https://doi.org/10.1093/bib/bbac172
**Problem Solving Protocol**

# A parameter-free deep embedded clustering method for single-cell RNA-seq data

Yuansong Zeng, Zhuoyi Wei, Fengqi Zhong, Zixiang Pan, Yutong Lu and Yuedong Yang

Corresponding author: Yuedong Yang. E-mail: yangyd25@mail.sysu.edu.cn

## Abstract

Clustering analysis is widely used in single-cell ribonucleic acid (RNA)-sequencing (scRNA-seq) data to discover cell heterogeneity and cell states. While many clustering methods have been developed for scRNA-seq analysis, most of these methods require to provide the number of clusters. However, it is not easy to know the exact number of cell types in advance, and experienced determination is not always reliable. Here, we have developed ADClust, an automatic deep embedding clustering method for scRNA-seq data, which can accurately cluster cells without requiring a predefined number of clusters. Specifically, ADClust first obtains low-dimensional representation through pre-trained autoencoder and uses the representations to cluster cells into initial micro-clusters. The clusters are then compared in between by a statistical test, and similar micro-clusters are merged into larger clusters. According to the clustering, cell representations are updated so that each cell will be pulled toward centers of its assigned cluster and similar clusters, while cells are separated to keep distances between clusters. This is accomplished through jointly optimizing the carefully designed clustering and autoencoder loss functions. This merging process continues until convergence. ADClust was tested on 11 real scRNA-seq datasets and was shown to outperform existing methods in terms of both clustering performance and the accuracy on the number of the determined clusters. More importantly, our model provides high speed and scalability for large datasets.

**Keywords:** deep embedded clustering, dip-test, estimating the number of cell clusters, single-cell clustering, single-cell RNA sequencing

## Introduction

Recent advances in single-cell ribonucleic acid (RNA)-sequencing (scRNA-seq) technologies have paved the way for researchers to generate high-throughput single-cell gene expression [1]. A full characterization of transcriptome profiling at single-cell resolution holds enormous potential for discovering trajectories of different cell developmental states and investing the cellular heterogeneity [2, 3]. One important step to discover cell heterogeneity and cell states is to perform clustering analysis, which aims to group a set of cells into meaningful cell populations based on their transcriptome similarity [4, 5]. The clustering can be used as additional downstream analysis and provides a reference to build a cell atlas [6–8]. Nevertheless, the clustering is meeting grand challenges due to the characteristic of scRNA-seq data, such as sparsity and high-dimensional features [9–11].

To resolve these challenges, a wide variety of clustering algorithms have been developed for scRNA-seq analysis [4, 5]. Early popular algorithms are variants of *K*-means that divide cells into *K* clusters, with *K* as the predetermined number of clusters. For example, scDeepCluster [12] uses *K*-means to obtain initial centers of clusters and then pushes each cell to its most similar centers iteratively. Similar strategies have also been used by other methods such as SAIC [13], scVDMC [14], DESC [15] and the latest clustering method CaFew [16]. On the other hand, graph clustering is based on the community detection algorithms that cluster neighbored cells based on a resolution parameter. For example, Seurat [17], one of the most widely used toolkits for scRNA-seq analysis, connects cells into a KNN-graph and then partitions the graph into communities (clusters) through a predetermined resolution parameter, where a higher resolution generates a greater number of clusters. Similar strategies have also been used by other methods such as SNN-Cliq [18] and SCANPY [19]. While these two classes of methods are robust, they need a parameter (*K* or resolution) as a priori, which unfortunately is seldom known in advance.

To avoid the predetermined parameter, SIMLR [20] pre-estimates the number of clusters as the rank constraint and then combines graph diffusion to learn a cell similarity measure for clustering. Similarly, SC3 [21] also

**Yuansong Zeng** is a PhD student in the School of Computer Science and Engineering at the Sun Yat-Sen University.
**Zhuoyi Wei** is a master student in the School of Computer Science and Engineering at the Sun Yat-Sen University.
**Fengqi Zhong** is a master student in the School of Computer Science and Engineering at the Sun Yat-Sen University.
**Zixiang Pan** is a master student in the School of Computer Science and Engineering at the Sun Yat-Sen University.
**Yutong Lu** is a professor in the School of Computer Science and Engineering and the National Super Computer Center at Sun Yat-Sen University, Guangzhou.
**Yuedong Yang** is a professor in the School of Computer Science and Engineering and the National Super Computer Center at Sun Yat-Sen University, Guangzhou.
**Received:** December 21, 2021. **Revised:** March 25, 2022. **Accepted:** April 18, 2022

pre-estimates the number of clusters, but it then uses K-means to cluster cells from different eigenvectors and constructs a consensus matrix for clustering. However, the pre-estimated number of clusters is usually not accurate, causing low performance in the following clustering. Another strategy is to select optimal number of clusters according to the clustering results. For example, IKAP [22] clusters cells by overestimating the number of clusters in the PC space of principal component analysis (PCA) [23] and then iteratively merges the nearest clusters to determine the optimal number of clusters. The recently developed MultiK [24] generates multiple groups of clustering results using different number of clusters by tests and trials and selects the optimal number of clusters under certain evaluation criteria. Similar strategies are applied in methods such as Clustree [25], scClustViz [26] and TooManyCells [27]. Nevertheless, these methods are machine learning or statistics-based methods that have to decouple the feature extraction and clustering into two separate steps, whereas the pre-extracted features are not optimal for the subsequent clustering. At the same time, they learn cell representations through linear algorithms (mainly PCA), which cannot efficiently process the complex scRNA-seq data [28]. Additionally, since these algorithms need multiple tests and trials, they are time-consuming and cannot process large datasets with thousands of cells.

Here, we proposed ADClust, an automatic deep embedding clustering method for scRNA-seq data, which can accurately cluster cells without requiring a predefined number of clusters. Specifically, we first pre-train the autoencoder to learn the non-linear low-dimensional representation of original gene expression, which is used to cluster cells into a mass of micro-clusters. The micro-clusters are then compared in between through a statistical test for unimodality called Dip-test [29] to detect similar micro-clusters, and similar micro-clusters are merged through jointly optimizing the carefully designed clustering and autoencoder loss functions. This process continues until convergence. By benchmarked on 12 real scRNA-seq datasets, ADClust was shown to outperform existing methods in terms of both clustering performance and the accuracy on the number of the determined clusters. More importantly, ADClust showed a high speed and scalability on large datasets.

## Materials and methods

### Datasets and pre-processing

#### *Simulated datasets*

To evaluate the performance of ADClust on simulation datasets, we followed the study [30] to generate simulated scRNA-seq datasets using the package Splatter [31]. For simulating the scRNA-seq datasets with different clustering signal strengths, we generated datasets with different de.facScale in {0.2, 0.25, 0.3, 0.35, 0. 4}. A higher de.facScale value represents a stronger clustering signal, corresponding to easier datasets. Each dataset contains 2000 cells and 2000 genes. The parameter dropout.mid was set to 2 (fixed dropout rates around 45%). We followed the study [32] to set the parameter group.prob to {0.1, 0.15, 0.25, 0.5} so that each simulated data contained four groups of different proportions. We used default values for other parameters. We generate the dataset five times repeatedly with different random seeds for each setting and show the average results.

