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

1 Appendix A: Additional Related Work

We discuss additional related methods, which are not specific to the single-cell field of research.

In [Facco *et al.*, 2017], the authors have proposed TwoNN, an effective estimation of ID by computing the distances between each point and its first and second closest neighbors. In another work, the authors of [Ansuini *et al.*, 2019] have conducted an investigation to examine the behavior of ID and LID under the supervised learning paradigm. They have found that training a neural network based on the standard cross-entropy loss function leads to the emergence of *curved* latent manifolds with low intrinsic dimensions.

To address the coarse geometric transition between pretraining and clustering, the authors of [Mrabah *et al.*, 2022] have proposed FT-VGAE (Variational Graph Auto-Encoder supplied with a mechanism against Feature Twist). FT-VGAE has three training phases. The first training phase consists of reconstructing the graph structure using the inner product operation. The second training phase performs neighborhood-level self-supervised learning. In particular, neighbors of the latent codes are exploited for reconstructing the edges. The third training phase performs embedding clustering and reconstructs the input graph edges by considering the similarities between the latent codes and their centers.

Despite the progress achieved by FT-VGAE in clustering performance, this model has four limitations compared to our approach. First, FT-VGAE performs a unimodal decoding process. It only reconstructs the graph structure without considering the Euclidean representations. Unlike FT-VGAE, our approach reconstructs two data modalities (Euclidean representations and structural information), which can provide complementary views to the clustering task. Second, FT-VGAE has an additional training phase compared to our approach. The three-phase strategy reduces the efficiency of FT-VGAE compared to our two-phase methods. Third, FT-VGAE ignores the importance of the latent structures convexity and does not consider the interaction between local and global geometric configurations during the training process. Unlike FT-VGAE, our approach has an adversarial mechanism to adjust the global geometric configuration. This mechanism gradually transforms the latent structures into convex ones. Last but not least, FT-VGAE is not appropriate for clustering scRNA-seq data because it is not designed to capture the characteristics of the gene expression count matrix (discreteness, zero-inflation, and over-dispersion). Unlike FT-VGAE, our approach integrates the ZINB distribution to learn the gene expression profiles.

2 Appendix B: ZINB mass function

The ZINB mass function h_{ZINB} is expressed as follows:

$$h_{\text{ZINB}}(x\,|\pi,\, r,\, p) = \pi\,\delta_0(x) + (1-\pi)\,h_{\text{NB}}(x\,|\,r,\,p),$$

$$\delta_0(x) = \begin{cases} 1 & \text{if} \quad x = 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$h_{\text{NB}}(x\,|\,r,\,p) = \binom{x+r-1}{x}\,(1-p)^r\,p^x,$$

where π is the probability of a true zero; δ_0 is the indicator function; r and p are the parameters of a Negative Binomial (NB) distribution described by the mass function $h_{\text{NB}}(x\,|\,r,\,p)$, such that r represents the number of successes and p is the probability of success.

We can express the ZINB mass function h_{ZINB} using the mean μ and variance σ^2 of the associated negative binomial instead of the p and r as follows:

$$h_{\text{ZINB}}(x\,|\pi,\,\mu,\,\sigma^2) = \pi\,\delta_0(x) + (1-\pi)\,h_{\text{NB}}(x\,|\,\mu,\,\sigma^2),$$

$$h_{\text{NB}}(x\,|\,\mu,\,\sigma^2) = \frac{\Gamma(x+\sigma^2-\mu)}{x!\,\Gamma(\sigma^2-\mu)}\left(\frac{\mu}{\sigma^2}\right)^{\sigma^2-\mu}\left(\frac{\sigma^2-\mu}{\sigma^2}\right)^{x}.$$

The formulation in $h_{\text{NB}}(x\,|\,\mu,\,\sigma^2)$ is obtained by setting $p = \frac{\sigma^2-\mu}{\sigma^2}$ and $r = \sigma^2 - \mu$.

3 Appendix C: Proposed Algorithm

The training of our model is described in Algorithm 1.

