ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικό και Καποδιστριακό
Πανεπιστήμιο Αθηνών

Clustering single-cell RNA-seq data with a model-based deep learning approach

Tian Tian 1,3, Ji Wan2,3, Qi Song2 and Zhi Wei 1*

Name:	BATOUL KHALIL
Student ID:	7115152200025
Subject:	Final project report

Paper abstract:

Single-cell RNA sequencing (scRNA-seq) promises to provide higher resolution of cellular differences than bulk RNA sequencing. Clustering transcriptomes profiled by scRNA-seq has been routinely conducted to reveal cell heterogeneity and diversity. However, clustering analysis of scRNA-seq data remains a statistical and computational challenge, due to the per vasive dropout events obscuring the data matrix with prevailing 'false' zero count observations. Here, we have developed scDeepCluster, a single-cell model-based deep embedded clustering method, which simultaneously learns feature representation and clustering via explicit modelling of scRNA-seq data generation. Based on testing extensive simulated data and real datasets from four representative single-cell sequencing platforms, scDeepCluster outperformed state-of-the-art methods under various clustering performance metrics and exhibited improved scalability, with running time increasing linearly with sample size. Its accuracy and efficiency make scDeepCluster a promising algorithm for clustering large-scale scRNA-seq data.

Introduction:

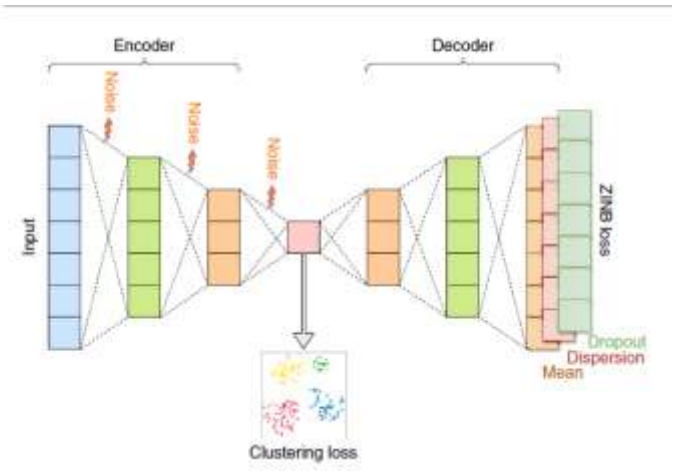
Cluster analysis is a critical component of machine learning and data mining, enabling the grouping of data into meaningful clusters. Deep clustering, which utilizes deep neural networks to learn clustering-friendly representations, has gained widespread use in various clustering tasks. However, existing surveys on deep clustering have predominantly focused on single-view fields and network architectures, overlooking the complexities of clustering in real-world scenarios.

Single-cell RNA sequencing (scRNA-seq) has emerged as a powerful tool for understanding cellular heterogeneity and diversity, facilitating research in complex biological questions. However, clustering scRNA-seq data presents unique challenges due to its sparsity, with most measurements being zeros (dropout events). These technical limitations, along with high variations in gene expression levels among cells, introduce significant noise, making clustering particularly challenging.

In this article, we introduce scDeepCluster, a model-based deep embedded clustering method, which is derived from the paper "Clustering single-cell RNA-seq data with a model-based deep learning approach." We evaluate the performance of this approach and compare it with classical clustering methods, using four real datasets utilized in the same paper.

Our results show that scDeepCluster outperforms classical clustering methods. However, its performance still falls short on complex real datasets, with lower accuracy. While scDeepCluster performs better than classical clustering, it cannot be solely relied upon. Additionally, we explored using the Gaussian Mixture Model (GMM) as an alternative clustering method in the pipeline, but it did not yield better results either. The effectiveness of each algorithm depends on the characteristics of the dataset, highlighting the need for dataset-specific approaches to achieve accurate clustering results.

scDeepCluster:



scDeepCluster is a novel single-cell model-based deep embedded clustering method designed to simultaneously learn feature representations and perform clustering by explicitly modeling the generation of scRNA-seq data.

