Computational methods for "atlas-scale" analysis of scRNA-seq data

Amir Alavi

CMU-CB-21-102

June 2021

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Ziv Bar-Joseph, Chair
Jian Ma
Maria Chikina
Guo-Cheng Yuan

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Research reported in this thesis was supported by the National Institute of General Medical Sciences of the National Institutes of Health under award number R01GM122096; the National Heart, Lung, and Blood Institute of the National Institutes of Health under award numbers U01HL145567 and R01HL128172; and the Office of the Director of the National Institutes of Health under award number OT2OD026682.

The department specifically disclaims responsibility for any analyses, interpretations, or conclusions. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, donor or the U.S. Government.

Keywords: scRNA-seq, Classification, Representation learning, Data integration, Batch-effect correction

For my family, Mom, Dad, and Ayda, who serve as the ultimate role-models in my life and always encourage me to pursue my dreams

Abstract

Single-cell RNA-sequencing (scRNA-seq) has allowed a higher resolution view into the transcriptional landscape of cells. The large amount of data collected as part of these experiments has brought with it new opportunities and challenges for computational analysis methods. Addressing these issues is a critical step for both, studies using scRNA-Seq to model specific biological processes and systems and recent large scale efforts focused on cellular atlases of human tissues.

One of the first questions that researchers face when analyzing such data is the identification of the cell types in the heterogeneous populations of cells. Supervised solutions for this question require the development of novel methods that can extract a rich set of feature representations and that can account for technical effects across datasets. Once cells are assigned to different types, several additional questions can be addressed. Of particular interest, especially for the large scale atlas efforts, is the issue of comparing cell types and tissues across a large set of samples to identify unique markers and marker combinations.

In this thesis, we present a set of computational methods that address each of these related issues. We first present a new computational approach (scQuery) for comparative analysis of scRNA-seq datasets that utilizes large publicly available scRNA-seq datasets of many cell types, instead of relying on marker gene information. We show that the supervised neural embedding models at the heart of this method can learn rich, compact embeddings that enable efficient comparisons between cells and can serve as an effective tool in exploratory scRNA-seq analysis. We next address the problem of batch effect correction. We propose two approaches: a supervised approach called scDGN as well as an unsupervised approach called SCIPR. In both cases, we show that our methods can accurately align cell type populations from different batches, and that our models utilize biologically relevant genes to apply their transformations. Finally, we extend the supervised classification-based approaches to identify marker genes for cell types across multiple tissues in recently collected HuBMAP consortium scRNA-seq data. Taken together, the methods we have developed in this thesis make significant strides for atlas scale analysis of the single-cell gene expression landscape.

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Ziv Bar-Joseph. Your incredible support over these five years has been unwavering. It has been a privilege to work with someone whose breadth of knowledge covers so much of computational biology and machine learning. Your vision and encouragement allowed me to work on some of the most interesting problems I could have hoped for. While world-wide pandemics and geographic distance were challenging variables during my PhD, your patience and commitment to making time for me was a treasured constant.

I would like to thank my thesis committee members, Maria Chikina, Jian Ma, and Guo-Cheng Yuan, for being a part of my committee and providing valuable feedback on the work in my thesis.

I would also like to acknowledge my incredible coauthors, including Matthew Ruffalo, Aiyappa Parvangada, Zhilin Huang, Songwei Ge, Haohan Wang, Eric Xing, Benjamin Lengerich, Maruan Al-Shedivat, Jennifer Williams, and Sami Labbaki. It has been an honor to work with such incredibly smart, creative, and hard working folks, and I'm very proud to have made meaningful contributions to our field with all of you.

I have to also acknowledge the administrative staff of CPCB, especially Nicole Stenger. From before I officially joined the program, through the post-defense process, through your hard work you have made sure that I had all of the resources and the help I needed to make it through grad school.

I would like to thank all of the members of the Bar-Joseph lab as well as my fellow students and postdocs in CPCB, for the helpful feedback, impromptu whiteboarding sessions, and paper sharing that helped me make progress in my own work. In particular, I'd like to thank Michael Kleyman, Easwaran Ramamurthy, Jenn Williams, Yifeng Tao, Yihang Shen, Dora Li, Cathy Su, Ruochi Zhang, Natalie Sauerwald, Ben Lengerich, Sabrina Rashid, Matt Ruffalo, and Jose Lugo-Martinez.

Last but not least, I want to thank my family. My mother, who set an incredibly high bar for me by attaining her PhD while raising two small children, and continuing to support me even from afar. My father, whose vision, passion, and drive through his own graduate school and career experiences has never failed to guide me in the right direction. And my sister, whose sense-of-self and motivation serve as a constant inspiration for me, and whose comic relief is always welcome. Without the love and support of these people, I'd be nowhere. I'd also like to acknowledge the loving support of my extended family, near and far, whose encouragement means so much to me.

Contents

List of Figures

List of Tables

Chapter 1

Background

1.1 scRNA-seq

1.1.1 Motivation and history

One of the most important problems in molecular biology is profiling the functional state of cells or groups of cells in tissues. In the field of functional genomics, transcriptomics have played an integral role in shedding light on the relationships and roles of genes in their biological contexts. With the advent of next generation sequencing (NGS) technologies, we gained the ability to assay the transcriptome of animal models in an unbiased manner, unlocking a global view that was previously unavailable with contemporary micro-array technology. These RNA-seq technologies fueled a systems biology research, particularly in the discovery of transcriptional signatures and regulatory and signaling networks. However, uncovering the state of a cell (a latent representation) is key to answering fundamental questions in biology such as predicting perturbations that control cell development, differentiation, and their responses to pathogens or cancer. Answering these questions at the cellular level requires higher-resolution assays than "bulk RNA-seq", which provides transriptomics for groups of (*find out how many typically*) cells together.

