nature machine intelligence

ARTICLES

https://doi.org/10.1038/s42256-019-0037-0

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

Tian Tian[1,3], Ji Wan[2,3], Qi Song[2] and Zhi Wei[1*]

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

Single-cell RNA sequencing (scRNA-seq) can reveal heterogeneity and diversity among cell populations and has helped researchers to better understand complex biological questions[1,2]. 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 recent decades. Many popular methods have been proposed, such as k-means[3], Gaussian mixture models (GMMs)[4] and spectral clustering[5]. However, clustering groups of cells in scRNA-seq datasets is still a statistical and computational challenge. The major problem in clustering scRNA-seq data, compared 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). A missing gene measurement is defined as a dropout event and results in a 'false' zero count observation. The recent advances in sequencing platforms have enabled a dramatic increase in the throughput to thousands of cells[6–9]. These technologies are particularly prone to dropout events due to the relatively shallow sequencing depth per cell[10]. 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 noise, 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 shared nearest-neighbour. Such a strategy can effectively handle high-dimensional data[11]. Much research uses sophisticated techniques that involve iterative clustering. These methods detect relationships between the subtypes, which are further validated by differential gene expression analysis[6,12,13]. DendroSplit[14] is an interpretable clustering framework that uses feature selection to uncover multiple levels of biologically meaningful populations in scRNA-seq data. Wang and colleagues employ multi-kernel learning (SIMLR) for single-cell interpretation[15]. SIMLR, essentially, is a spectral clustering method that 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 and colleagues also present a multi-kernel spectral clustering method, but propose to impose a sparse structure (MPSSC) via L1 penalty[16]. These two very recent methods represent current state-of-the-art clustering approaches for scRNA-seq data. Although these spectral clustering-based methods have shown decent performances, they have two issues. First, they rely on the full graph Laplacian matrix, which is prohibitively expensive to compute and store. Computation and storing of the Laplacian matrix usually have quadratic or super-quadratic complexities in terms of the number of data points (cell numbers here), and decomposition of the matrix requires cubic complexities[17]. These complexities suggest a severe scalability issue for spectral clustering. It is typical to cluster thousands of cells in many scRNA-seq studies, which requires supercomputers with huge memories. For example, a machine with 800 Gb memory was used to cluster thousands of cells in ref. [16]. This restricts the application of spectral-based clustering methods to large scRNA-seq datasets—the typical outputs of modern scRNA-seq platforms—as they usually profile thousands or even tens of thousands of cells simultaneously. Second, spectral clustering methods do not explicitly model the characteristics of scRNA-seq count data such as over-dispersion and zero inflation, 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 (clustering through imputation and dimensionality reduction)[18], scImpute[19], MAGIC[20] and SAVER[21], to name a few, and deep learning approaches DeepImpute[22], DCA (deep count autoencoder)[23] and scScope[24]. CIDR 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

[1]Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA. [2]CuraCloud Corporation, Seattle, WA, USA. [3]Theses authors contributed equally: Tian Tian, Ji Wan. *e-mail: zhiwei@njit.edu

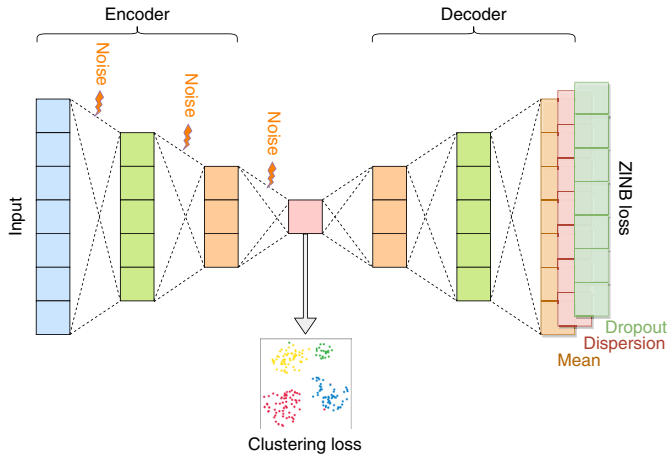

Fig. 1 | Network architecture of scDeepCluster. The encoder and decoder are fully connected neural networks. Clustering loss (KL-divergence) is applied to scatter the embedded points *z*. The ZINB loss has three components—mean, dispersion and dropout—which are estimated by three individual fully connected layers with different activation functions. The random Gaussian noise has been incorporated into the encoder to improve the embedded feature representation.

learn efficient feature representation in an unsupervised manner[25]. 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 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[26], 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[23]. 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, for example, k-means. Such a divided strategy is suboptimal for clustering, as shown in the comparison of our method with the DCA (scScope does not share their program).

