Imputation of single-cell gene expression with an autoencoder neural network

Md. Bahadur Badsha[1*], Rui Li[1*], Boxiang Liu[2*],

Yang I. Li[3], Min Xian[4], Nicholas E. Banovich[5], Audrey Qiuyan Fu[1†]

[1]Department of Statistical Science, Institute for Bioinformatics and Evolutionary Studies, Center for Modeling Complex Interactions, University of Idaho; [2]Stanford University; [3]University of Chicago; [4]Department Computer Science, University of Idaho; [5]The Translational Genomics Research Institute, Phoenix, AZ

Email: †audreyf@uidaho.edu

Abstract

Single-cell RNA-sequencing (scRNA-seq) is a rapidly evolving technology that enables measurement of gene expression levels at an unprecedented resolution. Despite the explosive growth in the number of cells that can be assayed by a single experiment, scRNA-seq still has several limitations, including high rates of dropouts. High dropout rates result in a large number of genes having zero read count in the scRNA-seq data, which complicates downstream analyses. To overcome this problem, we treated zeros as missing values and developed deep learning methods to impute the missing values, exploiting the dependence structure across genes and across cells. Specifically, our LATE (Learning with AuToEncoder) method trains an autoencoder directly on scRNA-seq data with random initial values of the parameters, whereas our TRANSLATE (TRANSfer learning with LATE) method further allows for the use of a reference gene expression data set to provide LATE with an initial set of parameter estimates. On both simulated and real data, LATE and TRANSLATE outperform existing scRNA-seq imputation methods. Importantly, LATE and TRANSLATE are highly scalable and can impute

gene expression levels in over 1 million cells in just a few hours. We implemented our methods using Google TensorFlow and our software can run on a GPU as well as a CPU.

Introduction

Due to dropout and other technical limitations in single-cell sequencing technologies, single-cell RNA-seq (scRNA-seq) data typically contain many zero expression values. This is particularly true for droplet based scRNA-seq technologies (such as 10x genomics), which are the most commonly used in the field (Kolodziejczyk, et al., 2015). This often results in a read count of zero for over 80% (sometimes over 90%) of all the measurements across genes and across cells (Ziegenhain, et al., 2017). The high rate of zeroes leads to difficulties in downstream analyses: for example, it may obscure gene-gene relationships, or blur differences in subpopulations of cells (cell types). Although some of the zeroes represent no expression, most are due to failures to capture the transcript and do not indicate the true expression level. Recovering true expression levels behind the zeros in scRNA-seq data is therefore of great interest (Eraslan, et al., 2018; Huang, et al., 2018; Li and Li, 2018; Lopez, et al., 2018; van Dijk, et al., 2018). In this work, we treat the zeros as missing values (see comments on this assumption in the Discussion) and aim to use a computational approach to impute these missing values using information available from nonzero values, exploiting the dependence in gene expression levels across genes and across cells. The imputed data will have better quality than the original data, and may be used for diverse downstream analyses. For example, using the imputed scRNA-seq data, one may perform clustering analysis to identify cell types, and differential expression analysis to identify genes that are expressed differently in different cell types, study relationships between

certain genes, or construct a co-expression network of genes in single cells (Bacher and Kendziorski, 2016; Stegle and Teichmann, 2015).

Here, we developed imputation methods based on autoencoders for scRNA-seq data. Autoencoders (Hinton and Salakhutdinov, 2006) are a type of architecture commonly used in deep learning and enable reconstruction of the input through dimension reduction. Our LATE (Learning with AuToEncoder) method trains an autoencoder *de novo* on the highly sparse scRNA-seq data. Our TRANSLATE (TRANSfer learning with LATE) method builds on LATE and further incorporates a reference gene expression data set (e.g., bulk gene expression, a larger scRNA-seq data set, data from a complementary scRNA-seq technology, or scRNA-seq data of similar cells types collected elsewhere) through transfer learning (Bengio, 2011). TRANSLATE learns the dependence structure among genes in the reference panel; this information is stored in the parameter estimates that are transferred to LATE for imputation of the scRNA-seq data of interest. Autoencoders have demonstrated powerful performance in other applications, such as reconstructing 2D images and 3D shapes (Zhu, et al., 2016). We show with synthetic and real data that they are also powerful at imputation in highly sparse scRNA-seq data.

Methods

The LATE (Learning with AuToEncoder) Method

An autoencoder is a neural network of one or more hidden layers that allows for reconstructing the input, which is the highly sparse scRNA-seq data here, and thus generates the output with the missing values imputed (**Fig. 1A**). Each hidden layer consists of many artificial neurons (or

nodes), each of which provides a certain representation of the input. An autoencoder typically contains a bottleneck layer of a dimension that is lower (often much lower) than the input, and thus achieves dimension reduction. From the input to the bottleneck layer, the salient features in the data are encoded in reduced dimensions; this half of the autoencoder is called the "encoder". From the bottleneck layer to the output, the compressed information is gradually restored to eventually reconstruct all the values in the input; this half is therefore the "decoder". When certain values are missing in the input, the autoencoder is therefore able to learn the dependence structure among available values and use the representations stored in the hidden layers to recover missing values.

Let $\boldsymbol{X}$ be the input scRNA-seq matrix with values being $\log_{10}$-transformed read counts with a pseudocount of 1 added, i.e., $\log_{10}$ (count+1). The $\log_{10}$ transformation reduces variance in the raw read counts, which may vary from 0 to a few thousands. Let $\boldsymbol{Y}$ be the output matrix, and $\boldsymbol{H}_k$ be the *k*th hidden layer. The input matrix $\boldsymbol{X}$ and output matrix $\boldsymbol{Y}$ have the same dimensions and layout. For now, we consider genes as features and cells as independent samples. Both $\boldsymbol{X}$ and $\boldsymbol{Y}$ have m genes (columns) and n cells (rows). The ith row in either matrix thus corresponds to the ith sample (cell). The first hidden layer $\boldsymbol{H}_1$ with l_1 nodes following the input $\boldsymbol{X}_i$ (row vector) is derived by the following model:

$$\boldsymbol{H}_1 = f(\boldsymbol{w}_1 X_i^T + \boldsymbol{b}_1),$$

where the superscript T represents transpose, the weight matrix for the first hidden layer $\boldsymbol{w}_1$ is $l_1 \times m$, and the bias term $\boldsymbol{b}_1$ is a vector of length l_1. Similarly, from the kth hidden layer to the $k+1$st, the model is:

$$\boldsymbol{H}_{k+1} = f(\boldsymbol{w}_{k+1}\boldsymbol{H}_k + \boldsymbol{b}_{k+1}).$$

We use the rectified linear unit (ReLU) function for the activation function f, which means that for an arbitrary value g,

$$f(g) = \max(g, 0).$$

From the last hidden layer to the output layer, we have

$$Y_i^T = \boldsymbol{w}_q \boldsymbol{H}_q + \boldsymbol{b}_q,$$

where q represents the last layer. Our autoencoder will minimize the loss function, which is defined as the mean squared error (MSE) between the input and output matrix on the nonzero values:

$$MSE = \frac{\sum_i^m \sum_j^n (\boldsymbol{Y}_{ij} - \boldsymbol{X}_{ij})^2 I(\boldsymbol{X}_{ij} \neq 0)}{\sum_i^m \sum_j^n I(\boldsymbol{X}_{ij} \neq 0)},$$

where $I(\boldsymbol{X}_{ij} \neq 0)$ is the indicator function that takes on value 1 for a nonzero value in the input, and 0 otherwise. On simulated and real data, we have experimented with 1, 3 and 5 hidden layers with 400 nodes for the single hidden layer, with 400, 200 and 400 nodes for the three hidden layers, respectively, and with 400, 300, 200, 300 and 400 nodes for the five hidden layers, respectively.

