Massive single-cell RNA-seq analysis and imputation via deep 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.

*These authors contributed equally to this work.

#To whom correspondence should be addressed:

lani.wu@ucsf.edu, steven.altschuler@ucsf.edu

Recent advances in large-scale 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) can provide 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[1-5]. Recent platforms, such as DropSeq[6, 7] and GemCode[8], have enabled large-scale scRNA-seq on millions of cells at a time, which offers an unprecedented resolution at which to dissect cell-type compositions.

However, these advances have led to two acute challenges. First, single-cell profiles are highly susceptible to transcript amplification noise and dropout events[9] and such artifacts can become even more pronounced as tradeoffs are made to sequence larger numbers of cells. Second, current analytical packages[9-13] are unable to scale to large datasets due to computational memory and/or speed restrictions. New approaches are needed to extract informative representations from these extremely noisy, massive, high-dimensional scRNA profiles.

To overcome these challenges, we developed scScope, a software package that uses deep learning to extract informative features (low-dimensional representations of gene expression profiles) per cell from massive single-cell data (**Fig. 1a**). A major innovation of scScope is the design of a self-correcting layer. This layer exploits a recurrent network structure to iteratively perform imputations on zero-valued entries of input scRNA-seq data (**Methods**). Importantly, scScope simultaneously conducts cellular-feature learning and dropout imputation in a joint framework and can be trained in parallel on multiple GPUs (**Fig. 1b**). To our knowledge, scScope is the first parallelized deep-learning framework for unsupervised, single-cell data modeling that can deal with both massive and noisy single-cell expression data.

To calibrate the performance of scScope, we initially made use of simulated single-cell expression profile datasets[9], where “ground truth” is known (**Methods**). We compared our results with published methods (ZIFA[9], SIMILR[10], SMAF[11] and PCA as well as with an unpublished Bayesian generative model, scVI[18] (**Methods** and **Supplementary Table 1**).

We began with moderate-sized datasets, containing 10,000 (10K) single-cell profiles. We first investigated the ability to discover a fixed number of subpopulations (k = 50) with RNA-seq dropout rates ranging from ~60% to ~90% (controlled by “decay rate” α ranging from 1 to 0.15, respectively; **Fig. 1c-d, Supplementary Table 2**). scScope performs similarly to other approaches for moderate dropout rates (**Fig. 1c**) and in terms of

imputation error (**Fig. 1d**). However, scScope outperforms all other approaches in finding subpopulations for high dropout rates (**Fig. 1c**). Additionally, the run time of scScope is significantly faster than other single-cell analytical packages (**Fig. 1e**). Second, we varied the number of hidden subpopulations (*k* between 10 to 200) for a fixed, moderate dropout rate (α = 0.5). As expected, clustering accuracy degraded for all approaches with increasing numbers of hidden clusters. Yet, scScope consistently outperformed all tested methods in identifying subpopulations with higher numbers of hidden clusters (**Supplementary Fig. 1**).

We then tested the ability of scScope to analyze massive data sets, containing one million (1M) single-cell profiles. While existing single-cell computational approaches could also be adapted on large-scale dataset by a down-sampling approach[6] (**Methods**), scScope can extract informative features (based on deep learning) and perform graph clustering (**Methods** and **Supplementary Fig. 2**) directly on millions of cells. Compared with down-sampling-based approaches, scScope greatly outperformed the other off-the-shelf single-cell computational packages for discovering cell-type compositions when either all subpopulations are of comparable size (**Fig. 1f**) or contain rare subpopulations (**Methods** and **Fig. 1g**). Importantly, we found that scScope can accurately identify subpopulations comprising as little as 0.1% of the overall cells, which would be missed by down-sampling the original data. Between the two models that can deal with large-scale data, the training speed of scScope is at least 15 times faster than scVI (**Fig. 1h**). Thus, our calibration showed that scScope can efficiently and accurately identify cell subpopulations from datasets with high dropout rates, large numbers of subpopulations, rare cell types and across a wide range of dataset sizes.

We next evaluated scScope on three experimental single-cell RNA datasets. Our goals were to test the ability of scScope to recover original cell-type compositions (retina dataset), to reveal changes in cell-type composition under perturbed conditions (intestinal dataset), and to scale to large datasets to reveal new subpopulations (brain dataset).

