilla GAE was replaced with GMM-VGAE (Variational Graph Auto-Encoder with Gaussian Mixture Models) (Hui et al., 2020). GMM-VGAE utilizes Gaussian Mixture Models to capture the variances between various clusters and enables joint clustering and embedding learning. We pretrained the network on adjacency reconstruction for 200 epochs. During the clustering phase, GMM-VGAE minimizes a linear combination of embedded clustering and reconstruction. The model generates latent codes using a multivariate Gaussian distribution that is parameterized by an encoding network. The clustering loss is imposed by imposing a GMM structure on the latent

space to capture the different clusters, as described by the following equation:

$$
\begin{aligned}
L_{clus}(P(Z(\theta))) = & \sum_{i=1}^{N} \sum_{k=1}^{K} p_{ik} \log \left(\frac{\pi_k}{p_{ik}} \right) \\
& - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{K} p_{ik} \left(\log \frac{\left| \operatorname{diag} \left(\sigma_k^2 \right) \right|}{\left| \operatorname{diag} \left(\tilde{\sigma}_i^2 \right) \right|} \right. \\
& + \operatorname{tr} \left(\operatorname{diag}^{-1} \left(\sigma_k^2 \right) \operatorname{diag} \left(\tilde{\sigma}_i^2 \right) \right) \\
& \left. + \left(\tilde{\mu}_i - \mu_k \right)^T \operatorname{diag}^{-1} \left(\sigma_k^2 \right) \left(\tilde{\mu}_i - \mu_k \right) + d \right)
\end{aligned}
$$

Here, we can represent the latent codes generated by the encoding network using a multivariate Gaussian distribution, where the mean vector of the distribution corresponding to z_i is denoted as $\tilde{\mu}_i$, and its variance is captured by $\operatorname{diag}\left(\tilde{\sigma}_i^2\right)$. The latent clusters are represented by K multivariate Gaussian distributions $\left\{\mathcal{N}\left(\mu_k, \operatorname{diag}\left(\sigma_k^2\right)\right)\right\}_{k=1}^{K}$. The dimension of the latent space is d. The probability of cluster assignments is represented by $P = (p_{ik})_{i,k}$, and π_k denotes the frequency of clustering assignments or the probability of cluster k among the other clusters.

The initial step (i.e., pre-train task) in the training process of GMM-VGAE involves initializing the clustering assignments $(p_{ik})_{i,k}$, the clustering centers $\{\mu_k\}_{k=1}^{K}$, and the covariance matrices $\left\{\operatorname{diag}\left(\sigma_k^2\right)\right\}_{k=1}^{K}$ according to GMM. The model is then trained (i.e., main task) to optimize joint clustering and reconstruction by minimizing the following loss function:

$$
L_{\mathrm{GMM-VGAE}} = L_{clus}(P(Z(\theta))) + L_{bce}(\hat{A}(Z(\theta)), A)
$$

In order to address the issues of FR and FD, the operator Ξ is used to gradually identify reliable nodes by utilizing the clustering assignments P. The set of reliable nodes, denoted by Ω, is constructed in a step-by-step manner. The clustering loss is then computed only on this set of high-confidence samples. Furthermore, to convert the reconstruction objective into a more clustering-oriented one, the operator Υ is utilized gradually. The resulting objective function can be expressed as follows:

$$
\begin{aligned}
L_{\mathrm{R-GMM-VGAE}} = & L_{\text{clus}}\left(P(\Xi(Z(\theta)))\right) \\
& + L_{bce}(\hat{A}(Z(\theta)), \Upsilon(A, P(\Xi(Z(\theta))), \Omega))
\end{aligned}
$$

