STT 490 Project Report

Deep learning clustering for single-cell RNA-seq data of mutant IDH1 glioma stem cells in brain cancer

Group 4: Wenting Liu, Yena Hong, Angelica Louise Gacis

I. Introduction

Understanding diseases at the smallest level, down to individual cells, is becoming much more relevant nowadays. One way to do that is through a technology called single-cell RNA sequencing (scRNA-seq). It enables us to study the complexities and heterogeneity of RNA transcripts at an individual cell level (Jovic et al., 2022). It is incredibly useful in studying complex diseases like cancer.

Despite scRNA-seq being powerful, it poses several challenges with the type of data it produces. Due to high dimensionality and often missing data and noise, it can be difficult to analyze. A useful tool in analyzing scRNA-seq data is unsupervised clustering. However, traditional clustering methods can struggle with this due to the complexity and noise in the data.

A recent study by Gan et al. (2022) has tried to address this problem. In their paper, "Deep structural clustering for single-cell RNA-seq data jointly through autoencoder and graph neural network", they introduce a new method called scDSC. This method uses an Zero-inflated Negative Binomial (ZINB) model-based autoencoder, a type of artificial neural network, along with a graph neural network (GNN) to better cluster the cells.

According to Gan et al. (2022), scDSC works better than the current best methods in terms of accuracy and efficiency. This highlights the potential of using advanced machine learning techniques to make sense of scRNA-seq data.

II. Project Description

For this project, we will utilize a total of 18 human samples (Alghamri et al., 2021) of brain cancer. Among these samples, there are 10 cases with mutant IDH1 and 8 controls with wild type IDH1. Within each sample, we will have a matrix where the rows represent 32,743 human genes, and the columns represent over 2000 cell types represented by short sequencing nucleotides.

The goals of this project are to utilize AI techniques to analyze scRNA-sequence data, with a specific focus on the impact of IDH1 mutation in glioma on the tumor infiltrating myeloid cells phenotype and function. We will apply a new deep structural clustering method for scRNA-seq data, named scDSC [4], which integrates the structural information into deep clustering of single cells. The proposed scDSC consists of a Zero-Inflated Negative Binomial (ZINB) model-based autoencoder, a graph neural network (GNN) module and a mutual-supervised module to clustering of tumor-infiltrating immune cells for identify the underlying differences in cell representation between the case groups (mutant IDH1) and the control groups (wild-type IDH1) in the future.

III. Exploratory Data Analysis

3.1 Individual sample data pre-processing by using Seurat Standard pre-processing workflow:

1. Quality Control (QC) and selecting cells for further analysis:

Seurat offers an easy-to-use interface for exploring QC metrics and customizing cell filtering based on user-defined criteria. Some commonly used QC metrics are unique gene count, total molecule count, and mitochondrial genome mapping.

To calculate mitochondrial quality control metrics, Seurat uses the PercentageFeatureSet() function, which analyzes the percentage of counts coming from a specific group of genes, typically genes starting with "MT-" to represent mitochondrial genes.

We apply filters to exclude cells that do not meet our criteria and ensures that only cells of desired quality are included in our analysis:

- Eliminate cells with unique feature counts exceeding 2,500 or falling below 200.
- Discard cells that exhibit mitochondrial counts exceeding 5%.

2. Normalizing the data:

After selecting the desired cells from the dataset, we employ a normalization method known as "LogNormalize." This method standardizes the feature expression measurements for each cell by the total expression. It further multiplies the normalized values by a default scaling factor and applies a logarithmic transformation to the resulting values.

3. Identification of highly variable features (feature selection):

We identify a set of features in the dataset that show significant differences between cells. These features have high expression levels in some cells and lower levels in others. Focusing on these genes highlights the important biological signals in single-cell datasets.

In Seurat, we directly capture the natural mean-variance relationship in single-cell data using the FindVariableFeatures() function. We choose and include 2,000 features from each dataset.

4. Scaling the data:

As a standard pre-processing step, we use a linear transformation. This function shifts the expression of each gene to make the mean expression across cells 0 and scales the expression of each gene to make the variance across cells 1. This ensures that each gene is equally important in subsequent analyses, preventing highly expressed genes from dominating the results.