Because of the 'curse of dimensionality', clustering performs much better on a small dimensionality than on a high one[27]. DNNs are a natural choice to parameterize a nonlinear transforming function that maps the original high-dimensional data to a small latent space. DNNs have demonstrated theoretical function approximation capability[28] and feature learning properties[29]. Recent studies have illustrated that deep learning can successfully achieve good performance on clustering tasks when applied to image and text datasets[30,31]. Meanwhile, a study demonstrates that DNNs can reduce the dimensions of scRNA-seq data in a supervised manner[32]. In contrast, a recently published deep generative model, scvis[33], is proposed to capture and visualize the low-dimensional structure in scRNA-seq data in an unsupervised manner.

We therefore propose a deep learning clustering method that integrates the ZINB model with clustering loss in a principled way (see Fig. 1 for its architecture). Our method aims to optimize clustering explicitly while performing dimension reduction. In the proposed framework, the nonlinear 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 latent space is performed by clustering with Kullback–Leibler (KL) divergence, as described in the 'deep embedded clustering' (DEC) algorithm[30]. We also introduce the denoising autoencoder technique[34,35] to the ZINB model-based autoencoder, which gives it greater power to learn a robust feature representation of data. We name the method 'single-cell model-based deep embedded clustering' (scDeepCluster; https://github.com/ttgump/scDeepCluster). Using both simulated and real scRNA-seq data, we demonstrate that scDeepCluster brings significant accuracy improvement over competing start-of-the-art clustering methods. Furthermore, we show that scDeepCluster requires less memory and time complexity than competing methods. This appealing computation efficiency makes scDeepCluster more suitable for the analysis of large scRNA-seq data.

Simulation evaluation of scDeepCluster

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 the R package Splatter[36] to simulate scRNA-seq read count data. We simulated 1,500 cells, each with 2,500 genes, forming three groups of the same size (500 cells per group). We used the following three performance metrics to evaluate the consistency between the obtained clustering and the true labels: normalized mutual information (NMI)[37], clustering accuracy (CA) and adjusted Rand index (ARI)[38]. The ranges of NMI and CA are from 0 to 1, while the ARI can yield negative values. The three metrics are statistics of the concordance of two clustering labels; the higher the values, the higher concordance of the clustering. We repeated all experiments 20 times, under the same setting. We compared scDeepCluster with seven competing methods: DCA[23] + k-means, two multi-kernel spectral clustering methods MPSSC[16] and SIMLR[15], CIDR[18], PCA + k-means, scvis[33] + k-means and DEC[30].

We first investigated the performance of the eight methods under different dropout rates, which are defined as the proportion of expressed genes being knocked out of read counts. To do this we varied the midpoint parameter of the dropout logistic function to generate datasets with different dropout rates ($12 \pm 0.3\%$, $17 \pm 0.4\%$, $23 \pm 0.5\%$ and $30 \pm 0.6\%$). The clustering accuracy for NMI (averaged over the 20 repeats within each setting) is presented in Fig. 2a (data for CA and ARI is provided in Supplementary Fig. 1). We note several interesting findings. First, for all methods, accuracy decreases with increasing dropout rate, confirming our speculation that dropout events make clustering challenging. In particular, the performance of PCA + k-means and DEC decreased dramatically with increasing dropout rates, but the decreasing of DCA is not so pronounced. The improvement with the DCA confirms that special modelling of the characteristics of scRNA-seq data can enhance the standard DEC model. Second, scDeepCluster significantly outperformed the other methods consistently under all dropout rate settings ($p < 0.01$, 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 even then remained decent (NMI ≈ 0.9). In contrast, the performances of the other 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 the clustering performance but also makes its performance more stable (Supplementary Fig. 2).

We next evaluated the performances under different clustering signal strengths. To do this we varied fold change levels of gene expression between cell types and groups, which can be controlled by adjusting the variance parameter sigma in the log-normal

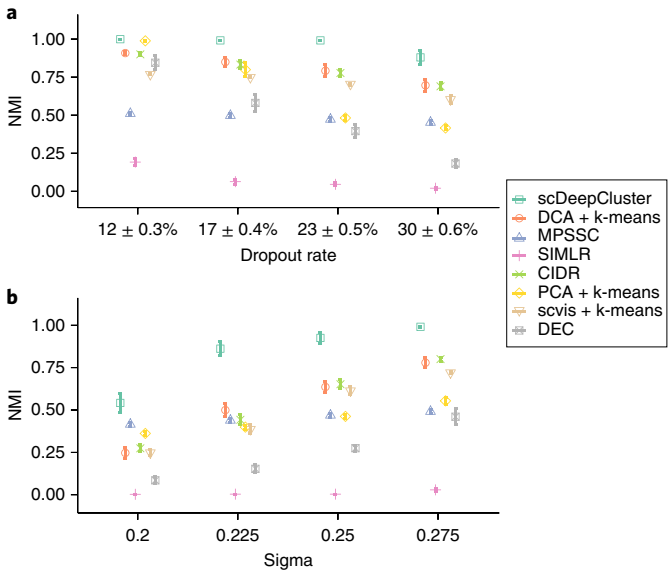

Fig. 2 | Simulation on evaluation. a,**b**, Clustering performance, measured by NMI of scDeepCluster, DCA + k-means, MPSSC, SIMLR, CIDR, PCA + k-means, scvis + k-means and DEC on simulated data with various dropout rates (**a**) and various clustering signal strengths (**b**) (the larger the sigma value, 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.