#### *Real datasets*

We employed the commonly used datasets from [30] that included 15 datasets. By removing seven small datasets containing <1000 cells, we finally kept eight datasets (Xin, Tasic, Baron Mouse, Klein, Romanov, Zeisel, Segerstolpe and Baron Human). In order to test the scalability of our model, we selected four largest datasets (Mouse retina, TM, PBMC 68K and COVID_19) from previous studies [12, 33, 34] containing 27 499, 54 865, 68 579 and 1.46 million cells, respectively. As detailed in Table 1, these datasets are involved in different biological processes and various tissues and contain different scales of cells ranging from thousands to tens of thousands derived from various scRNA-seq techniques. Each dataset was pre-processed using the standard procedure as proposed in Seurat. Concretely, we normalized the gene expression by the 'NormalizeData' function with the default parameter 'LogNormalize'('RC' for simulation data) and the scaling factor of 10 000. Then, the top 2000 highly variable genes were selected through the 'FindVariableFeatures' function based on the normalized matrix.

### The architecture of ADClust

This study proposed an automatic deep embedding clustering method that can accurately cluster cells without requiring to predefine the number of clusters. As shown in Figure 1, the ADClust model consists of two modules: the autoencoder and clustering optimization modules. The autoencoder aims to learn deep embedding representations of cells, and the clustering optimization module uses the learned embedding representations to cluster cells.

#### *Autoencoder module*

Autoencoder is used for embedding the input scRNA-seq gene expression data $X \in \mathbb{R}^{N\times d}$ into low-dimensional space, where $N$ and $d$ are the number of cells and the size of genes, respectively. Autoencoder is an unsupervised neural network that consists of the encoder and decoder modules [35]. The encoder tries to embed the input data into a latent, and the decoder tries to reconstruct the embedded data into its origin space. Thus, the autoencoder can efficiently learn the useful low-dimensional latent by minimizing the reconstruction loss $L_{\mathbf{res}}$ as follows:

$$L_{\mathbf{res}} = \frac{1}{|X|} \sum_{x\in X} \left\| x - \mathbf{dec}\left(\mathbf{enc}(x)\right)\right\|_2^2, \tag{1}$$

**Table 1.** Summary of the datasets used in this study

| Datasets | GSE/ID | Protocol | #Cells | #Genes | #Cell types |
|---|---|---|---|---|---|
| Xin | GSE81608 | SMARTer | 1600 | 39 851 | 8 |
| Tasic | GSE71585 | SMARTer | 1679 | 24 150 | 18 |
| Baron Mouse | GSE84133 | inDrop | 1886 | 14 878 | 13 |
| Klein | GSE65525 | inDrop | 2717 | 24 175 | 4 |
| Romanov | GSE74672 | Fluidigm C1 | 2881 | 24 341 | 7 |
| Zeisel | GSE60361 | STRT/C1 UMI | 3005 | 19 972 | 9 |
| Segerstolpe | E-MTAB-5061 | Smart-Seq2 | 3514 | 25 525 | 15 |
| Baron Human | GSE84133 | inDrop | 8569 | 20 125 | 14 |
| Mouse retina | GSE81904 | Drop-seq | 27 499 | 13 166 | 19 |
| TM | GSE109774 | Smart-Seq2 | 54 865 | 19 791 | 55 |
| PBMC 68k | SRP073767 | 10X chromium | 68 579 | 20 387 | 10 |
| COVID_19 | GSE158055 | 10X chromium | 1 462 702 | 27 943 | 12 |

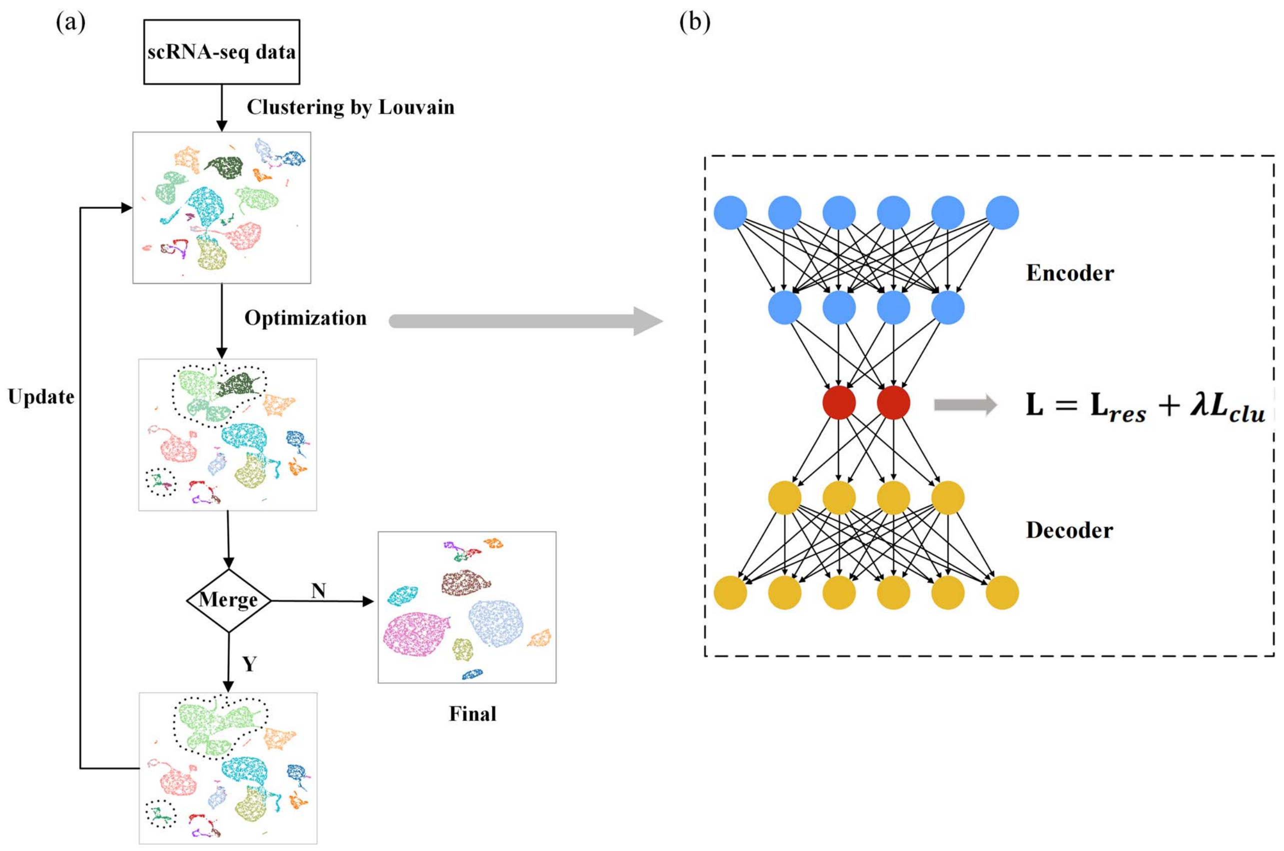

**Figure 1.** Overview of the (**A**) ADClust framework: scRNA-seq data are used to construct initial micro-clustering through the Louvain algorithm. The clustered results are input into (**B**) the clustering optimization module to optimize the reconstruction and clustering loss functions for the optimal cell representations according to which similar micro-clusters are detected through the Dip-test and merged to update the clustering results. This process iterates until no more cluster merging.

where $\|\cdot\|_2^2$ represents the square Euclidean, and the dec() and enc() represent encoder and decoder functions, respectively. The **enc**($x$) is the learned embedding representation for gene expression $x$ of individual cell.

### *The clustering optimization module*

Based on the learned embedding representations, cells are clustered through the Louvain algorithm [36] into plentiful initial micro-clusters. The micro-clusters are then compared in between by Dip-test, and similar micro-clusters are merged through a carefully designed clustering loss function.