Computational Pipeline: The scDeepCluster pipeline comprises several key steps:

1. Pre-processing: This step involves filtering out genes with zero counts and normalizing the data to ensure consistency.
2. Denoising: Utilizing a ZINB model-based autoencoder, the normalized data points are processed to predict the original uncorrupted data points, effectively denoising the data.
3. Deep Embedded Clustering with Local Structure Preservation: The denoised data is then fed into the deep embedded clustering algorithm, which preserves local structure information to enhance clustering accuracy.
4. Optimization: The optimization process relies on stochastic gradient descent and halts when the percentage of points changing cluster assignments between consecutive iterations falls below a predefined tolerance threshold.

Classical Clustering Pipeline: The classical clustering pipeline involves the following stages:

1. Preprocessing: The dataset is normalized using Z-score normalization.
2. Dimensionality Reduction: High-dimensional gene expression data is reduced to a lower-dimensional space using t-SNE.
3. Clustering: The reduced data is then clustered into the "best" number of cell "states" or clusters using Gaussian Mixture Modeling (GMM), with optimal parameters determined using the Bayesian Information Criterion (BIC).

The Use of ZINB Model and Denoising Autoencoder: scDeepCluster utilizes the Zero-Inflated Negative Binomial (ZINB) model, which is effective at handling discrete, over-dispersed, and zero-inflated count data present in microbiome sequencing. This model effectively addresses the challenge of dropout events, a common statistical issue in scRNA-seq data. Additionally, the incorporation of a denoising autoencoder to the ZINB model-based autoencoder enhances its ability to learn robust feature representations, leading to improved clustering performance and overall stability

The data we used:

Table 1 | Summary of four real scRNA-seq datasets

Dataset	Sequencing platform	Sample size/cell numbers	No. of genes	No. of groups
10X PBMC	10X	4,271	16,449	8
Mouse ES cells	Droplet barcoding	2,717	24,046	4
Mouse bladder cells	Microwell-seq	2,746	19,079	16
Worm neuron cells	sci-RNA-seq	4,186	11,955	10

We randomly sampled 2,100 cells from each dataset to carry out the experiments.

For the evaluation of scDeepCluster's performance, we utilized four real scRNA-seq datasets, each obtained from different representative sequencing platforms. The datasets are as follows:

1. PBMC 4k cells from the 10X genomics platform (10X PBMC).
2. Mouse embryonic stem cells from a droplet barcoding platform (mouse ES cells).
3. Mouse bladder cells from the Microwell-seq platform (mouse bladder cells).
4. Worm neuron cells from the sci-RNA-seq platform (worm neuron cells).

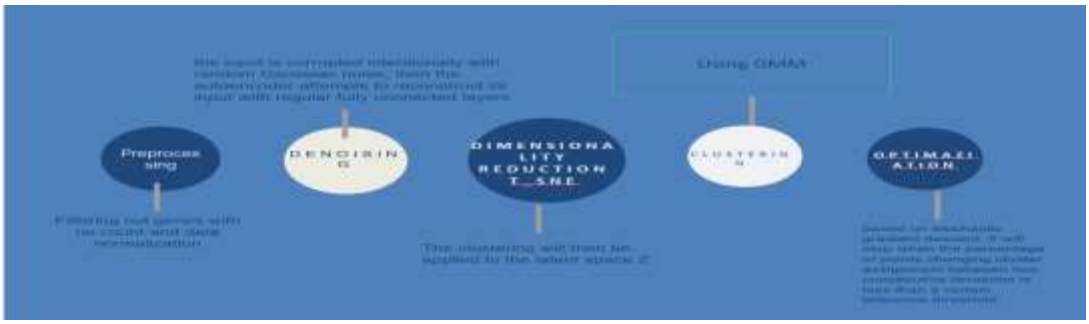
Table summarizing the datasets:

- 10X PBMC: 4,271 cells, 16,449 genes, and 8 groups per cluster.
- Mouse ES cells: 2,717 cells, 24,046 genes, and 4 groups per cluster.
- Mouse bladder cells: 2,746 cells, 19,079 genes, and 16 groups per cluster.
- Worm neuron cells: 4,186 cells, 11,955 genes, and 10 groups per cluster.

