

Deep learning tackles single-cell analysis - A survey of deep learning for scRNA-seq analysis

Mario Flores^{1§}, Zhentao Liu¹, Tinghe Zhang¹, Md Musaddaqui Hasib¹, Yu-Chiao C

2021-09-22

Contents

->

test text justification s

About the authors

§ Corresponding authors

Mario Flores (mario.flores@utsa.edu)

Yidong Chen (cheny8@uthscsa.edu)

Yufei Huang (yuh119@pitt.edu)

Affiliations

1Department of Electrical and Computer Engineering, the University of Texas at San Antonio, San Antonio, TX 78249, USA

2Greehey Children's Cancer Research Institute, University of Texas Health San Antonio, San Antonio, TX 78229, USA

3Department of Population Health Sciences, University of Texas Health San Antonio, San Antonio, TX 78229, USA

4Department of Microbiology and Molecular Genetics, University of Pittsburgh, Pittsburgh, Pennsylvania, PA 15232, USA

5Department of Medicine, School of Medicine, University of Pittsburgh, PA 15232, USA

6UPMC Hillman Cancer Center, University of Pittsburgh, PA 15232, USA

Book Maintainer

Hello there :D

Any feedback and contributions will be appreciated.

Mail: sumin.jo@utsa.edu

Website: [Huang-AI4Medicine-Lab](https://Huang-AI4Medicine-Lab.github.io/)

About this book

This book is full version of our research paper [**Please add paper publish information here**].

Keywords deep learning; single-cell RNA-seq; imputation; dimension reduction; clustering; batch correction; cell type identification; functional prediction; visualization

Abstract

Since its selection as the method of the year in 2013, single-cell technologies have become mature enough to provide answers to complex research questions. However, together with the growth of single-cell profiling technologies, there has also been an increase of computational challenges to process the generated datasets. It's here that by effectively leveraging large data sets, Deep Learning (DL) is positioning as the first option for single-cell analyses. Here we provide a unified mathematical description of the DL methods used in single cell RNA sequencing (scRNA-Seq) followed with the survey of the most representative published DL algorithms for scRNA-Seq in the field.

Key Points

- Single cell RNA sequencing technology generate large collection of transcriptomic profiles of up to millions of cells, enabling biological investigation of hidden structures or cell types, predicting their effects or responses to treatment more precisely, or utilizing subpopulation to address unanswered hypotheses.
- Current Deep Learning-based analysis approaches for single cell RNA seq data is systematically reviewed in this paper according to the challenge they address and their roles in the analysis pipeline.

- A unified mathematical description of the surveyed DL models is presented and the specific model features were discussed when reviewing each approach.
- A comprehensive summary of the evaluation metrics, comparison algorithms, and datasets by each approaches is presented.

Chapter 1

Introduction

Single cell sequencing technology has been a rapidly developing area to study genomics, transcriptomics, proteomics, metabolomics, and cellular interactions at the single cell level for cell-type identification, tissue composition and reprogramming (??) . Specifically, sequencing of the transcriptome of single cells, or single-cell RNA-sequencing (scRNA-seq), has become the dominant technology in many frontier research areas such as disease progression and drug discovery (??) . One particular area where scRNA-seq has made a tangible impact is cancer, where scRNA-seq is becoming a powerful tool for understanding invasion, intratumor heterogeneity, metastasis, epigenetic alterations, detecting rare cancer stem cells, and therapeutic response (refs). Currently, scRNA-seq is applied to develop personalized therapeutic strategies that are potentially useful in cancer diagnosis, therapy resistance during cancer progression, and the survival of patients (??). The scRNA-seq has also been adopted to combat COVID-19 to elucidate how the innate and adaptive host immune system miscommunicates resulting in worsening the immunopathology produced during this viral infection (??).

These studies have led to a massive amount of scRNA-seq data deposited to public databases such as 10X Single-cell gene expression dataset, Human Cell Atlas, and Mouse Cell Atlas. Expressions of millions of cells from 18 species have been collected and deposited, waiting for further analysis. On the other hand, due to biological and technical factors, scRNA-seq data presents several analytical challenges related to its complex characteristics like missing expression values, high technical and biological variance, noise and sparse gene coverage, and elusive cell identities (?) . These characteristics make it difficult to directly apply commonly used bulk RNA-seq data analysis techniques and have called for novel statistical approaches for scRNA-seq data cleaning and computational algorithms for data analysis and interpretation. To this end, specialized scRNA-seq analysis pipelines such as Seurat (?) and Scanpy (?). along with a large collection of task-specific tools, have been developed to address the intricate

technical and biological complexity of scRNA-seq data.

Recently, deep learning has demonstrated its significant advantages in natural language processing and speech and facial recognition with massive data (??). Such advantages have initiated the application of DL in scRNA-seq data analysis as a competitive alternative to conventional machine learning approaches for uncovering cell clustering (??) , cell type identification (??), gene imputation (???) , and batch correction (?) in scRNA-seq analysis. Compared to conventional machine learning (ML) approaches, DL is more powerful in capturing complex features of high-dimensional scRNA-seq data. It is also more versatile , where a single model can be trained to address multiple tasks or adapted and transferred to different tasks. Moreover, the DL training scales more favorably with the number of cells in scRNA-seq data size, making it particularly attractive for handling the ever-increasing volume of single cell data. Indeed, the growing body of DL-based tools has demonstrated DL’s exciting potential as a learning paradigm to significantly advance the tools we use to interrogate scRNA-seq data.

In this paper, we present a comprehensive review of the recent advances of DL methods for solving the present challenges in scRNA-seq data analysis (Table??) from the quality control, normalization/batch effect reduction, dimension reduction, visualization, feature selection, and data interpretation by surveying deep learning papers published up to April 2021. In order to maintain high quality for this review, we choose not to include any (bio)archival papers, although a proportion of these manuscripts contain important new findings that would be published after completing their peer-reviewed process. Previous efforts to review the recent advances in machine learning methods focused on efficient integration of single cell data (??) . A recent review of DL applications on single cell data has summarized 21 DL algorithms that might be deployed in single cell studies (?). It also evaluated the clustering and data correction effect of these DL algorithms using 11 datasets.

In this review, we focus more on the DL algorithms with a much detailed explanation and comparison. Further, to better understand the relationship of each surveyed DL model with the overall scRNA-seq analysis pipeline, we organize the surveys according to the challenge they address and discuss these DL models following the analysis pipeline. A unified mathematical description of the surveyed DL models is presented and the specific model features are discussed when reviewing each method. This will also shed light on the modeling connections among the surveyed DL methods and the recognition of the uniqueness of each model. Besides the models, we also summarize the evaluation matrices of these DL algorithms and compare the tools that integrate these DL algorithms. Access to these DL algorithms with the original research results, available datasets used by these methods are also listed to demonstrate the advantages and utility of the DL algorithms. We envision that this survey will serve as an important information portal for learning the application of DL for scRNA-seq analysis and inspire innovative use of DL to address a broader range of new challenges

in emerging multi-omics and spatial single-cell sequencing.

Chapter 2

Overview of scRNA-seq processing pipeline

Various scRNA-seq techniques (like SMART-seq, Drop-seq, and 10X genomics sequencing (??)) are available nowadays with their sets of advantages and disadvantages. Despite the differences in the scRNA-seq techniques, the data content and processing steps of scRNA-seq data are quite standard and conventional. A typical scRNA-seq dataset consists of three files: genes quantified (gene IDs), cells quantified (cellular barcode), and a count matrix (number of cells x number of genes), irrespective of the technology or pipeline used. A series of essential steps in scRNA-seq data processing pipeline and optional tools for each step with both ML and DL approaches are illustrated in Fig.??.

With the advantage of identifying each cell and unique molecular identifiers (UMIs) for expressions of each gene in a single cell, scRNA-seq data are embedded with increased technical noise and biases [23]. Quality control (QC) is the first and the key step to filter out dead cells, double-cells, or cells with failed chemistry or other technical artifacts. The most commonly adopted three QC covariates include the number of counts (count depth) per barcode identifying each cell, the number of genes per barcode, and the fraction of counts from mitochondrial genes per barcode (?).

Normalization is designed to eliminate imbalanced sampling, cell differentiation, viability, and many other factors. Approaches tailored for scRNA-seq have been developed including the Bayesian-based method coupled with spike-in, or BASiCS (?), deconvolution approach, scran (?), and sctransform in Seurat where regularized Negative Binomial regression was proposed (?). Two important steps, batch correction and imputation, will be carried out if required by the analysis:

- **Batch Correction** is a common source of technical variation in high-

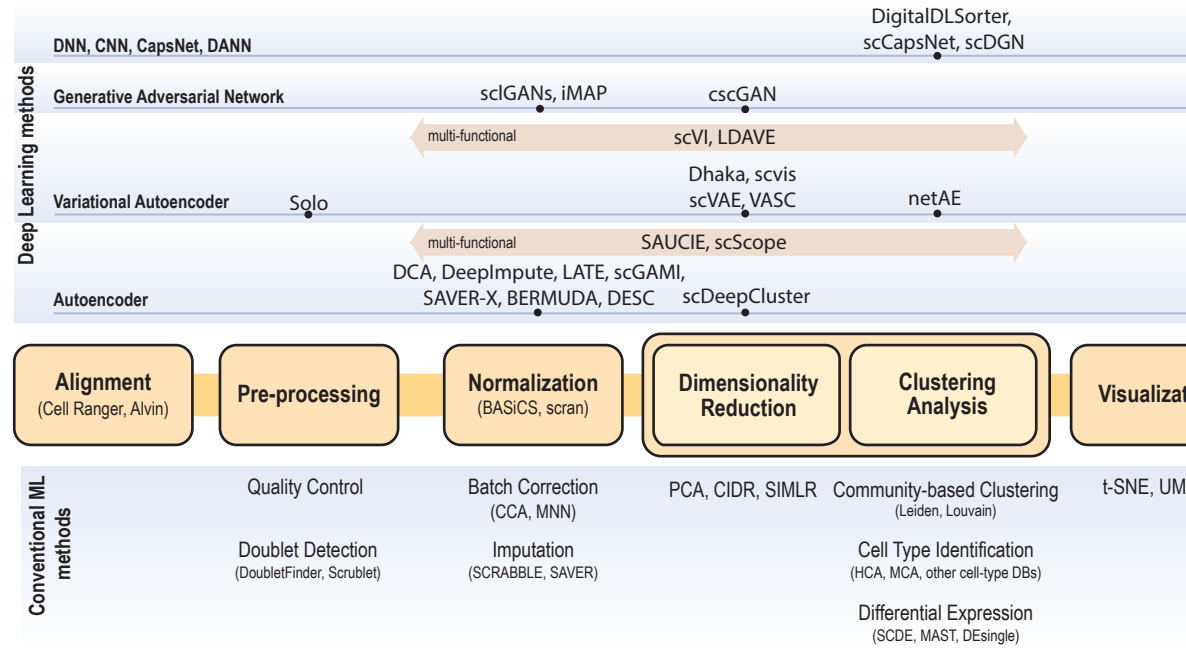


Figure 2.1: Single cell data analysis steps for both conventional ML methods (bottom) and DL methods (top). Depending on the input data and analysis objectives, major scRNA-se analysis steps are illustrated in the center flow chart. The conventional ML approaches along with optional analysis modules are presented below each analysis step. Deep learning approaches are categorized as neural network models (DNN, CNN, CapsNet, and DANN), Generative Adversarial Network (GAN), Variational Autoencoder, and Autoencoder. For each DL approach, optional algorithms are listed on top of each step in the pipeline.

throughput sequencing experiments due to variant experimental conditions such as technicians and experimental time, imposing a major challenge in scRNA-seq data analysis. Batch effect correction algorithms include detection of mutual nearest neighbors (MNNs) (?), canonical correlation analysis (CCA) with Seurat (?), and Harmony algorithm through cell-type representation (?).

- **Imputation** step is necessary to handle high sparsity data matrix, due to missing value or dropout in scRNA-seq data analysis. Several tools have been developed to “impute” zero values in scRNA-seq data, such as SCRABBLE (?), SAVER (?) and scImpute (?). Dimensionality reduction and visualization are essential steps to represent biological meaningful variation and high dimensionality with significantly reduced computational cost. Dimensionality reduction methods, such as PCA, are widely used in scRNA-seq data analysis to achieve that purpose. More advanced nonlinear approaches that preserve the topological structure and avoid overcrowding in lower dimension representation, such as LLE (?) (used in SLICER (?)), tSNE (?), and UMAP (?) have also been developed and adopted as a standard in single-cell data visualization.

Clustering analysis is a key step to identify cell subpopulations or distinct cell types to unravel the extent of heterogeneity and their associated cell-type-specific markers. Unsupervised clustering is frequently used here to categorize cells into clusters by their similarity often taken the aforementioned dimensionality-reduced representations as input, such as community detection algorithm Louvain (?) and Leiden (?), or data-driven dimensionality reduction followed with k-Means cluster by SIMLR (?).

Feature selection is another important step in single-cell RNA-seq analysis is to select a subset of genes, or features, for cell-type identification and functional enrichment of each cluster. This step is achieved by differential expression analysis designed for scRNA-seq, such as MAST that used linear model fitting and likelihood ratio testing (?); SCDE that adopted a Bayesian approach with a Negative Binomial model for gene expression and Poisson process for dropouts (?), or DEsingle that utilized a Zero-Inflated Negative Binomial model to estimate the dropouts (?).

Besides these key steps, downstream analysis can include cell type identification, coexpression analysis, prediction of perturbation response, where DL has also been applied. Other advanced analyses including trajectory inference and velocity and pseudotime analysis are not discussed here because most of the approaches on these topics are non-DL based.

Chapter 3

Overview of common deep learning models for scRNA-seq analysis

Unsupervised learning is the key step in the scRNA-Seq analysis, including batch correction, dimension reduction, imputation, and clustering, which lend themselves naturally to unsupervised DL models including the variational autoencoder (VAE), the autoencoder (AE), or generative adversarial networks (GAN). Also, adversarial transfer learning has been applied for cell-type classification. We started our review by introducing the general formulations of VAE, AE, and GAN for scRNA-seq together with their training strategies. These general formulations facilitate understanding the methodologies used by different papers in developing their specific algorithms, enabling us to focus on the different features of each method and bring attention to their uniqueness and novelty.