### The Dip-test and Dip-score

The Dip-test is a statistical test to measure modality, which was first developed by J. A. Hartigan and P. M. Hartigan in the 1980s [29]. The test detects the sample (cell) set of two clusters and outputs a Dip-score ∈[0,1] as the probability to unimodality of the sample set. Two micro-clusters with a larger Dip-score represent a higher structural similarity in between and will be merged in our method.

### Clustering loss

Similar micro-clusters were pulled together in the embedding space of autoecnoder through the clustering loss $L_{\mathrm{clu}}$ originally used in [37]:

$$L_{\mathrm{clu}} = \frac{(1+\mathbf{std}\,(D_C))}{\mathbf{mean}\,(D_C)}\frac{1}{|X|}\sum_{x\in X}\sum_{i=1}^{K}\tilde{P}\,(c_x, i)\,\|\mathrm{enc}(x)-\mu_i\|_2^2, \tag{2}$$

where $c_x$ is the cluster containing cell $x$, $\mu$ is the centers for $K$ clusters, $\tilde{P}(c_x, i) = P(c_x, i)/\sum_{j=1}^{K} P\,(c_x, j)$ reflects cluster similarity by normalizing the Dip-score $P\,(c_x, j)$ between the cluster centers of $c_x$, and $i$ estimated through the Dip-test [29]. The 'mean' and 'std' are the mean and SD of the set of cluster-pairwise distances $D_C$.

$$D_C = \left\{\sqrt{\|\mu_i-\mu_j\|_2^2} \mid i \in [1, K-1] \text{ and } j \in [i+1, K]\right\}. \tag{3}$$

Intuitively, our model will minimize the loss by reducing the distance between a cell and the center of its assigned center. At the same time, the cell will also be pulled toward its similar clusters with strength depending on the similarity to the cluster $i$. As a result, the model will reduce the distance between clusters if they are similar with a large Dip-score. This process will pull similar micro-clusters together. The division by the mean $(D_C)$ was used to prevent the autoencoder from only reducing the embedding scale to minimize loss $L_{\mathrm{clu}}$. We also apply the term **std** $(D_C)$ to impede the model's ability to reduce the scale and simultaneously push individual clusters far away.

Finally, we optimize ADClust with the following loss in an end-to-end manner:

$$L = L_{\mathrm{res}} + \lambda L_{\mathrm{clu}}, \tag{4}$$

where $\lambda$ is the hyper-parameter to balance contributions from the clustering loss. In this study, we set$\lambda = 1$ for all datasets.

#### *Merging process*

We merge two clusters if their corresponding Dip-score is larger than the Dip-score threshold. Cells in these two merging clusters will be assigned the same cell label, and a new center of these cells will be computed as the following:

$$\mu_{\mathrm{new}} = \underset{x\in C_i\cup C_j}{\mathbf{arg\ min}}\left(\left\|\mathrm{enc}(x) - \frac{|C_i|\,\mu_i + |C_j|\,\mu_j}{|C_i| + |C_j|}\right\|_2^2\right). \tag{5}$$

Based on the new center $\mu_{\mathrm{new}}$, we need to update the Dip-score matrix $P$ and $\tilde{P}$. The merging process is repeated if there is still a Dip-score in $P$ that is greater than the threshold. After the merging process, we continue optimizing the model and merging the clusters. This process continues until there is no Dip-score greater than the threshold.

## Hyper-parameters setting

The ADClust was implemented in PyTorch and C. The dimensions of the autoencoder were set to input-512-256-128-10-128-256-512-input. The training batch size was generally set as 128, while the size was increased for large datasets (1024 for above 10 000 cells) to further reduce the training time of each epoch like in ref [38]. The models were optimized through the Adam optimizer with a learning rate of 0.0001. We empirically set the number of epochs in pre-trained autoencoder to 100 and 400 for real datasets and simulated datasets, respectively. We empirically set resolution = 3 in the Louvain algorithm for all datasets to obtain the initial number of clusters that were much larger than the true number of clusters. (We listed the true number of clusters and the initial number of clusters for all datasets in Supplementary Table S1 available online at http://bib.oxfordjournals.org/.) We obtained the initial micro-clusters by Louvain algorithm, while the Louvain was not further optimized. The Dip-score threshold was set to 0.9, which determined whether two clusters should be merged. The number of epochs for the clustering process was set to 50. All results reported in this paper were conducted on Ubuntu 16.04.7 LTS with Intel® Core (TM) i7-8700K CPU @ 3.70 GHz and 256 GB memory, with the Nvidia Tesla P100 (16G).

## Evaluation criteria

#### *Evaluation metric for clustering*

Five common clustering metrics are used for evaluating cell-clustering results in this study: normalized mutual information (NMI) [39], adjusted rand index (ARI) [40], clustering accuracy (CA) [41], Fowlkes–Mallows index (FMI) [42] and silhouette coefficient (SC) [43].

The NMI is defined as

$$\mathrm{NMI}\,(Y, C) = \frac{2\times[H(Y) - H\,(Y|C)]}{[H(Y) + H(C)]}, \tag{6}$$

where C and Y are the predicted clusters and the true clusters (the same below), respectively. The term $H(\,)$ is used for computing the entropy.

The ARI is defined as

$$\mathrm{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i\binom{a_i}{2}\sum_j\binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{a_i}{2} + \sum_j\binom{b_j}{2}\right] - \left[\sum_i\binom{a_i}{2}\sum_j\binom{b_j}{2}\right]/\binom{n}{2}}, \tag{7}$$

[[DmEquation7]]where $a_i$ is the number of cells appearing in the $i$th cluster of C, $b_j$ is the number of cells appearing in the $j$th cluster of Y and $n_{ij}$ is the number of overlaps between the $j$th cluster of Y and the $i$th cluster of C.

The CA is calculated as

$$\mathrm{CA} = \max_{m} \frac{\sum_{i=1}^{n} 1\left\{l_i = m\left(c_i\right)\right\}}{n}, \quad (8)$$

where $n$ is the total number of cells and $m$ ranges over all probable one-to-one mapping between clustering assignment $c_i$ and real label $l_i$.

The FMI is derived from the true positives (TP), false positives (FP) and false negatives (FN) as follows:

$$\mathrm{FMI} = \sqrt{\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \cdot \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}}. \quad (9)$$

The SC for cell $i$ is defined as the following:

$$\mathrm{SC}(i) = \frac{b(i) - a(i)}{\max\left\{a(i), b(i)\right\}}, \quad (10)$$

where $b(i)$ and $a(i)$ represent the mean nearest-cluster distance and the mean intra-cluster distance for sample $i$, respectively.

#### *Accuracy evaluation of the number of determined clusters*

For evaluating the accuracy of each method in determining the number of clusters for all datasets, we use the mean absolute error (MAE) as the evaluation index of accuracy. The MAE is defined as

$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m}\left|\left(y_i - \tilde{y}_i\right)\right|, \quad (11)$$

where $y_i$ and $\tilde{y}_i$ is the true number of clusters and the determined number of clusters, respectively.

### Benchmark methods

To evaluate the clustering performance, we compared ADClust with other tools including MultiK, SIMLR, scDeepCluster, SC3, scQcut [44], IKAP, CIDR [45], Seurat (version 3.0), SHARP [46], scGMAI [47] and DESC. As MultiK outputs multiple estimated number of clusters, we selected the estimated number of clusters with the highest ARI. We set the 'NUMC' parameter of SIMLR as a range [2:20] to estimate the number of clusters followed [44]. We set the true number of clusters for scDeepCluster since it could not estimate the number of clusters. For other competing methods, we used the default hyperparameters recommended in the origin paper to estimate the number of clusters.