The authors randomly sampled 2,100 cells from each dataset. The random sampling procedure ensured that no groups were lost in any of the datasets. I used also the 2100 selected datasets to reduce the runtime.

Methods:

The SCDeeplearning pipeline:



The classical clustering pipeline:

During the execution of the study, the authors implemented their proposed scDeepCluster pipeline, with the only modification being the use of the GMM method instead of the k-means algorithm in the clustering stage. The comparison between the scDeepCluster and classical clustering pipelines was conducted to understand the complexity of the datasets used and evaluate the performance of the two methods.

One key aspect of the scDeepCluster pipeline is that it requires the number of clusters to be known beforehand, and for each dataset, the number of clusters was manually specified. However, in real-world scenarios, the number of clusters is often unknown, making manual specification impractical. To address this limitation, the suggestion is made to employ the classical clustering pipeline as a preliminary step.

The proposed approach involves the following steps:

1. Using the classical clustering pipeline: First, the unknown dataset is subjected to the classical clustering pipeline. This includes data pre-processing, dimensionality reduction (e.g., t-SNE), and clustering using GMM. The goal is to determine the optimal number of clusters and find the best parameters for GMM.
2. Applying scDeepCluster: Once the optimal number of clusters and GMM parameters have been determined through the classical pipeline, this information can be used as input for the scDeepCluster pipeline. This approach leverages the advantages of both methods while overcoming the limitation of manually specifying the number of clusters.

It is suggested to perform this procedure outside the scDeepCluster implementation because the scDeepCluster process itself is computationally expensive. By using the classical pipeline to identify the optimal number of clusters and GMM parameters beforehand, the scDeepCluster method can be applied more efficiently and effectively to cluster unknown datasets.

In conclusion, the combination of the classical clustering pipeline for determining the optimal number of clusters and GMM parameters, followed by the application of the scDeepCluster pipeline, offers a more robust and practical approach for clustering datasets with an unknown number of clusters.

In the scDeepCluster pipeline, each dataset requires a separate pretraining phase for the deep neural network to learn its specific data representation. This individualized pretraining process ensures that the neural network adapts to the unique characteristics and complexities present in each dataset. Consequently, the learned representations are tailored to the specific dataset, resulting in non-uniform weights for different datasets.

The need for dataset-specific pretraining is crucial because different scRNA-seq datasets can vary significantly in terms of their sparsity, gene expression patterns, technical noise, and overall data distribution. By pretraining the deep neural network on each dataset, the model can effectively capture the underlying structures and variations present in the data, leading to more accurate and meaningful clustering results.

As a result, the weights and learned features of the deep neural network will differ from one dataset to another, reflecting the dataset-specific information captured during pretraining. This dataset-specific adaptation is essential for achieving optimal clustering performance and addressing the inherent complexities and variations in scRNA-seq data.

Execution:

The Execution stage summarized by these step:

 Applying both the scDeepCluster and classical clustering pipelines to the four real scRNA-seq datasets.
 Assessing the performance of the methods using accuracy metrics.
 Analyzing the complexity of the datasets and understanding the clustering performance of the two methods.implementation of the scDeepCluster algorithm for scRNA-seq data clustering. It uses a deep autoencoder model to learn a low-dimensional representation of the input data and performs clustering based on this representation.

the main steps performed in the code:

1. Pretraining: The autoencoder is pretrained on the input data using the Kullback-Leibler divergence loss function.
2. Initialization: The cluster centers are initialized using k-means clustering on the encoded data.
3. Deep clustering: The model is trained iteratively, updating the target distribution and cluster centers based on the current model predictions. The training includes both reconstruction loss and clustering loss.
4. Evaluation: The clustering performance is evaluated at regular intervals using metrics such as accuracy, normalized mutual information (NMI), and adjusted Rand index (ARI).
5. Model checkpoints: Intermediate models are saved at certain intervals during training.