5. Perform linear dimensional reduction:

We perform PCA on the scaled data using only the identified variable features. Seurat provides helpful methods for visualizing both cells and features in the PCA results. These methods include VizDimReduction(), DimPlot(), and DimHeatmap().

DimHeatmap() is particularly useful for exploring the key factors that contribute to variation in the dataset. It helps identify the principal components (PCs) to focus on by arranging cells and features based on their PCA scores. The function can plot the most extreme cells at both ends by specifying a number, which is beneficial for large datasets. It is effective for investigating sets of correlated features.

6. Determine the 'dimensionality' of the dataset:

Seurat employs cell clustering based on PCA scores to handle technical noise in scRNA-seq data. Each principal component (PC) represents a combined measure of related features, acting as a 'metafeature.' The top principal components compress the dataset and provide robust information. However, choosing the right number of components is crucial.

To identify significant principal components, we use a resampling test inspired by the JackStraw procedure. By randomly permuting a subset of the data and rerunning PCA, we create a "null distribution" of feature scores. Significant PCs exhibit a high enrichment of low p-value features. The JackStrawPlot() function visually compares the p-value distribution of each PC to a uniform distribution, identifying significant PCs above a dashed line.

The ElbowPlot() function generates an Elbow plot ranking the principal components by the percentage of variance they explain. The plot shows an 'elbow' around PC9-PC10, indicating that most of the genuine signal is captured within the first 10 principal components.

7. Cluster the cells:

Our clustering method is based on recent research that utilized graph-based clustering techniques in scRNA-seq (SNN-Cliq, Xu and Su, Bioinformatics, 2015) and CyTOF (PhenoGraph, Levine et al., Cell, 2015) data. These techniques represent cells as nodes in a graph, where similar cells are connected. The goal is to partition the graph into closely connected groups known as 'quasi-cliques' or 'communities'.

We start by creating a KNN graph using the Euclidean distance in the PCA space. To improve the edge weights between cells, we consider the shared overlap in their local neighborhoods using the Jaccard similarity. This process is performed with the FindNeighbors() function, using the first 10 principal components.

For clustering, we utilize modularity optimization techniques like the Louvain algorithm or SLM. These algorithms group cells together to maximize the modularity function. The FindClusters() function implements this process, allowing control over clustering granularity with a resolution parameter. Higher values produce more clusters. In our project with around 2,000 cells, setting the resolution parameter equals to 0.5 generally gives satisfactory results.

8. Run non-linear dimensional reduction (UMAP/tSNE):

We use the same PCs obtained from the clustering analysis as input for non-linear dimensional reduction techniques like tSNE and UMAP provided by Seurat. These techniques help visualize and explore the datasets by learning the underlying structure of the data, making it easier to group similar cells in a lower-dimensional space. By using the same PCs as input, we ensure that cells belonging to the previously determined graph-based clusters exhibit co-localization on the resulting dimension reduction plots.

However, it was determined that relying solely on individual sample data is inadequate for achieving effective clustering. As a result, we have undertaken the process of data integration to enhance our analysis.

3.2 Data integration:

1. Perform integration

We first identify shared 'anchors' or common points between our different datasets. These are found using a special tool designed to analyze complex biological data. Once these shared points are identified, they serve as a guide to combine the different datasets into one. The result is a unified dataset that better captures the complexity of our samples, thereby facilitating more effective data analysis.