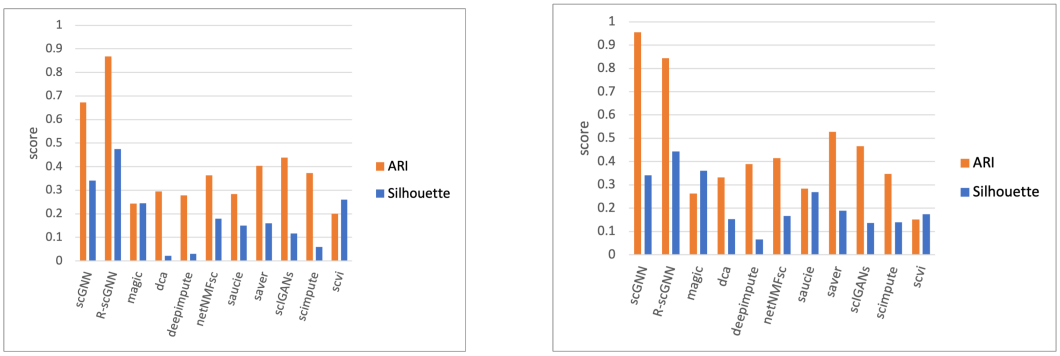
4 RELATED WORK

Single-cell RNA-sequencing (scRNA-seq) techniques have enabled transcriptome-wide gene expression measurements in individual cells, which are crucial for identifying cell-type clusters, inferring the arrangement of cell populations based on trajectory topologies, and characterizing cellular heterogeneity in complex diseases. However, scRNA-seq analysis for biological inference remains challenging due to the complex and undetermined data distribution, large volume, and high rate of dropout events. To address this, some pioneer methodologies such as Phenograph (Levine et al., 2015), MAGIC (Van Dijk et al., 2018), and Seurat (Butler et al., 2018) have used a k-nearest-neighbor (KNN) graph to model cell relationships. However, these representation may oversimplify the complex relationships between cells and genes in the global cell population.

Recently, the GNNs have emerged as a promising solution for deciphering node relationships in a graph by propagating neighbor information through a deep learning architecture (Wang et al., 2020; Kipf & Welling, 2016a; Fang et al., 2021). Unlike other autoencoders (Amodio et al., 2019; Eraslan et al., 2019) used in scRNA-Seq analysis to recreate input data, GAEs have the unique ability to learn a low-dimensional representation of the graph topology and train node relationships from a global perspective of the entire graph (Kipf & Welling, 2016b). Notably, the scGNN (Wang et al., 2021) is a recently introduced multi-modal framework that leverages GAEs to model heterogeneous cell-cell relationships and complex gene expression patterns in scRNA-Seq data.

5 RESULTS

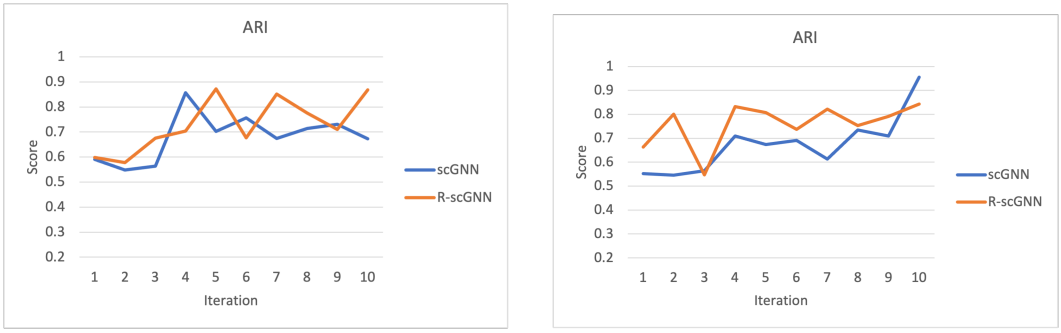
In this study, we assessed the performance of R-scGNN in comparison to scGNN. Additionally, we compared the performance of these models with nine imputation tools (i.e., MAGIC(Van Dijk et al., 2018), SAUCIE(Amodio et al., 2019), SAVER(Huang et al., 2018), scImpute(Li & Li, 2018), scVI(Lopez et al., 2018), DCA(Eraslan et al., 2019), DeepImpute(Arisdakessian et al., 2019), scIGANs(Xu et al., 2020), and netNMF-sc(Elyanow et al., 2020)) using the results that were presented in the scGNN paper (Wang et al., 2021). The predicted cell labels were evaluated based on 10 different criteria, including ARI and Silhouette score (Fig. 2 and see Appendix). ARI (Hubert & Arabie, 1985) was used to determine the similarities between all pairs of samples that were assigned to clusters in the current and previous clustering, adjusted by random permutation (see Appendix). The Silhouette coefficient score (Rousseeuw, 1987) is distinct from ARI in that it does not rely on known ground truth labels. Instead, it measures how similar an object is to its own cluster relative to other clusters (see Appendix).