3.1 Variational Autoencoder (VAEs) for scRNA-Seq data

Let x_n represent a $G \times 1$ vector of gene expression (UMI counts or normalized, log-transformed expression) of G genes in cell n , where x_{gn} denotes gene g 's expression, which is assumed to follow some distribution $p(x_{gn}|v_{gn}, \alpha_{gn})$ (e.g., zero-inflated negative binomial (ZINB) or Gaussian), where v_{gn} and α_{gn} are parameters of the distribution (e.g., mean, variance, or dispersion) (Fig.??A). We consider the first parameter v_{gn} to be of particular interest (e.g., the mean counts) for the scRNA-seq analysis and is thus further modeled as a function of a d -dimension latent variable $z_n \in R^d$ and an observed variable s_n (e.g., the batch ID) by a decoder neural network D_θ (Fig.??A) as

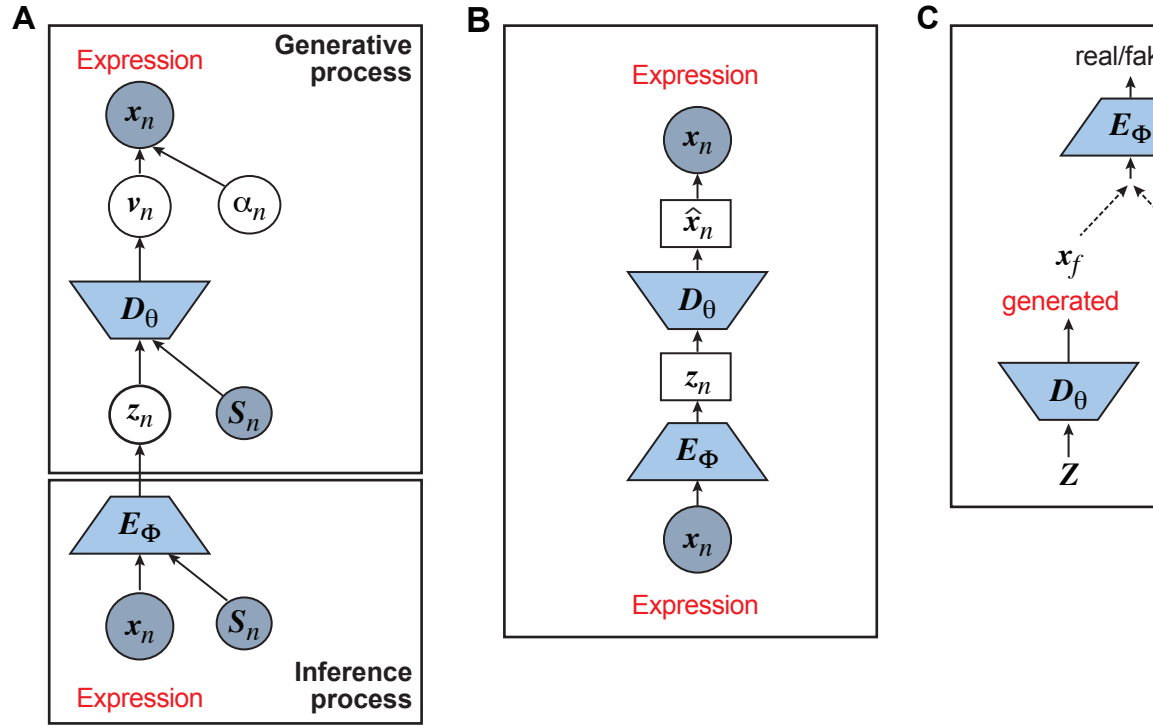


Figure 3.1: Graphical models of the surveyed DL models including A) Variational Autoencoder (VAE); B) Autoencoder (AE); and C) Generative Adversarial Network (GAN)

$$v_n = D_\theta(z_n, s_n) \quad (3.1)$$

where the g th element of v_n is v_{gn} and θ is a vector of decoder weights, z_n represents a low-dimensional latent representation of gene expression and is used in all the works for visualization and clustering. For VAE, z_n is commonly assumed to follow a multivariate standard normal prior, i.e., $p(z_n) = N(0, I_d)$ with I_d being a $d \times d$ identity matrix. The second parameter α_{gn} of $p(x_{gn}|v_{gn}, \alpha_{gn})$ is a nuisance parameter, which is assumed with a prior distribution $p(\alpha_{gn})$ and can be either estimated or marginalized in variational inference. Now define $\Theta = \{\theta, \alpha_{ng} \forall n, g\}$ as the collection of the unknown model parameters. Then, $p(x_{gn}|v_{gn}, \alpha_{gn})$ and ((??)) together define the likelihood $p(x_{gn}|z_n, s_{gn}, \Theta)$.

The goal of training or inference is to compute the maximum likelihood estimate of

$$\hat{\Theta}_{ML} = \operatorname{argmax}_{\Theta} \sum_{n=1}^N \log(x_n|s_n, \Theta) \approx \operatorname{argmax}_{\Theta} \sum_{n=1}^N L(\Theta) \quad (3.2)$$

where $p(x_n|s_n, \Theta) = \int p(x_n|z_n, s_n, \Theta)p(z_n)dz_n$ is the marginal likelihood, which is in general analytically intractable but can be lower-bounded by the evidence lower bound (ELBO) $L(\Theta)$, expressed as

$$L(\Theta) = E_{q(z_n|x_n, s_n, \Theta)}[\log p(x_n|z_n, s_n, \Theta)] - D_{KL}[q(z_n|x_n, s_n, \Theta) \| p(z_n)] \quad (3.3)$$

where $q(z_n|x_n, s_n)$ is an approximate to the intractable posterior distribution $p(z_n|x_n, s_n)$. To make the variational inference tractable $q(z_n|x_n, s_n)$ is assumed in most cases as a multivariate Gaussian

$$q(z_n|x_n, s_n) = N(\mu_{z_n}, \operatorname{diag}(\sigma_{Z_n}^2)) \quad (3.4)$$

whose means and variances $\mu_{z_n}, \sigma_{Z_n}^2$ are given by an encoder network E_ϕ applied to x_n and s_n (Fig.??A) as

$$\mu_{z_n}, \sigma_{Z_n}^2 = E_\phi(x_n, s_n) \quad (3.5)$$

where θ is the vector of the unknown decoder weights. Because of the approximation by $L(\Theta)$ in ((??)) and the introduction of the decoder network in eq.((?)), the model parameters to be estimated become $\theta = \{\theta, \phi, \alpha_{ng}, \forall n, g\}$. Optimization of $L(\Theta)$ in ((??)) is computed efficiently by stochastic optimization, where the gradient is calculated by backpropagation.

All the surveyed papers that deploy VAE follow this general modeling process. However, an alternative and more general formulation treats it as optimization with a loss function expressed as

$$L(\Theta) = -L(\Theta) + \sum_{k=1}^K \lambda_k L_k(\Theta) \quad (3.6)$$

where $L_k \forall k = 1, \dots, K$ are K additional function-specific losses introduced to constrain the model for different functions (clustering, cell type prediction, etc), and λ_k s are the Lagrange multipliers. With this general formulation, for each paper, we focus on the survey the specific choice of data distribution $p(x_{gn}|v_{gn}, \alpha_{gn})$ that defines $L(\Theta)$, different L_k designed for specific functions, and how the decoder and encoder are applied to model different aspects of scRNA-seq data.

3.2 Autoencoders (AEs) for scRNA-seq data

AEs have been proposed to learn the low dimensional latent representation of expression x_n . The AE includes an encoder E_ϕ and a decoder D_θ (Fig.??B) such that

$$z_n = E_\phi(x_n); \hat{x}_n = D_\theta(z_n) \quad (3.7)$$

where like VAE $z_n \in R^d$ is the d-dimension latent variable, $\Theta = \{\theta, \phi\}$ are encoder and decoder weight parameters, and \hat{x}_n defines the parameters (e.g. mean) of data distribution and thus the likelihood $p(x_n|\Theta)$ (Fig.??B). Note that the mean of $p(x_n|\Theta)$ is often considered as the imputed and denoised expression of x_n . For most common AEs, $p(x_n|\Theta)$ assumes a Gaussian distribution and \hat{x}_n becomes the mean of the Gaussian and can be directly used as imputed, normalized gene expression. Nevertheless, additional designs can be introduced to attend imputation specifically. $p(x_n|\Theta)$ can also be negative binomial (NB) or ZINB as in DCA (?) to model the reads count directly with their parameters defined as functions of \hat{x}_n . Additional design can be included in the AE model for batch correction, clustering, and other functions.

The training of the AE is generally carried out by stochastic gradient descent algorithms to minimize the loss with the general expression similar to that of VAE in eq.(??)

$$L(\Theta) = L_0(\Theta) + \sum_{k=1}^K \lambda_k L_k(\Theta) \quad (3.8)$$

3.3. GENERATIVE ADVERSARIAL NETWORKS (GANS) FOR SCRNA-SEQ DATA 21

where L_0 is $-\log p(x_n|\Theta)$, and L_k s are K additional function-specific losses. When $p(x_n|\Theta)$ is the Gaussian, L_0 becomes one of the most commonly used mean square error (MSE) loss

$$L_0(\Theta) = \sum_{n=1}^N \|x_n - \hat{x}_n\|_2^2 \quad (3.9)$$

Because different AE models differ in their AE architectures and the loss functions, we will discuss the specific architecture and the loss functions for each reviewed model.

3.3 Generative adversarial networks (GANs) for scRNA-seq data

GANs have been used for imputation, data generation and augmentation of the scRNA-seq analysis. Without loss of generality, the GAN, when applied to scRNA-seq, is designed to learn to generate gene expression profiles from p_x , the distribution of x_n , the count or normalized expression vectors of the scRNA data. The vanilla GAN consists of two deep neural networks (?). The first network is the generator $G_\theta(z_n, y_n)$ with parameter θ , which is essentially a decoder that takes a noise vector z_n from the distribution p_z and a class label y (e.g. cell type) as input and is trained to generate x_f , a “fake” sample of a gene expression profile (Fig.??C). Note that including a class label y_n at the input is optional and when it is included, the model is known as the conditional GAN. The second network is the discriminator network D_{ϕ_D} with parameters ϕ_D , which is a classifier trained to distinguish between the “real” sample x and fake data x_f (Fig.??C). The generator G_θ and discriminator D_{ϕ_D} are trained to outplay each other, resulting in a minimax game, in which G_θ is forced by D_{ϕ_D} to produce better samples, which, when converge, can fool the discriminator D_{ϕ_D} , thus becoming samples from p_x . The vanilla GAN suffers heavily from training instability and mode collapsing (?). To that end, Wasserstein GAN (WGAN) was developed to effectively improve the training stability and convergence (?). The WGAN loss computes the Wasserstein distance, also called the earth moving distance between the real and fake sample distributions (?):

$$L(\Theta) = \max_{\phi_D} \sum_{n=1}^N D_{\phi_D}(x_n) - \sum_{n=1}^N D_{\phi_D}(G_\theta(z_n, y_n)) \quad (3.10)$$

Additional terms can also be added to ((?)) to constrain the functions of the generator. Training based on the WGAN loss in Eq. ((?)) amounts to a min-max optimization, which iterates between the discriminator and generator, where each optimization is achieved by a stochastic gradient descent algorithm

through backpropagation. The WGAN requires D to be K -Lipschitz continuous (?), which can be satisfied by adding the gradient penalty to the WGAN loss (?). Once the training is done, the generator G_{ϕ_G} can be used to generate gene expression profiles of new cells.

Chapter 4

Overview of datasets and evaluation metrics

A variety of datasets and performance evaluation metrics have been used to demonstrate the performance of the surveyed DL models for different tasks. We summarize these datasets and evaluation metrics in Table ?? & ??. We detail the mathematical definition of the evaluation metrics in the following.

4.1 Evaluation methods

An extensive list of evaluation methods has been proposed for different scRNA-seq analysis tasks (???). We provide an overview here the methods adopted in the surveyed papers. We discuss them according to the key categories on which the surveyed papers are organized, namely, imputation, batch effect correction, dimension reduction and clustering, cell type identification, and functional analysis.

4.1.1 Imputation

The evaluation of the performance of imputation methods considers their ability to recover biological signals and improve downstream analyses. For this two main methods have been used. First is the evaluation of similarity between bulk and imputed scRNA-seq data. Second is the evaluation of imputation on unsupervised clustering.

The first approach consist in assessing the similarity between bulk and imputed scRNA-seq data. For a given scRNA-seq dataset, the “pseudobulk”, or the

average of normalized (log2-transformed) scRNA-seq counts across cells, is calculated first, and the Spearman's rank correlation coefficient (SCC) between the pseudobulk and the bulk RNA-seq profile of the same cell type is evaluated. The statistical significance is assessed whether SCCs between the bulk and pseudobulks from two imputation methods are equal.

The second approach consist in measuring the accuracy of several clustering assignments methods using four metrics:

- Entropy of accuracy (H_{acc}) and entropy of purity (H_{pur}). H_{acc} (H_{pur}) measures the diversity of the ground-truth labels (predicted cluster labels) within each predicted cluster group (ground-truth group), respectively.

$$H_{acc} = -1 \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^{N_i} p_i(x_j) \log p_i(x_j)$$

$$H_{pur} = -1 \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{M_i} q_i(x_j) \log q_i(x_j)$$

where M is the total number of predicted clusters from the clustering algorithm, N is the number of ground-truth clusters, M_i , (or N_i) is the number of predicted clusters (or ground-truth clusters) in the i th ground-truth cluster (or predicted cluster), respectively. $p_i(x_j)$ (or $q_i(x_j)$) are the proportions of cells in the j th ground-truth cluster (or predicted cluster) relative to the total number of cells in the i th predicted cluster (or ground-truth clusters), respectively. A smaller value of H_{acc} means the cells in a predicted cluster are constantly labeled from the same ground-truth group, while A smaller value of H_{pur} means the cells in the ground-truth groups are homogeneous with the same predicted cluster labels (?). However, smaller H_{acc} (or H_{pur}) can lead to over-clustering (or under-cluster), when each predicted cluster contains 1 cell ($H_{acc} = 0$) or all cells in one predicted cluster ($H_{pur} = 0$).

- Adjusted Rand index (ARI). Rand index (RI) is another measure of constancy between two clustering outcomes. If a (or b) is the count of number of pairs of cells in one cluster (or different clusters) from one clustering algorithm but also fall in the same cluster (or different clusters) from the other clustering algorithm, then, $RI = (a + b) / \binom{n}{2}$, where $\binom{n}{2}$ is the total number of pairs when given n cells. The RI has a value between 0 and 1, with 0 indicating that the two clustering algorithms do not agree on any pair of cells and 1 indicating that the two clustering algorithms are exactly the same. ARI is a corrected-for-chance version of RI , or

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

where $E[RI]$ is the expected Rand Index (?).

- Median Silhouette index. The Silhouette index is defined as

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where $a(i)$ is the average dissimilarity of i th cell to all other cells in the same cluster, and $b(i)$ is the average dissimilarity of i th cell to all cells in the closest cluster. The range of $s(i)$ is $[-1, 1]$, with 1 to be well-clustered with appropriate labels, and -1 to be completely misclassified. $s(i) = 0$ indicates the cell could be assigned to nearest clusters (or overlapping clusters).

A good imputation method should allow perform downstream (clustering) analyses without introducing any artifacts or false signals.

4.1.2 Batch effect correction

When evaluating the performance of a batch correction method, we need to consider how well it mixes the shared cell types between different batches and at the same time identifies batch-specific cells. The existing metrics can be classified as cluster-level and cell-level metrics. Cluster level metrics are those used for evaluating clustering performance and include adjusted rand index (ARI), normalized mutual information (NMI), and silhouette coefficients. They are easy to compute but do not measure local mixture of cells from different batches. This drawback is addressed by the cell-level metrics, which includes k-nearest neighbor batch-effect test (kBET), local inverse Simpson's index (LISI), and classifier-based. Because the cluster-level metrics will be discussed in detail in Section ??, we focus on discussing cell-level metrics in this section.

Entropy of mixing. This metric evaluates the mixing of cells from different batches in the neighborhood of each cell (?). It first randomly sample N cells and then for each cell it calculates the regional entropy of mixing as

$$E = \sum_{i=1}^C p_i \log(p_i)$$

where C is the number of batches and p_i is the proportion of cells from batch i among N nearest cells (e.g. $N = 100$). The total entropy is the sum of regional entropies. The computation repeats K times to obtain an empirical distribution of the entropy of mixing.

