Supplementary Material for: "Exploring the Interaction between Local and Global Latent Configurations for Clustering Single-cell RNA-seq: A Unified Perspective"

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

Appendix A: Additional Related Work

We discuss additional related methods, which are not specific to the single-cell field of research. In particular, we focus on the models that tackle the trade-off between FR and FD.

ADEC (Adversarial Deep Embedding Clustering) (Mrabah, Bouguessa, and Ksantini 2022) is the first deep clustering method that tackles the trade-off between FR and FD. In addition to the auto-encoding architecture, ADEC leverages a discriminator network to minimize an adversarially constrained reconstruction along with an embedding clustering loss. To mitigate FR, ADEC penalizes the generation of latent codes, which could not be decoded into realistic data points. To alleviate the impact of FD, the back-propagation of the self-supervised loss is restrained to the decoder layers. Similar to GAN (Goodfellow et al. 2014), ADEC is subject to inherent stability limitations such as mode collapse (Arjovsky, Chintala, and Bottou 2017) and failure of convergence (Arjovsky and Bottou 2017).

DynAE (Dynamic Auto-Encoder) (Mrabah et al. 2020) introduces a different strategy to tackle the trade-off between FR and FD. Unlike ADEC, DynAE does not require the adversarial training scheme. Instead of that, it uses the decoder to gradually construct images of the latent centers. This aspect alleviates the strong competition between self-supervision and pseudo-supervision. To mitigate FR, DynAE exploits the global latent configuration to extract the most reliable samples for embedding clustering. However, the encoding process of DynAE and ADEC can not exploit the structural information of the input data. Furthermore, their decoding process is limited to generating Euclidean representations.

R-GAE (Rethinking Graph Auto-Encoder) (Mrabah et al. 2021) constitutes the first strategy to tackle FR and FD for attributed graph data. To mitigate FR, R-GAE leverages a sampling operator that selects clustering-reliable nodes according to the confidence of their clustering assignments. To alleviate FD, R-GAE augments the constructed graph topology by connecting the nodes to their associated centroids. Unlike ADEC and DynAE, R-GAE can preserve the structural information of the input data by exploiting the graph convolutional operation. Morever, the decoding process of R-GAE is faithful to the structural information of the data.

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Despite the advancement achieved by ADEC, DynAE, and R-GAE in controlling FR and FD, these models have three main limitations compared to our approach. First, ADEC, DynAE, and R-GAE perform a unimodal decoding process, generating either Euclidean representations or structural information. However, generating the two data modalities (Euclidean representations and structural information) can provide complementary views to the clustering task. Second, these models fail to consider the interaction between local and global latent configurations to adjust the pseudo-supervised and self-supervised tasks. The interaction between local and global configurations can help to detect the most reliable samples for the clustering task. Last but not least, ADEC, DynAE, and R-GAE are not appropriate for clustering scRNA-seq data because their decoding process fail to capture the characteristics of the gene expression count matrix (discreteness, zero-inflation, and over-dispersion).

Appendix B: ZINB mass function

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

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

$$\delta_0(x) = \begin{cases} 1 & \text{if} \quad x = 0 \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

$$f_{\text{NB}}(x\,|\,r,\,p) = \binom{x+r-1}{x}\,(1-p)^r\,p^x, \quad (3)$$

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 $f_{\text{NB}}(x\,|\,r,\,p)$, such that r represents the number of successes and p is the probability of success.

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

$$f_{\text{ZINB}}(x\,|\pi,\,\mu,\,\sigma^2) = \pi\,\delta_0(x) + (1-\pi)\,f_{\text{NB}}(x\,|\,\mu,\,\sigma^2), \quad (4)$$

$$f_{\text{NB}}(x\,|\,\mu,\,\sigma^2) = \frac{\Gamma(x+\sigma^2-\mu)}{x!\,\Gamma(\sigma^2-\mu)}\left(\frac{\mu}{\sigma^2}\right)^{\sigma^2-\mu}\left(\frac{\sigma^2-\mu}{\sigma^2}\right)^{x}. \quad (5)$$

The formulation in Eq. (5) is obtained by setting $p = \frac{\sigma^2 - \mu}{\sigma^2}$ and $r = \sigma^2 - \mu$.

