ABSTRACT

MODEL-BASED DEEP AUTOENCODERS FOR CHARACTERIZING DISCRETE DATA WITH APPLICATION TO GENOMIC DATA ANALYSIS

by

TIAN TIAN

Deep learning techniques have achieved tremendous successes in a wide range of real applications in recent years. For dimension reduction, deep neural networks (DNNs) provide a natural choice to parameterize a non-linear transforming function that maps the original high dimensional data to a lower dimensional latent space. Autoencoder is a kind of DNNs used to learn efficient feature representation in an unsupervised manner. Deep autoencoder has been widely explored and applied to analysis of continuous data, while it is understudied for characterizing discrete data. This dissertation focuses on developing model-based deep autoencoders for modeling discrete data. A motivating example of discrete data is the count data matrix generated by single-cell RNA sequencing (scRNA-seq) technology which is widely used in biological and medical fields. scRNA-seq promises to provide higher resolution of cellular differences than bulk RNA sequencing and has helped researchers to better understand complex biological questions. The recent advances in sequencing technology have enabled a dramatic increase in the throughput to thousands of cells for scRNA-seq. However, analysis of scRNA-seq data remains a statistical and computational challenge. A major problem is the pervasive dropout events obscuring the discrete matrix with prevailing ‘false’ zero count observations, which is caused by the shallow sequencing depth per cell. To make downstream analysis more effective, imputation, which recovers the missing values, is often conducted as the first

step in preprocessing scRNA-seq data. Several imputation methods have been proposed. Of note is a deep autoencoder model, which proposes to explicitly characterize the count distribution, over-dispersion, and sparsity of scRNA-seq data using a zero-inflated negative binomial (ZINB) model. This dissertation introduces a model-based deep learning clustering model – scDeepCluster for clustering analysis of scRNA-seq data. The scDeepCluster is a deep autoencoder which simultaneously learns feature representation and clustering via explicit modeling of scRNA-seq data generation using the ZINB model. Based on testing extensive simulated datasets and real datasets from different representative single-cell sequencing platforms, scDeepCluster outperformed several state-of-the-art methods under various clustering performance metrics and exhibited improved scalability, with running time increasing linearly with the sample size. Although this model-based deep autoencoder approach has demonstrated superior performance, it is over-permissive in defining ZINB model space, which can lead to an unidentifiable model and make results unstable. Next, this dissertation proposes to impose a regularization that takes dropout events into account. The regularization uses a differentiable categorical distribution - Gumbel-Softmax to explicitly model the dropout events, and minimizes the Maximum Mean Discrepancy (MMD) between the reconstructed randomly masked matrix and the raw count matrix. Imputation analyses showed that the proposed regularized model-based autoencoder significantly outperformed the vanilla model-based deep autoencoder.

MODEL-BASED DEEP AUTOENCODERS FOR CHARACTERIZING DISCRETE DATA WITH APPLICATION TO GENOMIC DATA ANALYSIS

by
Tian Tian

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

May 2019

MODEL-BASED DEEP AUTOENCODERS FOR CHARACTERIZING DISCRETE DATA WITH APPLICATION TO GENOMIC DATA ANALYSIS

TIAN TIAN

Dr. Zhi Wei, Thesis Advisor Date
Professor of Computer Science, NJIT

Dr. Guiling Wang, Committee Member Date
Professor of Computer Science, NJIT

Dr. Usman Roshan, Committee Member Date
Associate Professor of Computer Science, NJIT

Dr. Wenge Guo, Committee Member Date
Associate Professor of Mathematical Science, NJIT

Dr. Martin Renqiang Min, Committee Member Date
Research Staff Member, Machine Learning Group, NEC laboratories

Dr. Zhigen Zhao, Committee Member Date
Associate Professor of Statistical Science, Temple University

BIOGRAPHICAL SKETCH

Author: Tian Tian

Degree: Doctor of Philosophy

Date: May 2019

Undergraduate and Graduate Education:

- Ph.D. in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2019

- M.S. in Computer Science,
  New York University Tandon school of Engineering, Brooklyn, NY, 2014

- M.S. in Biology,
  Chinese Academy of Sciences, Shanghai, China, 2009

- B.S. in Biology,
  Wuhan University, Wuhan, China, 2009

Major: Computer Science

Publications and presentations

T. Tian, J. Wan, Q. Song, Z. Wei, Clustering Single-Cell RNA-Seq Data with A Model-Based Deep Learning Approach, *Nature Machine Intelligence* (2019) 1(4), 191-198.

F. Tan, **T. Tian***, et. al.*, Deep Learning for DNA Methylation on N6-Adenine, *to be submitted*.

T. Tian, Z. Wei, An Empirical Bayes Change Point Model for Transcriptome Time Course Data, *to be submitted*.

T. Tian, J. Wan, Y. Han, H. Liu, F. Gao, Y. Pan, Q. Song, Z. Wei, A Comprehensive Survey of Immune Cytolytic Activity-Associated Gene Co-Expression Networks Across 17 Tumor and Normal Tissue Types, *Cancers* (2018) 10 (9), 307 (IF = 5.326).

T. Tian, Z. Wei, X. Chang, Y. Liu, RE Gur, PMA Sleiman, H. Hakonarson, The Long Noncoding RNA Landscape in Amygdala Tissues from Schizophrenia Patients, *EBioMedicine* (2018) 34, 171-181 (IF = 6.183)

S. Banerjee, Z. Wei, **T. Tian**, N. Shih, M. Feldman, T. Khoury, A. Michele, J. Alwine, E. Robertson, Impact of the Breast Cancer Microbiome and Its Relationship to Disease Prognosis, *to be submitted*.

N. Auslander, G. Zhang, J. Lee, D. Frederick, B. Miao, T. Moll, **T. Tian**, Z. Wei, S. Madan, R. Sullivan, G. Boland, K. Flaherty, M. Herlyn, E. Ruppin, Robust Prediction of Response to Immune Checkpoint Blockade Therapy in Metastatic Melanoma, *Nature Medicine*, 2018 (IF = 32.621).

R. Jenkins, A. Aref, P. Lizotte, E. Ivanova, S. Stinson, C. Zhou, M. Bowden, J. Deng, H. Liu, D. Miao, M. He, W. Walker, G. Zhang, **T. Tian**, C. Cheng, Z. Wei, *et. al.*, Ex Vivo Profiling of PD-1 Blockade Using Organotypic Tumor Spheroids, *Cancer Discovery* (2018) 8 (2), 196-215 (IF = 24.373)

G. Zhang, L. Wu, M. Rokni, M. Hammond, O. Ope, C. Cheng, S. Randell, N. Sadek, A. Beroard, M. Xiao, **T. Tian**, *et. al.*, Induction of Telomere Dysfunction Prolongs Disease Control of Therapy-Resistant Melanoma, *Clinical Cancer Research* (2018): clincanres-2773. (IF = 10.199)

S. Banerjee, **T. Tian**, Z. Wei, N. Shih, M. Feldman, K. Peck, A. DeMichele, J. Alwine, E. Robertson, Distinct Microbial Signatures Associated with Different Breast Cancer Types, *Frontiers in Microbiology* 9 (2018). (IF = 4.019)

A. Sahu, J.S. Lee, G. Zhang, Z. Wang, **T. Tian**, *et. al.*, Genome-Wide Prediction Of Synthetic Rescue Mediators Of Resistance To Targeted And Immunotherapy, *bioRxiv*, 284240

R. Somasundaram, G. Zhang, M. Kalabis, M. Perego, C. Krepler, X. Xu, C. Wagner, D. Hristova, J. Zhang, **T. Tian**, Z. Wei, *et. al.*, Tumor-Associated B-Cells Induce Tumor Heterogeneity and Therapy Resistance, *Nature Communications* 8.1 (2017): 607. (IF = 12.353)

I Julia, J. Leu, T. Barnoud, G. Zhang, **T. Tian**, Z. Wei, M. Herlyn, M. Murphy, D. George, Inhibition Of Stress-Inducible HSP70 Impairs Mitochondrial Proteostasis And Function, *Oncotarget* 8.28 (2017): 45656. (IF = 5.168)

S. Banerjee, **T. Tian**, Z. Wei, K. Peck, N. Shih, A. Chalian, B. O'malley, G. Weinstein, M. Feldman, J. Alwine, E. Robertson, Microbial Signatures Associated With Oropharyngeal And Oral Squamous Cell Carcinomas, *Scientific Reports* 7.1 (2017): 4036. (IF = 4.122)

S. Banerjee, **T. Tian**, Z. Wei, N. Shih, M. Feldman, J. Alwine, G. Coukos, E. Robertson, The Ovarian Cancer Oncobiome, *Oncotarget* 8.22 (2017): 36225. (IF = 5.168)

D. Zhang, G. Zhang, X. Hu, L. Wu, Y. Feng, S. He, Y. Zhang, Z. Hu, L. Yang, **T. Tian**, W. Xu, Z. Wei, *et. al.*, Oncogenic RAS Regulates Long Non-Coding RNA Orilnc1 In Human Cancer, Cancer Research (2017): canres-1768. (IF = 9.130)

G. Zhang, D. Frederick, L. Wu, Z. Wei, C. Krepler, S. Srinivasan, Y. Chae, X. Xu, H. Choi, E. Dimwamwa, O. Ope, B. Shannan, D. Basu, D. Zhang, M. Guha, M. Xiao, S. Randell, K. Sproesser, W. Xu, J. Liu, G Karakousis, L. Schuchter, T. Gangadhar, R. Amaravadi, M. Gu, C. Xu, A. Ghosh, W. Xu, **T. Tian**, J. Zhang, *et. al.*, Targeting Mitochondrial Biogenesis to Overcome Drug Resistance To MAPK Inhibitors, The Journal of Clinical Investigation 126.5 (2016): 1834-1856. (IF = 13.251)

Dedicate to my parents and friends.

Tian Tian

ACKNOWLEDGMENT

First and foremost, I wish to take this opportunity to express my heartfelt appreciation to my doctoral advisor Dr. Zhi Wei. It is his generous help and sustained encouragement that bring me the courage to overcome the challenges in the road of pursuing the scientific truth. Dr. Wei is my friend and mentor. With his tremendous effort, invaluable guidance, and infinite patience, I am able to bring this dissertation to its culmination. I will always be indebted to Dr. Wei for generously supporting and inspiring me in this critical stage of my life.

Second, I am extremely grateful to Dr. Guiling Wang, Dr. Usman Roshan, Dr. Wenge Guo, Dr. Martin Renqiang Min and Dr. Zhigen Zhao for serving on my committee. They have provided me with academic advice. This dissertation would not have been possible without their invaluable guidance and generous help. In addition, I would like to extend special thanks to my collaborators, Dr. Ji Wan from CuraCloud Inc., Dr. Gao Zhang from University of Duke, and Dr. Sagarika Banerjee from University of Pennsylvania. I wish to thank Dr. Ali Mili, Dr. Reza Curtmola, Dr. Cristian M. Borcea, Dr. George Olsen, Ms. Clarisa Gonzalez-Lenahan for supporting and helping me all the time. I would also like to thank my lab mates and all graduate colleagues for their assistance and support.

Third, I thank my beloved parents for their endless love, support, and encouragement. Lastly, but not the least, I am thankful to all who have helped me directly or indirectly for the past five years. It is their support and encouragement that give me the courage and strength to pursue my PhD degree abroad.

TABLE OF CONTENTS

Chapter **Page**

LIST OF FIGURES

LIST OF TABLES

Table **Page**

CHAPTER 1

BACKGROUND

Recently developed single-cell mRNA-sequencing (scRNA-seq) methods have enabled unbiased, high-throughput, and high-resolution transcriptomic analysis of individual cells. They provide an additional dimension to transcriptomic information relative to traditional methods that profile bulk populations of cells [1]. A major analysis for scRNA-seq is the unbiased identification of cellular subpopulations from heterogeneous populations of cells. The identity of cell types can reveal the insights of functions and regulatory mechanisms. However, a major challenge of scRNA-seq data analysis is the pervasive existing of dropout events which are caused by the shallow sequencing depth per cell. In this dissertation, we develop model-based deep autoencoder for clustering and imputation of scRNA-seq data.

The organization and contributions of this dissertation are outlined as following:

Chapter 2: In this Chapter, we discussed the major challenge in the analysis of scRNA-seq data and the current state-of-arts statistical and machine learning methods. These methods include state-of-arts multi-kernel spectral clustering methods – SIMLR and MPSSC, and the imputation and clustering method – CIDR. We also briefly introduced the deep learning method – autoencoder and a special autoencoder designed for analysis of the discrete scRNA-seq count data – deep count autoencoder (ZINB model-based autoencoder). Finally, a recent classical deep learning approach for clustering, the deep embedding clustering (DEC) algorithm, is introduced.

Chapter 3: In this Chapter, we have developed scDeepCluster, a single-cell model-based deep embedded clustering method, which simultaneously learns feature representation and clustering via explicit modeling 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. The material reported in this chapter is published in Tian *et al.* [2].

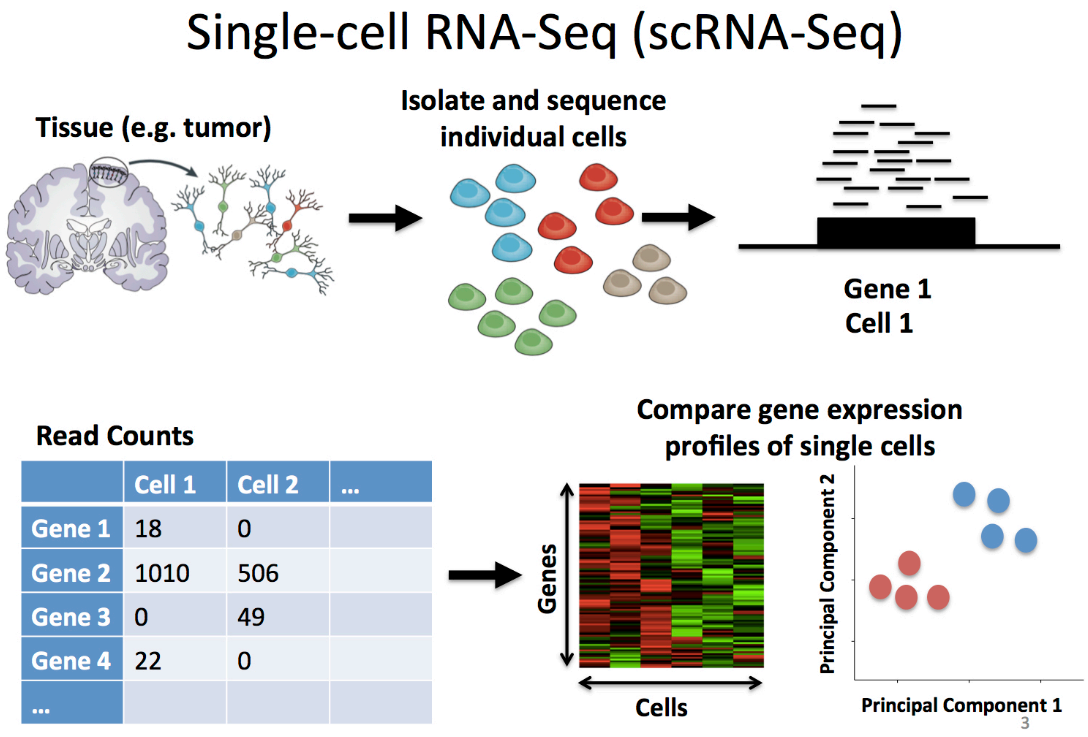
Chapter 4: In this chapter, we introduce a Maximum Mean Discrepancy regularization that further improves the imputation performance of the model-based deep autoencoder. The regularization uses a differentiable categorical distribution - Gumbel-Softmax to explicitly model the dropout events, and minimizes the Maximum Mean Discrepancy (MMD) between the reconstructed randomly masked matrix and the raw count matrix. The imputation results on simulated data show that the regularization can improve the model-based autoencoder in estimating missing values. We expect to do more simulation and real-data experiments to evaluate this regularization.

CHAPTER 2

STATISTICAL AND MACHINE LEARNING METHODS FOR ANALYSIS OF SCRNA-SEQ DATA

2.1 Single Cell RNA-seq Technology

Cell is a fundamental unit in biology. Multi-cellular organisms are composed of a plethora of cells with distinct types. Cell types can be distinguished by their shape and size under a microscope. Single cell sequencing technologies have made it possible to discover cell types at molecular level [3]. Specifically, single-cell RNA sequencing (scRNA-seq) can be used to define cell types, which profiles all gene expression activities (transcriptome) of each cell. Cells are compared based on their transcriptome similarity and tend to cluster into different groups based on their cell types. From the analytic point of view, this cell type discovery procedure is considered as unsupervised clustering. To sequence mRNA from a single cell, two major steps are (1) to capture single cells, and (2) to amplify and sequence mRNAs produced from a cell (**Figure 2.1**). Technologies designed for capturing single cells include micro-pipetting micromanipulation, laser capture microdissection, fluorescence activated cell sorting (FACS) and microdroplets, etc. After isolating single cells, the next step is to use unique molecular identifiers (UMIs) to barcode each individual mRNA molecule with a cell. UMIs can let us track the single cell origin of each mRNA molecule. The following steps, reverse transcription, cDNA amplification, and sequencing library preparation, are the same as sequencing bulk mRNAs. Briefly, mRNA is fragmented into pieces, which is then reversely transcribed into cDNA followed by amplification. One or two ends of the cDNA fragments are sequenced, resulting in millions of short reads, between 50 to 200 base pairs in length. The reads go through quality control, which removes low quality bases and sequencing primers. After that, the reads are typically then mapped to a reference genome. The mapping procedure is to find the matching positions in the reference genome of the reads. The mapped reads are then generally pooled into regions of interest for summarization. The most common case is to pool and count reads by gene, in which case the data consist of nonnegative counts indicating the number of reads observed for each gene. Therefore, like the RNA-seq technology, scRNA-seq generates discrete count data. Different to bulk RNA-seq reads, scRNA-seq reads have the UMI sequences, which can be used to track the cell origin of each mRNA molecule. As a result, the output of scRNA-seq takes the form of n x p matrices, representing the read counts mapped to the p genes across n cells. The read count values represent the relative expression levels of each mRNA in cells. Higher count value means higher relative expression levels of this gene. Library size is the total number of reads that were aligned to the reference genome per sample/cell. A relative expression level of a gene in a cell is represented by the proportion of the read counts of this gene in the library size of the cell. To compare gene expression levels across samples/cells, the read counts need to be normalized by the library size and other biases. The software for quality control, alignment and read count summarization and normalization have been developed for scRNA-seq data. To list a few, they are Seurat[4] and cell rangers. In this dissertation, we will assume that quality control, mapping, pooling, and summarization of the raw reads have already been performed. Our analysis starts from the read count matrices. Owing to the low initial amounts of RNA obtained from each single cell, scRNA-seq data exhibit much higher levels of noise and much more zeros. The false zero values can rise from the steps of both library preparation and sequencing. During library preparation, some mRNA molecule might be lost due to the tiny initial amounts. The sequencing step can also generate “false” zero values, caused by the relative shallow sequencing depth per cell. It is common to observe more than 50% entries of the measured data matrix are zero values [3]. There is a trade-off between sequencing depth and sample size (the number of cells that can be profiled), given the same budget. The larger sample size, the lower sequencing depth, and then the more dropout events. As a result, most, if not all, scRNA-seq studies suffer from dropout events. To address this issue and make downstream analysis more effective, many methods have been proposed. We introduce some representative ones here.


Single-cell RNA-Seq (scRNA-Seq)
Tissue (e.g. tumor)
Isolate and sequence individual cells
Gene 1
Cell 1
Read Counts
Compare gene expression profiles of single cells
Genes
Cells
Principal Component 2
Principal Component 1
3

	Cell 1	Cell 2	...
Gene 1	18	0	
Gene 2	1010	506	
Gene 3	0	49	
Gene 4	22	0	
...			

Figure 2.1 (Source: https://learn.gencore.bio.nyu.edu/single-cell-rnaseq/, accessed on April 29 2019) Illustration of the procedure of scRNA-seq experiment and analysis.

2.2 Multi-kernel Based Spectral Clustering

SIMLR – single-cell interpretation via multi-kernel learning was one of the earliest method proposed for the analysis of scRNA-seq data[5]. A key challenge for analysis of scRNA-seq data is the pervasive dropout events commonly encountered in single-cell sequencing. . Unsupervised clustering, dimension reduction, and visualization all rely on similarity metrics that would be heavily interfered by dropout events. Missing values will corrupt similarity matrix estimation. SIMLR uses multi-kernels to mitigate this issue. The overview of SIMLR is given in **Figure 2.2**. Briefly, to overcome the issues of noise and dropout events, SIMLR learns a robust distance metric that best fits the structure of the data by combining multiple Gaussian kernels. The robust similarities learned by SIMLR can be adapted into downstream clustering and dimension reduction.