Single-cell RNA-sequencing, or scRNA-seq, is a transcriptomics assay that allows researchers to profile the expression of every gene (or transcript) in *individual cells*, in a high-throughput and unbiased manner [2]. A relatively new technology, scRNA-seq has already become an indispensable tool for transcriptomics research. Since its development in 2009, an number of improvements to library preparation protocols have been introduced, mostly driven by the development of droplet microfluidics [1, 3–5] and split-pool barcoding [6, 7] methods (See Figure 1.1 for an example). These technologies have improved the throughput of scRNA-seq to tens of thousands of cells in a single experiment. As these technologies have become commercialized, access to this technology has become available to a broad range of researchers, conducting experiments with scRNA-seq ranging from cell type identification [8, 9], spatial characterization of cells in tissues [10, 11], and tracking cellular development [12, 13].

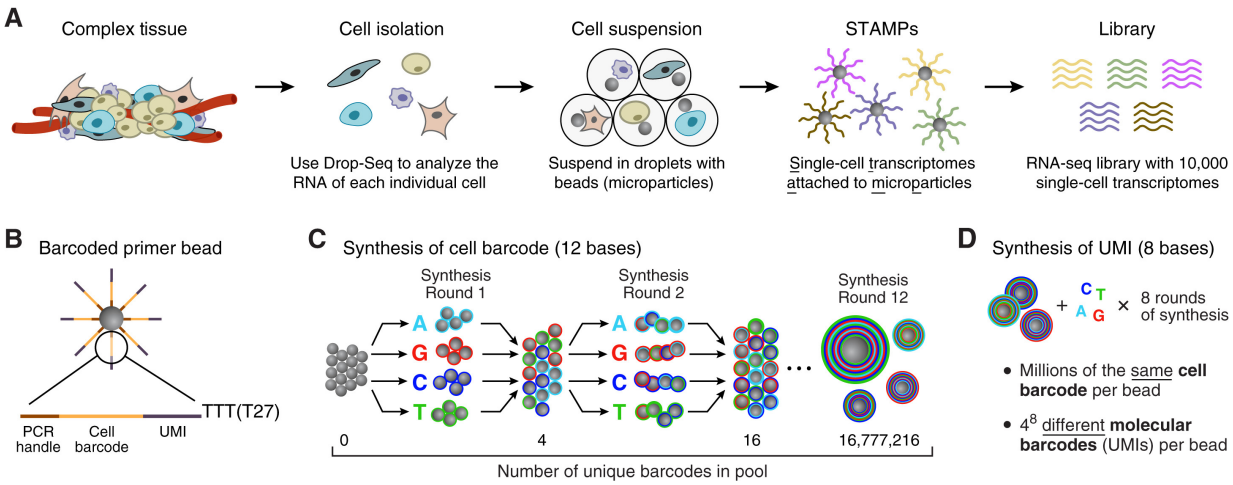


Figure 1.1: The Drop-seq sequencing protocol. Individual cells are isolated into nanoliter-sized aqueous droplets. y for quickly profiling thousands of individual cells by separating them into nanoliter-sized aqueous droplets, associating a different barcode with each cell's RNAs, and sequencing them all together. (A) The barcoding schematic, where individual cells are suspended together with microparticles which deliver primers which include cell and UMI barcodes. (B) Primers found on each microparticle. (C) Split-and-pool procedure for synthesizing cell barcodes. (D) Synthesis of unique molecular identifiers. Figure taken from [1]

1.1.2 Typical experimental pipelines

A typical scRNA-seq workflow is depicted in Figure 1.2. Different protocols (e.g. Drop-seq, inDrop, CEL-seq, 10x Chromium, or others) may vary in the "Single Cell Isolation" up through the "Sequencing Library" stages. These different protocols may use (Drop-seq reaction chambers explanation) to conduct the amplification stage. Importantly, most scRNA-seq protocols make use of "multiplexing", or the ability run a single library preparation protocol for thousands of single cells at the same time. This is achieved by attaching short, unique cellular barcodes to the RNA molecules of each cell. Then, the reads originating from distinct cells can be determined after sequencing by matching read barcodes to cells.

Another development that is now commonplace is the use of unique molecular identifiers (UMIs) [14]. Due to the low number of mRNA molecules in single cells, the sensitivity of RNA-seq is very low. Thus, all protocols involve an amplification step, to boost the signal of mRNA presence in the sample. Ideally, this amplification would be evenly applied to all mRNA molecules. However, it is a well-known issue with PCR that the amplification is biased. To increase detection, while also accounting for amplification bias, UMIs (which are short, unique barcode sequences) are bound to each mRNA molecule prior to amplification. This allows us to filter out duplication reads and correct PCR biases downstream.

Once mRNAs have been reversed transcribed to cDNAs, hybridized with cell barcodes and UMIs, and amplified, the sample is now ready for sequencing. This is done via NGS technology, such as Illumina. Different library preparation protocols and sequencers my have different read quality control steps. Afterwards, reads are stored in their raw sequence form as FASTQ files. The raw reads in the FASTQ files are then aligned with a reference genome to quantify the level of expression of each transcript (or genes). Alternatively, alignment-free methods may also be used in scRNA-seq studies if the quantification is all one needs [15]. The final result

of the experimental pipeline will be a read-count matrix with N rows (one row for each cell, or barcode), and G columns (one column for each expressed gene, or transcript if transcript-level quantification is used). The values in the matrix are the number of reads for each gene in each cell.