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. Their performances are presented in Fig. 2b and Supplementary Fig. 3. The t-SNE[39] plots of these simulated datasets confirm that the clustering signal strength increases with the sigma value (Supplementary Fig. 4). We can see, again, that scDeepCluster outperforms the other methods significantly under all different clustering signal strengths ($p < 0.01$, one-sided paired t-test). Not surprisingly, the clustering performance of the different methods generally improved with increasing signals. Of note, the performance of scDeepCluster improved dramatically as the signal increased above a weak signal (sigma = 0.2), quickly reaching good clustering (NMI ≈ 0.86) when sigma increased to 0.225. The improvement of the MPSSC with increasing sigma was steady but very small, indicating its lack of efficiency in exploiting the increased signal. The SIMLR failed (NMI close to 0) for all signals. The DCA, PCA, scvis and DEC also showed improvement with 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 (Supplementary Fig. 5).

Finally, we investigated the performance of scDeepCluster with three groups with imbalanced sample sizes. Following ref. [30], we generated different imbalance levels by varying the minimum retention rate r_{min} (Supplementary Notes) when allocating the 1,500 cells to the three groups. In short, the smaller the r_{min}, the more substantial the imbalance, with the expected sample size of the largest group being $1/r_{min}$ times as large as the smallest. Fixing the dropout rate at 17% and setting sigma = 0.4 (default values recommended by Splatter), we varied the minimum retention rate from 0.1 to 0.9.

Table 1 | Summary of four real scRNA-seq datasets

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

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

As shown in Supplementary Fig. 6, scDeepCluster's performance is fairly robust. It retains perfect clustering concordance (NMI ≈ 1) until r_{min} decreases to 0.3, and its performance remains decent (NMI > 0.85) even when the imbalance is extreme ($r_{min} = 0.1$).

Application to real data

We applied scDeepCluster to four real scRNA-seq datasets to demonstrate its performance. The four datasets were generated from four representative sequencing platforms: PBMC 4k cells from the 10X genomics platform (10X PBMC)[7], mouse embryonic stem cells from a droplet barcoding platform (mouse ES cells)[8], mouse bladder cells from the Microwell-seq platform (mouse bladder cells)[9] and worm neuron cells from the sci-RNA-seq platform (worm neuron cells)[40], as summarized in Table 1. The four datasets, respectively, had 4,271, 2,717, 2,746 and 4,186 cells per sample, with 16,449, 24,046, 19,079 and 11,955 genes after pre-processing, and form 8, 4, 16 and 10 groups per cluster. A detailed description of the datasets is provided in the 'Real data' section of the Methods. The spectral-based clustering methods (MPSSC, SIMLR) were imposed with quadratic space complexity, and we failed to run them with even large memory (for example, 256G). To make a comparison, we randomly sampled 2,100 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 the clustering performance are visualized in Fig. 3a. 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[39] plots (Supplementary Fig. 7). The increased performance after the k-means initialization is also observed on the four real datasets.

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 applied t-SNE to visualize the final embedded points in the two-dimensional (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. Figure 3b shows that scDeepCluster separates the mouse ES cells of four different leukaemia inhibitory factor (LIF) withdrawal intervals (0, 2, 4 and 7 days, highlighted in different colours) well, with only a few red samples mingled with the blue, and a few green ones with the yellow. In contrast, the green and yellow samples were mixed together with the other competing methods. Similar observations were made for the other three datasets, where cells of the same type were separated well in the low-dimensional embedded representation of scDeepCluster, with only some outliers, and much better than the competing methods. 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 therefore added for evaluation two

