*Published online 31 July 2017*

*Nucleic Acids Research, 2017, Vol. 45, No. 17* ***e156***
*doi: 10.1093/nar/gkx681*

# Using neural networks for reducing the dimensions of single-cell RNA-Seq data

**Chieh Lin[1], Siddhartha Jain[2], Hannah Kim[3] and Ziv Bar-Joseph[1,3,*]**

[1]Machine Learning Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA, [2]Computer Science Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA and [3]Computational Biology Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Received April 13, 2017; Revised July 19, 2017; Editorial Decision July 21, 2017; Accepted July 24, 2017



## ABSTRACT

**While only recently developed, the ability to profile expression data in single cells (scRNA-Seq) has already led to several important studies and findings. However, this technology has also raised several new computational challenges. These include questions about the best methods for clustering scRNA-Seq data, how to identify unique group of cells in such experiments, and how to determine the state or function of specific cells based on their expression profile. To address these issues we develop and test a method based on neural networks (NN) for the analysis and retrieval of single cell RNA-Seq data. We tested various NN architectures, some of which incorporate prior biological knowledge, and used these to obtain a reduced dimension representation of the single cell expression data. We show that the NN method improves upon prior methods in both, the ability to correctly group cells in experiments not used in the training and the ability to correctly infer cell type or state by querying a database of tens of thousands of single cell profiles. Such database queries (which can be performed using our web server) will enable researchers to better characterize cells when analyzing heterogeneous scRNA-Seq samples.**

## INTRODUCTION

Single cell RNA-seq (scRNA-seq) which profiles the transcriptome of individual cells (as opposed to ensemble of cells) has already led to several new and interesting findings. These include the level of heterogeneity within a population of cells (1), the identification of new markers for specific types of cells (2) and the temporal stages involved in the progression of various developmental processes (3).

While promising, single cell data have also raised new computational challenges. An important and exciting application of single cell sequencing is the ability to identify and characterize new cell types and cell states (4,5). Recent work has used single cell expression profiles to discover new cells in developing lungs (6), new brain cells (4) and to refine several aspects of cell state transitions in differentiation studies (7,8). A key question that all such studies had to address is how to determine the similarity of the expression profiles of a pair (or larger sets) of cells? Another application for which the ability to compare single cell expression data between cells is critical in retrieval of similar cell types. Consider an experiment in which a population of cells taken from a diseased individual, or from a tumor, is profiled. One question that may be important for such analysis is to identify the specific types of cells that are present in the sample that was profiled, for example to determine which immune cells may have penetrated the diseased tissue (9). While such analysis is often performed using markers, a much more comprehensive solution is to compare the various cell expression profiles to a set of curated single cells with known types.

In the above examples, comparisons or similarity analysis can either be performed using the measured expression values or after performing dimensionality reduction which may help reduce the noise associated with specific values. Indeed, several methods have been used and developed for performing such comparisons. The simplest, though one of the most popular, is based on principal component analysis (PCA). PCA has been used extensively for clustering single cells (1,10,11). Other groups have developed new methods which extend and improve PCA. These include pcaReduce (12), which uses a novel agglomerative clustering method on top of PCA to cluster the cells. SNN-Cliq (13) constructs a *shared k*-nearest neighbor (KNN) graph over all the cells with the weight of each edge being the difference between *k* and the highest averaged ranking of the common KNN between two cells. It then tries to find maximal cliques in that graph in order to cluster the cells. ZIFA (14) uses a dimensionality reduction technique that takes into account the dropout characteristics of single cell sequencing data. SINCERA provides a pipeline for the analysis of single cell gene expression data, one of whose tasks is to identify new

*To whom correspondence should be addressed. Tel: +1 412 268 8595; Email: zivbj@cs.cmu.edu

cell types (15). Clustering is done via hierarchical clustering using centered Pearson correlation as the similarity measure. SIMLR (16) is another open-source tool that performs dimensionality reduction and clustering based on a cell similarity metric.

While PCA and other unsupervised approaches have been successful, they have mostly been used to analyze datasets generated by a specific group. In contrast, for problems including retrieval we would like to obtain a reduced dimension for *all* cell types and experiments across different labs. In addition, PCA is an unsupervised method and so it is not directly trying to distinguish between specific cell types. Thus it may not be the best method for a discriminative analysis goal including retrieval of cell types based on their expression.

Here, we propose to replace PCA-based dimensionality reduction with a supervised method based on neural networks (NN). These networks are universal function approximators (17) and, while training such networks takes longer than the unsupervised methods discussed above, they are very scalable in terms of training when using GPUs. Past work has used unsupervised versions of NN for the analysis of bulk expression data. For example, Tan *et al*. (18) used denoising autoencoders (DAEs) to reconstruct bulk gene expression data. Using these networks, they were able to assign genes to clusters and find clusters enriched for genes that are differentially activated between strains of *Pseudomonas aeruginosas* during cellular response to low oxygen. Gupta *et al*. (19), use DAEs to analyze time series bulk gene expression data. In contrast, while we also use DAEs for pre-training our network, we fine-tune it with a supervised training pass afterwards to learn a latent representation that disentangles cell types.

Neural network models use multiple layers, often with fewer nodes than the number of input values. The networks are trained by maximizing a target function and in several cases, one of the intermediate layers with small cardinality (compared to the number of inputs) can serve as a reduced dimensionality representation for the input data (20). In our implementation we use gene expression values as the input and formulate a target function whose goal is the identify the correct cell type from the values computed by intermediate layer nodes (Figure 1). We tested various architectures for such networks, including architectures informed by prior biological data (such as protein–protein (PPI) and protein–DNA interactions (PDI)) and compared their performance to prior methods for analyzing single cell data. As we show, the learned networks captured several important biological aspects of the data. We used the values from the intermediate layer in the neural network (NN) for clustering and retrieval of scRNA-seq data. As we show, using the NN values improved performance when compared to using the measured expression data, various unsupervised methods for reducing the dimension of the data, and prior methods for clustering single cell data.

## MATERIALS AND METHODS

### Datasets used in our analysis

We collected a total of 33 datasets with more than 17 000 single cell expression profiles from published papers and from the Gene Expression Omnibus (21) (GEO). Supplementary Table S1 and the Supporting Website provide full details on these datasets. We used 3 of these datasets which, combined, profiled 16 different types of cells and a total of 402 single cells for initial training, testing and evaluation of the method. We used 31 of the 33 datasets for the retrieval analysis (to avoid using different datasets from the same lab). All datasets are used in the retrieval application which is available from the Supporting Website. We curated all 33 datasets and assigned cell type labels to all single cell expression profiles.

### Normalization and imputation

We tested a number of methods for normalizing data obtained from different labs and different platforms. We initially tested a novel normalization method for single-cell RNA sequencing data which is based on pooling across cells (22). However, while results for the clustering analysis using this method were similar to the results presented below (see Supplementary Figures S1 and 2), the method required us to manually set several parameters (such as pool size) which made it hard to use for the larger retrieval analysis. Instead, following prior work we normalized the data by converting all datasets to the the Transcripts Per Million (TPM) format (10,23–25).

To combine PPI, PDI and single cell data (see below), we have only used a subset of 9437 genes that were present in the three single cell training datasets and in the PPI and PDI datasets. For the much larger set of profiles used in the retrieval analysis, we used the same set of genes. Since these datasets were generated by different platforms and groups, counts for 2% of the genes were missing (on average) from each dataset. In order to use the NN method for analyzing these datasets, we performed imputation for these missing genes as follows. Missing values for the retrieval analysis were first assigned the median gene expression value for the cell and then imputed with the average expression value for the $k$-nearest neighbor genes (we used $k = 10$), where nearest neighbors were computed based on overall correlation (26). Following TPM normalization and imputation, each gene was normalized to the standard normal distribution across samples since this is an essential step for NN training. We choose not to do the log-transformation because we found that it did not help the performance.