2. Perform an integrated analysis

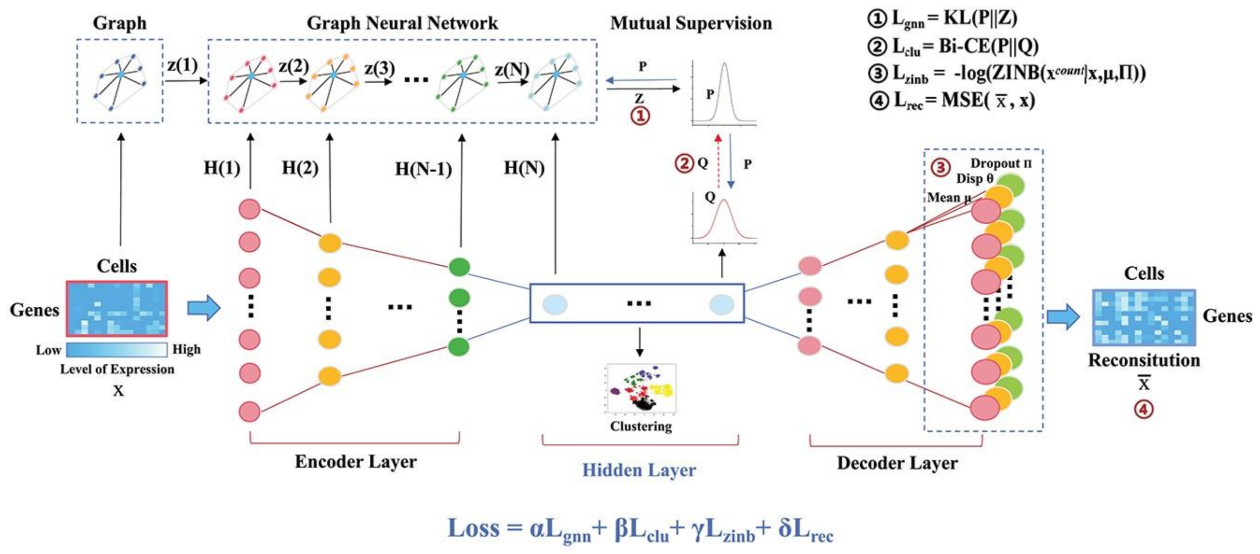
Next, we run our combined data through several processes to prepare it for visual inspection and clustering. We scale the data and apply PCA, a statistical procedure that simplifies the complexity of high-dimensional data while retaining trends and patterns. This helps us reduce the dimensionality of our dataset without losing critical information.

After PCA, we employ a technique called UMAP (Uniform Manifold Approximation and Projection) to further reduce the complexity of our data and enable two-dimensional visualization. We also use a neighbor-searching algorithm to identify related data clusters within the data.

The culmination of this process leads to the identification of data clusters with a specific level of resolution. These clusters group similar cells together based on their characteristics.

IV. Methods

The scDSC architecture is designed to learn important features of cells and genes in scRNA-Seq data for effective cell clustering. In our pipeline, we first process the scRNA-seq data and use an autoencoder with the ZINB model to reduce its dimensionality and capture important gene expression patterns. Simultaneously, we construct a KNN graph based on these patterns to understand the relationships between cells. By connecting the autoencoder with the graph neural network (GNN), we combine the learned representations and structural information. To optimize the autoencoder and GNN, we use a mutual supervision strategy. This ensures accurate cell segregation while keeping the analysis time manageable.



The overview of the proposed method scDSC, *Yanglan Gan et al.*

1. ZINB model based autoencoder module:

We use an unsupervised autoencoder based on the ZINB model to represent scRNA-seq data effectively. This model is suitable for handling gene expression data that is sparse and overdispersed.

In our approach, the input scRNA-seq data is represented as a matrix X with dimensions C (number of cells) by G (number of genes). The autoencoder consists of two parts: an encoder and a decoder. The encoder transforms the input X into a compressed representation called H, while the decoder reconstructs X from H. The encoder has multiple layers, each with weights (w) and bias (b), and it learns to transform the data at each layer. The output of the lth layer is denoted as $H_{(l)}$, and it is calculated using a function(∅) that applies the weights and bias to the previous layer's output $H_{(l-1)}$. Therefore, $H_{(l)} = \emptyset(w_{(l)} H_{(l-1)} + b_{(l)})$. The encoder stage of the autoencoder maps X to the latent representation H as $f_{enc}(WX + b)$. The decoder stage of the autoencoder maps$H$ to the reconstruction $\overline{X}$ as $f'_{dnc}(W'H + b')$.