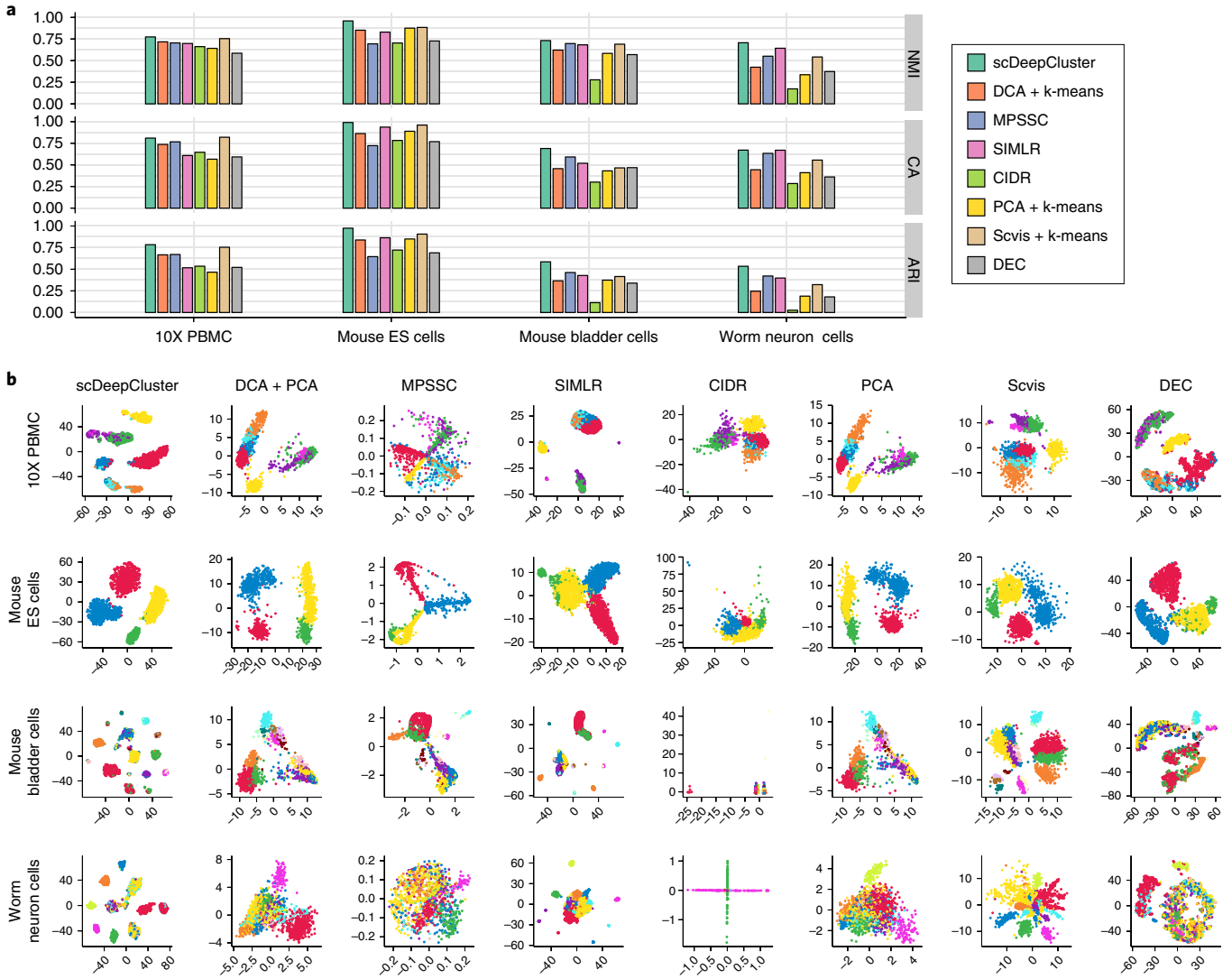

Fig. 3 | Benchmark results on four real scRNA-seq datasets with true labels. a, Comparison of clustering performances of scDeepCluster, DCA + k-means, MPSSC, SIMLR, CIDR, PCA + k-means, scvis + k-means and DEC, measured by NMI, CA and ARI. The larger value means more concordance between the predicted 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 colours of the points represent the true labels. No method uses the true label information.

additional real data sets with 27,000 and 68,000 cells, respectively (Supplementary Table 1). SIMLR and MPSSC failed to run over these two large datasets. We observed similar superiority of scDeepCluster over the other competing methods (Supplementary Fig. 8).

Scalability of scDeepCluster

Practitioners are now being confronted with increasing numbers of cells profiled in scRNA-seq experiments. It has thus become essential to establish an analytic method capable of handling large datasets. To assess the scalability of scDeepCluster, we summarized its running time on datasets with various sample sizes. We simulated a large dataset of 100,000 cells with 3,000 genes, with 10 groups in these cells, and down-sampled this dataset from 5,000 to 100,000 cells (5,000, 7,500, 10,000, 25,000, 50,000, 75,000, 100,000). Figure 4a reports the running time for the pre-training and clustering stages of scDeepCluster on these down-sampled datasets . We can see that, unlike 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 the analysis of large scRNA-seq datasets. Furthermore, we found that the clustering performance of scDeepCluster is quite robust to varying sample sizes (Fig. 4b).

