Toward Convex Manifolds: A Geometric Perspective for Deep Graph Clustering of Single-cell RNA-seq Data

Nairouz Mrabah *, **Mohamed Mahmoud Amar**, **Mohamed Bouguessa**, **Abdoulaye Banire Diallo**
University of Quebec at Montreal, Montreal, Quebec, Canada
{mrabah.nairouz, amar.mohamed_mahmoud}@courrier.uqam.ca,
{bouguessa.mohamed, diallo.abdoulaye}@uqam.ca

Abstract

The deep clustering paradigm has shown great potential for discovering complex patterns that can reveal cell heterogeneity in single-cell RNA sequencing data. This paradigm involves two training phases: pretraining based on a pretext task and fine-tuning using pseudo-labels. Although current models yield promising results, they overlook the geometric distortions that regularly occur during the training process. More precisely, the transition between the two phases results in a coarse flattening of the latent structures, which can deteriorate the clustering performance. In this context, existing methods perform euclidean-based embedding clustering without ensuring the *flatness* and *convexity* of the latent manifolds. To address this problem, we incorporate two mechanisms. First, we introduce an overclustering loss to flatten the local curves. Second, we propose an adversarial mechanism to adjust the global geometric configuration. The second mechanism gradually transforms the latent structures into convex ones. Empirical results on a variety of gene expression datasets show that our model outperforms state-of-the-art methods.

1 Introduction

Single-cell RNA sequencing (scRNA-seq) facilitates the study of individual cells, thus providing an in-depth understanding of the basic unit of biology. The assessment of transcriptional heterogeneity within a cell population can provide answers to a wide range of biological research questions. For example, scRNA-seq datasets have been commonly used in the discovery of new cell types [Saviano *et al.*, 2020], the identification of highly pathogenic cells in tumor tissue [Tirosh *et al.*, 2016], the detection of hyper-responsive immune cells [Shalek *et al.*, 2014], the analysis of resistance to treatments [Ocasio *et al.*, 2019], and the study of developmental processes in specific cellular conditions such as cancer [Brady *et al.*, 2017] and pulmonary epithelium differentiation [Treutlein *et al.*, 2014]. Since manual cell annotation is labor intensive and can be prone to mislabeling, unsupervised cell identification remains a critical research direction.

The analysis of scRNA-seq data heavily relies on cell type identification. To accomplish this task, various clustering algorithms have been used. In particular, researchers have applied traditional methods such as k-means, spectral clustering [Park and Zhao, 2018], density-based clustering [Ester *et al.*, 1996], and hierarchical clustering [Johnson, 1967] to detect cell populations. Furthermore, dimensionality reduction techniques including PCA, TSNE, and UMAP have been applied before clustering to overcome the curse of dimensionality. Unfortunately, the traditional clustering algorithms often produce suboptimal results. This is because the transcriptomic data have complex relationships between cells within a cluster, significant sparsity caused by the dropout events, and strong technical variability of the gene expression levels.

To address the shortcomings of traditional clustering approaches, CIDR [Lin *et al.*, 2017], MAGIC [van Dijk *et al.*, 2017], and SAVER [Huang *et al.*, 2018], first impute the missing values, also known as dropouts, and then cluster the imputed data. Despite the positive effects of imputation, CIDR, MAGIC, and SAVER struggle to capture the complex structure inherent to scRNA-seq data. Some other approaches, such as SIMLR [Wang *et al.*, 2017], and MPSSC [Park and Zhao, 2018], leverage multi-kernel spectral clustering to learn robust similarity measures. However, computing the Laplacian matrix is time-consuming, which in turn precludes its applicability to large-scale datasets. Moreover, these methods overlook important characteristics of the transcriptional data, such as zero inflation and over-dispersion.

Recently, the deep clustering paradigm has found success in many biological applications. This paradigm relies on a two-fold strategy consisting of self-supervised learning followed by a pseudo-supervision task. During the initial phase, a pretext task is solved as a proxy to acquire high-level representations. The second phase simultaneously learns clustering assignments and clustering-oriented embeddings by training with pseudo-labels. Existing deep clustering methods for scRNA-seq data include scziDesk [Chen *et al.*, 2020], scDC [Tian *et al.*, 2019], and scDCC [Tian *et al.*, 2021]. These models operationalize auto-encoding architectures. Furthermore, all of them integrate the Zero-Inflated Negative Binomial distribution (ZINB) to model the key properties of the gene expression data (over-dispersion, zero inflation, and dis-

*Contact Author

creteness). However, they fail to consider cell-cell relationships, which can make the clustering task more challenging. As a result, more recent models, such as scTAG [Yu *et al.*, 2022] and scGAE [Luo *et al.*, 2021], leverage graph neural networks (GNNs) to preserve the neighborhood relationships.

Although deep clustering has achieved promising results for scRNA-seq data, existing studies fail to consider two important issues: 1) the suitability of the geometric configuration obtained at the end of the pretraining phase to the clustering task, and 2) the geometric distortions that regularly occur during the training process. Several measures can be used to analyze the geometry of a manifold, such as curvature and torsion. In this work, we focus on two measures, namely *Intrinsic Dimension* (ID) and *Linear Intrinsic Dimension* (LID). Intuitively, ID computes the number of independent directions a point can move in without leaving the manifold. The LID computes the smallest dimension of a linear subspace that best approximates the manifold in a least-squares sense at each point. An increasing discrepancy between ID and LID during the training process indicates that the ongoing geometric transformation makes the manifold more curved.

The authors in [Mrabah *et al.*, 2022] have conducted a geometric investigation to examine the behavior of ID and LID under the deep clustering paradigm. They have shown that self-supervised learning (i.e., pretraining phase) leads to the emergence of *curved* latent manifolds with low intrinsic dimensions. Moreover, introducing pseudo-supervised learning after the first phase flattens the latent structures coarsely, which in turn degrades the clustering performance by twisting the curvatures [Mrabah *et al.*, 2022]. The main goal of our approach is to avoid this sharp geometric transition caused by minimizing a euclidean-based clustering loss after the first phase. Without prior knowledge, there is no systematic way to identify the optimal non-euclidean metric that can capture the latent similarities. Therefore, it is crucial to gradually adjust the geometric configuration of the latent manifolds, so that it becomes appropriate for euclidean-based clustering.