(a) Zeizel dataset (b) Klein dataset

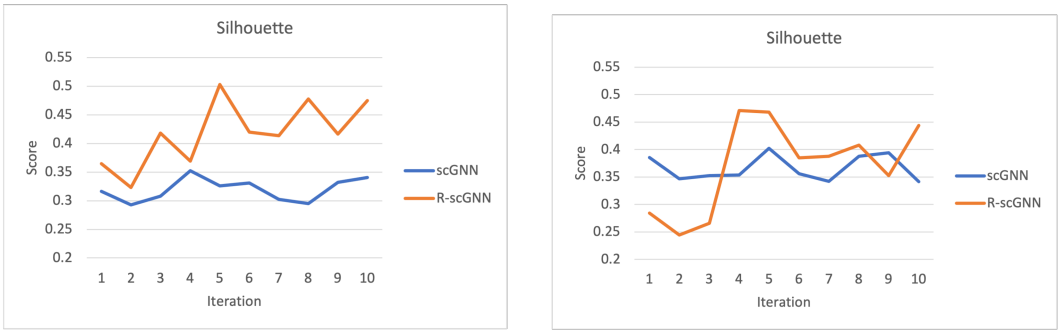
Figure 2: Comparision Silhouette and ARI scores among scGNN, R-scGNN, and nine tools

Figure 2 depicts the ARI and Silhouette scores obtained by various methods and tools on the Zeisel and Klein benchmark datasets. To ensure a fair comparison, both scGNN and R-scGNN were trained with identical hyperparameters that are common to both models, as well as the same hyperparameters utilized in the original study. After running for 10 iterations, our reformulated model R-scGNN outperformed all other methods in terms of both ARI and Silhouette scores. Specifically, R-scGNN achieved a higher Silhouette score on both datasets compared to scGNN. Although scGNN obtained a higher ARI score on the Klein dataset than R-scGNN (Fig. 2b), we observe that R-scGNN achieved a higher ARI score over the iterations until the final iteration (Fig. 3b).



(a) Zeizel dataset (b) Klein dataset

Figure 3: Comparision of ARI scores between scGNN and R-scGNN over the iterations



(a) Zeizel dataset

(b) Klein dataset

Figure 4: Comparision Silhouette scores between scGNN and R-scGNN over the iterations

Overall, the experimental results demonstrate that the proposed R-scGNN model outperforms other state-of-the-art methods in terms of clustering performance on scRNA-seq benchmark datasets. The comparison between R-scGNN and scGNN suggests that the adjustment of GNN towards clustering objectives has resulted in improved performance, as measured by the Silhouette score. Additionally, the R-scGNN model demonstrated consistent improvement in both ARI and Silhouette scores over the iterations on both datasets (Fig. 3, 4), indicating its ability to learn clustering-oriented task-specific features effectively.

In conclusion, our proposed R-scGNN model demonstrates promising performance in effectively learning task-specific features for scRNA-seq clustering analysis. Future research can explore the potential of R-scGNN on different clustering tasks and investigate its usefulness in other related problems.

REFERENCES

Matthew Amodio, David Van Dijk, Krishnan Srinivasan, William S Chen, Hussein Mohsen, Kevin R Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha Venkataswamy, et al. Exploring single-cell data with deep multitasking neural networks. *Nature methods*, 16(11):1139–1145, 2019.

Cédric Arisdakessian, Olivier Poirion, Breck Yunits, Xun Zhu, and Lana X Garmire. Deepimpute: an accurate, fast, and scalable deep neural network method to impute single-cell rna-seq data. *Genome biology*, 20(1):1–14, 2019.

Andrew Butler, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology*, 36(5):411–420, 2018.

Woosung Chung, Hye Hyeon Eum, Hae-Ock Lee, Kyung-Min Lee, Han-Byoel Lee, Kyu-Tae Kim, Han Suk Ryu, Sangmin Kim, Jeong Eon Lee, Yeon Hee Park, et al. Single-cell rna-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. *Nature communications*, 8(1):1–12, 2017.

Rebecca Elyanow, Bianca Dumitrascu, Barbara E Engelhardt, and Benjamin J Raphael. netnmf-sc: leveraging gene–gene interactions for imputation and dimensionality reduction in single-cell expression analysis. *Genome research*, 30(2):195–204, 2020.

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

Chao Fang, Dong Xu, Jing Su, Jonathan R Dry, and Bolan Linghu. Deepan: dee p pa tient graph convolutional n etwork integrating clinico-genomic evidence to stratify lung cancers for immunotherapy. *NPJ digital medicine*, 4(1):14, 2021.

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

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

Binyuan Hui, Pengfei Zhu, and Qinghua Hu. Collaborative graph convolutional networks: Unsupervised learning meets semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4215–4222, 2020.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Allon M Klein, Linas Mazutis, Ilke Akartuna, Naren Tallapragada, Adrian Veres, Victor Li, Leonid Peshkin, David A Weitz, and Marc W Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201, 2015.

Aleksandra A Kolodziejczyk, Jong Kyoung Kim, Jason CH Tsang, Tomislav Ilicic, Johan Henriksson, Kedar N Natarajan, Alex C Tuck, Xuefei Gao, Marc Bühler, Pentao Liu, et al. Single cell rna-sequencing of pluripotent states unlocks modular transcriptional variation. *Cell stem cell*, 17 (4):471–485, 2015.