Maximum Mean Discrepancy (MMD) is a non-parametric distance between distributions based on the reproducing kernel Hilbert space (RKHS) (?), or, MMD is a distance-based measure between two distribution p and q based on the mean embeddings μ_p and μ_q in a reproducing kernel Hilbert space F ,

$$MMD(F, p, q) = \sup_{f \in F} \|\mu_p - \mu_q\|_f$$

MMD will vanish for most finite samples x_k and y_k only if two distributions are the same.

k-Nearest neighbor batch-effect test (kBET). kBET assesses the batch mixing by comparing the batch-specific distribution within k -nearest neighbors (kNNs) of a cell with the global distribution of batches (?). It uses a X^2 -based test for random neighborhoods of fixed size to determine whether they are well mixed. Given a dataset of N cells from L batches with N_l denoting the number of cells in batch l . Under the null hypothesis that cells are ‘well mixed’, that is the absence of batch effect, we have the test statistics as

$$a_n^k = \sum_{l=1}^L \frac{(N_{nl}^k - k * f_l)^2}{k * f_l} \sim X_{L-1}^2$$

where N_{nl}^k is the number of cells from batch l in the k -nearest neighbors of cell n , f_l is the global fraction of cells in batch l , or $f_l = \frac{N_l}{N}$, and X_{L-1}^2 denotes the X^2 distribution with $L - 1$ degrees of freedom. The averaged rejection rate of the X^2 test for all cells is used to define the performance of a batch correction method.

Local Inverse Simpson’s Index (LISI). Like kBET, LISI also compares the batch mixing at local level with global batch distribution. However, unlike kBET, which is agnostic of cell types, LISI requires well mixing of cells from the same cell type but not of those from different types (?). LISI evaluates cell-type-specific mixing using an inverse Simpson’s Index in a local neighborhood of each cell. LISI builds Gaussian kernel-based distributions of local neighborhoods sensitive to local diversity. It calculates inverse Simpson’s Index in the k -nearest neighbors of cell n for all batches as $\frac{1}{\lambda(n)} = \frac{1}{\sum_{l=1}^L (p(l))^2}$, where $p(l)$ denotes the proportion of batch l in the k -nearest neighbors. The score reports the effective number of batches in the k -nearest neighbors of cell n . Inverse Simpson’s Index in the k -nearest neighbors of cell n can also be calculated to evaluate the diversity of different cell types. However, in an ideal case, LISI score should be 1, reflecting a separation of unique cell types.

Classifier-based. Although LISI addresses the issue of cell-type proportion of different batches but it is hard to summarize all single cell-level LISI scores into a simple statistic for comparing across different methods (?). The classifier-based approach addresses this issue by using two distinct local classifiers for each single

cell. The first classifier classifies every single cell as positive and negative cells. A single cell n is positive if at least 50 cells of its k -nearest neighbor (KNN) cells are from the same cell-type label, otherwise ‘negative’. The positive cells are further classified into true and false positive cells, where true positive cells are those surrounded by appropriate portions of cells with L batches. In other words, if we sample k cells from this cell-type cluster, the expected number of cells from batch l will be $k * f_l$, where f_l is the global fraction of cells in batch l . A positive cell in this cluster is a true positive when the observed cell numbers for each batch among its k neighbors are within 3 standard deviations around the expected numbers. The proportions of positive cells and true positive cells are used as the summary metrics to evaluate the overall performance of batch effect removal. The higher the proportions, the better the algorithm.

4.1.3 Clustering

Evaluating the performance of clustering algorithms is not as trivial as counting the number of errors like supervised learning. In general, the clustering performance evaluation metric should not just take absolute corrected labelled cells into account but also consider if the clustering defines a good similarity or separation in the dataset compared to ground truth. When ground truth is not available, evaluation must be performed using model itself such as clustering distance, dispersion, etc. Similar measures, such as Adjusted Rand Index (ARI) and Silhouette Index discussed in Section ?? can also be employed here to measure the agreement between predicted assignment to the ground-truth assignment.

Normalized Mutual Information (NMI). The mutual information (MI) [47] is a measure of mutual dependency between two cluster assignments U and V . It quantifies the amount of information we could have about one assignment by observing the other assignment. For N samples, we have the entropy for cluster assignments U and V as

$$H(U) = \sum_{i=1}^{|U|} P_U(i) \log(P_U(i)), H(V) = \sum_{i=1}^{|V|} P_V(i) \log(P_V(i))$$

where $P_U(i) = \frac{|U_i|}{N}$ and $P_V(j) = \frac{|V_j|}{N}$. Also, define the joint distribution probability is $P_{UV}(i, j) = \frac{|U_i \cap V_j|}{N}$. Then, the mutual information of U and V is defined as

$$MI(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P_{UV}(i, j) \log \frac{P_{UV}(i, j)}{P_U(i)P_V(j)}$$

The NMI is a normalization of the MI score between 0 and 1. For example, the average NMI is defined as (?)

$$NMI(U, V) = \frac{2 \times MI(U, V)}{[H(U) + H(V)]}$$

Homogeneity, Completeness, and V-Measure. The homogeneity score (HS) measures the extent to which its clusters contain only samples that belong to one cell type, or $HS = 1 - H(P(U|V))/H(P(U))$, where $H()$ is the entropy, and U is the ground-truth assignment and V is the predicted assignment. The HS range from 0 to 1, where 1 indicates perfectly homogeneous labelling. Similarly, the completeness score (CS) is defined as $CS = 1 - H(P(V|U))/H(P(V))$, its values range from 0 to 1, where 1 indicates all member from a ground-truth label are assigned to a single cluster.

The V-Measure [54(*Reference Not Found*)] is the harmonic mean between HS and CS , defined as $V_\beta = \frac{(1+\beta)HS \times CS}{\beta HS + CS}$, where β indicates the weight of HS . V-Measure is a more symmetric, i.e. switching the true and predicted cluster labels does not change V-Measure.

Chapter 5

Survey of deep learning models for scRNA-Seq analysis

In this section, we survey applications of DL models for scRNA-seq analysis. To better understand the relationship between the problems that each surveyed work addresses and the key challenges in the general scRNA-seq processing pipeline, we divide the survey into sections according to steps in the scRNA-seq processing pipeline illustrated in Fig.?? . For each DL model, we present the model details under the general model framework introduced in Section ?? and discuss the specific loss functions. We also survey the evaluation metrics and summarize the evaluation results. To facilitate cross-references of the information, we summarized all algorithms reviewed in this section in Table ?? and tabulate the datasets and evaluation metrics used in each paper in Tables ?? & ??. We also listed all other algorithms that each surveyed method evaluated against in Fig.??, highlighting the extensiveness these algorithms were assessed for their performance.

5.1 Imputation

The goal of imputation is to estimate the missing gene expression values due to dropout, or the failure to amplify the original RNA transcripts. These missing expression values can affect downstream bioinformatics analysis significantly as it decreases the power of the studies and introduces biases in gene expression (?). VAE, AE, and GAN have been applied for imputation and we review their specific model designs in this section.

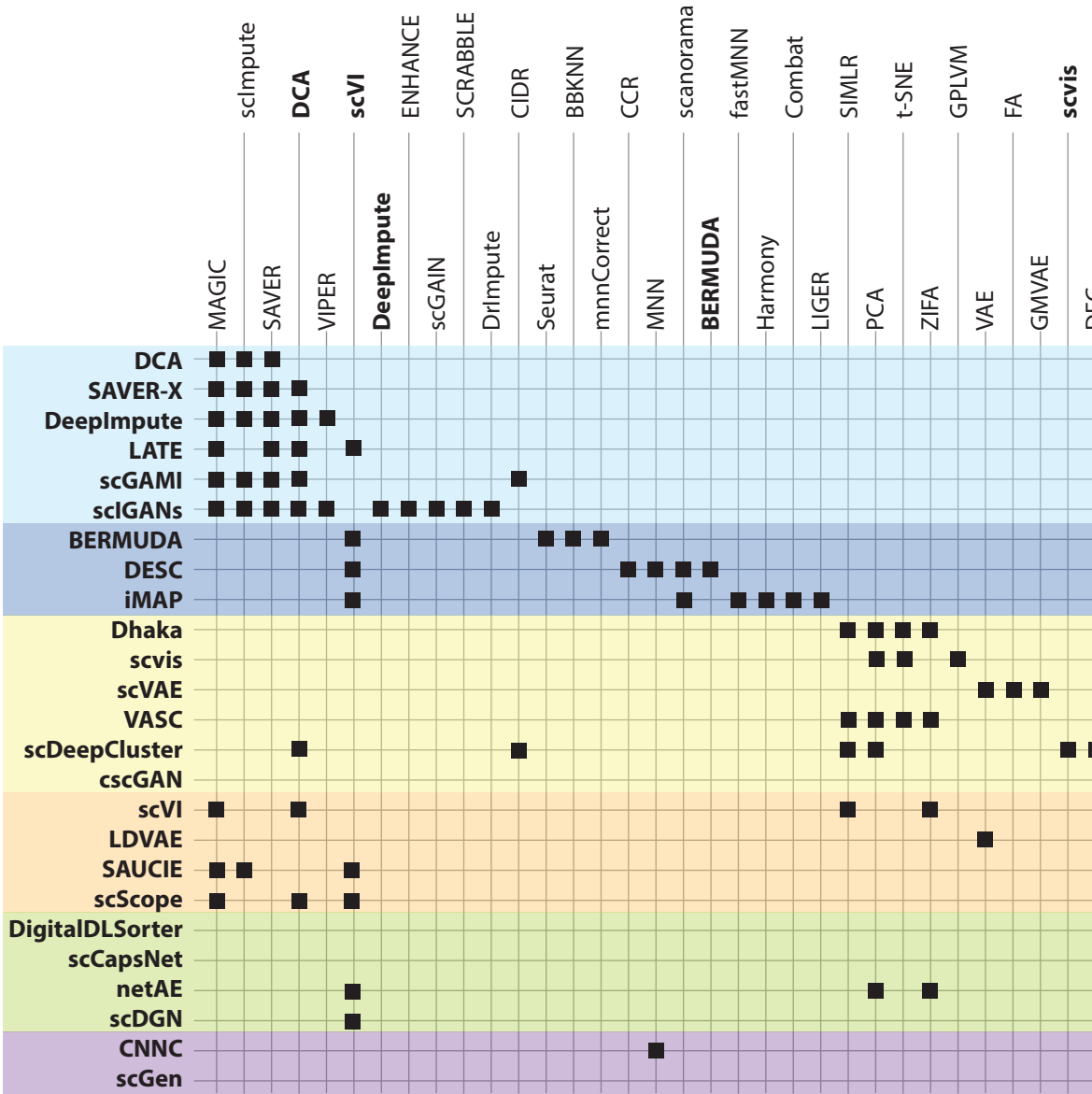


Figure 5.1: Add figure caption here

5.1.1 DCA:deep count autoencoder

DCA (?) is an AE designed for imputation (Fig.??-B. DCA is implemented in Python as a command line and also integrated into the Scanpy framework.

Model. DCA models UMI counts of a cell with missing values using the ZINB distribution as

$$p(x_{gn}|\Theta) = \pi_{gn}\delta(0) + (1 - \pi_{gn})NB(v_{gn}, \alpha_{gn}), \text{ for } g = 1, \dots, G; n = 1, \dots, N \quad (5.1)$$

where $\delta(\cdot)$ is a Dirac delta function, $NB(\cdot, \cdot)$ denotes the negative binomial distribution and $\pi_{gn}, v_{gn}, \text{ and } \alpha_{gn}$ are dropout rate, mean, and dispersion, which are functions of DCA decoder output $\hat{\mathbf{x}}_n$ as

$$\pi_n = \text{sigmoid}(\mathbf{W}\hat{\mathbf{x}}_n); \mathbf{v}_n = \exp(\mathbf{W}_v\hat{\mathbf{x}}_n); \alpha_n = \exp(\mathbf{W}_\alpha\hat{\mathbf{x}}_n) \quad (5.2)$$

where the g th element of π_n , \mathbf{v}_n , and α_n are π_{gn} and α_{gn} , respectively and \mathbf{W} , \mathbf{W}_v , and \mathbf{W}_α are additional weights to be estimated. The DCA encoder and decoder follow the general AE formulation as in Eq. ((?)) but the encoder takes the size factor normalized, log-transformed expression as input. The encoder and decoder architecture are the conventional deep neural networks. The parameters to be trained are $\Theta = \{\theta, \phi, W_\pi, W_v, W_\alpha\}$. To train the model, DCA uses a constrained log-likelihood as the loss function as

$$L(\Theta) = \sum_{n=1}^N \sum_{g=1}^G (-\log p(x_{gn}|\Theta) + \lambda \pi_{gn}^2) \quad (5.3)$$

where the second term functions as a ridge prior on the dropout probabilities. Once the DCA is trained, the mean counts v_n are used as the denoised and imputed counts for cell n .

Evaluation metrics. A Density Resampled Estimate of Mutual Information (DREMI) measure was adapted for the higher dimensionality and sparsity of scRNA-seq data.

Results. For evaluation, DCA is compared to other methods using simulated data (using Splatter R package), and a real bulk transcriptomics data from a developmental *C. elegans* time-course experiment was used with added simulating single-cell specific noise. For this, gene expression was measured from 206 developmentally synchronized young adults over a twelve-hour period (*C. elegans*). Single-cell specific noise was added in silico by genewise subtracting values drawn from the exponential distribution such that 80 of values were zeros. The paper analyzed the Bulk contains less noise than single-cell transcriptomics data and can thus aid in evaluating single-cell denoising methods

by providing a good ground truth model. The authors also did a comparison of other methods like SAVER (?), scImpute (?), and MAGIC (?). DCA denoising recovered original time-course gene expression pattern while removing single-cell specific noise. Overall, DCA demonstrates the strongest recovery of the top 500 genes most strongly associated with development in the original data without noise; is shown to outperform other existing methods in capturing cell population structure in real data (using PBMC, CITE-seq, runtime scales linearly with the number of cells).

5.1.2 SAVER-X: single-cell analysis via expression recovery harnessing external data

SAVER-X (?) (please check this) is an AE model developed to denoise and impute scrRNA-seq data with transfer learning from other data resources. SAVER-X is implemented in R with the support of Python package sctransfer. Model. SAVER-X decomposes the variation in the observed counts x_n with missing values into three components: i) predictable structured component that represents the shared variation across genes, ii) unpredictable cell-level biological variation and gene-specific dispersions, and iii) technical noise. Specifically, x_{gn} is modeled as a Poisson-Gamma hierarchical model,

$$p(x_{gn}|\Theta) = \text{Poisson}(l_n x'_{gn}), p(x'_{gn}|v_{gn}, \alpha_g) = \text{Gamma}(v_{gn}, \alpha_g v_{gn}^2) \quad (5.4)$$

where l_n is the sequencing depth of cell n , v_{gn} is the mean, and α_g is the dispersion. This Poisson-Gamma mixture is an equivalent expression to the NB distribution. As a result, the ZINB distribution as Eq. ((?)) in DCA is also adopted to model missing values. To train the model, a similar log-likelihood function as Eq. ((?)) in DCA is used as the loss function. However, v_{gn} is initially learned by an AE pre-trained using external datasets from an identical or similar tissue and then transferred to and updated using data x_n to be denoised. Such transfer learning can be applied to data between species (e.g., human and mouse in the study), cell types, batches, and single-cell profiling technologies. After v_{gn} is inferred, SAVER-X generates the final denoised data \hat{x}_{gn} by an empirical Bayesian shrinkage.

Evaluation metrics. t -SNE visualization and ARI were used to evaluate the clustering performance after imputation. Pearson correlation coefficients between protein and corresponding gene expression levels after denoising CITE-seq data were computed.

Results. SAVER-X was applied to multiple human single-cell datasets of different scenarios: i) T-cell subtypes, ii) a cell type (CD4+ regulatory T cells) that was absent from the pretraining dataset, iii) gene-protein correlations of CITE-seq data, and iv) immune cells of primary breast cancer samples with

a pretraining on normal immune cells. SAVER-X with pretraining on HCA and/or PBMCs outperformed the same model without pretraining and other denoising methods, including DCA (?), scVI (?), scImpute (?), and MAGIC (?). The model achieved promising results even for genes with very low UMI counts. SAVER-X was also applied for a cross-species study in which the model was pretrained on a human or mouse dataset and transferred to denoise another. The results demonstrated the merit of transferring public data resources to denoise in-house scRNA-seq data even when the study species, cell types, or single-cell profiling technologies are different.