The mouse retina dataset[6] contains 44K cells from dissociated mouse retinas. We applied scScope to the 384 most variable genes identified from our preprocessing step and used PhenoGraph[15] to identify subpopulations (**Methods**). The original study identified 39 cell subpopulations using fine-grained manual adjustment and expert visual inspection, which we took as a reference for our comparisons. Compared to other approaches, scScope automatically identified the most similar clustering (number and assignment) to those reported in the original study (**Fig. 2a**, left) and also exhibited the tightest groupings (**Fig. 2a**, right and **Methods**). We annotated the clusters to cell types based on gene markers reported in the original study (**Supplementary Table 3** and **Methods**). (The subpopulation of pericytes, identified by manual adjustment in the original study, was missed by all tested methods.) Overall, clusters identified by scScope showed the most statistically significant enrichment of specific cell-type markers (larger fold-changes) (**Fig.**

2b) and were highly consistent with previous, microscopy validation estimates of cell-type composition and proportion[16] (**Fig. 2c** and **Supplemental Fig. 3**). Thus, scScope performed well at recovering cell-type compositions that are biologically interpretable.

The intestinal dataset[17] contained ~10K cells obtained from mouse intestines with different infection conditions. We applied scScope to the 1,000 most variable genes (**Fig. 2d** and **Methods**). In the original study, enterocytes were identified as a single cluster. Interestingly, scScope further subdivided this cell type into four subpopulations: differential expression of markers provided delineation of distal *vs*. proximal enterocyte subpopulations, while expression levels of these markers provided further delineation into immature *vs*. mature subtypes. The assessment of maturity was based on expression levels of distal or proximal markers (**Fig. 2e** and **Supplementary Fig. 4**). Identifying these refined enterocyte subpopulations allowed us to make 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 (**Fig. 2f** and **Supplementary Table 4**). This geographic pattern of *Saa1* expression is known for *Salmonella* infection, but is a novel prediction arising from scScope analysis for *H. polygyrus* infection. Thus, scScope can be used to rapidly explore scRNA-seq data to predict novel gene function and identify new cell subtypes from perturbed conditions.

The brain dataset contained 1.3M cells obtained from multiple brain regions, including the cortex, hippocampus and ventricular zones, of two embryonic mice. We applied scScope to the 1000 most variable genes (**Methods**). 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. 2g**-**h** and **Supplementary Table 5**): 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 previous brain study[19] (**Fig. 2g**). Our entire analysis of the 1.3M brain cells required only ~2 hours and 64GB memory (**Fig. 2i**, left and **Supplementary Fig. 5**); in comparison, the original study reported requiring over 350 hours and 300 GB memory to process the same data. Using down-sampling to analyze 1.3M cells with competing single-cell packages revealed fewer cell subpopulations (**Supplementary Fig. 6**), and cell proportions identified by these methods showed poorer agreement with reference proportions (neuron *vs.* non-neuron, **Fig. 2j**). We desired to determine whether we could identify biological meaning to the 36 clusters, some of which contained fewer than 1000 cells. Satisfyingly, by comparing our top overexpressed genes with known cell-type markers[19-21] (**Supplementary Table 6**), we were able to assign two thirds of the clusters to known cell types (**Fig. 2h**). Thus scScope can rapidly, automatically and directly identify *bona fide*, rare cell types from datasets with over a million single-cell transcriptional profiles.

Taken together, scScope offers a platform that will keep pace with the rapid advances in RNA-seq, enabling rapid exploration and accurate dissection of heterogeneous biological states within extremely large datasets of single-cell transcriptional profiles containing dropout measurements.

Acknowledgements

We thank Drs. Jeremy Chang, Satwik Rajaram and Laura Sanman for their helpful comments. We gratefully acknowledge the support of NIH R01 EY028205 and GM112690, NSF PHY-1545915 and SU2C/MSKCC 2015-003 to SJA, NCI-NIH RO1 CA185404 & CA184984 to LFW and the Institute of Computational Health Sciences (ICHS) at UCSF to SJA and LFW.

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.

Competing financial interests

The authors declare no competing interests.