Jacob H Levine, Erin F Simonds, Sean C Bendall, Kara L Davis, D Amir El-ad, Michelle D Tadmor, Oren Litvin, Harris G Fienberg, Astraea Jager, Eli R Zunder, et al. Data-driven phenotypic dissection of aml reveals progenitor-like cells that correlate with prognosis. *Cell*, 162(1):184–197, 2015.

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

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pp. 413–422. IEEE, 2008.

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

Nairouz Mrabah, Mohamed Bouguessa, and Riadh Ksantini. Adversarial deep embedded clustering: on a better trade-off between feature randomness and feature drift. *IEEE Transactions on Knowledge and Data Engineering*, 34(4):1603–1617, 2020.

Nairouz Mrabah, Mohamed Bouguessa, Mohamed Fawzi Touati, and Riadh Ksantini. Rethinking graph auto-encoder models for attributed graph clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

David Van Dijk, Roshan Sharma, Juozas Nainys, Kristina Yim, Pooja Kathail, Ambrose J Carr, Cassandra Burdziak, Kevin R Moon, Christine L Chaffer, Diwakar Pattabiraman, et al. Recovering gene interactions from single-cell data using data diffusion. *Cell*, 174(3):716–729, 2018.

Juexin Wang, Anjun Ma, Qin Ma, Dong Xu, and Trupti Joshi. Inductive inference of gene regulatory network using supervised and semi-supervised graph neural networks. *Computational and Structural Biotechnology Journal*, 18:3335–3343, 2020. ISSN 2001-0370. doi: https://doi.org/10.1016/j.csbj.2020.10.022. URL `https://www.sciencedirect.com/science/article/pii/S200103702030444X`.

Juexin Wang, Anjun Ma, Yuzhou Chang, Jianting Gong, Yuexu Jiang, Ren Qi, Cankun Wang, Hongjun Fu, Qin Ma, and Dong Xu. scgnn is a novel graph neural network framework for single-cell rna-seq analyses. *Nature communications*, 12(1):1882, 2021.

Yungang Xu, Zhigang Zhang, Lei You, Jiajia Liu, Zhiwei Fan, and Xiaobo Zhou. scigans: single-cell rna-seq imputation using generative adversarial networks. *Nucleic acids research*, 48(15): e85–e85, 2020.

Amit Zeisel, Ana B Muñoz-Manchado, Simone Codeluppi, Peter Lönnerberg, Gioele La Manno, Anna Juréus, Sueli Marques, Hermany Munguba, Liqun He, Christer Betsholtz, et al. Cell types in the mouse cortex and hippocampus revealed by single-cell rna-seq. *Science*, 347(6226):1138–1142, 2015.

A APPENDIX

Performance metric definitions:

The **ARI** is defined as:

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]}$$

where the unadjusted Rand Index (RI) is $\frac{a+b}{C_2^n}$. a is the number of pairs correctly labeled in the same sets, b is the number of pairs correctly labeled as not in the same set, and C_2^n is the total number of possible pairs. $E[RI]$ is the expected RI of random labeling.

The **Silhouette** coefficient score is defined as:

$$Silhouette = \frac{b - a}{\max(a, b)}$$

where a is the mean distance between a sample and all other points in the same cluster, and b is the mean distance between a sample and all other points in the next nearest cluster. The value of the Silhouette coefficient ranges from -1 to 1, where a score closer to 1 indicates better clustering.

1. chs - Calinski Harabasz score
2. dbs - Davies Bouldin score
3. ami - Adjusted Mutual Information
4. nmi - Normalized Mutual Information
5. cs - Completeness metric of a cluster labeling given a ground truth
6. fms - Fowlkes Mallows score
7. vms - V-measure score
8. hs - Homogeneity score