5.1.3 DeepImpute (Deep neural network Imputation)

DeepImpute (?) is a deep neural network model that imputes genes in a divide-and-conquer approach. DeepImpute implemented in Keras framework/TensorFlow environment.

Model. For each dataset, DeepImpute selects to impute a list of genes or highly variable genes (variance over mean ratio, default = 0.5). Each sub-neural network aims to understand the relationship between the input genes (input layer) and a subset of target genes (output layer). Genes are first divided into N random subsets of size 512 called target genes. For each subset, a neural network of four layers (input, dense, dropout and output layers) is trained where the input layer includes genes (predictor genes) who are among top 5 best correlated genes to target genes but not part of the target genes in the subset. The loss is defined as the weighted MSE

$$\mathcal{L}_c = \sum x_n (x_n - \hat{x}_n)^2 \quad (5.5)$$

This function gives higher weights to genes with higher expression values, thus emphasizing accuracy on high confidence values and avoiding over penalizing genes with extremely low values.

Evaluation metrics. DeepImpute computes mean squared error (MSE) and Pearson’s correlation coefficient between imputed and true expression.

Result. DeepImpute had the highest overall accuracy and offered faster computation time with less demand on computer memory compared to other methods like MAGIC, DrImpute, ScImpute, SAVER, VIPER, and DCA. Using simulated and experimental datasets (Table ??), DeepImpute showed benefits in increasing clustering results and identifying significantly differentially expressed genes. DeepImpute and DCA, show overall advantages over other methods and between which DeepImpute performs even better. The properties of DeepImpute contribute to its superior performance include 1) a divide-and-conquer approach which contrary to an autoencoder as implemented in DCA, resulting in a lower complexity in each sub-model and stabilizing neural networks, and 2) the sub-networks are trained without using the target genes as the input which reduces

overfitting while enforcing the network to understand true relationships between genes.

5.1.4 LATE: Learning with AuToEncoder

LATE (?) is an AE whose encoder takes the log-transformed expression as input. LATE implemented in Python with TensorFlow.

Model. LATE sets zeros for all missing values at the input and generates the imputed expressions at the decoder's output. LATE experimented with three different network architectures composed of 1, 3 and 5 hidden layers. LATE minimizes the MSE loss as defined in Eq. ((?)). One problem with this model is that it assumes that all the zeros in the scRNA-seq data are missing values but some zeros could be real and reflect the actual lack of expression.

Evaluation metrics. Like DeepImpute, LATE used MSE to evaluate the performance.

Result. Using synthetic data generated from pre-imputed data followed with random dropout selection at different degree, LATE is shown to outperform other existing methods like MAGIC, SAVER, DCA, scVI, particularly when the ground truth contains only a few or no zeros. However, when the data contain many zero expression values, DCA achieved a lower MSE than LATE, although LATE still has a smaller MSE than scVI. This result suggests that DCA likely does a better job identifying true zeros gene expression, partly due to that LATE does not make assumptions on the statistical distributions of the single-cell data that potentially have inflated zero counts.

5.1.5 scGMAI

Technically, scGMAI (?) is a model for clustering but it includes an AE in the first step to combat dropout. The scGMAI's AE model is implemented with TensorFlow.

Model. To impute the missing values, scGMAI applies an AE like LATE to reconstruct log-transformed expressions with dropout. One difference is that it chooses Softplus as the activation function since it is smoother than ReLU and thus more suitable for scRNA-seq data. The MSE loss as in (9) is adopted. After imputation, scGMAI uses fast independent component analysis (ICA) on the AE reconstructed expression to reduce the dimension and then applies a Gaussian mixture model on the ICA reduced data to perform the clustering.

Evaluation metrics. It used clustering metrics including NMI, ARI, Homogeneity, and Completeness to evaluate the performance.

Results. To assess the performance, the AE in scGMAI was replaced by five other imputation methods including SAVER (?), MAGIC (?), DCA (?), scImpute (?), and CIDR (?). A scGMAI implementation without AE was also

compared. Seventeen scRNA-seq data (part of them are listed in Table ?? as marked) were used to evaluate cell clustering performances. The results indicated that the AEs significantly improved the clustering performance in eight of seventeen scRNA-Seq datasets.

5.1.6 scIGANs

Imputation approaches based on information from cells with similar expressions suffer from oversmoothing, especially for rare cell types. scIGANs (?) is a GAN-based imputation algorithm, which overcomes this problem by using the observed samples with missing values to train a GAN to generate samples with imputed expressions.

Model. The gene expression vector x_n is first reshaped into a square image-like format and scIGAN takes the reshaped data as input. The model follows a BE-GAN (?) framework, which substitutes the generator with an autoencoder that includes an encoder E and a decoder G and also replaces the discriminator D with a function R_{ϕ_R} that computes the reconstruction error of the autoencoder (e.g. MSE). Then, the Wasserstein distance between the reconstruction errors of the real and generated samples are computed as the loss

$$L(\theta, \phi) = \max_{\phi_R} \sum_{n=1}^N R_{\phi_R}(x_n) - \sum_{n=1}^N R_{\phi_R}(G_{\theta}(E_{\phi}(x_n), y)) \quad (5.6)$$

The encoder and decoder are trained in a GAN fashion to minimize this Wasserstein distance. This framework forces the model to meet two computing objectives, i.e. reconstructing the real samples and discriminating between real and generated samples. Proportional Control Theory was applied to balance these two goals during the training.

After training, the decoder G_{θ} is used to generate new samples of a specific cell type. Then, the k-nearest neighbors (KNN) approach is applied to the real and generated samples to impute the real samples' missing expressions.

Evaluation metrics. It used a variety of clustering- and classification-based metrics including ARI, ACC, AUC and F-score.

Results. scIGANs was first tested on simulated samples with different dropout rates. Performance of rescuing the correct clusters was compared with 11 existing imputation approaches including DCA, DeepImpute, SAVER, scImpute, MAGIC, etc. scIGANs reported the best performance for all metrics. scIGAN was next evaluated for correctly clustering cell types on the Human brain scRNA-seq data and showed superior performance than existing methods again. scIGANs was next evaluated for identifying cell-cycle states using scRNA-seq datasets from mouse embryonic stem cells. The results showed that scIGANs outperformed competing existing approaches for recovering subcellular states of cell cycle dynamics. scIGANs was further shown to improve the identification

of differentially expressed genes and enhance the inference of cellular trajectory using time-course scRNA-seq data from the differentiation from H1 ESC to definitive endoderm cells (DEC). Finally, scIGAN was also shown to scale to scRNA-seq methods and data sizes.

5.2 Batch effect correction

The goal of imputation is to estimate the missing gene expression values due to dropout, or the failure to amplify the original RNA transcripts. These missing expression values can affect downstream bioinformatics analysis significantly as it decreases the power of the studies and introduces biases in gene expression (?). VAE, AE, and GAN have been applied for imputation and we review their specific model designs in this section.

5.2.1 BERMUDA: Batch Effect ReMoval Using Deep Autoencoders

BERMUDA (?) deploys a transfer-learning method to remove the batch effect. It performs correction to the shared cell clusters among batches and therefore preserves batch-specific cell populations. Model. BERMUDA has a conventional AE architecture that takes normalized, log-transformed expression of a cell as input. It has the general loss function but consists of two parts as

$$L(\Theta) = L_0(\Theta) + \lambda L_{MMD}(\Theta) \quad (5.7)$$

where $L_0(\Theta)$ is the MSE reconstruction loss as defined in Eq. ((?)) and L_{MMD} is the maximum mean discrepancy (MMD) loss that measures the differences in distributions among similar cell clusters in different batches. MMD is a non-parametric distance between distributions based on the reproducing kernel Hilbert space (RKHS) (?). Instead of applying the MMD loss on batches entirely, BERMUDA considers the loss only between pairs of similar cell clusters shared among batches, where the MMD loss is defined as:

$$L_{MMD}(\Theta) = \sum_{i_a, i_b, j_a, j_b} M_{i_a, i_b, j_a, j_b} MMD(z_{i_a, j_a}, z_{i_b, j_b}) \quad (5.8)$$

where $z_{i,j}$ is the latent variable of $x_{i,j}$, the input expression profile of a cell from cluster j of batch i , M_{i_a, i_b, j_a, j_b} is 1 if cluster i_a of batch j_a and cluster i_b of batch j_b are determined to be similar by MetaNeighbor (?) and 0, otherwise. $MMD()$ equals zero when the underlying distributions of the observed samples are the same. By minimizing the MMD loss between the distributions of the latent variables of similar clusters, BERMUDA can be trained to remove batch effects in its latent variables.

Evaluation metrics. Evaluation of BERMUDA included three metrics: KBET, entropy of mixing, and silhouette index.

Results. BERMUDA was shown to outperform other methods like mnnCorrect (?), BBKNN (?), Seurat (?), and scVI (?) in removing batch effects on simulated and human pancreas data while preserving batch-specific biological signals. BERMUDA provides several improvements compared to existing methods: 1) capable of removing batch effects even when the cell population compositions across different batches are vastly different; and 2) preserving batch-specific biological signals through transfer-learning which enables discovering new information that might be hard to extract by analyzing each batch individually.

5.2.2 DESC: batch correction based on clustering

DESC (?) is an AE that removes batch effect through clustering with hypothesis that batch differences in expressions are smaller than true biological variations between cell types, and, therefore, properly performing clustering on cells across multiple batches can remove batch effects without the need to define batches explicitly.

Model. DESC has a conventional AE architecture. Its encoder takes normalized, log-transformed expression as the input and uses decoder output \hat{x}_n as the reconstructed gene expression, which is equivalent to a Gaussian data distribution with \hat{x}_n being the mean. The loss function is

$$L(\Theta) = L_0(\Theta) + \gamma L_c(\Theta) \quad (5.9)$$

where L_0 is reconstruction MSE as defined in Eq. ((?)) and L_c is the clustering loss that regularizes the learned feature representations to form clusters. The clustering loss follows the design in the deep embedded clustering (?) . Let $k \in \{1, \dots, K\}$ be the cluster index of a cell and assume that there are K clusters. Then, the clustering loss is defined as a Kullback–Leibler (KL) divergence

$$L_c(\Theta) = KL(P||Q) = \sum_{n=1}^N \sum_{k=1}^K p_{nk} \log \frac{p_{nk}}{q_{nk}} \quad (5.10)$$

where q_{nk} is the probability of cell n belonging to cluster k and computed as

$$q_{nk} = \frac{(1 + \|z_n - \mu_k\|^2)^{-1}}{\sum_{k'} (1 + \|z_n - \mu_{k'}\|^2)^{-1}} \quad (5.11)$$

where μ_k is the center of cluster k and p_{nk} is the target distribution calculated by normalizing q_{nk}^2 with frequency per cluster

$$p_{nk} = \frac{q'_{nk}}{\sum_{k'=1}^K q'_{nk'}}, \text{ and } q'_{nk} = \frac{q_{nk}^2}{\sum_{n=1}^N q_{nk}} \quad (5.12)$$

A standard k -means clustering algorithm is used to initialize cluster centers. The model is first trained to minimize L_0 only to obtain the initial weights before the model is trained to optimize the combined loss (19). When the training is done, each cell's cluster index can be assigned based on p_{nk} . After the training, each cell is assigned with a cluster ID.

Evaluation metrics. ARI was used for clustering and the KL divergence (Eq. (??)) was used to assess the batch effect removal.

Results. DESC was applied to the macaque retina dataset, which includes animal level, region level, and sample-level batch effects. The results showed that DESC is effective in removing the batch effect, whereas CCA (?), MNN (?), Seurat 3.0 (?), scVI (?), BERMUDA (?), and scanorama (?) were all sensitive to batch definitions. DESC was then applied to human pancreas datasets to test its ability to remove batch effects from multiple scRNA-seq platforms and yielded the highest ARI among the comparing approaches mentioned above. When applied to human PBMC data with interferon-beta stimulation, where biological variations are compounded by batch effect, DESC was shown to be the best in removing batch effect while preserving biological variations. DESC was also shown to remove batch effect for the monocytes and mouse bone marrow data and DESC was shown to preserve the pseudotemporal structure. Finally, DESC scales linearly with the number of cells and its running time is not affected by the increasing number of batches.

5.2.3 iMAP: Integration of Multiple single-cell datasets by Adversarial Paired-style transfer networks

iMAP (?) combines AE and GAN for batch effect removal. It is designed to remove batch biases while preserving dataset-specific biological variations.

Model. iMAP consists of two processing stages, each including a separate DL model. In the first stage, a special AE, whose decoder combines the output of two separate decoders D_{θ_1} and D_{θ_2} , is trained such that

$$z_n = E_{\phi}; \hat{x}_n = D_{\theta}(z_n, s_n) = ReLu(D_{\theta_1}(s_n) + D_{\theta_2}(z_n, s_n)) \quad (5.13)$$

where s_n is the one-hot encoded batch number of cell n . D_{θ_1} can be understood as decoding the batch noise, whereas D_{θ_2} reconstructs batch-removed expression from the latent variable z_n . The training minimizes the loss

$$L(\Theta) = L_0(\Theta) + \gamma L_t(\Theta) \quad (5.14)$$

5.3. DIMENSION REDUCTION, LATENT REPRESENTATION, CLUSTERING, AND DATA AUGMENTATION

where $L_0(\Theta)$ is the MSE reconstruction loss defined in eq. ((?)) and L_c is the content loss

$$L_t(\Theta) = \sum_{n=1}^N \|z_n - E_\phi(D_\theta(z_n, \tilde{s}_n))\|_2^2 \quad (5.15)$$

where \tilde{s}_n is a random batch number. Minimizing $L_t(\Theta)$ further ensures the reconstructed expression \hat{x}_n would be batch agnostic and has the same content as x_n . However, the author indicated that due to the limitation of AE, this step is still insufficient for batch removal. Therefore, a second stage is included to apply a GAN model to make expression distributions of the shared cell type across different batches indistinguishable. To identify the shared cell types, a mutual nearest neighbors (MNN) strategy adapted from (?) was developed to identify MNN pairs across batches using batch effect independent z_n as opposed to x_n . Then, a mapping generator G_{θ_G} is trained using MNN pairs based on GAN such that $x_n^{(A)} = G_{\theta_G}(x_n^{(S)})$, where $x_n^{(S)}$ and $x_n^{(A)}$ are the MNN pairs from batch S and an anchor batch A . The WGAN-GP loss as in Eq. ((?)) was adopted for the GAN training. After training, G_{θ_G} is applied to all cells of a batch to generate batch-corrected expression.

Evaluation metrics. The classifier-based metric as described in section ?? was used.

Results: iMAP was first tested on benchmark datasets from human dendritic cells and Jurkat and 293T cell lines and then human pancreas datasets from five different platforms. All the datasets contain both batch-specific cells and batch-shared cell types. iMAP was shown to separate the batch-specific cell types but mix batch shared cell types and outperformed 9 other existing batch correction methods including Harmony, scVI, fastMNN, Seurat, etc. iMAP was then applied to the large-scale Tabula Muris datasets containing over 100K cells sequenced from two platforms. iMAP could not only reliably integrate cells from the same tissues but identify cells from platform-specific tissues. Finally, iMAP was applied to datasets of tumor-infiltrating immune cells and shown to reduce the dropout ratio and the percentage of ribosomal genes and noncoding RNAs, thus improving detection of rare cell types and ligand-receptor interactions. iMAP scales with the number of cells, showing minimal time cost increase after the number of cells exceeds thousands. Its performance is also robust against model hyperparameters.

