Published in final edited form as:
Nat Methods. 2019 April ; 16(4): 311–314. doi:10.1038/s41592-019-0353-7.

Scalable analysis of cell type composition from single-cell transcriptomics using deep recurrent learning

Yue Deng[1,*], **Feng Bao**[2,*], **Qionghai Dai**[2], **Lani F. Wu**[1,#], **Steven J. Altschuler**[1,#]

[1]Department of Pharmaceutical Chemistry, University of California, San Francisco, San Francisco, CA 94158, USA.

[2]Department of Automation, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing, 100084, P. R. China.

Abstract

Recent advances in large-scale single cell RNA-seq enable fine-grained characterization of phenotypically distinct cellular states within heterogeneous tissues. We present scScope, a scalable deep-learning based approach that can accurately and rapidly identify cell-type composition from millions of noisy single-cell gene-expression profiles.

Single-cell RNA-seq (scRNA-seq) provides high-resolution dissection of complex biological systems, including identification of rare cell subpopulations in heterogeneous tissues, elucidation of cell-developmental programs, and characterization of cellular responses to perturbations[1–3]. Recent scRNA-seq experimental platforms[4–6] have enabled interrogation of millions of cells, which offers an unprecedented resolution at which to dissect cell-type compositions.

These advances led to two acute challenges. First, single-cell transcriptomics is susceptible to amplification noise and dropout events[7, 8], which become more pronounced as tradeoffs are made to sequence larger numbers of cells. Second, computational memory and/or speed restrictions may render analytical packages[7–12] poorly scalable for large datasets, including all cells and measured genes.

To extract informative representations from extremely noisy, massive, high-dimensional scRNA profiles, we developed scScope, a deep-learning based software package (Fig. 1a). scScope utilizes a recurrent network layer to iteratively perform imputations on zero-valued entries of input scRNA-seq data (Methods). scScope's architecture allows imputed output to be iteratively improved through a selected number of recurrent steps (T). We note that for T = 1, the architecture reduces to a standard autoencoder (AE)[13]. In one joint framework,

[#]To whom correspondence should be addressed: lani.wu@ucsf.edu, steven.altschuler@ucsf.edu.
Author Contributions
Y.D., F.B. and Q.D. developed the deep learning algorithms. Y.D. and F.B. conducted experimental analysis on both simulated and biological datasets. The manuscript was written by Y.D., F.B., L.F.W. and S.J.A. All authors read and approved the manuscript.
[*]These authors contributed equally to this work.

Competing Financial Interests Statement
The authors declare no competing interests.

scScope conducts batch effect removal, cellular-feature learning, dropout imputation and parallelized training (when multiple GPUs are available) (Supplementary Fig. 1).

We evaluated scScope on its scalability, ability to identify cell subpopulations, impute dropout gene information and correct batch effects, using datasets with varying degrees of size, complexity and prior biological annotation. We compared scScope performance with four "non-deep" methods (PCA, MAGIC[11], ZINB-WaVE[8] and SIMILR[9]) and three "deep" learning models (Autoencoder (AE), scVI[14] and DCA[15]) (Methods and Supplementary Table 1). The AE was implemented via scScope with T = 1 (Methods) and, for all comparisons, unless otherwise stated, scVI, DCA, AE and scScope were run on a single GPU using default parameters (Methods, Supplemental Table 2).

We first tested the scalability and training speed of scScope on a mouse brain dataset, which contained 1.3M cells, using 1000 highly variable genes (Fig 1b,Methods). As expected, the generic machine-learning tool PCA was the fastest. scScope was able to complete its analysis of the full dataset in under an hour, which was comparable to the AE, and this runtime was significantly dropped by using the option for multiple GPU training. The non-deep single-cell software packages were unable to scale beyond 100K cells, and the deep approaches, while able to scale to 1M cells, required at least seven times more computing time than scScope. Thus, scScope offers a scalable and highly efficient approach for analyzing large scRNA-seq datasets.

To calibrate the accuracy of scScope on simulated datasets, we made use of two third-party packages for generating scRNA-seq data (Methods and Supplementary Table 3). First, we used Splatter[16] to generate moderate-sized datasets of varying sparsity levels (percentage of 0-valued genes), containing 2K scRNA-seq profiles with 500 genes, and three underlying subpopulations. In terms of discovering these underlying subpopulations, scScope performs similarly to other approaches when there are only minor dropout effects, but shows a large advantage in accuracy as dropout rates increase to realistic ranges observed in biological data[4, 6] (Fig. 1c). In terms of imputation error, at low sparsity (<50%) scScope and MAGIC outperformed all other methods, though at high sparsity scScope outperformed all other approaches (Supplementary Fig. 2). We found for scScope that a recurrence of T = 2 (its default value) provides a good tradeoff between speed and accuracy (Supplementary Fig. 3).

Second, we used the simulation framework in SIMLR to generate massive, heterogeneous datasets of varying sparsity levels containing: 1M scRNA-seq profiles with 500 genes, and 50 underlying subpopulations. The deep approaches were able to operate over the full datasets, while the non-deep single-cell packages required down-sampled training strategies (Methods). scScope performed well, particularly at high sparsity levels (Supplementary Fig. 4). An increasingly important task for scRNA-seq profiling approaches is to identify rare cell subpopulations within large-scale data. As might be expected, methods that did not require down sampling were better able to detect rare cell subpopulations. Overall, scScope performed reasonably well on this challenging task (Fig. 1d, Methods). Our calibration suggested that scScope can efficiently identify cell subpopulations from complex datasets with high dropout rates, large numbers of subpopulations and rare cell types.

We next evaluated scScope on four experimental single-cell RNA datasets containing varying degrees of biological "ground truth". These datasets were used to test the ability of scScope to: remove batch effects (Fig. 2a; lung tissue), recover dropout genes (Fig. 2b; CBMC dataset), identify minor subpopulations (Supplementary Fig. 5; retina dataset), and test clustering accuracy for a large dataset with varying numbers of analyzed genes (Fig. 2c; mouse cell atlas).