Discussion and conclusion

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[30], we may employ an elbow-method strategy by introducing a metric—generalizability (G, Supplementary Notes). Briefly, we split the data into training and validation sets and set G to be the ratio between the training and validating clustering losses. We computed G under various numbers of clusters k. We observed a sharp drop in G when k became greater than the optimal cluster number, which indicated that G could be used as a good metric for estimating the optimal number of clusters (Supplementary Fig. 12). Furthermore, during the clustering stage, we employ the reconstruction loss function in our autoencoder, as

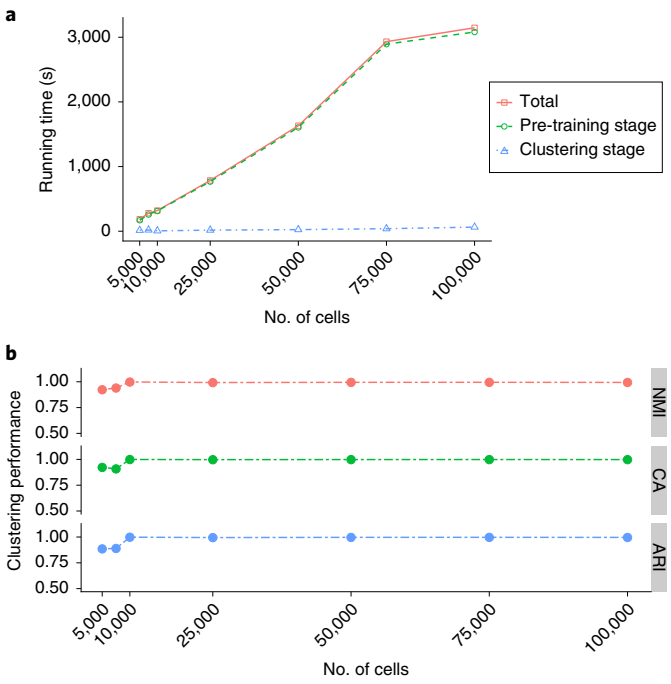

Fig. 4 | Applying scDeepCluster on various down-sampled simulated data. a, Running times of the pre-training stage, the clustering stage and the total scDeepCluster process for different sample sizes. **b**, Clustering performance of scDeepCluster on different sample sizes measured by NMI, CA and ARI. The results are obtained on Nvidia Tesla P100 (16G).

a data-dependent regularization term. Such a regularization term can help to prevent the deep embedding function from overfitting[31]. Generally, clustering methods can be divided into two subcategories: discriminative clustering algorithms (for example, spectral clustering) and generative clustering algorithms (for example, our method). It has been argued that training discriminative models can suffer from overfitting[41]. Our empirical experiments also showed that the spectral clustering model SIMLR mistakenly estimated the number of clusters in the simulated datasets (Supplementary Fig. 13).

In conclusion, we have proposed scDeepCluster—a model-based deep learning approach 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-the-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 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.

Methods

Read count data pre-processing. Raw scRAN-seq read count data are pre-processed by the Python package SCANPY[42]. 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 we denote the library size (number of total read counts) of cell i as s_i, then the size factor of cell i is s_i/median(s). The last step is to take the log transform and scale of the read counts, so that count values follow unit variance and zero mean.
The pre-processed read count matrix is treated as the input for our denoising ZINB model-based autoencoder.

Denoising ZINB model-based autoencoder. The autoencoder is a type of artificial neural network used to learn efficient feature representation in an unsupervised manner[25]. The denoising autoencoder is an autoencoder that receives corrupted data points as input and is trained to predict the original uncorrupted data points as its output[34]. The autoencoder usually has a low-dimensional bottleneck layer to learn a latent feature representation. The denoising autoencoder is proved to be more powerful in learning a robust representation of data, because 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 map the input of read counts to an embedded space to carry out clustering. In practice, we first corrupt the input with random Gaussian noise, then construct the autoencoder with regular fully connected layers. Formally, input X is corrupted by noise

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

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[35]. We define the encoder function as $z = f_W(X^{\text{corrupt}})$ and the decoder function $X' = g_{W'}(z)$. The encoder and decoder functions are both fully connected neural networks with rectifier activation[43]. Here W and W' are the learned weights of the functions. The learning process of the denoising autoencoder minimizes the loss function

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

where L is the loss function.

To capture the characters of scRNA-seq data, we apply a ZINB model-based autoencoder[23] instead of a regular autoencoder, which is trained to attempt to reconstruct its input. Unlike the regular autoencoder, the loss function of the ZINB model-based autoencoder is the likelihood of a ZINB distribution. 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 the point mass of probability at zero (the probability of dropout events):

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

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

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

$$M = \text{diag}(s_i) \times \exp(W_\mu D)$$

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

$$\Pi = \text{sigmoid}(W_\pi D)$$

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 'data pre-process' part and are included as an independent input to the deep learning model. The activation function chosen for mean and dispersion is exponential because the mean and dispersion parameters are non-negative values, while the activation function for the additional coefficient π is sigmoid, and represents the dropout probability. Dropout probability is in the interval of 0–1, so sigmoid is a suitable choice of activation function. The loss function of the ZINB model-based autoencoder is the negative log of the ZINB likelihood:

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

Deep embedded clustering with local structure preservation. The clustering stage follows the deep embedded clustering[31,30]. 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 nonlinear mapping $f_W \sim x_i \to z_i$, 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.