5.3 Dimension reduction, latent representation, clustering, and data augmentation

Dimension reduction is indispensable for many type of scRNA-seq data analysis, considering the limited number of cell types in each biospecimen. Furthermore,

biological processes of interests often involve the complex coordination of many genes, therefore, latent representation which capture biological variation in reduced dimensions are useful in interpreting many experiment conditions. In addition, many deep learning models further exploit latent dimensions and generative factors to produce augmented data that may in return to enhance the clustering, e.g., due to low representation of certain cell types. Therefore, we categorize all these algorithms together in this section.

5.3.1 Dimension reduction by AEs with gene-interaction constrained architecture

This study (?) considers AEs for learning the low-dimensional representation and specifically explores the benefit of incorporating prior biological knowledge of gene-gene interactions to regularize the AE network architecture.

Model. Several AE models with single or two hidden layers that incorporate gene interactions reflecting transcription factor (TF) regulations and protein-protein interactions (PPIs) are implemented. The models take normalized, log-transformed expressions and follow the general AE structure, including dimension-reducing and reconstructing layers, but the network architectures are not symmetrical. Specifically, gene interactions are incorporated such that each node of the first hidden layer represented a TF or a protein in the PPI; only genes that are targeted by TFs or involved in the PPI were connected to the node. Thus, the corresponding weights of E_ϕ and D_θ are set to be trainable and otherwise fixed at zero throughout the training process. Both unsupervised (AE-like) and supervised (cell-type label) learning were studied.

Evaluation metrics. Performance of cells clustering was evaluated by six metrics including NMI, ARI, completeness, Fowlkes–Mallows score (?), homogeneity, and v-measure (?). Performance of cell-type retrieval was evaluated by the mean of average precision.

Results. Regularizing encoder connections with TF and PPI information considerably reduced the model complexity by almost 90% (7.5-7.6M to 1.0-1.1M). The clusters formed on the data representations learned from the models with or without TF and PPI information were compared to those from PCA, NMF, independent component analysis (ICA), t-SNE, and SIMLR (?). The model with TF/PPI information and 2 hidden layers achieved the best performance by five of the six measures (0.87-0.92) and the best average performance (0.90). In terms of the cell-type retrieval of single cells, the encoder models with and without TF/PPI information achieved the best performance in 4 and 3 cell types, respectively. PCA yielded the best performance in only 2 cell types. The DNN model with TF/PPI information and 2 hidden layers again achieved the best average performance (mean of average precision, 0.58) across all cell types. In summary, this study demonstrated a biologically meaningful way to

5.3. DIMENSION REDUCTION, LATENT REPRESENTATION, CLUSTERING, AND DATA AUGMENTATION

regularize AEs by the prior biological knowledge for learning the representation of scRNA-Seq data for cell clustering and retrieval.

5.3.2 Dhaka: a VAE-based dimension reduction model

Dhaka (?) was proposed to reduce the dimension of scRNA-Seq data for efficient stratification of tumor subpopulations.

Model. Dhaka adopts a general VAE formulation. It takes the normalized, log-transformed expressions of a cell as input and outputs the low-dimensional representation.

Evaluation metrics. ARI was used to determine the quality of the resulting clustering for each dimensionality reduction method. Spearman rank correlation was assessed to the scoring metric (lineage or differentiation) to contrast with other programs.

Result. Dhaka was first tested on the simulated dataset. The simulated dataset contains 500 cells, each including 3K genes, clustered into 5 different clusters with 100 cells each. The clustering performance was compared with other methods including t-SNE, PCA, SIMLR, NMF, an autoencoder, MAGIC, and scVI. Dhaka was shown to have an ARI higher than most other comparing methods. Dhaka was then applied to the Oligodendroglioma data and could separate malignant cells from non-malignant microglia/macrophage cells. It also uncovered the shared glial lineage and differentially expressed genes for the lineages. Dhaka was also applied to the Glioblastoma data and revealed an evolutionary trajectory of the malignant cells where cells were gradually progressing from a stemlike state to a more differentiated state. In contrast, other methods failed to capture this underlying structure. Dhaka was next applied to the Melanoma cancer dataset (?) and uncovered two distinct clusters that showed the intra-tumor heterogeneity of the Melanoma samples. Dhaka was finally applied to copy number variation data (?) and shown to identify one major and one minor cell clusters, of which other methods could not find.

5.3.3 scvis: a VAE for capturing low-dimensional structures

scvis (?) is a VAE network that learns the low-dimensional representations capture both local and global neighboring structures in scRNA-Seq data.

Model: scvis adopts the generic VAE formulation described in section ?? However, it has a unique loss function defined as

$$L(\Theta) = -\mathcal{L}(\Theta) + \lambda L_t(\Theta) \quad (5.16)$$

where $L(\Theta)$ is ELBO as in Eq. (??) and L_t is a regularizer using non-symmetrized t-SNE objective function (?), which is defined as

$$L_t(\Theta) = \sum_{i=1}^N \sum_{j=1, j \neq i}^N p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \quad (5.17)$$

where i and j are two different cells, $p_{i|j}$ measures the local cell relationship in the data space, and $q_{j|i}$ measures such relationship in the latent space as

$$p_{i|j} = \frac{\exp(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2})}, q_{j|i} = \frac{(1 + \|z_i - z_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|z_i - z_k\|^2)^{-1}} \quad (5.18)$$

with σ_i defined as the perplexity (?). Because t-SNE algorithm preserves the local structure of high dimensional space after projecting to the lower dimension, L_t promotes the learning of local structures of cells.

Evaluation matrices. KNN preservation and log-likelihood of low dimensional mapping are used to evaluate model performance.

Results. scvis was tested on the simulated data and outperformed t-SNE in a nine-dimensional space task. scvis preserved both local structure and global structure. The relative positions of all clusters were well kept but outliers were scattered around clusters. Using simulated data and comparing to t-SNE, scvis generally produced consistent and better patterns among different runs while t-SNE could not. scvis also presented good results on adding new data to an existing embedding, with median accuracy on new data at 98.1% for K= 5 and 94.8% for K= 65, when train K cluster on original data then test the classifier on new generated sample points. scvis was subsequently tested on four real datasets including metastatic melanoma, oligodendroglioma, mouse bipolar and mouse retina datasets. In each dataset, scvis was showed to preserve both the global and local structure of the data.

5.3.4 scVAE: VAE for single-cell gene expression data

scVAE (?) includes multiple VAE models for denoising gene expression levels and learning the low-dimensional latent representation of cells. It investigates different choices of the likelihood functions in the VAE model to model different data sets.

Model. scVAE is a conventional fully connected network. However, different distributions have been discussed for $p(x_{gn}|v_{gn}, \alpha_{gn})$ to model different data behaviors. Specifically, scVAE considers Poisson, constrained Poisson, and negative binomial distributions for count data, piece-wise categorical Poisson for data including both high and low counts, and zero-inflated version of these distributions to model missing values. To model multiple modes in cell expressions,

5.3. DIMENSION REDUCTION, LATENT REPRESENTATION, CLUSTERING, AND DATA AUGMENTATION

a Gaussian mixture is also considered for $q(z_n|x_n, s_n)$, resulting a GMVAE. The inference process still follows that of a VAE as discussed in section 3.1.

Evaluation metrics. ARI was used to assess the performance.

Results. scVAEs were evaluated on the PBMC data and compared with factor analysis (FA) models. The results showed that GMVAE with negative binomial distribution achieved the highest lower bound and ARI. Zero-inflated Poisson distribution performed the second best. All scVAE models outperformed the baseline linear factor analysis model, which suggested that a non-linear model is needed to capture single-cell genomic features. GMVAE was also compared with Seurat and shown to perform better using the withheld data. However, scVAE performed no better than scVI (?) or scvis (?), both are VAE models.

5.3.5 VASC: VAE for scRNA-seq

VASC (?) is another VAE for dimension reduction and latent representation but it models dropout.

Model: VASC’s input is the log-transformed expression but rescaled in the range $[0,1]$. A dropout layer (dropout rate of 0.5) is added after the input layer to force subsequent layers to learn to avoid dropout noise. The encoder network has three layers fully connected and the first layer uses linear activation, which acts like an embedded PCA transformation. The next two layers use the ReLU activation, which ensures a sparse and stable output. This model’s novelty is the zero-inflation layer (ZI layer), which is added after the decoder to model scRNA-seq dropout events. The probability of dropout event is defined as $e^{-\hat{x}^2}$ where \hat{x} is the recovered expression value obtained by the decoder network. Since backpropagation cannot deal with stochastic network with categorical variables, a Gumbel-softmax distribution (?) is introduced to address the difficulty of ZI layer. The loss function of the model takes the form $L = L_0(\Theta) + \lambda L_c(\Theta)$, where L_0 is the binary entropy because of the input is scaled to $[0, 1]$, and L_c a loss performed using KL divergence $KL(Q(z|x)||P(z))$, where z is the latent variables (Gaussian distribution). After the model is well trained, the latent code can be used as the dimension-reduced features for downstream tasks and visualization.

Evaluation metrics. Four measures are used to assess the performance including NMI, ARI, homogeneity, and completeness.

Results. VASC was compared with PCA, t-SNE, ZIFA, and SIMLR on 20 datasets. In the study of embryonic development from zygote to blast cells, all methods roughly re-established the development stages of different cell types in the dimension-reduced space. However, VASC showed better performance to model embryo developmental progression. In the Goolam, Biase and Yan datasets, scRNA-seq data were generated through embryonic development stages from zygote to blast, VASC re-established development stage from 1, 2,

4, 8, 16 to blast, while other methods failed. In the Pollen, Kolodziejczyk, and Baron dataset, VASC formed appropriate cluster, either with homogeneous cell type, preserved proper relative positions, or minimal batch influence. Interestingly, tested on the PBMC dataset, VASC showed to identify the major global structure (B cells, CD4+, CD8+ T cells, NK cells, Dendritic cells), it also detected subtle differences within monocytes (FCGR3A+ vs CD14+ monocytes), indicating the capability of VASC handling large number of cells or cell types. Quantitative clustering performance in NMI, ARI, homogeneity and completeness was also performed. VASC always ranked top two in all the datasets. In terms of NMI and ARI, VASC best performed on 15 and 17 out of 20 datasets, respectively.

5.3.6 scDeepCluster

scDeepCluster (?) is an AE network that simultaneously learns feature representation and performs clustering via explicit modeling of cell clusters as in DESC.

Model: Similar to DCA, scDeepCluster adopts a ZINB distribution for x_n as in Eq. ((?)) and ((?)). The loss function is

$$L(\Theta) = L_0(\Theta) + \gamma L_c(\Theta) \quad (5.19)$$

where L_0 is the negative log-likelihood of the ZINB data distribution as defined in Eq. ((?)) and L_c a clustering loss performed using KL divergence as Eq. ((?)) defined in DESC algorithm. Comparing to scvis, in terms of clustering regularization, scvis uses t-SNE objective function which is faithful to feature representation by keeping local structure, while scDeepcluster uses KL divergence based clustering which focuses more on clustering assignment.

Evaluation metrics. The following three metrics are used to evaluate the performance, NMI, clustering accuracy, and ARI.

Results. scDeepCluster was first tested on the simulation data and compared with other seven methods including DCA (?), two multi-kernel spectral clustering methods MPSSC (?) and SIMLR (?), CIDR (?), PCA + k-mean, scvis (?) and DEC (?). In different dropout rate simulations, scDeepCluster significantly outperformed the other methods consistently. In signal strength, imbalanced sample size, and scalability simulations, scDeepcluster outperformed all other algorithms and scDeepCluster and most notably advantages for weak signals, robust against different data imbalance levels and scaled linearly with the number of cells. scDeepCluster was then tested on four real datasets (10X PBMC, Mouse ES cells, Mouse bladder cells, Worm neuron cells) and shown to outperform all other comparing algorithms. Particularly, MPSSC and SIMLR failed to process the full datasets due to quadratic complexity.

5.3.7 cscGAN: Conditional single-cell generative adversarial neural networks

cscGAN (?) is a GAN model designed to augment the existing scRNA-seq samples by generating expression profiles of specific cell types or subpopulations.

Model. Two models, csGAN and cscGAN, were developed following the general formulation of WGAN described in section ?? The difference between the two models is that cscGAN is a conditional GAN such that the input to the generator also includes a class label y or cell type, i.e. $\phi_G(z, y)$. The projection-based conditioning (PCGAN) method (?) was adopted to obtain the conditional GAN. For both models, the generator (three layers of 1024, 512, and 256 neurons) and discriminator (three layers of 256, 512, and 1024 neurons) are fully connected DNNs.

Evaluation metrics. The performance of cscGAN was assessed qualitatively by comparing the t-SNE plots of the real and generated samples. In the first approach, cluster-specific marker genes were obtained and SCENIC (?) was applied to learn the regulons for data including real and generated samples. Then, the ability of cscGAN to uncover the regulons from the real data was assessed. In the second approach, MMD between real and generated samples from cscGAN and other approaches were computed to measure their similarity. In the third approach, a classifier was trained to discriminate real from generated samples and the classification AUC was compared with that of a random guess.

Results: The performance of scGAN was first evaluated using PBMC data. The generated samples were shown to capture the desired clusters and the real data's regulons. Additionally, the AUC performance for classifying real from generated samples by a Random Forest classifier only reached 0.65, performance close to 0.5. Finally, scGAN's generated samples had a smaller MMD than those of Splatter, a state-of-the-art scRNA-seq data simulator (?). Even though a large MMD was observed for scGAN when compared with that of SUGAR, another scRNA-seq simulator, SUGAR (?) was noted for prohibitively high runtime and memory. scGAN was further trained and assessed on the bigger mouse brain data and shown to model the expression dynamics across tissues. Then, the performance of cscGAN for generating cell-type-specific samples was evaluated using the PBMC data. cscGAN was shown to generate high-quality scRNA-seq data for specific cell types. Finally, the real PBMC samples were augmented with the generated samples. This augmentation improved the identification of rare cell types and the ability to capture transitional cell states from trajectory analysis.

5.4 Multi-functional models

Given the versatility of AE and VAE in addressing different scRNA-seq analysis challenges, DL models possessing multiple analysis functions have been devel-

oped. We survey these models in this section.

5.4.1 scVI: single-cell variational inference

scVI (?) is designed to address a range of fundamental analysis tasks, including batch correction, visualization, clustering, and differential expression.