## Results

### Clustering performance

In this section, we evaluated the clustering performance of ADClust on both simulated and real datasets. We first evaluated our method under different scenes through simulated scRNA-seq datasets generated from Splatter. As shown in Supplementary Figure S1**A**, available online at http://bib.oxfordjournals.org/, the ARIs of all methods increased with the growth of de.facScale since higher de.facScale represented a stronger signal, corresponding to easier datasets. Overall, the average ARI of our method (0.78) was 16% higher than the second-ranked method Seurat. Although MultiK and scQcut achieved comparable results with our method at the de.facScale of 0.4, MultiK and scQcut performed low at de.facScale of 0.2 with ARI <0.1. IKAP and scGMAI could achieve decent results at de.facScale of 0.4 with ARI of 0.7 and 0.6, respectively. As shown in Supplementary Figure S1**B**–**E**, available online at http://bib.oxfordjournals.org/, similar trends could also be observed in terms of other evaluation criteria.

To evaluate the clustering performance of ADClust on real datasets, we applied our model to 11 scRNA-seq datasets, including 8 small datasets (containing <10 000 cells) and 3 large datasets (containing >20 000 cells). On the eight small datasets, our model showed superior clustering performance compared with competing algorithms (Figure 2**A**). On average, ADClust achieved ARI of 0.78, which was 8% higher than the one achieved by the second best method MultiK. The third-ranked method scQcut is a graph partitioning algorithm achieving an ARI of 0.61. This value was 10% higher than Seurat, another graph partitioning algorithm. The better performance by scQcut is likely because scQcut optimized the number of neighbors for the KNN-graph [44]. The fourth-ranked method DESC achieved decent performance since it jointly optimized cell labels assignment and learned the latent representation that was fitted for clustering. CIDR and SIMLR achieved a similar and low performance since their pre-estimated number of clusters in advance was usually incorrect. SHARP and scGMAI performed better than SC3 and SIMLR, while they were still 25% and 28% lower than our method in terms of average ARI. scDeepCluster ranked the ninth, although it was inputted with the real number of clusters. This is likely because its performance heavily relied on the initialized results of *K*-means. SC3 performed the worst since it was sensitive to parameters used in dimension reduction and tended to overestimate the number of clusters, as also indicated in previous studies [24, 48]. We also tested CaFew by the combination of SC3. Although it achieved an average ARI of 0.47, 15% better than SC3 (similar to 14% as reported in the original paper [16]), but worse than our method, MultiK, scQcut, DESC, IKAP and SHARP (average ARIs of 0.78, 0.70, 0.61, 0.56, 0.54 and 0.53, respectively).

On four large datasets with the number of cells >20 000, ADClust consistently achieved the best clustering performance (Figure 2B). On average, ADClust achieved ARI of 0.70, 14% higher than the one achieved by the second best method scDeepCluster. The third-ranked method DESC achieved a similar performance with the fourth-ranked method Seurat. IPKA and scQcut only could run on the large dataset Mouse retina, and the ARI values of them were 81% and 8% smaller than our method (ARI = 0.93), respectively. Our method

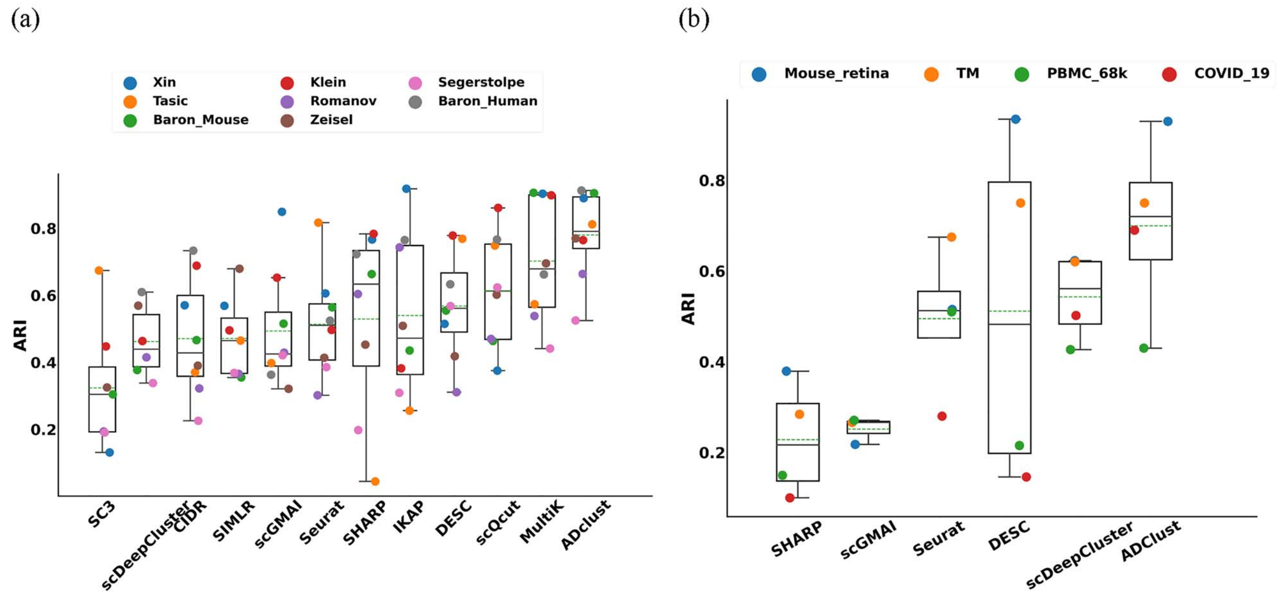

**Figure 2.** Clustering performance of all methods on (**A**) small scRNA-seq datasets with <10 000 cells and (**B**) large scRNA-seq datasets with cells >20 000. The dashed line indicates the mean value of ARI.

outperformed Seurat, scDeepCluster, DESC and SHARP when clustering the dataset COVID_19 with 1.46 million cells. We did not compare with IPKA and scQcut (on large datasets containing >50 000 cells), scGMAI (on COVID_19 dataset), CIDR, SC3, SIMLR and MultiK due to occurring errors (scQcut), out of memory (SIMLR and CIDR), 'NAN' values generation (SC3), segmentation fault in function FastICA (scGMAI) or the runtime of more than two days (IPKA and MultiK).

We also show the comparisons on 11 scRNA-seq datasets for evaluation criteria CA, NMI, FMI and SC in Supplementary Figure S2, available online at http://bib.oxfordjournals.org/, and similar trends could also be observed.

## The evaluation on the number of determined clusters

We first evaluated the accuracy of ADClust in the number of determined clusters on simulated datasets. As shown in Supplementary Figure S3 available online at http://bib.oxfordjournals.org/, the MAE (0.79) by our method was slightly better than the second-ranked method SIMLR (1.12). SHARP and IPKA ranked third and fourth but had higher MAEs (1.24 and 1.68, respectively). The MAE of scQcut was significantly higher than other methods due to the overestimation of the number of clusters when de.facScale was <0.3.