References

1. Gawad, C., Koh, W. & Quake, S.R. Single-cell genome sequencing: current state of the science. *Nature Reviews Genetics* **17**, 175-188 (2016).
2. Kolodziejczyk, A.A., Kim, J.K., Svensson, V., Marioni, J.C. & Teichmann, S.A. The Technology and Biology of Single-Cell RNA Sequencing. *Molecular Cell* **58**, 610-620 (2015).
3. Saliba, A.-E., Westermann, A.J., Gorski, S.A. & Vogel, J. Single-cell RNA-seq: advances and future challenges. *Nucleic Acids Research* **42**, 8845-8860 (2014).
4. Shalek, A.K. et al. Single-cell RNA-seq reveals dynamic paracrine control of cellular variation. *Nature* **510**, 363-+ (2014).
5. Rajaram, S. et al. Sampling strategies to capture single-cell heterogeneity. *Nature Methods* **14**, 967-+ (2017).
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.Y. et al. Massively parallel digital transcriptional profiling of single cells. *Nature Communications* **8** (2017).
8. Eisenstein, M. Startups use short-read data to expand long-read sequencing market. *Nature Biotechnology* **33**, 433-435 (2015).
9. Pierson, E. & Yau, C. ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biology* **16** (2015).
10. 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).
11. Cleary, B., Le, C., Cheung, A., Lander, E.S. & Regev, A. Efficient Generation of Transcriptomic Profiles by Random Composite Measurements. *Cell* **171**, 1424-+ (2017).
12. Kiselev, V.Y. et al. SC3: consensus clustering of single-cell RNA-seq data. *Nature Methods* **14**, 483-+ (2017).
13. Butler, A., Hoffman, P., Smibert, P., Papalexi, E. & Satija, R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology* (2018).
14. Bengio, Y. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* **2**, 1-127 (2009).
15. Levine, J.H. et al. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. *Cell* **162**, 184-197 (2015).
16. Jeon, C.J., Strettoi, E. & Masland, R.H. The major cell populations of the mouse retina. *Journal of Neuroscience* **18**, 8936-8946 (1998).
17. Haber, A.L. et al. A single-cell survey of the small intestinal epithelium. *Nature* **551**, 333-+ (2017).
18. Lopez, R., Regier, J., Cole, M.B., Jordan, M. & Yosef, N. Bayesian Inference for a Generative Model of Transcriptome Profiles from Single-cell RNA Sequencing. *bioRxiv* (2018).
19. Rosenberg, A.B. et al. Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science* **360**, 176-+ (2018).
20. Tasic, B. et al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nature Neuroscience* **19**, 335-+ (2016).

21. Mi, D. et al. Early emergence of cortical interneuron diversity in the mouse embryo. *Science* **360**, 81-85 (2018).
22. 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).
23. Kingma, D.P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
24. Abadi, M. et al. in USENIX Symposium on Operating Systems Design and Implementation, Vol. 16 265-283 (2016).
25. Hubert, L. & Arabie, P. Comparing Partitions. *Journal of Classification* **2**, 193-218 (1985).
26. Rand, W.M. Objective Criteria for Evaluation of Clustering Methods. *Journal of the American Statistical Association* **66**, 846-850 (1971).
27. Calinski, T. A Dendrite Method for Cluster Analysis. *Biometrics* **24**, 207-& (1968).
28. Kharchenko, P.V., Silberstein, L. & Scadden, D.T. Bayesian approach to single-cell differential expression analysis. *Nature Methods* **11**, 740-U184 (2014).