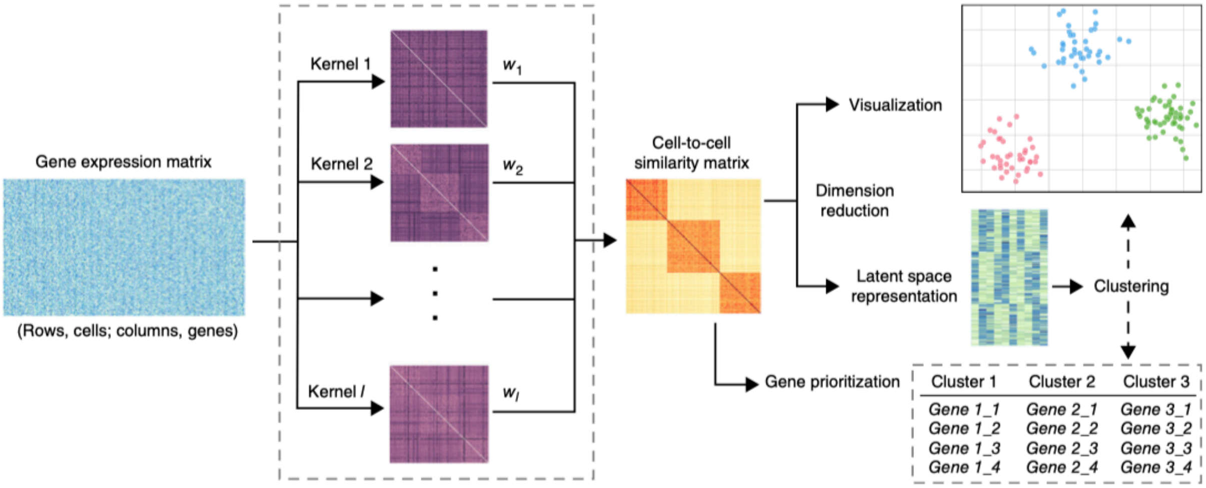


Figure 2.2 (Source: ref [5]) The overview of multi-kernel clustering for scRNA-seq data.

SIMLR represents one of the state-of-arts clustering algorithms for scRNA-seq data. Later, an improved version , MPSSC – spectral clustering based on learning similarity matrix, has been proposed [6]. MPSSC is also based on spectral clustering. The similarities or the Laplace matrix is learned by SIMLR's multi-kernel procedure. MPSSC furtherly introduce a stronger sparsity constraint, which is shown to bring better clustering performance. Both SIMLR and MPSSC require the calculation of the similarity matrix or the Laplace matrix, which, however, is computationally expensive. The space complex (memory to store) for the Laplace matrix is quadratic or super-quadratic. It requires large memory to analyze large datasets such as thousands of samples. This limitation restricts the application of these multi-kernel spectral methods to large datasets.

2.3 Imputation of scRNA-seq data

As mentioned earlier, the large number of missing values due to dropout events are the major challenge in analysis of scRNA-seq data. A direct approach to address this problem is to do imputation (estimation of missing values). If imputation can recover missing values accurately, down-stream dimension reduction and clustering analyses would be more effective. Clustering analysis is primarily based on the similarity. CIDR – Clustering through Imputation and Dimensionality Reduction is one example of the imputation and clustering methods that have been proposed for scRNA-seq [7]. CIDR uses a simple implicit imputation approach that borrows information from near neighbors to alleviate the impact of dropouts in scRNA-seq data. For gene *k* in each pair of cell *i* and *j*, if we define the empirical dropout rate is *P* and the real expression is X_{kj}, the observed expression is

$$\hat{X}_{ki} = \left(1 - P(X_{kj})\right) X_{kj} + P(X_{kj}) X_{ki}$$

Then the similarity metric between i and j is calculated between $\hat{X}_{.i}$ and $\hat{X}_{.j}$, and it is followed by a hierarchical clustering based on the first few principal coordinates. Many other methods have been introduced for imputation. The scImpute is the first method dedicated for cell-wise and genomic-wise imputation [8]. The first step of the scImpute is to detect cell subpopulations and outliers, which is achieved by PCA followed by spectral clustering to determine cell groups. Next, it uses a mixture model to decide which zero counts are caused by dropout, namely, false zeros to be imputed. Then, it uses the non-missing values of a candidate neighbor set of cells similar to the target cell to impute its false zero counts (missing values). The scImpute essentially relies on the correlation structure of scRNA-seq data and imputes missing values by leveraging information on the similarities between cells and/or genes.

2.4 Autoencoder and Model-based Autoencoder

An autoencoder is a neural network which learns an efficient compressed representation of data in an unsupervised fashion (**Figure 2.3**). The autoencoder has a low dimensional bottleneck layer and learns to reconstruct its input.

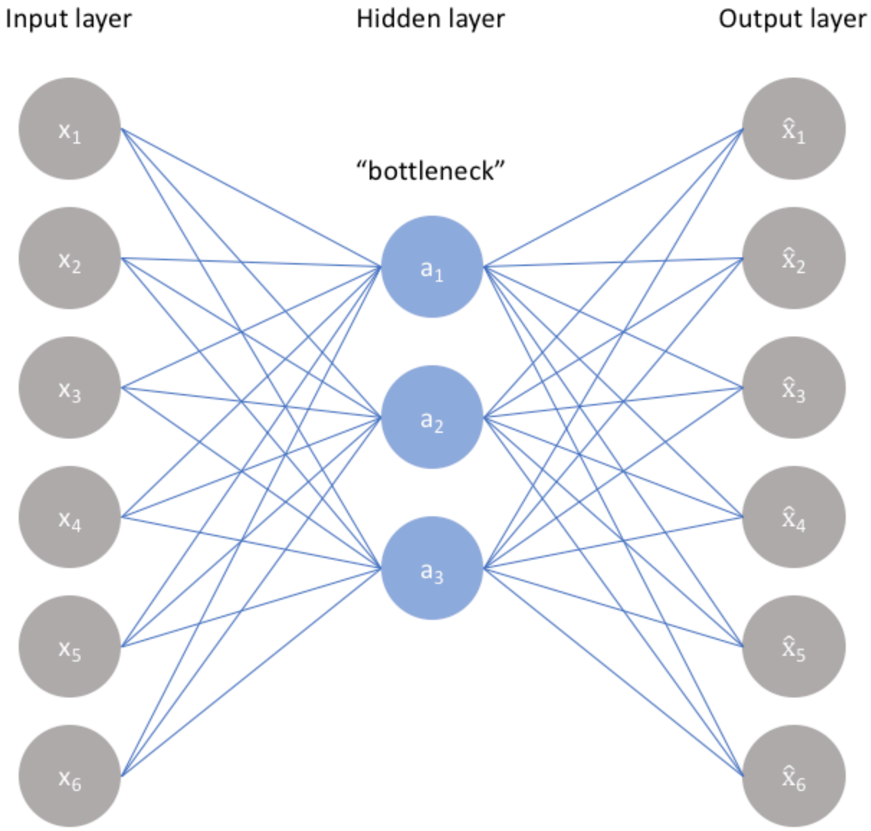


Figure 2.3 (Source: https://www.jeremyjordan.me/autoencoders/, accessed on April 28, 2019) The architecture of autoencoder

.

Each layer in an autoencoder can be a fully connected or a more complex layer (such as convolution layer). Formally, assume x is the input and z is the output

$$z = \sigma(Wx + b)$$

Here σ is the activation function. These layers can be stacked and construct a deep autoencoder. The typical autoencoder learns to reconstruct it's input. The loss function is mean squared errors (MSE);

$$L(x, x') = \|x - x'\|^2$$

The compression forces the autoencoder to learn the essential latent features and the reconstruction ignores non-essential features such as noise. Autoencoder is a widely used method in unsupervised learning

The simplest and most common autoencoder is the so called under-complete autoencoder, in which the bottleneck layer usually has much smaller dimensions than the input samples'. By optimizing the autoencoder network according to minimize the reconstruction loss, the model is forced to learn the most important attributes of the input data. This under-complete autoencoder can be considered as a generalization of principal component analysis (PCA). Not like PCA which is limited to linear transformation, autoencoder can capture the more complex relationship between samples by learning a non-linear manifold.

Autoencoder has many variants. We introduce two important ones: denoising autoencoder and variational autoencoder [9] here.

Denoising autoencoder takes a partially corrupted input and is trained to recover the original undistorted input. Based on the properties of data, the choice of noise can be different kinds. For continuous data, Gaussian noise is a common choice. For the binary data, such as the MNIST hand writing digit images (only white or black), the noise can be a flipping of the binary values. The technique has been introduced with a specific approach to learn good representation. Since it learns a more robust representation when the input is corrupted by the random noise. The denoising autoencoder can also be stacked and noise can be added in each encoder layer [10, 11].

Since the most interesting ability of autoencoder is its ability to learn the latent representation that can reflect the major variations in the data. In some cases, we assume the lae tent representation can be characterized by some pre-defined distributions (prior distribution) to improve the latent feature learning. The variational autoencoder is designed for this purpose. Variational autoencoder can be either deterministic or non-deterministic. For the non-deterministic autoencoder, it introduces random variances into the latent space (**Figure 2.4**). In the framework of variational autoencoder, it models the output of the bottleneck layer by a prior distribution. The most common choice is the standard Gaussian distribution. Other distributions can be used for different purposes, such as a Gaussian mixture model for different groups of data. To encourage the learned bottleneck representation to be similar to the prior, a regularization of Kullback-Leibler (KL) divergence is added

$$L(x, x') + KL(q(z|x)||p(z))$$

where z is the latent representation and $p(z)$ is the prior distribution. By adding the regularization, the variational autoencoder can learn the feature representations to be similar to the prior distribution. Variational autoencoder can also be used as a generative model. We can use prior distribution to randomly generate a latent representation, then the decoder can transfer this latent representation to a new generated data sample. The recent advances of variational autoencoder include replacing KL-divergence to the more informative f-divergence (Wasserstein distance or Maximum Mean Discrepancy) and using more complex prior distributions[12, 13].

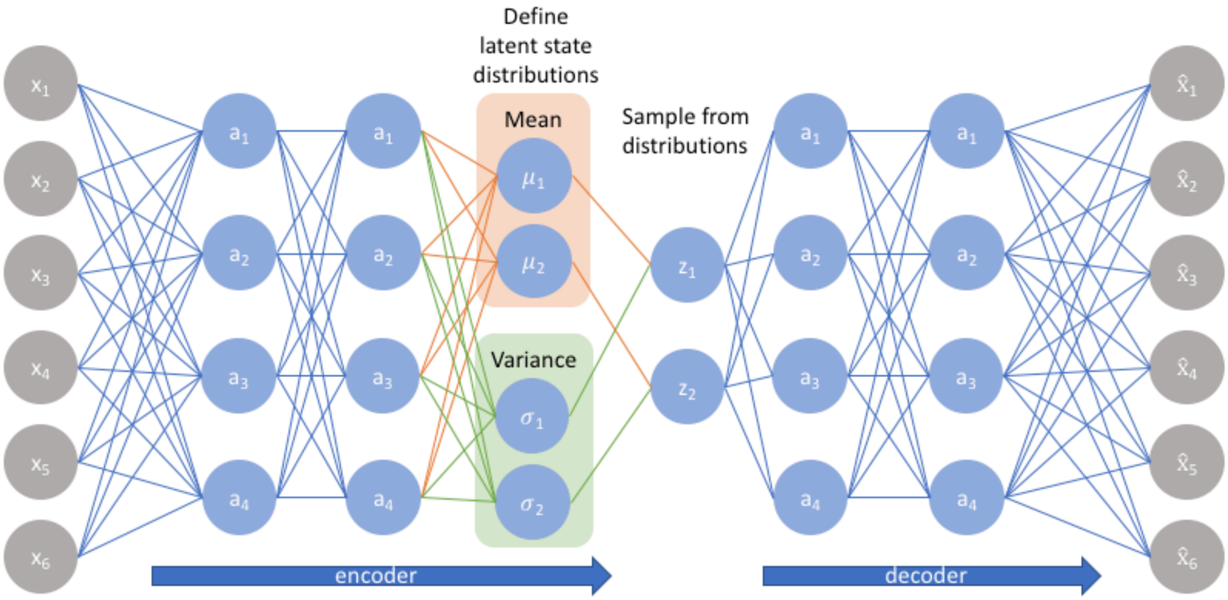


Figure 2.4 (Source: https://www.jeremyjordan.me/variational-autoencoders/, accessed on April 28, 2019) The architecture of the non-deterministic autoencoder.

To characterize the discrete scRNA-seq count data, zero-inflated model-based autoencoder (Deep Count Autoencoder, DCA) has been introduced [14]. RNA-seq count data can be characterized by the negative binomial distribution [15, 16]. Due to the pervasive zero values, the scRNA-seq count data can be modeled by zero-inflated negative binomial (ZINB) distribution. To denoise and impute the scRNA-seq data, ZINB model-based autoencoder uses the trick to define the reconstruction error as the likelihood of ZINB distribution instead of constructing the input data itself (**Figure 2.5**).

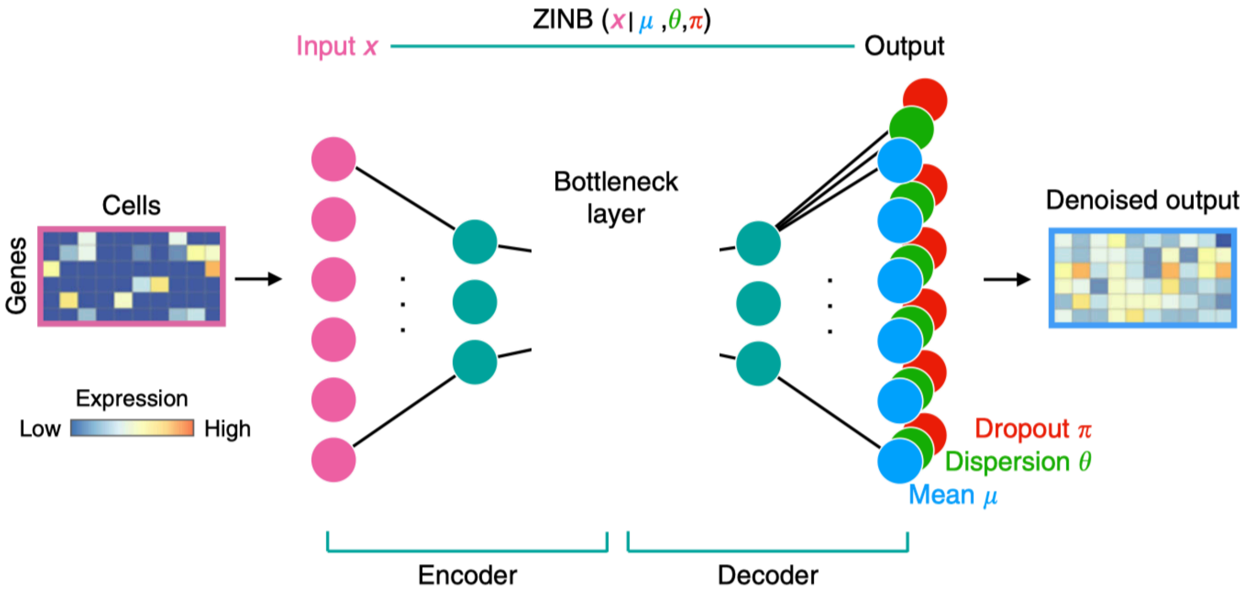


Figure 2.5 The overall architecture of ZINB model-based autoencoder (Source: ref [14]).

The ZINB distribution has three parameters: mean (μ), dispersion (θ) and dropout (π) which are learned by the autoencoder. For each input *X*, the ZINB model-based autoencoder learns the likelihood of it.

$$ZINB(X|\pi, \mu, \theta) = \pi\delta_0(X) + (1 - \pi)NB(X|\mu, \theta)$$

The loss function of ZINB autoencoder is to maximize the likelihood of all of *X*s. Note the ZINB likelihood is differentiable and can be calculated in the back-propagate optimization algorithm. Imputation and clustering both rely on the correlation or similarity structure between cells, and the nature of autoencoder is learning the major latent features, so the ZINB model-based autoencoder is supposed to perform nicely on these tasks. The empirical experiments illustrate this conclusion. ZINB model-based autoencoder represents the first application of autoencoder for latent feature learning of the discrete count data.

The variational autoencoder has also been introduced for scRNA-seq analysis. The scVI and scvis are variational autoencoders but uses a complex hierarchical Bayesian model as the prior distribution to capture the scRNA-seq data with high dropout events and dispersions [17, 18].

2.5 Deep Embedded Clustering

The majority of researches in deep neural networks focus on supervised learning. Clustering is an essential unsupervised machine learning task that groups input samples based on their similarities. Due to the "curse of dimensionality", clustering performances are much better in low dimensions than high dimensions. The feature representation ability makes autoencoder an appealing method for clustering. Deep embedding clustering (DEC) is the first method that combines the autoencoder feature learning and clustering [19].

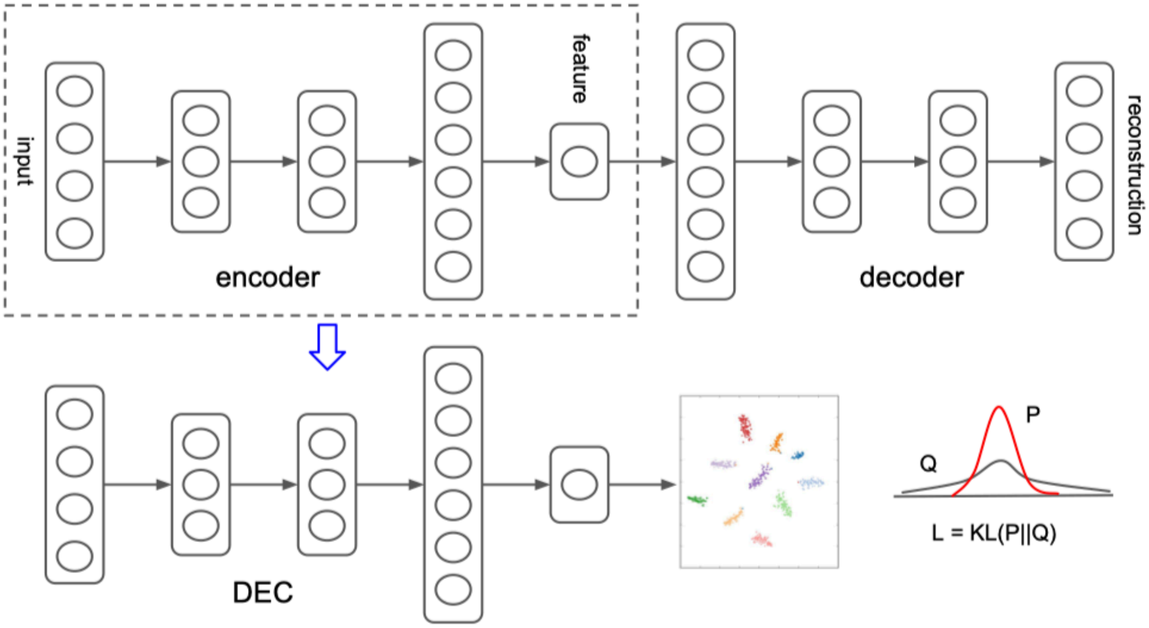


Figure 2.6 Network structure of deep embedding clustering (Source: ref [19])

DEC clusters data by two parts: (1) latent feature representation learned by a deep autoencoder and (2) clustering optimization (**Figure 2.6**). DEC clusters data by simultaneously learning a set of k cluster centers in the latent space. The clustering part in DEC is called soft assignment which can be updated by the stochastic gradient descent. Consider the problem of clustering a set of n samples X with each sample $x_i \in \mathbb{R}^d$ into k clusters. Instead of directly clustering in the data space X, deep embedded clustering first finds a non-linear mapping $f_W : x_i \to z_i$, where Z is the latent feature space and Z is typically much smaller than X. The clustering will be applied to the latent space Z. The non-linear mapping of X to Z is learned by a deep autoencoder. The clustering algorithm is defined as Kullback-Leibler (KL) divergence between distribution P and Q, where Q is the distribution of soft label assignments measured by Student's *t*-distribution and P is the derivation of the target distribution from Q. Formally, the clustering loss is

$$L_c = KL(P \parallel Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

where q_{ij} is the soft label of embedded point z_i, which is defined as the similarity between z_i and cluster center μ_i measured by Student's t-distribution [20, 21]:

$$q_{ij} = \frac{\left(1 + \|z_i - \mu_j\|^2\right)^{-\frac{\alpha+1}{2}}}{\sum_{j'} \left(1 + \|z_i - \mu_{j'}\|^2\right)^{-\frac{\alpha+1}{2}}}$$

The empirical results suggest the choice $\alpha = 1$.Meanwhile, p_{ij} is the target distribution computed by first raising q_{ij} to the second power and then normalizing by frequency per cluster:

$$p_{ij} = \frac{q_{ij}^2 / \sum_j q_{ij}}{\sum_{j'} \left(q_{ij'}^2 / \sum_{j'} q_{ij'} \right)}$$

This training strategy can be seen as self-training [22], since the target distribution P is defined based on Q.

The original DEC uses the separated optimization procedure, which means it optimize the deep autoencoder first, then use this pretrained autoencoder weights as the initial weight to optimize the clustering part. Next, the algorithm improved deep embedding clustering (IDEC) proposes to combine the two parts and optimize them simultaneously that can get better clustering results [23]. The feature representation ability of the ZINB model-based autoencoder and the advantage of combining clustering and feature learning in DEC bring us the idea for developing the ZINB model deep autoencoder clustering method for scRNA-seq data.

CHAPTER 3

MODEL-BASED DEEP AUTOENCODER FOR CLUSTERING ANALYSIS OF SINGLE CELL RNA-SEQ DATA

3.1 Introduction

Single-cell RNA sequencing (scRNA-seq) can reveal heterogeneity and diversity among cell populations, which has helped researchers to better understand complex biology questions [24, 25]. Clustering analysis has been routinely conducted in most scRNA-seq studies. Clustering is a classical unsupervised machine learning problem and has been studied extensively in the past decades. Many popular methods have been proposed, such as k-means [26], Gaussian Mixture Models (GMM) [27] and spectral clustering [28]. However, clustering groups of cells in scRNA-seq datasets is still a statistical and computational challenge. The major problem for clustering of scRNA-seq data, comparing to bulk RNA-seq data and microarray data, is that they are so sparse that most of the measurements are zeros, due to the low RNA capture rate. The missing measurement of genes is defined as dropout event which results in a “false” zero count observation. Recent advantages of sequencing platforms have enabled a dramatic increase in the throughput of thousands of cells [1, 29-31]. These technologies are particularly prone to dropout events due to relatively shallow sequencing depth per cell [32]. In addition, scRNA-seq exhibits high variations in gene expression levels even among cells from one group. Together, these technical and biological factors introduce substantial variations and noises, which makes clustering a particularly challenging task.