To test the ability to remove batch effects, we made use of the lung tissue dataset (part of the mouse cell atlas), which contained ~7K scRNA-seq profiles obtained from three different batches and used the 1,000 most variable genes (Methods). As with the two other software packages that offer batch correction, scScope performed well (Fig. 2a, top) without compromising its fast runtime (Fig. 2a, bottom).

To investigate how imputation accuracy depends on gene expression level, we made use of the cord blood mononuclear cell (CBMC) dataset, with ~8K scRNA-seq profiles and 1,000 most variable genes (Methods). We sequentially simulated dropouts for genes[14] (Methods) and reported results based on octile of expression ranking (Fig. 2b). For reconstructing small count values, MAGIC and scVI performed well, while for large count values DCA worked well. scScope showed small imputation errors consistently across the entire range of expression.

To test the ability to identify minor subpopulations in biological data, we made use of the mouse retina dataset, with 44K cells and 384 most variable genes (Methods). scScope automatically identified the most similar clustering (number and assignment) to the 39 cell subpopulations identified in the original study (Supplementary Fig. 5a). We annotated the clusters to cell types based on gene markers reported in the original study (Supplementary Table 4 and Methods). Clusters identified by scScope showed the most statistically significant enrichment of specific cell-type markers (larger fold-changes) (Supplementary Fig. 5b) and were highly consistent with previous, microscopy-validated estimates of cell-type composition and proportion[18] either excluding (Supplementary Fig. 5c) or including (Supplementary Fig. 5d) the major cell type, rod cells. Additionally, when analyzing "pure" cell types within the Retina dataset, scScope maintained the underlying simple subpopulation structure and achieved high reconstruction accuracy (Supplementary Table 5).

To test the ability to analyze increasing numbers of genes, we made use of the mouse cell atlas, with 400K cells sampled from 51 tissues. Only the deep learning algorithms were able to scale to these data sizes. To perform automatic identification of subpopulations on large datasets, we designed a scalable clustering approach (Methods and Supplementary Fig. 6). We used the 51 known tissue types to assess accuracy of the clustering results. For 1,000 and 2,000 genes all three approaches performed similarly (Fig. 2c). Only scScope was able to scale to levels above 10,000 genes, by using its option for scalable memory allocation (Methods).

Finally, we applied scScope to investigate novel biology in datasets. We focused on the ability to reveal changes in cell-type composition under perturbed conditions

(Supplementary Fig. 7; intestinal dataset) and scale to large datasets and reveal new subpopulations (Fig. 3; brain dataset,Methods).

We applied scScope to the intestinal dataset, with ~10K mouse intestinal cells obtained from different infection conditions and the 1,000 most variable genes (Supplementary Fig. 7a and Methods). In the original study, enterocytes were identified as a single cluster. Interestingly, scScope subdivided this cell type into four subpopulations: differential expression of markers delineated distal *vs*. proximal enterocyte subpopulations; expression levels of these markers delineated immature *vs*. mature subtypes (Supplementary Fig. 7b and Supplementary Fig. 8). These refined enterocyte subpopulations suggested predictions about specific cell-type response to infection. For example, the pro-inflammatory gene *Saa1* was overexpressed during both *Salmonella* and *H. polygyrus* (day 10) infections in distal enterocytes, but not in proximal enterocytes (Supplementary Fig. 7c and Supplementary Table 6). This geographic pattern of *Saa1* expression is known for *Salmonella* infection, but is a novel prediction for *H. polygyrus* infection. Thus, scScope can be used to rapidly explore scRNA-seq data across perturbed conditions to predict novel gene function and identify new cell subtypes.

The 1.3M cells in the brain dataset were obtained from multiple brain regions, including the cortex, hippocampus and ventricular zones, of two embryonic mice. Here, we made use of scScope's ability to rapidly explore this large dataset via its multi-GPU learning functionality with limited training iterations (Methods, Supplementary Table 2). scScope automatically identified 36 clusters, and we assigned each cluster to one of three major cell types based on criteria from the Allen Brain Atlas (http://brain-map.org) (Fig. 3, Supplementary Table 7 and Methods): glutamatergic neurons, GABAergic neurons and non-neuronal cells. The proportions of neurons and non-neurons identified by scScope were consistent with cell proportions reported by a previous brain study[19] (Fig. 3a). We investigated whether we could identify biological meaning for the 36 clusters, some of which contained fewer than 1,000 cells. Satisfyingly, by comparing our top overexpressed genes with known cell-type markers[19, 20] (Supplementary Table 8), we were able to assign two thirds of the clusters to known cell types (Fig. 3b). Thus, scScope is able to rapidly, automatically and directly identify *bona fide*, rare cell types from large and complex biological datasets.

Taken together, scScope offers a platform that will help keep pace with the rapid advances in scRNA-seq, enabling rapid exploration of heterogeneous biological states within large and noisy datasets of single-cell transcriptional profiles.

Online Methods

1. scScope model and training.

Architecture—The architecture of the scScope network has four modules (Fig. 1a). The parameters in these layers are learned from data in an end-to-end manner through optimization. We note that scScope is flexible in terms of normalizing and scaling of input data, as long as the input values are non-negative.

Batch Effect Correction: scScope offers the option to correct for batch effects, inspired by a previously developed approach[8]. The batch correction layer $f_B(\cdot)$ is given by:

$$x_c = f_B(\tilde{x}_c) = r(\tilde{x}_c - Bu_c)$$

Here, we denote: the input single-cell profile by $\tilde{x}_c \in \mathbb{R}^N$; the number of batches by K; the binary experimental batch effects indicator vector $u_c \in \mathbb{R}^K$ (non-zero entry indicates the batch of $\tilde{x}_C$); and the learnable batch correction matrix as $B \in \mathbb{R}^{N \times K}$. Throughout, $r(\cdot)$ denotes the standard rectified linear unit (ReLU), $r(v) = \max(0, v)$; the ReLU enforces $x_c \geq 0$, which is expected for actual gene count data, and is widely used in deep learning. By default, $u_c = 0$ assuming a single batch.