Figures and **Legends**

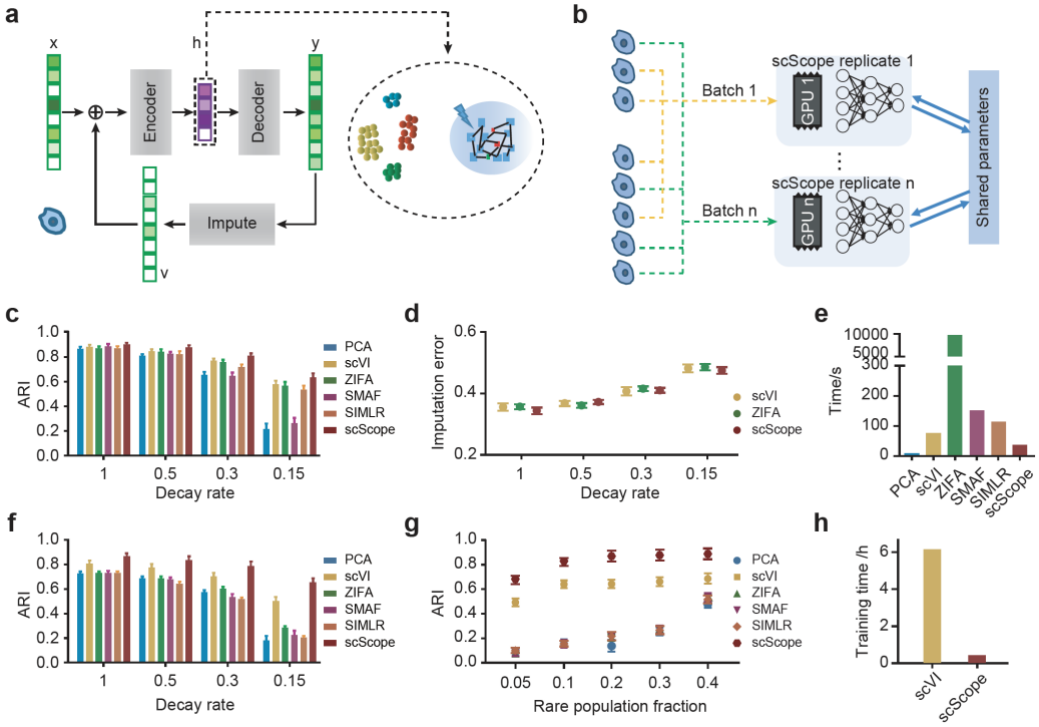

Fig. 1. Overview of scScope architecture and performance on simulated datasets of 10K and 1M single-cell profiles.

a-**b**) **Overview of the recurrent network architecture of scScope.** (**a**) An input single-cell profile x with dropout gene measurements (white entries) 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 raw 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 (left) or gene network response to perturbations (right). (**b**) Training of scScope is implemented on multiple GPUs to enable fast learning and low memory costs (**Methods**).

c-**e**) **Comparison of clustering, imputation accuracy and run time on simulated single-cell datasets of 10K single-cell profiles.** Method comparisons for varying fractions of dropout genes (controlled by decay rate). Accuracy measurement is based on (**c**) adjusted Rand index or (**d**) fractional measure of imputation errors. (**e**) Comparison of run time performance (decay rate = 0.5). On each simulated condition, 10 random replicates were simulated and tested by different approaches; mean values and standard deviations (error bars) are reported on (**c**-**d**).

f-**h**) **Comparison of clustering accuracy and training time on simulated single-cell datasets of 1M single-cell profiles.** Method comparisons for varying fractions of dropout genes with varying (**f**) fractions of dropout genes or (**g**) rare population fractions. Adjusted Rand index (Mean + STD) was calculated on 10 replicates with the same parameters. scScope and scVI directly analyzed the whole dataset while other methods analyzed 20K down-sampled subset. (**h**) Comparisons of training costs on the full 1M dataset.