Several clustering methods have been developed recently to overcome these challenges. For instance, Xu and Su propose SNN-Clip, a method that utilizes the concept of the shared nearest neighbor. Such a strategy can effectively handle high-dimensional data [33]. Many researches use sophisticated techniques that involve iterative clustering. These methods would detect relationships between the subtypes, which are further validated by differential gene expression analysis [29, 34, 35]. DendroSplit [36] is an interpretable clustering framework that uses feature selection to uncover multiple levels of biologically meaningful populations in the scRNA-seq data. Wang *et al*. employ multi-kernel learning (SIMLR) for single-cell interpretation [5]. SIMLR essentially is a spectral clustering method based on multiple kernels. It combines multiple kernels to learn a robust distance metric that best fits the structure of the data. To characterize the sparsity of scRNA-seq data, Park *et al*. present also a multi-kernel spectral clustering method, but propose to impose a sparse structure (MPSSC) via L1 penalty [6]. These very recent two methods represent current state-of-arts clustering approaches for scRNA-seq data. Although these spectral clustering-based methods have shown decent performance, they have two issues. First of all, they rely on the full graph Laplacian matrix, which is prohibitively expensive to compute and store. The computing and storing of the Laplacian matrix usually have quadratic or super quadratic complexities in terms of the number of data points (cell numbers here); the decomposing of the matrix even requires cubic complexities [37]. Such complexities imply a severe scalability issue for spectral clustering. It is typical to cluster thousands of cells in many scRNA-seq studies, which requires to run on super machines with huge memory. For example, a machine with 800 Gb memory was used to cluster thousands of cells in [6]. This limitation restricts the application of spectral based clustering methods to large scRNA-seq datasets, which are typical outputs of modern scRNA-seq platforms, as they can usually profile thousands or even tens of thousands of cells simultaneously. Secondly, spectral clustering methods don't model the characteristics of scRNA-seq count data such as over-dispersion and zero-inflation explicitly, leaving room for further improvement.

Another line of research, which is relevant, focuses on imputing missing values (false zeros) caused by dropout in scRNA-seq data. Such imputation is expected to improve various downstream analyses, including clustering. Several methods and tools have been developed recently in this regard, including statistical models CIDR [7], scImpute [8], MAGIC [38] and SAVER [39], to name a few, and deep learning approaches DeepImpute [40], DCA [14], and scScope [41]. CIDR (Clustering through Imputation and Dimensionality Reduction) is a fast PCA-like algorithm that takes dropouts into account. It incorporates a simple implicit imputation approach to alleviate the impact of dropouts in scRNA-seq data, followed by clustering based on the first few principal coordinates. DeepImpute applies standard deep neural network to predict missing values of target genes using highly correlated genes with sufficient reads coverage as input. Both scScope and DCA are based on autoencoder. Autoencoder is a kind of deep neural networks (DNNs) used to learn efficient feature representation in an unsupervised manne r[42]. The scScope essentially repeats running a regular autoencoder three times, using the final output from the previous run as the initial input for the next run. Compared with regular autoencoder, the DCA (deep count autoencoder) proposes to replace the conventional mean square error (MSE) loss function with a zero-inflated negative binomial (ZINB) model-based loss function for better characterizing scRNA-seq data. We have used the ZINB model for differential expression analysis of microbiome sequencing data [43], and show it can effectively characterize discrete, over-dispersed and zero-inflated count data. The results of DCA also show that its imputation using ZINB can improve a diverse set of typical downstream scRNA-seq data analyses [14]. Such improvements suggest that the ZINB model is effective at characterizing the pervasive dropout events, the main statistical challenge in scRNA-seq data.

However, these methods focusing on imputation are not designed and optimized for clustering, although we can, as a naive solution, impute scRNA-seq data first, which is then followed by simple clustering using, e.g., k-means. Such a divided strategy is suboptimal for clustering, as shown in the comparison of our method with the DCA.

Because of "curse of dimensionality", clustering performs much better on a small dimensionality than on a high one [44]. Deep neural networks (DNNs) are a natural choice to parameterize a non-linear transforming function that maps the original high dimensional data to a small latent space. DNNs have revealed the theoretical function approximation capability [45] and demonstrated feature learning properties [46]. Recent studies have illustrated that deep learning can successfully gain good performance on clustering tasks when applied to image and text datasets [19, 23]. Meanwhile, a study demonstrates that DNNs can reduce the dimensions of scRNA-seq data in a supervised manner [47]. In contrast, a recently published deep generative model scvis [18] is proposed to capture and visualize the low-dimensional structure in scRNA-seq data in an unsupervised manner.

Therefore, we develop a deep learning clustering method, which integrates the ZINB model with clustering loss in a principled way (see **Figure 3.1** for its architecture). Our method aims to optimize clustering explicitly while performing dimension reduction.

In the developed framework, the non-linear function mapping the read count matrix of scRNA-seq data to a low dimensional latent representation is learned by the ZINB model-based autoencoder, while the clustering task on the latent space is performed by clustering with Kullback-Leibler (KL) divergence as described in the "deep embedded clustering" (DEC) algorithm [19]. We further introduce the denoising autoencoder technique [10, 11] to ZINB model-based autoencoder, which makes it more powerful to learn a robust feature representation of data. We name the method as single-cell model-based deep embedded clustering (scDeepCluster: https://github.com/ttgump/scDeepCluster, accessed on April 28, 2019). Using both simulated and real scRNA-seq data, we demonstrate that scDeepCluster brings significant accuracy improvement over the competing start-of-art clustering methods. Furthermore, we show that scDeepCluster requires less memory and time complexity than competing methods. This appealing computation efficiency makes scDeepCluster more feasible for analysis of large scRNA-seq data.

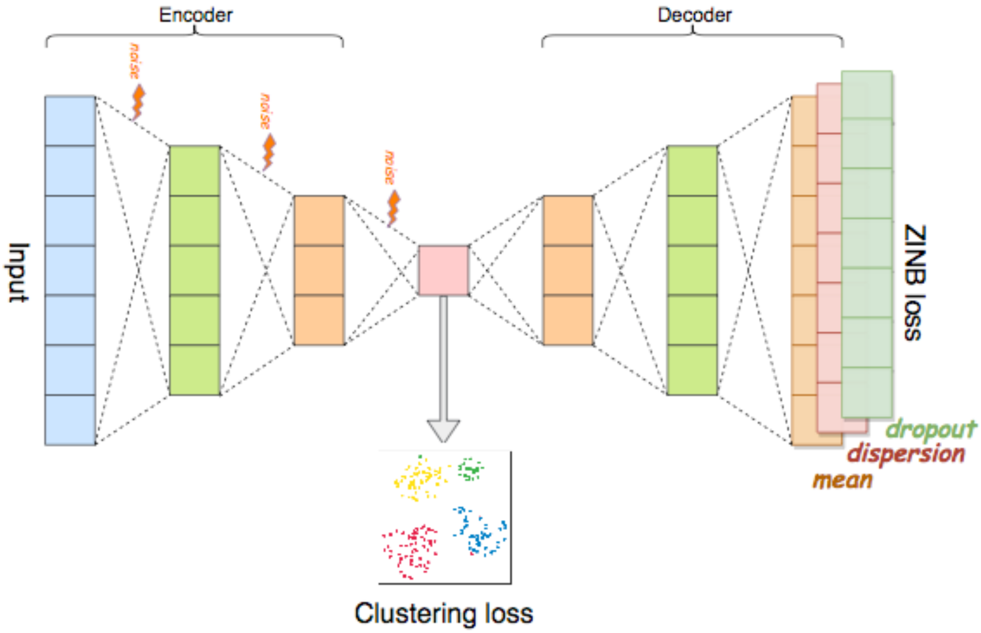


Figure 3.1 The network architecture of scDeepCluster. The encoder and decoder are fully connected neural networks. Clustering loss (KL-divergence) is applied to scatter the embedded points z. The ZINB (zero-inflated negative binomial) loss has three components: mean, dispersion and dropout, which are estimated by three individual fully connected layers with different activation functions. The random Gaussian noise has been incorporated into the encoder to improve the embedded feature representation.

3.2 Methods

3.2.1 Read count data preprocessing

Raw scRAN-seq read count data is preprocessed by the Python package SCANPY [48]. First, genes with no count in any cell are filtered out. Second, size factors are calculated and read counts are normalized by library size, so total counts are same across cells. Formally, if denoting library size (number of the total read counts) of cell i as s_i, then size factor of cell i is $s_i/median(s)$. The last step is to take log transform and scale of read counts, so that count values follow unit variance and zero mean. The preprocessed read count matrix is treated as input for our denoising ZINB model-based autoencoder.

3.2.2 Denoising ZINB model-based autoencoder

Autoencoder is a kind of artificial neural networks used to learn efficient feature representation in an unsupervised manner [42]. The denoising autoencoder is an autoencoder that receives a corrupted data points as input and is trained to predict the original uncorrupted data point as its output [10, 11]. The autoencoder usually has a low dimensional bottleneck layer to learn a latent feature representation. Denoising autoencoder is proved to be more powerful to learn a robust representation of data, since it has the ability to learn the representations of the input that are corrupted by small irrelevant changes in the input. Here, we apply the denoising autoencoder technique to mapping input of read counts to an embedded space to do clustering. In practice, we first corrupt the input by the random Gaussian noise, then construct the autoencoder by the regular fully connected layers. Formally, input X is corrupted by noise

$$X^{corrupt} = X + e$$

where e represents the random Gaussian noise. Note that the noise can be incorporated into every layer of the encoder, which is defined as a stacked denoising autoencoder [11]. We define encoder function as $z = f_W(X^{corrupt})$ and decoder function $X' = g_{W'}(z)$. Encoder and decoder functions are both fully connected neural networks with the rectifier activation [49]. Here W and $W'$are the learned weight of the functions. The learning process of denoising autoencoder is to minimize the loss function

$$L\left(X, g_{W'}\left(f_W(X^{corrupt})\right)\right)$$

where L is the loss function.

To capture the characters of scRNA-seq data, we apply ZINB model-based autoencoder [14] instead of a regular autoencoder which is trained to attempt to reconstruct its input. Not like the regular autoencoder, the loss function of the ZINB model-based autoencoder is the likelihood of zero-inflated negative binomial distribution (ZINB). ZINB is applied to characterize the dropout events in scRNA-seq. Formally, ZINB is parameterized with the mean (μ), the dispersion (θ) of the negative binomial distribution and with an additional coefficient (π) that represents the weight of point mass of probability at zero (the probability of dropout events)

$$NB(X^{count}|\mu,\theta) = \frac{\Gamma(X^{count}+\theta)}{X^{count}!\ \Gamma(\theta)}\left(\frac{\theta}{\theta+\mu}\right)^{\theta}\left(\frac{\mu}{\theta+\mu}\right)^{X^{count}}$$

$$ZINB(X^{count}|\pi,\mu,\theta) = \pi\delta_0(X^{count}) + (1-\pi)NB(X^{count}|\mu,\theta)$$

where X^{count} represents the raw read counts. The ZINB model-based autoencoder estimates the parameters μ, θ and π. Say $D = g'_{W'}\left(f_W(X^{corrupt})\right)$ represents the last hidden layer of decoder, we append three independent fully connected layers to D to estimate parameters

$$\mathrm{M} = diag(s_i) \cdot exp\left(W_\mu\, D\right)$$

$$\Theta = exp(W_\theta\, D)$$

$$\Pi = sigmoid(W_\pi\, D)$$

where M, Θ, and Π represent the matrix form of estimations of mean, dispersion and dropout probability, respectively. The size factors s_i are calculated in the part of "data preprocess" and included as an independent input to the deep learning model. The activation function chosen for mean and dispersion is exponential since mean and dispersion parameters are non-negative values; the activation function for the additional coefficient π is sigmoid which represents the dropout probability. Dropout probability is in the interval of 0 to 1, so sigmoid is a suitable choice of the activation function. The loss function of ZINB model-based autoencoder is the negative log of ZINB likelihood

$$L_{ZINB} = -\log\left(ZINB(X^{count}|\pi, \mu, \theta)\right)$$

3.2.3 Deep embedded clustering with local structure preservation

The clustering stage follows the deep embedded clustering [19, 23]. Consider the problem of clustering a set of n cells X with each sample $x_i \in \mathbb{N}^d$ (x_i represents the read counts of d genes in the ith cell) into k clusters. Instead of directly clustering in the data space X, deep embedded clustering first finds a non-linear mapping $f_W : x_i \to z_i$, where Z is the latent feature space and Z is typically much smaller than X. The clustering will be applied to the latent space Z.

In order to capture the characters of single-cell RNA-seq data, we apply denoising ZINB model-based autoencoder to learn the non-linear mapping of X to Z (we build a clean encoder sharing the same weights but without Gaussian noise for the clustering layer, Appendix Figure 1). The clustering algorithm is same as [19], which is defined as Kullback-Leibler (KL) divergence between distribution P and Q , where Q is the distribution of soft labels measured by Student's *t*-distribution and P is the derivation of the target distribution from Q. Formally, the clustering loss is

$$L_c = KL(P \parallel Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

where q_{ij} is the soft label of embedded point z_i, which is defined as the similarity between z_i and cluster center μ_i measured by Student's t-distribution [20, 21]:

$$q_{ij} = \frac{\left(1 + \left\|z_i - \mu_j\right\|^2\right)^{-1}}{\sum_{j'}\left(1 + \left\|z_i - \mu_{j'}\right\|^2\right)^{-1}}$$

Meanwhile, p_{ij} is the target distribution computed by first raising q_{ij} to the second power and then normalizing by frequency per cluster:

$$p_{ij} = \frac{q_{ij}^2 / \sum_j q_{ij}}{\sum_{j'}\left(q_{ij'}^2 / \sum_{j'} q_{ij'}\right)}$$

This training strategy can be seen as self-training [22], since the target distribution P is defined based on Q.

We pre-train the stacked denoising ZINB model-based autoencoder before the clustering stage. The initializing of cluster centers is obtained by standard k-means clustering in the embedded feature space after the pre-training of denoising ZINB model-based autoencoder.

As a result, the model has two components: the denoising ZINB model-based autoencoder and the clustering part. Then the objective function of scDeepCluster is

$$L = L_{ZINB} + \gamma L_c$$

where L_{ZINB} and L_c are the ZINB loss function of denoising ZINB model-based autoencoder and the clustering loss respectively, and $\gamma > 0$ is the coefficient that controls the relative weights of the two losses. According to [23], this form of loss has the advantage that it can preserve the local structure of the data generating distribution.

3.2.4 Optimization

The deep neural network (DNN) and cluster centers $\{\mu_j\}$ are jointly optimized by stochastic gradient descent (SGD) and backpropagation. The gradients of clustering loss L_c with respect to embedded point z_i and cluster center μ_j are computed as:

$$\frac{\partial L_c}{\partial z_i} = 2\sum_j (1 + \|z_i - \mu_j\|^2)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j)$$

$$\frac{\partial L_c}{\partial \mu_j} = -2\sum_i (1 + \|z_i - \mu_j\|^2)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j)$$

During optimization, given a mini-batch size m and a learning rate lr, cluster center μ_j is updated by

$$\mu_j = \mu_j - \frac{lr}{m}\sum_{i=1}^{m} \frac{\partial L_c}{\partial \mu_j}$$

The decoder's weight W' is updated by

$$W' = W' - \frac{lr}{m}\sum_{i=1}^{m} \frac{\partial L_{ZINB}}{\partial W'}$$

The encoder's weight W is updated by

$$W = W - \frac{lr}{m}\sum_{i=1}^{m} (\frac{\partial L_{ZINB}}{\partial W} + \gamma \frac{\partial L_c}{\partial W})$$

where L_{ZINB} is the ZINB loss and γ is the coefficient that controls the relative weights of the ZINB and clustering losses. To discover cluster assignments, the optimization procedure will stop when less than $tol\%$ of points change cluster assignment between two consecutive iterations. Note the above procedure is described in [19, 23].

3.2.5 Implementation

scDeepCluster is implemented in Python 3 using Keras (https://github.com/keras-team/keras, accessed on April 28, 2019) with TensorFlow [50] backend. The random Gaussian noise is implemented by the Keras layer “GaussianNoise”, and the noise level (argument *"stddev"*) is set to 2.5. The denoising ZINB model-based autoencoder is first pre-trained by 400 epochs (for the simulated data, we pre-trained 600 epochs) by the optimizer AMSGrad variant of Adam [51, 52] with the setting of initial learning rate $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. The optimizer for clustering stage is Adadelta [53] with the setting of $lr = 1.0$, $rho = 0.95$. The sizes of hidden fully connected layer in encoder are set to (256, 64), the decoder is the reverse of the encoder, and the bottleneck layer (the latent space) has the size of 32. The choice of coefficient γ is 1. The batch size of pre-training and clustering is 256. The convergence threshold for clustering is 0.1% of the delta clustering labels per batch. All experiments were conducted on Nvidia Tesla P100 (16G).

3.2.6 Competing methods

DCA[14], SIMLR[5], MPSSC[6], CIDR[7], PCA + k-means, scvis[18] and DEC[19] are used as competing methods. DCA is conducted directly by using the authors’ API functions (https://github.com/theislab/dca, accessed on April 28, 2019). DCA is not designed for clustering. To do clustering, we first apply the DCA (with the default parameters given by the authors) to denoise the raw read count data (impute the dropouted counts); then reduce the high-dimensional denoised read count matrix to the 2-D space by PCA (principal component analysis). K-means clustering was conducted on the projected 2-D space. This method is called as “DCA + k-means”. We preprocess the read count matrix then use the preprocessed data as the input for the SIMLR, PCA + k-means and MPSSC. First, the read count matrix is normalized by library size, so total counts are the same across cells. Next, normalized read counts are log transformed. SIMLR is a spectral clustering method, where similarities between cells are learned by multi-kernel. SIMLR is set to use default settings. MPSSC is a multi-kernel spectral clustering framework with the imposition of sparse structures on a target matrix. The parameters for MPSSC are $rho = 0.2$, $lam = 0.0001$, $lam2 = 0.0001$, $eta = 1$, $c = 0.1$. "PCA + k-means" is a method that applies PCA to project the processed raw read count matrix to 2-D space directly then followed by k-means clustering. We follow the steps described by the authors for CIDR (https://github.com/VCCRI/CIDR, accessed on April 28, 2019). The input for CIDR is a *scData* R object constructed by the raw count matrix. The clustering steps for CIDR include determining the dropout events and imputation weighting thresholds, computing CIDR dissimilarity matrix, reducing dimensionality and clustering. We use the first two principal components computed by CIDR to show the latent representations. The scvis is a variational autoencoder [54] based model to capture the low-dimensional representation of scRNA-seq data. We use scvis to reduce scRNA-seq data to 2-D space then apply k-means clustering. For scvis, we follow the preprocessing steps described by the authors: the expression of each gene is quantified as $\log_2$(CPM/10+1), where "CPM" standards for "counts per million". Next, the data is projected to a 100-dimensional space by PCA which is used as input for scvis. DEC (https://github.com/XifengGuo/DEC-keras, accessed on April 28, 2019) use the same inputs as scDeepCluster: the raw count matrix is library-size normalized, log transformed, scaled and centered. The hyperparameters in DEC remain the same as the authors' originals (e.g. the sizes of hidden layers are 500, 500, 2000, 10).

3.2.7 Evaluation metrics

All clustering results are measured by Normalized Mutual Information (NMI) [55], Clustering Accuracy (CA) and Adjust Rand Index (ARI).

Given the two clustering assignments U and V on a set of n data points, which have C_U and C_V clusters, respectively. Then NMI is defined as mutual information between U and V divided by the entropy of the clustering U and V. Specifically,

$$NMI = \frac{\sum_{p=1}^{C_U} \sum_{q=1}^{C_V} \left|U_p \cap V_q\right| \ \log \frac{n\left|U_p \cap V_q\right|}{\left|U_p\right| \times \left|V_q\right|}}{\max\left(-\sum_{p=1}^{C_U} \left|U_p\right| \log \frac{\left|U_p\right|}{n}, -\sum_{q=1}^{C_V} \left|V_q\right| \log \frac{\left|V_q\right|}{n}\right)}$$

The CA is defined as the best matching between a cluster assignment and the ground truth assignment. Given a data point i, Let l_i be the ground truth label, and u_i be the assignment of the clustering algorithm, then the CA is defined as

$$CA = \max_{m} \frac{\sum_{i=1}^{n} 1\{l_i = m(u_i)\}}{n}$$

where n is the number of data points, and m ranges over all possible one-to-one mapping between cluster assignments and true labels. The best mapping can be efficiently found by the Hungarian algorithm [56].

Rand index [57] is a simple measure of agreement between two cluster assignments U and V. The Adjust Rand Index (ARI) corrects for the lack of a constant value of the Rand index when the cluster assignments are selected randomly [58]. The ARI is calculated by four quantities. Specifically, we define a, the number of pairs of two objects in the same group in $both$ U and V; b, the number of pairs of two objects in different groups in both U and V; c, the number of pairs of two objects in the same group in U but in different groups in V; and d, the number of pairs of two objects in different groups in U but in the same group in V. The ARI is formally defined as

$$ARI = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c) + (c+d)(b+d)]}{\binom{n}{2} - [(a+b)(a+c) + (c+d)(b+d)]}$$

3.2.8 Data simulation

Simulated data are generated by the Splatter R package [59]. The R function *splatSimulate* is used to simulate the scRNA-seq count data. In all settings, we simulated three cell groups, 1500 cells of 2500 genes, and 20 datasets were repeatedly generated for each setting.