We propose a novel single-cell graph auto-encoder model that follows the deep clustering paradigm. Unlike previous methods, our approach tackles the coarse geometric transition between pretraining and clustering. We incorporate two mechanisms to address this problem. We argue that enforcing *local flatness* and *global convexity* in a gradual manner is favorable for euclidean-based embedding clustering. Our first mechanism targets the local geometric configuration. In particular, we supply the pseudo-supervision module with an overclustering loss. This loss is used to *flatten* the strong local curves without destroying the global structure. The second mechanism adjusts the global geometric configuration. More precisely, we introduce an adversarial loss to gradually transform the latent manifolds into *convex* ones. We build a discriminator for each cluster. Our discriminators are trained to make the auto-encoder push convex combinations of samples with high-confidence clustering assignments inside their corresponding manifold. By ensuring convexity, the clusters become linearly separable [Boyd *et al.*, 2004], which in turn improves the clustering performance.

Contributions. (1) We establish the first single-cell deep clustering model that tackles the coarse geometric transition between the first and second training phases. **(2)** We propose a mechanism that performs a local geometric transformation of the latent manifolds. In particular, we introduce an overclustering loss to flatten the local curves without destroying the global structures. **(3)** We design a second mechanism that adjusts the global geometric configuration. More precisely, we introduce an adversarial loss to gradually transform the latent manifolds into convex ones. **(4)** We conduct several experiments on real-world scRNA-seq datasets. The obtained results confirm the validity of our contributions and show that the proposed model outperforms state-of-the-art methods.

2 Related Work

We discuss two main strategies: 1) the deep embedding methods, which perform clustering and feature learning separately, 2) the deep clustering models, which perform clustering and feature learning jointly. Other relevant methods outside the field of single-cell research are discussed in Appendix A (all appendices are provided in the Supplementary Material[†]).

Deep embedding methods. In [Yu *et al.*, 2021], scGMAI leverages a deep auto-encoder as an imputation technique, followed by the Fast Independent Component Analysis (FastICA) for dimensionality reduction, and a Gaussian Mixture Model (GMM) for clustering. Instead of performing vanilla reconstruction, scDCA [Eraslan *et al.*, 2019] operationalizes a denoising auto-encoder. The decoding task of scDCA is modeled by the ZINB distribution to capture the characteristics of the gene expression data. In another work, scGNN [Wang *et al.*, 2021] achieves multi-modal reconstruction using several auto-encoders. Moreover, this model relies on a GNN to aggregate cell–cell relationships. A left-truncated GMM is applied to uncover regulatory signals within the gene expression count matrix. Unfortunately, the existing deep embedding methods overlook the suitability of the geometric configuration for the clustering task. More precisely, self-supervised learning leads to curved latent manifolds, thus we do not expect euclidean-based approaches such as GMM or k-means to identify the clustering structures effectively.

Deep clustering methods. Compared with scDCA, scDC [Tian *et al.*, 2019] combines the denoising task with an embedding clustering process. Both tasks are optimized jointly. More precisely, scDC adopts the clustering loss of DEC [Xie *et al.*, 2016]. To improve scDC, scDCC [Tian *et al.*, 2021] expresses prior information and domain-specific knowledge as soft pairwise constraints, then integrate these constraints as an additional term in the final loss function. In another work, scziDesk [Chen *et al.*, 2020] uses a different loss function compared with scDC. It employs a weighted k-means loss to strengthen the initial pairwise similarities after encoding. Recently, several methods leverage a graph convolutional encoder to maintain the proximity of similar data points in the latent space. For instance, scGAE [Luo *et al.*, 2021] replaces the vanilla auto-encoding framework of scDC with a graph convolutional auto-encoder. In [Yu *et al.*, 2022], scTAG adopts a topology adaptive graph convolutional auto-encoder. Unlike scGAE and scTAG, scDSC [Gan *et al.*, 2022] encodes

[†]https://github.com/MMAMAR/scTConvexMan

the gene expression data and the associated structural similarities using two encoders, and then maximizes the mutual information between their outputs. However, all these methods overlook the coarse geometric transformation caused by the sharp transition between pretraining and clustering.

3 Proposed Approach

We define the gene expression count matrix $X = (x_{ij}) \in \mathbb{R}^{n_s \times n_g}$, where n_g represents the number of genes, n_s is the number of cells, and the x_{ij} element is the count of the j^{th} gene in the i^{th} cell. We consider the problem of clustering the scRNA-seq data X and we denote by n_c the corresponding number of clusters. Let $C = (c_1, ..., c_{n_s})$ be the sequence of clustering indices for each cell such that $c_i \in \{1, ..., n_c\}$.

The cell representations for scRNA-seq data can have pronounced similarities based on the gene expression count. To this end, we build a non-directed graph to capture the local structure characterizing these pronounced similarities. We define the graph structure by considering the relationship between each cell and its nearest neighbors. Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ be an attributed graph, where $\mathcal{V} = \{v_1, v_2, ..., v_{n_s}\}$ is a set of n_s nodes associated with different cells; $e_{ij} \in \mathcal{E}$ specifies the existence of an edge between the i^{th} and j^{th} nodes; and X is used as the attribute matrix of $\mathcal{G}$. Two nodes v_i and v_j are connected if and only if x_i (x_j, respectively) is located within the k nearest neighbors of x_j (x_i, respectively) based on the Euclidean distance. Let $A = (a_{ij}) \in \mathbb{R}^{n_s \times n_s}$ be the resultant adjacency matrix of $\mathcal{G}$, where $a_{ij} = 1$ if v_i and v_j are connected, otherwise a_{ij} is set equal to zero.