Encoder: For each cell c, the encoder layer $f_E(\cdot)$ compresses the high-dimensional batch-corrected single-cell expression profile $x_c \in \mathbb{R}^N$ into a low-dimensional, latent representation $h_c \in \mathbb{R}^M$, $M < N$. The encoder layer is given by:

$$h_c = f_E(x_c) = r(W_E x_c + b_E)$$

with learnable parameters $W_E \in \mathbb{R}^{M \times N}$ and $b_E \in \mathbb{R}^M$.

Decoder: The decoder layer $f_D(\cdot)$ decompresses the latent representation h_c to an output $y_c \in \mathbb{R}^N$ of the same dimension as the input single-cell profile, and is given by:

$$y_c = f_D(h_c) = r(W_D h_c + b_D)$$

with learnable parameters $W_D \in \mathbb{R}^{N \times M}$ and $b_D \in \mathbb{R}^N$.

Imputer: We developed a self-correcting layer to impute missing entries, inspired by a previously developed reconstruction approach[21]. To reduce the number of parameters, we transformed the decoder layer output y_c to a p-dimensional latent vector by:

$$u_c = r(W_U y_c + b_U) \in \mathbb{R}^p$$

with learnable parameters $W_U \in \mathbb{R}^{p \times N}$ and $b_U \in \mathbb{R}^p s$, $p < N$ (we set $p = 64$). Then, we performed imputation by:

$$v_c = P_{Z_c}\left[r(W_V u_c + b_V)\right] \in \mathbb{R}^N$$

with learnable parameters $W_V \in \mathbb{R}^{N \times p}$ and $b_V \in \mathbb{R}^N$. Here, $Z_c\left(resp.\bar{Z}_c\right)$ is the set of zero- (*resp.* non-zero-) valued genes in profile x_c of the c-th cell, and the entry-sampling operator P_{Z_C} sets all entries not in Z_c to zero (*i.e.*, for vector r, with entries r_j, $P_{Z_c}\left(r_j\right) = r_j$ if $j \in Z_c$ and $P_{Z_C}\left(r_j\right) = 0, j \notin Z_C$). We used Z_c as we are only interested in imputation for zero-valued genes in the profile x_c.

After obtaining the imputed vector v_c, we calculated the corrected single-cell expression profile: $\hat{x}_c = x_c + v_c$. This corrected single-cell profile $\hat{x}_c$ was then re-sent through the encoder-decoder framework to re-learn an updated latent representation.

Learning the objective function.—The imputation layer imposes a recurrent structure on the scScope network architecture. For clarity of exposition, the recurrent scScope can be unfolded into multiple time steps (Supplementary Fig. 9 shows three steps). Then, the whole recurrent scScope framework can be described as:

$$x_c = f_B\left(\tilde{x}_c\right), h_c^t = f_E\left(\hat{x}_c^t\right) = f_E\left(x_c + v_c^{t-1}\right),\ y_c^t = f_D\left(h_c^t\right),\ v_c^t = f_I\left(y_c^t\right),\ v_c^0 = 0,$$

for iterations $t = 1 \ldots T$. At the first step, the correcting layer's output v_c^0 is set as zero. For $T = 1$, scScope is a standard auto-encoder (AE).

The learning objective for scScope is defined by the pursuit of unsupervised, self-reconstruction (as typically used in AE training):

$$f_B, f_E, f_D, f_I = argmin\, L = \sum_{c=1}^{n} \sum_{t=1}^{T} \left\| P_{\bar{z}_c}\left[y_c^t - x_c\right] \right\|^2$$

The entry-sampling operator $P_{\bar{Z}_c}$ forces loss computation only on non-zero entries of x_c. The parameters in the batch correction layer (f_B), encoder layer (f_E), decoder layer (f_D) and imputation layer (f_I) are all learned by minimizing the above loss function.

Our multiple GPU training strategy is outlined in Supplementary Fig. 1.

Cell subpopulation discovery.—Representations outputted by scScope at each step can be concatenated as a long feature vector, which is easily integrated with any clustering method. We used the graph-based method PhenoGraph[17], as it performs automated robust discovery of subpopulations, as well as determines subpopulation numbers automatically.

1) Graph clustering for moderate-scale data.: We directly applied the PhenoGraph software to datasets of moderate scale. All clustering results were obtained using a Python-implemented PhenoGraph package (version 1.5.2). We followed the suggested setting and considered 30 nearest neighbors when constructing graphs.

2) Scalable graph clustering for large-scale data.: scScope enables the feature learning on millions of cells. However, PhenoGraph is unable to handle millions of cells due to the extreme computational costs (Supplementary Fig. 10) and memory requirements in graph construction. To leverage the power of graph clustering on analyzing these large-scale data, we designed a density down-sampling clustering strategy by combining k-means and PhenoGraph.

In detail, we divided cells into M groups with equal size and performed k-means clustering on each group independently (Supplementary Fig. 6). The whole dataset was split to $M \times K$ clusters and we only input the cluster centroids into PhenoGraph for graph clustering. Finally, each cell was assigned to graph clusters according to the cluster labels of its nearest centroids.

In our implementation on the dataset of 1.3 million mouse brain cells, we took $M = 500$ and $K = 400$, which made it possible to process millions of data in only tens of minutes without loss of accuracy.

3) Scalable memory allocation for analyzing large numbers of genes in large datasets.: For large datasets and gene numbers, scScope implements a scalable memory allocation strategy that allows the dataset to be broken into a smaller number of batches that can be loaded directly into memory. We note that when this option is used, minibatches are only selected from within each batch during training. This option was only used in Fig. 2c for the case of ≥10K genes; here the 400K mouse cell atlas dataset was broken into four batches of size 100K.

2. Methods compared.

All compared methods were run on the same server with Xeon E5 CPU, 64 GB memory, Nvidia Titan X GPU and Ubuntu 14.04 operation system. Further, all comparisons were performed using log transformed input (we observed similar relative performance of the six compared methods above across four different choices of input scaling or normalization methods; Supplementary Fig. 11).

Unless otherwise noted: Software packages were used with their default values. We observed that all methods were reasonably robust to changes in the latent dimension M (e.g. Supplementary Fig. 12), and to avoid an intractable number of possible comparisons, we set M = 50 for all comparisons. For all deep learning methods (AE, DCA, scVI and scScope), we set the learning rate to 10^{-3}, batch size to 64, and epoch = 100.