To evaluate the accuracy in the number of determined clusters on real datasets, we applied our model on all real scRNA-seq datasets. Since Seurat, scDeepCluster and DESC could not estimate the number of clusters, we did not compare with them. As shown in Figure 3**A**, on eight small datasets, the median absolute deviation of the number of clusters determined by our model was closest to zero, which was the smallest of the seven methods. Additionally, we evaluated all methods by the MAE between the actual and determined numbers of clusters. As shown in Figure 3**A**, our method achieved the lowest MAE of 2.88, which was less than the second-ranked method scQcut (3.75). IPKA and Multik achieved a similar MAE of around 4.3. SC3 achieved the worst performance in terms of MAE due to the overestimation on the number of clusters. SHARP and scGMAI achieved MAEs of 7.75 and 3.88, worse than our method (2.88). We further showed the specific number of clusters determined by each method in Supplementary Table S2 available online at http://bib.oxfordjournals.org/. For the eight small datasets, the number of clusters determined by our model in four datasets was the most accurate. scQcut and IKAP achieved the second best performance and made the most accurate estimation for three datasets. The number of clusters determined by SIMLR, CIDR and SC3 was usually incorrect. When tested on three large datasets, our model achieved more accurate estimation than IKAP and scQcut (Supplementary Table S2 available online at http://bib.oxfordjournals.org/). Other clustering methods could not achieve corresponding results on larger datasets due to error generation or the runtime of >2 days.

To view the accuracy of the number of clusters determined by each method more clearly, we further drew a scatter plot with the number of determined clusters and the number of true clusters. On eight small datasets, as shown in Figure 3**B** and Supplementary Figure S4, available online at http://bib.oxfordjournals.org/, the number of clusters determined by our model was more similar to the true number of clusters when compared with MultiK, the method with the second-highest clustering performance. SC3 tended to overestimate the number of clusters, but SIMLR and CIDR tended to underestimate

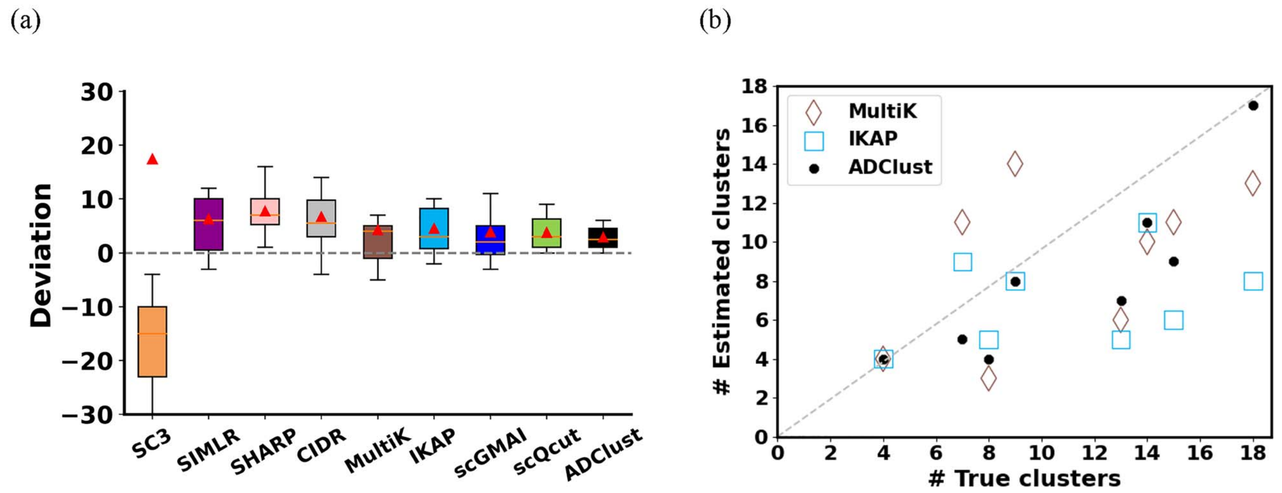

**Figure 3.** The accuracy in the number of determined clusters. (**A**) The deviations between the actual and determined number of clusters. The red triangle represents the MAE between the actual and determined number of clusters on eight small datasets. (**B**) Correlations between the actual and determined number of clusters.

the number of clusters. Our model also tended to underestimate the number of clusters on a few datasets.

To investigate why our model underestimated the number of clusters on datasets Xin, Baron Mouse, Segerstolpe, Mouse retina and TM, we analyzed the cell composition of these datasets. As shown in Supplementary Table S3–S7, available online at http://bib.oxfordjournals.org/, we found that these datasets contained multiple rare cell clusters [49], and these rare cell clusters consist of $\leq$1.5% of the total cell population on average (Xin < 1.5%, Baron Mouse < 0.7%, Segerstolpe < 0.5%, Mouse retina < 1% and TM < 0.09%). In summary, all methods, except SC3, tended to underestimate the number of clusters due to the inclusion of rare cell clusters in many datasets. However, the number of clusters estimated by SC3 was much larger than the true number of clusters.

## Contribution of components to the clustering

To investigate the contributions of components for the clustering performance of ADClust, we conducted ablation studies on all scRNA-seq datasets. As shown in Table 2, the initial clustering results of ADClust achieved the worst performance with 0.236, 0.620 and 0.347 in terms of ARI, NMI and CA on average, respectively. The results showed that ADClust failed to achieve the desired performance when the number of clusters was overestimated. We noticed that the value of NMI was much greater than both ARI and CA. This is likely because initially the number of micro-clusters was much larger than the number of actual clusters and each initial micro-cluster might contain only one cell type, resulting in a wrongly high NMI value. The removal of both clustering and autoencoder losses caused decreases of 7%, 6% and 9.6% in terms of ARI, NMI and CA, respectively. The changes indicated ADClust could achieve decant performance by jointly optimizing cell labels assignment and learning embedded representations. The removal of the clustering loss caused decreases of 6%, 3.9% and 7.5% in terms of ARI, NMI and CA on average, respectively. The results indicated that the similar micro-clusters were efficiently pulled together in the low-embedding representation of the autoencoder by optimizing clustering loss. The removal of autoencoder loss in the clustering phase caused a small but significant drop (3.4%, 2.2% and 2.5% in terms of ARI, NMI and CA, respectively), indicating the importance of autoencoder for improving the representation. In summary, the better clustering of the scRNA-seq data relied on the cooperation of the modules.

We have conducted sensitivity experiments on the hyper-parameter $\lambda$ and Dip-score threshold on real datasets. As shown in Supplementary Figure S5, available online at http://bib.oxfordjournals.org/, our method was insensitive to the hyper-parameters with average ARIs changes <2%. Similar trends could be found when evaluated by NMI, CA, FMI and SC.

## Illustration of the ADClust

To illustrate how our model worked, we visualized the merging process through UMAP [50]. Here, we took the Baron Human dataset containing 14 original cell types as an example. As shown in Figure 4**A**, the Baron Human dataset was clustered into 34 initial classes in this example by using the Louvain algorithm with resolution = 3.0. By minimizing clustering and autoencoder loss functions, similar micro-clusters were pulled together. As shown in Figure 4**B**, most of the initial clusters were mixed with their similar clusters, resulting in multiple larger clusters with the characteristics of intra-cluster compactness and inter-cluster separability. Compared with the true cell clusters, as shown in Figure 4**C**, most similar micro-clusters were correctly combined by our model. The results indicated our model could efficiently cluster cells without requiring a predefined

**Table 2.** Average results of ablation experiments on all real datasets

| Ablation tests | ARI | NMI | CA |
|---|---|---|---|
| Initial micro-clusters | 0.236 ± 0.1 | 0.620 ± 0.09 | 0.347 ± 0.09 |
| ADClust—clustering and autoencoder losses | 0.690 ± 0.16 | 0.730 ± 0.1 | 0.724 ± 0.09 |
| ADClust—clustering loss | 0.70 ± 0.18 | 0.751 ± 0.09 | 0.745 ± 0.11 |
| ADClust—autoencoder loss | 0.726 ± 0.18 | 0.768 ± 0.11 | 0.795 ± 0.11 |
| ADClust | 0.760 ± 0.16 | 0.790 ± 0.1 | 0.820 ± 0.09 |