For the simulation of various dropout rates, we set the parameter *dropout.shape* = -1, and vary *dropout.mid* from -0.5 to 1 (-0.5, 0, 0.5, 1, the corresponding dropout rates are 12±0.3%, 17±0.4%, 23±0.5%, 30±0.6%, 39±0.6%, default *dropout.mid* in Splatter is 0), *de.fracScale* = 0.3, the other parameters are set to be default. Each setting generates 20 datasets with different random seeds. After generating the read count matrix, Splatter implements a logistic function to produce a probability that a count should be zero. The read count matrix before dropout is the true counts, and the read count matrix after dropout is the raw counts which are used as input to do clustering. Due to the randomness, datasets with the same setting have slightly different dropout rates. We summarize the mean and standard deviation of dropout rates of each setting. The formula to calculate the dropout rate is defined as

$$dropout\ rate = \frac{number\ of\ 0\ counts\ in\ raw\ counts - number\ of\ 0\ counts\ in\ true\ counts}{number\ of\ counts > 0\ in\ true\ counts}$$

To simulate datasets with various signal strengths, we vary the parameter *de.fracScale* in 0.2, 0.225, 0.25, 0.275, and fix the *dropout.shape* = -1, *dropout.mid* = 0, the other parameters are set to be default. In Splatter, the parameter *de.fracScale* is the sigma parameter of a log-normal distribution that controls the multiplicative differential expression factors.

We simulated a large dataset of 100k cells by 3000 genes to evaluate the running time of scDeepCluster. The 100k cells are divided into 10 groups. The parameters are: *dropout.shape* = -1, and vary *dropout.mid* = 0, *de.fracScale* = 0.3. We down-sampled the simulated large dataset to different cell numbers: 5k, 7.5k, 10k, 25k, 50k, 75k and 100k. Down-sampling do not drop any group.

3.2.9 Real data

10X PBMC dataset (4K PBMCs from a healthy donor) is provided by 10X scRNA-seq platform [30], which profiles the transcriptome of the peripheral blood mononuclear cells (PBMCs) from a healthy donor. The total number of cells is about 4000. PBMC 4k data is downloaded from the website of 10X genomics (https://support.10xgenomics.com/single-cell-gene-expression/datasets/2.1.0/pbmc4k, accessed on April 28, 2019). We download filtered gene/cell matrix and cell labels identified by graph-based clustering (method description: https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/output/analysis, accessed on April 28, 2019).

Mouse embryonic stem cells dataset [1] is downloaded from GSE65525 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE65525, accessed on April 28, 2019). The transcriptomes are profiled by the droplet-microfluidic approach for parallel barcoding. Allon M.K. *et.al* analyzed the heterogenous onset of differentiation of mouse embryonic stem cells after leukemia inhibitory factor (LIF) withdrawal. We download the read count matrices of mouse ES cells sample 1, mouse ES cells LIF- 2days, mouse ES cells LIF- 4days and mouse ES cells LIF- 7days, and put all cells together. The labels of cells are defined as the intervals after LIF withdrawal.

Mouse bladder cells dataset of the Mouse Cell Atlas project [31] is provided by the authors (https://figshare.com/s/865e694ad06d5857db4b, accessed on April 28, 2019). We download the raw digital expression matrix of the all 400,000 single cells sorted by tissues and the table of cell assignments. The authors identified the cell types and described the method in [31]. From the raw count matrix, we select the cells from bladder tissue.

Worm neuron cells dataset is profiled by sci-RNA-seq (single-cell combinatorial indexing RNA sequencing) [60]. The authors profiled about 50,000 cells from the nematode *Caenorhabditis elegans* at the L2 larval stage and identified the cell types (http://atlas.gs.washington.edu/worm-rna/docs/, accessed on April 28, 2019). We select the subset of the neural cells and removed the cells with the label of "Unclassified neurons". As a result, we get 4186 neural cells.

We further evaluate the performance of scDeepCluster on 2 large real datasets (more than 10k cells): PBMC 68k[30] (https://github.com/10XGenomics/single-cell-3prime-paper, accessed on April 28, 2019) and mouse retina cells[61] (https://scrnaseq-public-datasets.s3.amazonaws.com/scater-objects/shekhar.rds, accessed on April 28, 2019). We failed to run CIDR on the PBMC 68k cells even with 141G memory, so we down-sampled the dataset to 20k cells for CIDR only (other methods on PBMC 68k use the full dataset).

3.3 Results

3.3.1 Simulation evaluation

To evaluate the performance of scDeepCluster in clustering analysis of scRNA-seq, we designed the following simulations under extensive settings approximating different biological scenarios. Specifically, we applied R package Splatter [59] to simulate scRNA-seq read count data. We simulated 1500 cells with 2500 genes each, which formed three groups of the same size (500 cells/group). We used the following three performance metrics to evaluate the consistency between the obtained clustering and the true labels: Normalized Mutual Information (NMI) [55], Clustering Accuracy (CA) and Adjusted Rand Index (ARI) [58]. The ranges of NMI and CA are from 0 to 1, and ARI can yield negative values. The three metrics are statistics of concordance of two clustering labels; the higher the values, the higher concordance of the clustering. We repeated all experiments under the same setting 20 times. We compared the proposed method scDeepCluster with seven competing methods: the DCA[14] + k-means, two multi-kernel spectral clustering methods MPSSC [6] and SIMLR [5], CIDR[7], PCA + k-means, scvis[18] + k-means, and deep embedding clustering (DEC)[19].

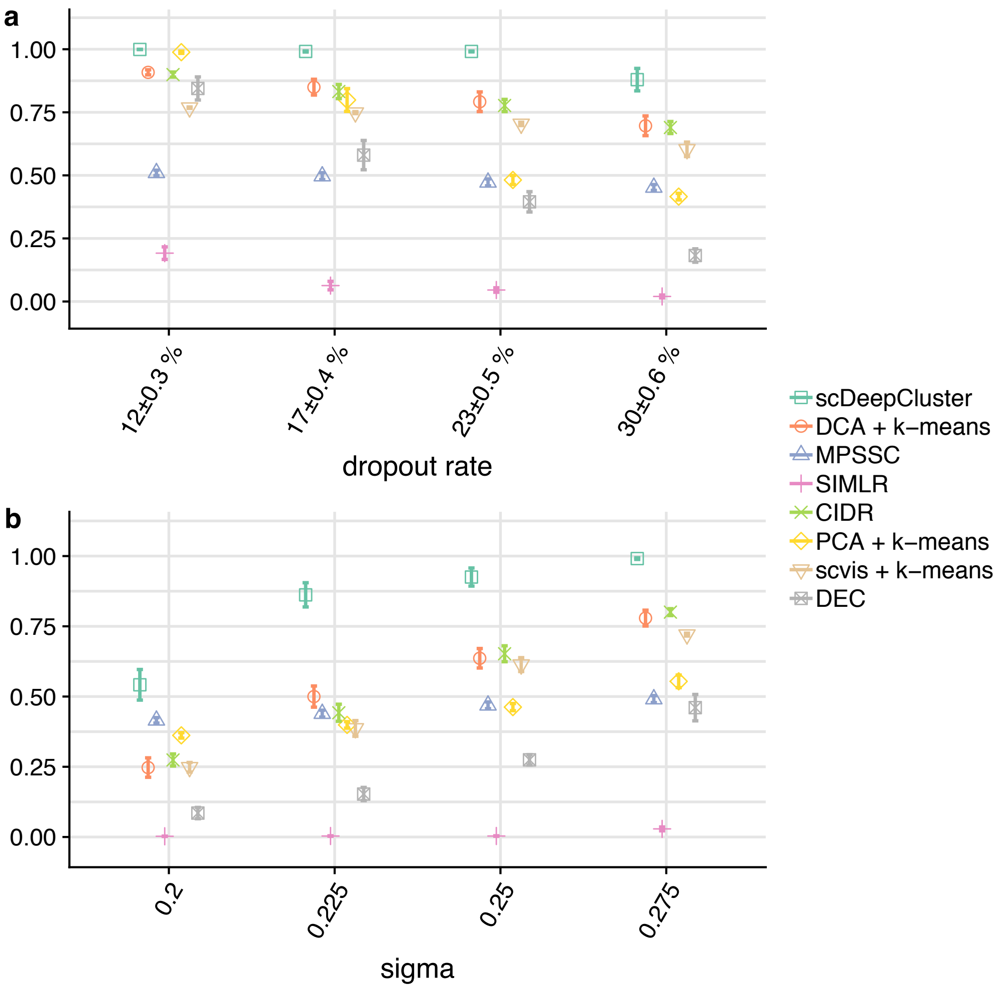


Figure 3.2 Clustering performance, measured by Normalized Mutual Information (NMI), of scDeepCluster, DCA + k-means, MPSSC, SIMLR, CIDR, PCA + k-means, scvis + k-means and DEC on simulated data with **(a)** various dropout rates, and **(b)** various clustering signal strengths (the larger the sigma value, the stronger the signal). The averaged NMIs over 20 repeats with standard errors are shown for each simulation setting. The larger NMI means more concordance between the predicted labels and the true labels.

We first investigated the performance of the eight methods under different dropout rates, which are defined as the proportion of the expressed genes being knocked out of read counts. To do that, we varied the midpoint parameter of the dropout logistic function to generate datasets with different dropout rates (12±0.3%, 17±0.4%, 23±0.5%, and 30±0.6%). The accuracy in terms of NMI (averaged over the 20 repeats within each setting) is presented in **Figure 3.2a**, and CA and ARI in **Figure 3.3**. We note several interesting findings. First, all methods have a decreasing accuracy with the increasing dropout rate, confirming our speculation that dropout event makes clustering challenging. Especially, for PCA + k-means and DEC, their performance decreased dramatically with the increasing of dropout rates, but the DCA can improve over it, which illustrates the imputation effect of the DCA. The improvement brought by DCA confirms that special modeling the characteristics of scRNA-seq data can enhance the standard DEC model. Second, scDeepCluster outperformed the other competing methods consistently under all dropout rate settings significantly (*p-value* < 0.01, one-sided paired t-test). Third, scDeepCluster was more robust against the increasing dropout rate. The clustering performance of scDeepCluster was perfect (NMI ≈ 1) until the dropout rate increased to 30% but remained decent (NMI ≈ 0.9). In contrast, the performances of the other competing methods began to deteriorate quickly and significantly when the dropout rate increased. These superior performances suggest that scDeepCluster can effectively characterize dropout events in scRNA-seq data. In fact, the ZINB loss function not only helps scDeepCluster to improve clustering performance but also makes its performance more stable (**Figure 3.4**).

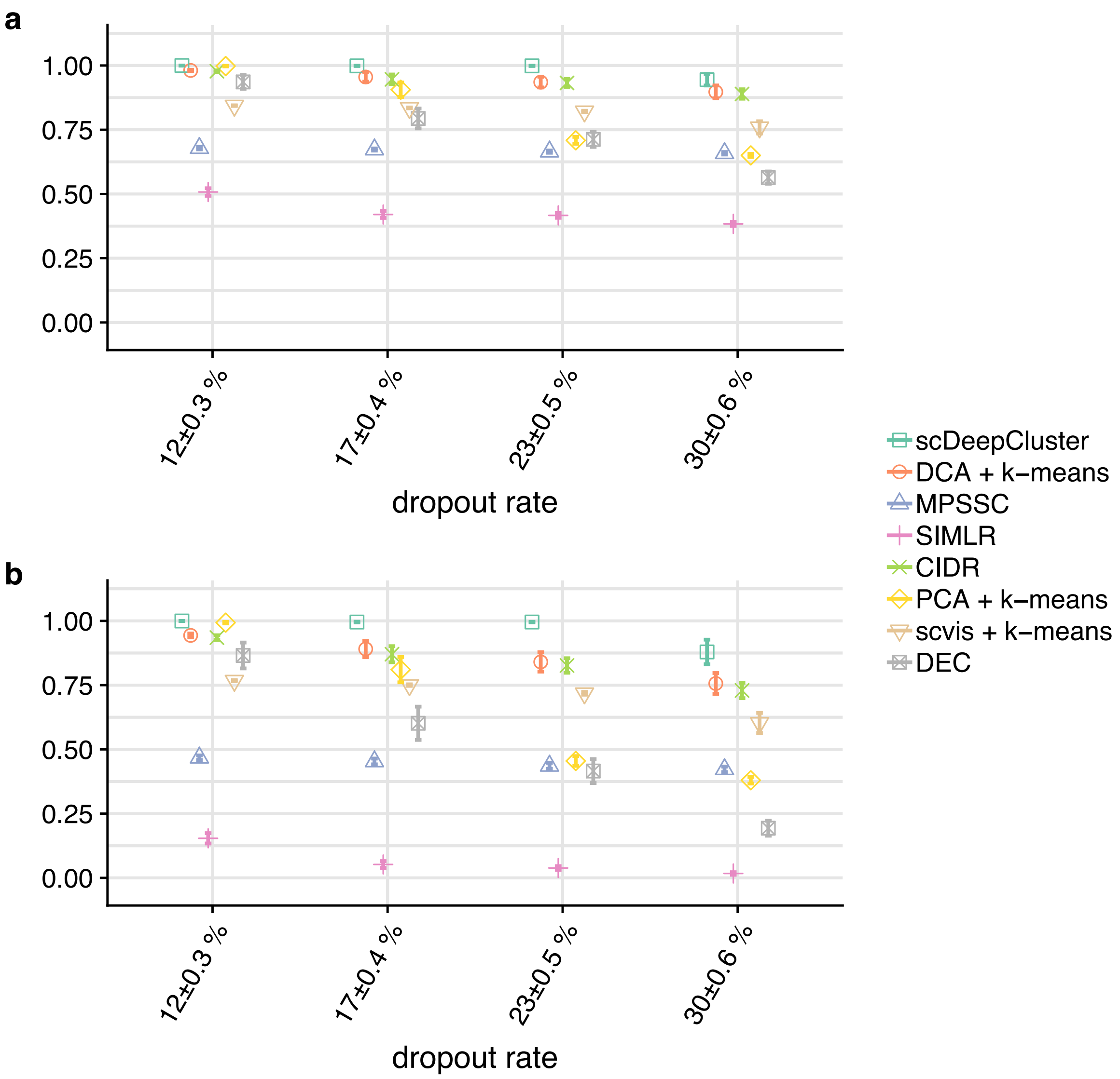


Figure 3.3 Clustering performance of scDeepCluster, DCA + k-means, MPSSC, SIMLR, CIDR, PCA + k-means, scvis + k-means and DEC on simulated data measured by **(a)** Clustering Accuracy (CA) and **(b)** Adjust Rand Index (ARI). The averaged values over 20 repeats with standard errors are shown for each simulation setting. The larger value means the more concordance between predicted labels and true labels.

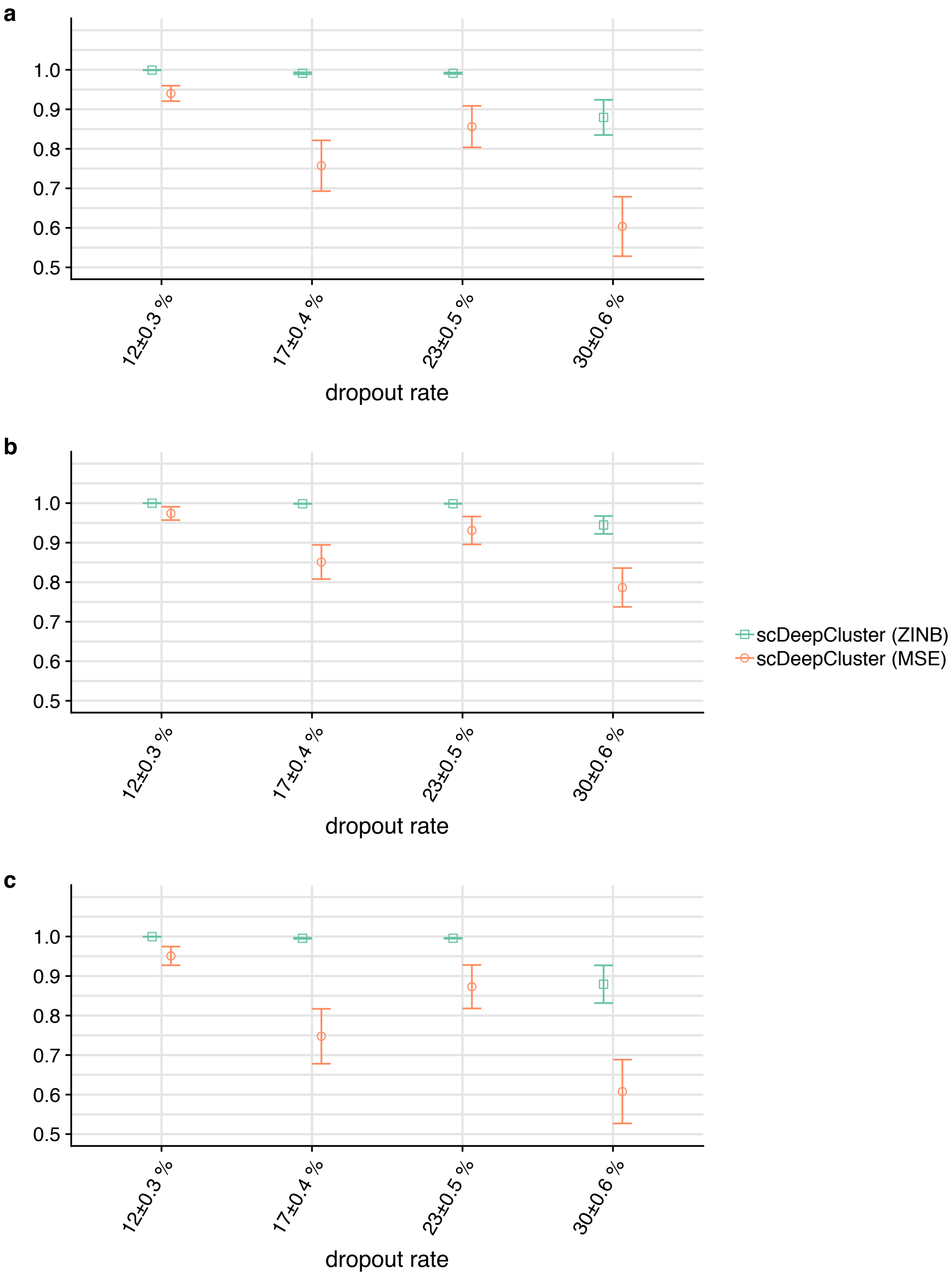


Figure 3.4 Clustering performance of scDeepCluster with ZINB loss and MSE loss on simulated data with various dropout rates measured by **(a)** NMI, **(b)** CA and **(c)** ARI. The averaged values over 20 repeats with standard errors are shown for each simulation setting. The larger value means more concordance between the predicted labels and the true labels.

We next evaluated performance under different clustering signal strengths. To do that, we varied fold change levels of gene expression between cell types/groups, which can be controlled by adjusting the variance parameter sigma in the log-normal distribution employed by Splatter for generating various differential expression intensities. The larger the variance parameter in the log-normal distribution, the larger distance between samples from different clusters, and the stronger the clustering signal. Therefore, fixing the dropout rate at 17% (default setting of Splatter), we evaluated the methods under various sigma values, and their performances are presented in **Figure 3.2b** and **Figure 3.5**. The t-SNE [20] plots of these simulated datasets confirm that the clustering signal strength increases with the sigma value (**Figure 3.6**). We can see that, again, scDeepCluster outperformed the other methods significantly under all different clustering signal strengths (*p-value* < 0.01, one-sided paired t-test). Not surprisingly, the clustering performance of different methods in general improved with the increasing of the signals. Of note, scDeepCluster improved dramatically from the weak signal (sigma = 0.2) and quickly reached nice clustering (NMI ≈ 0.86) when sigma increased to 0.225. The improvement of the MPSSC over the increasing sigma was steady but very small, indicating its inefficiency in exploiting the increased signal. The SIMLR failed (NMI close to 0) for all signals. The DCA, PCA, scvis and DEC also had improvement with the increasing sigma, but the DCA and scvis improved more quickly than the PCA and DEC. On these simulated datasets, we found that scDeepCluster yielded better clustering results after k-means initialization on the embedded spaces, which highlights the contribution of the deep learning clustering stage (**Figure 3.7**).