PCA: PCA is implemented using the Python package scikit-learn v0.18.

Autoencoder: The simple autoencoder (AE) is implemented using scScope with T = 1. Variations of the AE are described in the legend of Supplementary Fig. 13.

Markov Affinity-based Graph Imputation of Cells (MAGIC): The MAGIC algorithm was performed using the python-based package magic[11]. We inputted raw data and employed the library_size_normalize() function provided by the software for all simulated and real data.

Zero-Inflated Negative Binomial-based Wanted Variation Extraction (ZINB-WaVE): We employed the R package zinbwave[8].

Single-cell interpretation via multi-kernel learning (SIMLR): We used the Python implementation of SIMLR[9]. SIMLR needs to take the desired cluster number as input. For our simulated dataset, we input the known cluster numbers. For the retina dataset, where the "true" cluster number is unknown, we set it to 39, which is the cluster number reported in the original study[4].

Deep count autoencoder (DCA): DCA[15] is an unpublished software based on TensorFlow framework. We installed the Python package of DCA (download date: Sep. 4, 2018).

Single-cell Variational Inference (scVI): We used the Torch-based Python package of scVI[14] (download date: June 5, 2018).

scScope: scScope was implemented in Python 3.6 with *TensorFlow-GPU* 1.4.1, *Numpy* 1.13.0, *Scikit-learn* 0.18.1 packages. Unless noted otherwise, for comparisons of methods, scScope was run with default setting (e.g. single GPU mode); hyperparameter choices are given in Supplementary Table 2 and the results of varying them are demonstrated in Supplementary Fig. 14.

Down-sampling training strategies on large-scale dataset: Some non-deep-learning based approaches could not run directly on extremely large datasets. For these approaches, we randomly down-sampled datasets to 20K cell subsets. On these down-sampled datasets, single-cell feature vectors were learned by the respective method and clustered by PhenoGraph. A support vector machine (SVM) was trained on this subset in the latent feature space and then used to assign labels for the rest of cells in the unsampled dataset. The deep-learning approaches we tested could learn features on millions of cell profiles, but the software packages did not provide a function for automatic clustering. For comparisons, we passed the output of their learned single-cell features to the scalable clustering approach.

3. Evaluation of clustering and batch correction

ARI: We used the adjusted Rand index (ARI)[22, 23] to compare label sets of two clustering methods. For two clustering results U and V with r and s clusters on a data set of n cells, n_{ij} denotes the number of cells shared between cluster i in U and cluster j in V. And ARI is defined as

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i*}}{2}\sum_j\binom{n_{*j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{n_{i*}}{2} + \sum_j\binom{n_{*j}}{2}\right] - \left[\sum_i\binom{n_{i*}}{2}\sum_j\binom{n_{*j}}{2}\right] / \binom{n}{2}}$$

where $n_{i*} = \sum_j n_{ij}$, $n_{*j} = \sum_i n_{ij}$ and n is the number of cells in the data set.

Entropy of batch mixing: To evaluate the performance of batch correction, we made use of the score developed in mutual nearest neighbors (MNNs)[24]. In brief, 100 cells were randomly sampled from the entire population, the entropy of the distributions of batches for the nearest 100 neighbors was calculated and the process was iterated 100 times and shown as box plots.

4. Evaluation of Imputation

Imputation error for simulated data: On simulated data, the ground truth dropout vector l is known. The imputation accuracy was defined as the normalized distance between the imputed log count entries and log count ground truth entries. We constructed lists l and $\hat{l}$ whose elements correspond to either ground truth or imputed values (respectively) for all dropouts entries across all cells (Supplementary Fig. 2). We defined the normalized error as:

$$error = \frac{\|l - \hat{l}\|_1}{\|l\|_0}$$

where $\| \|_p$ means the l^p norm of a vector.

Reconstruction error on held-out biological data: For real biological data, the ground truth values for missing genes are unknown. To evaluate scScope's imputation accuracy on real biological data, we followed the same down-sampling strategy as used for scVI[14]. Namely, we randomly split the entire collection of n cells into n_{train} training cells and n_{val} validation cells. We used the different imputation methods to build gene models from the n_{train} cells. On each of the n_{val} cells, we randomly set p% of its non-zero genes as "simulated" missing genes and set their corresponding count values to zero. The real measured values of these simulated missing genes were then used to generate the ground truth list l, and the list $\hat{l}$ was based on inferred values for the simulated missing genes from the n_{val} cells. The reconstruction error was calculated as for simulated data above.

5. Simulated datasets.

Simulation with Splatter—The simulation package Splatter[16] is designed to generate realistic scRNA-seq data. We used this package to generate data with 2000 cells, 3 subpopulation groups, and dropout rates from 1 to 5.

Simulation with SIMLR—SIMLR[9] was used to generate large-scale scRNA-seq data due to limitations in scalability of Splatter. High-dimensional single-cell expression data $x_c \in \mathbb{R}^N$

is generated by a latent code $z_c \in \mathbb{R}^P (P < N)$, which is not observable. Z_c is sampled from a Gaussian mixture model with k Gaussian components, *i.e.* $z_c \sim \sum_{j=1}^{k} \pi_j N(\mu_j, \Sigma_j)$. The mixture coefficients π_j were chosen from a uniform distribution and normalized to sum to 1, the mean vector $\mu_j \in \mathbb{R}^P$ was uniformly sampled in $[0, 1]^P$, and the covariance matrix $\Sigma_j \in \mathbb{R}^{P \times P}$ was chosen to be $\Sigma = 0.1 \times I$, for identity matrix I.