The results:

mouse_ES_cell_select_2100.h5

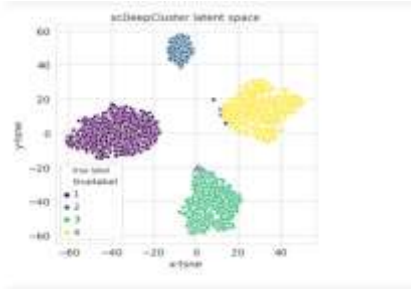
number of clusters: 4

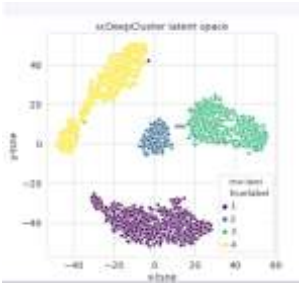
k_mean: GMM: Classical Clustering

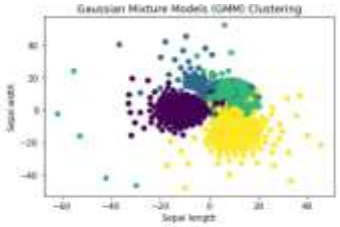
```
Reached tolerance threshold. Stopping training.
saving model to: scDeepCluster1/scDeepCluster_model_final.h5
Final: ACC= 0.9795, NMI= 0.9433, ARI= 0.9604
Clustering time: 358 seconds.
```

```
saving model to: scDeepCluster/scDeepCluster_model_final.h5
Final: ACC= 0.8100, NMI= 0.8221, ARI= 0.7687
Clustering time: 243 seconds.
```

```
Accuracy: 0.2723809523809524
```







Based on the obtained results, it is evident that the scDeepCluster method demonstrated excellent performance in the "mouse_ES_cell_select_2100.h5" dataset, achieving a remarkably high accuracy and successfully clustering nearly all cells. Conversely, the classical clustering approach did not yield satisfactory results, with a considerably low accuracy.

In terms of the comparison between GMM and KMeans within the scDeepCluster pipeline, it was observed that KMeans outperformed GMM in terms of clustering accuracy. However, it is noteworthy that GMM exhibited faster execution time compared to KMeans.

These findings highlight the superiority of the scDeepCluster method over classical clustering in the specific dataset mentioned. Furthermore, within the scDeepCluster pipeline, KMeans was a more suitable choice for clustering, despite being relatively slower in execution compared to GMM.

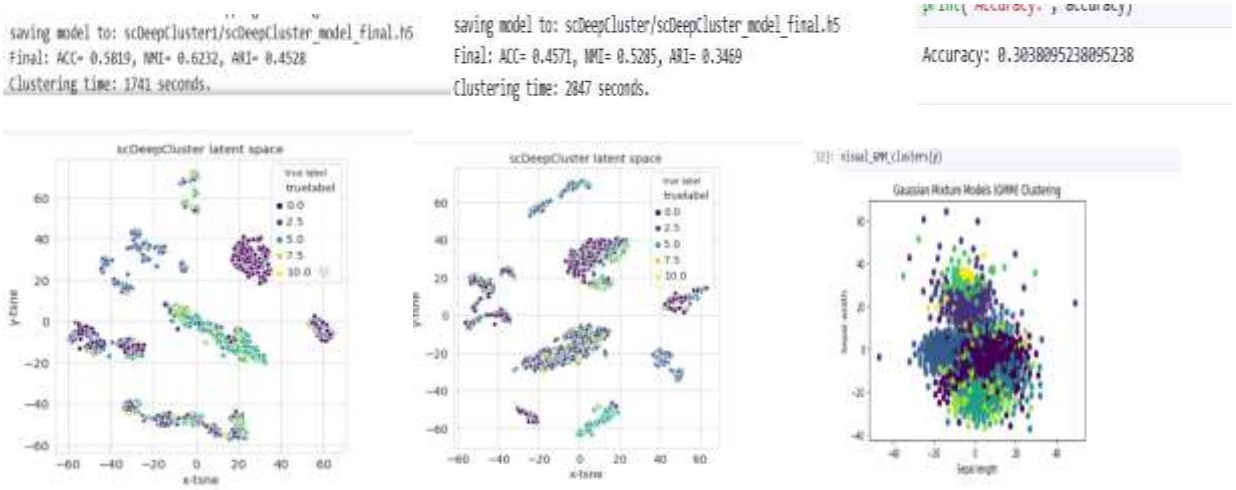
The results suggest that scDeepCluster has the potential to effectively handle the complexity of certain datasets and achieve superior clustering performance compared to traditional methods like