4 Appendix D: Computational Complexity

We perform the complexity analysis under three standard assumptions: (1) the training weights of the layers (encoder, decoder, and discriminators) are assumed to have the same dimension d, (2) we assume that the clustering phase requires T_2 iterations and (3) we assume that the number of

Algorithm 1 Training strategy of our model

Input: initial graph: $\mathcal{G}$, # of iterations: T_1, # of overclusters: n_o, # of clusters: n_c, overclustering threshold: β_o, clustering threshold: β_c, balancing hyperparameters: γ_c, γ_o, γ_g

for $i = 0$ **to** T_1 **do**
  $Z \leftarrow f_E(A, X)$;
  $\hat{A} = f_D^{(1)}(Z)$;
  $\hat{X} = f_D^{(2)}(Z)$;
  Compute L_{SS} according to Eq. (4);
  Update W to minimize L_{SS};
end for
Compute ID and LID;
$i \leftarrow 0$;
Ω_c, $\Omega_o \leftarrow \varnothing$;
Compute $\left\{\Phi_j^c\right\}_{j=1}^{n_c}$ by performing spectral clustering on A and projecting the centers to the latent space;
Compute $\left\{\Phi_j^o\right\}_{j=1}^{n_o}$ by performing k-means on the latent codes Z;
while ID $<$ LID **do**
  $Z \leftarrow f_E(A, X)$;
  $\hat{A} = f_D^{(1)}(Z)$;
  $\hat{X} = f_D^{(2)}(Z)$;
  Compute L_{SS} according to Eq. (4);
  for $k = 1$ **to** n_c **do**
    Construct n_{cvx} points $\hat{x}_{\text{cvx}}^{(k)}$ according to Eq. (11);
    Compute $\mathcal{L}_d^{(k)}$ according to Eq. (12);
    Update $W_{\text{adv}}^{(k)}$ to minimize $\mathcal{L}_d^{(k)}$ using Adam optimizer;
  end for
  if $i \,\%\, 3 == 0$ **then**
    Compute (p_{ij}^c) according to Eq. (5);
    $\Omega_c \leftarrow \left\{x_i \in \mathbb{R}^{n_g} \,|\, \tau_i^1 - \tau_i^2 \geqslant \beta_c\right\}$;
    Compute (q_{ij}^c) according to Eq. (6);
    Compute $\mathcal{L}_c$ according to Eq. (7);
    Update $\left\{W, \left\{\Phi_j^c\right\}\right\}$ to minimize $\mathcal{L}_{\text{SS}} + \gamma_c \mathcal{L}_c$ using Adam optimizer;
  else if $i \,\%\, 3 == 1$ **then**
    Compute (p_{ij}^o) according to Eq. (8);
    $\Omega_o \leftarrow \left\{x_i \in \mathbb{R}^{n_g} \,|\, \lambda_i^1 - \lambda_i^2 \geqslant \beta_o\right\}$;
    Compute (q_{ij}^o) according to Eq. (9);
    Compute $\mathcal{L}_o$ according to Eq. (10);
    Update $\left\{W, \left\{\Phi_j^o\right\}\right\}$ to minimize $\mathcal{L}_{\text{SS}} + \gamma_o \mathcal{L}_o$ using Adam optimizer;
  else if $i \,\%\, 3 == 2$ **then**
    Compute $\mathcal{L}_g$ according to Eq. (13);
    Update W to minimize $\mathcal{L}_{\text{SS}} + \gamma_g \mathcal{L}_g$ using Adam optimizer;
  end if
  Compute ID and LID;
  $i \leftarrow$ i + 1;
end while
return (p_{ij}^c);

Dataset	Muraro	Plasschaert	QX_LM	QS_Diaph	QS_Heart	QS_LM	Wang_Lung	Young
# Cells	2122	6977	3909	870	4365	1090	9519	5685
# Genes	19046	28205	23341	23341	23341	23341	14561	33658
# Classes	9	8	6	5	8	6	2	11
Platform	CEL-seq2	inDrop	10x	Smart-seq2	Smart-seq2	Smart-seq2	10x	10x

Table 1: Dataset statistics.