Appendix C: Training Algorithm

The training strategy of our model is described below:

Algorithm 1: Training strategy of our model

Input: initial graph: $\mathcal{G}$, number of clusters: J, number of pretraining iterations: R_1, number of neighbors for building $\mathcal{G}^{\text{latent}}$: k, filtering threshold: ϵ, balancing hyperparameter: γ

for $i = 0$ **to** R_1 **do**
  Compute $L_{\text{SS}}(\mathcal{G})$ according to Eq. (4);
  Update $\left\{W_E^{(l)}, W_D^{(l)}, W_\pi, W_\mu, W_\sigma\right\}$ to minimize $L_{\text{SS}}(\mathcal{G})$ using Adam optimizer;
end for
$\Theta \leftarrow \varnothing$;
Compute $\{\Omega_j\}_{j=1}^{J}$ by performing spectral clustering on A and projecting the centers to the latent space;
while $|\Theta| \leq 0.8 * |\mathcal{V}|$ **do**
  $Z \leftarrow f_E(A, X)$;
  Compute (p_{ij}) according to Eq. (5);
  Θ, $\mathcal{G}^{\text{latent}} \leftarrow \mathcal{F}(Z, (p_{ij}), \epsilon, k)$;
  $\mathcal{G}^{\text{clus}} \leftarrow \mathcal{T}(Z, (p_{ij}), \Theta, \mathcal{G}^{\text{latent}}, \mathcal{G}, \{\Omega_j\})$;
  Compute L_{PS} according to Eq. (7);
  Compute $L_{\text{SS}}(\mathcal{G}^{\text{clus}})$ according to Eq. (9);
  Update $\left\{W_E^{(l)}, W_D^{(l)}, W_\pi, W_\mu, W_\sigma, \{\Omega_j\}\right\}$ to minimize $L_{\text{SS}}(\mathcal{G}^{\text{clus}}) + \gamma\, L_{\text{PS}}$ using Adam optimizer;
end while

Appendix D: Computational Complexity

We perform the complexity analysis under three standard assumptions: (1) all the hidden layers (encoding and decoding) and the initial feature matrix are assumed to have the same dimension d, (2) we assume that the clustering phase involves R_2 iterations and (3) we assume that the number of cells $N \gg \kappa$, d, J, L_E, and L_D. The time complexity to construct the graph $\mathcal{G}$ using κ-NN is $\mathcal{O}(2^d\, \kappa\, N \log(N))$. The time complexity to compute the latent codes Z using the GCN encoding layers is $\mathcal{O}(d\, L_E\, (\kappa + d)\, N)$. The time complexity to compute the generated structure $\hat{A}$ is $\mathcal{O}(d\, N^2)$. The computational complexity to compute the generated gene expression count matrix $\hat{X}$ is $\mathcal{O}\big(\kappa\, d\, L_E\, N + d^2\, (L_E + L_D)\, N\big)$. Accordingly, the time complexity to compute the pertaining loss $L_{\text{SS}}(\mathcal{G})$ is also $\mathcal{O}(d\, N^2)$, and the computational complexity of the complete pretraining phase is $\mathcal{O}(d\, R_1\, N^2)$. The time complexity of Algorithm 1 (main manuscript) is dominated by the construction of $\mathcal{G}^{\text{latent}}$ in $\mathcal{O}(2^d\, k\, N \log(N))$. Hence, the time complexity to compute the loss L_{PS} is $\mathcal{O}(2^d\, k\, N \log(N))$. The computational complexity of Algorithm 2 (main manuscript) is $\mathcal{O}\big((d + \kappa + k\, J)\, N\big)$. Thus, the time complexity to compute the loss $L_{\text{SS}}(\mathcal{G}^{\text{clus}})$ is $\mathcal{O}(d\, N^2)$. Accordingly, the computational complexity of the complete clustering phase is $\mathcal{O}(d\, R_2\, N^2)$. The complexity of the complete training algorithm is $\mathcal{O}(d\, (R_1 + R_2)\, N^2)$. Globally, the construction of A^{gen} is the only operation with quadratic computational complexity, which briefly explains the quadratic complexity of our training algorithm.