To simulate single-cell gene vectors, we generated a projection matrix $A \in \mathbb{R}^{N \times P}$ to map the low-dimensional latent code to a high-dimensional space. First, we simulated ground truth, x_C^{true}, which cannot be observed:

$$x_C^{true} = Az_c + b$$

where each entry in A was independently sampled from the uniform distribution $U(-0.5, 0.5)$ and bias $b = 0.5$ was added to avoid negative gene expression in the high-dimensional mapping. Second, we simulated the observed profile, x_C^{ObS}, which may contain gene-specific noise and dropout artifacts due to the sequencing technique and platform. Noise was added to the true gene profile by: $x_{cg}^{noise} = x_{cg}^{true} + \varepsilon_{cg}$, where $\varepsilon_{cg} \sim N(0, \Sigma_g^{noise})$ and Σ_g^{noise} was uniformly sampled in the range of [0, 0.2] independently for each gene. Dropout events were added via a double exponential model with decay parameter[25] α:

$$x_{cg}^{obs} = x_{cg}^{noise} \delta\left[q_{cg} > \exp\left(-\alpha x_{cg}^{noise^2}\right)\right]$$

where x_{cg} denotes the g^{th} gene of x_c, q_{cg} was randomly sampled in [0, 1], and $\delta = 1$ if its argument is true and = 0 otherwise.

Simulation with rare cell subgroups—To generate cell subpopulations with rare cell types, we sampled the mixture coefficients π_j from a non-uniform distribution as

$$\pi_{1 \ldots k_m} = q; \pi_{k_m + 1 \ldots k} = \frac{1 - q * k_m}{k - k_m}$$

where $q \ll 1/k$ is the mixture fraction for each minor cluster, k_m is the number of rare cell subpopulations, and k is the total number of subpopulations.

6. Biological datasets.

(a) Lung tissue data—The lung tissue dataset is part of the Mouse Cell Atlas data[6]. This dataset was downloaded from Gene Expression Omnibus (GEO) database (accession number: GSE108097). In this dataset, 6,940 cells were sequenced via three independent experiments by Microwell-seq, with 2,512, 1,414 and 3,014 cells in each batch. 1,000 highly variable genes were selected for analysis.

(b) Cord blood mononuclear cells dataset—In this dataset, 8,617 cord blood mononuclear cells (CBMC) were profiled by CITE-seq, a new technology which enabled the simultaneous measurement of protein levels and transcriptome levels for each cell[26]. This dataset was downloaded from GEO database (accession number: GSE100866). Cell types in CBMC have been extensively studied and identified. Based on this prior knowledge, 13 monoclonal antibodies were chosen to identify *bone fide* cell types. These antibodies serve as an "orthogonal" ground truth to evaluate analysis results based on RNA-seq data.

In the data pre-processing stage, we removed the spiked-in mouse cells and only kept 8,005 human cells for analysis using the cell-filtering strategy introduced in original study[26]. The top 1,000 most variable human genes were selected for downstream analysis after the log1p transformation and normalization by library size. For antibody data, we used the centered log ratio transformed antibody-derived tags (ADTs), which is also provided by authors.

To evaluate the performance of each method, we first automatically identified cell populations based on ADTs data using PhenoGraph. Then, scRNA-seq data were input to each method to learn latent representations which were used by PhenoGraph to predict cell types. The ADTs-derived cell types were taken as ground truth to evaluate the accuracy of cell types by scRNA-seq data.

(c) Retina data—In this dataset, 44,808 cells were captured from the retinas of 14-day-old mice and sequenced by DropSeq[4]. Data were obtained from the GEO database (accession number: GSE63473). We followed previous procedures to select the 384-most variable genes[4] and then perform the log1p transformation on their expression values. After clustering, we annotated clusters using the same marker genes as in the original study[4].

We identified candidate cell types based on the highest average type-specific marker expression (Supplementary Table 4). For each cluster, we calculated the fold-change values of all cell-type markers, and if at least one type-specific gene marker was expressed significantly higher ($\log_2$ fold change > 0.5) than in all other clusters, we assigned the cluster with the candidate cell type. Otherwise the cluster was assigned to the cell type "Rod cell".

(d) Mouse cell atlas data—The mouse cell atlas (MCA) dataset is designed to offer a comprehensive investigation of all major cell types in mouse[6]. Data were downloaded from the GEO database (accession number GSE108097). In the dataset, 405,796 cells were sampled from 51 tissues and were sequenced by Microwell-seq.

Data were first normalized by library size, and 1,000, 2,000, 5,000, 10,000 or 20,000 top-variable genes were selected to test the scalability of each method on gene numbers. Only the deep-learning based single cell tools (DCA, scVI and scScope) could be applied directly to this large-scale dataset. Further, to identify clusters in the MCA dataset, we applied our scalable clustering approach to the latent features.

In most of the 51 tissues, one major cell type dominated the cell population (see Figure 2b–c in ref[6]). Therefore, we used the tissue identify as a proxy for ground truth to evaluate cell-type discovery.

(e) Intestinal data—In this dataset, intestinal epithelial cells were captured from 10 mice and sequenced using droplet-based scRNA-seq[27]. Data were downloaded from the GEO database (accession number GSE92332). Among all cells, 1,770 cells from 2 mice were infected by *Salmonella* for 2 days; 2,121 cells (2 mice) and 2,711 cells (2 mice) were infected by *H. polygyrus* for 3 and 10 days, respectively. An additional 3,240 cells were sequenced from 4 healthy mice as a control group. We again followed the same procedure that log-transformed the expression data and selected top 1,000 most variable genes as input to scScope. For cell subpopulation annotation, we first assigned clusters to one of 7 major cell types (stem, cell-cycle related, distal enterocyte, proximal enterocyte, goblet, enteroendocrine, and tuft) according to the maximum averaged expression of cell-type makers (Supplementary Table 9). Second, cell-cycle related clusters were subdivided into increasing stages of maturation (transit amplifying early stage, transit amplifying late stage, and enterocyte progenitor) based on the ratio of cell-cycle & stem cell markers to enterocyte expression (Supplementary Table 10). Third, the distal and proximal enterocyte clusters were further classified (immature *vs*. mature) based on increasing expression levels of the enterocyte gene markers.

After annotating clusters, we calculated the cell proportion for each mouse and then averaged the proportions among mice of the same infection condition. For significant tests of proportion changes after infection, we compared proportions of mice in control group and in infection group using a two-sided t-test and rank-sum test. P-values were obtained under the null hypothesis that no changes happened in proportions after infection.

Overexpressed genes for each cluster were also identified by the same differential expression analysis.