Unlike the traditional autoencoder, the ZINB autoencoder incorporates three separate fully connected layers that connect to the last layer of the decoding layer. These layers estimate the three parameters of ZINB distribution: $Dropout = Sigmoid(b_{\pi} D)$, $Dispersion = Exp(w_{\theta} D)$, $Mean = diag(S_i) \times exp(w_{\mu} D)$. Our loss function is defined by the sum of the negative log of ZINB distribution.

2. GNN based on K-nearest neighbor graph (KNN):

We use a GNN approach, based on the KNN graph, to explore cell relationships. This involves constructing a KNN graph using gene expression data, where each node represents a cell and connections represent cell relationships.

To capture structural details, we compare four methods of constructing KNN graphs: Cosine Similarity, Pearson correlation coefficient, normalized Cosine Similarity, and Heat Kernel.

After constructing the KNN, we implement a GNN to propagate and integrate information. The GNN learns representation through weighted calculations, with the last layer connecting to the hidden layer using Softmax activation. This provides a probability distribution, indicating a cell's likelihood of belonging to a specific cluster.

This process results in two data representations: one for gene expression features and another for cell relationships. Both representations enhance the depth and breadth of our integrated dataset analysis.

3. Mutual supervision module:

By integrating the coding layer of the autoencoder with the GNN module, we create a robust data representation. However, this representation is not ready for clustering. To address this, we implement a mutual supervision strategy to train our network and perform clustering.

The strategy involves three distributions: target distribution P, clustering distribution Q, and probability distribution Z. These distributions are combined into one framework, enabling end-to-end network learning.

We assign soft labels, qit within Q, to each cell i and cluster t. These labels represent the similarity between the data representation h(i) and the clustering center vector μ(t). We use the Student t distribution to quantify the similarity.

To achieve better clustering results, we aim for the data to be close to the clustering center. We use the soft label frequency F to obtain a higher confidence data representation, pit within P. We minimize the binary cross entropy between P and Q.

In our approach, P is calculated by Q, and Q supervises P during the learning process. This mutual supervision helps the autoencoder learn a superior data representation, leading to improved clustering. The GNN propagates information layer by layer, resulting in a representation Z that includes data features and cell relationships. P also supervises Z during the learning process.

Both distribution Q and distribution Z share the common learning goal of approximating the target distribution P. This interactive relationship enables them to supervise each other in learning and optimizing the goal.

In our scDSC architecture, which includes a ZINB model and a GNN, we have four optimization objectives. The model's loss function balances these objectives, with α, β, γ, and ε controlling the relative weights of the four losses.

scDSC is implemented in Python 3 (version 3.10.10) using PyTorch (version 2.0.1).

V. Results

1. Data preprocessing:

Table 1: Data integration

Groups	#	Before Integration	After Integration
Cases (mutant IDH1)	10	In each sample: ● Rows: 32,743 human genes ● Columns: 2000+ cell types	X: 5105 (genes) × 21209 (cell types) y: 5105 (genes) × 1 (include 12 clusters)
Controls (wild type IDH1)	8		X: 1548 (genes) × 2000 (cell types) y: 1548 (genes) × 1 (include 11 clusters)