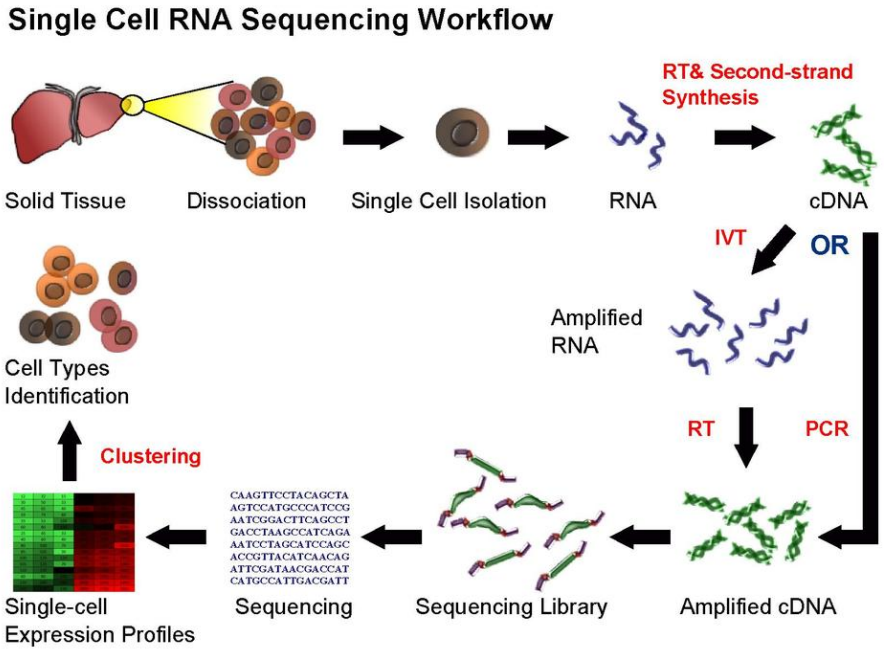


Figure 1.2: Illustration of a typical scRNA-seq workflow. Much of the procedure is identical to bulk RNA-sequencing (e.g. reverse transcription, cDNA amplification, sequencing via NGS). The key difference is that first individual cells are isolated, and in isolation, they are lysed and their mRNAs captured and barcoded with cell barcodes. The isolation method and primer delivery methods may differe between different scRNA-seq protocols. Drop-seq is one example (Figure 1.1). Image taken from `https://commons.wikimedia.org/wiki/File:RNA-Seq_workflow-5.pdf`; Attribution: Yijyechern, CC BY-SA 3.0 <`https://creativecommons.org/licenses/by-sa/3.0`>, via Wikimedia Commons

1.2 Computational techniques used in the analysis of scRNA-seq data

While single cell technologies revolutionized our ability to profile cells and the genes they express, they also present several computational challenges. There are many data analysis and bioinformatics techniques that are used to address these challenges in the process of analyzing and interpreting scRNA-seq data. We utilize most of these analyses in some form in the projects discussed in this thesis. We thus provide a brief introduction and description of these methods here, as well as the computational challenges involved.

1.2.1 Sparsity and dropout

Introduction

The $N \times G$ read-count matrix from scRNA-seq is often very sparse. Here "sparse" means "contains many zeros." These zero expression values may be "real" and due to the stochastic nature of transcription, or they may be "technical" and due to the low starting amount of mRNA in individual cells and low capture efficiency of scRNA-seq [16]. These excessive, technical zeros are referred to as "dropout" events [16–19].

Methods to account for dropout

Dealing with the zero-inflated nature of scRNA-seq read matrices is an important computational challenge. However, there are no dedicated methods to "remove" or "fix" the dropout issue in scRNA-seq. Rather, dropout is something that must be accounted for when undertaking other data analysis tasks with scRNA-seq data. We highlight these cases in the following sections. For example, the field has developed dimensionality reduction techniques to account for dropouts [16], imputation methods to infer the expression of dropout genes [17, 20, 21], and bayesian approaches to probabilistically modeling dropout events alongside gene expression [19, 20]. In particular, differential expression (DE) analysis methods are typically statistical methods, and while we have many methods for DE from bulk RNA-seq analysis, the main bottleneck in applying these methods to scRNA-seq data is that they do not account for the dropout issue that is specific to scRNA-seq. As discussed in the next section, the main adaptation that researchers have made to these methods to make them appropriate for application to scRNA-seq data is to explicitly model the dropout component in their statistical DE approaches.

1.2.2 Differential expression analysis

Motivations

Gene expression profiling is an invaluable tool in biomedical research, and has been useful in drug discovery, diagnosis, cancer, and toxicology [22]. Central to all of these applications, is the need to identify genes that are differentially regulated or expressed in different conditions. If we are able to identify lists of differentially expressed genes for certain conditions or phenotypes, then these lists can serve as "markers" for these phenotypes (see Figure 1.3 for an example). In drug discovery or toxicology settings, these lists could contain potential drug or therapeutic targets. It is worth noting that in whole-transcriptome settings, such lists are usually seen as a hypothesis generation step, to guide further investigation experimental validation.