Model. scVI is a VAE that models the counts of each cell from different batches. scVI adopts a ZINB distribution for x_{gn}

$$p(x_{gn}|\pi_{gn}, L_n, \nu_{gn}, \alpha) = \pi_{gn}\delta(0) + (1 - \pi_{gn})NB(L_n\nu_{gn}, \alpha_g) \quad (5.20)$$

which is defined similarly as Eq ((?)) in DCA, except that L_n denotes the scaling factor for cell n , which follows a log-Normal ($\log \mathcal{N}$) prior as $p(L_n) = \log \mathcal{N}(\mu_{L_n}, \sigma_{L_n}^2)$, therefore, ν_{gn} represents the mean counts normalized by L_n . Now, let $s_n \in \{0, 1\}^B$ be the batch ID of cell n with B being the total number of batches. Then, ν_{gn} and π_g are further modeled as functions of the d-dimension latent variable $z_n \in \mathbb{R}^d$ and the batch ID s_n by the decoder networks D_{θ_ν} and D_{θ_π} as

$$x_{gn} = D_{\theta_\nu}(z_n, s_n), \pi_{gn} = D_{\theta_\pi}(z_n, s_n) \quad (5.21)$$

where the g th element of x_{gn} and π_{gn} are ν_{gn} and π_g , respectively, and θ_ν and θ_π are the decoder weights. Note that the lower layers of the two decoders are shared. For inference, both z_n and L_n are considered as latent variables and therefore $q(x_n, s_n) = q(z_n|x_n, s_n)q(L_n|x_n, s_n)$ is a mean-field approximate to the intractable posterior distribution $p(z_n, L_n|x_n, s_n)$ and

$$q(z_n | x_n, s_n) = \mathcal{N}(\mu_{z_n}, \text{diag}(\sigma_{z_n}^2)), q(L_n | x_n, s_n) = \log \mathcal{N}(\mu_{L_n}, \text{diag}(\sigma_{L_n}^2)) \quad (5.22)$$

whose means and variances $\{\mu_{z_n}, \sigma_{z_n}^2\}$ and $\{\mu_{L_n}, \sigma_{L_n}^2\}$ are defined by the encoder networks E_Z and E_L applied to x_n and s_n as

$$\{\mu_{z_n}, \sigma_{z_n}^2\} = E_{\phi_z}(x_n, s_n), \{\mu_{L_n}, \sigma_{L_n}^2\} = E_{\phi_L}(x_n, s_n) \quad (5.23)$$

where ϕ_z , and ϕ_L are the encoder weights. Note that, like the decoders, the lower layers of the two encoders are also shared. Overall, the model parameters to be estimated by the variational optimization is $\Theta = \{\theta_\nu, \theta_\pi, \phi_z, \phi_L, \alpha_g\}$. After inference is made, the latent vectors z_n are used for visualization and clustering. ν_{gn} provides a batch-corrected, size-factor normalized estimate of gene expression for each gene g in each cell n . An added advantage of the probabilistic representation by scVI is that it provides a natural probabilistic treatment of

the subsequent differential analysis, resulting in lower variance in the adopted hypothesis tests.

Evaluation metrics: Run-time vs. dataset size was used to assess the scalability. To assess imputation, randomly selected non-zero entries in the count matrix were used as hold-out data and the L_1 distance between the imputed and the original values in the hold-out data were used to measure the imputation accuracy. For clustering, ARI, NMI, and the Silhouette index were used. The entropy of batch mixing was adopted to measure batch correction performance.

Results: scVI was evaluated for its scalability, the performance of imputation. For scalability, ScVI was shown to be faster than most nonDL algorithms and scalable to handle twice as many cells as nonDL algorithms with a fixed memory. For imputation, ScVI, together with other ZINB-based models, performed better than methods using alternative distributions. However, it underperformed for the dataset (HEMATO) with fewer cells. For the latent space, scVI was shown to provide a comparable stratification of cells into previously annotated cell types. Although scVI failed to ravel SIMLR, it is among the best in capturing biological structures (hierarchical structure, dynamics, etc.) and recognizing noise in data. For batch correction, it outperforms ComBat. For normalizing sequencing depth, the size factor inferred by scVI was shown to be strongly correlated with the sequencing depth. Interestingly, the negative binomial distribution in the ZINB was found to explain the proportions of zero expressions in the cells, whereas the zero probability π_{gn} is found to be more correlated with alignment errors. For differential expression analysis, scVI was shown to be among the best.

5.4.2 LDVAE: linearly decoded variational autoencoder

LDVAE (?) is an adaption of scVI to improve the model interpretability but it still benefits from the scalability and efficiency of scVI. Also, this formulation applies to general VAE models and thus is not restricted to scRNA-seq analysis.

Model. LDVAE follows scVI’s formulation but replaces the decoder D_{θ_ν} in ((?)) by a linear model

$$\mathbf{v}_n = \mathbf{W}\mathbf{z}_n \quad (5.24)$$

where $\mathbf{W} \in \mathbb{R}^{d \times G}$ is the weight matrix. Being the linear decoder provides interpretability in the sense that the relationship between latent representation \mathbf{z}_n and gene expression \mathbf{v}_n can be readily identified. LDVAE still follows the same loss and non-linear inference scheme as scVI.

Evaluation metrics. The reconstruction errors of the VAEs were used to assess the performance.

Results. LDVAE’s latent variable z_n could be used for clustering of cells with similar accuracy as a VAE. Although LDVAE had a higher reconstruction error than VAE, due to the linear decoder, the variations along the different axes of z_n establish direct linear relationships with input genes. As an example from analyzing mouse embryo scRNA-seq, $z_{1,n}$, the second element of z_n , is shown to relate to simultaneous variations in the expression of gene *Pou5f1* and *Tdgf1*. In contrast, such interpretability would be intractable without approximation for a VAE. LDVAE was also shown to induce fewer correlations between latent variables and improved the grouping of the regulatory programs. LDVAE also scaled to a large dataset with $\sim 2M$ cells.

5.4.3 SAUCIE

SAUCIE (?) is an AE designed to perform multiple functions, including clustering, batch correlation, imputation, and visualization. SAUCIE is applied to the normalized data instead of count data.

Model. SAUCIE includes multiple model components designed for different functions.

1. Clustering: SAUCIE first introduced a “digital” binary encoding layer $\mathbf{h}^c \in \{0, 1\}^J$ in the decoder D that functions to encode the cluster ID. To learn this encoding, an entropy loss is introduced

$$L_D = \sum_{k=1}^K p_k \log p_k \quad (5.25)$$

where p_k is the probability (proportion) of activation on neuron k by the previous layer. Minimizing this entropy loss promotes sparse neurons, thus forcing a binary encoding. To encourage clustering behavior, SAUCIE also introduced an intracluster loss as

$$L_C = \sum_{i,j: h_i^c = h_j^c} \|\hat{x}_i - \hat{x}_j\|^2 \quad (5.26)$$

which computes the distance L_C between the expressions of a pair of cells (\hat{x}_i, \hat{x}_j) that have the same cluster ID ($h_i^c = h_j^c$).

2. Batch correction: To correct the batch effect, an MMD loss is introduced to measure the differences in terms of the distribution between batches in the latent space

$$L_B = \sum_{l=1, l \neq ref}^B MMD(\mathbf{z}_{ref}, \mathbf{z}_l) \quad (5.27)$$

where B is the total number of batches and z_{ref} is the latent variable of an arbitrarily chosen reference batch.

3. Imputation and visualization: The output of the decoder is taken by SAUCIE as an imputed version of the input gene expression. To visualize the data without performing an additional dimension reduction directly, the dimension of the latent variable z_n is forced to 2.

Training the model includes two sequential runs. In the first run, an autoencoder is trained to minimize the loss $L_0 + \lambda_B L_B$ with L_0 being the MSE reconstruction loss defined in ((?)) so that a batch-corrected, imputed input \tilde{x} can be obtained at the output of the decoder. In the second run, the bottleneck layer of the encoder from the first run is replaced by a 2-D latent code for visualization and a digital encoding layer is also introduced. This model takes the cleaned \tilde{x} as the input and is trained for clustering by minimizing the loss $L_0 + \lambda_D L_D + \lambda_C L_C$. After the model is trained, \tilde{x} is the imputed, batch-corrected gene expression. The 2-D latent code is used for visualization and the binary encoder encodes the cluster ID.

Evaluation metrics. For clustering, the Silhouette index was used. For batch correction, a mixing score similar in spirit to KBET is introduced to assess how well samples from two batches mix. For visualization, the precision and recall of the consistency of the neighboring cells of each cell between the original data space and the SAUCIE embedding space were calculated. For imputation, in addition to visual comprision of the imputed results, R^2 statistics of the imputation results on a synthetic data were reported.

Results. SAUCIE was evaluated for clustering, batch correction, imputation, and visualization on both simulated and real scRNA-seq and scCyToF datasets. The performance was compared to minibatch kmeans, Phenograph (?) and 22 for clustering; MNN (?) and canonical correlation analysis (CCA) (?) for batch correction; PCA, Monocle2 (?), diffusion maps, UMAP (?), tSNE (?) and PHATE (?) for visualization; MAGIC (?), scImpute (?) and nearest neighbors completion (NN completion) for imputation. Results showed that SAUCIE had a better or comparable performance with other approaches. Also, SAUCIE has better scalability and faster runtimes than any of the other models. SAUCIE's results on the scCyToF dengue dataset were further analyzed in greater detail. SAUCIE was able to identify subtypes of the T cell populations and demonstrated distinct cell manifold between acute and healthy subjects.

5.4.4 scScope

scScope (?) is an AE with recurrent steps designed for imputation and batch correction.

Model. scScope has the following model design for batch correction and imputation. 1. Batch correction: A batch correction layer is applied to the input expression as

$$\mathbf{x}_n^c = \text{ReLU}(\mathbf{x}_n - B u_c) \quad (5.28)$$

where ReLU is the ReLU activation function, $B \in \mathbb{R}^{G \times K}$ is the batch correction matrix, $u_c \in \{0, 1\}^{K \times 1}$ is an indicator vector with entry 1 indicates the batch of the input, and K is the total number of batches.

2. Recursive imputation: Instead of using the reconstructed expression \hat{x}_n as the imputed expression like in SAUCIE, scScope adds an imputer to \hat{x}_n to recursively improve the imputation result. The imputer is a single-layer autoencoder, whose decoder performs the imputation as

$$\hat{x}_n = P_I[D_I(\hat{\mathbf{z}}_n)] \quad (5.29)$$

where $\hat{\mathbf{z}}_n$ is the output of the imputer encoder, D_I is the imputer decoder, and P_I is a masking function that set the elements in $\hat{\mathbf{x}}_n$ that correspond to the non-missing values to zero. Then, $\hat{\mathbf{x}}_n$ will be fed back to fill the missing value in the batch corrected input as $x_n^c + \hat{\mathbf{x}}_n$, which will be passed on to the main autoencoder. This recursive imputation can iterate multiple cycles as selected. The loss function modifies the conventional reconstruction loss L_0 as

$$L_0 = \sum_{n=1}^N \sum_{t=1}^T \|P_I^-[x_n^c - \hat{x}_n^t]\|^2 \quad (5.30)$$

where T is the total number of recursion, \hat{x}_n^t is the reconstructed expression at t th recursion, P_I^- is another masking function that forces the loss to compute only the non-missing values in x_n^c .

Evaluation metrics. The performance was tested on both real and simulated datasets. ARI was used to evaluate the ability to match SAUCIE outputs to capture cell subpopulations. For imputation, the reconstruction error on held-out biological data was used. For batch correction, entropy of mixing was applied.

Results. scScope was evaluated for its scalability, clustering, imputation, and batch correction. It was compared with PCA, MAGIC, ZINB-WaVE, SIMLR,

AE, scVI, and DCA. For scalability and training speed, scScope was shown to offer scalability (for >100K cells) with high efficiency (faster than most of the approaches). For clustering results, scScope was shown to outperform most of the algorithms on small simulated datasets but offer similar performance on large simulated datasets. For batch correction, scScope performed comparably with other approaches but with faster runtime. For imputation, scScope produced smaller errors consistently across a different range of expression. scScope was further shown to be able to identify rare cell populations from a large mix of cells.

5.5 Doublet Classification

In scRNA-Seq, doublets are technical artifacts formed when two or more cells are encapsulated into one reaction volume by chance, therefore need to be removed before quantifying gene expression.

5.5.1 Solo

Solo (?) is a semi-supervised deep learning approach that identifies doublets with greater accuracy than other existing methods.

Model. Solo identifies doublets by embedding cells using VAE first and then appends a feed-forward neural network layer to the encoder to form a supervised classifier on the labeled training samples. The method assumes that most cells in an experiment are singlets and one can approximate a view into the doublet population by generating simulated doublets in silico from the observed data. In Solo, doublets are assumed to have less than twice the UMI depth on average but an option to scale the merged counts by a specific factor is also provided. The method operates in three phases: (1) doublet simulation where a number N_d of doublets are generated by taking the sum of randomly chosen observed cells considered putative singlets, and then model was trained to differentiate these in silico doublets from the observed data; (2) cell embedding where Solo embeds cells in a nonlinear latent space via a VAE that follows the scVI architecture; and (3) classifier training where a standard discriminative classifier is added to the end of the scVI encoder and trained using the simulated doublet samples and the observed cells to predict doublet status. Only the added classifier layer was trained with a binary cross-entropy loss, the scVI encoder was fixed during training.

Evaluation metrics. The performance was measured using precision-recall (PR) and receiver operator characteristic (ROC) curves and summarized with average precision (AP) and area under the receiver operator curve (AUROC).

Results. Solo performance was tested on a singlet/doublet population simulated using the Splatter toolkit (?) and several experimental datasets, where Solo ex-

ceeds the performance of previous non-deep learning methods like Scrublet (?) and DoubletFinder (?) by greater margins in the larger and more complex datasets. Solo also outperformed existing computational methods for this task on a variety of cell line and tissue datasets with experimental doublet annotations.

5.6 Automated cell type identification

scRNA-Seq is able to catalog cell types in complex tissues under different conditions. However, the commonly adopted manual cell typing approach based on known markers is time-consuming and less reproducible. We survey deep learning models that take the task of automated cell type identification.

5.6.1 DigitalDLSorter

DigitalDLSorter (?) was proposed to identify and quantify the immune cells infiltrated in tumors captured in bulk RNA-seq, utilizing single-cell RNA-seq data.

Model. DigitalDLSorter is a 4-layer DNN (2 hidden layers of 200 neurons each and an output of 10 cell types). The DigitalDLSorter is trained with two single-cell datasets: breast cancers (?) and colorectal cancers (?). For each cell, it is determined to be tumor cell or non-tumor cell using RNA-seq based CNV method (?), followed with using xCell algorithm (?) to determine immune cell types for non-tumor cells. Different pseudo bulk (from 100 cells) RNA-seq datasets were prepared with known mixture proportion to train the DNN. The output of DigitalDLSorter is the predicted proportions of cell types in the input bulk sample.

Evaluation. Pearson Correlation between the predicted and real cell type proportions was used to assess the performance.

Result. DigitalDLSorter was first tested on simulated bulk RNA-seq samples. DigitalDLSorter achieved excellent agreement (linear correlation of 0.99 for colorectal cancer, and good agreement in quadratic relationship for breast cancer) at predicting cell types proportion. The proportion of immune and nonimmune cell subtypes of test bulk TCGA samples was predicted by DigitalDLSorter and the results showed very good correlation to other deconvolution tools including TIMER (?), ESTIMATE (?), EPIC (?) and MCPCounter (?). Using DigitalDLSorter predicted CD8+ (good prognosis for overall and disease-free survival) and Monocytes-Macrophages (MM, indicator for protumoral activity) proportions, it is found that patients with higher CD8+/MM ratio had better survival for both cancer types than those with lower CD8+/MM ratio. Both EPIC and MCPCounter yielded non-significant survival association using their cell proportion estimate.

5.6.2 scCapsNet

scCapsNet (?) is an interpretable capsule network designed for cell type prediction. The paper showed that the trained network could be interpreted to inform marker genes and regulatory modules of cell types.

Model. Two-layer architecture of scCapsNet takes log-transformed, normalized expressions as input to form a feature extraction network (consists of L parallel single-layer neural networks) followed by a capsule network for cell-type classification (type capsules). For each of L parallel feature extraction layer, it generates a primary capsule $u_l \in \mathbb{R}^{d_p}$ as

$$u_l = ReLU(W_{P,l}x_n) \forall l = 1, \dots, L \quad (5.31)$$

where $W_{P,l} \in \mathbb{R}^{d_p \times G}$ is the weight matrix. Then, the primary capsules are fed into the capsule network to compute K type capsules $v_k \in \mathbb{R}^{d_t}$, one for each cell type, as

$$v_k = squash(\sum_l^L c_{kl} W_{kl} u_l) \forall k = 1, \dots, K \quad (5.32)$$

where squash is the squashing function (?) to normalize the magnitude of its input vector to be less than one, W_{kl} is another trainable weight matrix, and $c_{kl} \forall l = 1, \dots, L$ are the coupling coefficients that represent the probability distribution of each primary capsule's impact on the prediction of cell type k . c_{kl} is not trained but computed through the dynamic routing process proposed in the original capsule networks (@ ?). The magnitude of each type of capsule v_k represents the probability of a single cell x_n belonging to cell type k , which will be used for cell type classification.