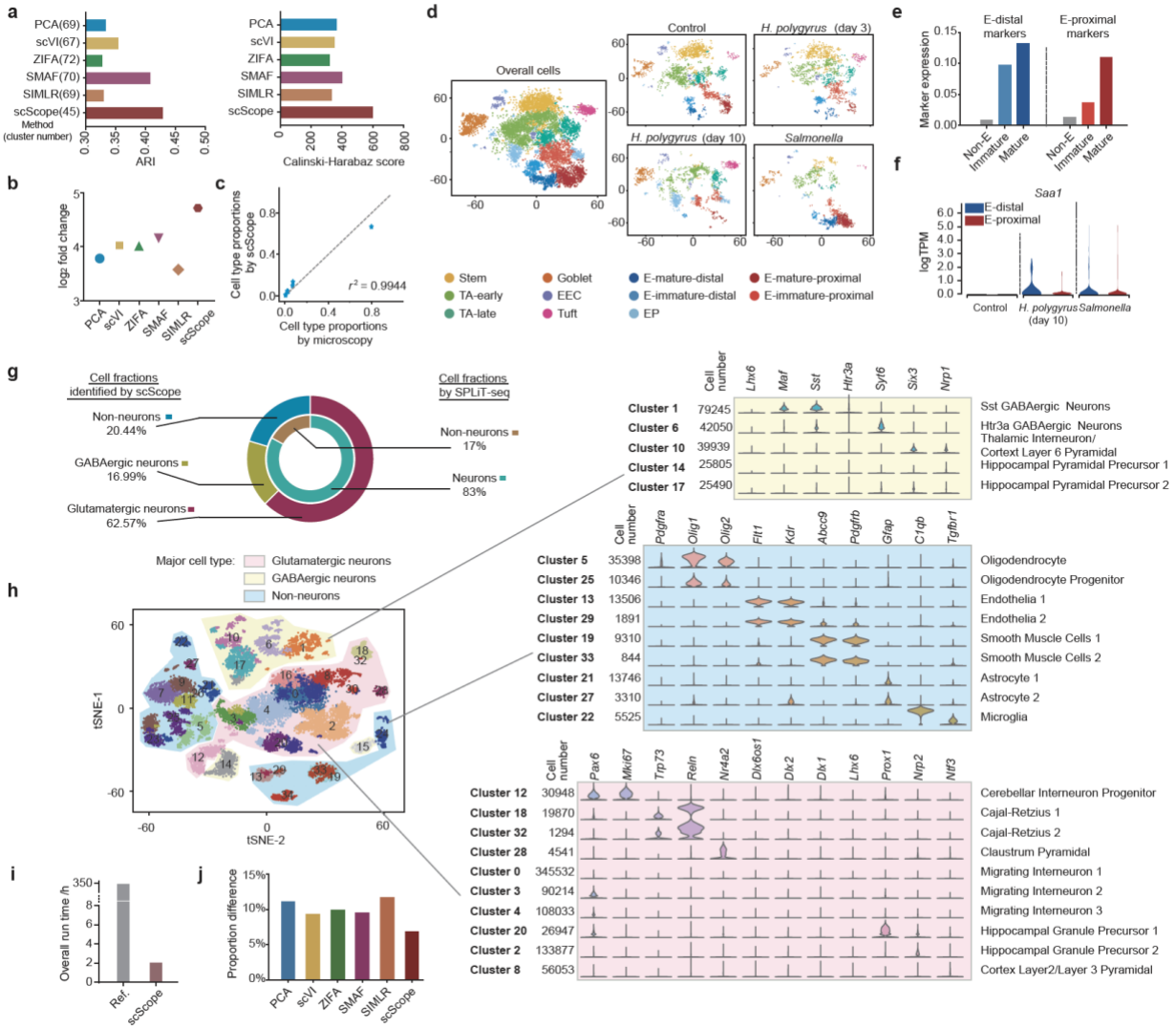

Fig. 2. Application of scScope to identify cell compositions on three experimental mouse scRNA-seq datasets

a-**c**) **Analysis of retina scRNA-seq dataset.** (**a**) Comparison of clustering consistency with original study performed on mouse retina dataset based on adjusted Rand index (left) and Calinski-Harabaz scores (right). Cluster numbers derived from each method are indicated in parentheses (left). (**b**) Averaged fold-change comparisons of cell-type discovery for different methods. (**c**) Consistency between cell proportions identified by scScope and reported cell proportions by microscopy.

d-**f**) **Analysis of intestinal scRNA-seq dataset.** (**d**) Changes in cell-type composition of mouse intestinal epithelial under different infection conditions, visualized via tSNE plots. TA: transit amplifying, EEC: enteroendocrine, EP: enterocyte progenitor, E: enterocyte. scScope identified four subtypes of enterocyte cells. (**e**) Identification of mature vs. immature and distal vs. proximal enterocyte subpopulations. Shown are expression levels

of E-distal and E-proximal gene markers (average UMI count) on the four enterocyte subtypes predicted by scScope and, for comparison, all other clusters (Non-E). (**f**) Discovery of differential expression of the gene *Saa1* in distal and proximal enterocytes after *Salmonella* and *H. polygyrus* infections.

g-**j**) **Analysis of brain scRNA-seq dataset.** (**g**) 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. (**h**) (**left**) scScope results visualized using tSNE on 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. 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): previously reported cell subtype-specific genes, used to provide cluster annotation. (**i**) Comparisons of overall run time between scScope, and previously reported reference ("Ref.") results (left) on 1.3 million mouse brain single-cell profiles. (**j**) Differences between neuron fractions obtained by comparison methods and known proportions on 1.3M mouse brain data. scScope and scVI were directly applied to the whole dataset while other methods required down-sampling to 20K before performing analysis.

Methods

1. scScope model and training

The architecture of the scScope network has three layers (**Fig. 1a**). The parameters in these layers are learned from data in an end-to-end manner through optimization.

scScope network architecture

Encoder

For each cell c, scScope makes use of an encoder layer to compress the high-dimensional single-cell expression profile $x_c \in \mathbb{R}^N$ into a low-dimensional, latent representation $h_c \in \mathbb{R}^M, M < N$. Here,

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

where the encoder layer $f_E(\cdot)$ is a composition of a linear transform with learnable parameters $W_E \in \mathbb{R}^{N \times M}$ and $b_E \in \mathbb{R}^M$ followed by the nonlinear activation $r(\cdot)$. Throughout, we use $r(\cdot)$ to denote the standard rectified linear unit (ReLU):

$$r(v) = v^+ = \max(0, v).$$

Decoder

A decoder layer is established to decompress the latent representation h_c to an output $y_c \in \mathbb{R}^N$ that is of the same dimension as the input single-cell profile,