classical clustering. Additionally, the choice between GMM and KMeans within the scDeepCluster pipeline should be made based on the balance between accuracy and computational efficiency, considering the specific characteristics of the dataset at hand

10X_PBMC_select_2100.h5:

Number of clusters :8

K_MM GMM: Classical Clustering:



Based on the observed results, it is evident that scDeepCluster consistently outperformed classical clustering methods. Within the scDeepCluster pipeline, KMeans exhibited higher accuracy compared to GMM. Additionally, KMeans demonstrated faster clustering time in comparison to GMM.

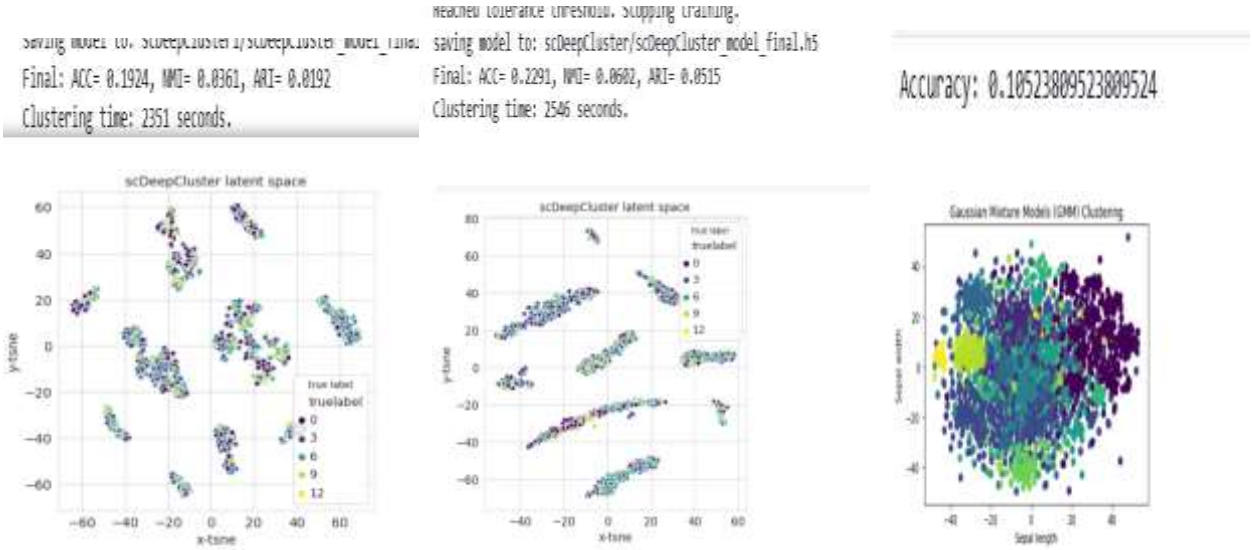
Furthermore, while scDeepCluster showcased superior performance compared to classical clustering methods, the accuracy achieved in this specific dataset was not as high as observed in the previous dataset. Despite being at a medium level, the accuracy of scDeepCluster remained satisfactory.

The medium level of accuracy achieved by scDeepCluster in this particular dataset indicates that the performance of the method might vary depending on the complexity and characteristics of the dataset.

worm_neuron_cell_select_2100.h5

number of clusters : 10

K_mean : GMM: Classical Clustering:



The analysis of the results indicates that while scDeepCluster, as a deep clustering method, continues to outperform classical clustering techniques, there are some trade-offs to consider. The complexity associated with applying scDeepCluster may be higher compared to classical clustering methods, which could make its implementation more challenging and resource-intensive.