To account for the drop-outs in the imputation procedure, we tested a probabilistic model that would randomly assign values for 0 to a specific fraction $z$ of the imputed genes instead of relying on the nearest neighbors for as discussed above. We tested several different values for $z$ including 0 (no drop outs), 0.01, 0.03 and 0.05 using a cross validation strategy. Our results indicate that the best performance is achieved using $z$ = 0 (see Supplementary Table S7) and so this is what we used for the rest of the paper.

### Protein–protein and protein–DNA interaction data

We used PPI and PDI data to determine the architecture of some of the NN we tested. We constructed a weighted, partially directed, protein interaction network using several databases including BIOGRID (27), HPRD (28) and also

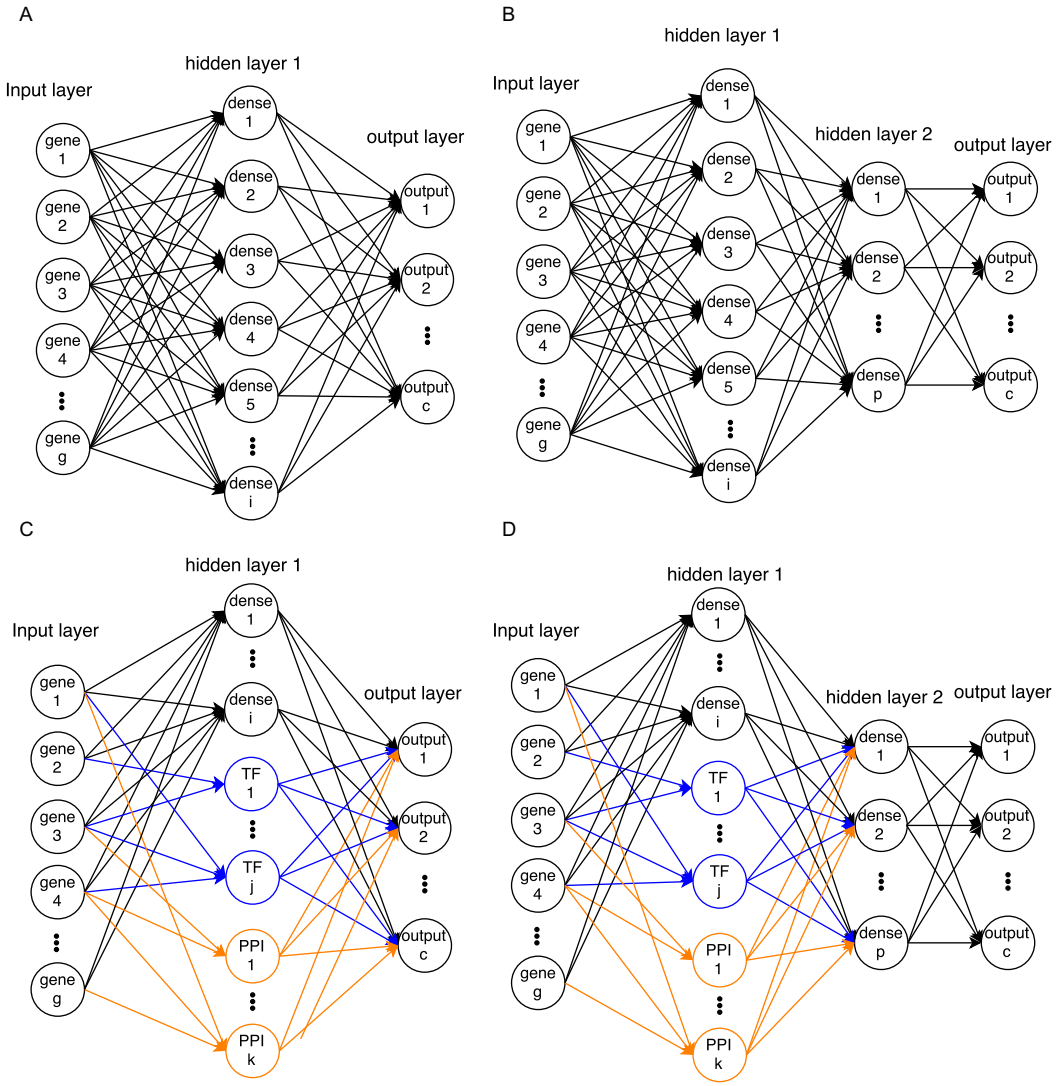


**Figure 1.** Network architectures of NN models used in this paper. (**A**) Single-layered fully connected network. (**B**) Two-layered fully connected network. (**C**) Single-layered network with TF and PPI cluster nodes (connected only to their member genes) and fully connected dense nodes. (**D**) Two-layered network that is similar to the model in (C) but with an additional fully connected layer. Note that bias nodes are also included in each model.

used Post-translational Modification Annotations from the HPRD. PDI were based on data from (29). The data contained 160 000 PPI edges and 60 000 PDI edges between 16 671 genes and 348 TFs. The PPI data were weighted based on the different types of experimental evidence supporting each interaction (30).

### A neural network representation of single cell expression data

We evaluated four types of neural network architectures (Figure 1), and trained a total of five models (detailed number of nodes for each model are shown in Table 1). All architectures include an input layer, one or two hidden layers (more hidden layers did not improve performance) and an output layer. The input layer encodes the expression values for the genes that are used in the analysis. The output layer encodes the probability of each cell type. Hidden layers are functions of the input and are used to efficiently encode the inputs such that the output can be the correct cell type given a set of expression values.

Specifically, we formulate our neural network model as follows. Let the vector $\boldsymbol{x}^{(i)}$ denote the output of $i$th hidden *layer*. We use $\boldsymbol{x}^{(0)}$ to represent the input of NN. To compute $\boldsymbol{x}^{(i)}$, we perform forward propagation: (Equation 1)

$$\boldsymbol{x}^{(i)} = a(\boldsymbol{W}^{(i)}\boldsymbol{x}^{(i-1)} + \boldsymbol{b}^{(i-1)}) \quad (1)$$

where, $a$ is the activation function, $\boldsymbol{b}$ is an intercept term and $\boldsymbol{W}$ is the weight matrix of each edges in the neural network. $\boldsymbol{W}$ and $\boldsymbol{b}$ are the parameters we need to learn.

We tested a number of possible activation functions including sigmoid, linear, relu and tanh. Our analysis indicates that the hyperbolic tangent activation function (tanh) (Equation 2) leads to the best performance among these, and so we used it in the remainder of this paper. The tanh

**Table 1.** The five different types of NN used in the paper

| No. | Model | Layer1#node | Layer2#node | #Parameters (million) |
|---|---|---|---|---|
| 1 | Dense | 796 | X | 7.52 M |
| 2 | Dense | 100 | X | 0.95 M |
| 3 | Dense | 796 | 100 | 7.59 M |
| 4 | PPI/TF+dense | 696 + 100 dense | X | 1.01 M |
| 5 | PPI/TF+dense | 696 + 100 dense | 100 | 1.08 M |

Note that No.1 and No.2 represent the same architecture (Figure 1A) using different number of hidden layer nodes. The 696 in the PPI/TF models is from 348 TFs + 348 PPI groups. The number of nodes in the Dense models corresponds to the number of nodes in PPI/TF models for comparison. The additional 100 dense in each model are manually selected.

function is defined as:

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \quad (2)$$

For the output layer, which performs discrete classification, we use the softmax activation function (Equation 3). Let $\boldsymbol{x}$ denote the input of the output layer (which is also the output of the last hidden layer), then we have the following:

$$\text{output}(\boldsymbol{x}) = \text{softmax}(\boldsymbol{x}) = \left[\frac{\exp(x_1)}{\sum_c \exp(x_c)} \cdots \frac{\exp(x_C)}{\sum_c \exp(x_c)}\right]^T \quad (3)$$

Where, C encodes indices for all the cell types in training set.

Here, the output $f(\boldsymbol{x}^{(0)})_c$ of each node $c$ in the output layer represents the probability:

$$f(\boldsymbol{x}^{(0)})_c = p(y = c|\boldsymbol{x}^{(0)}) \quad (4)$$

that the input sample $\boldsymbol{x}^{(0)}$ belongs to cell type $c$. The loss function is categorical cross-entropy function (Equation 5):

$$\text{loss} = -\log f(\boldsymbol{x}^{(0)})_y \quad (5)$$

where, $y$ is the true cell type (label) of the input $\boldsymbol{x}^{(0)}$.