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

where the decoder layer $f_D(\cdot)$ is a composition of a linear transform with learnable parameters $W_D \in \mathbb{R}^{N \times M}$ and $b_D \in \mathbb{R}^N$ followed by the nonlinear activation $r(\cdot)$. The nonlinear ReLU activation $r(\cdot)$ sets all negative values to zero, which makes experimental sense for gene transcript abundance measured by nonnegative values. When minimizing the differences between y_c and x_c ($\forall\, c = 1 \ldots . n$) in the training set (of size n), the above encoder-decoder network acts as a typical auto-encoder (AE) neural network. AEs exhibit the potential for generating a "clean" signal y_c by removing additive noise from x_c via L_p norm minimization (typically, $p = 2$)[14]. However, the AE paradigm lacks a mechanism for dealing with dropout entries (missing data), which can be an even more severe problem than noise for scRNA-seq.

Imputer

We next focus on estimating missing data measurements. Zero-valued measurements can be due either to biological reasons (un-transcribed genes), or to technical reasons

(dropouts). Our imputation step is designed to deal with the later problem, namely to estimate dropout gene expression values. Of course, a challenge shared by all single-cell-based imputation is to distinguish between zero-expressed *vs.* dropout measurements.

To this end, we developed a self-correcting layer to impute missing entries caused by dropout during sequencing. Our imputation approach implicitly makes use of the fact that subsets of genes are often co-regulated (*e.g.* by common transcription factors or pathway activation) and that their patterns of co-expression can be learned by observation of sufficiently many cells. The decoder in our AE framework is enabled to find such patterns from the latent space representation. The studies of transcriptional regulatory mechanisms have led to the development of many approaches for network-based pathway recovery.

For our purposes of imputing expression values from initial zero values, we chose to make use of a widely used sparse graph-reconstruction approach[22]. We only perform imputation operations for zero measurements of the original single cell profile x_c. Assume for cell c that gene g has zero expression value in the expression profile x_c. Our goal is to impute v_{cg}, the value of gene g in cell c, using the output of the decoder layer, y_c. We do this by:

$$v_{cg} = r(w_g^T y_c + b_g)$$

where learnable parameters $w_g \in \mathbb{R}^N$ is a sparse vector and b_g is a bias for each gene g. (The ith entry in vector w_g indicates the influence of a gene i on the expression of gene g.) These gene-specific parameters are learned from regression across all cells. We used y_c rather than the original input x_c to represent gene expression values as it is expected to contain less expression noise after AE reconstruction. As above, the nonlinear ReLU activation $r(\cdot)$ forces regressed values to be non-negative; the hope for truly zero-expressed genes is that regression will return non-positive values, which are set to zero after ReLU activation.

As with our previous network layers, we can describe the imputation step in matrix form as follows. We define Z_c (*resp*. $\bar{Z}_c$) as the set of zero- (*resp*. non-zero-) valued genes in profile x_c of the c-th cell. We can now write the imputation layer as:

$$v_c = f_I(y_c) = P_{Z_c}[r(W_I y_c + b_I)]$$

where the imputation layer $f_I(\cdot)$ is the triple composition of a linear transform (with sparse weight matrix $W_I = [w_1^T; \ldots; w_n^T] \in \mathbb{R}^{N \times N}$ and bias $b_I \in \mathbb{R}^N$), an ReLu $r(\cdot)$, and "entry-sampling" operator P_{Z_c}. P_{Z_c} sets to zero all entries not in Z_c, as we are only interested in imputation for zero-valued genes in the profile x_c (*i.e.*, for any vector $r, P_{Z_c}(r_j) = r_j \; if \; j \in Z_c$ and $P_{Z_c}(r_j) = 0, j \notin Z_c$).

After obtaining imputed vector v_c, we obtain a corrected single-cell expression profile: $\hat{x}_c = x_c + v_c$. This corrected single cell profile $\hat{x}_c$ can be used as the input to the encoder layer to learn an updated latent representation through the encoder-decoder framework. Because missing values have been estimated in $\hat{x}_c$, the new latent representation is expected to be better than the original one encoded from x_c.

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. 7** shows three steps). Then, the whole recurrent scScope framework can be described as:

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

for iterations $t = 1 \ldots T$. At the first step, the correcting layer's output v_c^0 is set as zero. We note that if we only consider one learning step (T =1) then scScope is a standard normal auto-encoder network.

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

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

The entry-sampling operator $P_{\bar{Z}_c}$ forces loss computation only on (non-zero) measured entries of x_c. The parameters in the encoder layer (f_E), decoder layer (f_D) and imputation layer (f_I) are all learned by minimizing the above loss function. We note that scScope learns its informative latent representation (h_c^T) in an unsupervised manner. scScope requires no extra information (*e.g.* cell type) to accomplish its deep-learning training.