We use a suite of techniques to train the autoencoder and obtain optimal estimates of the weights and biases. Specifically, we use the backprop (Rumelhart, et al., 1986) algorithm to calculate the gradients in each layer, and the Adaptive Moment estimation (Adam) (Kingma, 2014) algorithm to perform stochastic optimization using those gradients. We divide samples (cells) in the input data by the ratio 70:15:15 for training, validation and testing (although testing is typically not performed for imputation on the same input matrix). Optimization is performed iteratively with a randomly-generated set of parameter values as the starting point. The number of epochs (i.e., iterations) required for optimization is determined by comparing two learning curves: the curve

of the MSE on the training data and that of the MSE on the validation data over epochs. We expect both learning curves to reach the minimal point and have the smallest gap. If the learning curve on the training data is much lower than that on the validation data, it suggests overfitting; and it suggests underfitting if we have both large training and validation MSEs.

In addition, we adopt several strategies during optimization to improve efficiency and further reduce overfitting. For example, we randomly "remove" a certain percentage of the nodes in the input layer or a hidden layer, e.g., 20% or 50%. This technique, known as "dropout"(Dahl, 2013) in deep learning, provides a computationally inexpensive but powerful method to reduce overfitting (this "dropout" technique is not to be confused with the "dropout" phenomenon in single-cell sequencing data that results in a high percentage of zeros). In each epoch, we randomly divide the training set into mini-batches (typically of 256 samples per batch), and one optimization epoch trains on multiple mini-batches until all the samples have been used for training (Courville, 2016).

The input data matrix for LATE treats genes as features and cells as samples, assuming that the cells are independent of one another. However, cells can be dependent as well: cells sampled from the same cell type or differentiation stage tend to be more informative of one another than cells from another cell type or another stage. Therefore, to impute the expression value of a gene in a cell, we need to account for dependence among cells in addition to that among genes. Indeed, the MAGIC method exploits the dependence structure among cells and establish neighborhoods of cells based on the gene expression profile (van Dijk, et al., 2018). Motivated by this idea, we develop the "LATE combined" method that accounts for information from both dimensions in

the input data. Specifically, we perform imputation with LATE twice, first using genes as features and cells as samples, and next using cells as features and genes as samples. We obtain two imputed matrices, denoted $\boldsymbol{Y}^g$ and $\boldsymbol{Y}^c$, respectively, both of which have genes in the rows and cells in the columns. Next, we merge them by genes with a smaller MSE after imputation: for gene i, we consider the ith row in the two matrices, denoted by $\boldsymbol{y}^g_{(i)}$ and $\boldsymbol{y}^c_{(i)}$, with the corresponding MSE $MSE^g_{(i)}$ and $MSE^c_{(i)}$, respectively. If $MSE^g_{(i)} < MSE^c_{(i)}$, then we choose $\boldsymbol{y}^g_{(i)}$. Otherwise we choose $\boldsymbol{y}^c_{(i)}$. The resulting imputed matrix contains values from both matrices and has an MSE that is lower than the MSE of either $\boldsymbol{Y}^g$ or $\boldsymbol{Y}^c$.

TRANSLATE: TRANSfer learning with LATE

When prior knowledge about the gene-gene relationships exists, we can incorporate this knowledge through transfer learning, which effectively runs LATE twice (**Fig. 1B**). In the first step, we train LATE on a reference gene expression data set using genes as features. In the second step, we train LATE on the scRNA-seq input data. Instead of using random values as the initial parameter values, we use the parameters (weights and biases) learned from the first step as initial values. As we demonstrate later, gene expression from bulk samples or from a larger single-cell sample of similar cell types may be used as the reference.

Similar to the LATE combined approach described above, we also allow for the TRANSLATE combined option, which applies TRANSLATE to the input data matrix with genes being features, and applies LATE to the input matrix with cells being features. We use the same approach as in the LATE combined method to combine the two imputation matrices.

Assessing imputation accuracy

We calculate the mean squared error (MSE) with respect to the nonzero values in the input data matrix as an overall measure of imputation accuracy. When the ground truth $\boldsymbol{T}$ is available, which has the same dimension as $\boldsymbol{Y}$, we can also calculate another MSE with respect to all the values in $\boldsymbol{T}$ in order to assess how close the imputed data matrix is to the truth:

$$MSE'_{all} = \frac{\sum_i^m \sum_j^n (\boldsymbol{Y}_{ij} - \boldsymbol{T}_{ij})^2}{mn}.$$

There are cases where zeros in the ground truth may still represent missing values: for example, we may use a real single-cell data set with a high nonzero rate as the ground truth and mask certain values to generate synthetic input. Then the zeros in the ground truth still represent missing values and should not be interpreted as no expression. In such cases, it is sensible to exclude the zeros when calculating the MSE:

$$MSE'_{nz} = \frac{\sum_i^m \sum_j^n (\boldsymbol{Y}_{ij} - \boldsymbol{T}_{ij})^2 I(\boldsymbol{X}_{ij} \neq 0)}{\sum_i^m \sum_j^n I(\boldsymbol{X}_{ij} \neq 0)}.$$

Relationship to other scRNA-seq imputation methods

Several methods have been developed recently to impute missing values in scRNA-seq data. Some methods, such as MAGIC (van Dijk, et al., 2018) and scImpute (Li and Li, 2018), use similarity among cells for imputation, whereas other methods, including SAVER (Huang, et al., 2018), DCA (Eraslan, et al., 2018) and scVI (Lopez, et al., 2018), rely on similarity among genes. The latter methods effectively treat cells as independent samples and model the read counts in scRNA-seq data with a negative binomial distribution. Additionally, DCA and scVI take a deep learning approach and develop deep neural networks also based on autoencoders. However, whereas DCA and scVI assume read counts to follow a negative binomial distribution and

estimate parameters of this distribution as part of the inference with their autoencoders, we do not make explicit assumptions on read counts. Additionally, scVI accounts for batch effects in their statistical model and removes batch effects in imputation. Other methods, including ours, do not address batch effects.