Performance evaluation on Klein dataset for 10 iterations

Model	EM_iteration	silhouette	chs	dbs	ari	ami	nmi	cs	fms	vms	hs
scGNN	1	0.3856	890.8797	1.2841	0.5526	0.7073	0.7078	0.6156	0.6674	0.7078	0.8324
	2	0.3468	762.0735	1.3910	0.5455	0.6598	0.6603	0.6014	0.6607	0.6603	0.7320
	3	0.3524	689.8105	1.4846	0.5646	0.6334	0.6340	0.5810	0.6759	0.6340	0.6976
	4	0.3536	658.9734	1.4071	0.7093	0.7779	0.7783	0.7156	0.7867	0.7783	0.8529
	5	0.4024	1085.8677	1.2202	0.6744	0.7595	0.7599	0.6953	0.7603	0.7599	0.8375
	6	0.3558	833.2872	1.5412	0.6914	0.7308	0.7311	0.7146	0.7743	0.7311	0.7485
	7	0.3421	802.0616	1.4282	0.6133	0.6961	0.6966	0.6382	0.7133	0.6966	0.7667
	8	0.3879	983.5029	1.2511	0.7351	0.7524	0.7527	0.7443	0.8077	0.7527	0.7613
	9	0.3942	1075.6419	1.3072	0.7091	0.7418	0.7421	0.7330	0.7887	0.7421	0.7515
	10	0.3414	797.5842	1.2370	0.9557	0.9284	0.9285	0.9329	0.9682	0.9285	0.9241
R-scGNN	1	0.2843	835.7129	1.4532	0.6632	0.7789	0.7793	0.6860	0.7538	0.7793	0.9020
	2	0.2444	697.2457	1.3464	0.8004	0.8099	0.8103	0.7310	0.8557	0.8103	0.9087
	3	0.2657	598.2633	1.4409	0.5464	0.6942	0.6946	0.6511	0.6657	0.6946	0.7445
	4	0.4707	1332.9637	0.8573	0.8318	0.8420	0.8423	0.7890	0.8778	0.8423	0.9033
	5	0.4678	1460.3053	1.1336	0.8076	0.7965	0.7968	0.7847	0.8602	0.7968	0.8092
	6	0.3849	1073.4454	1.4720	0.7377	0.7294	0.7297	0.7142	0.8085	0.7297	0.7459
	7	0.3874	938.8107	1.3940	0.8223	0.8038	0.8041	0.7918	0.8709	0.8041	0.8168
	8	0.4079	947.1517	1.2793	0.7528	0.7772	0.7776	0.7230	0.8190	0.7776	0.8411
	9	0.3521	859.0536	1.2658	0.7908	0.8068	0.8070	0.8150	0.8506	0.8070	0.7991
	10	0.4436	1401.3147	1.2367	0.8432	0.8215	0.8217	0.8111	0.8862	0.8217	0.8326

Performance evaluation on Zeisel dataset for 10 iterations

Model	EM_iteration	silhouette	chs	dbs	ari	ami	nmi	cs	fms	vms	hs
scGNN	1	0.3162	571.4686	1.3218	0.5912	0.7360	0.7370	0.6782	0.6756	0.7370	0.8069
	2	0.2927	565.1862	1.4899	0.5483	0.6923	0.6934	0.6555	0.6370	0.6934	0.7358
	3	0.3081	672.4607	1.2514	0.5640	0.7050	0.7060	0.6661	0.6504	0.7060	0.7510
	4	0.3526	736.8790	1.1518	0.8567	0.8072	0.8078	0.8238	0.8873	0.8078	0.7924
	5	0.3262	653.4614	1.2398	0.7019	0.7483	0.7491	0.7180	0.7630	0.7491	0.7830
	6	0.3308	657.0090	1.3342	0.7558	0.7498	0.7505	0.7508	0.8059	0.7505	0.7502
	7	0.3023	638.0317	1.3300	0.6743	0.7089	0.7098	0.7020	0.7400	0.7098	0.7177
	8	0.2950	635.7290	1.3097	0.7145	0.7347	0.7355	0.7358	0.7731	0.7355	0.7353
	9	0.3323	793.1103	1.3208	0.7307	0.7284	0.7292	0.7266	0.7856	0.7292	0.7317
	10	0.3404	788.0472	1.1691	0.6722	0.7117	0.7125	0.7048	0.7383	0.7125	0.7204
R-scGNN	1	0.3648	1067.5064	1.1607	0.5980	0.7367	0.7377	0.6787	0.6815	0.7377	0.8078
	2	0.3229	1061.4516	1.3502	0.5772	0.7179	0.7190	0.6615	0.6635	0.7190	0.7873
	3	0.4184	1380.8169	0.9424	0.6759	0.7267	0.7276	0.6961	0.7419	0.7276	0.7620
	4	0.3691	1073.8655	1.2992	0.7037	0.7532	0.7540	0.7243	0.7644	0.7540	0.7862
	5	0.5032	1734.6884	0.7472	0.8726	0.8174	0.8179	0.8351	0.9000	0.8179	0.8014
	6	0.4196	1405.2762	1.0884	0.6769	0.7329	0.7337	0.7262	0.7422	0.7337	0.7414
	7	0.4134	1052.9282	0.9305	0.8519	0.8076	0.8082	0.8250	0.8836	0.8082	0.7920
	8	0.4780	1361.3542	1.0087	0.7755	0.7607	0.7614	0.7631	0.8217	0.7614	0.7597
	9	0.4167	1350.0821	0.9814	0.7101	0.7593	0.7601	0.7287	0.7698	0.7601	0.7943
	10	0.4749	1504.1955	0.8056	0.8678	0.8179	0.8184	0.8356	0.8962	0.8184	0.8019