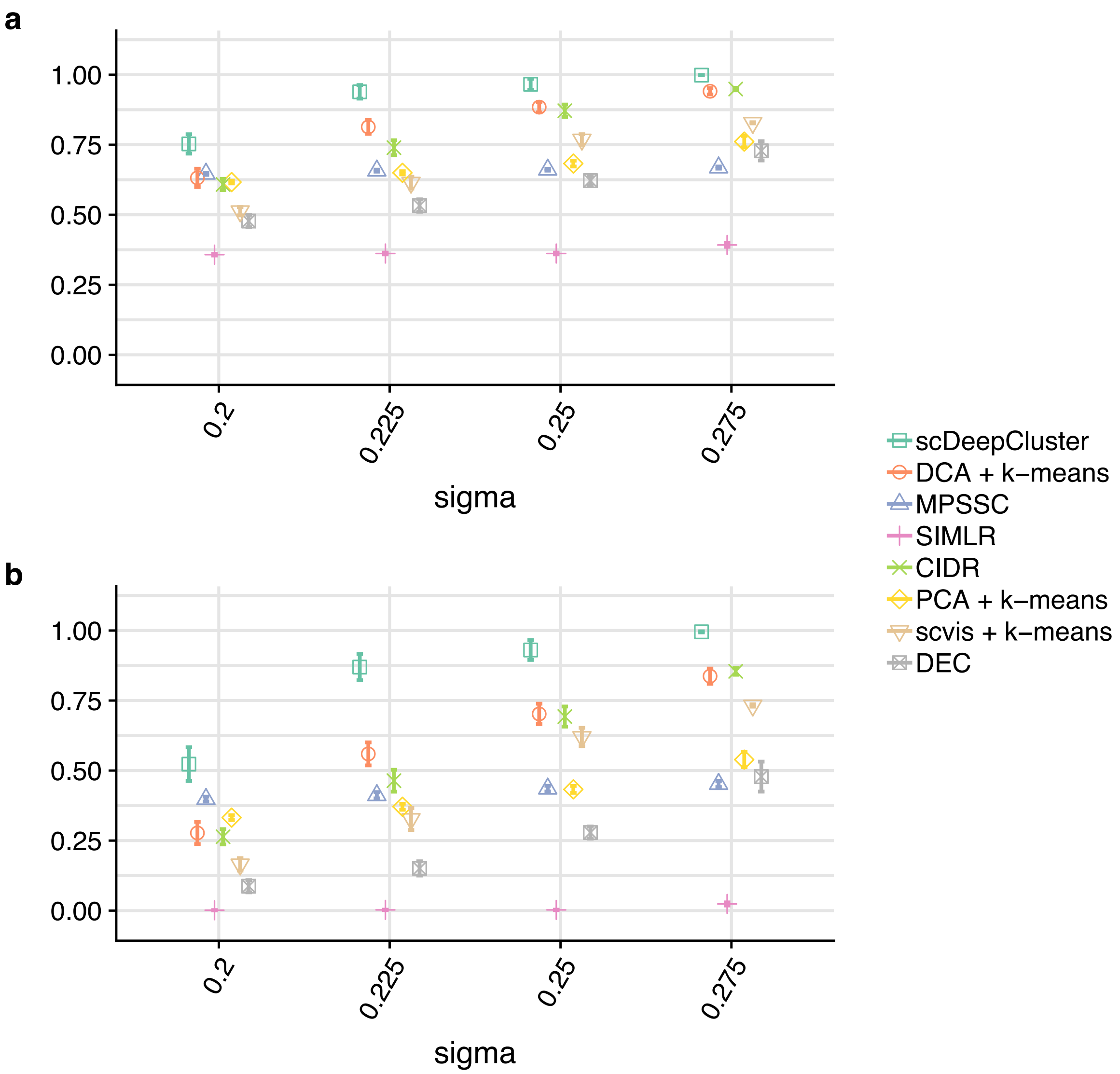


Figure 3.5 Clustering performance of scDeepCluster, DCA + k-means, MPSSC, SIMLR, CIDR, PCA + k-means, scvis + k-means and DEC on simulated data measured by **(a)** Clustering Accuracy (CA) and **(b)** Adjust Rand Index (ARI). The averaged dropout rates are 17% (*dropout.shape* = -1, *dropout.mid* = 0). The averaged values over 20 repeats with standard errors are shown for each simulation setting. The larger value means the more concordance between predicted labels and true labels.

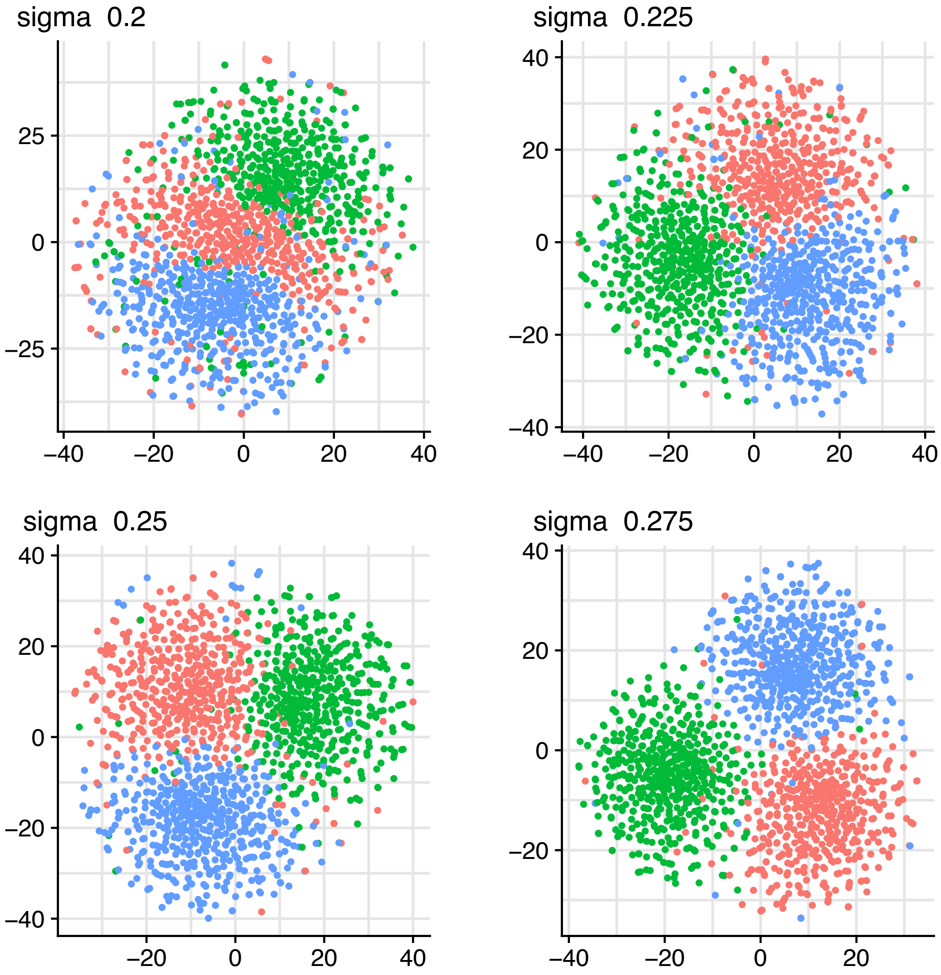


Figure 3.6 The t-SNE plots of simulated data with the different settings of sigma of the log-normal distribution (*de.facScale* parameter in Splatter). One example from each simulation setting is displayed. Distinct colors represent different true labels. The inputs for t-SNE plots are the library size normalized and log-transformed read count matrix.

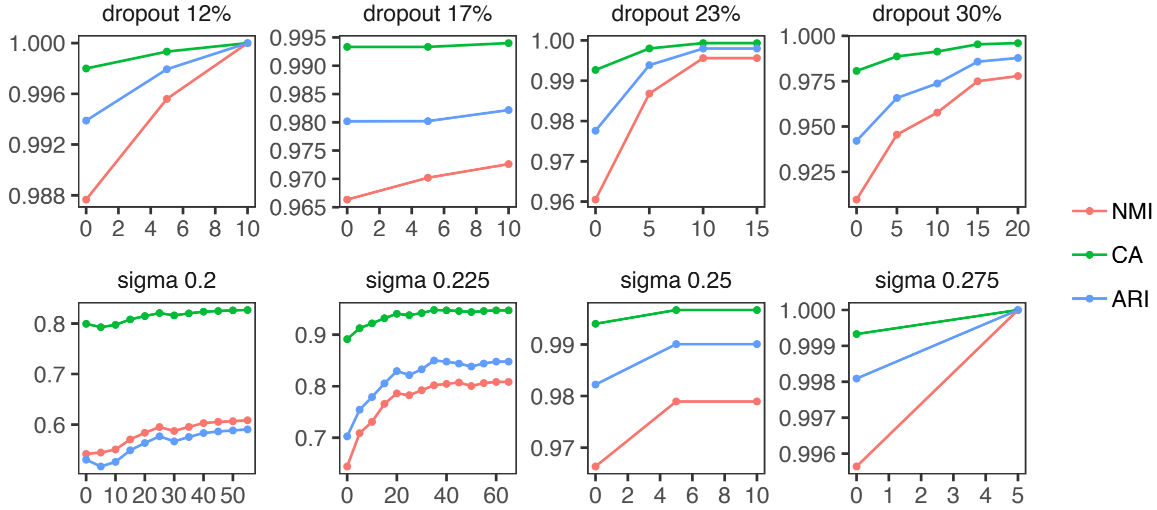


Figure 3.7 Clustering performance per batch of scDeepCluster after k-means initialization on the embedded spaces. One example from each simulation setting is displayed. The x-axes are the batches, and the y-axes are the values of NMI, CA and ARI. We observe the clustering performance improves after k-means initialization, which illustrates the contribution of the clustering loss.

Finally, we investigated the performance of scDeepCluster when the three groups have imbalanced sample sizes. Following [19], we generated different imbalance levels by varying minimum retention rate r_{min} when allocating the 1500 cells to the three groups (1500 cells by 2500 genes in each dataset, 1500 cells were assigned into three groups. For minimum retention rate r_{min}, data points of group 0 will be kept with relative proportion r_{min} and group 2 with relative proportion 1, with the group 1 with the relative proportion of the middle: $(r_{min} + 1)/2$. Then, the resulting the largest group will have $1/r_{min}$ times as large as the minimum one. Fixing the *dropout.shape* = -1, *dropout.mid* = 0, and *de.facScale* = 0.4 (default values recommended by Splatter), we varied the minimum retention rate from 0.1 to 0.9. As shown in **Figure 3.8**, scDeepCluster's performance is fairly robust. It retains perfect clustering concordance (NMI ≈ 1) until r_{min} decreases to

0.3, its performance remains decent (NMI > 0.85) even when the imbalance is extreme ($r_{min} = 0.1$).

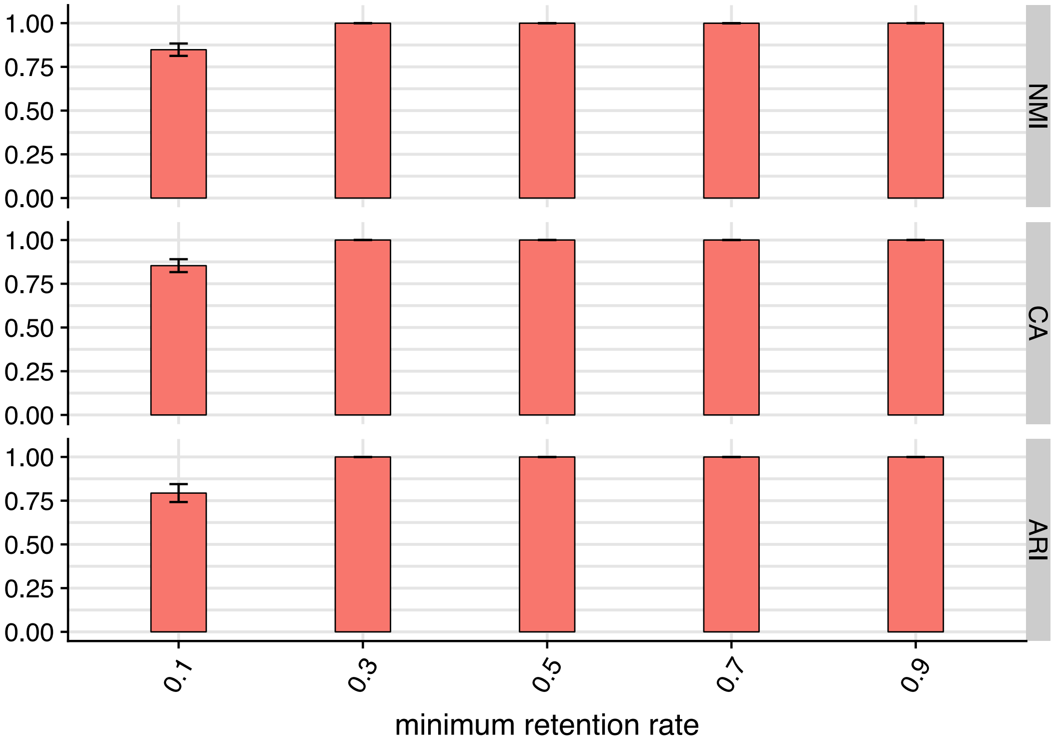


Figure 3.8 Clustering performance of scDeepCluster measured by NMI, CA and ARI on imbalanced simulated scRNA-seq data. The averaged values and standard errors on the 20 datasets of each simulation setting are shown. The larger value means more concordance between the predicted labels and the true labels. The minimum retention rate is a measurement of imbalance levels.

3.3.2 Application to real data

We apply scDeepCluster to four real scRNA-seq datasets to demonstrate the performances. The four datasets are generated from four representative sequencing platforms: PBMC 4k cells from 10X genomics platform (10X PBMC) [30], mouse embryonic stem cells from droplet barcoding platform (Mouse ES cells) [1], mouse bladder cells from the Microwell-seq platform (Mouse bladder cells) [31] and worm neuron cells from sci-RNA-seq platform

(Worm neuron cells) [60], as summarized in **Table 3.1**. The four datasets, respectively, have 4271, 2717, 2746 and 4186 cells/samples, with 16,449, 24,046, 19,079 and 11,955 genes after preprocessing, and form 8, 4, 16 and 10 groups/clusters. The detailed description of these datasets is provided in the "Real data" section of the method part. The spectral based clustering methods (MPSSC, SIMLR) were imposed with quadratic space complexity, and we failed to run them with even large memory (e.g., 256G). To make a comparison, we randomly sampled 2100 cells from each dataset. It is noted that the random sampling did not drop any groups in any dataset. The three metrics (NMI, accuracy, and ARI) of clustering performances are visualized in **Figure 3.9a**. We observe that the proposed deep learning clustering scDeepCluster outperformed all the other methods including MPSSC, SIMLR, CIDR and scvis in all four datasets. We visualized the progression of the embedded presentation of the four datasets using t-SNE [20] plots (**Figure 3.10**). The increased performance after the k-means initialization is also observed on the four real datasets.

Table 3.1
Summary of Four Real scRNA-seq Datasets*

Dataset	Sequencing Platform	Sample size / Cell Numbers	#Genes	#Groups
10X PBMC	10X	4271	16,449	8
Mouse ES cells	Droplet Barcoding	2717	24,046	4
Mouse bladder cells	Microwell-seq	2746	19,079	16
Worm neuron cells	sci-RNA-seq	4186	11,955	10

* We randomly sampled 2100 cells from each dataset to do the experiments

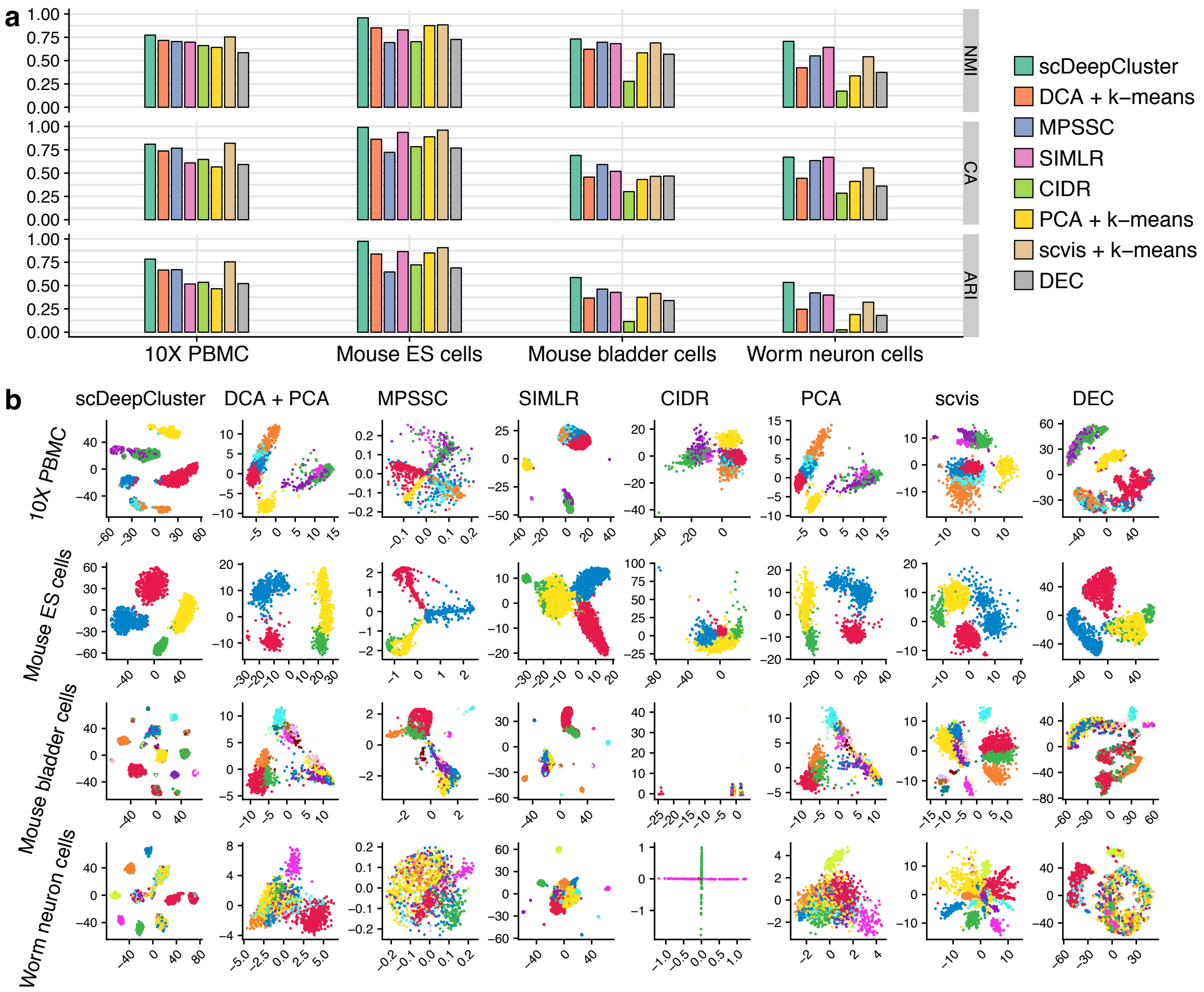


Figure 3.9 Benchmark results on four real scRNA-seq datasets with true labels. **(a)** Comparison of clustering performances of scDeepCluster, DCA + k-means, MPSSC, SIMLR, CIDR, PCA + k-means, scvis + k-means and DEC which are measured by Normalized Mutual Information (NMI), Clustering Accuracy (CA) and Adjusted Rand Index (ARI). The larger value means more concordance between the predicted labels and the true labels. **(b)** Comparison of 2D visualization of embedded representations. The axes are arbitrary units. Each point represents a cell. The distinct colors of the points represent the true labels. No method uses the true label information.

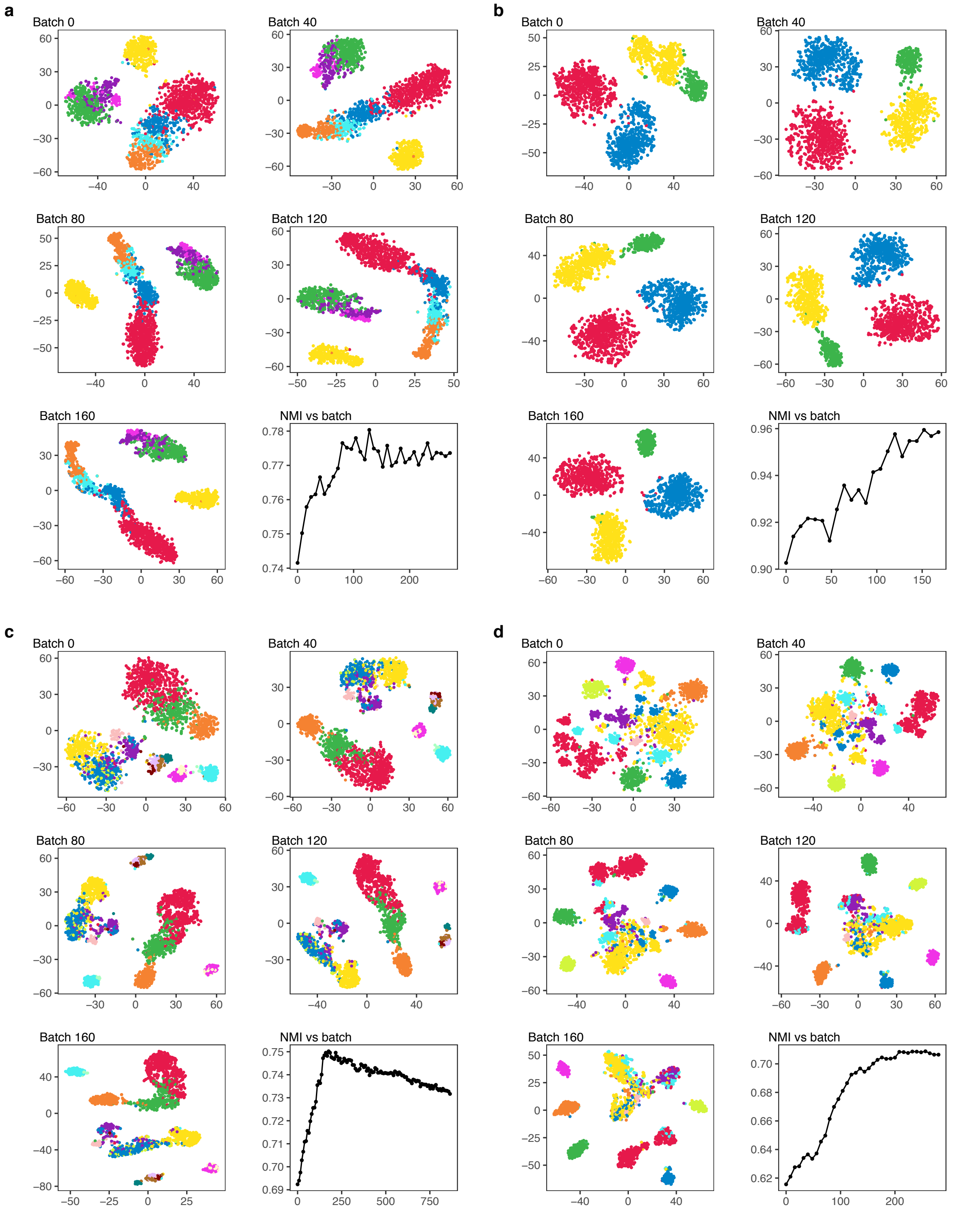


Figure 3.10 Visualization of latent representations during the clustering stage on the 2100 cells subsets of **(a)** 10X PBMC, **(b)** Mouse ES cells, **(c)** Mouse bladder cells and **(d)** Worm neuron cells. Different colors mark different groups of cells. The NMIs per batch are also shown.