We build a GNN architecture to extract high-level representations from the graph $\mathcal{G}$. In particular, we devise a graph auto-encoder [Kipf and Welling, 2016] to learn the node embeddings. The first component is an encoder f_E. The encoding process relies on L_E graph convolutional layers [Kipf and Welling, 2017] that project the graph $\mathcal{G}$ into a low-dimensional latent space. We denote by $Z = f_E(X, A) \in \mathbb{R}^{n_s \times d}$ the matrix of embedded codes obtained by the encoding process, where d is the dimension of the latent space. The graph convolutional operation of the l^{th} layer is described:

$$Z^{(l)} = \text{ReLU}(\widetilde{\Delta}^{-\frac{1}{2}} \widetilde{A} \widetilde{\Delta}^{-\frac{1}{2}} Z^{(l-1)} W_E^{(l)}), \qquad (1)$$

where $W_E^{(l)}$ represents the training weights of the l^{th} layer; ReLU is the corresponding activation function; $Z^{(l)}$ is the output of this layer, $Z^{(0)} = X$, and $Z^{(L_E)} = Z$; $\widetilde{\Delta}^{-\frac{1}{2}} \widetilde{A} \widetilde{\Delta}^{-\frac{1}{2}}$ is the normalized adjacency matrix such that $\widetilde{A} = A + I$, $I \in \mathbb{R}^{n_s \times n_s}$ is the identity matrix, and $\widetilde{\Delta} = \text{Diag}(\widetilde{A}\, \mathbf{1}_{n_s})$.

The second component of the auto-encoding model is a two-head decoder. The decoding process aims to reconstruct the input graph $\mathcal{G}$. We denote by $\mathcal{G}^D$ the attributed graph generated by the decoding process. The first head $f_D^{(1)}$ is devoted to computing the structure of $\mathcal{G}^D$ using the inner product operation. The output of the first head is the adjacency matrix of $\mathcal{G}^D$ denoted $\hat{A} = f_D^{(1)}(Z)$. The second head $f_D^{(2)}$ is devoted to computing the attributes of $\mathcal{G}^D$. It has L_D fully-connected layers with ReLU activations. The output of the second head is the feature matrix $\hat{X} = f_D^{(2)}(Z)$. We denote by $W_D^{(l)}$ the training weights of the l^{th} decoding layer and all the training weights of our auto-encoder are denoted by the set W.

In addition to the auto-encoding framework, our approach includes an adversarial training strategy. We use the graph auto-encoder constrained to the second decoding head (that is, $f_D^{(2)} \circ f_E$) as the generator network. Moreover, we devise n_c fully connected discriminators $f_{\text{adv}}^{(k)} : \mathbb{R}^d \to [0, 1]$ with L_{adv} layers, such that $k \in \{1, ..., n_c\}$. We denote by $W_{\text{adv}}^{(k,l)}$ the training weights of the l^{th} layer in the k^{th} discriminator.

3.1 Training Losses

According to the deep clustering paradigm, we train our model based on self-supervision for the pretraining phase, then we introduce a pseudo-supervision loss for the clustering phase. To avoid the coarse geometric transition from pretraining to clustering, we propose two mechanisms. The first mechanism targets the *local* geometric configuration of the latent manifolds. In particular, we supply the pseudo-supervision module with an overclustering loss to *flatten* the strong local curves without destroying the globally-curved shape. The second mechanism adjusts the global geometric configuration. More precisely, we introduce an adversarial loss to gradually transform the manifolds into *convex* ones.

Self-supervision Module

It consists of tackling a secondary task that requires a certain level of data understanding to be solved. The self-supervision loss is optimized for the first and second phases. The pretext task of our approach consists of reconstructing the input graph $\mathcal{G}$. We model the decoding process for the reconstruction task by the probability distribution $p(\hat{A}, \hat{X} \mid Z)$. We develop a two-task decoding design to perform both structure reconstruction and feature reconstruction. Accordingly, we factorize the joint distribution $p(\hat{A}, \hat{X} \mid Z)$ into two independent distributions $p(\hat{A} \mid Z)$ and $p(\hat{X} \mid Z)$. We parameterize the probability distribution $p(\hat{A} \mid Z)$ by the first decoding head. We opt for the Bernoulli distribution to capture the binary nature of each generated edge as described by:

$$p(\hat{A}|Z) = \prod_{i,j=1}^{n_s} p(\hat{a}_{ij} \mid z_i, z_j) = \prod_{i,j=1}^{n_s} \mathcal{B}(\hat{a}_{ij} \mid \beta_{ij}), \qquad (2)$$

where the probability of generating each edge $\mathcal{B}(\hat{a}_{ij}|\beta_{ij})$ is a Bernoulli distribution parametrized by $\beta_{ij} = \text{sigmoid}(z_i^T z_j)$.

For feature reconstruction, we parameterize the probability distribution $p(\hat{X} \mid Z)$ by the second decoding head. We operationalize the ZINB distribution [Wang *et al.*, 2021] to capture the key aspects of the gene expression data: 1) zero-inflation (high-sparsity) caused by the true and dropout zeros, 2) discreteness, and 3) over-dispersion (variance greater than the mean). We model the distribution $p(\hat{X} \mid Z)$ as follows:

$$p(\hat{X}|Z) = \prod_{i=1}^{n_s} p(\hat{x}_i \mid z_i) = \prod_{i=1}^{n_s} \mathcal{Z}(\hat{x}_i \,|\pi_i, \mu_i, \sigma_i^2), \qquad (3)$$

where $\mathcal{Z}(\hat{x}_i|\pi_i, \mu_i, \sigma_i^2)$ is the ZINB distribution parameterized by $\pi_i, \mu_i, \sigma_i^2 \in \mathbb{R}^{n_g}$; $\pi_i = \text{sigmoid}(W_\pi f_D^{(2)}(z_i))$ is

the zero rate vector of this distribution and W_π denotes the weight matrix of π; $\mu_i = \exp(W_\mu f_D^{(2)}(z_i))$ is the mean of the associated negative binomial and W_μ is the weight matrix of μ; $\sigma^2 = \exp(W_\sigma f_D^{(2)}(z_i))$ is the variance of the negative binomial and W_σ is the weight matrix of σ^2. The complete expression of the ZINB mass function with respect to its parameters π_i, μ_i, σ_i^2 is provided in Appendix B to save space.

The self-supervision module reconstructs the input graph by minimizing the loss function L_{SS}. More precisely, we optimize the log-likelihood of the input graph based on the distribution $p(\hat{X},\ \hat{A}|\ Z)$ as described by:

$$L_{\text{SS}} = -\log\big(p(\mathcal{G}^D = \mathcal{G}\,|\,Z)\big) = L_X + L_A, \tag{4}$$

where $L_X (L_A,$ respectively) is the log-likelihood of the gene count matrix X (adjacency matrix $A,$ respectively).