(f) Brain data—Data were obtained from 10× Genomics (http://10xgenomics.com). 1,308,421 cells from embryonic mice brains were sequenced by Cell Ranger 1.3.0 protocol. We transformed unique molecular identifier (UMI) count data into log(TPM+1)[4] and calculated the dispersion measure (variance/mean) for each gene. According to the rank of the dispersion measure, we selected the top 1,000 most variable genes for analysis.

For fast exploration of the data in Fig. 3, where no comparisons to other methods were used, we used a batch size of scScope to 512, trained the model for 10 epochs and used 4 GPUs. Cells were further clustered into 36 groups by our density down-sampling clustering. We annotated clusters to three major types (excitatory neurons, inhibitory neurons and non-neuronal cells) based on maximal-expressed maker genes (Supplementary Table 7).

To identify cluster-specific overexpressed genes, we then conducted differential expression analysis for each gene. We normalized UMI-count to the range of [0 1] for each gene, enabling comparisons across genes. Then gene-expression fold-change and rank-sum P-values were calculated between cells within vs. outside each cluster. Significantly overexpressed genes were identified using the criteria of $\log_2$ fold change > 0.5 and rank-sum P-value < 0.05.

Data set was download from https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.3.0/1M_neurons on December 10, 2017.

The data analysis by 10xgemonics was obtained from http://storage.pardot.com/172142/31729/LIT000015_Chromium_Million_Brain_Cells_Application_Note_Digital_RevA.pdf.

7. Software used in study

MAGIC: https://github.com/KrishnaswamyLab/MAGIC

ZINB-WaVE: https://github.com/drisso/zinbwave

SIMLR: https://github.com/bowang87/SIMLR_PY

DCA: https://github.com/theislab/dca

scVI: https://github.com/YosefLab/scVI

PhenoGraph: https://github.com/jacoblevine/PhenoGraph

Splatter: https://github.com/Oshlack/splatter-paper

8. Life Sciences Reporting Summary

Detailed information about experimental design is available in attached Reporting Summary.

9. Code availability

scScope can be obtained as an installable Python package, which can now be obtained via “pip install scScope”, and is available under the Apache license. All software, instructions and software updates will be maintained on our lab’s github page: https://github.com/AltschulerWu-Lab/scScope.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgements

We thank Drs. Jeremy Chang, Satwik Rajaram, Laura Sanman and Susan Shen for their helpful comments. We gratefully acknowledge the support of NIH R01 EY028205 and GM112690, NSF PHY-1545915 and SU2C/MSKCC 2015–003 to S.J.A., NCI-NIH RO1 CA185404 & CA184984 to L.F.W., the Institute of Computational Health Sciences (ICHS) at UCSF to S.J.A. and L.F.W., and Project of NSFC (No. 61327902) to Q.D. and F.B.

References

1. Gawad C, Koh W & Quake SR Single-cell genome sequencing: current state of the science. Nature Reviews Genetics 17, 175–188 (2016).
2. Saliba A-E, Westermann AJ, Gorski SA & Vogel J Single-cell RNA-seq: advances and future challenges. Nucleic Acids Research 42, 8845–8860 (2014). [PubMed: 25053837]
3. Shalek AK et al. Single-cell RNA-seq reveals dynamic paracrine control of cellular variation. Nature 510, 363–+ (2014). [PubMed: 24919153]

4. Macosko EZ et al. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. Cell 161, 1202–1214 (2015). [PubMed: 26000488]

5. Zheng GXY et al. Massively parallel digital transcriptional profiling of single cells. Nature Communications 8 (2017).

6. Han X et al. Mapping the mouse cell atlas by Microwell-seq. Cell 172, 1091–1107. e1017 (2018). [PubMed: 29474909]

7. Pierson E & Yau C ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. Genome Biology 16 (2015).

8. Risso D, Perraudeau F, Gribkova S, Dudoit S & Vert J-P A general and flexible method for signal extraction from single-cell RNA-seq data. Nature communications 9, 284 (2018).

9. Wang B, Zhu J, Pierson E, Ramazzotti D & Batzoglou S Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. Nature Methods 14, 414–+ (2017). [PubMed: 28263960]

10. Cleary B, Le C, Cheung A, Lander ES & Regev A Efficient Generation of Transcriptomic Profiles by Random Composite Measurements. Cell 171, 1424–+ (2017). [PubMed: 29153835]

11. Van Dijk D et al. Recovering Gene Interactions from Single-Cell Data Using Data Diffusion. Cell 174, 716–729,(2018). [PubMed: 29961576]

12. Butler A, Hoffman P, Smibert P, Papalexi E & Satija R Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nature biotechnology 36, 411 (2018).

13. 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. Journal of Machine Learning Research 11, 3371–3408 (2010).

14. Lopez R, Regier J, Cole MB, Jordan MI & Yosef N.J.N.m. Deep generative modeling for single-cell transcriptomics. Nature Methods 15, 1053–1058 (2018). [PubMed: 30504886]

15. Eraslan G, Simon LM, Mircea M, Mueller NS & Theis FJ Single cell RNA-seq denoising using a deep count autoencoder. bioRxiv, 300681 (2018).

16. Zappia L, Phipson B & Oshlack A Splatter: simulation of single-cell RNA sequencing data. Genome biology 18, 174 (2017). [PubMed: 28899397]

17. Levine JH et al. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. Cell 162, 184–197 (2015). [PubMed: 26095251]

18. Jeon CJ, Strettoi E & Masland RH The major cell populations of the mouse retina. Journal of Neuroscience 18, 8936–8946 (1998). [PubMed: 9786999]

19. Rosenberg AB et al. Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. Science 360, 176–+ (2018). [PubMed: 29545511]

20. Tasic B et al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. Nature Neuroscience 19, 335–+ (2016). [PubMed: 26727548]

21. Franke L et al. Reconstruction of a functional human gene network, with an application for prioritizing positional candidate genes. American Journal of Human Genetics 78, 1011–1025 (2006). [PubMed: 16685651]

22. Hubert L & Arabie P Comparing Partitions. Journal of Classification 2, 193–218 (1985).

23. Rand WM Objective Criteria for Evaluation of Clustering Methods. Journal of the American Statistical Association 66, 846–850 (1971).

24. Haghverdi L, Lun AT, Morgan MD & Marioni JC Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. Nature biotechnology 36, 421 (2018).