cells $n_s \gg d$, n_g, n_c, n_o, n_{cvx}, k, m, L_E, L_D, and L_{adv}. The computational complexity to construct the graph $\mathcal{G}$ using k-NN is $\mathcal{O}(2^d\, k\, n_s \log(n_s))$. The time complexity to compute the latent codes Z using the GCN encoding layers is $\mathcal{O}(d\, L_E\, (k+d)\, n_s)$. The time complexity to compute the generated structure $\hat{A}$ is $\mathcal{O}(d\, n_s^2)$. The computational complexity to compute the generated gene expression count matrix $\hat{X}$ is $\mathcal{O}\big(k\, d\, L_E\, n_s + d^2\, (L_E + L_D)\, n_s\big)$. Accordingly, the time complexity to compute self-supervision loss L_{SS} is also $\mathcal{O}(d\, n_s^2)$, and the computational complexity of the complete pretraining phase is $\mathcal{O}(d\, T_1\, n_s^2)$. The computational complexity to compute the clustering loss L_c is $\mathcal{O}\big(d\, (L_E\, (k+d) + n_c)\, n_s\big)$. The computational complexity to compute the overclustering loss L_o is $\mathcal{O}\big(d\, (L_E\, (k+d) + n_o)\, n_s\big)$. The computational complexity to compute the generator loss L_g is $\mathcal{O}\big(k\, d\, L_E\, n_s + d^2\, (L_{\text{adv}} + L_D)\, n_c\, n_{\text{cvx}} + m\, d\, n_c\, n_{\text{cvx}}\big)$. The computational complexity to compute each of the discriminator losses $L_d^{(k)}$ is $\mathcal{O}\big(k\, d\, L_E\, n_s + d^2\, (L_E + L_D)\, n_s + d^2\, (L_{\text{adv}} + L_D)\, n_c\, n_{\text{cvx}} + m\, d\, n_c\, n_{\text{cvx}}\big)$. The computational complexity of the complete clustering phase is $\mathcal{O}(d\, T_2\, n_s^2)$ because the second phase is dominated by the quadratic complexity of the structure reconstruction loss function L_X. The total computational complexity of Algorithm 1 is $\mathcal{O}(d\, (T_1 + T_2)\, n_s^2)$.

5 Appendix E: Data Description and Preprocessing

We used eight real datasatest for the expriments. Muraro [Muraro *et al.*, 2016], Plasschaert [Plasschaert *et al.*, 2018], QX_LM [Consortium and others, 2018], QS_Diaph [Consortium and others, 2018], QS_Heart [Consortium and others, 2018], QS_LM [Consortium and others, 2018], Wang_Lung [Wang *et al.*, 2018],Young [Young *et al.*, 2018]. In Table 1, we summarizes the relevant informations of these datasets. More details are provided as follows:

- **Muraro**: This dataset contains the gene expression levels of human pancreas at the single-cell resolution. The transcriptome is made up of 2,122 cells and 19,046 genes.
- **Plasschaert**: This dataset contains the gene expression profiles of mouse tracheal epithelial cells. It is composed of 6,977 cells and 28,205 genes.
- **QX_LM**: This dataset contains the gene expression profiles of mouse limb muscle cells processed on the 10X gene sequencing platform. The number of cells is 3,909 and the number of genes is 23,341.
- **QS_Diaph**: This dataset contains the gene expression profiles of mouse diaphragm cells processed on the Smart-seq2 genome sequencing platform. The number of cells is 870 and the number of genes is 23,341.
- **QS_Heart**: This dataset contains the gene expression levels of mouse heart cells processed on the Smart-seq2 sequencing platform. The number of cells is 4,365 and the number of genes is 23,341.
- **QS_LM**: This dataset contains the gene expression levels of mouse heart cells processed on the Smart-seq2 gene sequencing platform. The number of cells is 1,090 and the number of genes is 23,341.
- **Wang_Lung**: This dataset contains the pulmonary alveolar epithelium gene expression levels processed by the gene sequencing platform 10X. It is made up of 9,519 cells and 14,561 genes.
- **Young**: This dataset contains the cell transcriptome of human renal tumors and normal tissues from adult, pediatric and fetal renal samples processed by the gene sequencing platform 10X. It is made up of 5,685 cells and 33,658 genes.

Considering the characteristics of the gene expression count matrix, we perform three widely-applied prepossessing steps [Hao *et al.*, 2021], [Wang *et al.*, 2021], [Yu *et al.*, 2022]. First, we remove the underrepresented genes that are only present in less than 1% of the cells to alleviate the high dropout rate. Second, we normalize the data. Formally, we rescale the gene expression values for each cell by dividing by the associated library size, which is the total number of read counts for this cell. The obtained matrix $\bar{X} = (\bar{x}_{ij}) \in \mathbb{R}^{n_s \times n_g}$ is expressed according to:

$$\bar{x}_{ij} = \log\Big(\text{Med}(X)\, \frac{x_{ij}}{\sum_{j=1}^{n_g} x_{ij}}\Big), \tag{1}$$

where $\text{Med}(X)$ denotes the median of the gene expression values among all cells. The log transformation is used to smooth the normalized values. Third, we filter out the low variable genes based on the normalized dispersion values.