### Architectures used in our NN method

While all networks used the same input and output layers, they differed in the structure and number of hidden layers. We tested models with 1 or 2 fully connected hidden layers (for the NN with 2 hidden layers, 796 nodes in the first hidden layer and 100 nodes in second hidden layer, Figure 1A and B). While these models have at most 2 hidden layers, the number of parameters that are fitted is very large (the architecture with the most parameters had 7.6 million parameters).

The above architectures do not make use of prior biological knowledge. To incorporate such information, we also tested architectures that were based on PPI and PDI. Unlike the fully connected layers in the architectures described above, when using the biological data we connect the nodes in the first hidden layer to only a small subset of the input nodes. We do this by fixing the weight to 0 for all connections from genes to PPI/TF nodes for which the genes do not belong to that node. This greatly reduces the number of parameters that needs to be learned which can help with issues related to overfitting. Specifically, we have 348 hidden layer nodes based on the PDI data (one for each TF, connected only to genes regulated by that TF) and 348 nodes for the PPI. To divide the PPI graph into 348 subnetworks (where each subnetwork is connected to a node in the hidden layer) we used the ClusterOne (31) algorithm, which produces overlapping clusters so that, similar to the dense nodes, a gene can be connected to more than one PPI node. We also added 100 fully connected nodes to the first hidden layer which account for missing data and TFs in the PPI and PDI—for instance if we do not have a set of targets for a specific TF that should be grouped together or if the PPI network is missing edges leading to missed clusters (PPI nodes). Still, the total number of parameters for this architecture is about 1 million, an order of magnitude less than for the fully dense architectures.

### Unsupervised pre-training

The discussion above focused on supervised learning of NN (where the label is the cell type). In addition to supervised learning NN can also use unlabeled data, a practice that has proven useful in other domains (32). A specific type of this NN is termed 'denoising autoencoder (DAE)' since the goal is to reconstruct the input layer values using a small number of hidden layer nodes (since the target is the input expression values, no labels are needed to train autoencoders). While unsupervised, autoencoders have been shown to successfully identify input combinations that affect the overall set of values (33–35). Given the large number of parameters in a NN the ability to train autoencoders and use the parameters learned as priors for a supervised learning procedure improves the initialization of the model and often leads to better generalization performance (32). We have thus tested the use of 1-layer DAE when testing the method on large datasets (retrieval datasets). We train DAE to reconstruct the original input from corrupted input with the noise sampled from the Normal distribution with zero mean and 0.1 standard deviation. The architecture of DAE is similar to Figure 1A except that the output layer is changed to be the reconstructed input. All layers in the DAE use the tanh activation function and mean square error as the loss function. Here we used 100 and 796 nodes in the hidden layer of the DAE, similar to the numbers used for the supervised models to make sure that the weights of DAE can be used as pre-trained weights of supervised models. We copy the trained weights of DAE to the dense nodes of our models.

### Learning model parameters

All models were implemented using Keras ( https://github.com/fchollet/keras) with some modifications to accommodate the sparse layer connections for TF and PPI nodes. All

the NNs are first initialized with Glorot initialization (36). The models are trained using stochastic gradient descent with Nesterov accelerated gradient with a learning rate of 0.1, decay $10^{-6}$, momentum 0.9. The learning rate for epoch K is the original learning rate multiplied by 1/(1+decay*K). We used 100 epochs (which were manually selected and enough for reaching convergence) to train each model with a mini-batch size of 10, which is the number of samples to fit at each update step. Detailed information about the dimensions of the different architectures is provided in Table 1. It took us 30 s to train the largest supervised NN (7.6 million parameters, 402 cells), 40 min to train the largest unsupervised NN (15 million parameters, 17 000 cells) on a machine with 4 Intel(R) Xeon(R) CPU E5-2620 v3 (2.40GHz each core, 24 cores in total), 4 Nvidia GTX 1080 GPUs and 128 GB RAM. Note that each NN model including the one used for the final outcome of our method (retrieval analysis in Table 4) is only trained using a single GPU.

### Biological analysis of learned models

To determine the biological relevance of the parameters learned from the NN we analyzed significant gene groups for each cell type in the PPI/PDI model (Figure 1C). For this we identify the top 10 most highly weighted (hidden layer) nodes for each output layer node (corresponding to the different cell types). Some of the selected nodes are explained by the TF or the PPI they represent. For the other (100 nodes in the hidden layer initially connected all input genes) we perform GO analysis based on the set of input genes that are connected to these nodes with a high (absolute value) weight. We used gprofiler (37) to perform GO analysis because it provides command-line interface for access.

### Comparisons to prior clustering and dimensionality reduction methods

To perform dimensionality reduction based on the NN results we extract the values computed by the last hidden layer of each architecture we tested. We next use a simple clustering method (K-means++ clustering (38)) to perform unsupervised grouping of cells using a test set (not used in the NN training) and the results are compared to prior methods suggested for clustering single cell expression data. For such comparisons, we perform experiments in which we left out 2, 4, 6 or 8 random cell types of the 16 types in our analysis set. We next cluster the left out data using the reduced representation obtained from the last hidden layer of the NN and use the adjusted random index (ARI) (39) to compare the clustering results with the true labels. ARI counts the number of agreements and disagreement between two groupings while adjusting for random performance into account. It is defined as follows. Let $X = \{X_1, X_2, ..., X_r\}$, $Y = \{Y_1, Y_2, ..., Y_s\}$ be two groupings. We can summarize the overlap between $X$ and $Y$ using a table $N$ where $N_{ij} = |X_i \bigcap Y_j|$ is the number of objects in common. Let $a_i = \sum_j N_{ij}$, $b_j = \sum_i N_{ij}$, $n$ be the total number of samples, then we set:

$$\begin{aligned} \mathrm{ARI} &= \frac{\mathrm{Index} - \mathrm{ExpectedIndex}}{\mathrm{MaxIndex} - \mathrm{ExpectedIndex}} \\ &= \frac{\sum_{ij}\binom{N_{ij}}{2} - \left(\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right)/\binom{n}{2}}{\frac{1}{2}\left(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right) - \left(\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right)/\binom{n}{2}} \end{aligned} \quad (6)$$

We repeat the experiment for 20 times with fixed random seeds 0–19, each time with different random left out cell type and obtain an average ARI value. We also test other scoring methods for clustering: adjusted mutual information (40), Fowlkes–Mallows score (41) , homogeneity, completeness and v-measure scores (42).

### Cell retrieval method

To measure the performance of cell retrieval, the percentage of the desired cell type in the top $k$ nearest neighbors (here, we used $k = 100$) is calculated for different single cell expression data representations (NN, PCA, measured values). To reduce the ability of the retrieval analysis to rely on artifacts for correctly identifying the cell types (e.g. experiments from the same lab) we only perform this analysis on cell types that were profiled in *different* datasets out of the 31 used for this analysis. We thus used 9 different cell types for this analysis though the database itself contains over 100 cell types (including subtypes). Cell types that are not selected can thus still be in the retrieved set of $k$ nearest cells which makes the analysis more similar to how we envision the usage of such database. We use the mean of average precision (MAP) to evaluate the retrieval performance. Average precision corresponds to the area under the precision-recall curve for the retrieved cells. The final result for all cell types is a weighted mean so that every dataset (cell type) has equal weight.

### Hyper parameter selection

To select hyper parameters, we split the data into a training and test set randomly with 90% (or 80%) of the data used for training and 10% (or 20%) of the data for testing. We repeat this process 10 times and select the best hyper parameter based on the average test performance in these 10 random experiments. Note that since we evaluate the methods in an unsupervised (clustering) setting, this training uses a completely independent dataset from any of the data we use for evaluation. The hyper parameters for the learning rate affects the training time but it does not affect the performance very much. We performed additional analysis to test the influence of different rates on accuracy. We explored different hyper parameters for a dense model with combinations of learning rates [0.01, 0.1], momentum [0.5, 0.9] and decay [0.001, 1e-06]. The results, presented in Supplementary Table S9, are very similar for all these values.