Current approaches

The task of finding genes that are (significantly) up or down regulated in one group compared to another is called differential expression (DE) analysis. Most of the early method development in this area was for microarray data. The problem of detecting DE genes is posed as a statistical hypothesis test, where the null hypothesis is that the gene is expressed at the same level in two

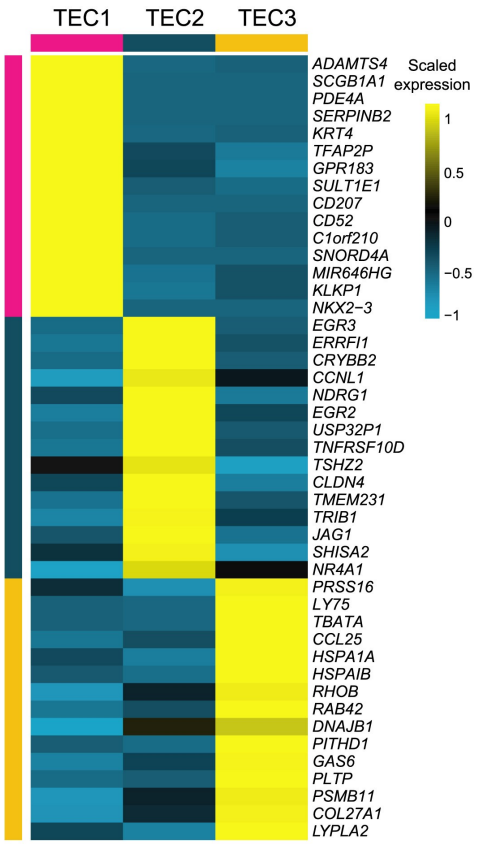


Figure 1.3: An example of differentially expressed genes among three distinct clusters of thymic epithelial cells (TECs) from [23]. Heatmap shows the average scaled expression of genes in these clusters. The top 15 significant up-regulated differentially expressed genes for each cluster are listed and arranged to show the block structure of modules of up-regulated genes for each cluster. Figure taken from [23].

groups, and the alternative is that it is not. The t-test for continuous data and its variants were common early approaches, and are still used to this day.

With the advent of scRNA-seq, new approaches for DE analysis have been developed to account for the specific characteristics of this data. In particular, because NGS technologies return count data, researchers have experimented with different statistical distributions that better model this type of data. The negative-binomial is a popular discrete distribution for modeling scRNA-seq count data, and is used by DESeq2 [24] and edgeR [25]. It is worth noting that the Poisson distribution is also sometimes used, and can be seen as a special case of the negative-binomial distribution [25]. With these underlying distributions, a likelihood ratio test is conducted and a p-value is attained, which is usually corrected for multiple testing [24]. The aforementioned methods were originally developed for bulk RNA-seq data, but continue to be applied to scRNA-seq data as well.

Another important aspect to account for when running DE analysis for scRNA-seq data is the zero-inflated nature of the data. To account for these dropout events, some approaches use a bimodal distribution, where the "expressed" peak is modeled by a log-normal distribution, and the dropout component is modeled as a constant point-mass at zero [26]. Other recent approaches

use a mixture distribution, for example in SCDE the expression of a gene is modeled as a mixture of a negative-binomial (the expressed component) and a low-magnitude Poisson process (the zero component) [19].

The approaches for DE analysis mentioned above are all parametric methods, meaning that they require the parameters of statistical distributions to be fit. A popular alternative is to use non-parametric approaches, such as the Wilcoxon rank sum test [27]. Most of the approaches mentioned in this summary, or variants thereof, are implemented in both scanpy and Seurat [28–30]. For a detailed review and comparison of the various methods for DE analysis of scRNA-seq data, see the review by Soneson and Robinson [31].

1.2.3 Gene set enrichment analysis

Introduction and motivation

After running DE analysis, it is common to have a resulting list of hundreds or even thousands of significantly differentially expressed genes, and the task of then interpreting such a list for biological relevance is difficult. The approach that is usually used to aid in interpretation is called gene set enrichment analysis (GSEA) [32]. The basic premise of this test is that given a query list of genes of interest for a phenotype (e.g. the list of significant DE), we would like to compare this list to other reference lists of related genes which are annotated with prior biological knowledge, and determine if our query list significantly overlaps with any of these reference lists [33]. If so, we are able to say that the query list is "enriched for" or "over-represented in" the reference list(s). These reference lists may come from various sources of annotated genes; a popular choice is the Gene Ontology (GO) [34, 35]. GO is a manually curated ontology, which is a hierarchical organization of biological terms, such that terms at the specific "leaf" levels also belong to the more general parent terms higher levels (Figure 1.4). The GO is organized as three directed acyclic graphs (DAGs), with separate ontologies for the different domains of "biological process", "cellular component", and "molecular function" [34]. Another source of annotated gene lists is the Kyoto Encyclopedia of Genes and Genomes (KEGG) [36–38], which contains many databases of higher order molecular functions that genes (or gene products) are involved in, and its "PATHWAY" database is commonly used in GSEA analysis for transcriptomics studies.

Statistical formulation

A GSEA is framed as a statistical hypothesis test, in which the null hypothesis is that there is no over-representation of the query list in the reference list (i.e. that the overlap between the two is can be explained by random chance) and the aternative hypothesis is that the query list is over-represented in the reference list (some non-random process is involved). This is commonly formalized with a hypergeometric test (which is equivalent to a one-sided Fisher's exact test) [39]. It is important to account for the "background" set, which is the set of all genes that were assayed in your experiment.