The latent space in the proposed scDeepCluster model is an ideal low-dimensional embedded representation of the high dimensional input data. To illustrate the representation effectiveness of latent space, we apply t-SNE to visualize the final embedded points in the 2D space which were learned by the ZINB model-based autoencoder in the scDeepCluster model. The 2D space representations of other methods are also plotted. As shown in **Figure 3.9b**, scDeepCluster separates the mouse ES cells of four different leukemia inhibitory factor (LIF) withdrawal intervals (0 days, 2 days, 4 days and 7 days, highlighted in different colors) well, with only a few red samples mingled with the blue, and a few green ones with the yellow. In contrast, the green and the yellow samples were mixed together for the other competing methods. Similar observations were made for the other three datasets that the cells of the same type are separated well in the low-dimensional embedded representation of scDeepCluster with only some outliers, and much better than the competing methods. The results reported so far are done by the 2100 randomly sampled datasets. We summarized the effects of the random Gaussian in **Figure 3.11** and the choices of γ in **Figure 3.12** on these four datasets to illustrate the hyperparameters tuning. We also summarized the clustering results of scDeepCluster on the four full datasets (**Figure 3.13**). The performance of scDeepCluster is robust for the full and randomly sampled datasets. We observed little improvements for the full datasets which indicate more data help scDeepCluster learns better feature represetations.

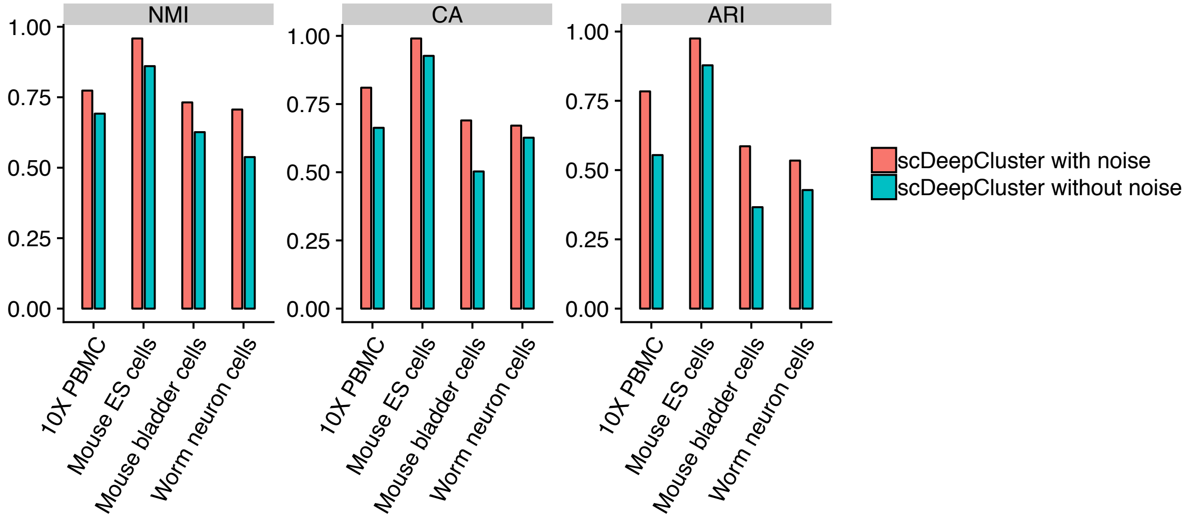


Figure 3.12 The denoising autoencoder helps to learn a more robust feature representation [10, 11]. Clustering performance of scDeepCluster with random Gaussian noise and without random Gaussian noise on the 2100 cells subsets of 10X PBMC, mouse ES cells, mouse bladder cells, and worm neuron cells is displayed (these datasets were described in the following subsection).

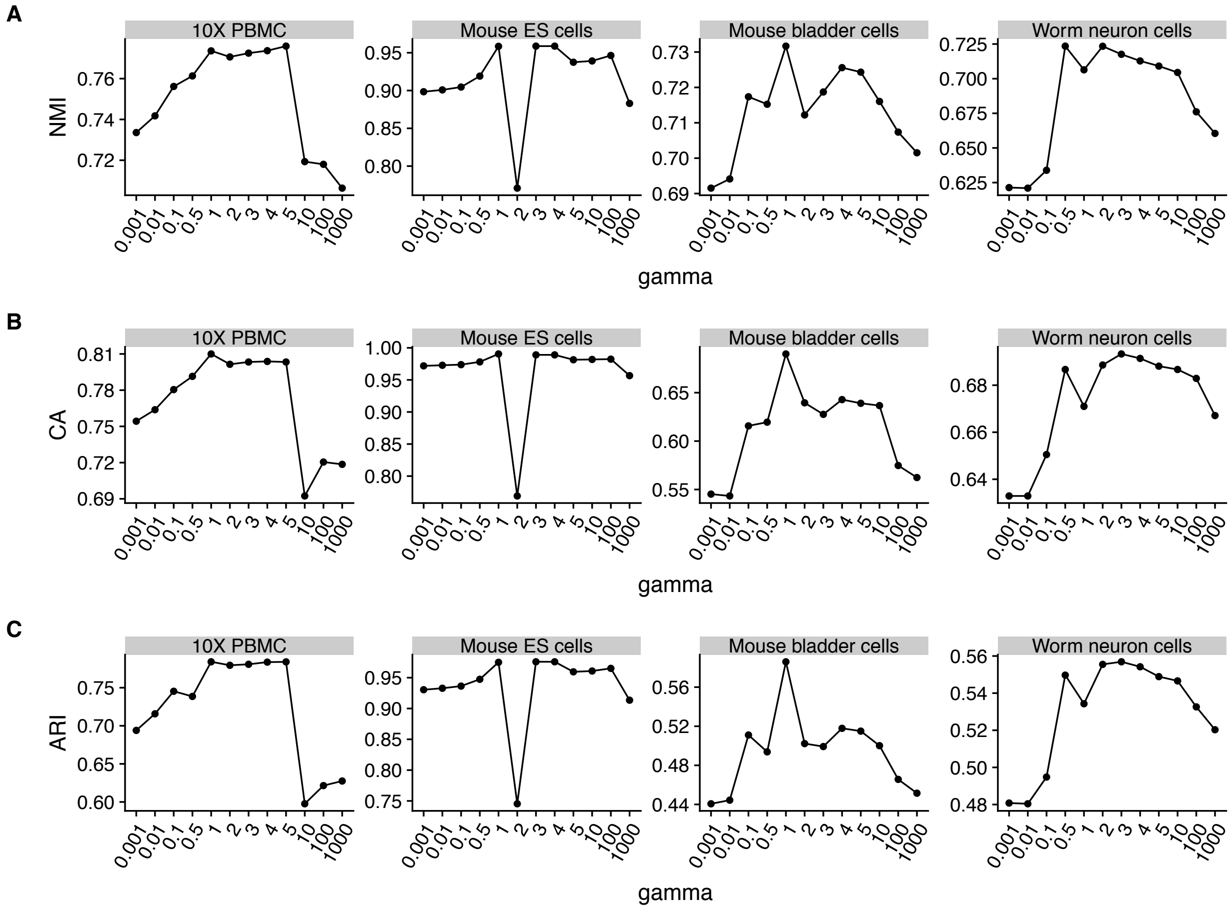


Figure 3.13 The effect of the clustering coefficient γ on the performance for the 2100 cells subsets of 10X PBMC, mouse ES cells, mouse bladder cells, and worm neuron cells.

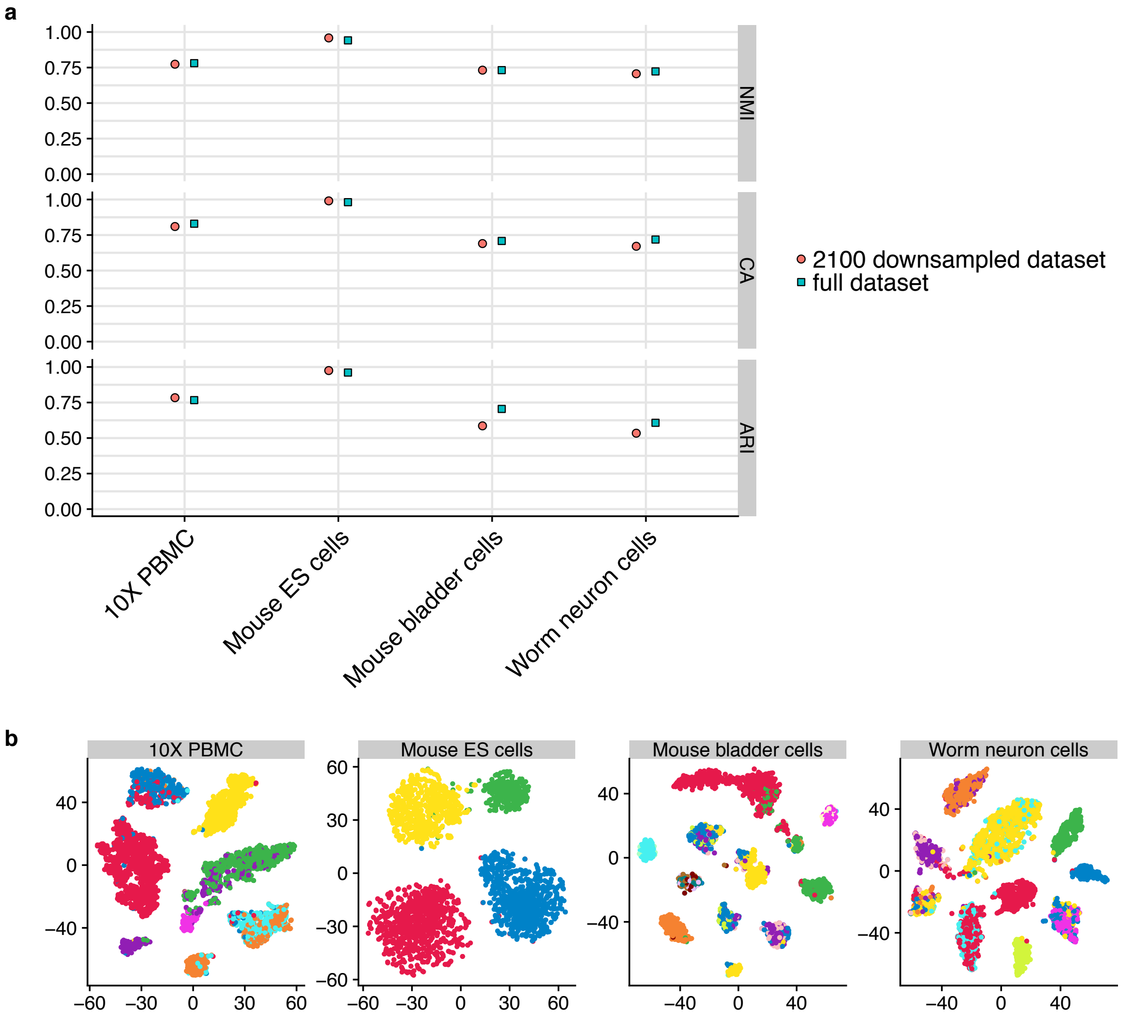


Figure 3.13 The clustering performance of scDeepCluster on the 2100 cells downsampled and the full real scRNA-seq datasets. **(a)** The clustering metrics (NMI, CA and ARI) of scDeepCluster on the 2100 cells down-sampled and the full datasets. **(b)** The t-SNE visualization of the final latent spaces of scDeepCluster on the full real scRNA-seq datasets.

It is interesting to see the performance of these methods on very large scRNA-seq datasets, such as tens of thousands of cells. To this end, we added for evaluation two additional real datasets with 27k and 68k cells, respectively (**Table 3.2**). SIMLR and MPSSC failed to run over these two large datasets. We observed similar superiority of scDeepCluster over the other competing methods (**Figure 3.14**).

Table 3.2 Summary of Two Large Real Datasets

Dataset	**Sequencing Platform**	**Sample size / Cell Numbers**	**#Genes**	**#Groups**
PBMC 68k	10X	68,579	20,387	10
Mouse retina cells	Drop-seq	27,499	13,166	19

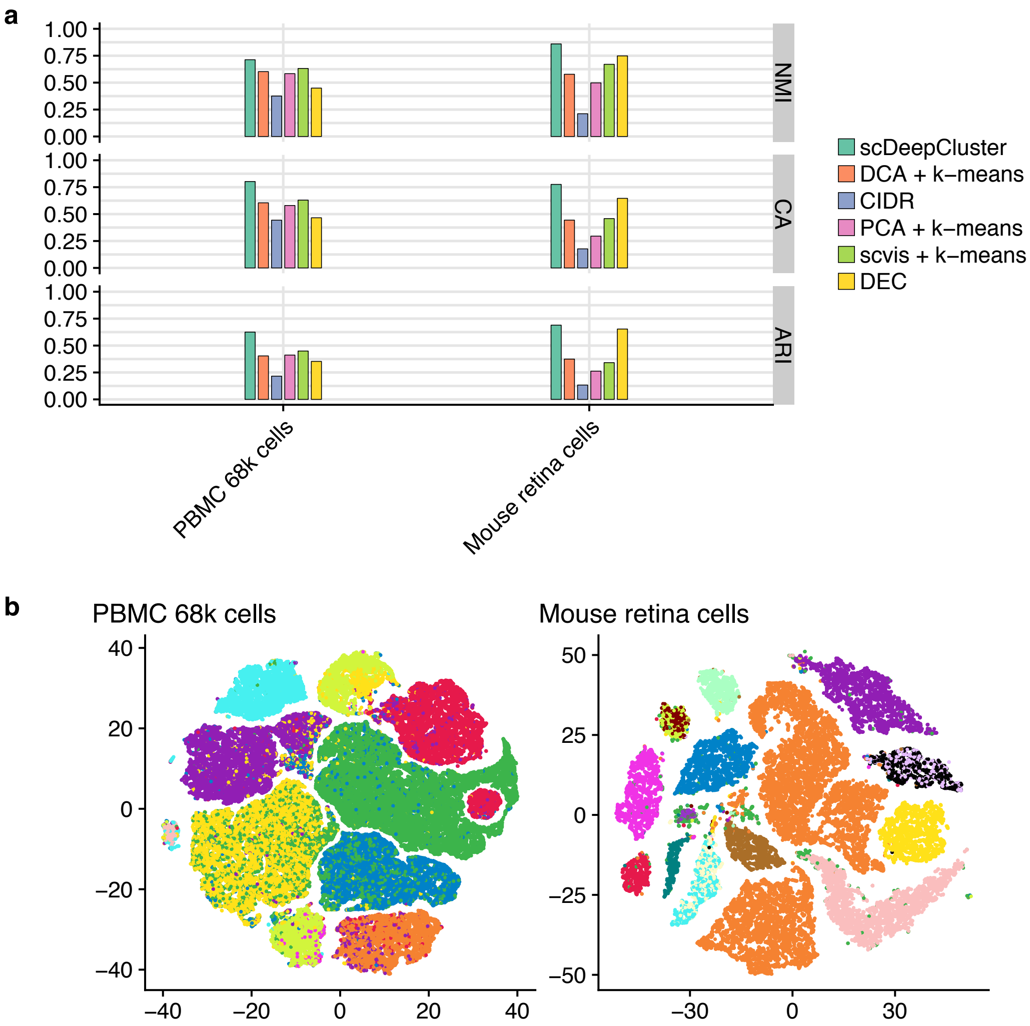


Figure 3.14 Clustering performance of scDeepCluster evaluated on two large real datasets. **(a)** Cluster results measured by NMI, CA and ARI (The performance for CIDR on PBMC 68k is the results on the down-sampled 20k cells). **(b)** Visualization of the final latent spaces of scDeepCluster on the PBMC 68k and mouse retina cells datasets. The distinct colors of the points represent the true labels.

3.3.3 Scalability

Practitioners are confronted with a larger and larger number of cells profiled in scRNA-seq experiments nowadays. It becomes essential for an analytic method to be capable of handling large datasets. To assess the scalability of scDeepCluster, we summarized its running time on datasets with various sample sizes. Particularly, we simulated a large dataset of 100k cells (10^5 cells) by 3000 genes, with 10 groups in these cells, and down-sampled this dataset from 5k to 100k cells (5k, 7.5k, 10k, 25k, 50k, 75k, 100k). We report the running time of pre-training and clustering stages of scDeepCluster on these down-sampled datasets in **Figure 3.15a**. We can see that, not like the quadratic running time complexity of spectral clustering, the running time of scDeepCluster scales linearly with the number of cells. Such computational efficiency makes scDeepCluster a very appealing tool for analysis of large scRNA-seq datasets. Furthermore, we found that the clustering performance of scDeepCluster is quite robust to varying sample sizes (**Figure 3.15b**).

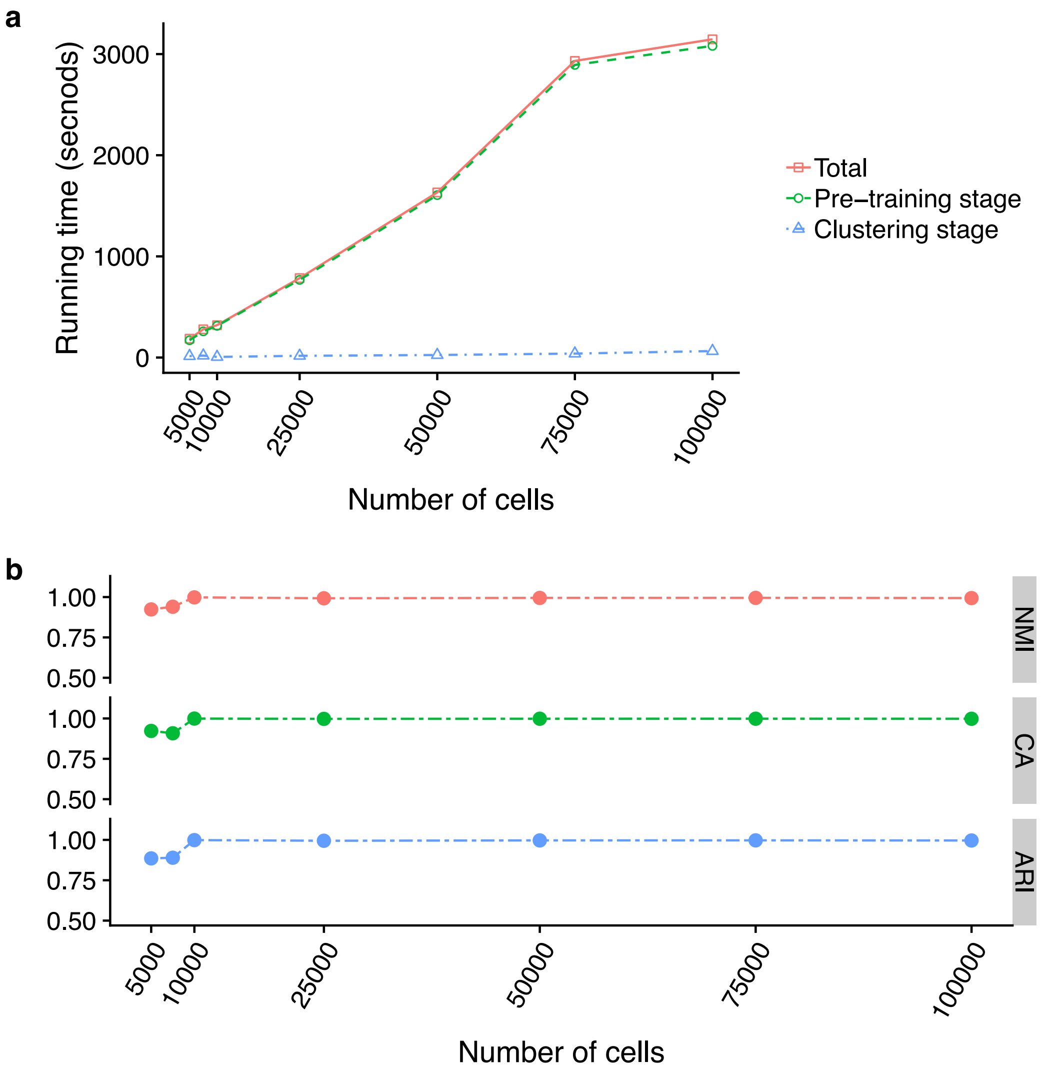


Figure 3.15 Applying scDeepCluster on various down-sampled simulated data. **(a)** The running time (seconds) of the pre-training stage, the clustering stage and the total of scDeepCluster on different sample size. **(b)** Clustering performance of scDeepCluster on different sample size measured by NMI, CA and ARI. The results are obtained on Nvidia Tesla P100 (16G).

3.4 Discussion

Following most clustering studies, we have assumed that *k*, the number of clusters, is given in the comparison between competing algorithms. In practice, this information is usually unknown. When there is a substantial mismatch between the k used and the true number of clusters, a model mismatch issue may result. A method for determining the optimal number of clusters is desired. For this purpose, following other clustering methods[19], we may employ an elbow-method strategy by introducing a metric, generalizability (*G*). Formally, *G* is defined as the ratio between training and validation clustering loss (we split the data to training and validation data):

$$G = \frac{L_{train}}{L_{validation}}$$

We modify the scDeepCluster model, so it has no reconstruction loss during clustering stage. As a result, *G* is the ratio between the clustering loss of training data and validation data. We first train scDeepCluster model (only clustering loss in clustering stage) on training data. Next, we calculate the clustering loss, predict the labels of validation data (by using the model trained on training data) and calculate *G*.