Appendix E: Datasets Description and Preprocessing

We used eight real datasatest for the expriments. Muraro (Muraro et al. 2016), Plasschaert (Plasschaert et al. 2018), QX_LM (Consortium et al. 2018), QS_Diaph (Consortium et al. 2018), QS_Heart (Consortium et al. 2018), QS_LM (Consortium et al. 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 $\bar{X} = (\bar{x}_{ij}) \in \mathbb{R}^{N \times M}$ is expressed according to:

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

Parameter	Muraro	Plasschaert	QX_LM	QS_Diaph	QS_Heart	QX_LM	Wang_Lung	Young
Confidence threshold: ϵ	2	2	2	4	6	4	2	6
# neighbours in $\mathcal{G}^{\text{latent}}$: k	0.7	0.7	0.7	0.9	0.9	0.9	0.8	0.8

Table 2: Data-dependent hyperparameters.

Table 3: Fixed hyperparameters for all datasets.

Level	Parameter	Value
Data preprocessing	# nearest neighbors κ	15
Architecture	Encoding dimensions	128 - 15
	Decoding dimensions	128 - 256 - 512
Pretraining phase	Optimizer	Adam
	Learning rate	5.10^{-4}
	Number of epochs R_1	300
Clustering phase	Optimizer	Adam
	Learning rate	5.10^{-4}
	Clustering loss weight γ	1.5

$$\bar{x}_{ij} = \log\Big(\text{Med}(X)\ \frac{x_{ij}}{\sum_{j=1}^{M} x_{ij}}\Big), \tag{6}$$

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.

Appendix F: 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 hyperparameters are listed in Table 3. 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 and the number of pretraining epochs is fixed to 300. The second category is formed by two hyperparameters that depends on the input dataset: ϵ and k. We fix ϵ and k from the ranges $[0.7,\ 0.75,\ 0.8,\ 0.85,\ 0.9]$ and $[2,\ 4,\ 6,\ 8,\ 10]$, respectively, using grid search. The data-dependant hyperparameters are listed in Table 2

Appendix G: Hardware and Software

We conduct all experiments using a Linux server under the same hardware and software environment. In Table 4, we specify the software libraries and the used hardware.

Table 4: 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	1
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

Appendix H: Evaluation Metrics

ACC, NMI, and ARI are standard evaluation metrics to assess the clustering performance. The values of these metrics are within the interval $[0, 1]$. Higher values are associated with better clustering quality. The expressions of ACC, NMI, and ARI are provided in Eq. (7), Eq. (8), and Eq. (9).

$$\text{ACC} = \frac{\sum_{i=1}^{N} \mathbf{1}\left\{y_i = \widetilde{y}_i\right\}}{N}, \tag{7}$$

$$\text{NMI} = \frac{I(Y,\ C)}{\frac{1}{2}[H(Y) + H(C)]}, \tag{8}$$

$$\text{ARI} = \frac{\sum_{ij}\binom{c_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{c}{2}}, \tag{9}$$

where $\mathbf{1}\{.\}$ is the indicator function; $C = (c_i)_{1 \leqslant i \leqslant N}$ and $Y = (y_i)_{1 \leqslant i \leqslant N}$ are the vectors of predicted and true labels, respectively; $\widetilde{Y} = (\widetilde{y}_i)_{1 \leqslant i \leqslant N}$ is the output of the Hungarian algorithm, which identifies the permutation with the best matching between C and Y among all possible permutations of C; $I(Y,\ C)$ denotes the mutual information between the discrete distributions captured by Y and C; $H(Y)$ and $H(C)$