Formally, We quantify the similarity between two lists of marker genes A and B by using the hypergeometric test for overrepresentation. The premise of this test is that we have an urn filled with two types of balls, m black balls and n white balls (for a total of m + n balls in this urn).

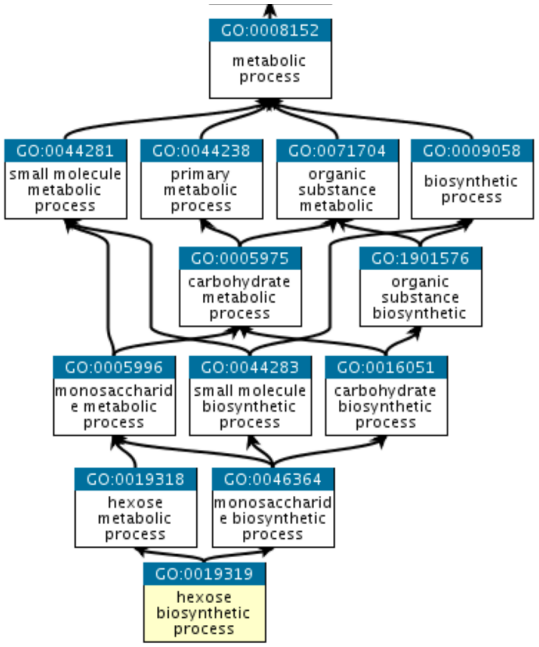


Figure 1.4: A diagram depicting a portion of the Biological Process domain of the Gene Ontology. The ontology is represented as DAG, and each term (a rectangle in the diagram) has a unique identifier, and also has a list of genes that are annotated as being associated with each GO term. These annotation lists are the reference lists when used for GSEA. Image taken from `http://geneontology.org/docs/ontology-documentation/`.

Then, given k random draws of a single ball from this urn without replacement, and given that q of these k balls are white, we then ask the question: what is the probability of drawing q or more white balls from this urn? This is the probability of our result under the null hypothesis that the white and black balls are distributed equally and your probability of drawing a white ball is based on proportion of white and black balls in the urn, and not biased by some non-random process. If this probability of drawing q or more white balls out of your k draws from your urn is very low, then you will reject the null hypothesis, as your observed draws are not well-explained by the null hypothesis.

1.2.4 Dimensionality reduction and visualization

Introduction

A major challenge of working with scRNA-seq data is its high-dimensionality. Typical scRNA-seq experiments assay the relative expression of on the order of 20,000 genes (in humans or mice), for each of thousands to tens of thousands of cells. Visualizing each cell as a point in a 20,000 dimensional space is unwieldy, and downstream analysis such as clustering and classification do not scale well to extremely high-dimensional cases. As a result, dimensionality reduction is usually an early step in scRNA-seq analysis. For the reasons listed above, dimensionality reduction and visualization often go hand-in-hand.

Current approaches

Principle components analysis (PCA) is a classical method for dimensionality reduction [40]. It is a linear method, whose objective is to find a set of successive axes in the original data space which most accurately represent the data. Here, accurate data representation is synonymous with capturing as much variance in the data as possible. These axes form an orthonormal basis, and the data can be projected onto them, while only keeping the first few projections to reduce the dimensions (e.g. projecting onto the first two axes for 2D visualizations). PCA and Variants of PCA have been developed and applied to scRNA-seq data [41]. PCA is a linear method, and factor analysis (FA) is another similar linear technique. While PCA focuses on modeling covariances, FA models correlations. For scRNA-seq data, zero-inflated factor analysis (ZIFA) is an example of a dimensionality reduction technique that accounts for the high dropout in scRNA-seq data [16].

While PCA is a matrix decomposition approach, another category of dimensionality reduction methods often applied to scRNA-seq data are manifold learning techniques. The most famous example of manifold learning is t-distributed stochastic neighbor embedding (t-SNE) (Figure 1.5a) [42]. t-SNE is a statistical approach, which aims to learn a reduced dimension whose pair-wise distances between the points remains faithful to the pair-wise distances between points in the original high-dimensional space. This fidelity to the original data is measured by the Kullback-Leibler divergence measure between pairwise distance distributions in the low and high-dimensional spaces [42]. This divergence is minimized via gradient descent, though because the objective function is non-convex, t-SNE may return different solutions each time it is run. Another criticism of t-SNE, which has more implications for scRNA-seq data analysis, is that since it only seeks to model pair-wise distances, t-SNE does not capture global structure in the data. In other words, seeing some cells embedded close to other cells may be indicative of their proximity in the original gene space, but observations of distances between distinct clusters of cells are less informative in t-SNE embeddings.

To address the latter criticism of t-SNE, Uniform Manifold Approximation and Projection (UMAP) was developed [43] (Figure 1.5a). UMAP is competitive with t-SNE, but arguably preserves global structure in the data more faithfully. UMAP is similar to t-SNE in many ways, as both are k-neighbour graph based algorithms. However, the formulation of UMAP is based on Riemannian geometry and algebraic topology [43]. In addition to better global structure fidelity in practice, UMAP also has much lower run time cost, and so is more scalable to large data. For these reasons, UMAP has become the most popular method for visualizing large scRNA-seq datasets [44].

It is worth noting that for both t-SNE and UMAP, PCA is usually run beforehand as an initial dimensionality reduction step, and then t-SNE and/or UMAP embeddings are computed from the PCA coordinates of the data.