## RESULTS

We learned parameters for the different NN architectures discussed in the 'Materials and Methods' section and used the resulting models to test the method and to compare it to prior methods for dimensionality reduction of scRNA-Seq data and for clustering and comparing such data.

### Testing and comparing the NN method

To test our method and to compare it to prior methods for clustering single cell time series expression data we used 3 single cell expression datasets which, combined, included 16 cell types (See Supplementary Table S8 for complete list). Data were downloaded mostly from GEO and processed as discussed in 'Materials and Methods'. We identified 9437 genes that had expression values in all training datasets and used them to learn the NN using various architectures. For each architecture, input values were directly connected to the first hidden layer and the output layer predicted the true cell type (label) for each of the datasets. Thus, the goal of the NN was to identify a reduced dimension representation of the expression values (where the number of nodes/values is a function of the specific architecture) that leads to the most accurate assignment of cell type for each dataset. Training and testing accuracy of all NN models reached nearly 100%.

Given the ability of the method to accurately assign training data, we next asked how well the resulting representation can be used on test (unobserved) data. For this we performed a number of different analyses in which we divided the single cell datasets using subset of the cell types to learn the model and the rest to test it. For testing, we compared the ability of a simple clustering method ($k$-means++ algorithm with $k$ representing the known number of left out cell types) to accurately sub-divide test data according to their cell types. For this we first learn parameters for the NN using the portion of cells used for training. Next, for each of the test cells we run them through the NN and extract the values of the smallest hidden layer (depending on the architecture) and use these vectors in the clustering procedure. Thus, clustering is based on gene expression combinations determined by the NN. We also note that the training and test sets are comprised of different cell types and so this is not a supervised classification task. Instead, the goal is to see if the parameters learned using the labeled data can improve the *unsupervised* analysis of the other cell types. We also calculate the performance of pre-trained models, and the result are similar to the model without pre-training. We used the clustering results to compare the NN-based analysis to a number of unsupervised clustering and dimensionality reduction methods that have been proposed for single cell analysis. Specifically, we compared our method to PCA, pcaReduce, Independent Component Analysis (ICA), Non-negative matrix factorization (NMF), t-distributed stochastic neighbor embedding (tSNE) ,SINCERA, SIMLR and SNN-Cliq. For the unsupervised dimensionality reduction methods (PCA, ICA, NMF, tSNE) we tested several possible settings (number of components). Note that we were unable to test more than 10 components for tSNE and ICA since these methods generated run time errors (NaN, Inf values) when using more than 10 components. As for the other methods, PcaReduce generates hierarchical structure of clustering based on PCA reduced dimensions. SINCERA provides a pipeline of analyzing single cell data, including clustering. For clustering, SINCERA uses hierarchical clustering, tight clustering (43) or consensus clustering (44). We only show the result of hierarchical clustering here because it is the default setting of SINCERA and the other clustering methods often generated error messages when applied to our dataset. SNN-Cliq uses shared nearest neighbors (SNN) to define the similarities between cells and then clusters them using graph based methods. SNN-Cliq sometimes generates error messages when the number of cell types ($k$) is 2, so we left it out in this comparison. We also tried to compare our results to ZIFA. However, ZIFA did not finish after running for two days when trying to cluster 300 cells with 9437 expression values each. To improve its run time we reduced the number of genes to 1356 (selecting only genes that have non-zero values in 90% of samples) but the performance of ZIFA on this data was not good (much lower than the results presented in the comparison table) and so we left it out. Table 2 presents a comparison of the clustering results for all methods we tested and Figure 2 presents a visualization of these results for a few of the methods. As can be seen, in general the variants of the NN we tested outperformed all other methods. One way to explain this result is the fact that unlike other methods our method learns the best way to represent a reduced dimension for single cell data (even though the comparison is based on an unsupervised clustering task and learning is done on a completely different set of experiments and cell types) whereas the other methods are fully unsupervised. While all NN architectures performed better than other methods, the best was the PPITF variant that integrated prior biological knowledge and dense nodes which did slightly better than the Dense 796 node networks. However, as we show below, when testing on a much larger set of data the architectures that incorporate prior biological knowledge do better than the fully dense ones.

### Functional analysis of hidden layer nodes

While NN are often treated as 'black boxes' following recent studies we attempted to characterize the input and hidden nodes (genes/TFs) that had the highest impact on the ability to accurately classify various cell types (18). Such analysis can provide both, functional interpretation of the parameters that are learned for the method as well as characterization of the different types of genes and their regulators that are most relevant to specific types of cells. To perform such analysis, we analyze the top 10 most highly weighted groups (hidden nodes) for each cell type using the NN in Figure 1C. To analyze the groups we either used the known characteristics of the node (since, as mentioned in 'Materials and Methods' some of the nodes represent groups of genes known to be co-regulated by the same TF or to be interacting (PPI)) or GO analysis on densely connected nodes to determine the function of genes associated with that node. Table 3 presents results for a subset of the cell types and nodes. As can be seen, the top scoring nodes were often very relevant for the specific functions that are performed by the cells or the TFs that regulate these functions. For example, the learned NN correctly associated genes related to proliferation and differentiation with ES cells while nodes that scored high for immune response categories were mainly associated with Bone Marrow-derived Dendritic Cells (BDMCs) (45–49). See also Supporting Website for tables summarizing more than 1000 significant GO categories for combinations of cell types and nodes based on this analysis.

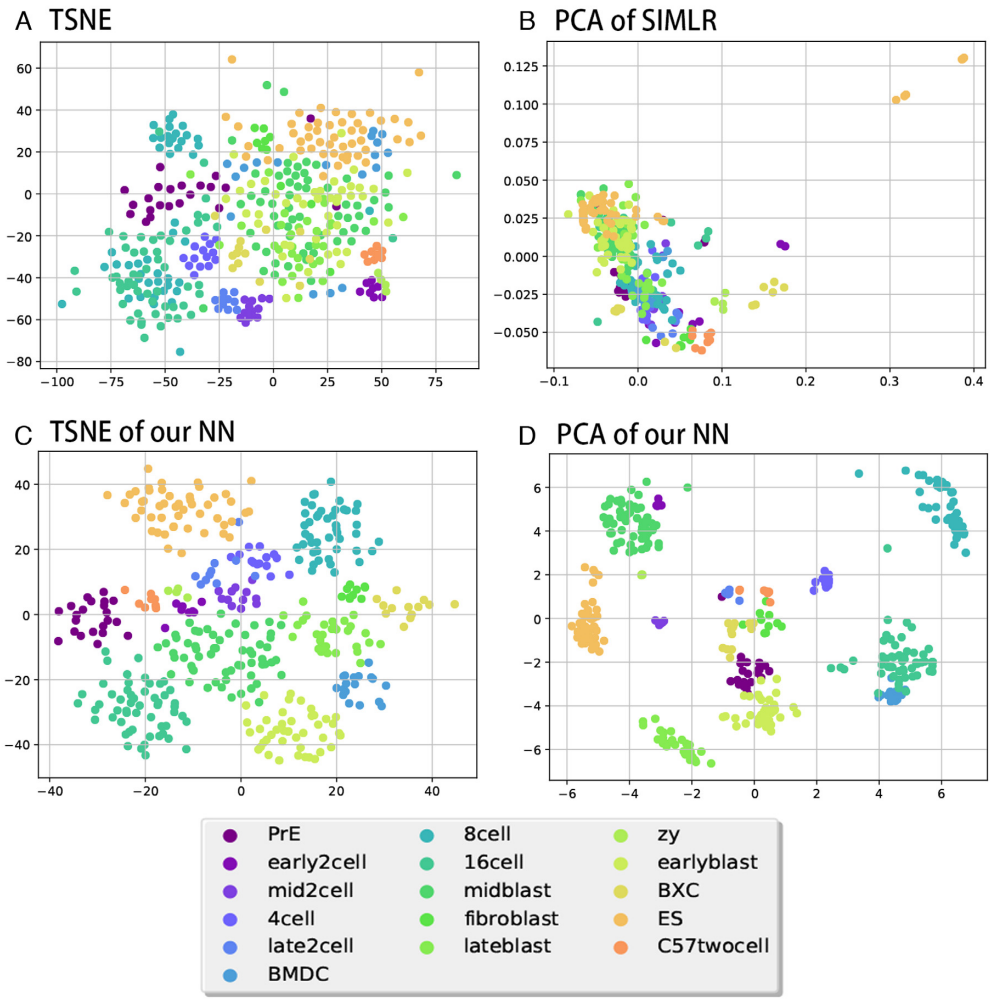


**Figure 2.** 2D visualizations for some of the methods in Table 2. Colors represents cell types as shown below the figures. (**A**) 2D TSNE of the original data. (**B**) 2D PCA for SIMLR-transformed data. (**C**) 2D TSNE of the second layer of the PPI/TF NN. (**D**) 2D PCA of the second layer of the PPI/TF NN.