To capture the characters of single-cell RNA-seq data, we apply the denoising ZINB model-based autoencoder to learn the nonlinear mapping of X to Z. The clustering algorithm is the same as in ref. [30], 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

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

where q_{ij} is the soft label of embedded point z_i, which is defined as the similarity between z_i and cluster centre μ_j measured by Student's t-distribution[39,44]:

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

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[45], because the target distribution P is defined based on Q.

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

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

where L_{ZINB} and L_c are the ZINB loss functions of denoising the 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 ref. [31], this form of loss has the advantage that it can preserve the local structure of the data-generating distribution.

Optimization. The DNN and cluster centres $\{\mu_j\}$ 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 centre 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 centre 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

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

The encoder's weight W is updated by

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

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 the percentage of points changing cluster assignment between two consecutive iterations is less than a certain tolerance threshold. Note that the above procedure is described in refs. [30,31].

Implementation. scDeepCluster is implemented in Python 3 using Keras (https://github.com/keras-team/keras) with a TensorFlow[46] backend. The random Gaussian noise is implemented by the Keras layer 'GaussianNoise', and the noise level (argument *'stddev'*) is set to 2.5 (the effects of noise are shown in Supplementary Fig. 9). 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[47,48], setting the initial learning rate $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. The optimizer for the clustering stage is Adadelta[49] with setting $lr = 1.0$, $rho = 0.95$. The sizes of the hidden fully connected layer in the encoder are set to (256, 64), the decoder is the reverse of the encoder, and the bottleneck layer (the latent space) has a size of 32. The choice of coefficient γ is 1 (choice of γ in Supplementary Fig. 10). The batch size for 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).

Competing methods. DCA[23], SIMLR[15], MPSSC[16], CIDR[18], PCA + k-means, scvis[33] and DEC[30] are used as competing methods. DCA is conducted directly by using the authors' API functions (https://github.com/theislab/dca). DCA is not designed for clustering. So, 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 2D space by principal component analysis (PCA). k-means clustering was conducted on the projected 2D space. This method is called 'DCA + k-means'. We pre-process the read count matrix then use the pre-processed 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 2D space directly, followed by k-means clustering. We follow the steps described by the authors for CIDR (https://github.com/VCCRI/CIDR). 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 the CIDR dissimilarity matrix, reducing the dimensionality and clustering. We use the first two principal components computed by CIDR to show the latent representations. The scvis is a variational autoencoder[50] based model used to capture the low-dimensional representation of scRNA-seq data. We use scvis to reduce scRNA-seq data to 2D space then apply k-means clustering. For scvis, we follow the pre-process steps described by the authors: the expression of each gene is quantified as $\log_2(\mathrm{CPM}/10+1)$, where 'CPM' stands for 'counts per million'. Next, the data are projected to a 100-dimensional space by PCA and used as input for scvis. DEC (https://github.com/XifengGuo/DEC-keras) uses the same inputs as scDeepCluster: the raw count matrix is library-size normalized, log transformed, scaled and centred. The hyperparameters in DEC remain the same as the authors' originals (for example, the sizes of the hidden layers are 500, 500, 2,000, 10).

Evaluation metrics. All clustering results are measured by NMI[51], CA and 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

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

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

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

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 with the Hungarian algorithm[52].

The Rand index[53] is a simple measure of agreement between two cluster assignments U and V. The ARI corrects for the lack of a constant value of the Rand index when the cluster assignments are selected randomly[38]. The ARI is calculated using four quantities. Specifically, we define (1) the number of pairs of two objects in the same group in both U and V, (2) the number of pairs of two objects in different groups in both U and V, (3) the number of pairs of two objects in the same group in U but in different groups in V, (4) 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

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

Data simulation. Simulated data are generated by the Splatter R package[36]. The R function splatSimulate is used to simulate the scRNA-seq count data. In all settings, we simulated three cell groups, 1,500 cells of 2,500 genes, and 20 datasets were repeatedly generated for each setting.

For the simulation of various dropout rates, we set the parameter *dropout.shape* = −1, 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) and *de.fracScale* = 0.3; the other parameters are set to default values. 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 perform 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

$$\text{Dropout rate} = \frac{\text{No. of 0 counts in raw counts} - \text{no. of 0 counts in true counts}}{\text{No. of counts} > 0 \text{ in true counts}}$$

To simulate datasets with various signal strengths, we vary the parameter *de.fracScale* in 0.2, 0.225, 0.25, 0.275 and fix *dropout.shape* = −1, *dropout.mid* = 0; the other parameters are set to default values. In Splatter, the parameter *de.fracScale* is the sigma parameter of a log-normal distribution that controls the multiplicative differential expression factors. The effects of different *de.fracScale* settings are visualized in Supplementary Fig. 4.

We simulated a large dataset of 100,000 cells by 3,000 genes to evaluate the running time of scDeepCluster. The 100,000 cells are divided into 10 groups. The parameters are *dropout.shape* = −1, and we vary *dropout.mid* = 0, *de.fracScale* = 0.3. We down-sampled the simulated large dataset to different cell numbers: 5,000, 7,500, 10,000, 25,000, 50,000, 75,000 and 100,000. Down-sampling did not drop any group.