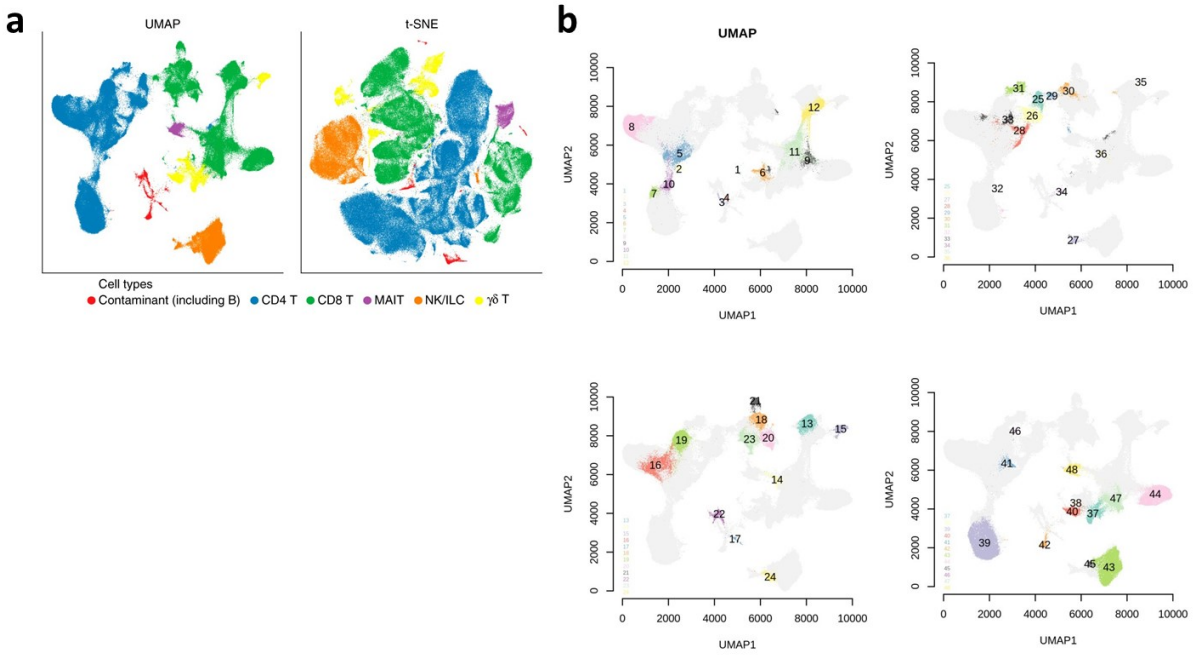


Figure 1.5: An example of dimensionality reduction and clustering of scRNA-seq data. (a) UMAP and t-SNE embeddings of over 300,000 single cells from a multi-tissue immune cell atlas [45], colored by the broad cell lineages. (b) Louvain-based Phenograph clustering result visualized on the UMAP embedding. Only twelve clusters shown per plot for clarity. Figure adapted from [44].

1.2.5 Clustering

Introduction and motivation

Characterizing the heterogeneity of cells in tissues is the predominant motivating factor for using scRNA-seq technology. However, researchers rarely have sub-type or even cell-type label information about each cell *a priori*. Because of this, clustering (an unsupervised machine learning technique) is almost always required in scRNA-seq data analysis to characterize the cellular populations in the data [46]. These clustering approaches are also usually applied to PCA-reduced dimensions of the scRNA-seq data.

Current approaches

K-means, a greedy algorithm for finding a user specified number of clusters, is a common general approach for clustering [47]. However, the tendency to find equal size clusters (therefor washing out distinct rare cells) and the greedy nature of the algorithm (different solutions based on initialization) are not ideal for scRNA-seq data analysis. Extensions and adaptations to k-means have been developed to better suit scRNA-seq data. SC3 arrives at a consensus clustering by repeated runs of k-means [48], and RaceID accounts for rare cells by incorporating an outlier-detection step [49].

Another category of clustering algorithms that have been applied to scRNA-seq data are

nearest-neighbors graph based algorithms. Phenograph is an example of this approach, which works by first constructing a weighted graph based on nearest neighbors of cells, and then the Louvain algorithm (a community detection algorithm) is run to partition the graph so as to maximize modularity [50, 51] (Figure 1.5b). Modifications to this approach have been to use the Leiden community detection algorithm instead of the Louvain algorithm, as the Leiden algorithm has stronger gaurantees of community connectedness and runs faster [52]. Both of these algorithms are the most popular in this category, and have been implemented in the popular scanpy (Python) [28] and Seurat (R) [29, 30] software packages.

1.2.6 Cell type assignment

Introduction and motivation

A central task in scRNA-seq analysis is to assign each cell to a phenotype from set of phenotypes of interest. This could be the "age" of the cell, the tissue or origin, the "stem-ness", or most commonly, the cell type, such as T cell, B cell, muscle cell, etc. This computational task is central to the work in this thesis, as many of the methods proposed herein rely on having cell type labeled data. Furthermore, cell type classification is also posed as a machine learning objective that some of our methods use during their fitting procedure.