The training of the network minimizes the cross-entropy loss by the back-propagation algorithm. Once trained, the interpretation of marker genes and regulatory modules can be achieved by determining first the important primary capsules for each cell type and then the most significant genes for each important primary capsule (identified based on c_{kl} directly). To determine the genes that are important for an important primary capsule l , genes are ranked base on the scores of the first principal component computed from the columns of $W_{P,l}$ and then the markers are obtained by a greedy search along with the ranked list for the best classification performance.

Evaluation metrics: Accuracy of the predicted cell types was evaluated.

Results. scCapsNet's performance was evaluated on human PBMCs (?) and mouse retinal bipolar cells (?) datasets and shown to have comparable accuracies (99% and 97% respectively) with DNN and other popular ML algorithms (SVM, random forest, LDA and nearest neighbor). However, the interpretability of scCapsNet was demonstrated extensively. First, examining the coupling

coefficients for each cell type showed that only a few primary capsules have high values and thus is effective. Subsequently, a set of core genes were identified for each effective capsules using the greedy search on the PC-score ranked gene list. GO enrichment analysis showed that these core genes were enriched in cell type related biological functions. Mapping the expression data into space spanned by PCs of the columns of $W_{P,l}$ corresponding to all core genes uncovered regulatory modules that would be missed by the T-SNE of gene expressions, which demonstrated the effectiveness of the embeddings learned by scCapsNet in capturing the functionally important features.

5.6.3 netAE: network-enhanced autoencoder

netAE (?) is a VAE-based semi-supervised cell type prediction model that deals with scenarios of having a small number of labeled cells.

Model. netAE works with UMI counts and assumes a ZINB distribution for x_{gn} as in Eq. ((?)) in scVI. However, netAE adopts the general VAE loss as in eq. (6) with two function-specific loss as

$$L(\Theta) = -\mathcal{L}(\Theta) + \lambda_1 \sum_{n \in S} Q(\mathbf{z}_n) + \lambda_2 \sum_{n \in S_L} \log f(y_n | \mathbf{z}_n) \quad (5.33)$$

where S is a set of indices for all cells and S_L is a subset of S for only cells with cell type labels, Q is modified Newman and Girvan modularity (?) that quantifies cluster strength using z_n , f is the softmax function, and y_n is the cell type label. The second loss in ((?)) functions as a clustering constraint and the last term is the cross-entropy loss that constrains the cell type classification.

Evaluation metrics. Accuracy of cell type classification was used to assess the performance.

Results: netAE was compared with popular dimension reduction methods including scVI, ZIFA, PCA and AE as well as a semi-supervised method scANVI (?). For different dimension reduction methods, cell type classification from latent features of cells was carried out using KNN and logistic regression. The effect of different labeled samples sizes on classification performance was also investigated, where the sample size varied from as few as 10 cells to 70% of all cells. Among 3 test datasets (mouse brain cortex, human embryo development, and mouse hematopoietic stem and progenitor cells), netAE outperformed most of the baseline methods. Latent features were visualized using t-SNE and cell clusters by netAE were tighter than those by other embedding spaces. There was also consistency of better cell-type classification with improved cell type clustering by netAE. This suggested that the latent spaces learned with added modularity constraint in the loss helped identify clusters of similar cells. Ablation study by removing each of the three loss terms in Eq (45) showed a drop of cell-type classification accuracy, suggesting all three were necessary for the optimal performance.

5.6.4 scDGN - supervised adversarial alignment of single-cell RNA-seq data

scDGN (?), or Single Cell Domain Generalization Network, is an domain adversarial network that aims to accurately assign cell types of single cells while performing batch removal (domain adaptation) at the same time. It benefits from the superior ability of domain adversarial learning to learn representations that are invariant to technical confounders.

Model. scDGN takes the log-transformed, normalized expression as the input and has three main modules: i) an encoder ($E_\phi(x_n)$) for dimension reduction of scRNA-Seq data, ii) cell-type classifier, $C_{\phi_C}(E_\phi(x_n))$ with parameters ϕ_C , and iii) domain (batch) discriminator, $D_{\phi_D}(E_\phi(x_n))$. The model has a Siamese design and the training takes a pair of cells (x_1, x_2) , each from the same or different batches. The encoder network contains two hidden layers with 1146 and 100 neurons. C_{ϕ_C} classifies the cell type and D_{ϕ_D} predicts whether x_1 and x_2 are from the same batch or not. The overall loss is denoted by

$$L(\phi, \phi_C, \phi_D) = L_C(C_{\phi_C}(E_\phi(x_1))) - \lambda L_D(D_{\phi_D}(E_\phi(x_1)), D_{\phi_D}(E_\phi(x_2))) \quad (5.34)$$

where L_C is the cross entropy loss, L_D is a contrastive loss as described in (?). Notice that ((?)) has an adversarial formulation and minimizing this loss maximizes the misclassification of cells from different batches, thus making them indistinguishable. Similar to GAN training, scDGN is trained to iteratively solve: $\hat{\phi}_D = \operatorname{argmin}_{\phi_D} L(\hat{\phi}, \hat{\phi}_C, \phi_D)$ and $(\hat{\phi}, \hat{\phi}_C) = \operatorname{argmin}_{\phi, \phi_C} L(\phi, \phi_C, \hat{\phi}_D)$.

Evaluation metrics. Prediction accuracy was used to evaluate the performance.

Results. scDGN was tested for classifying cell types and aligning batches ranging in size from 10 to 39 cell types and from 4 to 155 batches. The performance was compared to a series of deep learning and traditional machine learning methods, including Lin et al. DNN (?), CaSTLe (?), MNN (?), scVI (?) [13], and Seurat (?). scDGN outperformed all other methods in the classification accuracy on a subset of scQuery datasets (0.29), PBMC (0.87), and 4 of the six Seurat pancreatic datasets (0.86-0.95). PCA visualization of the learned data representations demonstrated that scDGN overcame the batch differences and clearly separated cell clusters based on cell types, while other methods were vulnerable to batch effects. In summary, scDGN is a supervised adversarial alignment method to eliminate the batch effect of scRNA-Seq data and create cleaner representations of cell types.

5.7 Biological function prediction

Predicting biological function and/reponse to treatment at single cell level or cell types is critical to understand cell system functioning and potent response

to stimulation. Utilize DL models that captures gene-gene relationship and their property at latent space, several models we reviewed below provide some exciting approach to achieve complex biological functions and outcomes.

5.7.1 CNNC: convolutional neural network for coexpression

CNNC (?) is proposed to infer causal interactions between genes from scRNA-seq data.

Model. CNNC is a Convolutional Neural Network (CNN), the most popular DL model. CNNC takes expression levels of two genes from many cells and transforms them into a 32 x 32 image-like normalized empirical probability function (NEPDF), which measures the probabilities of observing different coexpression levels between the two genes. CNNC includes 6 convolutional layers, 3 max-pooling layers, 1 flatten layer, and one output layer. All convolution layers have 32 kernels of size 3x3. Depending on the application, the output layer can be designed to predict the state of interaction (Yes/No) between the genes or the causal interaction between the input genes (no interaction, gene A regulates gene B, or gene B regulates gene A).

Evaluation matrices. Prediction AUROC, AUPRC, and accuracy were assessed.

Result. CNNC was trained to predict transcription factor (TF)-Gene interactions using the mESC data from scQuery (?), where the ground truth interactions were obtained from the ChIP-seq dataset from the GTRD database (?). The performance was compared with DNN, count statistics (?), and mutual information-based approach (?). CNNC was shown to have more than 20% higher AUPRC than other methods and reported almost no false-negative for the top 5% predictions. CNNC was also trained to predict the pathway regulator-target gene pairs. The positive regulator-gene pairs were obtained from KEGG (?), Reactome (?), and negative samples were genes pairs that appeared in pathways but not interacted. CNNC was shown to have better performance of predicting regulator-gene pairs on both KEGG and Reactome pathways than other methods including Pearson correlation, count statistics, GENIE3 (?), Mutual information, Bayesian directed network (BDN), and DREMI (?). CNNC was also applied for causality prediction between two genes, that is if two genes regulate each other and if they do, which gene is the regulator. The ground truth causal relationships were also obtained from the KEGG and Reactome datasets. Again, CNNC reported better performance than BDN, the common method developed to learn casual relationship from gene expression data. CNNC was finally trained to assign 3 essential cell functions (cell cycle, circadian rhythm, and immune system) to genes. This is achieved by training CNNC to predict pair of genes from same function (e.g. Cell Cycle defined by mSigDB from gene set enrichment analysis (GSEA (?)) as 1 and all other pairs as 0. The performance was compared with “guilt by association” and DNN,

and CNNC were shown to have more than 4% higher AUROC and reported all positives for the top 10% predictions.

5.7.2 scGen, a generative model to predict perturbation response of single cells across cell types

scGen (?) is designed to learn cell response to certain perturbation (drug treatment, gene knockout, etc) from single cell expression data and predict the response to the same perturbation for a new sample or a new cell type. The novelty of scGen is that it learns the response in the latent space instead of the expression data space.

Model. ScGen follows the general VAE for scRNA-seq data but uses the “latent space arithmetics” to learn perturbations’ response. Given scRNAseq samples of perturbed (denoted as p) and unperturbed cells (denoted as unp), a VAE model is trained. Then, the latent space representation z_p and z_{unp} are obtained for the perturbed and unperturbed cells. Following the notion that VAE could map nonlinear operations (e.g., perturbation) in the data space to linear operations in the latent space, ScGen estimates the response in the latent space as $\delta = \bar{z}_p - \bar{z}_{unp}$, where \bar{z} is the average representation of samples from the same cell type or different cell types. Then, given the latent representation z'_{unp} of an unperturbed cell for a new sample from the same or different cell type, the latent representation of the corresponding perturbed cell can be predicted as $z'_p = z'_{unp} + \delta$. The expression of the perturbed cell can also be estimated by feeding z'_p into the VAE decoder. The scGen can also be expanded to samples and treatment across two species (using orthologues between species). When scGen is trained for species 1, or s_1 , with both perturbed and unperturbed cells but species 2, s_2 , with only unperturbed cells, then the latent code for the perturbed cells from s_2 can be predicted as $z_{s_2,p} = \frac{1}{2}(z_{s_1,p} + z_{s_2,unp} + \delta_s + \delta_p)$ where $\delta_p = z_{s_1,unp} - z_{s_1,p}$ captures the response of perturbation and $\delta_s = z_{s_1} - z_{s_2}$ represents the difference between species.

Evaluation metrics. Correlation and UMAP visualization were used to assess the performance.

Result. scGen was applied to predict perturbation of out-of-samples response in human PBMCs data, and scGen showed a higher average correlation ($R^2=0.948$) between predicted and real data for six cell types. Comparing with other methods including CVAE (?), style transfer GAN [124:reference not found], linear approaches based on vector arithmetics (VA) (?) and PCA+VA, scGen predicted full distribution of ISG15 gene (strongest regulated gene by IFN- β) response to IFN- β (?), while others might predicted mean (CAVE and style transfer GAN) but failed to produce the full distribution. scGen was also tested on predicting the intestinal epithelial cells’ response to infection (?). For early transit-amplifying cells, scGen showed good prediction ($R^2=0.98$) for both H. poly and Salmonella infection. Finally, scGen was evaluated for perturbation

across species using scRNA-seq data set by Hagai et al (?), which comprised of bone marrow-derived mononuclear phagocytes from mice, rats, rabbits, and pigs perturbed with lipopolysaccharide (LPS). scGen’s predictions of LPS perturbation responses were shown to be highly correlated ($R^2 = 0.91$) with the real responses.

Chapter 6

Conclusion and Discussions

Single cell RNA-Seq technologies are tools that help exploring the cell-type composition of a particular sample and how it is dysregulated during development and disease.

Chapter 7

Tables

App	Algorithm	Models	Evaluation	Environment	Codes	Refs
Imputation						
	DCA	AE	DREMI	Keras, Tensorflow, scanpy	https://github.com/theislabs/dca	[17]
	SAVER-X	AE+TL	t-SNE, ARI	R/sctransfer	https://github.com/iingshuw/SAVERX	[51]
	DeepImpute	DNN	MSE, Pearson's correlation	Keras/Tensorflow	https://github.com/lanaqarmire/DeepImpute	[19]
	LATE	AE	MSE	Tensorflow	https://github.com/audreyvfu/LATE	[52]
	scGAMI	AE	NMI, ARI, HS and CS	Tensorflow	https://github.com/QUST-AIBDRC/scGAMI	[53]
	scIGANs	GAN	ARI, ACC, AUC, and F-score	PyTorch	https://github.com/xuyungang/scIGANs	[18]
Batch correction						
	BERMUDA	AE+TL	kBET, entropy of Mixing, SI	PyTorch	https://github.com/xwang/BERMUDA	[57]
	DESC	AE	ARI, KL	Tensorflow	https://github.com/eleozzz/DESC	[61]
	iMAP	AE+GAN	kBET, LISI	PyTorch	https://github.com/Svyord/iMAP	[64]
Clustering, latent representation, dimension reduction, and data augmentation						
	Dhaka	VAE	ARI, Spearman Correlation	Keras/Tensorflow	https://github.com/MicrosoftGenomics/Dhaka	[66]
	scvis	VAE	KNN preservation, log-likelihood	Tensorflow	https://bitbucket.org/ericy00/scvis-dev/src/master/	[69]
	scVAE	VAE	ARI	Tensorflow	https://github.com/scvae/scvae	[70]
	VASC	VAE	NMI, ARI, HS, and CS	H5py, keras	https://github.com/wanq-research/VASC	[71]
	scDeepCluster	AE	ARI, NMI, clustering accuracy	Keras, Scanpy	https://github.com/tigum/scDeepCluster	[73]
	cscGAN	GAN	t-SNE, marker genes, MMD, AUC	Scipy, Tensorflow	https://github.com/imsb-uke/scGAN	[76]
Multi-functional models (IM: imputation, BC: batch correction, CL: clustering)						
	scVI	VAE	IM: L ₁ distance; CL: ARI, NMI, SI; BC: Entropy of Mixing	PyTorch, Anndata	https://github.com/YosefLab/scvi-tools	[16]
	LDVAE	VAE	Reconstruction errors	Part of scVI	https://github.com/YosefLab/scvi-tools	[80]
	SAUCIE	AE	IM: R ² statistics; CL: SI; BC: modified kBET; Visualization: Precision/Recall	Tensorflow	https://github.com/KrishnaswamyLab/SAUCIE/	[14]
	scScope	AE	IM: Reconstruction errors; BC: Entropy of mixing; CL: ARI	Tensorflow, Scikit-learn	https://github.com/AltshulerWu-Lab/scScope	[86]
Cell type Identification						
	DigitalDLSorter	DNN	Pearson correlation	R/Python/ Keras	https://github.com/carto/digitalDLSorter	[87]
	scCapsNet	CapsNet	Cell-type Prediction accuracy	Keras, Tensorflow	https://github.com/wanqif19/scCaps	[94]
	netAE	VAE	Cell-type Prediction accuracy, t-SNE for visualization	pyTorch	https://github.com/LeoZDong/netAE	[98]
	scDGN	DANN	Prediction accuracy	pyTorch	https://github.com/SongweiGe/scDGN	[101]
Function analysis						
	CNNC	CNN	AUROC, AUPRC, and accuracy	Keras, Tensorflow	https://github.com/xiaoyeye/CNNC	[104]
	scGen	VAE	Correlation, visualization	Tensorflow	https://github.com/theislab/scgen	[113]