Real data. The 10X PBMC dataset (4K PBMCs from a healthy donor) was provided by the 10X scRNA-seq platform[7], which profiles the transcriptome of the peripheral blood mononuclear cells (PBMCs) from a healthy donor. The total number of cells was ~4,000. PBMC 4k data were downloaded from the website of 10X genomics (https://support.10xgenomics.com/single-cell-gene-expression/datasets/2.1.0/pbmc4k). We downloaded filtered gene/cell matrix and cell labels identified by graph-based clustering (for the method description see https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/output/analysis).

The mouse ES cells dataset[8] was downloaded from GSE65525. The transcriptomes were profiled by the droplet-microfluidic approach for parallel barcoding. Allon and colleagues analysed the heterogeneous onset of differentiation of mouse embryonic stem cells after LIF withdrawal[8]. We downloaded the read count matrices of mouse ES cells sample 1, mouse ES cells LIF - 2 days, mouse ES cells LIF - 4 days and mouse ES cells LIF - 7 days, and put all cells together. The labels of cells are defined as the intervals after LIF withdrawal.

The mouse bladder cells dataset of the Mouse Cell Atlas project[9] was provided by the authors (https://figshare.com/s/865e694ad06d5857db4b). We downloaded the digital expression matrix, with the batch gene background removed, of all 400,000 single cells sorted by tissues and the table of cell assignments. The authors identified the cell types (and describe the method in ref. [9]). From the raw count matrix, we selected cells from bladder tissue.

The worm neuron cells dataset was profiled by sci-RNA-seq (single-cell combinatorial indexing RNA sequencing)[40]. 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/). We select the subset of neural cells and removed the cells with the label 'Unclassified neurons'. We thus obtained 4,186 neural cells.

The performance of scDeepCluster on the full four scRNA-seq data is provided in Supplementary Fig. 11.

Data availability

The scRNA-seq data that support the findings of this study are available in GitHub: https://github.com/ttgump/scDeepCluster/tree/master/scRNA-seq%20data.

Code availability

The source code, weights of trained models and the real scRNA-seq data used for experiments of scDeepCluster are available in GitHub: https://github.com/ttgump/scDeepCluster.

Received: 1 October 2018; Accepted: 8 March 2019;
Published online: 9 April 2019

References

1. Shapiro, E., Biezuner, T. & Linnarsson, S. Single-cell sequencing-based technologies will revolutionize whole-organism science. *Nat. Rev. Genet.* **14**, 618–630 (2013).
2. Kolodziejczyk, A. A., Kim, J. K., Svensson, V., Marioni, J. C. & Teichmann, S. A. The technology and biology of single-cell RNA sequencing. *Mol. Cell* **58**, 610–620 (2015).
3. MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability* Vol. 1, 281–297 (Univ. of California Press, 1967).
4. Bishop, C. *Pattern Recognition and Machine Learning* (Springer, 2006).
5. von Luxburg, U. A tutorial on spectral clustering. *Stat. Comput.* **17**, 395–416 (2007).
6. Macosko, E. Z. et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell* **161**, 1202–1214 (2015).
7. Zheng, G. X. et al. Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.* **8**, 14049 (2017).
8. Klein, A. M. et al. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell* **161**, 1187–1201 (2015).
9. Han, X. et al. Mapping the mouse cell atlas by Microwell-seq. *Cell* **172**, 1091–1107 (2018).
10. Angerer, P. et al. Single cells make big data: new challenges and opportunities in transcriptomics. *Curr. Opin. Syst. Biol.* **4**, 85–91 (2017).
11. Xu, C. & Su, Z. Identification of cell types from single-cell transcriptomes using a novel clustering method. *Bioinformatics* **31**, 1974–1980 (2015).
12. Zeisel, A. et al. Brain structure. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science* **347**, 1138–1142 (2015).
13. Tasic, B. et al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nat. Neurosci.* **19**, 335–346 (2016).
14. Zhang, J. M., Fan, J., Fan, H. C., Rosenfeld, D. & Tse, D. N. An interpretable framework for clustering single-cell RNA-seq datasets. *BMC Bioinformatics* **19**, 93 (2018).
15. Wang, B., Zhu, J., Pierson, E., Ramazzotti, D. & Batzoglou, S. Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nat. Methods* **14**, 414–416 (2017).
16. Park, S. & Zhao, H. Spectral clustering based on learning similarity matrix. *Bioinformatics* **34**, 2069–2076 (2018).
17. Jianbo, S. & Malik, J. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 888–905 (2000).
18. Lin, P., Troup, M. & Ho, J. W. CIDR: ultrafast and accurate clustering through imputation for single-cell RNA-seq data. *Genome Biol.* **18**, 59 (2017).
19. Li, W. V. & Li, J. J. An accurate and robust imputation method scImpute for single-cell RNA-seq data. *Nat. Commun.* **9**, 997 (2018).
20. van Dijk, D. et al. Recovering gene interactions from single-cell data using data diffusion. *Cell* **174**, 716–729 (2018).
21. Huang, M. et al. SAVER: gene expression recovery for single-cell RNA sequencing. *Nat. Methods* **15**, 539–542 (2018).
22. Arisdakessian, C., Poirion, O., Yunits, B., Zhu, X. & Garmire, L. DeepImpute: an accurate, fast and scalable deep neural network method to impute single-cell RNA-seq data. Preprint at https://doi.org/10.1101/353607 (2018).
23. Eraslan, G., Simon, L. M., Mircea, M., Mueller, N. S. & Theis, F. J. Single-cell RNA-seq denoising using a deep count autoencoder. *Nat. Commun.* **10**, 390 (2019).
24. Deng, Y., Bao, F., Dai, Q., Wu, L. & Altschuler, S. Massive single-cell RNA-seq analysis and imputation via deep learning. Preprint at https://doi.org/10.1101/315556 (2018).
25. Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science* **313**, 504–507 (2006).
26. Chen, J. et al. An omnibus test for differential distribution analysis of microbiome sequencing data. *Bioinformatics* **34**, 643–651 (2018).
27. Bellman, R. E. *Adaptive Control Processes: A Guided Tour* (Princeton Univ. Press, 1961).
28. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **4**, 251–257 (1991).
29. Bengio, Y., Courville, A. & Vincent, P. Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1798–1828 (2013).
30. Xie, J., Girshick, R. & Farhadi, A. Unsupervised deep embedding for clustering analysis. In *Proc. 33rd International Conference on Machine Learning* 478–487 (2016).
31. Guo, X., Gao, L., Liu, X. & Yin, J. Improved deep embedded clustering with local structure preservation. In *Proc. 26th International Joint Conference on Artificial Intelligence* 1753–1759 (2017).
32. Lin, C., Jain, S., Kim, H. & Bar-Joseph, Z. Using neural networks for reducing the dimensions of single-cell RNA-seq data. *Nucleic Acids Res.* **45**, e156 (2017).