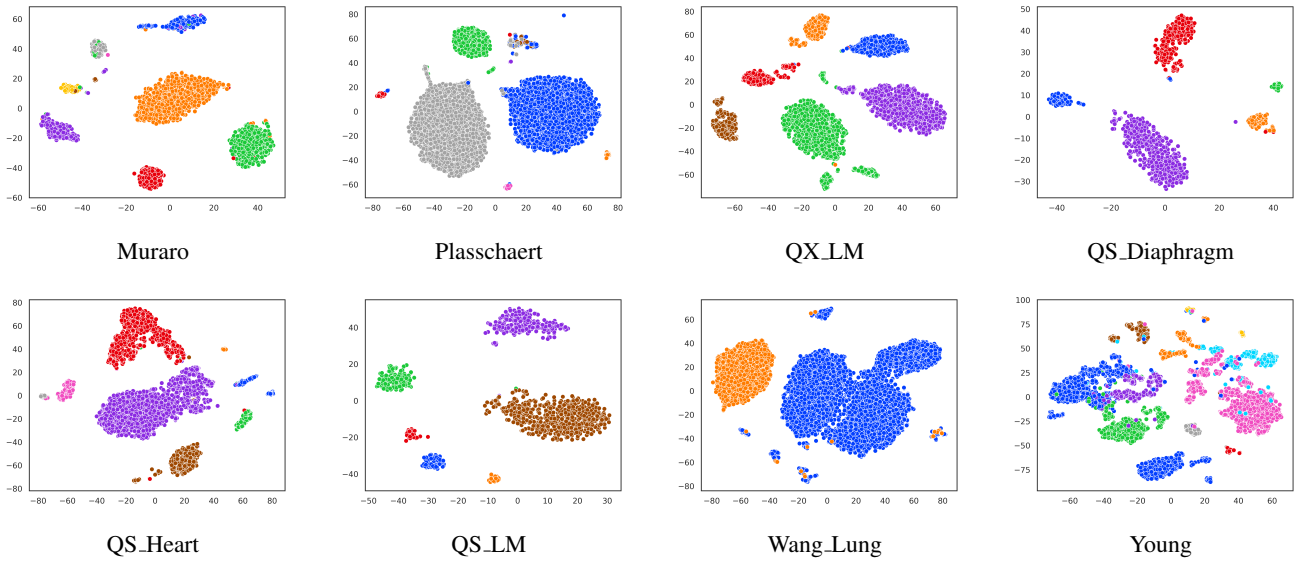

Figure 1: 2D T-SNE visualizations of the latent representations.

compute the entropy of the discrete distributions captured by Y and C respectively; where c is the number of all pairs, a is the number of pairs with the same cluster, and b is the number of pairs with different clusters.

We additionally use the metrics Λ_{FR} and Λ_{FD} (Mrabah, Bouguessa, and Ksantini 2022) to assess the level of FR and FD, respectively. The FR metric is the cosine similarity between two gradients: (i) The gradient of the pseudo-supervised loss and (ii) the gradient of the associated supervised loss. This metric is defined as follows:

$$\Lambda_{FR} = \cos\left(\frac{\partial L_{\text{PS}}}{\partial W}, \frac{\partial L_{\text{S}}}{\partial W}\right), \tag{10}$$

where W represents the vector of training weights, L_{PS} represents the pseudo-supervised loss, L_{S} is the associated supervised loss obtained by replacing the matrix Q (described in eq. (6) of the main paper) by the matrix $Q' = (q'_{ij})$ described in eq. (11). Higher Λ_{FR} values are associated with less randomness during the learning process.

$$q'_{ij} = \begin{cases} 1 \text{ if } \widetilde{y}_i = j, \\ 0 \text{ otherwise.} \end{cases} \tag{11}$$

The FD metric Λ_{FD} computes the cosine similarity between two gradients: The gradient of the pseudo-supervised loss and the gradient of the self-supervised loss. This metric is defined as follows:

$$\Lambda_{FD} = \cos\left(\frac{\partial L_{\text{PS}}}{\partial W}, \frac{\partial L_{\text{SS}}}{\partial W}\right). \tag{12}$$

Higher values of Λ_{FD} are associated with less competition between pseudo-supervision and self-supervision.