Generating synthetic data for assessing performance

Using real gene expression data sets (from bulk tissues and single cells) in human and mouse, we generated several synthetic data sets, including the input and the corresponding ground truth, for assessing the performance of imputation methods (see summary statistics of the data sets in **Table S1**).

1. The MAGIC_mouse data. The authors of the MAGIC method analyzed a mouse bone marrow data set of 16,114 genes and 2,576 cells (Paul, et al., 2015; van Dijk, et al., 2018) (see "URLs for Data Sets"). This data set contains cells at multiple differentiation stages of hematopoiesis; each stage had been associated with the expression of a number of surface marker genes. This data set therefore contains dynamic changes of gene expressions and is valuable for assessing how the imputation method captures the dynamics.

The original scRNA-seq data is highly sparse. To enhance the dynamic changes in gene expression and the correlation structure among genes, we applied the MAGIC method to generate the imputed data set. Although the dynamic patterns produced in the imputed data set may not be real, the imputed data provide the ground truth with sharply-defined nonlinear patterns (see additional detail in "Recovery of nonlinear gene-gene relationships" in Results). We randomly selected 30% of the cells in the imputed data and split the data into two sets. The

set containing 30% of the cells was used as the ground truth. We then generated the input data by randomly masking 90% of the values in the ground truth as zero. We applied LATE and other methods to the input and compared the imputed data matrix with the input and with ground truth. We used the other set that contained 70% of the cells as the reference expression data for TRANSLATE.

2. A subset of the human GTEx data (GTEx_4tissues). The GTEx consortium contains the bulk gene expression data from 53 tissues (GTEx Portal v7; see "URLs for Data Sets"), which is useful for us to assess the impact of using relevant and irrelevant tissues as the reference in TRANSLATE. This data set also contains many zeros that are more reliable indication of no expression, which we may use to assess how the imputation method perform in the presence of true zeros. From the 53 tissues, we selected four tissues with large sample sizes as the ground truth: muscle (564), heart (600), skin (1203) and adipose (797). This data contains 56,202 genes from 3,164 samples, with a nonzero rate of 49.8%. We then randomly masked 90% of the values as zero to create an input data set. When assessing the performance of TRANSLATE, we considered two references: i) the entire GTEx data from all the tissues, including the four tissues in the input; and ii) the GTEx data without the four tissues in the input. We were interested in whether including the right tissues in the reference is necessary for imputation. Note that when calculating MSE' for GTEx_4tissues, we needed to calculate MSE'_{all}, instead of MSE'_{nz} that were used for other data sets. This is because zeros in the GTEx data were obtained from bulk sequencing and therefore should be interpreted as no expression.

3. A small subset of the data in human peripheral blood mononuclear cells (PBMCs) from 10x Genomics (Zheng, et al., 2017) (PBMC_G949). The original data set contains 68K cells (see "URLs for Data Sets"). We selected 949 genes in 21,065 cells, such that the total read count per gene is >10K and the total read count per cell is >1.5K. This data set has a nonzero rate of 41% and was used as the ground truth (zeros were discarded when calculating MSE'). We randomly masked nonzero values so that the resulting data contains only 10% of nonzero values. When applying TRANSLATE, we explored several options for the reference, which is explained in detail in Results.

4. Another subset of the 10x Genomics PBMC data with cell type labels (PBMC_G949_10k). To assess the ability of the methods in separating different cell types, we generated another data set from the same PBMC data as mentioned above. The original data set contains 40K cells with cell type labels and 10 cell types. We focused on the same 949 genes as in the PBMC_G949 data set, subsampled 10K cells while retaining the relative frequencies of the ten cell types; this data set, with a nonzero rate of 33%, is used as ground truth. Similar to before, we randomly masked values in the ground truth to generate the input, such that the resulting nonzero rate is 10%. We refer to the masked data set as PBMC_G949_10K (see summary statistics in **Table S1**).

Data sets without the ground truth

We assessed the method performance also on the following real scRNA-seq data sets without the ground truth (**Table S1**):

1. A medium-sized PBMC data set (PBMC_G5561) of 5,561 genes in 53,970 cells, such that the total read count is >1K per gene and >1K per cell, and that the nonzero rate is 10%.

2. A large PBMC data set (PBMC_G9987) of 9,987 genes and 53,970 cells with the total read count >200 per gene and >1K per cell and a nonzero rate of 6%. Note that the medium-sized and large data sets do not have ground truth.

3. A smaller mouse brain data set from 10x Genomics (mouse_brain_G10K) of 10K genes and all the 1.3M cells with a nonzero rate of 20% (see "URLs for Data Sets").

4. The entire mouse brain data set (mouse_brain_G28K) of all the 27,998 genes and 1.3M cells with a nonzero rate of 7%.

PCA-based tSNE plots for visualization

When investigating the separation of cell types, we generated PCA-based tSNE plots to visualize how cells are grouped. To generate such plots, we first applied PCA to the data matrix, extracted the top principal components (e.g., 50 or 100), and then generated a tSNE plot using these top PCs as the input. Since important features are summarized in top PCs, a tSNE plot based on these PCs provides a better visualization of key features than that based on the data matrix directly.

Results

Overall imputation accuracy

We first assessed the performance between LATE and TRANSLATE. We applied LATE and TRANSLATE with different architectures (with 1, 3, and 5 hidden layers), and calculated MSE with respect to the input and ground truth, respectively (**Tables S2 and S3**). We observed only small differences in MSE (with respect to the input) across different architectures, and these differences typically disappeared after we fine-tuned the hyperparameters (e.g., the number of nodes in a hidden layer, the learning rate, the retain probability in the "dropout" procedure, the

mini-batch size, etc.). We also did not observe substantial differences between LATE and TRANSLATE in MSE' (with respect to the ground truth), except on the MAGIC mouse data. Compared to LATE, TRANSLATE reduces both MSE and MSE' by half on the MAGIC mouse data, indicating that the reference data significantly improves imputation. However, the reference and the input in the MAGIC mouse data come from the same data set, which means that the dependence structure is essentially identical in the reference and the input. In reality, however, such perfect reference is usually not available.

We next compared our methods with other imputation methods, such as MAGIC, SAVER, DCA and scVI and summarized the MSEs in **Table S3**. scImpute had issues running on larger data sets and was therefore excluded from the comparison (see Section "Scalability" in Results for details of running scImpute). When the ground truth is available, our methods have a lower MSE' (with respect to the ground truth) than the other methods on all the synthetic data sets except for the GTEx_4tissues data. Note that we calculated MSE'_{all} for GTEx_4tissues, but MSE'_{nz} for other data sets. For GTEx_4tissues, DCA achieves the lowest MSE', followed by our methods, MAGIC, scVI and SAVER. Recall that both DCA and scVI estimate the "dropout probability" of an expression value of zero is due to scRNA-seq dropout and impute the zeros only when they have a high dropout probability. The MSE comparison above indicates that DCA can be successful at detecting true zeros and is more successful than scVI, and that on the other hand, the DCA model for nonzero values may be biased. It is also worth noting that for this data set, SAVER achieved the lowest MSE (with respect to the input), but the highest MSE' (20 times higher than that of LATE and TRANSLATE), suggesting that the SAVER model is overfitting the nonzero values in the input and does not extrapolate well for zeros.