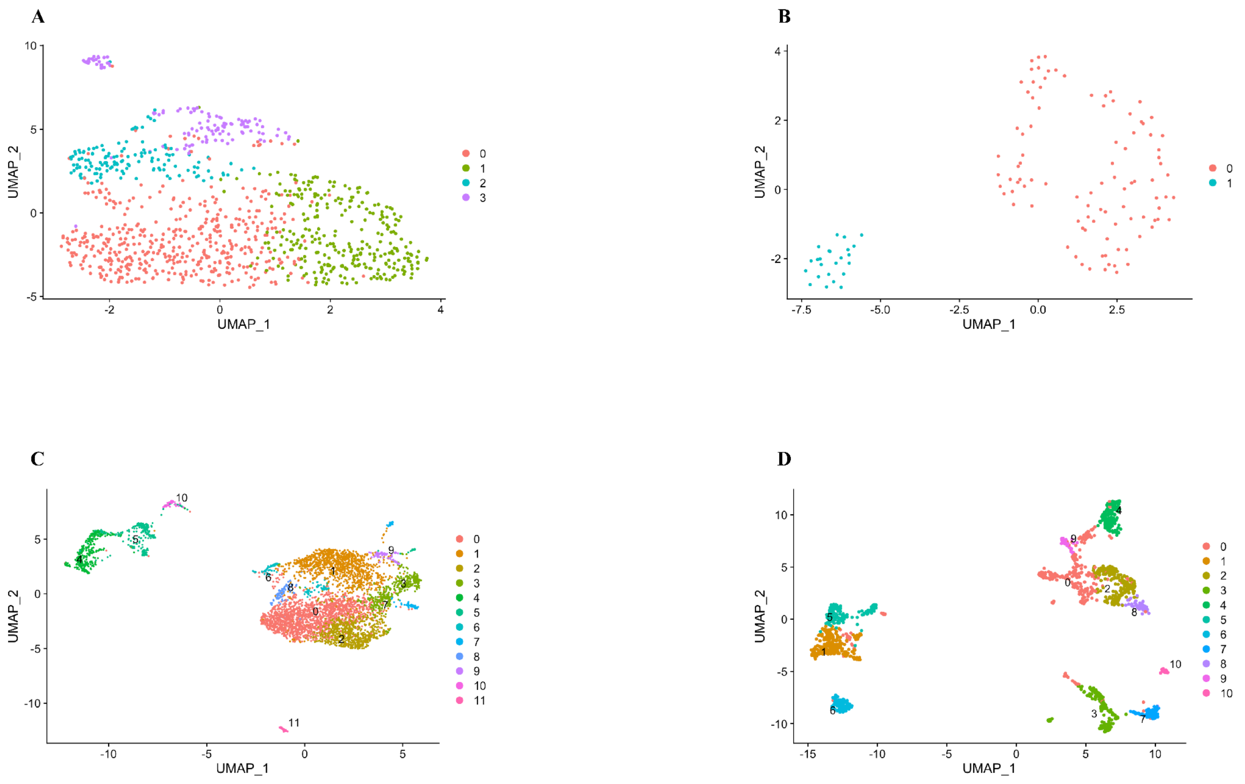


Fig. 1 - Umaps for clustering results before and after data integration: (A) single case sample. (B) single control sample. (C) integrate 10 case samples. (D) integrate 8 control samples.

2. Pre-trained model:

Table 2: Pre-trained model

Groups	Model Parameters	Optimizer Parameters	Evaluation Parameters	Results
Cases (mutant IDH1)	enc_1 (dec_3) = 500, enc_2 (dec_2) = 500, enc_3 (dec_1) = 2000	Adam batch_size = 1024, lr = 1e-3, epoch = 200	n_cluster = 5, n_init = 20, n_input = 20219 / 2000, n_z = 10	Fig. 2
Controls (wild type IDH1)				Fig. 3

* *enc_1, enc_2, enc_3 are from encoder layer, dec_1, dec_2, dec_3 are from decoder layer*

* *n_cluster and n_init are for calculate* $\hat{y}$ *by using K-Means*

* *n_input and n_z are used to define input layer and output layer variable numbers.*