In addition to the analysis of connection weights for the NN that were learned using labeled data, we also analyzed the values for the nodes obtained by the pre-trained models. Recall that for the pre-training we are using a fully unsupervised approach where the goal is to reconstruct the input data using a small set of highly connected nodes (100 in our case). We hypothesized that some of the values learned for these nodes may actually reflect key genes that either regulate other genes (and so their values are enough to reconstruct the full expression set) or important groups of co-expressed genes. We thus selected, for each of the 100 dense nodes in the pre-trained mode, the set of three most similar genes based on Pearson correlation. The results are presented in Supplementary Table S11. As can be seen, many of these genes are ribosmal genes which are indeed a large and coherently expressed group that is captured by some of the nodes. GO analysis of the selected nodes (Supplementary Table S12 shows part of the results.) indicated that a a significant category for these is 'nucleic acid binding' (corrected $P$-value $= 9*10^{-9}$) indicating that the model captures some of the TFs that are likely regulating the expression in the different single cell types. See Supporting Results and Website for complete details.

In addition to edge weight-based analysis discussed above, we have also performed an analysis of the significant input genes for each cell type analysis using DeepLift (https://arxiv.org/abs/1704.02685). Instead of focusing directly on the parameters learned by the model for such analysis, DeepLift works by comparing the activity in each internal node between a positive input (correct cell type) and a reference input (a vector of zeros in this case) and then propagates these differences to all nodes in the network including input nodes. Thus, DeepLift can directly assign significance to each of the input nodes (genes). Using DeepLift we selected the top 100 genes for each cell type and performed similar gprofiler analysis on this set. The results agreed in many cases with the edge weight analysis discussed above. For example, for ES cells both analyses correctly identified E2F3 and IRF6 as two of the top TFs (though the edge weight based method had better $P$-values for these). Similarly for BDMCs both identified immune response as the top function (though here DeepLift had slightly better $P$-values for one function). See Supplementary Table S10 for

**Table 2.** Average performance of different scoring metrics for clustering four cell types not used in the NN training

| Feature | Homo | Comp | Vmes | ARI | AMI | FM | Average |
|---|---|---|---|---|---|---|---|
| Original | 0.785 | 0.875 | 0.82 | 0.73 | 0.775 | 0.841 | 0.805 |
| pca 2 | 0.833 | 0.883 | 0.854 | 0.786 | 0.821 | 0.873 | 0.842 |
| tsne 2 | 0.179 | 0.169 | 0.172 | 0.113 | 0.127 | 0.391 | 0.192 |
| ica 2 | 0.833 | 0.882 | 0.853 | 0.785 | 0.82 | 0.873 | 0.841 |
| nmf 2 | 0.659 | 0.735 | 0.69 | 0.592 | 0.638 | 0.75 | 0.677 |
| pca 5 | 0.754 | 0.863 | 0.798 | 0.685 | 0.741 | 0.814 | 0.776 |
| tsne 5 | 0.034 | 0.035 | 0.034 | -0.009 | -0.006 | 0.332 | 0.07 |
| ica 5 | 0.662 | 0.839 | 0.733 | 0.615 | 0.649 | 0.787 | 0.714 |
| nmf 5 | 0.707 | 0.848 | 0.76 | 0.693 | 0.689 | 0.824 | 0.753 |
| pca 10 | 0.759 | 0.873 | 0.805 | 0.695 | 0.747 | 0.823 | 0.784 |
| tsne 10 | 0.049 | 0.043 | 0.045 | 0.0 | 0.004 | 0.299 | 0.073 |
| ica 10 | 0.452 | 0.733 | 0.545 | 0.416 | 0.431 | 0.706 | 0.547 |
| nmf 10 | 0.717 | 0.812 | 0.751 | 0.676 | 0.688 | 0.812 | 0.743 |
| pca 50 | 0.692 | 0.867 | 0.759 | 0.637 | 0.68 | 0.799 | 0.739 |
| nmf 50 | 0.481 | 0.625 | 0.532 | 0.427 | 0.457 | 0.681 | 0.534 |
| pca 100 | 0.742 | 0.879 | 0.795 | 0.695 | 0.731 | 0.829 | 0.779 |
| nmf 100 | 0.397 | 0.635 | 0.472 | 0.358 | 0.37 | 0.672 | 0.484 |
| pcaReduce | 0.768 | 0.89 | 0.821 | 0.747 | 0.754 | 0.849 | 0.805 |
| SIMLR_20 | 0.793 | 0.806 | 0.799 | 0.718 | 0.77 | 0.82 | 0.784 |
| SIMLR_30 | 0.806 | 0.82 | 0.811 | 0.747 | 0.779 | 0.838 | 0.8 |
| SIMLR_40 | 0.834 | 0.831 | 0.83 | 0.747 | 0.797 | 0.835 | 0.812 |
| SNN-Cliq | 0.751 | 0.905 | 0.802 | 0.716 | 0.726 | 0.843 | 0.79 |
| sincera_hc | 0.807 | **0.929** | 0.856 | 0.797 | 0.794 | 0.884 | 0.845 |
| Dense 1 layer 100 | 0.905 | 0.897 | 0.9 | 0.872 | 0.885 | 0.915 | 0.896 |
| Dense 1 layer 796 | 0.895 | 0.887 | 0.89 | 0.856 | 0.874 | 0.904 | 0.884 |
| Dense 2 layer 796/100 | 0.892 | 0.882 | 0.886 | 0.854 | 0.869 | 0.903 | 0.881 |
| PPITF 1 layer 696 + 100 | 0.897 | 0.896 | 0.896 | 0.866 | 0.882 | 0.911 | 0.891 |
| PPITF 2 layer 696 + 100/100 | **0.906** | 0.902 | **0.903** | **0.874** | **0.89** | **0.917** | **0.899** |
| Dense 1 layer 100 pre-train | 0.883 | 0.871 | 0.877 | 0.838 | 0.858 | 0.892 | 0.87 |
| Dense 1 layer 796 pre-train | 0.887 | 0.872 | 0.879 | 0.843 | 0.86 | 0.896 | 0.873 |
| Dense 2 layer 796/100 pre-train | 0.884 | 0.875 | 0.879 | 0.838 | 0.862 | 0.894 | 0.872 |
| PPITF 1 layer 696 + 100 pre-train | 0.878 | 0.873 | 0.875 | 0.83 | 0.858 | 0.889 | 0.867 |
| PPITF 2 layer 696 + 100/100 pre-train | 0.9 | 0.894 | 0.897 | 0.87 | 0.881 | 0.914 | 0.893 |

Results are averaged over 20 clustering experiments (using different random initializations). AMI: adjusted mutual information; ARI: adjusted random index; Comp: completeness; FM: Fowlkes–Mallows; Homo: homogeneity; Vmes: v-measure. Bold represents the highest value for each column.