On the four real scRNA-seq data sets with no ground truth, we calculated only MSE (with respect to the input) to compare the performance of different methods (**Table S3**). LATE and TRANSLATE produced similar MSEs, both being no more than a third of the MSE from scVI, SAVER and MAGIC. DCA achieved 41% of the MSE by LATE combined on the PBMC_G5561 data set but 449.6% of the MSE by LATE combined on the PBMC_G9987 data. Due to similar performance between LATE and TRANSLATE, we only ran LATE on the two mouse brain data sets. Other methods are unable to run on these two data sets (see detail in Section "Scalability" in Results).

Recovery of nonlinear gene-gene relationships

We expect an imputation method to perform well not only on the overall accuracy, but also in capturing the true dependence structure among genes. For example, two genes have drastically varying expression levels across cells but have a strong correlation. An imputation method may correctly estimate the expression values for each of genes, but allocate the estimates to the wrong cells and thus fail to account for the correlation between two genes. Therefore, although a good imputation method should result in a lower MSE, a low MSE alone does not necessarily indicate that the imputation data recover all the true features. When assessing pairwise relationships, linear relationships between genes are generally easy to capture, whereas nonlinear relationships are much more difficult. Here, we examine the performance of our methods and other methods on recapitulation nonlinear relationships in gene pairs.

The synthetic data sets we have generated contain different types of nonlinear relationships. In the MAGIC_mouse data, the diffusion process in the MAGIC method provides an approach to generate gene expression profiles that evolve over time and that enhance the correlation structure

over time (see examples in Fig 2B in van Dijk et al 2018 (van Dijk, et al., 2018). As the number of time points, denoted by 't', increases, the diffusion process produces more sharply defined nonlinear patterns). Although the imputation results from MAGIC are less satisfactory, their diffusion process allows us to generate synthetic single-cell data with sharply defined nonlinear patterns (the ground truth in **Fig. 2**; by contrast, gene-gene relationships from other synthetic data sets are more similar to having multiple clusters; see below). The step of masking down to a nonzero rate of 10% removes most of the nonlinear features in the input, although the upper "arm" remains (**Fig. 2** input). Both LATE and TRANSLATE recovered the nonlinear relationship. Surprisingly, even though the ground truth and input were generated by MAGIC, it could not recover this pattern. None of the other methods (SAVER, DCA and scVI) recovered this pattern, either.

The other type of nonlinear patterns may be better described as having multiple clusters (the ground truth in **Figs. 3** and **4**). Recall that the ground truth for the GTEx_4tissues data set (**Fig. 3**) is the partial GTEx data, which are bulk gene expression values, and that the ground truth for the PBMC_G949 data set is the scRNA-seq data in the PBMCs with a nonzero rate of 41%. The nonlinear patterns in these data sets are directly obtained from the real data. Since the data were not collected over time, it is reasonable that these nonlinear patterns may reflect subpopulations of cells, rather than a trajectory over which cells evolve.

On the input data derived from these ground truth data sets, LATE and TRANSLATE continue to recover the truth on selected gene pairs (**Figs. 3** and **4**). The performance of DCA and scVI improves drastically, although not necessarily better than our methods. These improved results suggest that DCA and scVI may be sensitive to how the data are generated. Unlike DCA and

scVI that model the read count of genes with negative binomial distributions, MAGIC focuses on the dependence structure among cells and does not model read counts. If the single-cell data were generated by the MAGIC, which is likely incompatible with the model behind DCA or scVI, the latter two methods may not be able to produce sensible results. The performance of MAGIC also improves somewhat, although it still produced patterns that capture only partial features of the true relationships. SAVER produced scatterplots that are similar to the input, suggesting that its imputation runs of risk of overfitting to the nonzero values in the input.

We also generated some additional gene pairs with different types of nonlinear relationship for MAGIC_mouse, GTEx_4tissues, PBMC_G949, PBMC_G5561 and PBMC_G9987 data (see **Supplementary Figures S2-S13**).

Separation of cell types

Single-cell data often consist a mixture of cell types, and the high dropout rate tends to obscure the differences among cell types. While broad cell type classifications are still possible with the aid of marker genes, the obscured relationship makes it difficult to identify novel or poorly characterized cell types. Imputation is therefore a powerful approach to help recovering cell types: it provides higher-quality data for downstream clustering analyses, which separate known cell types and may suggest novel ones. Using single-cell data with cell type labels being known, we can examine imputation performance on separating cell types.

We applied all the methods discussed above to the PBMC_G949_10K data, which was derived from a larger PBMC data set with cell type lables, generated tSNE plots and colored the cell types (**Fig. 5A**). When running TRANSLATE, we used the other 30K cells from the 40K cells with cell type labels as the reference, which we refer to as reference i) (**Table S2**). Indeed,

whereas cells of the same type (or color) are largely grouped in a cluster in the tSNE plot of the ground truth, different cell types are completely scrambled and indistinguishable in the tSNE plot of the input. LATE, TRANSLATE, DCA and scVI all managed to group similar cells and separate cell types after imputation, producing a tSNE plot similar to that of the ground truth. By contrast, neither SAVER or MAGIC separated the cell types: whereas SAVER produced a tSNE plot that is almost identical to the input, MAGIC separated cells into tiny clusters that do not reflect the cell types.

A closer look at the results from LATE, TRANSLATE, DCA and scVI reveals subtle differences: apart from the larger clusters being visually separated, TRANSLATE with the 30K PBMC single-cell data being the reference (30k_NZ33, which has a nonzero rate of 33%) managed to also separate the c8 cluster of 60 cells (0.6% of the total 10K cells; represented by yellow dots in **Fig. 5**) and the even smaller cluster of c9 (5 cells; turquoise in **Fig. 5**) from the larger clusters. Running tSNE again on the imputed data produces a different layout, but c8 and c9 remain separated from larger clusters. We further experimented with three other references (**Fig. 5B**; **Table S2**): ii) 'another sample': the single-cell PBMC data from another sample collection of the same individual; iii) '30k_NZ10': the 30K PBMC cells with masking (the nonzero rate is also 10%); and iv) all the tissues in the GTEx data. Most cells in c8 and all the c9 cells are clearly clustered together and separated from other clusters in the tSNE plots from references ii) and iii), whereas using GTEx as the reference does not lead to satisfactory results (**Fig. 5B**). However, the MSEs do not differ too much among the four references: both MSE and MSE' are only slightly higher under the GTEx reference than the other three references (**Table S3**). The poor performance with the GTEx reference on separating cell types is likely due to the tissue types in the GTEx data being very different from the PBMCs. As a result, the weights and biases

TRANSLATE learns from GTEx are a rather poor starting point for the single-cell input data. Among other methods we compare with, none of LATE, DCA and scVI was able to group the c8 or c9 cells together or separate them from other clusters. This analysis indicates the additional advantage of having a relevant reference in our TRANSLATE method, and also provides an example where a reference from irrelevant tissue or cell types may increase the MSE and produce less satisfactory imputation results.