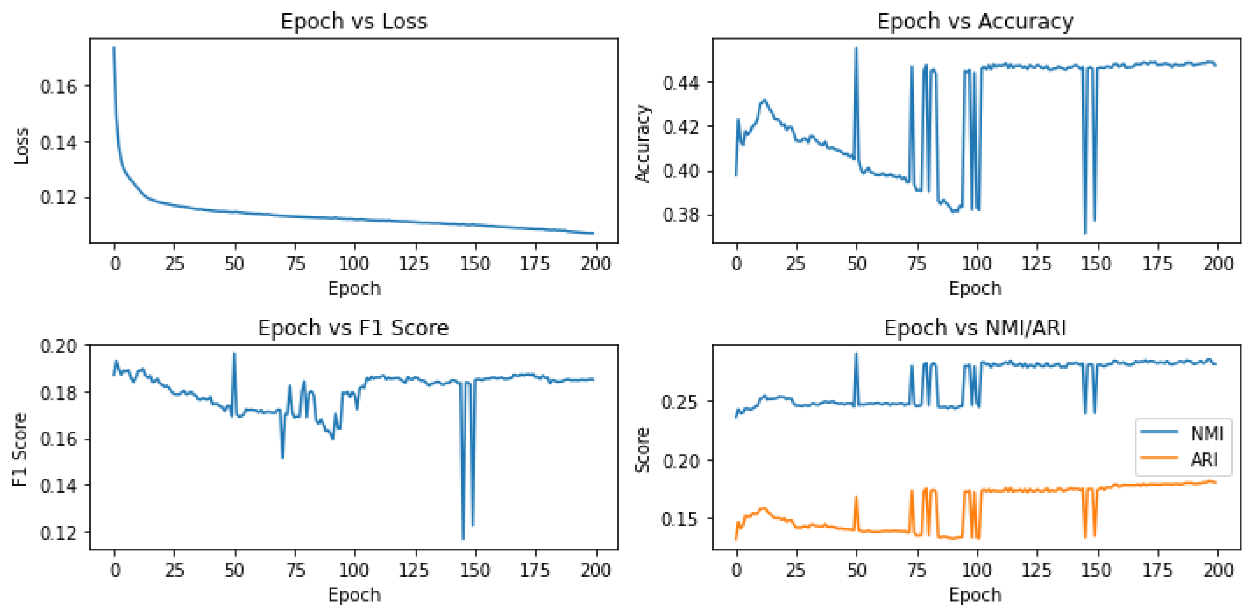


Fig.2 - Cases (mutant IDH1) Group Pre-trained Model Results

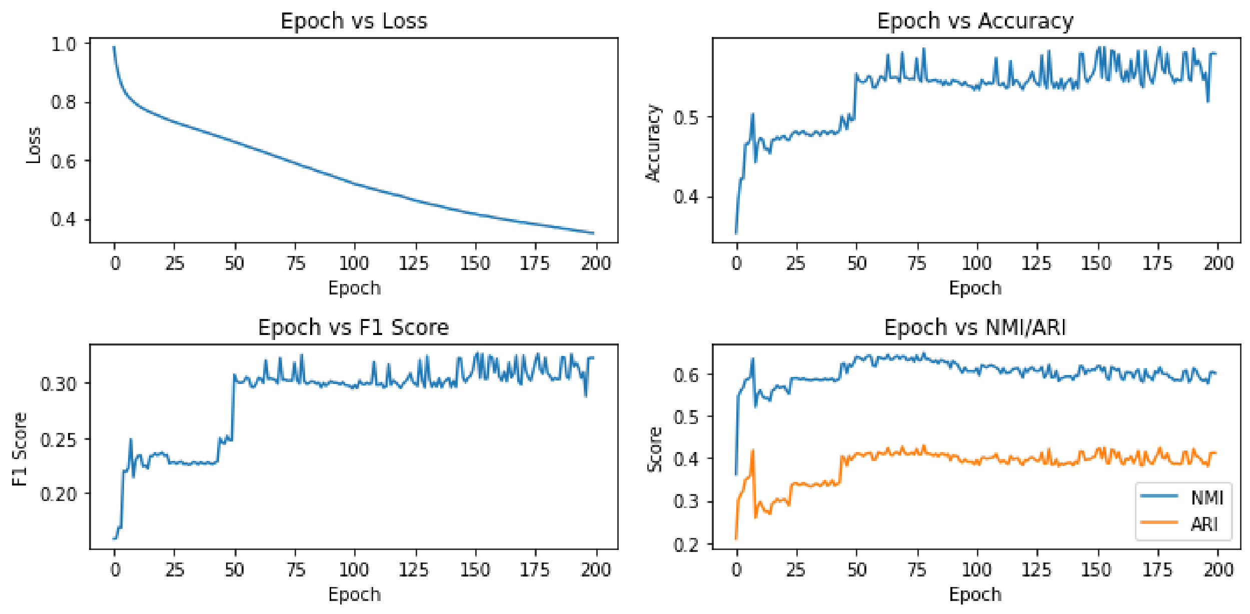


Fig. 3 - Controls (wild type IDH1) Group Pre-trained Model Results

3. Trained model:

Table 3: Trained model

Groups	Model Parameters	GNN Parameters	Optimizer	Evaluation

			Parameters	**Parameters**
Cases (mutant IDH1)	enc_1 (dec_3) = 1000, enc_2 (dec_2) = 1000, enc_3 (dec_1) = 4000	n_cluster = 7, n_z1 = 2000, n_z2 = 500, n_z3 = 10, n_input = 20219 / 2000, n_init = 20	Adam batch_size = 2048, lr = 1e-4, epoch = 80	binary_crossentropy _loss = 0.1, ce_loss = 0.01, re_loss = 1, zinb_loss = 0.1, balance = 0.5
Controls (wild type IDH1)				

** n_z1, n_z2, n_z3 are neuron numbers in each layer of the GNN models.*
** Pearson correlation coefficient is used to generate KNN models.*

(Unfortunately, we could not finish this module due to time limitations.)

VI. Conclusion

In this project, we acquired expertise in both single-cell RNA sequencing (scRNA-seq) and deep learning methodologies.

To begin with, we conducted a thorough literature review on deep learning methods for scRNA-seq analysis. During this research, we came across a fascinating paper titled 'Deep structural clustering for single-cell RNA-seq data jointly through autoencoder and graph neural network' authored by Yanglan Gan et al. This paper caught our attention due to its innovative approach. As a result, we made the decision to apply their method to a published brain cancer dataset mentioned in another paper titled 'G-CSF secreted by mutant IDH1 glioma stem cells abolishes myeloid cell immunosuppression and enhances the efficacy of immunotherapy' authored by Mahmoud S. Alghamri et al.

Similar to many data analyst projects, data processing plays a crucial role in determining the success of downstream analysis. In the case of scRNA-seq data, the preprocessing step is particularly significant and requires the utilization of well-designed pipelines. As a result, a significant portion of our project was dedicated to data processing. Initially, we started with a single sample and realized that relying solely on individual sample data was insufficient to achieve effective clustering. In response, we embarked on the process of data integration to augment our analysis and improve the quality of our results.