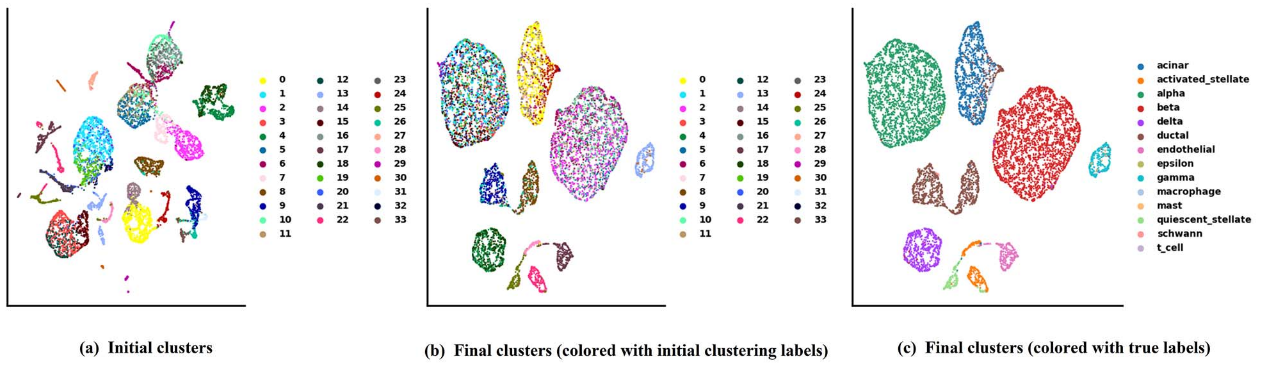

**Figure 4.** Visualizing our method on the Baron Human dataset containing 14 cell types. (**A**) The Baron Human dataset was clustered into 34 initial classes by using the Louvain algorithm with parameter resolution = 3.0. (**B**) Final clustering results colored with initial clustering labels. (**C**) Final clustering results colored with true labels.

number of clusters. We also visualized six other scRNA-seq datasets with different sequencing technologies in Supplementary Figure S6, available online at http://bib.oxfordjournals.org/, and our method consistently performed well.

To further confirm the clustering performance of ADClust, we visualized the wrongly clustered cells by the Sankey river plots on the Baron Human dataset. As shown in Figure 5, ADClust achieved CA, NMI and ARI values of 0.89, 0.88 and 0.913, respectively. For the two major cell types beta and alpha, which together account for the biggest portion (57%), our model could correctly assign 98% cells. The second best method CIDR could correctly assign 93% of cells. Other methods made the accuracy of 60–82% on beta and alpha cell types (Supplementary Figure S7 available online at http://bib.oxfordjournals.org/). One major source of wrong assignments in our model was the separation of the ductal cells into three clusters. The separation of ductal cells was also seen in all competing methods. These similar mistakes may come from the difficulty of clustering this cell type.

## Comparison of running time and memory usage

With advances in scRNA-seq technologies, the cells in emerging scRNA-seq datasets can exceed hundreds of thousands, requiring their scalability and efficiency of methods. For evaluating the runtimes of all methods and their scalability, we applied all methods to scRNA-seq datasets with a wide range of sizes. As shown in Figure 6, dramatic differences in runtimes can be observed among these methods with increasing the number of cells. ADClust was faster than all competing clustering methods. ADClust showed high scalability with about linear growth of runtimes with the number of cells: 36 s for about 2K cells and 900 s for about 70K. The next fastest method CIDR was close to our algorithm in speed for datasets with <4K cells, but the runtimes remarkably increased with the increase in the number of cells. When the number of cells reached 8K, CIDR was more than five times slower than our model. MultiK was the slowest method and significantly slower than all methods, which needed more than 2 days when running datasets with larger than 10K cells. Our method achieved comparable time costs to SHARP, and both of them were faster than scGMAI. scQcut and SIMLR did not take any strategies to filter genes. To fairly compare the time costs, we also tested scQcut and SIMLR by using 2000 highly variable genes. In spite of slight speed up, they are still 10 and 20 times slower than our method, respectively. We did not include partial algorithms for large datasets because they failed to run due to out of memory (SIMLR and CIDR) or 'NAN' values generation (SC3) or the runtime of more than 2 days (IKAP and MultiK). Although Seurat was faster than our model when the number of cells was <70K, its ARI was averagely 23% lower than our method (Supplementary Figure S8 available online at http://bib.oxfordjournals.org/).

We also tested the memory usage of all methods on real datasets. As shown in Supplementary Figure S9, available online at http://bib.oxfordjournals.org/, ADClust took almost constant memory with respect to the sample size because of the minibatch parameter update, with the increase attributed to the data loader.

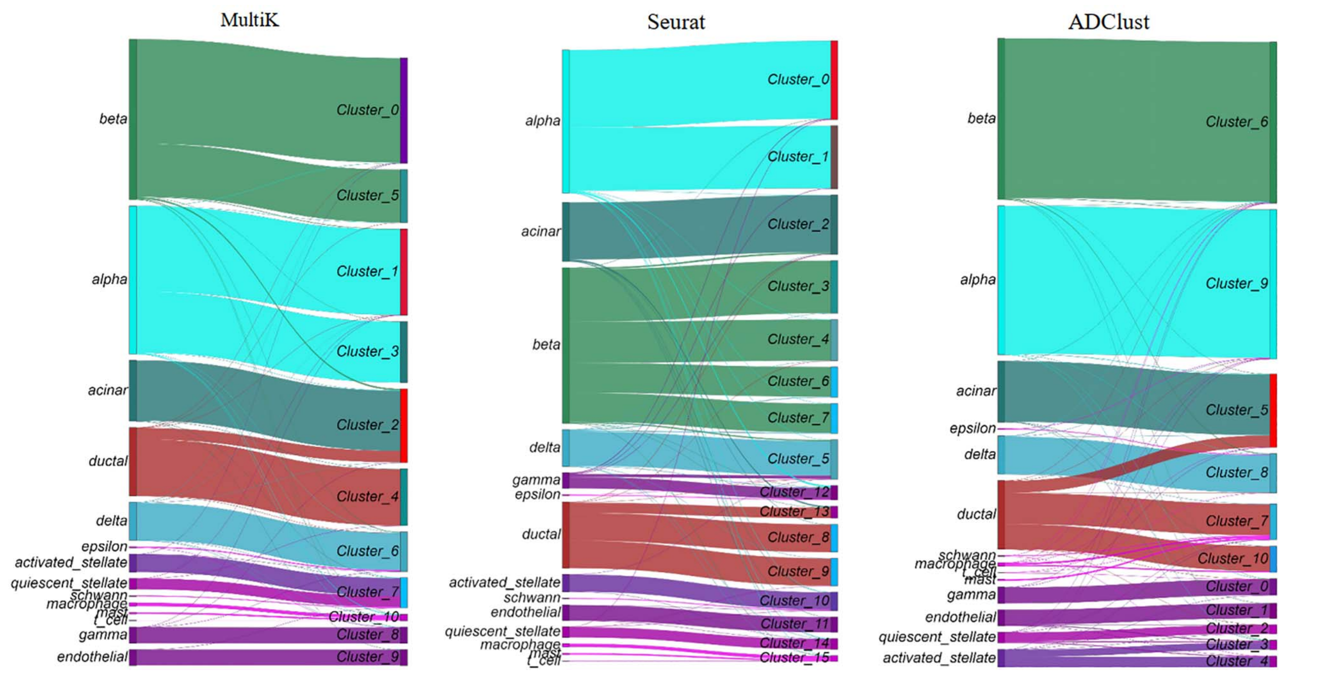

**Figure 5.** A Sankey river plot shows the match between the actual labels and clustering results on the Baron Human dataset.