Batch training

scScope training can be dramatically accelerated by using multiple GPUs (**Fig. 1b**). We replicate the same scScope network structure on different GPUs and aggregate all network parameters on the CPU. These network parameters include the connections and biases of all encoder, decoder and imputation layers of scScope. In a round of batch training, one GPU grabs the current network parameters from the CPU to use for its own network replicate of scScope. Then, for gradient calculation, the GPU processes a randomly chosen batch of m (= 64 or 512) single-cell expression profiles from a total of n single-cell profiles.

We apply a conventional gradient calculation framework for neural networks, which iteratively performs feed-forward and back propagation steps. In the feed-forward (FF) step, a GPU passes its batch of m single-cell samples through its locally stored scScope network and accumulates the losses for this batch. In the back propagation (BP) step, batch-dependent gradient information for network parameters on different layers is calculated by sequentially propagating accumulated loss from the end to the first network layer. This BP operation is performed by using gradient calculation functions wrapped in deep-learning packages (in our case TensorFlow). We apply this process independently across all k GPUs in a parallelized manner to obtain gradient information from a total $k \times m$ samples. The gradient information of those k GPUs is averaged by the CPU, *i.e.* $G^{(j)} = (G_1^{(j)} + \cdots + G_k^{(j)})/k$, where $G_k^{(j)}$ is the gradient calculated from the kth GPU in the jth round of optimization iteration. Finally, we apply adaptive moment estimation (ADAM)[23] with default TensorFlow[24] parameters to update the network parameters stored on the CPU. Iterations were terminated when either the objective function showed little changed (*i.e.* < 0.1%) or the number of iterations reached a maximal epoch (*e.g.* 100).

Cell subpopulation discovery

Representations outputted by scScope can be integrated with any clustering methods. Here we used the graph-based method PhenoGraph[15], 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. 8**) 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. 2**). 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 (**Supplementary Fig. 5**) without loss of accuracy (**Supplementary Fig. 9** and **Methods)**.

2. Imputing dropout genes

This output layer of the scScope neural network provides a complete set of gene profiles with all entries filled. Accordingly, we directly used filled values of this layer to impute missing results.

3. Implementation of comparison methods

Principal component analysis (PCA): The PCA algorithm was performed using the python-based machine-learning package *Scikit-learn*.

Single-cell Variational Inference (scVI): We used the Torch-based Python package of scVI. We followed the same parameter setting in original study[18] and setting the latent dimension to 50.

Zero-inflated factor analysis (ZIFA): We used the ZIFA python package with default parameters to extract 50-dimension features[9]. Specifically, when evaluating data imputation performance, ZIFA will remove genes where more than 95% cells have expressions lower than 1e-6 before imputation due to the high sparsity of the data.

Single-cell interpretation via multi-kernel learning (SIMLR): We used the Python implementation of SIMLR with the authors' default parameter settings. 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[6].

Sparse module activity factorization (SMAF): We used the Python package for SMAF and used the default parameters to compress gene expressions to 50-dimension features[11].

scScope: scScope was implemented in Python 3.6 with *TensorFlow-GPU* 1.4.1, *Numpy* 1.13.0, *Scikit-learn* 0.18.1 packages and was tested on a server with 4 GPUs (Nvida Titan X) and 64GB memory. For all experiments presented in this paper, we extracted 50-dimension representations under 2 recurrent learning steps (T = 2). If data are at moderate scale, we employ a medium batch size (64) and large epoch (100) in training; for large-scale data set, we use a large batch size (512) and limited epoch (10) to speed up the learning process.

Down-sampling training strategies on large-scale dataset: It is not possible to run directly single-cell packages ZIFA, SMAF and SIMLR on dataset with millions of cells. For these packages, we randomly down-sampled datasets containing more than one million cells

to a subset of 20K cells. 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. scVI can learn features on millions of cell profiles but the software does not provide a function for automatically clustering on such a large-scale dataset. Therefore, we used their deep learning framework for single-cell feature learning and our scalable graph clustering approach for large-scale clustering.

4. Evaluation of clustering performance

We used the adjusted Rand index (ARI)[25, 26] 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.