Within the scDeepCluster pipeline, it was observed that GMM exhibited better performance than KMeans in terms of clustering accuracy. However, KMeans showcased faster execution time compared to GMM.

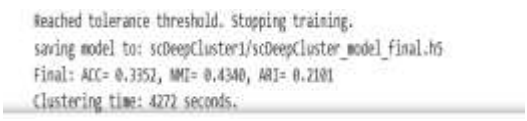
One possible explanation for the relatively low accuracy achieved in this scenario could be attributed to the higher number of clusters used in the analysis. With an increased number of clusters, the clustering results might become less meaningful or less informative. This can lead to a situation where the accuracy of the clustering method becomes less reliable.

Hence, the choice of the appropriate number of clusters is critical in clustering tasks. A high number of clusters can lead to over-segmentation and less meaningful clusters, negatively impacting the overall accuracy and interpretability of the results. To improve the accuracy, it is essential to carefully determine the optimal number of clusters that best represent the underlying structure of the data.

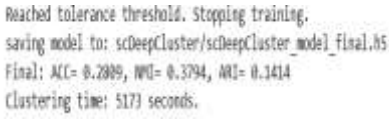
mouse_bladder_cell_select_2100.h5:

number of clusters: 16

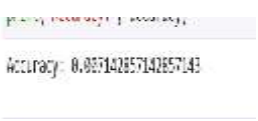
k_mean

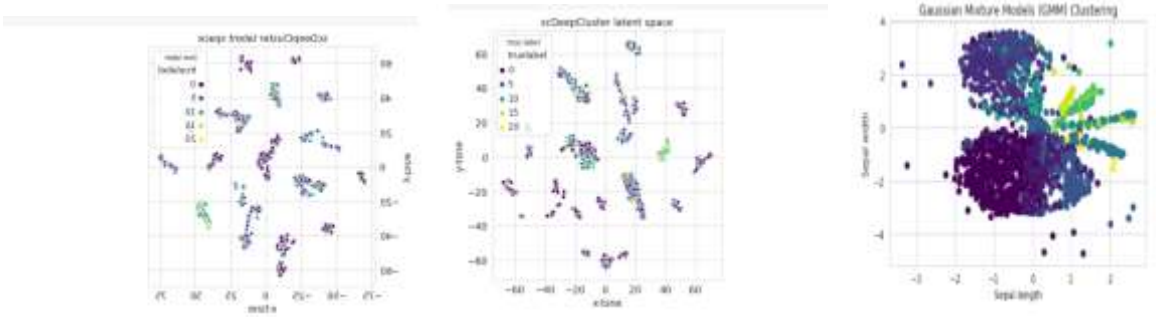


GMM:

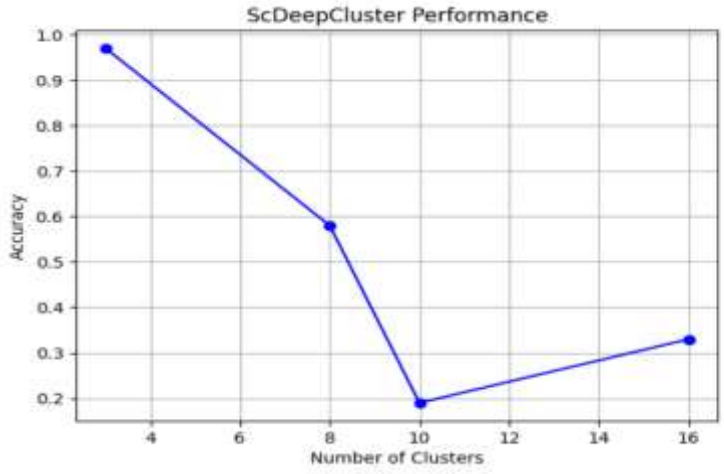


Classical Clustering:

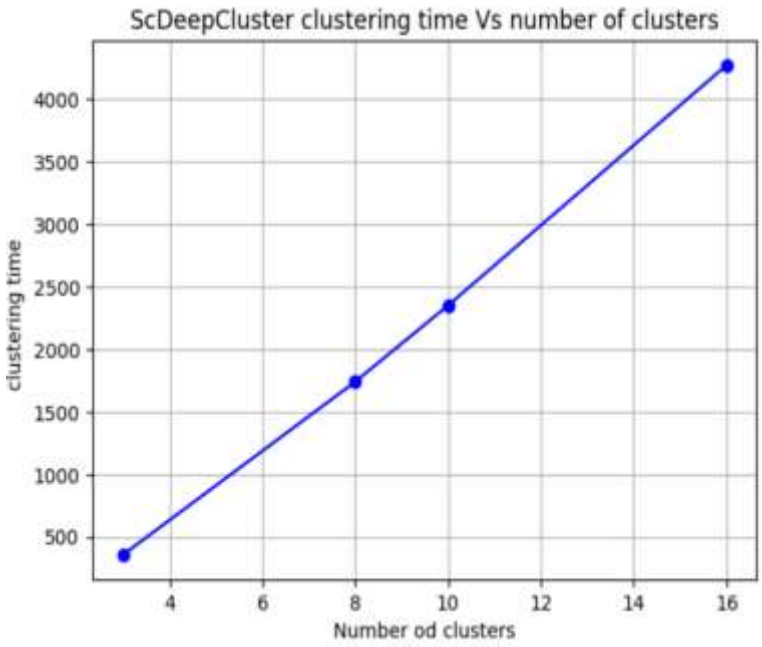




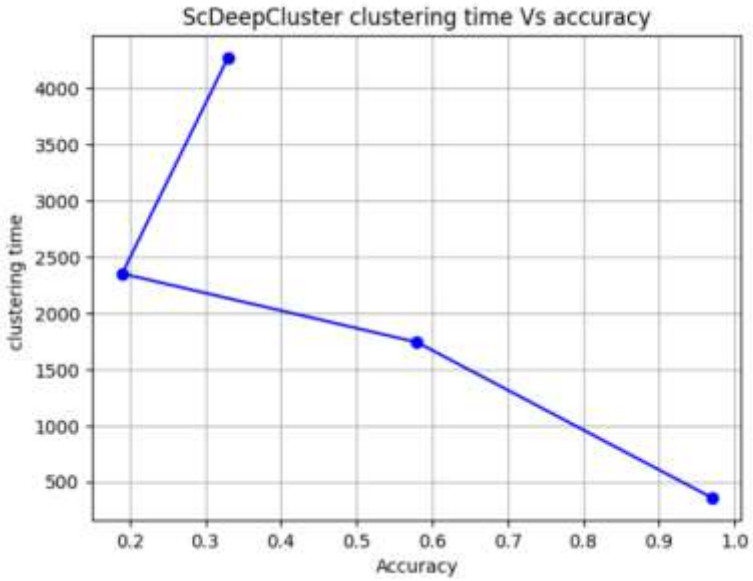
Within the scDeepCluster pipeline, it was observed that Kmean exhibited better performance than GMM in terms of clustering accuracy and faster execution time .



The plot shows the performance of scDeepclustering given the number of clusters.



	mouse_ES_cell_select_2100.h5	10X_PBMC_select_2100.h5:	mouse_bladder_cell_select_2100.h5:	worm_neuron_cell_select_2100.h5
Number of clusters	4	8	16	10
KMM accuracy	0.9795	0.5819	0.3352	0.1924
GMM accuracy	0.8100	0.4571	0.2809	0.2291
Classical accuracy	0.2723	0.3038	0.007	0.105
Kmm clustering time	358	1741	42272	2351
Gmm clustering time	243	2847	5173	2546



The utilized requirements in my execution :

- Python version : 3.6.3
- Keras: 2.1.4
- Tensorflow: 1.1.0
- Scanpy: 0.4.2

The Requirement from the author

Python --- 3.6.3

Keras --- 2.1.4

Tensorflow --- 1.1.0

Scanpy --- 1.0.4

Scanpy --- 1.0.4 is not available and it occurs errors.