6 Appendix F: Intrinsic Dimension and Linear Intrinsic Dimension

ID (Intrinsic Dimension) and LID (Linear Intrinsic Dimension) assess the geometric transformation during the training process. The first metric (i.e., ID) describes the minimum number of parameters required to precisely capture the principal features. We estimate the ID of the latent manifolds based on TwoNN [Facco *et al.*, 2017]; a recent estimator that only considers the two nearest neighbors of each sample. TwoNN

Parameter	Muraro	Plasschaert	QX_LM	QS_Diaph	QS_Heart	QS_LM	Wang_Lung	Young
Clustering threshold: β_c	0.7	0.75	0.92	0.88	0.93	0.92	0.92	0.75
# of overclusters: n_o	30	10	30	10	25	10	7	10
Overclustering threshold: β_o	0.2	0.25	0.1	0.2	0.1	0.3	0.1	0.15

Table 2: Data-dependent hyperparameters.

is computationally efficient and does not require the data density to effectively estimate the ID of highly-curved and non-uniformly sampled manifolds.

Let $X = \{x_i\}_{i=1}^{N}$ be a set of N points uniformly sampled from a data manifold, whose intrinsic dimension is equal to d. $r_1(i)$ and $r_2(i)$ are the distances between the sample x_i and its first and second nearest neighbors, respectively, among the set X. Let μ_i be the ratio between $r_2(i)$ and $r_1(i)$ (i.e., $\mu_i = r_2(i)/r_1(i)$). If the density between each point x_i and its second neighbor is constant, it has been proved [Facco *et al.*, 2017] that the ratio of a sample μ_i follows the Pareto distribution with a scale parameter equal to 1 and a shape parameter equal to d. Let $f(.|d)$ be the probability density function and $F(.|d)$ the cumulative distribution function of this Pareto distribution such that:

$$f(\mu_i|d) = d\,\mu_i^{-(d+1)} 1_{[1,+\infty]}(\mu_i),$$

$$F(\mu_i|d) = (1-\mu_i^{-d})\,1_{[1,+\infty]}(\mu_i).$$

By simple algebra, we can derive the intrinsic dimension d from $F(\mu_i|d)$ as follows:

$$d = \frac{\log(1-F(\mu_i))}{\log(\mu_i)}.$$

The empirical cumulative distribution function of μ_i is $F^{emp}(\mu_{\sigma(i)}) = i/N$, where σ is a permutation function that arranges the different μ_i for all $i \in [|1, N|]$ in ascending order. Therefore, we can estimate d using a linear regression on the dataset $\left\{\left(\log(\mu_i), -\log\left(1-F^{emp}(\mu_i)\right)\right)\right\}_{i=1}^{N}$.

LID (Linear Intrinsic Dimension) represents the dimension of the best subspace (with minimal rank) enclosing the data manifold. To estimate LID, we can use PCA (Principal Component Analysis) to identify the principal components (eigenvectors of the data's covariance matrix) that spans the subspace with the minimal projection error similar to [Ansuini *et al.*, 2019]. The difference between LID and ID indicates to what extent the data manifold is curved. For a highly-curved manifold, the linear intrinsic dimension is largely higher than the real intrinsic dimension (LID $\gg$ ID). For a flat manifold, the linear intrinsic dimension is equal to the real intrinsic dimension (LID $\approx$ ID).

7 Appendix G: Hardware and Software

We conduct all experiments using a Linux server under the same hardware and software environment. In Table 3, we specify the software libraries and the used hardware.

Table 3: Hardware and software .

Hardware	
RAM	132 GB
CPU model	Intel(R) Xeon(R) CPU E5-2620 @ 2.10GHz
# of CPUs	32
GPU model	GeForce RTX 2080 Ti
GPU memory	11 GB
# of GPUs	2
Software	
Op. System	Ubuntu 18.04.6 LTS
Python	3.8.8
Tensorflow	2.9.1
Scikit-learn	0.22.2
Scanpy	1.9.1
Anndata	0.8.0

8 Appendix H: Hyperparameter setting

We organize the hyperparameters of our model into two types. The first type is composed of constant hyperparameters that are not related to the processed dataset. These hyper-parameters are listed in Table 4. In this category, there are hyperparameters associated with the data preprocessing, architecture, the pretraining stage and the clustering stage. For example, the number of neighbors in the cell graph is set to 15, the number of pretraining epochs is fixed to 300 and the generator loss weight is set to 0.01. The second category is formed by three hyperparameters that depend on the input dataset: n_o, β_o and β_c. We fix n_o and β_o from the ranges [10, 15, 20, 25, 30, 35, 40, 45, 50] and [0.1, 0.15, 0.20, 0.25, 0.3], respectively, using grid search and β_c from the range [0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.80, 0.85, 0.9]. The data-dependant hyperparameters are listed in Table 2.