We next examined the imputation performance of our methods on data with batches. We reanalyzed the mouse retina data previously analyzed by scVI (**Fig 6**). This data contains 13,166 genes and 19,829 Cells with a nonzero rate of 6.7%. The tSNE plot on the input of log10-transformed count data displays the claw-shaped clusters of cell types; the claws reflect the two batches in this data set. scVI provides a cleaner separation than our LATE method does, although the claw shapes remain in several cell types even though scVI has removed the batch effect. This is particularly encouraging as our methods do not account for batch effect in the deep learning architecture. The tSNE plot indicates that LATE still managed to group similar cell types together while retaining the batch effect. For downstream analysis, one may apply batch correction (such as COMBAT (Johnson, Li and Rabinovic, 2007)) to LATE-imputed data.

Scalability

We used the mouse brain data also from 10x Genomics to test scalability. The entire mouse brain data consists of 28K genes and 1.3M cells, with a nonzero rate of 7% (mouse_brain_28K). Running on this data set took LATE 17 hours on a GPU (NVIDIA GTX 1080 Ti). TRANSLATE in principle can run on this data set as well, as it is essentially running LATE twice. We did not

run TRANSLATE on this large data set because LATE and TRANSLATE have similar performance on most data sets, and because our interest here is primarily scalability.

When we cut the mouse brain data in half and generated a smaller data set of 10K genes and 1.3M cells, with a nonzero rate of 20% (mouse_brain_10K). LATE took only 3 hours on the GPU. DCA took 42 hours on the same GPU, scVI did only 8% of training after running for 56 hours and MAGIC still failed to run, and SAVER reported memory error after running for more than 129 hours. We did not include scImpute, another imputation method, in the comparison above because it not only failed to run on mouse_brain_10K, but also took a much longer time than other methods on smaller data sets. For example, scImpute took 68 hours to run on a subsample of 15K cells from mouse_brain_10K (10K genes and 15K cells), and kept running after 6 days on a subsample containing 31K cells (10K genes and 31K cells), and failed to run on a subsample containing 62K cells (10K genes and 62K cells; half of the mouse_brain_10K data).

Discussion

Here we present our novel deep learning algorithms for imputing missing values in single-cell RNA-seq data that often have a low nonzero rate. Our algorithms do not make assumptions on the statistical distributions of the single-cell data, and can still efficiently recovery the truth. Similar results between LATE and TRANSLATE on most of the data sets examined here indicate the effectiveness of autoencoders in quickly reaching optimality or near optimality, regardless of the starting point. Our algorithms are also fast and highly scalable.

Additional work is needed to improve our methods. Here we manually tuned hyperparameters, but have not explored the possibility of systematically tuning these parameters, as that

implemented in DCA. It is also of interest to gain a better understanding of how the reference helps imputation, which may offer additional insights on which data sets are suitable as the reference. When combining imputation results from both dimensions, our current approach of combining two imputed data matrices is rather ad hoc; joint optimization for both dimensions simultaneously may help further improve the results.

In LATE and TRANSLATE we make the central assumption that all the zeros in the scRNA-seq data are missing values. As a result, our methods impute missing values always with a nonzero value. However, it is likely that some zeros are real and reflect the actual lack of expression. It is therefore desirable to distinguish true zeros from those due to technical artefacts. One way to approach this problem is to estimate the probability of an expression value is a dropout event. Both scVI and DCA have an explicit model for the read counts and estimate this probability, although their performance is rather unsatisfactory based on our simulation, as the MSE with respect to the ground truth is larger than that from LATE or TRANSLATE in most cases. When the ground truth contains many zeros (e.g., in ground truth for data GTEx_4tissues), DCA managed to achieve a lower MSE'_{all} (0.0535) than LATE combined (0.1134) or TRANSLATE combined (0.1148), whereas scVI still has a larger MSE'_{all} (0.7883). This suggests that DCA likely does a good job identifying true zeros, thus achieving a lower MSE. On the other hand, DCA has a higher MSE' than our methods on other data sets where the ground truth contains only a few or no zeros, which means that DCA does not recover nonzero values well. scVI uses a similar method to distinguish zeros but has a higher MSE in all the data sets examined here. It is therefore unclear whether the unsatisfactory performance in DCA and scVI is due to scRNA-seq

data not containing enough information for distinguishing the zeros, or whether these two methods currently being unable to make such distinction.

Software availability:

Our software is implemented in Python and builds on Google TensorFlow. It may be run on CPUs and GPUs. The source code is available at https://github.com/audreyqyfu/LATE.

URLs for data sets:

Mouse bone marrow data: GEO GSE72857.
https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE72857.

GTEx gene expression data (Gene TPMs): https://gtexportal.org/home/datasets.

Human PBMC and mouse brain data from 10x Genomics:
https://support.10xgenomics.com/single-cell-gene-expression/datasets.

Mouse retina data: provided in the scVI software package
(https://github.com/YosefLab/scVI/tree/master/tests/data); original data from GEO GSE81905.
https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE81905.

Acknowledgements

We acknowledge the extensive support on hardware and software provided by the Computational Resources Core (CRC) through the Institute for Bioinformatics & Evolutionary Studies at the University of Idaho, and in particular the assistance from Dr. Benjamin Oswald, the Director of the CRC.

Funding

This research is supported by NIH R00HG007368 (to A.Q.F.) and partially by the NIH/NIGMS grant P20GM104420 to the Center for Modeling Complex Interactions at the University of Idaho.

Declaration of Interests

The authors declare no competing interests.

Reference:

1. Bacher, R. and Kendziorski, C. Design and computational analysis of single-cell RNA-sequencing experiments. *Genome Biology* 2016;17:63.
2. Bengio, Y. Deep learning of representations for unsupervised and transfer learning. In, *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop* 2012. p. 17-37.
3. Goodfellow, I., Bengio, Y. and Courville, A. Deep Learing. *MIT Press, Chapter 8* http://www.deeplearningbook.org/ 2016.
4. Dahl, G.E., Sainath, T. N. and Hinton, G. E. Improving deep neural networks for LVCSR using rectified linear units and dropout. *IEEE International Conference* 2013: 8609-8613.
5. Eraslan, G., *et al.* Single cell RNA-seq denoising using a deep count autoencoder. *bioRxiv*. 2018; 300681.
6. Hinton, G.E. and Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* 2006;313(5786):504.
7. Huang, M., *et al.* SAVER: gene expression recovery for single-cell RNA sequencing. *Nature Methods* 2018;15(7):539-542.