We used the Calinski-Harabaz score[27] to evaluate the tightness of the clustering results. This score is widely used when the ground truth is unknown. It is defined as the ratio of average between-clusters and within-cluster dispersions:

$$s = \frac{tr(B_k)/(k-1)}{tr(W_k)/(n-k)}$$

where n is the sample number and k is the cluster number. $tr()$ is the trace operation. B_k is the covariance matrix between clusters and W_k is the covariance matrix within clusters, defined as:

$$B_k = \sum_q n_q (c_q - c)(c_q - c)^T$$

$$W_k = \sum_q \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

For cluster q, C_q is samples in the cluster, n_q is the sample size and c_q is the cluster center. c is the center of all samples.

5. Evaluation of Imputation performance

The imputation accuracy was defined as the normalized distance between the imputed entries and the ground truth entries. We defined the set of ground truth values for those missing entries as a vector g and the imputed expression values from scScope from an inferred vector $\hat{g}$. The imputation accuracy (used in **Fig. 1d**, bottom) was defined as:

$$acc = \frac{\left|\left|\hat{g} - g\right|\right|^2}{\left|\left|g\right|\right|^2}$$

6. In silico studies

Simulation with balanced cell subgroups

Following previous studies, we initially tested the performance of scScope for cell-subpopulation discovery using simulated data[9, 10]. We assumed the high-dimensional single-cell expression data $x_c \in \mathbb{R}^N$ is generated or controlled 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 $b = 0.5$ is 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[28] α:

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

where x_{cg} denotes the gth gene of x_c, q_{cg} was randomly sampled in [0, 1], and δ =1 if its argument is true and = 0 otherwise. This double-exponential model is motivated by the widely-used assumption that low-expressed genes have higher probability to be influenced by dropout events.

We use the aforementioned generative model to create N single-cell profiles $x_i^{obs}, i = 1 \ldots N$. In the reported results (**Fig. 1c-h**, and **Supplementary Fig. 1**), we repeat such random generation process for 10 times on n =10K and 1M simulated single-cell profiles under various conditions for performance comparisons among different approaches.

Simulation with rare cell subgroups

To generate cell subpopulations with rare cell types, *i.e.* cell clusters with very limited numbers of cells compared to the major clusters, we sample 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. With this minor modification from the previous section, we generate imbalanced cell-type compositions with k_m rare cell types.

In our simulation results reported in **Fig. 1g**, we generate large-scale, imbalanced datasets with 1M single-cell profiles according to the following parameters: $\alpha = 0.5$ (dropout rate), $k = 50$ (total number of clusters), $k_m = 5$ (the number of rare cell types) and varied the rare cell type mixture rate *q* in the range from 0.1%~0.4%. 10 replicates for each condition were analyzed. We randomly down-sample these 1M single cell profiles to 10K. On the subsampled datasets, after learning features we applied PhenoGraph for *de novo* cell subpopulation discovery. On the unsampled datasets, we used our scalable graph clustering approach (**Methods**). We note that subpopulation numbers are automatically determined by these clustering approaches. The cell-subpopulation discovery accuracy are calculated and reported (**Fig. 1f-g**).

7. Analysis of biological data sets

a) Retina data

Data were obtained from Gene Expression Omnibus (GEO) database (accession number: GSE63473). 44,808 Cells were captured from retinas of 14-day-old mice and sequenced by DropSeq. In order to be comparable with published results, we followed the same experimental procedures to select 384-most variable genes[6] and then to log transform their expression (log(TPM + 1)). After clustering, we annotated clusters obtained by each method using the same maker genes in the original study[6].

We identified a candidate cell-type based on the highest average type-specific marker expression (**Supplementary Table 3**). For each cluster, we calculated the fold-change

values of all cell-type markers, and if at least one of a 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”.

b) Gut data

Intestinal epithelial cells were captured from 10 mice and sequenced using droplet-based scRNA-seq (GEO accession: 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 7**). 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 8**). 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.

(c) Brain data

Data were obtained from 10x 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)[6] 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.

Due to the massive scale of the dataset, we set the batch size of scScope to 512 and trained the model for 10 epochs. 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 6**).

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.

8. Evaluation of the density down-sampling accuracy

To evaluate the accuracy of the proposed density down-sampling clustering method, we randomly selected a subset (n = 5,000, 10,000, 20,000, 50,000, 100,000) of 1.3 million data and directly input corresponding scScope features to PhenoGraph for clustering. Then, we compared subset labels with labels obtained by density down-sampling clustering on the whole dataset. The comparison was repeated 100 times on different randomly selected subsets (**Supplementary Fig. 9**).

9. Software used in study

ZIFA: https://github.com/epierson9/ZIFA

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

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

SMAF: https://github.com/brian-cleary/CS-SMAF

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

10. Code availability

Code will be made public after peer-reviewed.