**Table 3.** Examples of highly ranked nodes for some of the cell types used for learning of the NN

| Cell type | TF/PPI/dense node | Corrected *P*-value | GO function/reference |
|---|---|---|---|
| Stem cells (ES) | Dense 35 | 3.25E-07 | Cell differentiation |
| Stem cells (ES) | Dense 24 | 4.44E-15 | Factor: E2F-3; (45) |
| Stem cells (ES) | Dense 24 | 5.78E-12 | Factor: IRF6; (46) |
| Stem cells (ES) | Dense 24 | 2.53E-08 | System development |
| BMDC | Dense 10 | 1.56E-05 | Immune system process |
| BMDC | Dense 67 | 1.59E-11 | Positive regulation of immune system process |
| BMDC | Dense 36 | 2.06E-06 | Response to cytokine |
| Fibroblast | ppi 223 | 3.16E-06 | Focal adhesion (47) |
| Fibroblast | ppi 301 | 2.85E-20 | Acetyltransferase complex (48) |
| Zygote | ppi 10 | 6.85E-07 | Regulation of cell proliferation |
| Zygote | ppi 280 | 7.36E-09 | Cell junction |
| Zygote | TF: foxd3 | X | TF: foxd3 (49) |

Some of the nodes were based on TF–gene interactions and thus represent a specific TF (Foxd3). For these, we rely on the function of the TF to characterize the node in the table. Other nodes are either based on PPI (e.g. PPI223) or on groupings learned by the algorithm (e.g. dense24). For thes, we performed GO analysis on the set of highly ranked genes for these nodes. Several other relevant TFs and functions were found for other cell types as well. See Supporting Website for complete list.

a comparison of *P*-values for DeepLift and our edge weight analysis and Supporting Website for the full DeepLift results.

### Retrieval of single cells based on expression

In most single cell studies, including cancer (23), brain studies (50–53) and more, several different types of cells are profiled in the same experiment. In most cases at least some of the cells cannot be fully assigned since either they do not contain any of the known markers or they contain several subsets of such markers. Such cells are usually analyzed using clustering to identify several groups within the sampled cells. However, such analysis is unsupervised and so its not always clear what each of the clusters corresponds to. Identifying the composition of cells is important, for example in cancer studies where notable differences between outcomes have been attributed to the amount of immune cells that are

present in the tumor. Thus, an important problem in single cell expression analysis is assignment. One way to address this problem is to compare uncharacterized cells to cells that have already been characterized by prior experiments either using follow-up studies or because of their known origin. To enable such analysis we collected 31 single cell expression datasets with more than 17 000 samples and created a database in which we stored both the expression measurements for these cells as well as the assignment for the cell from the paper/experiment in was profiled in (if available). Using such database we can find, for each new cell profiled in an experiment, the most similar cells in the database and based on their annotations, annotate the uncharacterized cells.

A key issue for implementing such strategy is how to identify the most similar cells in the database when given a new query cell. An alternative to using the measured expression values, which is more efficient and may be more robust to noise is to first reduce the dimension of the database data and the query cell and then perform the retrieval search in the reduced dimensional space. Such reduction can be done by several methods as discussed above and can cut the storage requirements and run time by over 80%, depending on the architecture used. Specifically, for the 14 000 queries we performed on the 17 000 cells in the database we reduced the run time from 25 min (when using the observed expression values for 9437 genes) to <5 min when using the 796 features obtained from the NN. The datasets of the query cell types are listed in Supplementary Table S2.

In addition, and most importantly, such reduced dimension greatly improves performance. To test various ways of querying single cell expression data we held out complete datasets for which we had a similar dataset in the database from a different lab/paper. This helps ensure that results are not effected by unique lab methods but are rather related to biological similarities. We identified 31 different datasets that, combined, profiled close to 14K cells. For each of these held out datasets we searched for the most similar cells in our database. As before, we compared the NN results to results obtained when using the measured expression values and PCA with 100 or 796 dimensions. As for the NNs, we tested two NN variants. The first is similar to the one described above for clustering while the second is based on using DAE to initialize the parameters of the network (using unlabeled data) followed by supervised learning as discussed in 'Materials and Methods'.

To evaluate the different methods, for each query cell we identify the top $k$ most similar cells in the database (where similarity is based on Euclidean distance for all genes or for the reduced dimension representation obtained by each method). We use $k = 100$ in this paper, though results are similar when using $k = 10$. We next use the top $k$ matches to compute the MAP for the correct cell type for each query ('Materials and Methods').

As can be seen in Table 4, all methods that relied on reduced dimensions did much better than the method that used the measured expression values (20% average improvement for PCA and almost 40% improvement for some of the NN methods when compared to using the measured expression values for all genes). Comparing the reduced dimensionality methods themselves, we observe that NN with more hidden layers (in our case 2) are doing, on average, better than NN with a single hidden layer indicating that non linear relationships may be important for characterizing single cell expression. These multi-hidden layer NN are also performing better than PCA.

We also observe that both, the use of prior biological knowledge to define the NN architectures (PPITF networks) and the use of pre-training using DAE improves the overall accuracy of the retrieval. Specifically, the best performing method (achieving an improvement of more than 11% over PCA) is the PPITF 2 layer 696+100/pretrain which combines all these features (2 layers, pre-training and the use of prior knowledge). Other architectures that use prior knowledge are also better than their dense counterparts. In contrast, DAE on their own (fourth and fifth rows) are not as effective as supervised models and so they are probably best for initializing rather than for final model selection.

## DISCUSSION AND FUTURE WORK

While single cell analysis holds great promise, it also raises new questions. Given the number of cells that are profiled in each experiment, which can reach thousands (51,54), new methods are required for accurately and efficiently clustering these data while overcoming issues related to the stochastic nature of gene expression even in similar cells, noise and missing values. A related problem is the ability to compare expression profiles from cells so that cell type assignments can be determined not just based on a few marker genes but rather based on the overall expression of all genes in the cells profiled.

In this paper, we developed and tested solutions based on deep neural networks for these problems. The advantage of such networks is that they can learn the importance of different combinations of gene expression levels for defining cell types and such combination are usually more robust than values for individuals genes or markers. We tested several NN architectures, including architectures that are constrained by prior biological knowledge. As we show, the NN achieve very good classification performance on training data and improve upon prior methods when used to cluster datasets from experiments that were not used in the training. We also performed functional analysis of the set of highly weighted nodes for each cell type and showed that even though NN are often described as a 'black box' learning method, many of these are functionally related to the cell type they were selected for.

Several methods have relied on clustering to annotate genes in single cell studies (3,16). Here we used NN for this task. Unlike most prior methods that relied on co-expression for clustering, our method is based on the identification of a set of genes that, combined, can be used to discriminate between different cell types. These can either be co-expressed, co-repressed or not expressed at all and they are selected as part of a training procedure rather than as a post-processing step. Such analysis may prove beneficial for better grouping of genes since it is supervised which allows it to overcome groupings that are based on non-cell type aspects (for example, cell cycle phase)