Pseudo-supervision Module

It is trained using pseudo-labels during the second phase to achieve the primary task (i.e., clustering). We design a pseudo-supervision strategy that puts into action two loss functions: a clustering loss function $\mathcal{L}_c$ and an overclustering loss function $\mathcal{L}_o$. The overclustering procedure refers to clustering the data into a number of categories n_o greater than the real number of clusters n_c.

The goal of the clustering loss is to construct clustering-friendly latent structures that reduce the within-cluster variance and maximizes the between-cluster variance. We model the clustering process using two distributions $p(C|Z)$ and $q(C|Z)$. Each cluster is characterized by its center. Let $\{\Phi_j^c\}_{j=1}^{n_c}$ be the set of trainable clustering centers. We initialize the points of this set, after the pretraining phase, by encoding the centers obtained from applying spectral clustering on A. The first distribution $p(C|Z)$ measures the soft clustering assignments and factorizes as $p(C|Z) = \prod_{i=1}^{n_s} p(c_i\,|\,z_i)$. We compute p_{ij}^c, which is the probability of assigning the latent code z_i to the j^{th} cluster (i.e., $p(c_i = j|z_i)$), according to the Student's t-distribution as described by:

$$p_{ij}^c = \frac{(1 + \left\|z_i - \Phi_j^c\right\|^2)^{-1}}{\sum_{j'}(1 + \left\|z_i - \Phi_{j'}^c\right\|^2)^{-1}}. \tag{5}$$

The second distribution $q(C|Z)$ models the target clustering assignments (i.e., pseudo-labels). The factorization of this distribution is $q(C|Z) = \prod_{i=1}^{n_c} q(c_i\,|\,z_i)$. We compute the target assignments $q(c_i|z_i)$ of the i^{th} sample iteratively based on the high-confidence assignments of $p(c_i|z_i)$. We denote by q_{ij}^c the probability value $q(c_i = j|z_i)$. Let τ_i^1 (τ_i^2, respectively) be the highest (second highest, respectively) assignment score of the set $\{p_{ij}^c\}_{j=1}^{n_c}$. We denote by $\Omega_c = \{x_i \in \mathbb{R}^{n_g} \mid \tau_i^1 - \tau_i^2 \geqslant \beta_c\}$ the set of reliable samples for the clustering task, and β_c is a fixed threshold. To emphasize the high-confidence assignments, we compute q_{ij}^c using:

$$q_{ij}^c = \begin{cases} 1 \text{ if } x_i \in \Omega_c \text{ and } j = \arg\max_{j'}(p_{ij'}^c), \\ 0 \text{ if } x_i \in \Omega_c \text{ and } j \neq \arg\max_{j'}(p_{ij'}^c), \\ p_{ij}^c \text{ if } x_i \notin \Omega_c. \end{cases} \tag{6}$$

Our clustering loss $\mathcal{L}_c$ is the KL (Kullback–Leibler) divergence between the soft assignment distribution $p(C|Z)$ and the target distribution $q(C|Z)$ as described by:

$$\mathcal{L}_c = \text{KL}\big(q(C|Z)||p(C|Z)\big) = \sum_{i=1}^{n_s}\sum_{j=1}^{n_c} q_{ij}^c \log(\frac{q_{ij}^c}{p_{ij}^c}). \tag{7}$$

In addition to the clustering loss, we design an overclustering task to flatten the local structures without twisting the latent manifolds. More precisely, the goal of the overclustering process is to smooth the local curves while preserving the globally-curved geometric structures as shown by [Mrabah *et al.*, 2022]. Let n_o be the number of clusters for the overclustering task. We model the overclustering process using two distributions $p(O|Z)$ and $q(O|Z)$, where $O = (o_1, ..., o_{n_s})$ is a sequence of random variables representing the overclustering indices for each cell, such that $o_i \in \{1, ..., n_o\}$.

Let $\{\Phi_j^o\}_{j=1}^{n_o}$ be the set of trainable overclustering centers. We initialize these points, after the pretraining phase, by applying k-means to Z. The first distribution $p(O|Z)$ measures the soft overclustering assignments and factorizes as $p(O|Z) = \prod_{i=1}^{n_s} p(o_i\,|\,z_i)$. We denote by p_{ij}^o the probability $p(o_i = j|z_i)$, which is computed as described by:

$$p_{ij}^o = \frac{(1 + \left\|z_i - \Phi_j^o\right\|^2)^{-1}}{\sum_{j'}(1 + \left\|z_i - \Phi_{j'}^o\right\|^2)^{-1}}. \tag{8}$$

The second distribution $q(O|Z)$ models the target overclustering assignments. The factorization of this distribution is $q(O|Z) = \prod_{i=1}^{n_s} q(o_i\,|\,z_i)$. We denote by q_{ij}^o the probability value $q(o_i = j|z_i)$. Let λ_i^1 (λ_i^2, respectively) be the highest (second highest, respectively) assignment score of the set $\{p_{ij}^o\}_{j=1}^{n_o}$. We denote by $\Omega_o = \{x_i \in \mathbb{R}^{n_g} \mid \lambda_i^1 - \lambda_i^2 \geqslant \beta_o\}$ the set of reliable samples for the overclustering task and β_o is a fixed threshold. We compute q_{ij}^o as described by:

$$q_{ij}^o = \begin{cases} 1 \text{ if } x_i \in \Omega_o \text{ and } j = \arg\max_{j'}(p_{ij'}^o), \\ 0 \text{ if } x_i \in \Omega_o \text{ and } j \neq \arg\max_{j'}(p_{ij'}^o), \\ p_{ij}^o \text{ if } x_i \notin \Omega_o. \end{cases} \tag{9}$$

The overclustering loss $\mathcal{L}_o$ is the KL (Kullback–Leibler) divergence between the soft assignment distribution $p(O|Z)$ and the target distribution $q(O|Z)$ as described by:

$$\mathcal{L}_o = \text{KL}\big(q(O|Z)||p(O|Z)\big) = \sum_{i=1}^{n_s}\sum_{j=1}^{n_o} q_{ij}^o \log(\frac{q_{ij}^o}{p_{ij}^o}). \tag{10}$$

Adversarial Module

It handles the global configuration of the latent space. Our adversarial mechanism is responsible for gradually transforming the globally-curved structures into convex ones. During the training process, the number of reliable samples for clustering increases progressively. We train the adversarial module using samples from the set Ω_c. As a result, the adversarial module gradually enforces the convexity constraint from the core to the outer parts of the manifolds.