8. Johnson, W.E., Li, C. and Rabinovic, A. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics* 2007;8(1):118-127.
9. Kingma, D.P. and Ba, J. Adam: a method for stochastic optimization. *arXiv* 2014;1412.6980.
10. Kolodziejczyk, A.A., *et al.* The technology and biology of single-cell RNA sequencing. *Molecular cell* 2015;58(4):610-620.

11. Li, W.V. and Li, J.J. An accurate and robust imputation method scImpute for single-cell RNA-seq data. *Nature Communications* 2018;9(1):997.
12. Lopez, R., *et al*. Bayesian inference for a generative model of transcriptome profiles from single-cell RNA sequencing. *bioRxiv*. 2018;292037.
13. Paul, F., *et al.* Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell* 2015;163(7):1663-1677.
14. Rumelhart, D.E., Hinton, G.E. and Williams, R.J. Learning representations by back-propagating errors. *Nature* 1986;323:533.
15. Stegle, O., Teichmann, S.A. and Marioni, J.C. Computational and analytical challenges in single-cell transcriptomics. *Nature Reviews Genetics* 2015;16:133.
16. van Dijk, D., *et al.* Recovering gene interactions from single-cell data using data diffusion. *Cell* 2018;174(3):716-729.e727.
17. Zheng, G.X.Y., *et al.* Massively parallel digital transcriptional profiling of single cells. *Nature Communications* 2017;8:14049.
18. Zhu, Z., *et al.* Deep learning representation using autoencoder for 3D shape retrieval. *Neurocomputing* 2016;204(C):41-50.

19. Ziegenhain, C., *et al.* Comparative analysis of single-cell RNA sequencing methods. *Molecular Cell* 2017;65(4):631-643.e634.

Figure Titles and Legends

Figure 1: **Architectures of our deep learning methods LATE and TRANSLATE for imputing zeros in scRNA-seq data.**

Figure 2: **Two genes (Cd34 vs Gypa) with a nonlinear relationship in the synthetic data based on the mouse bone marrow data (MAGIC_mouse; 16,114 genes and 2,576 cells).** Each dot is a single cell. The color bar indicates the density of data points. The input contains only 10% of the values in the ground truth. Only LATE and TRANSLATE recovered the nonlinear relationship. The data points in the results from MAGIC are concentrated near the origin (see **Supplementary Figure S1**). Also, see three additional gene pairs with different types of nonlinear relationship in **Supplementary Figures S2-S4**.

Figure 3: **Two genes (ENSG00000237683 vs ENSG00000117115) with a nonlinear relationship in the synthetic data based on GTEx (GTEx_4tissues; 56,202 genes and 3,164**

samples). Each dot is a single cell. The color bar indicates the density of data points. LATE, TRANSLATE and DCA recovered the nonlinear relationship reasonably well. Also, see three additional gene pairs with different types of nonlinear relationship in **Supplementary Figures S5-S7**.

Figure 4: **Two genes (ENSG00000158869 vs ENSG00000090382) with a nonlinear relationship in the small synthetic data based on the 10x Genomics PBMC data (PBMC_G949; 949 genes and 21,065 cells)**. Each dot is a single cell. The color bar indicates the density of data points. LATE, TRANSLATE, DCA and scVI all recovered the nonlinear relationship reasonably well, although scVI inferred a much reduced range. Similar to **Figure 2**, the data points from MAGIC are also concentrated near the origin (see **Supplementary Figure S1**). Also, see two additional gene pairs with different types of nonlinear relationship in **Supplementary Figures S8-S9.**

Figure 5: **tSNE plots of cells from the synthetic data based on the 10x Genomics PBMC data with known cell types (PBMC_G949_10K; 949 genes and 10K cells).** Each dot is a single cell, and each color indicates a cell type. (**A**) Comparison of all the methods. LATE, TRANSLATE (30k_NZ33 as the reference), DCA and scVI all recovered the clustering. (**B**) Comparison of different reference gene expression data sets in TRANSLATE. Using other scRNA-seq data as the reference (e.g., another sample, 30k_NZ10, and GTEx), TRANSLATE managed to separate rare cell types (c8 in yellow and c9 in turquoise). The bulk gene expression of multiple tissues from GTEx does not serve as a good reference.

Figure 6: **Imputation performance on cell type separation in the mouse retina data with batch effects**. A tSNE plot is generated for the input, for the output from LATE, and for the output from scVI. The claw-shaped clusters are due to two batches in the data.

Figure 1

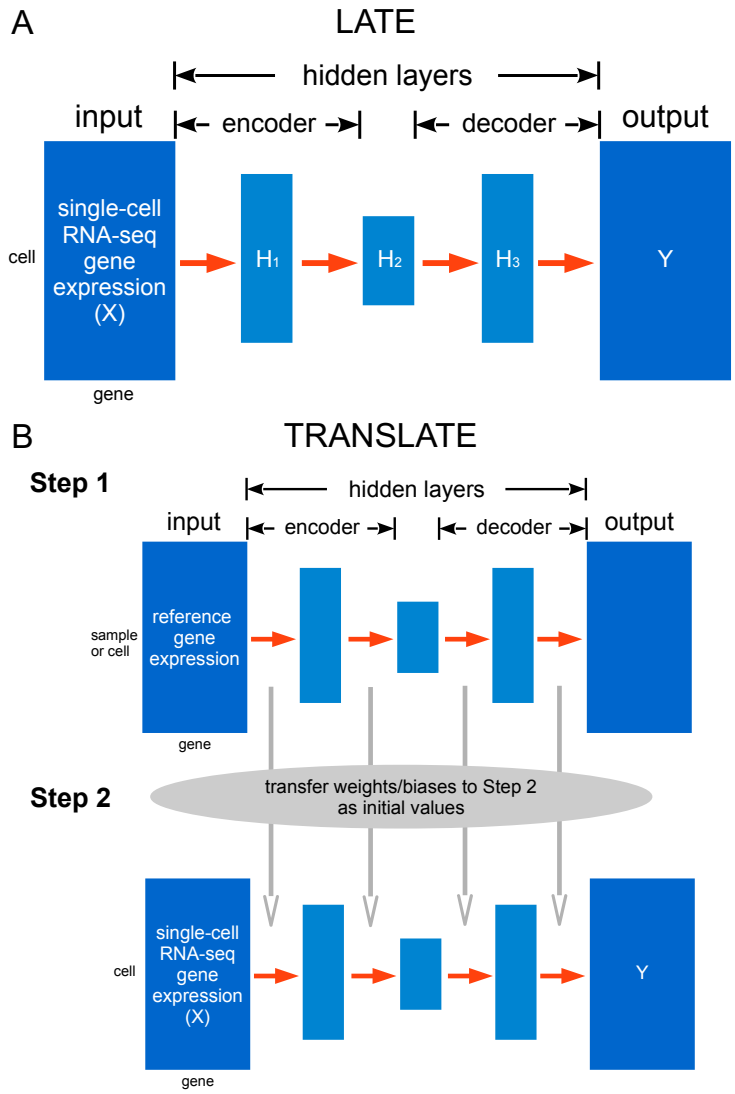

A
LATE
hidden layers
input
encoder
decoder
output
cell
single-cell RNA-seq gene expression (X)
H1
H2
H3
Y
gene
B
TRANSLATE
Step 1
hidden layers
input
encoder
decoder
output
sample or cell
reference gene expression
gene
Step 2
transfer weights/biases to Step 2 as initial values
cell
single-cell RNA-seq gene expression (X)
Y
gene