Note: The denoising ZINB model-based autoencoder is first pre-trained by 400 epochs in the author work I pre_trained by just 10 epochs because I have limited acress to GPU and the runtime will take high than I have

Nvidia Tesla K80 (12G)

Please note that if using different versions, the results reported in the paper might not be reproducable.

The limitation and challenges:

1. Compatibility with TensorFlow and Keras versions: The code appears to be written with an older version of TensorFlow and Keras. Over time, libraries like TensorFlow and Keras receive updates and improvements, and sometimes backward compatibility may not be maintained for older code. This can lead to issues or unexpected behavior when running the code with newer versions of these libraries. To address this, you may need to modify the code to be compatible with the specific version of TensorFlow and Keras that you are using or consider updating the code to work with the latest versions of these libraries.
2. Computational resources: The code uses deep neural networks and performs clustering on potentially large datasets. Training deep learning models can be computationally intensive, especially for large datasets, and may require significant computational resources. Using a GPU (Graphics Processing Unit) can significantly speed up the training process, but it might not always be available or affordable for everyone. Additionally, working with large datasets can consume a considerable amount of memory, and it's essential to ensure that the available resources are sufficient for the task at hand.
3. Debugging and error handling: As with any code, debugging and handling errors are essential. Deep learning models can be complex, and debugging errors or issues in the code can be challenging. It is crucial to have clear error messages and meaningful logging to aid in debugging and diagnosing problems. Proper error handling can also make the code more robust and prevent crashes or unexpected behaviors when it encounters invalid inputs or other issues.
4. Hyperparameter tuning: The performance of the deep learning models and clustering algorithm may depend on various hyperparameters. In the code, one specific hyperparameter mentioned is the number of clusters, which can significantly impact the clustering results. Finding the optimal set of hyperparameters for a specific task can be time-consuming and require experimentation. Techniques like grid search or random search can be employed to explore different hyperparameter combinations, but this process can also be computationally expensive.

Conclusion:

In conclusion, the experiment clearly shows that scDeepCluster outperforms classical clustering methods, particularly when dealing with datasets with a simple number of clusters. However, for complex datasets, the use of scDeepCluster may not be justified due to its relatively poor accuracy, despite being better than classical clustering.

Furthermore, the introduction of the GMM algorithm within the scDeepCluster pipeline did not yield significant improvements in the clustering results, suggesting that alternative methods like KMeans might be more suitable for achieving better accuracy and performance.

An important finding is that employing the classical clustering pipeline to predict the optimal number of clusters proves to be a valuable approach. This step helps in determining the appropriate number of clusters, which plays a crucial role in enhancing the quality and interpretability of the clustering results.

It is worth noting that the experiment's results might be less accurate than the authors' results, primarily because the authors pre-trained the model for a significant number of epochs, whereas in this experiment, a smaller number of epochs was used to reduce the runtime. This difference in pre-training might have impacted the overall performance and accuracy of the clustering results.

In summary, scDeepCluster is a powerful method for datasets with simple cluster structures, but its usefulness may be limited for more complex datasets. The choice of clustering algorithm and the number of clusters significantly affect the clustering results, and employing classical clustering techniques for preprocessing can be beneficial in enhancing the overall accuracy of the clustering process.

References:

1_Clustering single-cell RNA-seq data with a model-based deep learning approach
2_The scRNA-seq data that support the findings of this study are available in GitHub: https://github.com/ttgump/scDeepCluster/tree/master/scRNA-seq%20data.
3_Code availability
The source code, weights of trained models and the real scRNA-seq data used for experiments of scDeepCluster are available in GitHub: https://github.com/ttgump/scDeepCluster.

4_ Deep Clustering: A Comprehensive Survey, Yazhou Ren, Jingyu Pu, Zhimeng Yang, Jie Xu, Guofeng Li, Xiaorong Pu, Philip S. Yu, Lifang He