Definition 1. *A convex combination x of k points $x_1, ..., x_k$ is a linear combination of these points: $x = \theta_1 x_1 + ... + \theta_k x_k$, where $\theta_1, ..., \theta_k \in \mathbb{R}_+$ and $\sum_i \theta_i = 1$.*

Let C_k be the set of samples belonging to the k^{th} cluster. We compute the clustering index of the sample x_i using the formula $\max_j(p_{ij})$. Given the representations of m cells from the k^{th} cluster, $x_1, ..., x_m \in \Omega_c \cap C_k$, we randomly select m coefficients $\theta_1, ..., \theta_m \in \mathbb{R}_+$, such that $\sum_i \theta_i = 1$. We construct the point $\hat{x}_{\text{cvx}}^{(k)}$ as a convex combination of the points $x_1, ..., x_m$ in the latent space, as articulated by:

$$\hat{x}_{\text{cvx}}^{(k)} = f_D^{(2)}(\sum_{i=1}^{m} \theta_i f_E(x_i)). \tag{11}$$

For each cluster C_k, we generate n_{cvx} points according to Eq. (11). We obtain different samples by randomly selecting the initial points $x_1, ..., x_m$. We denote by Γ_k the set of samples generated for the k^{th} cluster. Each discriminator $f_{\text{adv}}^{(k)}$ is trained to distinguish between real samples reconstructed from the set $\Omega_c \cap C_k$ and generated samples belonging to the set Γ_k. The loss function $\mathcal{L}_d^{(k)}$ of the k^{th} discriminator is:

$$\mathcal{L}_d^{(k)} = \mathbb{E}_{\hat{x}}\big[\log(1-f_{\text{adv}}^{(k)}(\hat{x}))\big] + \mathbb{E}_{\hat{x}_{\text{cvx}}}\big[\log(f_{\text{adv}}^{(k)}(\hat{x}_{\text{cvx}}^{(k)}))\big]. \tag{12}$$

We use the graph auto-encoder constrained to the second decoding head as the generator for the adversarial module. We train the generator to fool the discriminator into considering the generated points $\hat{x}_{\text{cvx}}$ as real reconstruction samples. The loss function of the generator $\mathcal{L}_g$ is described by:

$$\mathcal{L}_g = \sum_{k=1}^{n_c} \mathbb{E}_{\hat{x}_{\text{cvx}}}\big[\log(1 - f_{\text{adv}}^{(k)}(\hat{x}_{\text{cvx}}^{(k)}))\big]. \tag{13}$$

The generator is trained to incorporate convex combinations of the latent codes into the embedded manifolds. The gradual inclusion of each cluster's convex hull leads to smoother latent structures and a higher level of convexity. Clustering convex sets is relatively straightforward because they have well-defined boundaries. Thus, it is easy to separate the points inside the set from the points outside it. Theorem 1 shows that disjoint convex sets are linearly separable.

Theorem 1. ***Hyperplane separation [Boyd* et al., *2004]****: if A and B are disjoint nonempty convex sets from $\mathbb{R}^d$, then there exists a vector $u \in \mathbb{R}^d - \{\mathbf{0}\}$ and a number $c \in \mathbb{R}$, such that:*

$$u^T x \geqslant c \quad \forall x \in A \quad \textit{and} \quad u^T x \leqslant c \quad \forall x \in B,$$

and the hyperplane $\{x \mid u^T x \geqslant c\}$ separates A and B.

3.2 Training Strategy

Our model undergoes a two-phase training process. We pretrain for T_1 iterations. For the first phase, we update the autoencoder parameters to minimize the self-supervision function $\mathcal{L}_{\text{SS}}$. For the second phase, we alternate between three training steps. The first step consists of minimizing the loss function $\mathcal{L}_{\text{SS}} + \gamma_c \mathcal{L}_c$. In the second step, we minimize the loss function $\mathcal{L}_{\text{SS}} + \gamma_o \mathcal{L}_o$. In the third step, we minimize the loss function $\mathcal{L}_{\text{SS}} + \gamma_g \mathcal{L}_g$. The hyperparameters γ_c, γ_o, and γ_g control the trade-off between self-supervision and the remaining tasks (i.e., clustering, overclustering, and adversarial training). Moreover, we train the discriminators at each step by minimizing the loss functions $\mathcal{L}_d^{(k)}$. Adam optimizer is used for all training phases and steps. The second phase ends when the average ID of all manifolds becomes equal to the LID. We illustrate the framework of the second phase in Figure 1. We provide our algorithm and its computational complexity analysis in Appendix C and Appendix D, respectively.

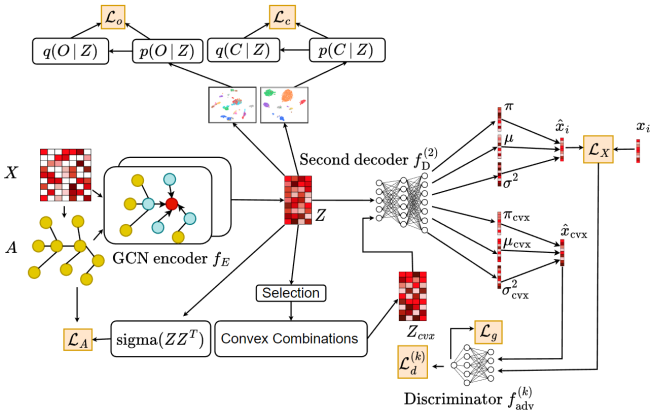

Figure 1: Illustration of the clustering phase of scTCM. The matrix $Z_{\text{cvx}} \in \mathbb{R}^{n_{\text{cvx}} \times d}$ represents the convex combinations of the latent codes. It worth to highlight that we use a discriminator for each cluster. We present a single discriminator to simplify the illustration.