33. Ding, J., Condon, A. & Shah, S. P. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nat. Commun.* **9**, 2002 (2018).
34. Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proc. 25th International Conference on Machine Learning* 1096–1103 (2008).
35. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.-A. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* **11**, 3371–3408 (2010).
36. Zappia, L., Phipson, B. & Oshlack, A. Splatter: simulation of single-cell RNA sequencing data. *Genome Biol.* **18**, 174 (2017).
37. Strehl, A. & Ghosh, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* **3**, 583–617 (2003).
38. Hubert, L. & Arabie, P. Comparing partitions. *J. Classif.* **2**, 193–218 (1985).
39. van der Maaten, L. & Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).
40. Cao, J. et al. Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science* **357**, 661–667 (2017).
41. Dizaji, K. G., Herandi, A., Deng, C., Cai, W. & Huang, H. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proc. IEEE International Conference on Computer Vision* 5747–5756 (IEEE, 2017).
42. Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.* **19**, 15 (2018).
43. Nair, V. & Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. In *Proc. 27th International Conference on Machine Learning* 807–814 (Omnipress, 2010).
44. Maaten, L. Learning a parametric embedding by preserving local structure. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics* Vol. 5 (eds Van Dyk, D. & Welling M.) 384–391 (PMLR, 2009).
45. Nigam, K. & Ghani, R. Analyzing the effectiveness and applicability of co-training. In *Proc. Ninth International Conference on Information and Knowledge Management* Vol. 5, 86–93 (2000).
46. Abadi, M. et al. TensorFlow: a system for large-scale machine learning. In *Proc. 12th USENIX Conference on Operating Systems Design and Implementation* 265–283 (USENIX Association, 2016).
47. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. Preprint at https://arxiv.org/abs/1412.6980 (2014).
48. Reddi, S. J., Kale, S. & Kumar, S. On the convergence of adam and beyond. In *Sixth International Conference on Learning Representations* (2018).
49. Zeiler, M. D. ADADELTA: an adaptive learning rate method. Preprint at https://arxiv.org/abs/1212.5701 (2012).
50. Kingma, D. P. & Welling, M. Stochastic gradient VB and the variational auto-encoder. In *Second International Conference on Learning Representations* (2014).
51. Strehl, A. & Ghosh, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* **3**, 583–617 (2003).
52. Kuhn, H. W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **2**, 83–97 (1955).
53. Rand, W. M. Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.* **66**, 846–850 (1971).

Author contributions

Z.W. and Q.S. conceived and supervised the project. Z.W. led the study. T.T. designed the methods and conducted the experiments with input from J.W. T.T., J.W. and Z.W. wrote the manuscript. All authors approved the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s42256-019-0037-0.

Reprints and permissions information is available at www.nature.com/reprints.

Correspondence and requests for materials should be addressed to Z.W.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.