Figure 2

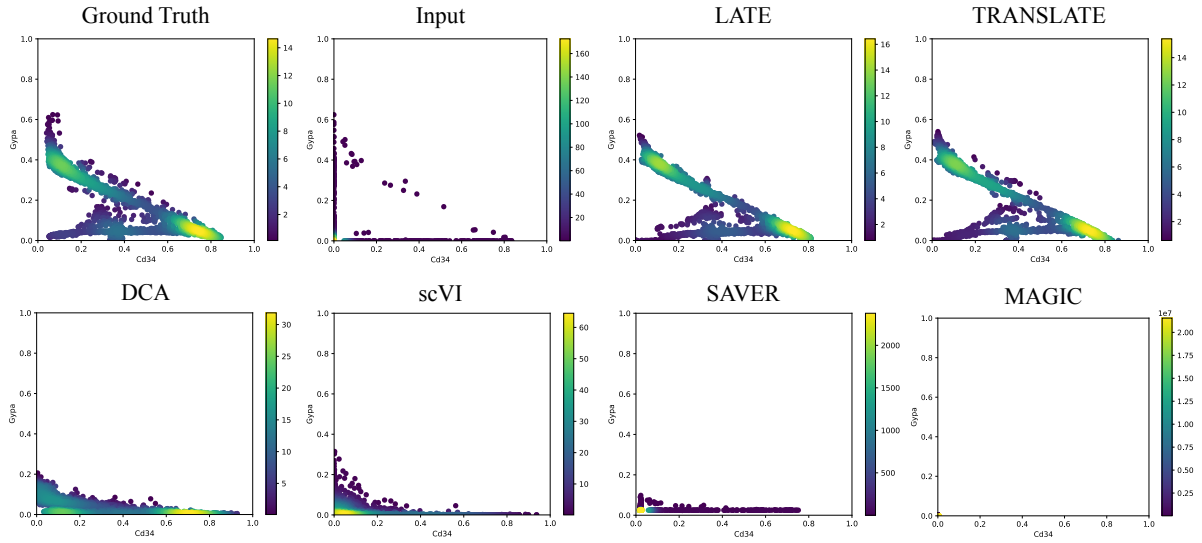

Ground Truth
Input
LATE
TRANSLATE
DCA
scVI
SAVER
MAGIC
Gypa
Cd34

Figure 3

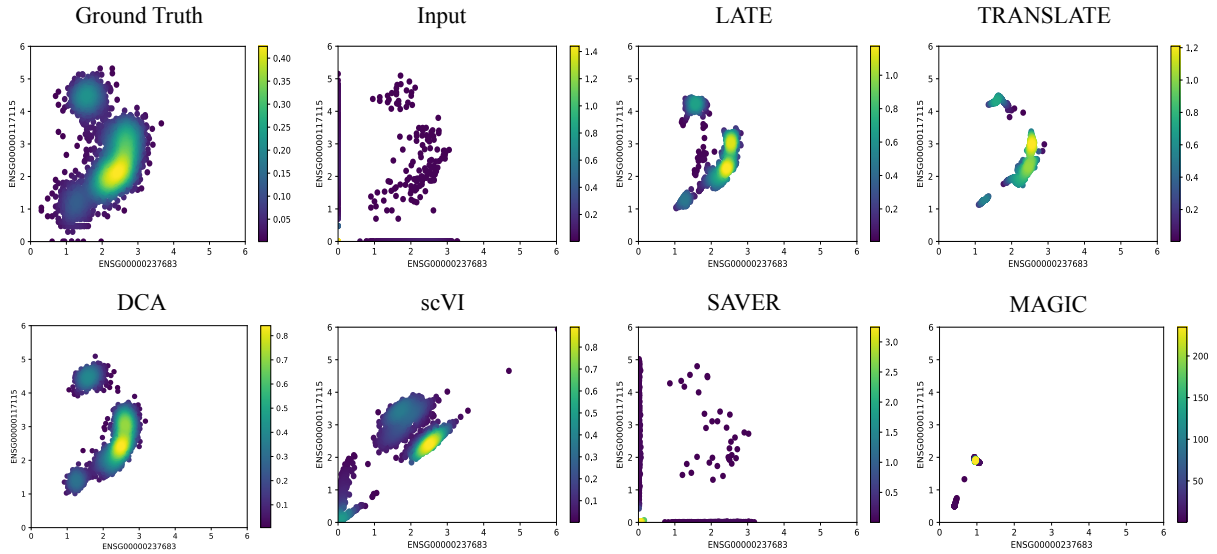

Ground Truth
Input
LATE
TRANSLATE
DCA
scVI
SAVER
MAGIC
ENSG00000117115
ENSG00000237683

Figure 4

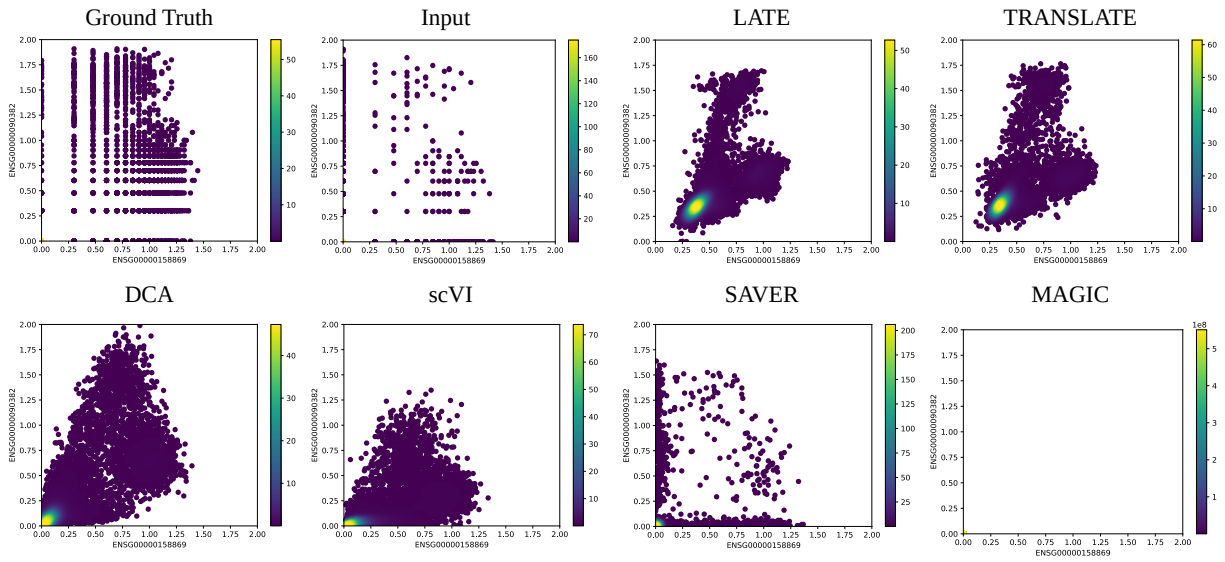

Ground Truth
Input
LATE
TRANSLATE
DCA
scVI
SAVER
MAGIC
ENSG00000090382
ENSG00000158869
1e8