To illustrate how *G* works, we did a simulation experiment. We use Splatter to simulate 20 scRNA-seq datasets which have 5000 cells of 2000 genes in 5 groups. The *de.fracScale* = 0.3, and other parameters were set to be default (e.g. *dropout.shape* = -1, *dropout.mid* = 0). For each simulated data, we split training and validation by the ratio of 9:1. We pre-train 600 epochs for each training data (pre-train the reconstruction loss of denoising ZINB model-based autoencoder), then the pre-trained autoencoder weights are used to initialize the model in clustering stage with various settings of the number of clusters (*k* = 3 - 8 clusters).

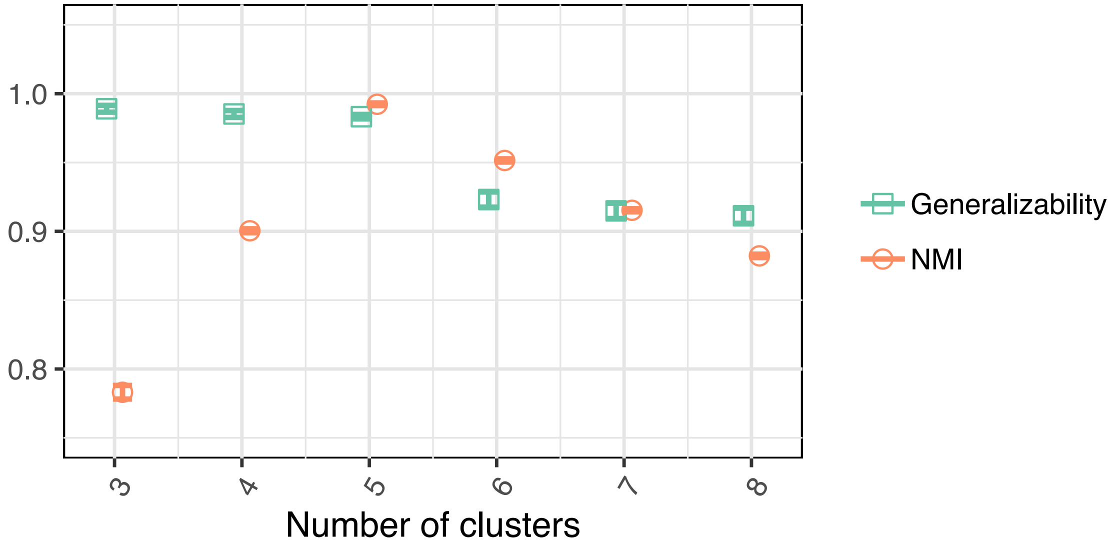


Figure 3.16 Estimate of the number of clusters on the 20 simulated datasets. Plot of Generalizability and NMI of validation data on the various number of clusters. The means and standard errors among 20 simulated data are plotted.

As reported in **Figure 3.16**, we observe a sharp drop in generalizability when the cluster number increases from 5 to 6, which suggests that 5 is the optimal number of clusters. We also observe that the NMI values for validation data are highest at 5. This result suggests that generalizability is a good metric to determine the number of clusters. We also summarized the results of *k* (number of clusters) estimation of the competing methods: CIDR and SIMLR (**Figure 3.17**). As we can see, CIDR estimates k correctly (k = 5) in most cases but makes mistakes in some cases, but SIMLR fails to estimate the correct k in all cases.

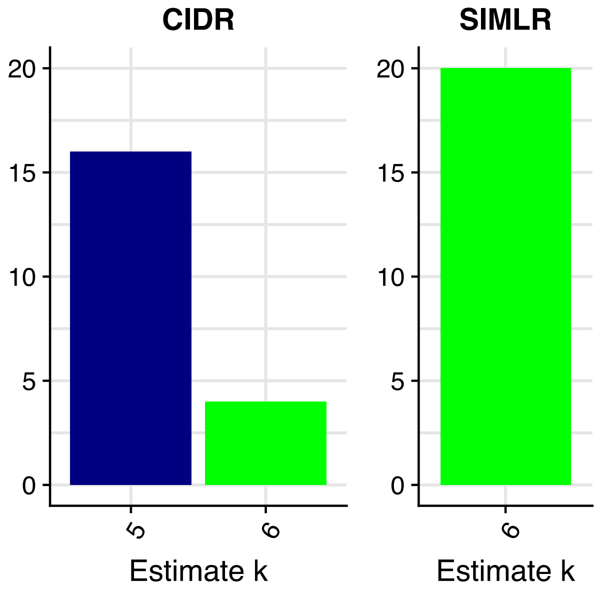


Figure 3.17 Estimate of the number of clusters (k) by CIDR and SIMLR on the 20 simulated datasets. The x-axis is the values of estimated k, and the y-axis is the counts of the estimations.

Generally, clustering methods can be divided into two subcategories: discriminative clustering algorithms (e.g., spectral clustering), and generative clustering algorithms (e.g., our method). It has been argued that training discriminative models can suffer from overfitting [62]. Our empirical experiments also showed that the spectral clustering model SIMLR mistakenly estimated the number of clusters on the simulated datasets.

In conclusion, we have proposed a method scDeepCluster based on deep learning techniques for clustering analysis of scRNA-seq data. scDeepCluster can learn a latent embedded representation that is optimized for clustering high dimensional input in a non-linear manner. In particular, we explicitly model scRNA-seq data generation using a parametric model appropriate for characterizing count data with excessive zeros. Both simulation studies and real data applications have shown that coupling deep learning with this parameterization can effectively capture the pervasive dropout events, the main challenge confronted in scRNA-seq data analysis. In contrast, previous state-of-art spectral clustering methods, MPSSC and SIMLR, rely on multiple Gaussian kernels, which are proved to be less effective in characterizing sparse count data. We have demonstrated the superior clustering performance of scDeepCluster on both on simulated and real datasets by comparison with several competing methods. We also illustrate the excellent scalability of scDeepCluster on large datasets. As an ever-growing number of large-scale scRNA-seq datasets become available, we expect more applications of our method.

CHAPTER 4

IMPROVE THE MODEL-BASED AUTOENCODER BY MAXIMUM MEAN DISCREPANCY REGULARIZATION

4.1 Motivation

The experiments presented in Chapter 3 have illustrated the model-based deep autoencoder can archive good performance in the analysis of the discrete count data. However, the zero-inflated negative binomial (ZINB) autoencoder has three parameters to estimate the reconstruction of one input sample. They are the mean, the dispersion, and the dropout. This high degree of freedom brings the concerns of over-fitting of the ZINB autoencoder. The high degree of freedom means the number of parameters is much larger than the number of input samples. For one input sample, the combinations of mean, dispersion and dropout might be infinite. In this part, we introduce a regularization to mitigate this issue of the high degree of freedom.

In the ZINB autoencoder, we assume the output of the mean layer represents the counts or expression levels before the dropout been introduced and optimize the ZINB likelihood to estimate the parameters. But, the relation between the mean layer and the count data is not explicitly defined. The combination of mean, dispersion and dropout for one input sample can be infinite. If we know the latent expression levels (expression levels without dropout) and the probabilities of dropout, we can generate the raw count matrix. The latent expression levels and dropout probabilities have been estimated by the ZINB model-based autoencoder, and dropout events can be modeled by the Bernoulli sampling process with the respecting of the estimated dropout probabilities. Therefore, we can establish an explicit model for generating the reconstruction data. The simple Bernoulli sampling process is not differentiable, and it is hard to be implemented in the deep neural networks. The recent advance of neural network research has brought the idea of the Gumbel-softmax, which is a differentiable categorical distribution that is used in the backpropagation [63]. A recent study illustrated the Gumbel-softmax can explicitly model the dropout events in scRNA-seq data [64]. Here, we use the Gumbel-softmax to design the regularization for our ZINB deep autoencoder.

By using the Gumbel-softmax, we can generate the count matrix that is randomly masked with the probabilities learned by the dropout layer in the ZINB model-based autoencoder. Due to the randomness, the generated matrices are different for each iteration. If the latent expression levels and dropout probabilities can be learned perfectly, the input raw count matrix can be considered to be generated by this distribution. Our goal is to optimize the divergence or distance between the raw count distribution and the ZINB distribution learned by the model-based deep autoencoder. The Maximum Mean Discrepancy (MMD) is the largest difference in expectations over functions in the unit ball of a reproducing kernel Hilbert space (RKHS) [65]. The MMD has been used as a statistic to determine if two samples are drawn from different distributions. Recent researches illustrated the advantage of MMD as a regularization to guide deep neural network to learn the target distribution[12, 66], such as MMD autoencoder and MMD generative adversarial nets. MMD has also been used to correct the batch effects in the high-throughput genomic data [67]. In this chapter, we introduce to use the MMD as a regularization to help our ZINB model-based autoencoder to learn the data generating distribution better.

4.2 Methods

4.2.1 The Gumbel-softmax as a zero-inflated sampling layer

An additional zero-inflated (ZI) layer was appended to the mean and dropout layer of decoder network. The ZI layer generate a matrix of stochastic 0, 1 values with respecting the dropout probabilities (0 means dropout, 1 means not). The back-propagation cannot deal with stochastic units. To sample the stochastic Bernoulli values, we use the Gumbel-softmax technic [63]. Suppose p is the probability for dropout, and $q = 1 - p$ is the probability for non-dropout. The sample s from Gumbel-softmax distribution was obtained by

$$s = \frac{\exp\left(\frac{\log(p) + g_0}{\tau}\right)}{\exp\left(\frac{\log(p) + g_0}{\tau}\right) + \exp\left(\frac{\log(q) + g_1}{\tau}\right)}$$

where g_0, g_1were sampled from a Gumbel (0,1) distribution:

$$g_0, g_1 \sim -\log(-\log(u)), u \sim (0,1)$$

As the hyper-parameter $\tau \rightarrow 0$, the generated samples s should be identical to the samples from the Bernoulli distribution. In practice, the too small τ will suffer the problem of gradient vanish and make the optimization algorithm fail. To fix it, we use an annealing strategy that decreases the τ per 100 epochs, from 1 to 0.5. After obtaining the matrix of stochastic Bernoulli 0,1 matrix S, we can easily write the generated reconstruction matrix by the using the output of the mean layer M

$$\text{Zero inflated count matrix} = M \cdot S$$

The dot means element-wise production.

4.2.2 The Maximum Mean Discrepancy

The Maximum Mean Discrepancy (MMD) is the largest difference in expectations over functions in the unit ball of a reproducing kernel Hilbert space (RKHS). MMD can be considered as the f-divergence between two distribution [65]. Given x and y which are sampled from two distributions X and Y, and $\mathcal{F}$ to be a class of functions $f\colon \mathrm{X} \rightarrow \mathbb{R}$, the MMD can be formally defined as

$$MMD[\mathcal{F}, X, Y] := \sup_{f \in \mathcal{F}} \left(\boldsymbol{E}_x[f(x)] - \boldsymbol{E}_y[f(y)]\right)$$

This equation usually doesn't has a closed form and hard to calculate. Fortunately, MMD has an unbiased U-statistic estimator which can be used in conjunction with stochastic gradient descent (SGD) methods. The unbiased estimator of MMD can be written as

$$MMD_u^2[\mathcal{F}, X, Y]$$

$$= \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k(y_i, y_j)$$

$$- \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k(x_i, y)$$

where $k()$ is the kernel. The kernel can be Gaussian kernel or IMQ kernel. We use IMQ kernel with multiple bandwidths (from 1e-3 to 1e3). We want to estimate the MMD between estimated zero-inflated mean matrix and the raw count matrix. The MMD of discrete counts might not easy to be calculated, we take log transform of them first. In our context, the MMD loss for ZINB model-based autoencoder can be written as

$$L_{MMD} = MMD(raw\ count\ matrix, reconstructed\ count\ matrix)$$

4.2.3 The ZINB model-based autoencoder with MMD regularization

The proposed MMD regularization can be easily plugged into the ZINB model-based autoencoder. The loss will be

$$L = L_{ZINB} + \lambda L_{MMD}$$

Figure 4.1 shows the architecture of the regularized ZINB model-based autoencoder.

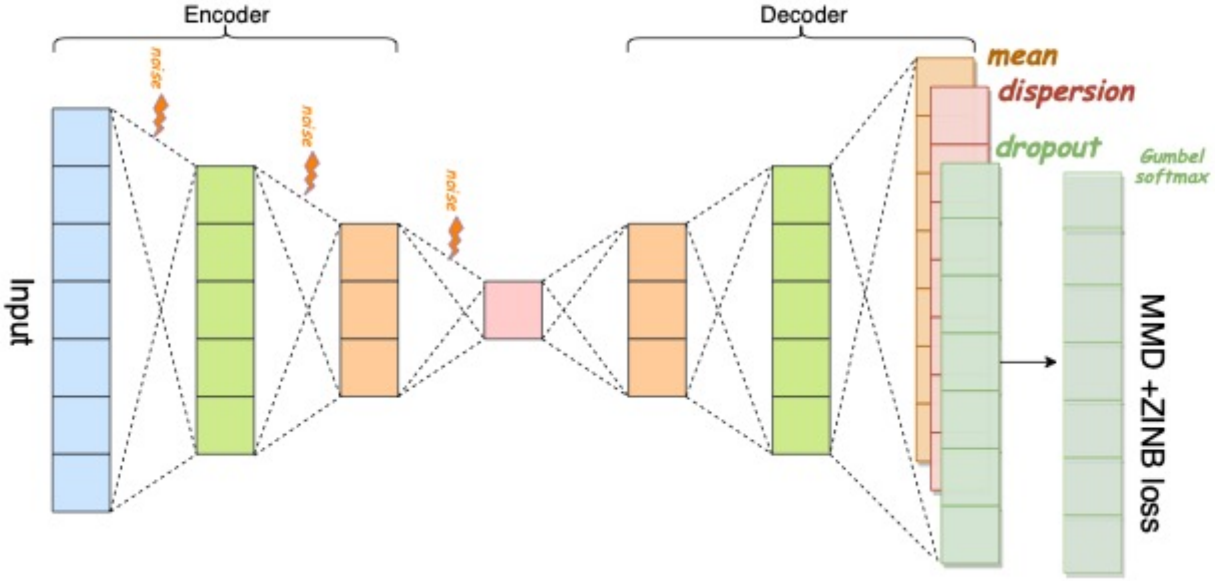


Figure 4.1 The network architecture of ZINB model-based autoencoder with MMD regularization.

The same architecture can be also used for clustering analysis, just by adding the clustering loss:

$$L = L_{ZINB} + \gamma L_c + \lambda L_{MMD}$$

We set λ to be 0.1 here.

4.2.4 Implementation

The ZINB model-based autoencoder with MMD regularization is implemented in Python 3 using Keras (https://github.com/keras-team/keras, accessed on April 28, 2019) with TensorFlow [50] backend. The autoencoder structure is set to have one hidden layer which has the size of 64, and the bottleneck layer's size is 32. The encoder, bottleneck and decoder layers all use the Relu activation and batch normalization technic[68]. We choose the IMQ kernel for MMD, which can be written as

$$k(x, y) = \frac{C}{C + \|x - y\|^2}$$

C is the bandwidth. To reduce the sensitivity on choice of the bandwidth, we use the mixture values $C \in [0.001, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5. 10. 20, 50, 100, 200, 500, 1000]$. The IMQ kernel has fatter tails than the classic radial basis function kernel, which makes the early training easier.

4.3 Results

We use Splatter to generate simulated read count data with various dropout rates (varying the *dropout.mid* from 3, 4, 5, 6). The other parameters are set to be default. For each setting, we generate 20 datasets, for each have 1500 cells of 2500 genes, equally from three groups. To evaluate the performance of imputation of dropout counts, we define the imputation error. For the counts that were missed, the imputation error is

$$\text{Imputation error} = \frac{\|X_{estimate} - X_{true}\|}{\|X_{true}\|_0}$$

The $\|X_{true}\|_0$ means the l0 norm of the true counts (true counts here are the latent true values of the counts need to be imputed, or the dropped counts). The estimated and true counts are log transferred. We compare the proposed MMD regularization (ZINB AE + MMD) with multiple methods: the vanilla ZINB model-based autoencoder (ZINB AE), the deep count autoencoder (DCA, default parameters and implementation suggested by authors), scImpute[8]. scImpute is the state-of-arts statistical model to impute the scRNA-seq data. For scImpute, we set the number of subpopulations to 3 and leave other parameters to be the default values. The hidden layer of ZINB model-based autoencoder is 64, and the bottleneck layer is 32.

The imputation results are summarized in **Figure 4.2**. As we can see, with the increasing of dropout rates, the imputation is more difficult, which is expected. Both our implemented ZINB model-based autoencoder and DCA has increasing imputation errors with the increasing of dropout rates, but the MMD regularization can mitigate this issue. The imputation errors of scImpute increase quickly with the dropout rate. When the dropout rate is extreme, the imputation result of scImpute is not much better than the raw data.

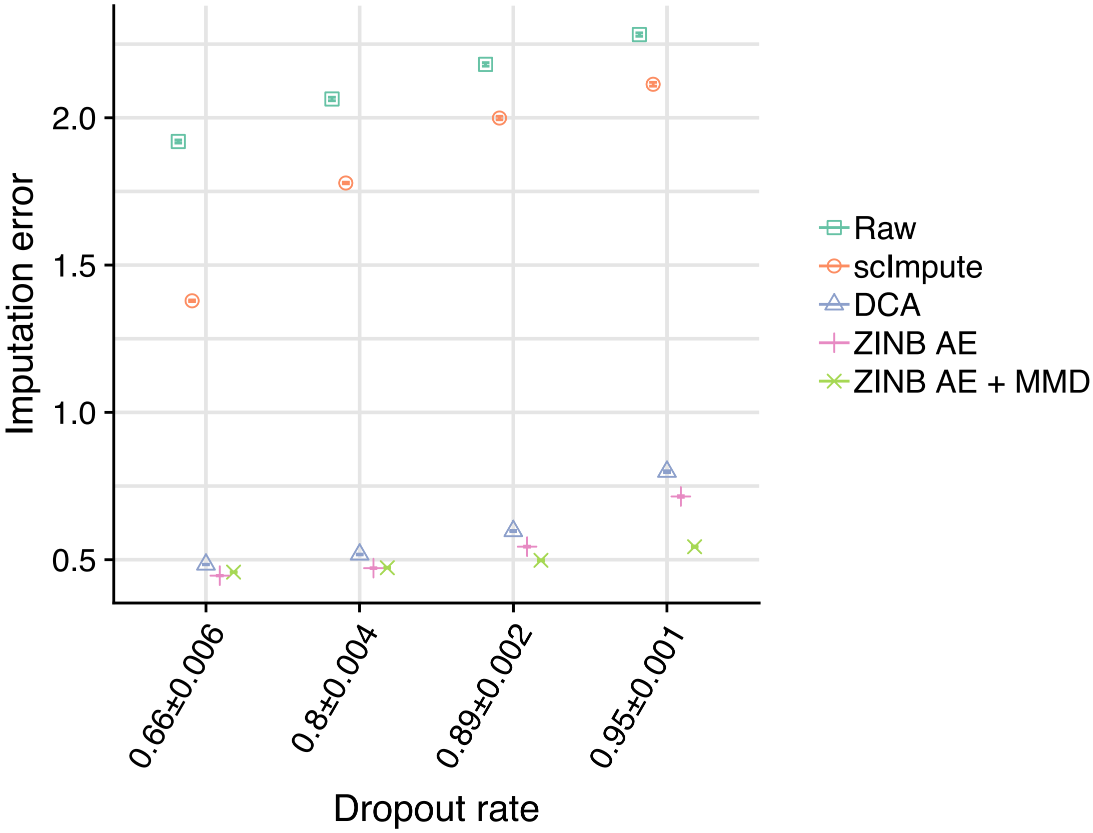


Figure 4.2 The performance of imputation of different under simulated data with different dropout rates. Imputation errors are based on log scaled.

4.4 Future work

Due to the high degree of freedom, it may suffer the issue of over-fitting for the ZINB model-based autoencoder. To mitigate the issue of a large number of parameters, we introduce an MMD regularization that helps the ZINB model-based autoencoder to better estimate the latent expression levels masked by dropout events. We evaluated our proposed method on the Splatter simulated data and achieved better results than the version of ZINB model-based autoencoder without regularization. Next, we will evaluate our method on more real scRNA-seq datasets.

Currently, the ZI layer to generate the randomly masked matrix only uses the mean and dropout parameters. We didn't use the information about dispersion. However, the sampling from negative binomial distribution has the same issue as the sampling from Bernoulli distribution. They are not differentiable. We use the differentiable Gumbel-sofmax to mimic Bernoulli sampling here. Therefore, the next research question is can we design a differentiable method to mimic the negative binomial sampling?

Basically, the scRNA-seq data can be modeled by a distribution. The distribution should have this form:

$$\text{scRNA-seq count matrix} \sim \Delta(X), X \sim \text{Some discrete distribution}$$

We use Δ to represent dropout events. The imputation is a process essentially learns the distribution of scRNA-seq data and tries to estimate the latent expression levels before dropout events. In other words, imputation leans the latent data distribution before dropout. The dropout events haven explicitly modeled by the ZI layer in our proposed model. The performance of learning the latent data distribution based on the metric of measuring the divergence between two distributions. The vanilla ZINB model-based autoencoder uses the maximizing the likelihood. m\Maximizing likelihood can be considered as optimizing the KL-divergence between the raw data distribution and the learned ZINB distribution. However, KL-divergence is point-wise, so we proposed to use the metric that estimated globally – MMD (global means MMD is estimated by all input samples and all reconstruction samples, not point-wise). Our simulation results illustrate MMD indeed improves the distribution learning ability of ZINB model-based autoencoder. The recent studies illustrate the power of adversarial learning such as Generative Adversarial Nets (GANs) [69]. The following studies prove the adversarial learning essentially optimize the f-divergence between the data distribution and the samples generated by neural networks, such as Wasserstein GANs [70]. Wasserstein distance has been successfully used in many deep neural network studies. In the following, we plan to study if the adversarial learning strategy can help the imputation of scRNA-seq data.