4 Experiments

We carry out multiple experiments on eight real-world scRNA-seq datasets of various species. These datasets are collected using popular sequencing platforms. Information on the data and preprocessing steps can be found in Appendix E. We compare the performance of our method, **scTCM** (deep clustering of **s**ingle-**c**ell RNA-seq data: **T**oward **C**onvex **M**anifolds), against state-of-the-art clustering techniques for scRNA-seq. Our comparison includes three deep embedding methods (scDCA [Eraslan *et al.*, 2019], scGMAI [Yu *et al.*, 2021], and scGNN [Wang *et al.*, 2021]), three deep clustering methods (scziDesk [Chen *et al.*, 2020], scDC [Tian *et al.*, 2019], and scDCC [Tian *et al.*, 2021]), and three deep graph clustering methods (scTAG [Yu *et al.*, 2022], scDSC [Gan *et al.*, 2022], and scGAE [Luo *et al.*, 2021]). All of these approaches are discussed in the related work section. We use the official code provided by the authors for each baseline and we tune the hyperparameters if no specific instructions are given. To ensure reproducibility, our code is available on https://github.com/MMAMAR/scTConvexMan.

The geometric configuration is evaluated by measuring the ID and LID of the latent manifolds. A comprehensive explanation of the strategies applied to estimate ID and LID can be found in Appendix F. We compare the efficiency of the different models based on their execution time in seconds. All of our experiments are performed under consistent hardware and software setups, as described in Appendix G. In Appendix H, we thoroughly outline the design and configuration of our ap-

Metric	Dataset	Deep Graph Clustering				Deep Clustering			Deep Embedding		
		scTCM	scTAG	scGAE	scDSC	scziDesk	scDC	scDCC	scDCA	scGMAI	scGNN
ACC	Muraro	96.41	89.35	89.75	86.52	73.66	75.12	84.35	73.61	61.40	66.30
	Plasschaert	97.46	89.01	83.83	91.76	93.56	52.60	61.34	62.66	48.07	53.56
	QX_LM	99.77	97.77	75.23	78.49	96.67	75.06	82.04	74.52	60.88	63.54
	QS_Diaph	99.08	98.39	55.51	94.94	97.82	71.84	71.26	72.30	47.93	68.04
	QS_Heart	98.95	96.13	89.46	92.05	94.46	55.53	68.23	70.72	60.09	63.50
	QS_LM	99.63	99.17	62.56	86.06	97.80	60.64	74.68	58.53	57.98	85.41
	Wang_Lung	99.30	95.14	77.11	96.02	97.59	98.97	90.51	98.62	38.56	97.97
	Young	85.30	81.35	70.72	67.88	79.81	58.75	67.41	64.57	42.00	50.30
	Average	96.99	93.28	75.52	86.71	91.42	68.56	74.97	71.94	52.11	68.57
NMI	Muraro	89.80	82.92	83.53	85.00	77.76	75.49	83.84	76.21	71.68	64.10
	Plasschaert	90.54	73.79	68.44	80.02	79.80	61.22	64.97	65.52	57.11	63.17
	QX_LM	99.01	93.64	79.39	80.61	91.28	84.05	87.98	83.69	76.39	74.43
	QS_Diaph	95.93	93.28	68.04	91.67	91.40	78.07	81.04	78.38	68.36	71.34
	QS_Heart	95.90	89.43	78.39	89.58	87.75	65.31	72.75	70.61	69.41	73.20
	QS_LM	98.40	96.44	72.71	82.73	92.15	70.48	79.94	71.71	71.98	80.35
	Wang_Lung	92.30	71.50	58.62	74.54	81.46	90.11	57.99	87.38	34.32	84.61
	Young	84.86	79.16	65.27	72.18	76.72	61.78	67.66	63.66	49.54	40.03
	Average	93.34	85.02	71.79	82.04	84.79	73.31	74.52	74.64	62.34	68.90
ARI	Muraro	93.72	87.02	88.21	89.51	66.30	66.09	74.10	64.59	51.32	49.19
	Plasschaert	94.70	76.97	70.64	86.99	85.88	40.70	49.76	50.69	57.11	46.40
	QX_LM	99.61	95.45	63.31	74.79	91.28	75.00	80.23	74.43	50.61	58.33
	QS_Diaph	97.88	96.37	42.90	96.15	93.69	64.79	66.03	64.98	41.11	56.09
	QS_Heart	98.15	94.09	84.89	94.78	92.08	46.73	58.18	59.87	43.68	56.74
	QS_LM	99.32	98.11	53.79	90.89	94.29	53.84	65.77	54.44	48.99	79.68
	Wang_Lung	96.90	80.14	59.98	83.48	89.74	95.50	64.10	94.02	13.25	91.31
	Young	79.74	70.61	58.87	54.61	66.81	44.69	52.85	50.06	32.87	28.98
	Average	95.00	87.34	65.32	83.90	85.00	60.91	63.87	64.13	42.36	58.34
Time	Muraro	330	54	61	88	106	74	83	240	152	733
	Plasschaert	766	159	490	242	378	165	216	546	589	1725
	QX_LM	387	70	189	120	278	73	82	198	321	816
	QS_Diaph	227	50	46	64	98	58	73	52	78	341
	QS_Heart	576	87	221	134	406	164	273	261	300	1475
	QS_LM	240	63	46	52	115	56	78	66	86	413
	Wang_Lung	956	519	718	150	632	307	190	202	572	1867
	Young	647	213	351	202	601	223	203	432	367	1243

Table 1: Clustering results on eight scRNA-seq datasets. Best methods in red and second best in blue.

Method		Muraro				Plasschaert				Young			
Adverserial	Overclustering	ACC	NMI	ARI	Time	ACC	NMI	ARI	Time	ACC	NMI	ARI	Time
✗	✗	93.49	86.80	90.75	**101**	96.60	88.51	93.06	**646**	82.11	82.41	74.33	**548**
✗	✓	95.75	88.38	92.63	120	97.18	89.26	94.04	649	84.57	84.08	75.71	577
✓	✗	93.87	86.93	91.54	325	97.00	89.29	93.44	732	84.22	84.01	77.59	632
✓	✓	**96.41**	**89.80**	**93.72**	330	**97.46**	**90.54**	**94.70**	766	**85.30**	**84.86**	**79.74**	647

Table 2: Impact of the proposed adversarial and overclustering mechanisms on the clustering performance of scTCM. Best results in bold.

proach, including the architecture, the learning rates, and any other hyperparameters. In particular, our approach has three data-dependent hyperparameters (β_c, β_o, and n_o), which are fixed using grid search as explained in Appendix H. We hold the remaining design choices constant across all the datasets. A qualitative evaluation is included in Appendix I.