**Table 4.** Average retrieval performance across the different cell types

| Models | HSC | 4cell | ICM | Spleen | 8cell | Neuron | Zygote | 2cell | ESC | Mean |
|---|---|---|---|---|---|---|---|---|---|---|
| Original | 0.081 | 0.361 | 0.058 | 0.987 | 0.279 | 0.372 | 0.468 | 0.556 | 0.705 | 0.430 |
| PCA 100 | 0.299 | 0.508 | 0.01 | **0.996** | 0.351 | 0.646 | 0.539 | 0.616 | 0.722 | 0.521 |
| PCA 796 | 0.227 | 0.548 | 0.022 | 0.994 | 0.242 | 0.622 | 0.419 | 0.642 | **0.833** | 0.505 |
| DAE 100 | 0.236 | 0.411 | 0.016 | 0.973 | 0.503 | 0.628 | 0.193 | 0.728 | 0.544 | 0.470 |
| DAE 796 | 0.14 | 0.423 | 0.035 | 0.992 | 0.399 | 0.692 | 0.399 | **0.743** | 0.432 | 0.473 |
| Dense 1 layer 100 | 0.102 | 0.662 | 0.038 | 0.953 | 0.739 | 0.485 | 0.522 | 0.717 | 0.604 | 0.536 |
| Dense 1 layer 100 pre-train | 0.23 | 0.49 | 0.153 | 0.984 | 0.463 | 0.64 | 0.408 | 0.734 | 0.532 | 0.515 |
| Dense 1 layer 796 | 0.082 | 0.599 | 0.096 | 0.988 | 0.472 | 0.389 | **0.563** | 0.732 | 0.683 | 0.512 |
| Dense 1 layer 796 pre-train | 0.116 | 0.515 | 0.073 | 0.991 | 0.463 | **0.702** | 0.423 | 0.714 | 0.446 | 0.494 |
| Dense 2 layer 796/100 | 0.069 | 0.77 | 0.065 | 0.956 | 0.896 | 0.563 | 0.275 | 0.673 | 0.583 | 0.539 |
| Dense 2 layer 796/100 pre-train | 0.164 | 0.648 | 0.035 | 0.987 | 0.715 | 0.633 | 0.498 | 0.747 | 0.507 | 0.548 |
| PPITF 1 layer 696 + 100 | 0.078 | 0.636 | 0.148 | 0.965 | 0.667 | 0.464 | 0.202 | 0.314 | 0.63 | 0.456 |
| PPITF 1 layer 696 + 100 pre-train | 0.168 | 0.557 | 0.028 | 0.982 | 0.55 | 0.647 | 0.447 | 0.665 | 0.569 | 0.513 |
| PPITF 2 layer 696 + 100/100 | 0.068 | **0.771** | 0.182 | 0.956 | **0.849** | 0.561 | 0.415 | 0.553 | 0.71 | 0.563 |
| PPITF 2 layer 696 + 100/100 pre-train | **0.397** | 0.614 | **0.185** | 0.975 | 0.725 | 0.626 | 0.435 | 0.688 | 0.554 | **0.578** |

Bold indicates the highest value for each column.

As a final application we used the reduced representation obtained form the NN to query a large database of over 17K single cell expression data in order to determine the cell type of a newly profiled single cell. As we show, using such representation greatly improved the performance of the retrieval analysis while reducing the overall runtime and storage required. The Supporting Website (http://sb.cs.cmu.edu/scnn/) provides an implementation of the retrieval method which can be used by researchers to determine cell types for newly profiled single cells.

Our supervised training strategy is focused on cell type-specific identification. Such target function leads the method to ignore similarities between expression profiles that are based on other commonalities which may be shared across cell types, for example cell cycle phase. Researchers interested in such commonalties can either train a separate method where the target reflects them or use the autoencoder version which is fully unsupervised and so likely to identify additional commonalties beyond cell types.

While the results are encouraging, there are several directions for future work which we would like to explore. These include testing more involved (deeper) architectures, integrating additional types of prior biological knowledge into the model and an automated tool that can download new single cell expression data in order to increase the set used by the retrieval application. A major challenge with the latter direction is the ability to automatically assign cell type from published expression data given the various ways in which people define and encode such information.

## AVAILABILITY

Supporting website: http://sb.cs.cmu.edu/scnn/.

## SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

## FUNDING

National Institutes of Health (NIH) [1R01HL128172, 1U01HL122626-01]; National Science Foundation [DBI-1356505]. Funding for open access charge: NIH [1R01HL128172].

*Conflict of interest statement*. None declared.

## REFERENCES

1. Buettner,F., Natarajan,K.N., Casale,F.P., Proserpio,V., Scialdone,A., Theis,F.J., Teichmann,S.A., Marioni,J.C. and Stegle,O. (2015) Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nat. Biotechnol.*, **33**, 155–160.
2. Jaitin,D.A., Kenigsberg,E., Keren-Shaul,H., Elefant,N., Paul,F., Zaretsky,I., Mildner,A., Cohen,N., Jung,S., Tanay,A. *et al.* (2014) Massively parallel single-cell RNA-seq for marker-free decomposition of tissues into cell types. *Science*, **343**, 776–779.
3. Trapnell,C., Cacchiarelli,D., Grimsby,J., Pokharel,P., Li,S., Morse,M., Lennon,N.J., Livak,K.J., Mikkelsen,T.S. and Rinn,J.L. (2014) The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nat. Biotechnol.*, **32**, 381–386.
4. Darmanis,S., Sloan,S.A., Zhang,Y., Enge,M., Caneda,C., Shuer,L.M., Gephart,M.G.H., Barres,B.A. and Quake,S.R. (2015) A survey of human brain transcriptome diversity at the single cell level. *Proc. Natl. Acad. Sci. U.S.A.*, **112**, 7285–7290.
5. Poulin,J.-F., Tasic,B., Hjerling-Leffler,J., Trimarchi,J.M. and Awatramani,R. (2016) Disentangling neural cell diversity using single-cell transcriptomics. *Nat. Neurosci.*, **19**, 1131–1141.
6. Treutlein,B., Brownfield,D.G., Wu,A.R., Neff,N.F., Mantalas,G.L., Espinoza,F.H., Desai,T.J., Krasnow,M.A. and Quake,S.R. (2014) Reconstructing lineage hierarchies of the distal lung epithelium using single-cell RNA-seq. *Nature*, **509**, 371–375.
7. Kolodziejczyk,A.A., Kim,J.K., Tsang,J.C., Ilicic,T., Henriksson,J., Natarajan,K.N., Tuck,A.C., Gao,X., Bühler,M., Liu,P. *et al.* (2015) Single cell RNA-sequencing of pluripotent states unlocks modular transcriptional variation. *Cell Stem Cell*, **17**, 471–485.
8. Hough,S.R., Thornton,M., Mason,E., Mar,J.C., Wells,C.A. and Pera,M.F. (2014) Single-cell gene expression profiles define self-renewing, pluripotent, and lineage primed states of human pluripotent stem cells. *Stem Cell Rep.*, **2**, 881–895.
9. Hackl,H., Charoentong,P., Finotello,F. and Trajanoski,Z. (2016) Computational genomics tools for dissecting tumour-immune cell interactions. *Nat. Rev. Genet.*, **17**, 441–458.
10. Shalek,A.K., Satija,R., Shuga,J., Trombetta,J.J., Gennert,D., Lu,D., Chen,P., Gertner,R.S., Gaublomme,J.T., Yosef,N. *et al.* (2014) Single cell RNA Seq reveals dynamic paracrine control of cellular variation. *Nature*, **510**, 363–369.
11. Usoskin,D., Furlan,A., Islam,S., Abdo,H., Lönnerberg,P., Lou,D., Hjerling-Leffler,J., Haeggström,J., Kharchenko,O., Kharchenko,P.V. *et al.* (2015) Unbiased classification of sensory neuron types by large-scale single-cell RNA sequencing. *Nat. Neurosci.*, **18**, 145–153.