Current approaches

Typically, this cell type assignment is achieved through unsupervised methods [8, 53]. After processing the scRNA-seq read count matrix, a researcher will run an unsupervised clustering algorithm, typically the graph-based algorithms like Phenograph (see section 1.2.5). After this, they will then run one-versus-all differential expression analysis (see section 1.2.2) to identify the up-regulated genes in each of their clusters. Then they will cross-reference these lists of top genes in each cluster with lists of known marker genes for cell types from the literature [54] and assign a cell type to the whole cluster based on this correspondence.

Another major category of cell type assignment methods are those that are based on supervised methodology. In these approaches, a labeled training dataset of scRNA-seq samples is used to fit a classification model, which is then applied to test data to evaluate its performance. Once a model is selected and fitted using labeled data, this model can now serve as a quantitative definition of cell types based on expression profiles, and can be used to predict the cell types of new scRNA-seq data. Examples of such methods include scPred which fits a support vector machine (SVM) classifier with a radial kernel on top of a PCA-reduced dimension representation of the data [55], and Moana which also uses SVMs (linear kernel) on top of PCA dimensions, but uses multiple independent classifiers structured in a hierarchy to first distinguish general cell types and then classify amongst subtypes for each broad cell type [56].

1.3 Biological applications of scRNA-Seq data

Given the high resolution capability of scRNA-seq discussed in section 1.1.1, as well as the many data analyses available to interpret this data modality that we introduced in the prior section, it

is no surprise that scRNA-seq has been applied to many problems in biology. In this section we discuss a few examples of these and highlight the importance of scRNA-seq.

1.3.1 Development

Cellular development and differentiation are complicated processes that depend on cell-cell interactions and gene regulatory programs. This cell-specific behavior is thus one of the most active areas of scRNA-seq application. In particular, large scale scRNA-seq experiments allow us to capture a snapshot of cellular differentiation at many intermediate stages. Intuitively, a single scRNA-seq sample can be viewed as actually being a time series, wherein each individual cell may be at a distinct time point along a development continuum. Unsupervised algorithms have been developed to recover this "pseduotime" ordering of the cells from scRNA-seq data [57]. One example of this analysis is the study by Nowotschin *et al.* [58], focusing on understanding early embyronic development. In particular, they model cellular differentiation in the mouse gut endoderm, and they apply pseudotime trajectory inference to their scRNA-seq data. They uncover a novel relationship between descendants of two primitive lineages, and also find that some cell fates are globally similar, but keep some characteristics of their specific lineage histories [58].

1.3.2 Disease

ScRNA-seq can also be applied to disease settings, where we are interested in understanding disease progression, cellular response to disease, and discovering potential targets for therapies. These studies often focus on identifying underlying cellular events and responses related to disease onset, outcomes, and treatment. As an example, Mathys *et al.* [59] applied scRNA-seq to Alzheimer's disease (AD), a neurodegenerative disease. They focused on characterizing the transcriptome of microglia cells, the primary resident immune cells in the brain, and believed to be involved in AD progression. Through time series scRNA-seq, they identified two distinct AD-reactive microglia phenotypes, both of which were characterized by genes that are involved in immune response and activation of immune effector cells. Their comprehensive study also revealed disease-stage-specific microglial cell states and illuminated the cellular reprogramming that happens in microglia during neurodegeneration using trajectory analysis of their scRNA-seq profiles.

1.3.3 Tissue mapping

Tissue-of-origin mapping of cell types is an important piece of understanding cellular identities. This is particularly the case when studying primary cells from donors, where their context in the body plays an important role in their function and identity. Tissue mapping is also critical in the cases where we are studying heterogeneous cell types that coexist in close proximity to each other (e.g. the brain or the embryo). ScRNA-seq can also be applied to this biological problem setting. In particular, Achim *et al.* [60] developed an approach based on scRNA-seq profiles that can identify the spatial origin of cells within a tissue of interest. They did this by comparing the scRNA-seq profile of cells with a reference positional gene expression atlas. They found that

their method could accurately assign cells to their locations in the brain of a marine ragworm model organism. This contribution is important for understanding how the brains of vertebrates evolved, as this model organism is commonly used for studying brain evolution [60].

1.4 Supervised neural networks

Several chapters in this thesis present methods that extend and use variants of neural networks. We thus provide a brief general introduction to these methods here and discuss specific architectures in more details in the following chapters.

1.4.1 Introduction

Artificial neural networks (ANNs or just NNs) are a popular class of machine learning models which are inspired by they way that networks of biological neurons operate in the human brain. They are a multi-paradigm technique in that they can be used for supervised, unsupervised, and reinforcement learning problems. The earliest descriptions of NNs were through the foundational work of neurophysiologists and mathematicians who sought to understand how the neurons in the central nervous system process signals [61]. These were then further developed, as the algorithms by which the parameters of these models were set were formally described [62]. Researchers began successfully applying NNs to real-world problems [63], and as more powerful processing hardware allowed for training more complicated models on larger datasets [64], the deep learning revolution took the world by storm.

1.4.2 Components

The basic building blocks of a ANN are a neuron and a non-linear activation function. The artificial neuron is a unit which takes in a real-valued signal from other neurons ($\boldsymbol{x}$) that are connected to it, and aggregates them via a weighted sum, and then computes a non-linear function (g) of this sum which it then passes on to other neurons that are connected to it (Figure 1.6). Formally, the pre-activation function (a) of a neuron given input x is:

$$a(\boldsymbol{x}) = b + \sum_i w_i x_i = b + \boldsymbol{w}^T \boldsymbol{x}$$

and the output activation of the neuron (h) is then:

$$h(\boldsymbol{x}) = g(a(\boldsymbol{x})) = g(b + \boldsymbol{w}^T \boldsymbol{x})$$

Choices for the activation function $g(z)$ include the sigmoid $\frac{1}{1+e^{-z}}$, hyperbolic tangent $\tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, the rectified linear unit $ReLU(z) = \max(0, z)$, and many more. The free parameters for the neuron are the weights $\boldsymbol{w}$.