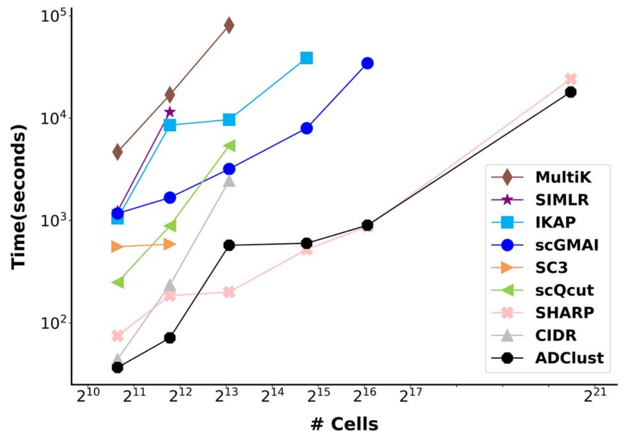

**Figure 6.** Comparison of different methods for the running time on variably sized datasets.

Although most methods took <5 GB of memory when the number of cells was within 8K, the memory requirement of them increased rapidly with the number of cells. For the large dataset with 1.46 million cells, ADClust could perform the clustering analysis within 20 GB memory. In contrast, SHARP required >100 GB of memory, which was not feasible for most computers. Our method achieved similar time costs but less memory consumption than SHARP on the large dataset COVID_19. Other methods could not deal with the large datasets due to various errors (see above). The results showed that our method could apply to large datasets with >1 million cells. We also noticed that most of our compared methods were implemented in R language, and our efficiency might result partly from our implementation in Python.

## Discussion

The optimization of clustering algorithms is being consumingly studied in scRNA-seq analysis. One critical challenge of clustering algorithms is to accurately cluster cells into meaningful groups without predefining the number of clusters. For this challenge, we proposed ADClust, an automatic deep embedding clustering method for scRNA-seq data, which can accurately cluster cells without requiring a predefined number of clusters. ADClust first clusters cells into the overestimated number of micro-clusters and then pushes micro-clusters sharing structural similarities together by jointly optimizing the clustering and autoencoder loss functions. On 12 real scRNA-seq datasets, our model demonstrated better performance in terms of both clustering performance and the accuracy on the number of the determined clusters. More importantly, our model provided high speed and scalability for large scRNA-seq datasets.

While a few methods, such as MultiK, are also used for simultaneously clustering scRNA-seq data and estimating the number of clusters through multiple tests and trials, it is necessary to strike a balance between performance and time consumption for these methods. In contrast, we cluster cells by iteratively merging similar micro-clusters through minimizing clustering and autoencoder loss functions. Our model achieved superior clustering performance by jointly optimizing the cell labels assignment and learning the representations that are suitable for the clustering. More importantly, ADClust is scalable and fast since we train our model with the means of mini-batches by using GPU. In short, our model achieved superior results in terms of both performance and efficiency.

The real datasets used in this study were from seven different sequencing technologies (shown in Table 1). These sequencing technologies included the most popular sequencing platforms, such as 10X chromium and Smart-Seq2. As shown in Supplementary Figure S10, available online at http://bib.oxfordjournals.org/, our method achieved similar trends with Seurat in terms of clustering performance on different sequencing technologies. The ARIs of SMARTer, inDrop and Drop-seq were higher than the average ARI, while the ARIs of Fluidigm C1, Smart-Seq2 and 10X chromium were lower than the average ARI. The difference in performance among these sequencing technologies may come from the variation in the efficiency of RNA molecule capture [51].

Despite the advantages of ADClust, our model can be improved in several aspects. First, our model may fail to distinguish between subtypes of cells since they have extremely similar gene expressions. We could add prior information such as marker genes into our model. Second, our model does not consider batch effects and we will add modules to remove batch effects [15]. This is important with the decreasing scRNA-seq costs and increasing international collaborations. Third, small and rare clusters may not be detected by our model since the Dip-test might identify two clusters as unimodal if they differ greatly in sizes.

In summary, we demonstrate that ADClust provides an automatic deep embedded clustering algorithm, which provides stable clustering solutions for scRNA-seq datasets without requiring the predefined number of clusters. In addition, it is worth noting that the concept of ADClust is applicable beyond scRNA-seq data, such as mass cytometry and scATAC-seq data.

**Key Points**

- Clustering analysis is widely utilized in single-cell RNA-sequencing (scRNA-seq) data to discover cell heterogeneity and cell states. While many clustering methods have been developed for scRNA-seq analysis, most of these methods require providing the number of clusters. However, it is not easy to know the exact number of cell types in advance.
- We proposed ADClus, a deep learning-based method for accurately clustering single-cell data without requiring a predefined cluster number by iteratively merging similar micro-clusters into larger clusters through jointly optimizing the carefully designed clustering and autoencoder loss functions.
- ADClust was tested on real scRNA-seq datasets, and shown to outperform existing methods in terms of both clustering performance and the accuracy on the number of the determined clusters. More importantly, our model provides high speed and scalability for large datasets.

## Supplementary data

Supplementary data are available online at http://bib.oxfordjournals.org/.

## Code availability

All source codes used in our experiments have been deposited at https://github.com/biomed-AI/ADClust.

## Data availability

The scRNA-seq datasets that support the findings of this study are available at https://www.synapse.org/#!Synapse:syn26524750/files/.

## Funding

National Key R&D Program of China (2020YFB0204803); National Natural Science Foundation of China (61772566); Guangdong Key Field R&D Plan (2019B020228001 and 2018B010109006); Introducing Innovative and Entrepreneurial Teams (2016ZT06D211); Guangzhou S&T Research Plan (202007030010).

## References

1. T. M. Consortium. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. *Nature* 2018;**562**(7727):367–72.
2. Tang F, Barbacioru C, Wang Y, *et al.* mRNA-Seq whole-transcriptome analysis of a single cell. *Nat Methods* 2009;**6**(5): 377–82.
3. Huang S. Non-genetic heterogeneity of cells in development: more than just noise. *Development* 2009;**136**(23):3853–62.
4. Krzak M, Raykov Y, Boukouvalas A, *et al.* Benchmark and parameter sensitivity analysis of single-cell RNA sequencing clustering methods. *Front Genet* 2019;**10**:1253. https://doi.org/10.3389/fgene.2019.01253.
5. Li R, Guan J, Zhou S. Single-cell RNA-seq data clustering: a survey with performance comparison study. *J Bioinform Comput Biol* 2020;**18**(4):2040005.
6. Peyvandipour A, Shafi A, Saberian N, *et al.* Identification of cell types from single cell data using stable clustering. *Sci Rep* 2020;**10**(1):1–12.
7. Rozenblatt-Rosen O, Stubbington MJ, Regev A, *et al.* The Human Cell Atlas: from vision to reality. *Nature* 2017;**550**(7677):451.
8. Davie K, Janssens J, Koldere D, *et al.* A single-cell transcriptome atlas of the aging *Drosophila* brain. *Cell* 2018;**174**(4):982–98.e20.
9. Svensson V. Droplet scRNA-seq is not zero-inflated. *Nat Biotechnol* 2020;**38**(2):147–50.
10. Vieth B, Ziegenhain C, Parekh S, *et al.* powsimR: power analysis for bulk and single cell RNA-seq experiments. *Bioinformatics* 2017;**33**(21):3486–8.
11. Rao J, Zhou X, Lu Y, *et al.* Imputing single-cell RNA-seq data by combining graph convolution and autoencoder neural networks. *Iscience* 2021;**24**(5):102393.
12. Tian T, Wan J, Song Q, *et al.* Clustering single-cell RNA-seq data with a model-based deep learning approach. *Nat Mach Intell* 2019;**1**(4):191–8. https://doi.org/10.1038/s42256-019-0037-0.
13. Lu Y, Liu J, Lu Q, *et al.* SAIC: an iterative clustering approach for analysis of single cell RNA-seq data. *BMC Genomics* 2017;**18**(S6):689.