Clustering results. Table 1 shows the clustering performance of our method against multiple state-of-the-art techniques. For each method, we report the average results of ten executions. We observe that the deep clustering approaches generally outperform the deep embedding methods. This finding underscores the importance of combining embedded learning and clustering. Additionally, we can see that the deep graph clustering methods tend to perform better than the deep clustering approaches. This finding supports the significance of extracting the graph from the gene expression count matrix to use it in the encoding process. Last but not least,

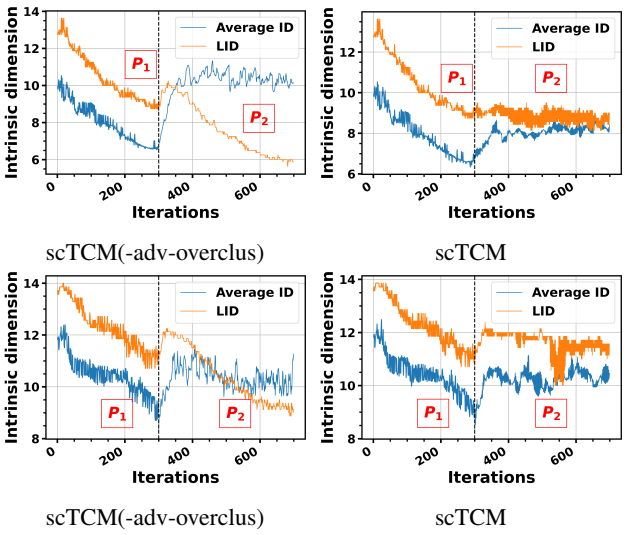

Figure 2: ID and LID of scTCM(-adv-overclus) and scTCM on Muraro (first row) and Plasschaert (second row). P_1 stands for the first training phase (self-supervision) and P_2 stands for the second training phase (pseudo-supervision).

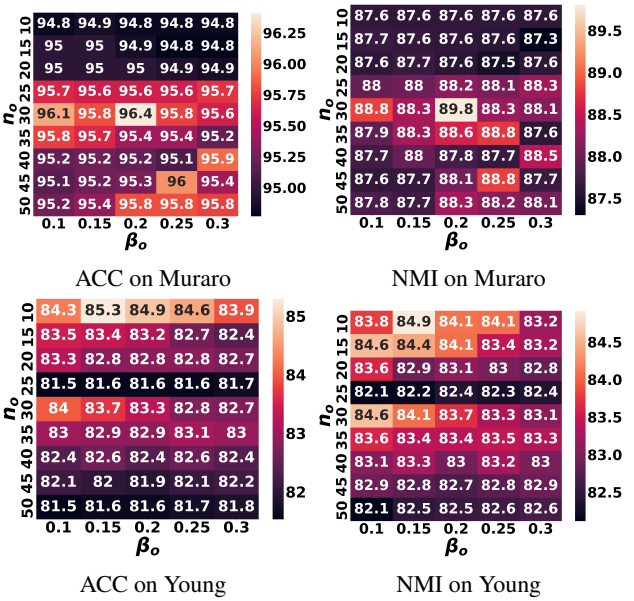

Figure 3: Sensitivity of scTCM to the hyperparameters n_o and β_o in terms of ACC and NMI.

the proposed method yields the best performance in terms of ACC, NMI, and ARI compared to all the other methods, across all the datasets. On average, the NMI and ARI scores of scTCM are more than 7% higher than the scores of the second-best method (that is, scTAG). Unlike previous methods, scTCM has two mechanisms to avoid the coarse geometric transition between the pretraining and clustering phases.

Geometric study. In Fig. 2, we investigate the evolution of the latent manifolds during the training process. To this end, we build a modified version of scTCM. We obtain this version, denoted scTCM(-adv-overclus), by ablating the adversarial module and the overclustering loss. For scTCM(-adv-overclus), we observe that the difference between ID and LID decreases rapidly after the pretraining phase. Within a few iterations, the latent manifolds undergo a coarse geometric transformation: from curved (evidenced by a significant difference between ID and LID) and low-dimensional structures to flattened and higher-dimensional ones. In contrast to scTCM(-adv-overclus), scTCM has a less pronounced decrease in LID and a more moderate increase in ID even after a considerable number of iterations. The proposed mechanisms *gradually* flatten the latent manifolds. Unlike scTCM(-adv-overclus), the clustering-oriented representations of scTCM (Euclidean geometry perspective) are slowly constructed while protecting the curves from twisting.

Ablation study. In this experiment, we analyze the impact of our contributions on clustering performance. First, we can see from Table 2 that the overclustering mechanism improves the clustering results consistently. Second, performing adversarial training also brings consistent improvement compared with scTCM(-adv-overclus). It is worth noting that the full model regularly outperforms all the other variants. These results validate the synergy of the proposed mechanisms. While overclustering flattens the local structures, adversarial training transforms the global configuration into a convex one. In addition, we can see from Table 2 that our mechanisms do not induce significant run-time overhead. These results align with the computational complexity provided in Appendix D.

Sensitivity analysis. We analyze the sensitivity of our model to the data-dependent hyperparameters (β_c, β_o, and n_o). We hold the remaining design decisions constant across all datasets. In Fig. 3, we study the sensitivity to the hyperparameters of the overclustering loss (i.e., β_o and n_o). In Appendix J, we study the sensitivity to the hyperparameter of the clustering loss (i.e., β_c). We found that the proposed model produces consistent results for a wide range of values. Especially, fixing n_o, scTCM yields stable clustering results as β_o varies. As we can see, the ACC and NMI spike when n_o is equal to 30 and 10 on Muraro and Young, respectively.

5 Conclusion

In this research, we propose a single-cell deep clustering model that addresses the coarse geometric transition between the pretraining and clustering phases. Our approach utilizes two mechanisms to make the latent representations suitable for euclidean-based clustering. More precisely, we focus on two geometric aspects: local flatness and global convexity. The first mechanism is a local geometric transformation that flattens the local curves without destroying the global structures. It consists of minimizing an overclustering loss. The second mechanism adjusts the global geometric configuration by transforming the latent manifolds into convex ones through an adversarial loss. Our method shows higher clustering performance against state-of-the-art approaches for scRNA-seq data. Empirical results provide strong evidence that this performance is imputed to the proposed mechanisms and particularly their ability to tackle the coarse geometric transition between pretraining and clustering.