25. Kharchenko PV, Silberstein L & Scadden DT Bayesian approach to single-cell differential expression analysis. Nature Methods 11, 740–U184 (2014). [PubMed: 24836921]

26. Stoeckius M et al. Simultaneous epitope and transcriptome measurement in single cells. Nature methods 14, 865 (2017). [PubMed: 28759029]

27. Haber AL et al. A single-cell survey of the small intestinal epithelium. Nature 551, 333–+ (2017). [PubMed: 29144463]

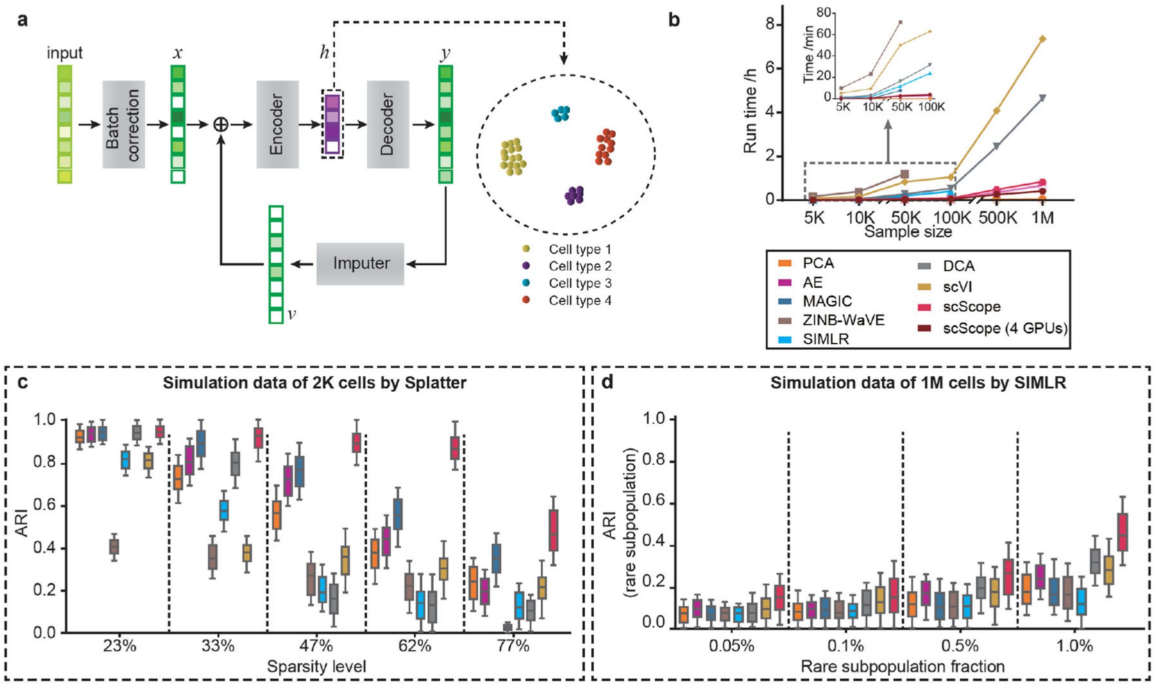

Fig. 1. Overview of scScope architecture and performance on simulated datasets

a) Overview of the recurrent network architecture of scScope.

An input single-cell profile with dropout gene measurements (white entries) is corrected for batch effects, then the corrected vector *x* is sequentially processed by an encoder layer (for feature extraction), decoder layer (for noise reduction) and imputation layer (for dropout imputation). The imputed vector *v* is added back to the batch-corrected input profile *x* to fill in missing values. This process proceeds recursively *T* times to produce a final signature feature vector output *h* used for biological discovery, such as identification of phenotypically distinct subpopulations.

b) Comparison of run time on dataset of different scales.

Datasets of varying size were randomly subsampled from a dataset containing 1.3 million mouse brain cells and used for comparison (Methods).

c) Clustering accuracy for 2K scRNA-seq data with varying fraction of sparsity.

Splatter was used to generate 2K cells with 3 subpopulations with varying dropout rates (Supplementary Table 3). Accuracy measurement is based on adjusted Rand index (Methods). For each simulated condition, n = 10 random replicates were simulated; Box plot: median (center line), interquartile range (box) and minimum-maximum range (whiskers).

d) Clustering accuracy for 1M scRNA-seq data with varying fraction of rare subpopulations.

The simulation strategy of SIMLR was used to generate 1M cells. Dropout rate = 0.5; total number of clusters = 50, number of rare subpopulations = 5; replicate number n = 10. For MAGIC, ZINB-WaVE and SIMLR, the 1M datasets were randomly down sampled to 10K, and PhenoGraph was used for *de novo* cell subpopulation discovery. For methods run on the 1M dataset, a scalable clustering approach was used to identify subpopulations (Methods). Box plot as in 1c.