14. Zhang H, Lee C, Li Z, *et al.* A multitask clustering approach for single-cell RNA-seq analysis in recessive dystrophic epidermolysis bullosa. *PLoS Comput Biol* 2018;**14**(4):e1006053.
15. Li X, Wang K, Lyu Y, *et al.* Deep learning enables accurate clustering with batch effect removal in single-cell RNA-seq analysis. *Nat Commun* May 11 2020;**11**(1):2338. https://doi.org/10.1038/s41467-020-15851-3.
16. Li R-Y, Guan J, Zhou S. Boosting scRNA-seq data clustering by cluster-aware feature weighting. *BMC Bioinform* 2021;**22**(6): 1–16.
17. Satija R, Farrell JA, Gennert D, *et al.* Spatial reconstruction of single-cell gene expression data. *Nat Biotechnol* 2015;**33**(5): 495–502. https://doi.org/10.1038/nbt.3192.
18. Xu C, Su Z. Identification of cell types from single-cell transcriptomes using a novel clustering method. *Bioinformatics* 2015;**31**(12):1974–80. https://doi.org/10.1093/bioinformatics/btv088.
19. Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol* 2018;**19**(1):15. https://doi.org/10.1186/s13059-017-1382-0.
20. Wang B, Zhu J, Pierson E, *et al.* Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nat Methods* 2017;**14**(4):414–6.
21. Kiselev V Y, Kirschner K, Schaub M T, *et al.* SC3: consensus clustering of single-cell RNA-seq data. *Nat Methods* 2017;**14**(5): 483–6.
22. Chen Y C, Suresh A, Underbayev C, *et al.* IKAP—identifying K mAjor cell population groups in single-cell RNA-sequencing analysis. *GigaScience* 2019;**8**(10):giz121.
23. Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. *Philos Trans Royal Soc A* 2016;**374**(2065):20150202.
24. Liu S, Thennavan A, Garay JP, *et al.* MultiK: an automated tool to determine optimal cluster numbers in single-cell RNA sequencing data. *Genome Biol* 2021;**22**(1):1–21.
25. Zappia L, Oshlack A. Clustering trees: a visualization for evaluating clusterings at multiple resolutions. *Gigascience* 2018;**7**(7):giy083.
26. Innes BT, Bader GD. scClustViz-single-cell RNAseq cluster assessment and visualization. *F1000Research* 2018;**7**:ISCB Comm J-1522.
27. Schwartz G W, Zhou Y, Petrovic J, *et al.* TooManyCells identifies and visualizes relationships of single-cell clades. *Nat Methods* 2020;**17**(4):405–13. https://doi.org/10.1038/s41592-020-0748-5.
28. Wang D, Gu J. VASC: Dimension reduction and visualization of single-cell RNA-seq data by deep variational autoencoder. *Genom Proteom Bioinform* 2018;**16**(5):320–31. https://doi.org/10.1016/j.gpb.2018.08.003.
29. Hartigan JA, Hartigan PM. The dip test of unimodality. *Ann Stat* 1985;**13**(1):70–84.
30. Zeng Y, Zhou X, Rao J, *et al.* Accurately clustering single-cell RNA-seq data by capturing structural relations between cells through graph convolutional network. In: *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Virtual Event, South Korea, pp. 519–22, 2020.
31. Zappia L, Phipson B, Oshlack A. Splatter: simulation of single-cell RNA sequencing data. *Genome Biol* 2017;**18**(1):174. https://doi.org/10.1186/s13059-017-1305-0.
32. Sun S, Zhu J, Ma Y, *et al.* Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis. *Genome Biol* 2019;**20**(1):269. https://doi.org/10.1186/s13059-019-1898-6.
33. Abdelaal T, Michielsen L, Cats D, *et al.* A comparison of automatic cell identification methods for single-cell RNA sequencing data. *Genome Biol* 2019;**20**(1):194. https://doi.org/10.1186/s13059-019-1795-z.
34. Ren X, Wen W, Fan X, *et al.* COVID-19 immune features revealed by a large-scale single-cell transcriptome atlas. *Cell* 2021;**184**(7):1895, e19–913.
35. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. *Science* 2006;**313**(5786):504–7.
36. Blondel VD, Guillaume J-L, Lambiotte R, *et al.* Fast unfolding of communities in large networks. *J Stat Mech Theory Exp* 2008;**2008**(10):P10008.
37. Bauer LG, Leiber C, Schelling B, *et al.* Dip-based deep embedded clustering with k-estimation. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 903–913.
38. Zeng Y, Zhou X, Pan Z, *et al.* A robust and scalable graph neural network for accurate single cell classification. bioRxiv 2021;**23**(2):bbab570.
39. Strehl A, Ghosh J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J Mach Learn Res* 2002;**3**(Dec):583–617.
40. Rand WM. Objective criteria for the evaluation of clustering methods. *J Am Stat Assoc* 1971;**66**(336):846–50.
41. Kuhn HW. The Hungarian method for the assignment problem. *Nav Res Logist Quarterly* 1955;**2**(1–2):83–97.
42. Fowlkes EB, Mallows CL. A method for comparing two hierarchical clusterings. *J Am Stat Assoc* 1983;**78**(383):553–69.
43. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J Comput Appl Math* 1987;**20**: 53–65.
44. Zand M, Ruan J. A completely parameter-free method for graph-based single cell RNA-seq clustering. bioRxiv 2021.
45. Lin P, Troup M, Ho JW. CIDR: ultrafast and accurate clustering through imputation for single-cell RNA-seq data. *Genome Biol* 2017;**18**(1):59. https://doi.org/10.1186/s13059-017-1188-0.
46. Wan S, Kim J, Won KJ. SHARP: hyperfast and accurate processing of single-cell RNA-seq data via ensemble random projection. *Genome Res* 2020;**30**(2):205–13.
47. Yu B, Chen C, Qi R, *et al.* scGMAI: a Gaussian mixture model for clustering single-cell RNA-Seq data based on deep autoencoder. *Brief Bioinform* 2021;**22**(4):bbaa316.
48. Duò A, Robinson MD, Soneson C. A systematic performance evaluation of clustering methods for single-cell RNA-seq data. *F1000Research* 2018;**7**:1141.
49. Jiang L, Chen H, Pinello L, *et al.* GiniClust: detecting rare cell types from single-cell gene expression data with Gini index. *Genome Biol* 2016;**17**(1):1–13.
50. McInnes L, Healy J, Melville J. Umap: uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:180203426 2018.
51. Mereu E, Lafzi A, Moutinho C, *et al.* Benchmarking single-cell RNA-sequencing protocols for cell atlas projects. *Nat Biotechnol* 2020;**38**(6):747–55.