Acknowledgments

This work has been supported by Research Grants from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[Boyd *et al.*, 2004] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Brady *et al.*, 2017] Samuel W Brady, Jasmine A McQuerry, Yi Qiao, Stephen R Piccolo, Gajendra Shrestha, David F Jenkins, Ryan M Layer, Brent S Pedersen, Ryan H Miller, Amanda Esch, et al. Combating subclonal evolution of resistant cancer phenotypes. *Nature communications*, 8(1):1–15, 2017.

[Chen *et al.*, 2020] Liang Chen, Weinan Wang, Yuyao Zhai, and Minghua Deng. Deep soft k-means clustering with self-training for single-cell rna sequence data. *NAR genomics and bioinformatics*, 2(2):lqaa039, 2020.

[Eraslan *et al.*, 2019] 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):1–14, 2019.

[Ester *et al.*, 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[Gan *et al.*, 2022] Yanglan Gan, Xingyu Huang, Guobing Zou, Shuigeng Zhou, and Jihong Guan. Deep structural clustering for single-cell rna-seq data jointly through autoencoder and graph neural network. *Briefings in Bioinformatics*, 23(2):bbac018, 2022.

[Huang *et al.*, 2018] 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.

[Johnson, 1967] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *NeurIPS workshop*, volume 32, pages 1–3, 2016.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Lin *et al.*, 2017] Peijie Lin, Michael Troup, and Joshua WK Ho. Cidr: Ultrafast and accurate clustering through imputation for single-cell rna-seq data. *Genome biology*, 18(1):1–11, 2017.

[Luo *et al.*, 2021] Zixiang Luo, Chenyu Xu, Zhen Zhang, and Wenfei Jin. A topology-preserving dimensionality reduction method for single-cell rna-seq data using graph autoencoder. *Scientific reports*, 11(1):1–8, 2021.

[Mrabah *et al.*, 2022] Nairouz Mrabah, Mohamed Bouguessa, and Riadh Ksantini. Escaping feature twist: A variational graph auto-encoder for node clustering. In *IJCAI*, pages 3351–3357, 2022.

[Ocasio *et al.*, 2019] Jennifer Karin Ocasio, Benjamin Babcock, Daniel Malawsky, Seth J Weir, Lipin Loo, Jeremy M Simon, Mark J Zylka, Duhyeong Hwang, Taylor Dismuke, Marina Sokolsky, et al. scrna-seq in medulloblastoma shows cellular heterogeneity and lineage expansion support resistance to shh inhibitor therapy. *Nature communications*, 10(1):1–17, 2019.

[Park and Zhao, 2018] Seyoung Park and Hongyu Zhao. Spectral clustering based on learning similarity matrix. *Bioinformatics*, 34(12):2069–2076, 2018.

[Saviano *et al.*, 2020] Antonio Saviano, Neil C Henderson, and Thomas F Baumert. Single-cell genomics and spatial transcriptomics: Discovery of novel cell states and cellular interactions in liver physiology and disease biology. *Journal of hepatology*, 73(5):1219–1230, 2020.

[Shalek *et al.*, 2014] Alex K Shalek, Rahul Satija, Joe Shuga, John J Trombetta, Dave Gennert, Diana Lu, Peilin Chen, Rona S Gertner, Jellert T Gaublomme, Nir Yosef, et al. Single-cell rna-seq reveals dynamic paracrine control of cellular variation. *Nature*, 510(7505):363–369, 2014.

[Tian *et al.*, 2019] Tian Tian, Ji Wan, Qi Song, and Zhi Wei. Clustering single-cell rna-seq data with a model-based deep learning approach. *Nature Machine Intelligence*, 1(4):191–198, 2019.

[Tian *et al.*, 2021] Tian Tian, Jie Zhang, Xiang Lin, Zhi Wei, and Hakon Hakonarson. Model-based deep embedding for constrained clustering analysis of single cell rna-seq data. *Nature communications*, 12(1):1–12, 2021.

[Tirosh *et al.*, 2016] Itay Tirosh, Benjamin Izar, Sanjay M Prakadan, Marc H Wadsworth, Daniel Treacy, John J Trombetta, Asaf Rotem, Christopher Rodman, Christine Lian, George Murphy, et al. Dissecting the multicellular ecosystem of metastatic melanoma by single-cell rna-seq. *Science*, 352(6282):189–196, 2016.

[Treutlein *et al.*, 2014] Barbara Treutlein, Doug G Brownfield, Angela R Wu, Norma F Neff, Gary L Mantalas, F Hernan Espinoza, Tushar J Desai, Mark A Krasnow, and Stephen R Quake. Reconstructing lineage hierarchies of the distal lung epithelium using single-cell rna-seq. *Nature*, 509(7500):371–375, 2014.

[van Dijk *et al.*, 2017] David van Dijk, Juozas Nainys, Roshan Sharma, Pooja Kaithail, Ambrose J Carr, Kevin R Moon, Linas Mazutis, Guy Wolf, Smita Krishnaswamy, and Dana Pe'er. Magic: A diffusion-based imputation method reveals gene-gene interactions in single-cell rna-sequencing data. *BioRxiv*, page 111591, 2017.

[Wang *et al.*, 2017] Bo Wang, Junjie Zhu, Emma Pierson, Daniele Ramazzotti, and Serafim Batzoglou. Visualization and analysis of single-cell rna-seq data by kernel-based similarity learning. *Nature methods*, 14(4):414–416, 2017.

[Wang *et al.*, 2021] 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):1–11, 2021.

[Xie *et al.*, 2016] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016.

[Yu *et al.*, 2021] Bin Yu, Chen Chen, Ren Qi, Ruiqing Zheng, Patrick J Skillman-Lawrence, Xiaolin Wang, Anjun Ma, and Haiming Gu. scgmai: a gaussian mixture model for clustering single-cell rna-seq data based on deep autoencoder. *Briefings in Bioinformatics*, 22(4):316, 2021.

[Yu *et al.*, 2022] Zhuohan Yu, Yifu Lu, Yunhe Wang, Fan Tang, Ka-Chun Wong, and Xiangtao Li. Zinb-based graph embedding autoencoder for single-cell rna-seq interpretations. *AAAI*, 36(4):4671–4679, 2022.