Performance evaluation on Chung dataset for 10 iterations

Model	EM_iteration	silhouette	chs	dbs	ari	ami	nmi	cs	fms	vms	hs
scGNN	1	0.6876	288.2659	0.7377	0.5202	0.6667	0.6721	0.5789	0.6505	0.6721	0.8011
	2	0.7220	511.4175	0.4704	0.5603	0.7207	0.7260	0.6075	0.6863	0.7260	0.9019
	3	0.7161	425.8960	0.5246	0.5603	0.7207	0.7260	0.6075	0.6863	0.7260	0.9019
	4	0.6305	201.2398	0.9086	0.5256	0.6772	0.6825	0.5878	0.6549	0.6825	0.8136
	5	0.6661	264.6249	0.7758	0.5175	0.6725	0.6778	0.5840	0.6484	0.6778	0.8076
	6	0.6591	310.1723	0.6652	0.5614	0.7211	0.7264	0.6079	0.6872	0.7264	0.9024
	7	0.6719	352.3501	0.6684	0.5264	0.6593	0.6658	0.5552	0.6588	0.6658	0.8312
	8	0.6376	244.7774	0.7520	0.5603	0.7207	0.7260	0.6075	0.6863	0.7260	0.9019
	9	0.6107	190.0878	0.9218	0.5256	0.6772	0.6825	0.5878	0.6549	0.6825	0.8136
	10	0.6830	357.8285	0.5903	0.5259	0.6686	0.6749	0.5652	0.6575	0.6749	0.8375
R-scGNN	1	-0.0049	14.6768	3.5045	0.0421	0.1272	0.1422	0.1258	0.2894	0.1422	0.1634
	2	-0.1858	11.5176	7.3145	0.0277	0.0849	0.1027	0.0867	0.2497	0.1027	0.1259
	3	0.0858	411.4884	3.1640	0.3048	0.4573	0.4682	0.4040	0.4831	0.4682	0.5568
	4	0.1708	53.9073	2.5506	0.2451	0.3507	0.3616	0.3153	0.4347	0.3616	0.4237
	5	0.5511	222.6985	0.8008	0.4907	0.6443	0.6502	0.5637	0.6268	0.6502	0.7680
	6	0.5807	307.2575	0.6220	0.5368	0.6947	0.7005	0.5874	0.6660	0.7005	0.8678
	7	0.3916	386.7627	2.8294	0.4933	0.6445	0.6514	0.5518	0.6290	0.6514	0.7950
	8	-0.0794	32.3185	2.7337	0.3809	0.5465	0.5550	0.4615	0.5371	0.5550	0.6960
	9	-0.0510	16.0191	3.3568	0.1656	0.2679	0.2821	0.2385	0.3624	0.2821	0.3452
	10	0.2372	79.6030	1.4326	0.3121	0.5065	0.5162	0.4390	0.4841	0.5162	0.6262