Since the dropout events are the major challenge of analysis of scRNA-seq data, the imputation analysis can recover the missed values and help the down-stream analysis to get better results. These down-stream analyses include cell type clustering (like scDeepCluster), differential expression analysis, cell lineage trajectory (the temporal ordering of cell differentiation). We plan to exam the properties of our proposed regularization method for these down-stream analyses.

APPENDIX

NETWORK STRUCTURE OF SCDEEPCLUSTER

In order to make the ZINB model-based autoencoder more robust to learn the latent representations, we use the denoising autoencoder technic. In order to get good clustering result, in scDeepCluster model, we didn’t add Gaussian noise for clustering. We build a clean encoder for clustering layer. The detailed structure is shown here.

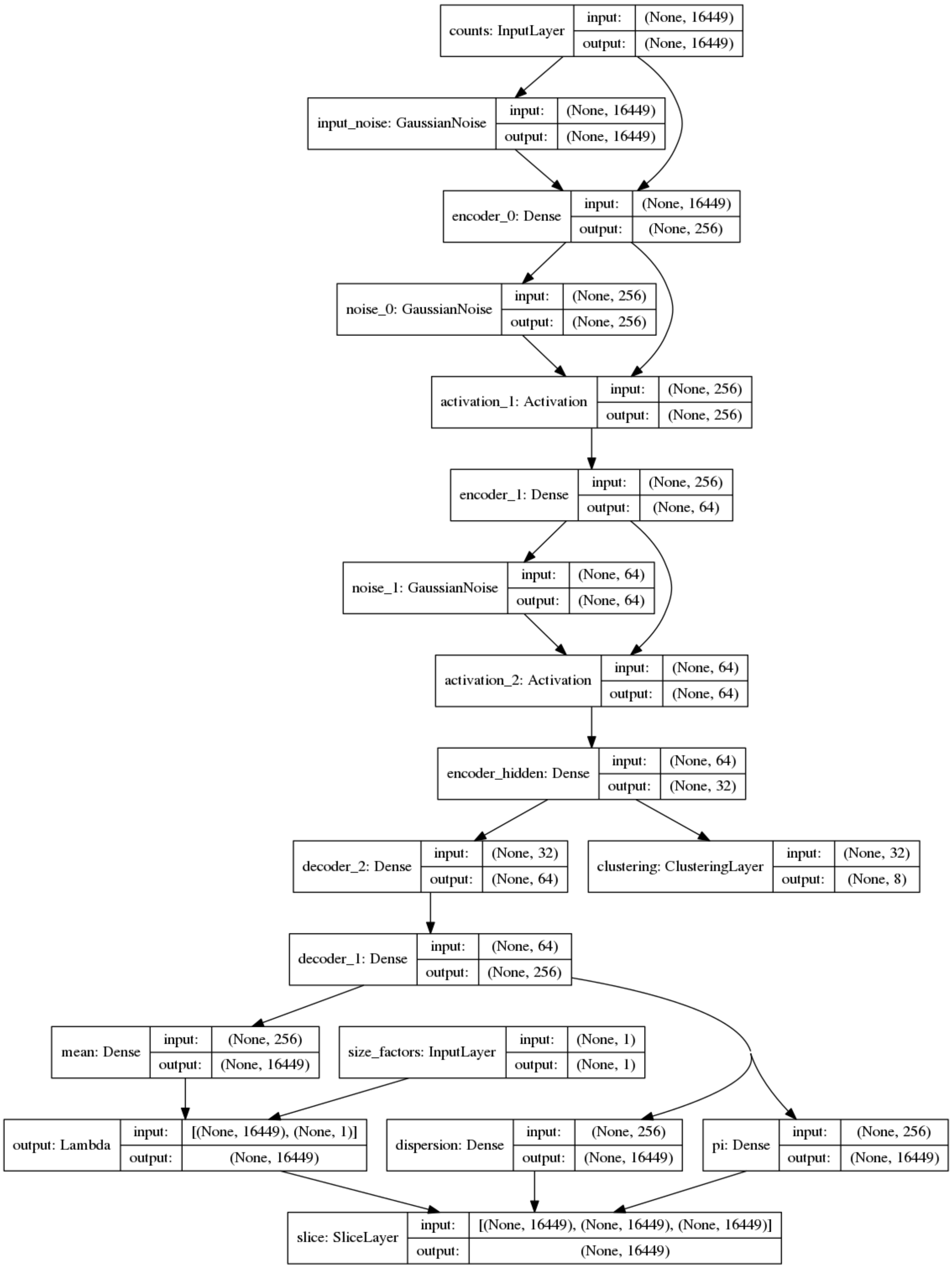

counts: InputLayer | input: (None, 16449) | output: (None, 16449)
input_noise: GaussianNoise | input: (None, 16449) | output: (None, 16449)
encoder_0: Dense | input: (None, 16449) | output: (None, 256)
noise_0: GaussianNoise | input: (None, 256) | output: (None, 256)
activation_1: Activation | input: (None, 256) | output: (None, 256)
encoder_1: Dense | input: (None, 256) | output: (None, 64)
noise_1: GaussianNoise | input: (None, 64) | output: (None, 64)
activation_2: Activation | input: (None, 64) | output: (None, 64)
encoder_hidden: Dense | input: (None, 64) | output: (None, 32)
decoder_2: Dense | input: (None, 32) | output: (None, 64)
clustering: ClusteringLayer | input: (None, 32) | output: (None, 8)
decoder_1: Dense | input: (None, 64) | output: (None, 256)
mean: Dense | input: (None, 256) | output: (None, 16449)
size_factors: InputLayer | input: (None, 1) | output: (None, 1)
output: Lambda | input: [(None, 16449), (None, 1)] | output: (None, 16449)
dispersion: Dense | input: (None, 256) | output: (None, 16449)
pi: Dense | input: (None, 256) | output: (None, 16449)
slice: SliceLayer | input: [(None, 16449), (None, 16449), (None, 16449)] | output: (None, 16449)

Figure A.1 The detailed network structure of scDeepCluster on 10X PMBC data

REFERENCES

1 Klein, A.M., Mazutis, L., Akartuna, I., Tallapragada, N., Veres, A., Li, V., Peshkin, L., Weitz, D.A., and Kirschner, M.W.: ‘Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells’, Cell, 2015, 161, (5), pp. 1187-1201

2 Tian, T., Wan, J., Song, Q., and Wei, Z.: ‘Clustering single-cell RNA-seq data with a model-based deep learning approach’, Nature Machine Intelligence, 2019, 1, (4), pp. 191-198

3 Kiselev, V.Y., Andrews, T.S., and Hemberg, M.: ‘Challenges in unsupervised clustering of single-cell RNA-seq data’, Nature Reviews Genetics, 2019

4 Butler, A., Hoffman, P., Smibert, P., Papalexi, E., and Satija, R.: ‘Integrating single-cell transcriptomic data across different conditions, technologies, and species’, Nature Biotechnology, 2018, 36, (5), pp. 411-420

5 Wang, B., Zhu, J., Pierson, E., Ramazzotti, D., and Batzoglou, S.: ‘Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning’, Nature Methods, 2017, 14, (4), pp. 414-416

6 Park, S., Zhao, H., and Birol, I.: ‘Spectral clustering based on learning similarity matrix’, Bioinformatics, 2018 12(34), pp. 2069-2076

7 Lin, P., Troup, M., and Ho, J.W.: ‘CIDR: Ultrafast and accurate clustering through imputation for single-cell RNA-seq data’, Genome Biology, 2017, 18, (1), pp. 59

8 Li, W.V., and Li, J.J.: ‘An accurate and robust imputation method scImpute for single-cell RNA-seq data’, Nature Communications, 2018, 9, (1), pp. 997

9 Goodfellow, I., Bengio, Y., and Courville, A.: ‘Deep Learning’ (The MIT Press, 2016. 2016)

10 Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A.: ‘Extracting and composing robust features with denoising autoencoders’. Proceedings of the 25th international conference on Machine Learning, Helsinki, Finland 2008 pp. 1096-1103

11 Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A.: ‘Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion’, J Mach Learn Res, 2010, 11, (Dec), pp. 3371-3408

12 Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B.: ‘Wasserstein Auto-Encoders’, 2017, arXiv preprint arXiv:1711.01558.

13 Tomczak, J., and Welling, M.: 'VAE with a VampPrior'. Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research2018 pp. 1214-1223

14 Eraslan, G., Simon, L.M., Mircea, M., Mueller, N.S., and Theis, F.J.: 'Single-cell RNA-seq denoising using a deep count autoencoder', Nature Communications, 2019, 10, (1), pp. 390

15 McCarthy, D.J., Chen, Y., and Smyth, G.K.: 'Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation', Nucleic Acids Research, 2012, 40, pp. 4288-4297

16 Love, M.I., Huber, W., and Anders, S.: 'Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2', Genome Biology, 2014, 15, (12), pp. 550

17 Lopez, R., Regier, J., Cole, M.B., Jordan, M.I., and Yosef, N.: 'A deep generative model for gene expression profiles from single-cell RNA sequencing', Nature Methods 2018, 15(12): 1053-1058.

18 Ding, J., Condon, A., and Shah, S.P.: 'Interpretable dimensionality reduction of single cell transcriptome data with deep generative models', Nature Communications, 2018, 9, (1), pp. 2002

19 Xie, J., Girshick, R., and Farhadi, A.: 'Unsupervised Deep Embedding for Clustering Analysis'. Proceedings of Machine Learning Research, New York, NY, USA 2016, 48: 478-487.

20 van der Maaten, L., and Hinton, G.: 'Visualizing Data using t-SNE', J Mach Learn Res, 2008, 9, pp. 2579-2605

21 Maaten, L.: 'Learning a Parametric Embedding by Preserving Local Structure'. Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research 2009, pp. 5: 384-391

22 Nigam, K., and Ghani, R.: 'Analyzing the effectiveness and applicability of co-training'. Proceedings of the ninth international conference on Information and knowledge management, McLean, Virginia, USA 2000 pp. 86-93

23 Guo, X., Gao, L., Liu, X., and Yin, J.: 'Improved Deep Embedded Clustering with Local Structure Preservation'. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia 2017 pp. 1753-1759

24 Shapiro, E., Biezuner, T., and Linnarsson, S.: 'Single-cell sequencing-based technologies will revolutionize whole-organism science', Nature Reviews Genetics, 2013, 14, (9), pp. 618-630

25 Kolodziejczyk, A.A., Kim, J.K., Svensson, V., Marioni, J.C., and Teichmann, S.A.: 'The technology and biology of single-cell RNA sequencing', Molecular Cell, 2015, 58, (4), pp. 610-620

26 MacQueen, J.: 'Some methods for classification and analysis of multivariate observations', in Editor: 'Book Some methods for classification and analysis of multivariate observations' (University of California Press, 1967, edn.), pp. 281-297

27 Bishop, C.: 'Pattern Recognition and Machine Learning' (Springer-Verlag New York, 2006)

28 von Luxburg, U.: 'A tutorial on spectral clustering', Statistics and Computing, 2007, 17, (4), pp. 395-416

29 Macosko, E.Z., Basu, A., Satija, R., Nemesh, J., Shekhar, K., Goldman, M., Tirosh, I., Bialas, A.R., Kamitaki, N., Martersteck, E.M., Trombetta, J.J., Weitz, D.A., Sanes, J.R., Shalek, A.K., Regev, A., and McCarroll, S.A.: 'Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets', Cell, 2015, 161, (5), pp. 1202-1214

30 Zheng, G.X., Terry, J.M., Belgrader, P., Ryvkin, P., Bent, Z.W., Wilson, R., Ziraldo, S.B., Wheeler, T.D., McDermott, G.P., Zhu, J., Gregory, M.T., Shuga, J., Montesclaros, L., Underwood, J.G., Masquelier, D.A., Nishimura, S.Y., Schnall-Levin, M., Wyatt, P.W., Hindson, C.M., Bharadwaj, R., Wong, A., Ness, K.D., Beppu, L.W., Deeg, H.J., McFarland, C., Loeb, K.R., Valente, W.J., Ericson, N.G., Stevens, E.A., Radich, J.P., Mikkelsen, T.S., Hindson, B.J., and Bielas, J.H.: 'Massively parallel digital transcriptional profiling of single cells', Nature Communications, 2017, 8, pp. 14049

31 Han, X., Wang, R., Zhou, Y., Fei, L., Sun, H., Lai, S., Saadatpour, A., Zhou, Z., Chen, H., Ye, F., Huang, D., Xu, Y., Huang, W., Jiang, M., Jiang, X., Mao, J., Chen, Y., Lu, C., Xie, J., Fang, Q., Wang, Y., Yue, R., Li, T., Huang, H., Orkin, S.H., Yuan, G.C., Chen, M., and Guo, G.: 'Mapping the Mouse Cell Atlas by Microwell-Seq', Cell, 2018, 172, (5), pp. 1091-1107 e1017

32 Angerer, P., Simon, L., Tritschler, S., Wolf, F.A., Fischer, D., and Theis, F.J.: 'Single cells make big data: New challenges and opportunities in transcriptomics', Current Opinion in Systems Biology, 2017, 4, pp. 85-91

33 Xu, C., and Su, Z.: 'Identification of cell types from single-cell transcriptomes using a novel clustering method', Bioinformatics, 2015, 31, (12), pp. 1974-1980

34 Zeisel, A., Munoz-Manchado, A.B., Codeluppi, S., Lonnerberg, P., La Manno, G., Jureus, A., Marques, S., Munguba, H., He, L., Betsholtz, C., Rolny, C., Castelo-Branco, G., Hjerling-Leffler, J., and Linnarsson, S.: 'Brain structure. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq', Science, 2015, 347, (6226), pp. 1138-1142

35 Tasic, B., Menon, V., Nguyen, T.N., Kim, T.K., Jarsky, T., Yao, Z., Levi, B., Gray, L.T., Sorensen, S.A., Dolbeare, T., Bertagnolli, D., Goldy, J., Shapovalova, N., Parry, S., Lee, C., Smith, K., Bernard, A., Madisen, L., Sunkin, S.M., Hawrylycz, M., Koch, C., and Zeng, H.: 'Adult mouse cortical cell taxonomy revealed by single cell transcriptomics', Nature Neuroscience, 2016, 19, (2), pp. 335-346

36 Zhang, J.M., Fan, J., Fan, H.C., Rosenfeld, D., and Tse, D.N.: 'An interpretable framework for clustering single-cell RNA-Seq datasets', BMC Bioinformatics, 2018, 19, (1), pp. 93

37 Jianbo, S., and Malik, J.: 'Normalized cuts and image segmentation', IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22, (8), pp. 888-905

38 van Dijk, D., Sharma, R., Nainys, J., Yim, K., Kathail, P., Carr, A.J., Burdziak, C., Moon, K.R., Chaffer, C.L., Pattabiraman, D., Bierie, B., Mazutis, L., Wolf, G., Krishnaswamy, S., and Pe'er, D.: 'Recovering Gene Interactions from Single-Cell Data Using Data Diffusion', Cell, 2018, 174, (3), pp. 716-729 e727

39 Huang, M., Wang, J., Torre, E., Dueck, H., Shaffer, S., Bonasio, R., Murray, J.I., Raj, A., Li, M., and Zhang, N.R.: 'SAVER: gene expression recovery for single-cell RNA sequencing', Nature Methods, 2018, 15, (7), pp. 539-542

40 Arisdakessian, C., Poirion, O., Yunits, B., Zhu, X., and Garmire, L.: 'DeepImpute: an accurate, fast and scalable deep neural network method to impute single-cell RNA-Seq data', biorxiv preprint at www.biorxiv.org/content/early/2018/06/22/353607, accessed on April 28, 2019

41 Deng, Y., Bao, F., Dai, Q., Wu, L., and Altschuler, S.: 'Massive single-cell RNA-seq analysis and imputation via deep learning', biorxiv preprint at www.biorxiv.org/content/early/2018/06/28/315556, accessed on April 28, 2019

42 Hinton, G.E., and Salakhutdinov, R.R.: 'Reducing the dimensionality of data with neural networks', Science, 2006, 313, (5786), pp. 504-507

43 Chen, J., King, E., Deek, R., Wei, Z., Yu, Y., Grill, D., Ballman, K., and Stegle, O.: 'An omnibus test for differential distribution analysis of microbiome sequencing data', Bioinformatics, 2018, 34, (4), pp. 643-651

44 Bellman, R.E.: 'Adaptive control processes: a guided tour' (Princeton university press, 1961)

45 Hornik, K.: 'Approximation Capabilities of Multilayer Feedforward Networks', Neural Networks, 1991, 4, (2), pp. 251-257

46 Bengio, Y., Courville, A., and Vincent, P.: ‘Representation Learning: A Review and New Perspectives’, IEEE Trans. Pattern Anal. Mach. Intell., 2013, 35, (8), pp. 1798-1828

47 Lin, C., Jain, S., Kim, H., and Bar-Joseph, Z.: ‘Using neural networks for reducing the dimensions of single-cell RNA-Seq data’, Nucleic Acids Research, 2017, 45, (17), pp. e156

48 Wolf, F.A., Angerer, P., and Theis, F.J.: ‘SCANPY: large-scale single-cell gene expression data analysis’, Genome Biology, 2018, 19, (1), pp. 15

49 Nair, V., and Hinton, G.E.: ‘Rectified linear units improve restricted boltzmann machines’. Proceedings of the 27th International Conference on International Conference on Machine Learning, Haifa, Israel 2010 pp. 807-814

50 Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X.: ‘TensorFlow: a system for large-scale machine learning’. Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation, Savannah, GA, USA 2016, pp. 265-283

51 Kingma, D.P., and Ba, J.: ‘Adam: A method for stochastic optimization’, arXiv preprint arXiv:1412.6980, 2014

52 Reddi, S.J., Kale, S., and Kumar, S.: ‘On the convergence of adam and beyond’. International Conference on Learning Representations, Vancouver, BC, Canada, ICLR 2018

53 Zeiler, M.D.: ‘ADADELTA: an adaptive learning rate method’, arXiv preprint arXiv:1212.5701, 2012

54 Kingma, D.P., and Welling, M.: ‘Stochastic Gradient VB and the Variational Auto-Encoder’. Second International Conference on Learning Representations, ICLR 2014

55 Strehl, A., and Ghosh, J.: ‘Cluster ensembles --- a knowledge reuse framework for combining multiple partitions’, J. Mach. Learn. Res., 2003, 3, pp. 583-617

56 Kuhn, H.W.: ‘The Hungarian method for the assignment problem’, Naval Research Logistics Quarterly, 1955, 2, (1-2), pp. 83-97

57 Rand, W.M.: ‘Objective Criteria for the Evaluation of Clustering Methods’, Journal of the American Statistical Association, 1971, 66, (336), pp. 846-850

58 Hubert, L., and Arabie, P.: ‘Comparing partitions’, Journal of Classification, 1985, 2, (1), pp. 193-218

59 Zappia, L., Phipson, B., and Oshlack, A.: 'Splatter: simulation of single-cell RNA sequencing data', Genome Biology, 2017, 18, (1), pp. 174

60 Cao, J., Packer, J.S., Ramani, V., Cusanovich, D.A., Huynh, C., Daza, R., Qiu, X., Lee, C., Furlan, S.N., Steemers, F.J., Adey, A., Waterston, R.H., Trapnell, C., and Shendure, J.: 'Comprehensive single-cell transcriptional profiling of a multicellular organism', Science, 2017, 357, (6352), pp. 661-667

61 Shekhar, K., Lapan, S.W., Whitney, I.E., Tran, N.M., Macosko, E.Z., Kowalczyk, M., Adiconis, X., Levin, J.Z., Nemesh, J., Goldman, M., McCarroll, S.A., Cepko, C.L., Regev, A., and Sanes, J.R.: 'Comprehensive Classification of Retinal Bipolar Neurons by Single-Cell Transcriptomics', Cell, 2016, 166, (5), pp. 1308-1323 e1330

62 Dizaji, K.G., Herandi, A., Deng, C., Cai, W., and Huang, H.: 'Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization'. IEEE International Conference on Computer Vision (ICCV) 2017 pp. 5747-5756

63 Jang, E., Gu, S., and Poole, B.: 'Categorical Reparameterization with Gumbel-Softmax'. International Conference on Learning Representations, ICLR 2017

64 Wang, D., and Gu, J.: 'VASC: Dimension Reduction and Visualization of Single-cell RNA-seq Data by Deep Variational Autoencoder', Genomics Proteomics Bioinformatics, 2018, 16, (5), pp. 320-331

65 Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., and Smola, A.: 'A kernel two-sample test', J. Mach. Learn. Res., 2012, 13, pp. 723-773

66 Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B.: 'MMD GAN: Towards Deeper Understanding of Moment Matching Network'. Advances in Neural Information Processing Systems, 2017, pp. 2203-2213

67 Shaham, U., Stanton, K.P., Zhao, J., Li, H., Raddassi, K., Montgomery, R., and Kluger, Y.: 'Removal of batch effects using distribution-matching residual networks', Bioinformatics, 2017, 33, (16), pp. 2539-2546

68 Ioffe, S., and Szegedy, C.: 'Batch normalization: accelerating deep network training by reducing internal covariate shift'. Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, Lille, France 2015 pp. 448-456

69 Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.: 'Generative Adversarial Nets', Advances in Neural Information Processing Systems, NIPS 2014, pp. 2672-2680

70 Arjovsky, M., Chintala, S., and Bottou, L.: 'Wasserstein GAN', 2017, arXiv preprint arXiv:1701.07875