9 Appendix I: Visualization of latent codes

In Fig. 1, we illustrate the latent space of our model using T-SNE visualizations at the end of the clustering phase. As we can see, our approach produces high-quality clusters with pronounced within-cluster compactness and noticeable between-cluster separability.

10 Appendix J: Sensitivity to hyperparameters

We explore the sensitivity of scTCM to the hyperparameter of the clustering loss (i.e., β_c). As we can see in Fig. 2, our model shows consistent results in terms of ACC and NMI in a wide range of values of β_c. The best results in terms of

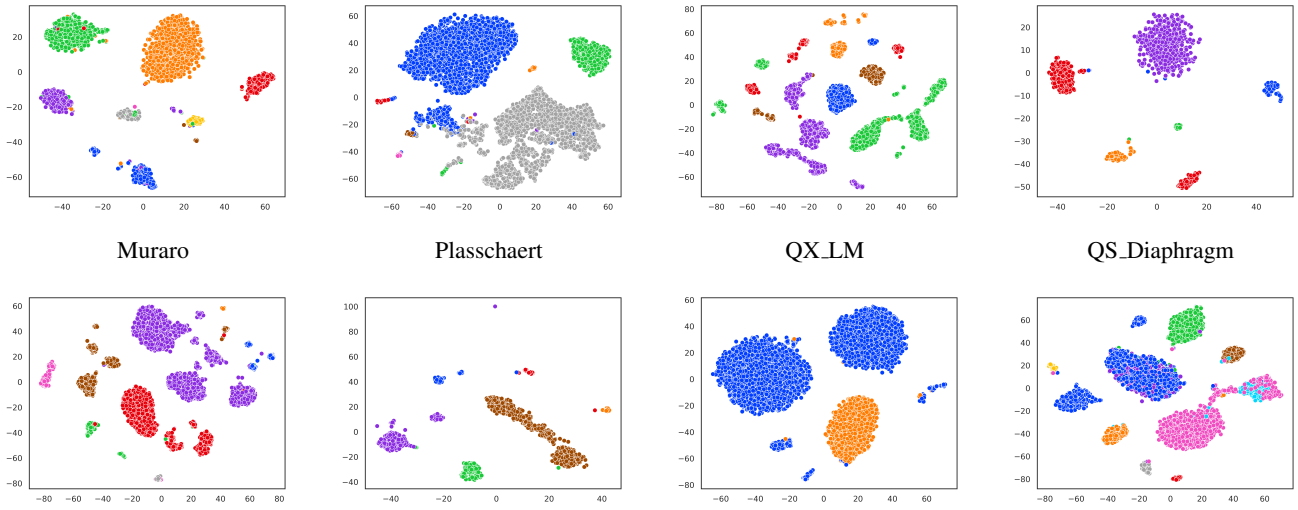

Figure 1: 2D T-SNE visualizations of the latent representations.