Performance evaluation on Kolodziejczyk dataset for 10 iterations

Model	EM_iteratior	silhouette	chs	dbs	ari	ami	nmi	cs	fms	vms	hs
scGNN	1	0.5640	599.3016	0.6706	0.3477	0.5692	0.5719	0.4358	0.5307	0.5719	0.8320
	2	0.5138	388.2597	0.9179	0.3970	0.5777	0.5802	0.4528	0.5716	0.5802	0.8071
	3	0.6606	543.7454	0.8030	0.4626	0.6131	0.6151	0.4947	0.6242	0.6151	0.8128
	4	0.6658	609.8360	0.7754	0.4921	0.6486	0.6504	0.5241	0.6477	0.6504	0.8568
	5	0.6450	618.8784	0.7631	0.4938	0.6528	0.6546	0.5275	0.6491	0.6546	0.8623
	6	0.6298	653.2862	0.7430	0.5905	0.7367	0.7381	0.5985	0.7248	0.7381	0.9626
	7	0.5895	504.0710	0.7478	0.4938	0.6528	0.6546	0.5275	0.6491	0.6546	0.8623
	8	0.7055	1265.0695	0.4506	0.4938	0.6528	0.6546	0.5275	0.6491	0.6546	0.8623
	9	0.6603	834.1474	0.5560	0.4938	0.6528	0.6546	0.5275	0.6491	0.6546	0.8623
	10	0.6967	1483.5470	0.4738	0.4938	0.6528	0.6546	0.5275	0.6491	0.6546	0.8623
R-scGNN	1	0.4563	407.2101	1.2603	0.3927	0.6073	0.6098	0.4629	0.5727	0.6098	0.8933
	2	0.4423	1006.3037	1.5275	0.4386	0.6563	0.6585	0.5018	0.6120	0.6585	0.9574
	3	0.1245	324.0309	3.0860	0.3970	0.5737	0.5760	0.4740	0.5752	0.5760	0.7338
	4	0.1651	234.2589	1.8491	0.4029	0.6022	0.6040	0.5278	0.5926	0.6040	0.7060
	5	0.5099	343.8125	0.8861	0.3934	0.5422	0.5446	0.4499	0.5722	0.5446	0.6899
	6	0.1356	456.5150	2.9257	0.4938	0.6528	0.6546	0.5275	0.6491	0.6546	0.8623
	7	-0.1766	19.9090	5.2419	0.1236	0.2653	0.2692	0.2237	0.3720	0.2692	0.3380
	8	-0.3044	54.8905	6.6831	0.0549	0.1669	0.1713	0.1431	0.3293	0.1713	0.2135
	9	0.0207	48.3586	7.0952	0.3314	0.4783	0.4809	0.3905	0.5202	0.4809	0.6260
	10	0.0802	321.2803	5.3962	0.4938	0.6528	0.6546	0.5275	0.6491	0.6546	0.8623

Performance scores after 10 iterations

results source	model/method	dataset	silhouette	chs	dbs	ari	ami	nmi	cs	fms	vms	hs
this study	scGNN	Dataset Chung	0.6830	357.8285	0.5903	0.5259	0.6686	0.6749	0.5652	0.6575	0.6749	0.8375
		Dataset Kolodzie	0.6967	1483.5470	0.4738	0.4938	0.6528	0.6546	0.5275	0.6491	0.6546	0.8623
		Dataset Klein	0.3414	797.5842	1.2370	0.9557	0.9284	0.9285	0.9329	0.9682	0.9285	0.9241
		Dataset Zeisel	0.3404	788.0472	1.1691	0.6722	0.7117	0.7125	0.7048	0.7383	0.7125	0.7204
	R-scGNN	Dataset Chung	0.2372	79.6030	1.4326	0.3121	0.5065	0.5162	0.4390	0.4841	0.5162	0.6262
		Dataset Kolodzie	0.0802	321.2803	5.3962	0.4938	0.6528	0.6546	0.5275	0.6491	0.6546	0.8623
		Dataset Klein	0.4436	1401.3147	1.2367	0.8432	0.8215	0.8217	0.8111	0.8862	0.8217	0.8326
		Dataset Zeisel	0.4749	1504.1955	0.8056	0.8678	0.8179	0.8184	0.8356	0.8962	0.8184	0.8019
Wang et. al (2021)	scGNN	Dataset Chung	0.6798	342.3147	0.5682	0.5561	0.7111	0.7166	0.5996	0.6828	0.7166	0.8903
		Dataset Kolodzie	0.6933	1247.5089	0.4875	0.4919	0.6470	0.6488	0.5229	0.6475	0.6488	0.8546
		Dataset Klein	0.3705	841.2180	1.2242	0.9599	0.9332	0.9333	0.9363	0.9712	0.9333	0.9303
		Dataset Zeisel	0.3316	714.0476	1.5136	0.6472	0.6741	0.6750	0.6676	0.7181	0.6750	0.6826
	magic	Dataset Chung	0.5776	616.1697	0.6683	0.3851	0.6195	0.6305	0.4891	0.5493	0.6305	0.8869
		Dataset Kolodzie	0.4493	677.1990	0.7468	0.2099	0.4930	0.4992	0.3458	0.4088	0.4992	0.8970
		Dataset Klein	0.3604	3969.9636	0.8186	0.2620	0.5899	0.5921	0.4283	0.4424	0.5921	0.9590
		Dataset Zeisel	0.2444	1002.7641	1.2485	0.2438	0.5763	0.5809	0.4458	0.3970	0.5809	0.8336
	dca	Dataset Chung	0.2865	233.4430	1.1323	0.2363	0.4150	0.4327	0.3280	0.4081	0.4327	0.6356
		Dataset Kolodzie	0.2239	318.2352	1.2525	0.1717	0.3883	0.3950	0.2774	0.3602	0.3950	0.6857
		Dataset Klein	0.1528	998.0465	1.4936	0.3320	0.6303	0.6317	0.4758	0.5026	0.6317	0.9394
		Dataset Zeisel	0.0209	268.5575	1.7819	0.2952	0.5494	0.5527	0.4494	0.4288	0.5527	0.7176
	deepimpute	Dataset Chung	0.1723	86.3652	1.6433	0.2626	0.3072	0.3225	0.2668	0.4392	0.3225	0.4076
		Dataset Kolodzie	0.1149	285.4319	1.5744	0.1930	0.3836	0.3892	0.2797	0.3832	0.3892	0.6397
		Dataset Klein	0.0657	435.5078	1.8677	0.3890	0.6087	0.6100	0.4740	0.5443	0.6100	0.8555
		Dataset Zeisel	0.0299	314.7404	1.8330	0.2778	0.5248	0.5285	0.4281	0.4111	0.5285	0.6901
	netNMFsc	Dataset Chung	0.3806	98.6587	1.0939	0.4568	0.6699	0.6778	0.5418	0.6078	0.6778	0.9051
		Dataset Kolodzie	0.4108	515.4548	1.0644	0.4212	0.6426	0.6452	0.4863	0.5983	0.6452	0.9586
		Dataset Klein	0.1659	1553.2700	1.4068	0.4151	0.6714	0.6725	0.5202	0.5718	0.6725	0.9510
		Dataset Zeisel	0.1792	629.9800	1.5186	0.3638	0.6320	0.6348	0.5138	0.5005	0.6348	0.8305
	saucie	Dataset Chung	0.3369	133.4865	1.1132	0.3706	0.6053	0.6165	0.4763	0.5372	0.6165	0.8739
		Dataset Kolodzie	0.2368	279.7260	1.1350	0.1943	0.4206	0.4261	0.3028	0.3853	0.4261	0.7192