With the processed data, our primary focus was on utilizing a deep structural clustering method, which incorporated several key components: a ZINB model-based autoencoder module, a Graph Neural Network (GNN) module, and a mutual-supervised module, as proposed by Yanglan Gan et al. We successfully implemented a pre-trained model based on this method. However, due to time constraints, we were unable to complete the training of the model within the project timeline.

Our ultimate objective is to compare the clustering results obtained before and after applying the deep structural clustering method. Therefore, our next step involves completing the training of the model and utilizing it to generate a new set of clustering results. This will allow us to assess the effectiveness and impact of the method on the overall clustering outcome.

Reference

1. Mahmoud S. Alghamri et al., G-CSF secreted by mutant IDH1 glioma stem cells abolishes myeloid cell immunosuppression and enhances the efficacy of immunotherapy.Sci. Adv.7,eabh3243(2021).DOI:10.1126/sciadv.abh3243
2. Seurat - Guided Clustering Tutorial, https://satijalab.org/seurat/articles/pbmc3k_tutorial.html
3. Introduction to scRNA-seq integration, https://satijalab.org/seurat/articles/integration_introduction.html
4. Yanglan Gan and others, Deep structural clustering for single-cell RNA-seq data jointly through autoencoder and graph neural network, Briefings in Bioinformatics, Volume 23, Issue 2, March 2022, bbac018, https://doi.org/10.1093/bib/bbac018
5. Jovic, D., Liang, X., Zeng, H., Lin, L., Xu, F., & Luo, Y. (2022). Single‐cell RNA sequencing technologies and applications: A brief overview. Clinical and Translational Medicine, 12(3), e694. https://doi.org/10.1002/ctm2.694
6. Eraslan G, Simon LM, Mircea M, et al. Single-cell rna-seq denoising using a deep count autoencoder. Nat Commun 2019;10(1):1–14.