Figure 5

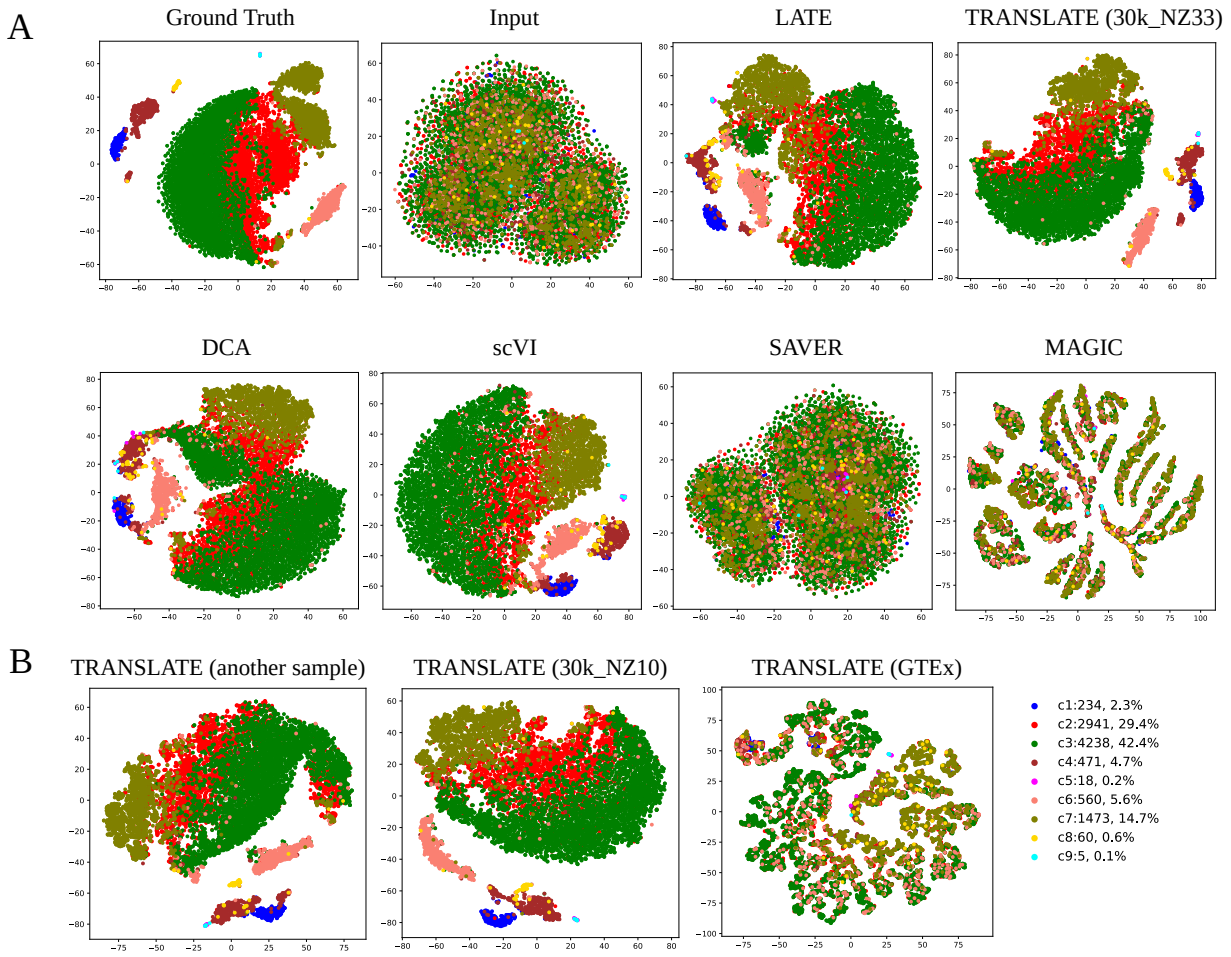
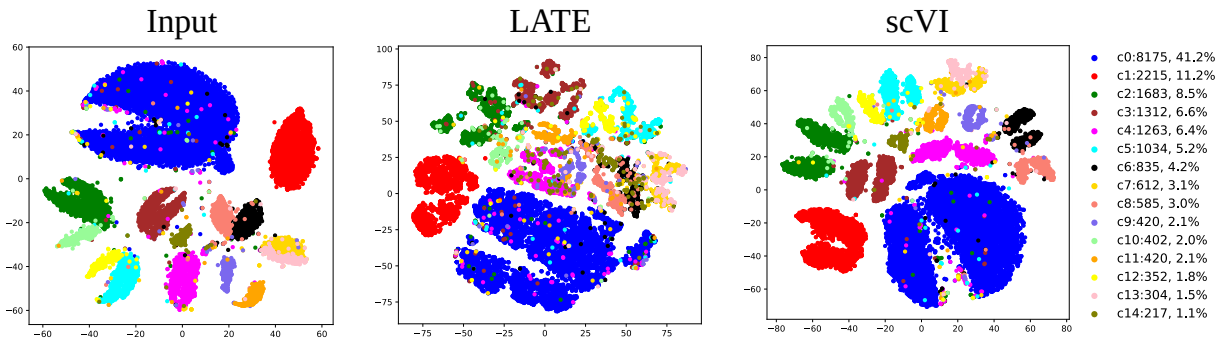

A
Ground Truth
Input
LATE
TRANSLATE (30k_NZ33)
DCA
scVI
SAVER
MAGIC
B
TRANSLATE (another sample)
TRANSLATE (30k_NZ10)
TRANSLATE (GTEx)
c1:234, 2.3%
c2:2941, 29.4%
c3:4238, 42.4%
c4:471, 4.7%
c5:18, 0.2%
c6:560, 5.6%
c7:1473, 14.7%
c8:60, 0.6%
c9:5, 0.1%

Figure 6


Input
LATE
scVI
c0:8175, 41.2%
c1:2215, 11.2%
c2:1683, 8.5%
c3:1312, 6.6%
c4:1263, 6.4%
c5:1034, 5.2%
c6:835, 4.2%
c7:612, 3.1%
c8:585, 3.0%
c9:420, 2.1%
c10:402, 2.0%
c11:420, 2.1%
c12:352, 1.8%
c13:304, 1.5%
c14:217, 1.1%