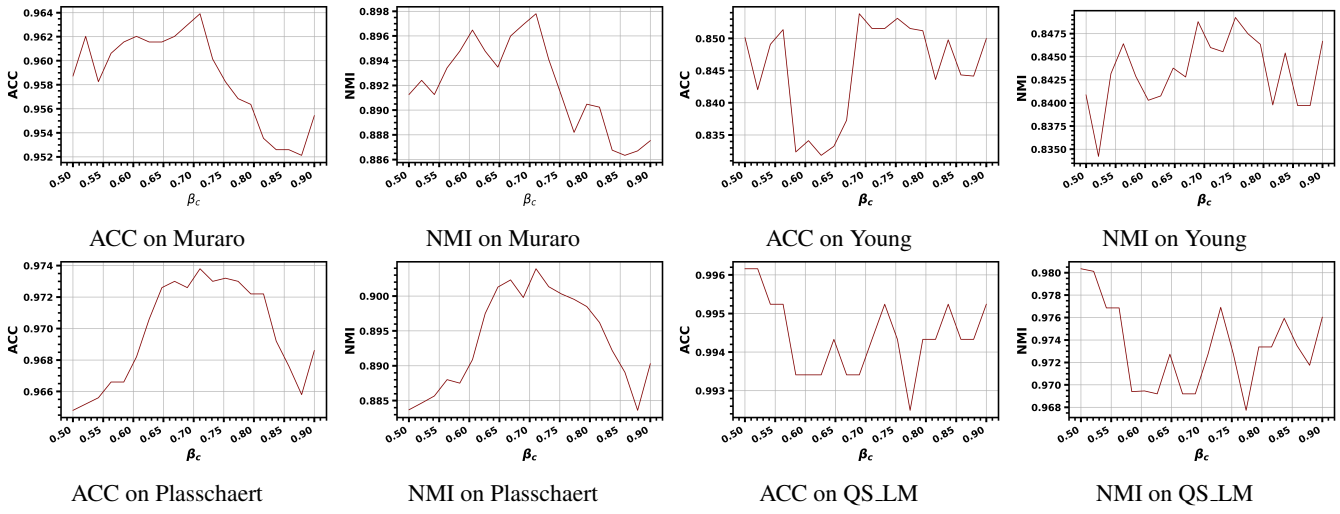

Figure 2: Sensitivity of scTCM to the hyperparameter β_c in terms of ACC and NMI.

Table 4: Fixed hyperparameters for all datasets.

Level	Parameter	Value
Preprocessing	# nearest neighbors κ	15
Architecture	Encoding dimensions	128 - 15
	Decoding dimensions	128 - 256 - 512
	Discriminator dimensions	64 - 64 - 1
Pretraining	Optimizer	Adam
	Learning rate	5.10^{-4}
	Number of epochs T_1	300
Clustering	Optimizer	Adam
	Clustering Learning rate	5.10^{-4}
	Overclustering Learning rate	5.10^{-4}
	Discriminator Learning rate	0.0001
	Generator Learning rate	0.0001
	Clustering weight γ_c	1.5
	Overclustering weight γ_o	0.5
	Generator weight γ_g	0.01

ACC and NMI are achieved when β_c is equal to 0.7 (0.75, respectively) on Muraro (Young, respectively).

References

[Ansuini *et al.*, 2019] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *NeurIPS*, 32, 2019.

[Consortium and others, 2018] Tabula Muris Consortium et al. Single-cell transcriptomics of 20 mouse organs creates a tabula muris. *Nature*, 562(7727):367–372, 2018.

[Facco *et al.*, 2017] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):1–8, 2017.

[Hao *et al.*, 2021] Yuhan Hao, Stephanie Hao, Erica Andersen-Nissen, William M. Mauck III, Shiwei Zheng, Andrew Butler, Maddie J. Lee, Aaron J. Wilk, Charlotte Darby, Michael Zagar, Paul Hoffman, Marlon Stoeckius, Efthymia Papalexi, Eleni P. Mimitou, Jaison Jain, Avi Srivastava, Tim Stuart, Lamar B. Fleming, Bertrand Yeung, Angela J. Rogers, Juliana M. McElrath, Catherine A. Blish, Raphael Gottardo, Peter Smibert, and Rahul Satija. Integrated analysis of multimodal single-cell data. *Cell*, 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.

[Muraro *et al.*, 2016] Mauro J Muraro, Gitanjali Dharmadhikari, Dominic Grün, Nathalie Groen, Tim Dielen, Erik Jansen, Leon Van Gurp, Marten A Engelse, Francoise Carlotti, Eelco Jp De Koning, et al. A single-cell transcriptome atlas of the human pancreas. *Cell systems*, 3(4):385–394, 2016.

[Plasschaert *et al.*, 2018] Lindsey W Plasschaert, Rapolas Žilionis, Rayman Choo-Wing, Virginia Savova, Judith Knehr, Guglielmo Roma, Allon M Klein, and Aron B Jaffe. A single-cell atlas of the airway epithelium reveals the cftr-rich pulmonary ionocyte. *Nature*, 560(7718):377–381, 2018.

[Wang *et al.*, 2018] Yanjie Wang, Zan Tang, Huanwei Huang, Jiao Li, Zheng Wang, Yuanyuan Yu, Chengwei Zhang, Juan Li, Huaping Dai, Fengchao Wang, et al. Pulmonary alveolar type i cell population consists of two distinct subtypes that differ in cell fate. *Proceedings of the National Academy of Sciences*, 115(10):2407–2412, 2018.

[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.

[Young *et al.*, 2018] Matthew D Young, Thomas J Mitchell, Felipe A Vieira Braga, Maxine GB Tran, Benjamin J Stewart, John R Ferdinand, Grace Collord, Rachel A Botting, Dorin-Mirel Popescu, Kevin W Loudon, et al. Single-cell transcriptomes from human kidneys reveal the cellular identity of renal tumors. *science*, 361(6402):594–599, 2018.

[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.