Appendix I: Additional Results

Visualisations: In Figure 1, we illustrate the latent space T-SNE visualizations of our model 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.

References

Arjovsky, M.; and Bottou, L. 2017. Towards Principled Methods for Training Generative Adversarial Networks. *ICLR*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *ICML*, 214–223.

Consortium, T. M.; et al. 2018. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. *Nature*, 562(7727): 367–372.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *NeurIPS*, 27.

Hao, Y.; Hao, S.; Andersen-Nissen, E.; III, W. M. M.; Zheng, S.; Butler, A.; Lee, M. J.; Wilk, A. J.; Darby, C.; Zagar, M.; Hoffman, P.; Stoeckius, M.; Papalexi, E.; Mimitou, E. P.; Jain, J.; Srivastava, A.; Stuart, T.; Fleming, L. B.; Yeung, B.; Rogers, A. J.; McElrath, J. M.; Blish, C. A.; Gottardo, R.; Smibert, P.; and Satija, R. 2021. Integrated analysis of multimodal single-cell data. *Cell*.

Mrabah, N.; Bouguessa, M.; and Ksantini, R. 2022. Adversarial Deep Embedded Clustering: On a Better Trade-off Between Feature Randomness and Feature Drift. *TKDE*, 34(04): 1603–1617.

Mrabah, N.; Bouguessa, M.; Touati, M. F.; and Ksantini, R. 2021. Rethinking Graph Auto-Encoder Models for Attributed Graph Clustering. *arXiv preprint arXiv:2107.08562*.

Mrabah, N.; Khan, N. M.; Ksantini, R.; and Lachiri, Z. 2020. Deep clustering with a dynamic autoencoder: From reconstruction towards centroids construction. *Neural Networks*, 130: 206–228.

Muraro, M. J.; Dharmadhikari, G.; Grün, D.; Groen, N.; Dielen, T.; Jansen, E.; Van Gurp, L.; Engelse, M. A.; Carlotti, F.; De Koning, E. J.; et al. 2016. A single-cell transcriptome atlas of the human pancreas. *Cell systems*, 3(4): 385–394.

Plasschaert, L. W.; Žilionis, R.; Choo-Wing, R.; Savova, V.; Knehr, J.; Roma, G.; Klein, A. M.; and Jaffe, A. B. 2018. A single-cell atlas of the airway epithelium reveals the CFTR-rich pulmonary ionocyte. *Nature*, 560(7718): 377–381.

Wang, J.; Ma, A.; Chang, Y.; Gong, J.; Jiang, Y.; Qi, R.; Wang, C.; Fu, H.; Ma, Q.; and Xu, D. 2021. scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses. *Nature communications*, 12(1): 1–11.

Wang, Y.; Tang, Z.; Huang, H.; Li, J.; Wang, Z.; Yu, Y.; Zhang, C.; Li, J.; Dai, H.; Wang, F.; et al. 2018. 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.

Young, M. D.; Mitchell, T. J.; Vieira Braga, F. A.; Tran, M. G.; Stewart, B. J.; Ferdinand, J. R.; Collord, G.; Botting, R. A.; Popescu, D.-M.; Loudon, K. W.; et al. 2018. Single-cell transcriptomes from human kidneys reveal the cellular identity of renal tumors. *science*, 361(6402): 594–599.

Yu, Z.; Lu, Y.; Wang, Y.; Tang, F.; Wong, K.-C.; and Li, X. 2022. ZINB-Based Graph Embedding Autoencoder for Single-Cell RNA-Seq Interpretations. *AAAI*, 36(4): 4671–4679.