	Dataset Klein	0.2692	2382.6210	1.1584	0.2833	0.5890	0.5907	0.4355	0.4590	0.5907	0.9179
	Dataset Zeisel	0.1498	863.9494	1.4834	0.2842	0.5863	0.5901	0.4640	0.4298	0.5901	0.8101
saver	Dataset Chung	0.2427	43.3898	1.5088	0.5470	0.7071	0.7135	0.5873	0.6773	0.7135	0.9087
	Dataset Kolodzie	0.2922	315.8934	1.4932	0.4441	0.6711	0.6733	0.5132	0.6168	0.6733	0.9784
	Dataset Klein	0.1894	1248.6048	1.6554	0.5277	0.7212	0.7221	0.5854	0.6570	0.7221	0.9420
	Dataset Zeisel	0.1596	523.0927	1.6727	0.4041	0.6388	0.6417	0.5242	0.5338	0.6417	0.8271
scIGANs	Dataset Chung	0.0982	14.0216	2.5202	0.4512	0.6102	0.6200	0.5048	0.5961	0.6200	0.8031
	Dataset Kolodzie	0.0817	45.0501	3.0817	0.3918	0.5703	0.5728	0.4458	0.5676	0.5728	0.8011
	Dataset Klein	0.1369	673.9638	1.6738	0.4662	0.7036	0.7047	0.5503	0.6135	0.7047	0.9795
	Dataset Zeisel	0.1158	386.4652	1.9007	0.4385	0.6970	0.6992	0.5771	0.5660	0.6992	0.8867
scimpute	Dataset Chung	0.0005	11.0332	2.8229	0.3967	0.4986	0.5070	0.4413	0.5531	0.5070	0.5956
	Dataset Kolodzie	0.1298	82.0841	2.5459	0.4226	0.6267	0.6291	0.4794	0.5978	0.6291	0.9147
	Dataset Klein	0.1389	1090.0368	1.7223	0.3475	0.6353	0.6365	0.4846	0.5128	0.6365	0.9273
	Dataset Zeisel	0.0596	274.0794	2.1354	0.3722	0.6245	0.6273	0.5146	0.5028	0.6273	0.8032
scvi	Dataset Chung	0.1560	169.7994	1.1967	0.1078	0.2464	0.2679	0.2072	0.2833	0.2679	0.3790
	Dataset Kolodzie	0.2400	74.1996	0.9290	0.1258	0.2805	0.2883	0.2021	0.3066	0.2883	0.5028
	Dataset Klein	0.1745	4608.2789	1.6178	0.1507	0.4133	0.4166	0.2986	0.3098	0.4166	0.6888
	Dataset Zeisel	0.2594	5284.8967	1.1028	0.2000	0.4598	0.4641	0.3713	0.3310	0.4641	0.6188