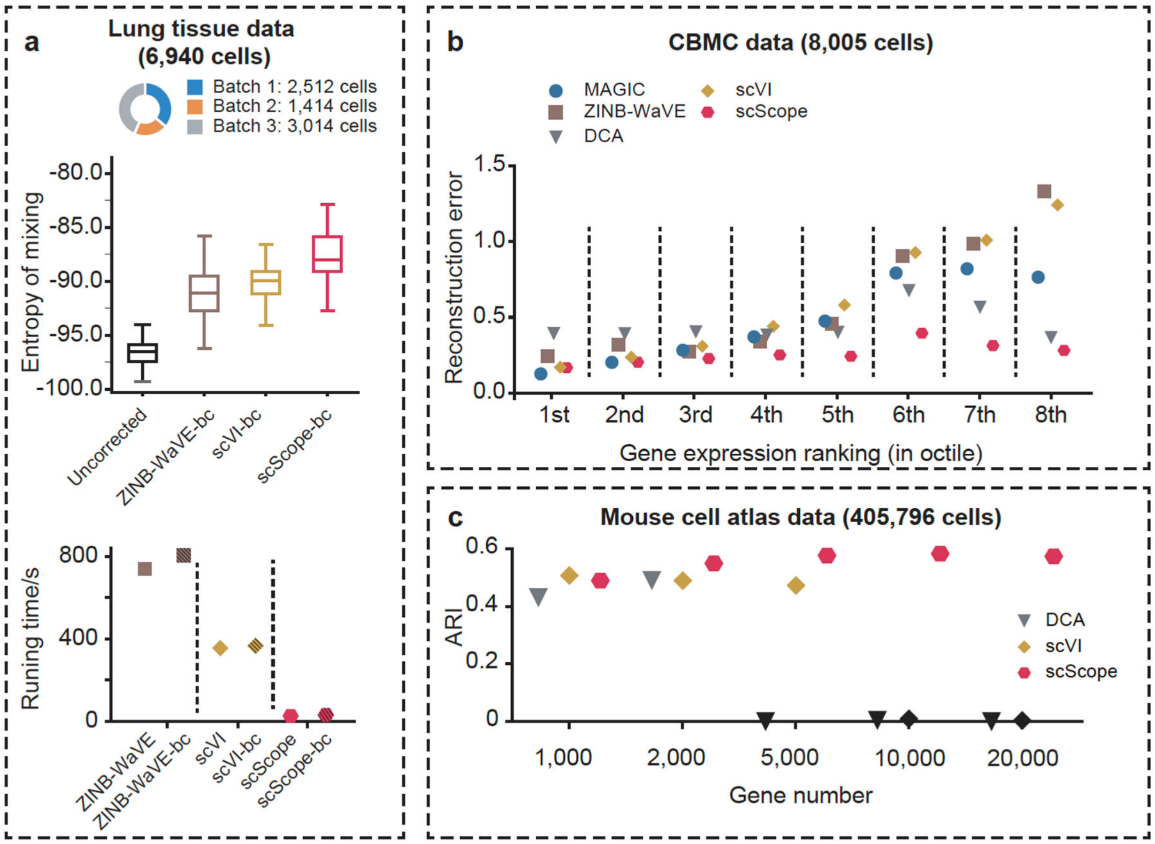

Fig. 2. Evaluation of methods on experimental scRNA-seq datasets.

a) Analysis of batch correction. Comparisons of (top) batch mixing entropy and (bottom) computational runtime without or with batch correction using mouse lung tissue scRNA-seq dataset. Top box plot: median (center line), interquartile range (box) and minimum-maximum range (whiskers); n = 100 replicates of 100 randomly selected cells across all batches. Bottom: run time to process whole dataset.

b) Analysis of imputation accuracy for different gene expression levels. Comparison of imputation error for dropout genes with different (octiles) gene expression levels using the cord blood mononuclear cell (CBMC) scRNA-seq dataset.

c) Analysis of subpopulation identification for increasing gene depth. Using the mouse cell atlas, we compared the ability of different approaches to identify the 51 known tissues in the atlas. Black color: provided software package was unable to complete the task.

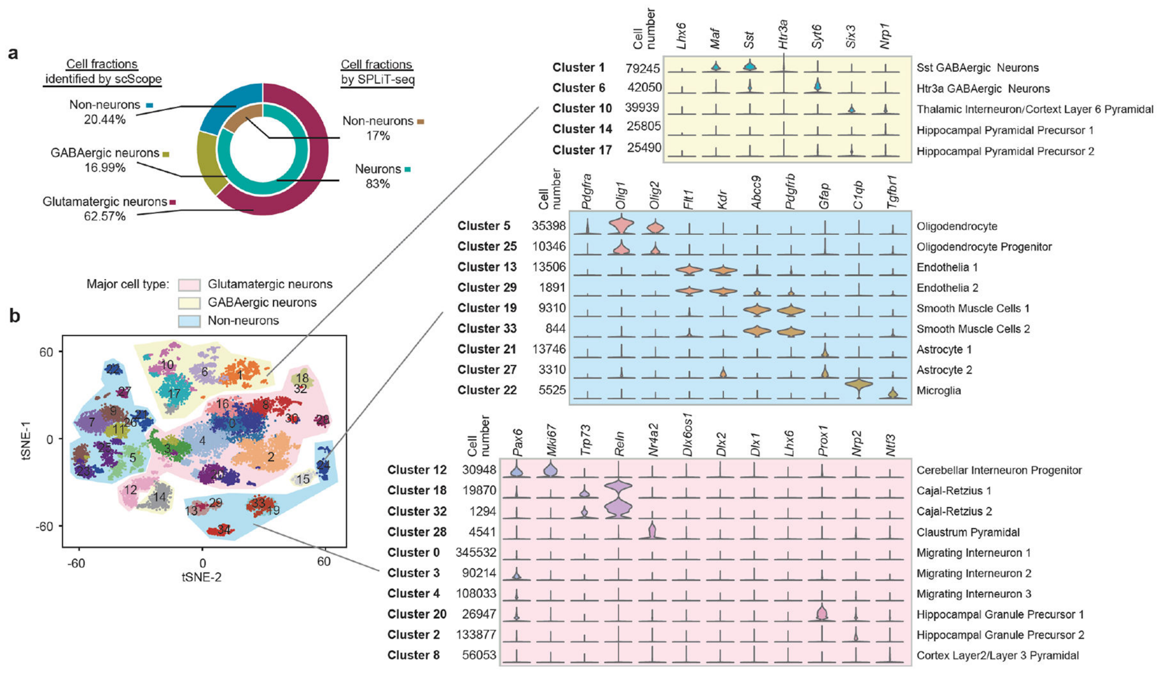

Fig. 3. Application of scScope to explore biology in 1.3M mouse brain dataset.

a) Fractions of three major cell types (glutamatergic neurons, GABAergic neurons and non-neurons) identified by scScope and comparisons with reported neuron fractions by previous SPLiT-seq research.

b) Left: scScope results visualized using tSNE on n = 30K cells (randomly sampled from the full dataset). Clusters were divided to three major types based on gene markers. Right: Large-scale annotation of clusters to known cell types according to top 10 overexpressed genes. Violin plots: expression distribution of marker genes for discovered clusters. Vertical axis (left): clusters with known cell type annotations and corresponding cell numbers. Horizontal axis: differentially expressed marker genes across shown clusters. Vertical axis (right): cluster annotation based on previously reported cell-subtype-specific genes.