12. Yau,C. *et al.* (2016) pcaReduce: hierarchical clustering of single cell transcriptional profiles. *BMC Bioinformatics*, **17**, 140.
13. Xu,C. and Su,Z. (2015) Identification of cell types from single-cell transcriptomes using a novel clustering method. *Bioinformatics*, **31**, 1974–1980.
14. Pierson,E. and Yau,C. (2015) ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biol.*, **16**, 241.
15. Guo,M., Wang,H., Potter,S.S., Whitsett,J.A. and Xu,Y. (2015) SINCERA: a pipeline for single-cell RNA-seq profiling analysis. *PLoS Comput. Biol.*, **11**, e1004575.
16. Wang,B., Zhu,J., Pierson,E., Ramazzotti,D. and Batzoglou,S. (2017) Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nat. Methods*, **14**, 414–416.
17. Hornik,K., Stinchcombe,M. and White,H. (1989) Multilayer feedforward networks are universal approximators. *Neural Netw.*, **2**, 359–366.
18. Tan,J., Hammond,J.H., Hogan,D.A. and Greene,C.S. (2016) ADAGE-based integration of publicly available pseudomonas aeruginosa gene expression data with denoising autoencoders illuminates microbe-host interactions. *mSystems*, **1**, doi:10.1128/mSystems.00025-15.
19. Gupta,A., Wang,H. and Ganapathiraju,M. (2015) Learning structure in gene expression data using deep architectures, with an application to gene clustering. In: *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on IEEE*. IEEE Conference, Washington DC, pp. 1328–1335.
20. Chopra,S., Hadsell,R. and LeCun,Y. (2005) Learning a similarity metric discriminatively, with application to face verification. In: *Computer Vision and Pattern Recognition, 2005 (CVPR 2005). IEEE Computer Society Conference on IEEE Vol.1*. IEEE Conference, San Diego, Vol. **1**, pp. 539–546.
21. Barrett,T., Wilhite,S.E., Ledoux,P., Evangelista,C., Kim,I.F., Tomashevsky,M., Marshall,K.A., Phillippy,K.H., Sherman,P.M., Holko,M. *et al.* (2013) NCBI GEO: archive for functional genomics data sets–update. *Nucleic Acids Res.*, **41**, D991–D995.
22. Lun,A.T., Bach,K. and Marioni,J.C. (2016) Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. *Genome Biol.*, **17**, 75.
23. Kimmerling,R.J., Szeto,G.L., Li,J.W., Genshaft,A.S., Kazer,S.W., Payer,K.R., de Riba Borrajo,J., Blainey,P.C., Irvine,D.J., Shalek,A.K. *et al.* (2016) A microfluidic platform enabling single-cell RNA-seq of multigenerational lineages. *Nat. Commun.*, **7**, 10220.
24. Shin,J., Berg,D.A., Zhu,Y., Shin,J.Y., Song,J., Bonaguidi,M.A., Enikolopov,G., Nauen,D.W., Christian,K.M., Ming,G.-l. *et al.* (2015) Single-cell RNA-seq with waterfall reveals molecular cascades underlying adult neurogenesis. *Cell Stem Cell*, **17**, 360–372.
25. Shalek,A.K., Satija,R., Adiconis,X., Gertner,R.S., Gaublomme,J.T., Raychowdhury,R., Schwartz,S., Yosef,N., Malboeuf,C., Lu,D. *et al.* (2013) Single-cell transcriptomics reveals bimodality in expression and splicing in immune cells. *Nature*, **498**, 236–240.
26. Troyanskaya,O., Cantor,M., Sherlock,G., Brown,P., Hastie,T., Tibshirani,R., Botstein,D. and Altman,R.B. (2001) Missing value estimation methods for DNA microarrays. *Bioinformatics*, **17**, 520–525.
27. Stark,C., Breitkreutz,B.-J., Reguly,T., Boucher,L., Breitkreutz,A. and Tyers,M. (2006) BioGRID: a general repository for interaction datasets. *Nucleic Acids Res.*, **34**(Suppl. 1), D535–D539.
28. Prasad,T.K., Goel,R., Kandasamy,K., Keerthikumar,S., Kumar,S., Mathivanan,S., Telikicherla,D., Raju,R., Shafreen,B., Venugopal,A. *et al.* (2009) Human protein reference database–2009 update. *Nucleic Acids Res.*, **37**(Suppl. 1), D767–D772.
29. Schulz,M.H., Devanny,W.E., Gitter,A., Zhong,S., Ernst,J. and Bar-Joseph,Z. (2012) DREM 2.0: improved reconstruction of dynamic regulatory networks from time-series expression data. *BMC Syst. Biol.*, **6**, 104.
30. Gitter,A. and Bar-Joseph,Z. (2013) Identifying proteins controlling key disease signaling pathways. *Bioinformatics*, **29**, i227–i236.
31. Nepusz,T., Yu,H. and Paccanaro,A. (2012) Detecting overlapping protein complexes in protein-protein interaction networks. *Nat. Methods*, **9**, 471–472.
32. Erhan,D., Bengio,Y., Courville,A., Manzagol,P.-A., Vincent,P. and Bengio,S. (2010) Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, **11**, 625–660.
33. Hinton,G.E. and Salakhutdinov,R.R. (2006) Reducing the dimensionality of data with neural networks. *Science*, **313**, 504–507.
34. Chen,Y., Li,Y., Narayan,R., Subramanian,A. and Xie,X. (2016) Gene expression inference with deep learning. *Bioinformatics*, **32**, 1832–1839.
35. Krizhevsky,A. and Hinton,G.E. (2011) Using very deep autoencoders for content-based image retrieval. In: *19th European Symposium on Artificial Neural Networks*. Bruges.
36. Glorot,X. and Bengio,Y. (2010) Understanding the difficulty of training deep feedforward neural networks. *Aistats*, **9**, 249–256.
37. Reimand,J., Arak,T., Adler,P., Kolberg,L., Reisberg,S., Peterson,H. and Vilo,J. (2016) g: Profiler–a web server for functional interpretation of gene lists (2016 update). *Nucleic Acids Res.*, **44**, W83–W89.
38. Arthur,D. and Vassilvitskii,S. (2007) k-means++: The advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, pp. 1027–1035.
39. Hubert,L. and Arabie,P. (1985) Comparing partitions. *Journal of classification*, **2**, 193–218.
40. Vinh,N.X., Epps,J. and Bailey,J. (2010) Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, **11**, 2837–2854.
41. Fowlkes,E.B. and Mallows,C.L. (1983) A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, **78**, 553–569.
42. Rosenberg,A. and Hirschberg,J. (2007) V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. *EMNLP-CoNLL*, **7**, 410–420.
43. Tseng,G.C. and Wong,W.H. (2005) Tight clustering: A resampling-based approach for identifying stable and tight patterns in data. *Biometrics*, **61**, 10–16.
44. Monti,S., Tamayo,P., Mesirov,J. and Golub,T. (2003) Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, **52**, 91–118.
45. Julian,L.M. and Blais,A. (2015) Transcriptional control of stem cell fate by E2Fs and pocket proteins. *Frontiers in genetics*, **6**, 161.
46. Bailey,C.M. and Hendrix,M.J. (2008) IRF6 in development and disease: a mediator of quiescence and differentiation. *Cell Cycle*, **7**, 1925–1930.
47. Rustad,K.C., Wong,V.W. and Gurtner,G.C. (2013) The role of focal adhesion complexes in fibroblast mechanotransduction during scar formation. *Differentiation*, **86**, 87–91.
48. Ghosh,A.K. and Varga,J. (2007) The transcriptional coactivator and acetyltransferase p300 in fibroblast biology and fibrosis. *Journal of cellular physiology*, **213**, 663–671.
49. Eckardt,S., McLaughlin,K.J. and Willenbring,H. (2011) Mouse chimeras as a system to investigate development, cell and tissue function, disease mechanisms and organ regeneration. *Cell Cycle*, **10**, 2091–2099.
50. Hu,Y., Huang,K., An,Q., Du,G., Hu,G., Xue,J., Zhu,X., Wang,C.-Y., Xue,Z. and Fan,G. (2016) Simultaneous profiling of transcriptome and DNA methylome from a single cell. *Genome biology*, **17**, 1.
51. Zeisel,A., Muñoz-Manchado,A.B., Codeluppi,S., Lönnerberg,P., La Manno,G., Juréus,A., Marques,S., Munguba,H., He,L., Betsholtz,C. *et al.* (2015) Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, **347**, 1138–1142.
52. Li,C.-L., Li,K.-C., Wu,D., Chen,Y., Luo,H., Zhao,J.-R., Wang,S.-S., Sun,M.-M., Lu,Y.-J., Zhong,Y.-Q. *et al.* (2016) Somatosensory neuron types identified by high-coverage single-cell RNA-sequencing and functional heterogeneity. *Cell research*, **26**, 83–102.
53. Kim,C.C., Nakamura,M.C. and Hsieh,C.L. (2016) Brain trauma elicits non-canonical macrophage activation states. *Journal of neuroinflammation*, **13**, 1.
54. Klein,A.M., Mazutis,L., Akartuna,I., Tallapragada,N., Veres,A., Li,V., Peshkin,L., Weitz,D.A. and Kirschner,M.W. (2015) Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, **161**, 1187–1201.