DL Model keywords: AE: autoencoder, AE+TL: autoencoder with transfer learning, AE: variational autoencoder, GAN: Generative adversarial network, CNN: convolutional neural network, DNN: deep neural network, DANN: domain adversarial neural network, CapsNet: capsule neural network

Figure 7.1: Deep Learning algorithms reviewed in the paper

Title	Algorithm	# Cells	Simulation methods	Reference
Splatter	DCA, DeepImpute, PERMUDA, scDeepCluster, scVI, scScope, solo	~2000	<u>Splatter/R</u>	[78]
CIDR	scIGAN	50	<u>CIDR simulation</u>	[54]
NB+dropout	Dhaka	500	<u>Hierachical model of NB/Gamma + random dropout</u>	
Bulk RNA-seq	SAUCIE	1076	<u>1076 CCLE bulk RNAseq + dropout conditional on expression level</u>	
SIMLR	scScope	1 million	<u>SIMLR, high-dimensional data generated from latent vector</u>	[43]

Figure 7.2: Simulated single-cell data/algorithms

Title	Algorithm	Cell origin	# Cells	Data Sources	Reference
68k PBMCs	DCA SAVER-X LATE, scVAE, scDeepCluster, scCapsNet, scDGN	Blood	68,579	10X Single Cell Gene Expression Datasets	
Human pluripotent	DCA	hESCs	1,876	GSE102176	[122]
CITE-seq	SAVER-X	Cord blood mononuclear cells	8,005	GSE100866	[123]
Midbrain and Dopaminergic Neuron Development	SAVER-X	Brain/ embryo ventral midbrain cells	1,977	GSE76381	[124]
HCA	SAVER-X	Immune cell, Human Cell Atlas	500,000	HCA data portal	
Breast tumor	SAVER-X	Immune cell in tumor micro-environment	45,000	GSE114725	[125]
293T cells	DeepImpute, IMAP	Embryonic kidney	13,480	10X Single Cell Gene Expression Datasets	
Jurkat	DeepImpute, IMAP	Blood/ lymphocyte	3,200	10X Single Cell Gene Expression Datasets	
ESC, Time-course	scGAN	ESC	350, 758	GSE75748	[126]
Baron-Hum-1	scGMAI, VASC	Pancreatic islets	1,937	GSM2230757	[127]
Baron-Hum-2	scGMAI, VASC	Pancreatic islets	1,724	GSM2230758	[127]
Camp	scGMAI, VASC	Liver cells	303	GSE96981	[128]
CEL-seq2	PERMUDA, DESC	Pancreas/Islets of Langerhans		GSE85241	[129]
Darmanis	scGMAI, scGAN, VASC	Brain/cortex	466	GSE67835	[130]
Tirosh-brain	Dhaka, scvis	Oligodendroglioma	>4800	GSE70630	[131]
Patel	Dhaka	Primary glioblastoma cells	875	GSE57872	[132]
Li	scGMAI, VASC	Blood	561	GSE146974	[61]
Tirosh-skin	scvis	melanoma	4645	GSE72056	[67]
xenograft 3, and 4	Dhaka	Breast tumor	~250	EGAS00001002170	[133]
Petropoulos	VASC/netAE	Human embryos	1,529	E-MTAB-3929	
Pollen	scGMAI, VASC		348	SRP041736	[134]
Xin	scGMAI, VASC	Pancreatic cells (α -, β -, δ -)	1,600	GSE81608	[135]
Yan	scGMAI, VASC	embryonic stem cells	124	GSE36552	[136]
PBMC3k	VASC, scVI	Blood	2,700	SRP073767	[96]
CytoF, Dengue	SAUCIE	Dengue infection	11 M, ~42 antibodies	Cytobank, 82023	[14]
CytoF, ccRCC	SAUCIE	Immunue profile of 73 ccRCC patients	3.5 M, ~40 antibodies	Cytobank: 875	[137]
CytoF, breast	SAUCIE	3 patients		Flow Repository: FR-FCM-ZYJP	[125]
Chung, BC	DigitalDLSorter	Breast tumor	515	GSE75688	[88]
LI, CRC	DigitalDLSorter	Colorectal cancer	2,591	GSE81861	[89]
Pancreatic datasets	scDGN	Pancreas	14693	SeuratData	
Kang, PBMC	scGen	PBMC stimulated by INF- β	~15,000	GSE96583	[116]

Figure 7.3: Human single-cell data sources used by different DL algorithms

Title	Algorithm	Cell origin	# Cells	Data Sources	Reference
Brain cells from E18 mice	DCA, SAUCIE	Brain Cortex	1,306,127	10x: Single Cell Gene Expression Datasets	
Midbrain and Dopaminergic Neuron Development	SAVER-X	Ventral Midbrain	1907	GSE76381	[81]
Mouse cell atlas	SAVER-X		405,796	GSE108097	[144]
neuron9k	DeepImpute	Cortex	9128	10x: Single Cell Gene Expression Datasets	
Mouse Visual Cortex	DeepImpute	Brain cortex	114601	GSE102827	[145]
murine epidermis	DeepImpute	Epidermis	1422	GSE67602	[146]
myeloid progenitors	LATE DESC, SAUCIE	Bone marrow	2,730	GSE72857	[147]
Cell-cycle	sciGAN	mESC	288	E-MTAB-2805	[148]
A single-cell survey		Intestine	7721	GSE92332	[126]
Tabula Muris	iMAP	Mouse cells	>100K		
Baron-Mou-1	VASC	Pancreas	822	GSM2230761	[132]
Biase	scGMAI, VASC	Embryos/SMA RTer	56	GSE57249	[149]
Biase	scGMAI, VASC	Embryos/Fluidigm	90	GSE59892	[149]
Deng	scGMAI, VASC	Liver	317	GSE45719	[150]
Klein	VASC scDeepCluster sciGAN	Stem Cells	2,717	GSE65525	[151]
Goolam	VASC	Mouse Embryo	124	E-METAB-3321	[152]
Kolodziejczyk	VASC	mESC	704	E-MTAB-2600	[153]
Usoskin	VASC	Lumbar	864	GSE59739	[154]
Zeisel	VASC, scVI, SAUCIE, netAE	Cortex, hippocampus	3,005	GSE60361	[155]
Bladder cells	scDeepCluster	Bladder	12,884	GSE129845	[156]
HEMATO	scVI	Blood cell	>10,000	GSE89754	[157]
retinal bipolar cells	scVI, scCapsNet SAUCIE	retinal	~25,000	GSE81905	[106]
Embryo at 9 time points	LDAVE	embryos from E6.5 to E8.5	116,312	GSE87038	[158]
Embryo at 9 time points	LDAVE	embryos from E9.5 to E13.5	~2 millions	GSE119945	[159]
CyTOF,	SAUCIE	Mouse thymus	200K, ~38 antibodies	Cytobank: 52942	[160]
Solo	solo	Mouse kidneys	~8,000	GSE140262	[93]
Nestorowa	netAE	hematopoietic stem and progenitor cells	1,920	GSE81682	[161]
small intestinal epithelium	scGen	Infected with Salmonella and worm H. polygyrus	1,957	GSE92332	[126]

Figure 7.4: Mouse single-cell data sources used by different DL algorithms

Title	Algorithm	Species	Tissue	# Cells	SRA/GEO	Ref
Worm neuron cells¹	scDeepCluster	C. elegans	Neuron	4,186	GSE98561	[1]
Cross species, stimulation with LPS and dsRNA	scGen	Mouse, rat, rabbit, and pig	bone marrow-derived phagocyte	5,000 to 10,000 /species	13 accessions in ArrayExpress	[1]

1. Processed data is available at <https://github.com/ttgump/scDeepCluster/tree/master/scRNA-seq%20data>

Figure 7.5: Single-cell data derived from other species

Title	Sources	Notes
10X Single-cell gene expression dataset	https://support.10xgenomics.com/single-cell-gene-expression/datasets	Contains large collection of scRNA-seq dataset generated using 10X system
<u>Tabula Muris</u>	https://tabula-muris.ds.czbiohub.org/	Compendium of scRNA-seq data from mouse
HCA	https://data.humancellatlas.org/	Human single-cell atlas
MCA	https://figshare.com/s/865e694ad06d5857db4b , or GSE108097	Mouse single-cell atlas
scQuery	https://scquery.cs.cmu.edu/	A web server cell type matching and key gene visualization. It is a good source for scRNA-seq data (processed with common pipeline)
SeuratData	https://github.com/satijalab/seurat-data	List of datasets, including PBMC and human pancreatic islet cells
cytoBank	https://cytobank.org/	Community of big data cytometry

Figure 7.6: Large single-cell data source used by various algorithms


Evaluation Method	Equations	Explanation
Pseudobulk RNA-seq		Average of normalized (log2-transformed) scRNA-seq counts across cells is calculated and then correlation coefficient between the pseudobulk and the actual bulk RNA-seq profile of the same cell type is evaluated.
Mean squared error (MSE)	$MSE = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2$	MSE assesses the quality of a predictor, or an estimator, from a collection of observed data x , with \hat{x} being the predicted values.
Pearson correlation	$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$	where $cov()$ is the covariance, σ_X and σ_Y are the standard deviation of X and Y , respectively.
Spearman correlation	$\rho_s = \rho_{r_X, r_Y} = \frac{cov(r_X, r_Y)}{\sigma_{r_X} \sigma_{r_Y}}$	The Spearman correlation coefficient is defined as the Pearson correlation coefficient between the rank variables, where r_X is the rank of X .
Entropy of accuracy, H_{acc} [20]	$H_{acc} = -\frac{1}{M} \sum_{i=1}^M \sum_{j=1}^{N_i} p_i(x_j) \log(p_i(x_j))$	Measures the diversity of the ground-truth labels within each predicted cluster group. $p_i(x_j)$ (or $q_i(x_i)$) are the proportions of cells in the j^{th} ground-truth cluster (or predicted cluster) relative to the total number of cells in the i^{th} predicted cluster (or ground-truth clusters), respectively.
Entropy of purity, H_{pur} [20]	$H_{pur} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^{M_i} q_i(x_j) \log(q_i(x_j))$	Measures the diversity of the predicted cluster labels within each ground-truth group
Entropy of mixing [31]	$E = \sum_{i=1}^c p_i \log(p_i)$	This metric evaluates the mixing of cells from different batches in the neighborhood of each cell. C is the number of batches, and p_i is the proportion of cells from batch i among N nearest cells.
Mutual Information (MI) [157]	$MI(U, V) = \sum_{i=1}^{ U } \sum_{j=1}^{ V } P_{UV}(i, j) \log\left(\frac{P_{UV}(i, j)}{P_U(i)P_V(j)}\right)$	where $P_U(i) = \frac{ U_i }{N}$ and $P_V(j) = \frac{ V_j }{N}$. Also, define the joint distribution probability is $P_{UV}(i, j) = \frac{ U_i V_j }{N}$. The MI is a measure of mutual dependency between two cluster assignments U and V .
Normalized Mutual Information (NMI) [158]	$NMI(U, V) = \frac{2 \times MI(U, V)}{H(U) + H(V)}$	where $H(U) = \sum P_U(i) \log(P_U(i))$, $H(V) = \sum P_V(j) \log(P_V(j))$. The NMI is a normalization of the MI score between 0 and 1.
Kullback-Leibler (KL) divergence [159]	$D_{KL}(P Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$	where discrete probability distributions P and Q are defined on the same probability space \mathcal{X} . This relative entropy is the measure for directed divergence between two distributions.
Jaccard Index	$J(U, V) = \frac{ U \cap V }{ U \cup V }$	$0 \leq J(U, V) \leq 1$. $J = 1$ if clusters U and V are the same. If U are V are empty, J is defined as 1.
Fowlkes-Mallows Index for two clustering algorithms (FM)	$FM = \sqrt{\frac{TP}{TP + FP} \times \frac{TP}{TP + FN}}$	TP as the number of pairs of points that are present in the same cluster in both U and V ; FP as the number of pairs of points that are present in the same cluster in U but not in V ; FN as the number of pairs of points that are present in the same cluster in V but not in U ; and TN as the number of pairs of points that are in different clusters in both U and V .
Rand index (RI)	$RI = (a + b) / \binom{N}{2}$	Measure of constancy between two clustering outcomes, where a (or b) is the count of number of pairs of cells in one cluster (or different clusters) from one clustering algorithm but also fall in the same cluster (or different clusters) from the other clustering algorithm.
Adjusted Rand index (ARI) [160]	$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$	ARI is a corrected-for-chance version of RI , where $E[RI]$ is the expected Rand Index.
Silhouette index	$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$	where $a(i)$ is the average dissimilarity of i^{th} cell to all other cells in the same cluster, and $b(i)$ is the average dissimilarity of i^{th} cell to all cells in the closest cluster. The range of $s(i)$ is $[-1, 1]$, with 1 to be well-clustered and -1 to be completely misclassified.
Maximum Mean Discrepancy (MMD) [58]	$MMD(F, p, q) = \sup_{\phi \in \mathcal{F}} \ \mu_p - \mu_q\ _{\mathcal{F}}$	MMD is a non-parametric distance between distributions based on the reproducing kernel Hilbert space, or, a distance-based measure between two distribution p and q based on the mean embeddings μ_p and μ_q in a reproducing kernel Hilbert space \mathcal{F} .
k-Nearest neighbor batch-effect test (kBET) [161]	$a_n^k = \sum_{i=1}^L \frac{(N_{n_i}^k - k \cdot f_i)^2}{k \cdot f_i} - X_{L-1}^2$	Given a dataset of N cells from L batches with N_i denoting the number of cells in batch i , $N_{n_i}^k$ is the number of cells from batch i in the k -nearest neighbors of cell n , f_i is the global fraction of cells in batch i , or $f_i = \frac{N_i}{N}$, and X_{L-1}^2 denotes the χ^2 distribution with $L - 1$ degrees of freedom. It uses a χ^2 -based test for random neighborhoods of fixed size to determine the significance ("well mixed").
Local Inverse Simpson's Index (LISI) [33]	$\frac{1}{\lambda(n)} = \frac{1}{\sum_{l=1}^k p(l)^2}$	This is the inverse Simpson's Index in the k -nearest neighbors of cell n for all batches, where $p(l)$ denotes the proportion of batch l in the k -nearest neighbors. The score reports the effective number of batches in the k -nearest neighbors of cell n .
Homogeneity	$HS = 1 - \frac{H(P(U V))}{H(P(U))}$	where $H()$ is the entropy, and U is the ground-truth assignment and V is the predicted assignment. The HS range from 0 to 1, where 1 indicates perfectly homogeneous labelling.
Completeness	$CS = 1 - \frac{H(P(V U))}{H(P(V))}$	Its values range from 0 to 1, where 1 indicates all member from a ground-truth label are assigned to a single cluster.
V-Measure [162]	$V_{\beta} = \frac{(1 + \beta)HS \times CS}{\beta HS + CS}$	where β indicates the weight of HS . V-Measure is symmetric, i.e. switching the true and predicted cluster labels does not change V-Measure.
Precision, recall	$Precision = \frac{TP}{TP + FP}, recall = \frac{TP}{TP + FN}$	TP: true positive, FP: false positive, FN, false negative.
Accuracy	$Accuracy = \frac{TP + TN}{N}$	N: all samples tested, TN: true negative
F1-score	$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$	A harmonic mean of precision and recall. It can be extended to F_{β} where β is a weight between precision and recall (similar to V-measure).
AUC, AUROC		Area Under Curve (grey area). Receiver operating characteristic (ROC) curve (red line). The similar measure can be performed on Precision-Recall curve (PRC), or AUROC. Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model (mostly for imbalanced dataset).

Figure 7.7: Evaluation metrics used in surveyed DL algorithms