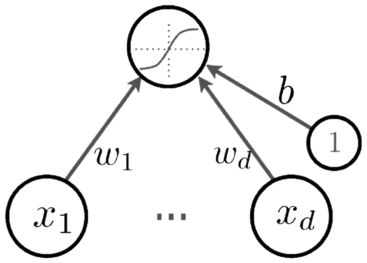


Figure 1.6: Diagram of a single artificial neuron. Image taken from `http://info.usherbrooke.ca/hlarochelle/ift725/1_01_artificial_neuron.pdf`

1.4.3 Organization and architecture

These artificial neurons are structured in sequential layers. Each layer is a group of neurons. Connections between neurons in the same layer are not allowed. In a typical feed-forward neural network, the input to each neuron in one layer is the vector of output activations from each of the neurons of the previous layer. Signals are propagated forward through a multi-layer neural network from the input layer through zero or more hidden layers and finally to the output layer (Figure 1.7). The architecture described above is sometimes also referred to as a "dense" neural network architecture because the neurons in adjacent layers are fully connected (except for connections between neurons in the same layer).

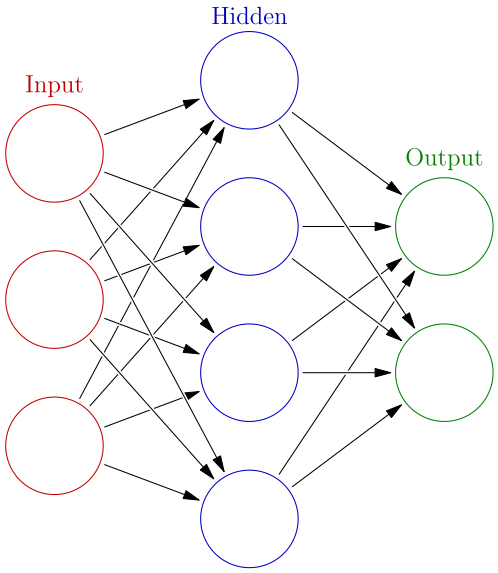


Figure 1.7: An example of a feed-forward neural network architecture with three input dimensions, a single hidden layer of width four, and two dimensional output layer. Image taken from `https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg`; Attribution: Glosser.ca, CC BY-SA 3.0 <`https://creativecommons.org/licenses/by-sa/3.0`>, via Wikimedia Commons

1.4.4 Fitting neural networks

The goal of fitting the weights $\boldsymbol{W}$ (where $\boldsymbol{W}$ is the collection of $\boldsymbol{w}$ weight vectors from all neurons in all layers of the neural network) for the supervised machine learning case is to adapt $\boldsymbol{W}$ so that the output signals predicted by the model given an input example match the expected ground-truth output for that example as much as possible. This is an optimization task, in which we have a cost function that we seek to minimize via stochastic gradient descent (SGD). The cost function neural networks are very complicated and highly non-convex, and many techniques such as dropout, adaptive learning rate optimization methods, and batch normalization have been developed to prevent overfitting and avoid getting stuck in local optima. A prerequisite to applying any gradient descent algorithm is to first compute the gradient of the cost function with respect to the model parameters. The backpropagation algorithm is an efficient procedure for computing this gradient that is used for training all neural networks [62].

1.4.5 Loss functions for supervised neural networks

Choices for loss functions in supervised learning very depending on the task. For regression tasks, the mean squared error (MSE) loss may be used, as is used in linear regression. For multiclass classification, the cross entropy loss may be used. In that case, the output of the neural network must be interpretable as the probability of each class. Another class of loss functions exist for the task of deep metric learning, which is concerned with learning how similar or dissimilar objects are from each other, not necessarily identifying their individual classes. Loss functions for this task include contrastive loss functions such as the Siamese loss [65, 66], the triplet loss [67], and the histogram loss [68].

1.4.6 Software packages

As part of the deep learning boom, several popular software packages have been developed to aid in deep learning research. At the heart of all of these packages is something called ”automatic gradient computation” or “autograd,” which is the ability to efficiently compute the gradient of a complicated function (e.g. a deep neural network) with the backpropagation algorithm. In addition to providing autograd, they also provide implementations of popular loss functions, optimizers, and other pieces necessary to tune neural networks. We use these packages throughout this thesis, and the two most popular at the time of writing are TensorFlow [69] and PyTorch [70].

1.5 Thesis organization

This thesis is presented in 5 chapters. Chapter 1 covers the background and the introductory material that sets the foundation for much of the work discussed in the rest of the thesis. Chapter 2 concerns the problem of comparative analysis of scRNA-seq samples, in the large-data setting when prior knowledge about marker genes might not be available. Chapter 3 addresses another major theme in comparative analyses: accounting for batch effects. This chapter discusses two

contributions to this area, a supervised and an unsupervised approach. Chapter 4 discusses the problem of finding marker genes that are specific to a cell type and simultaneously specific to a tissue, and explores an approach applied to a recent human scRNA-seq dataset. Chapter 5 concludes the thesis by summarizing the major themes of the work, reiterating the limitations of the work, and discussing directions for future research.

Chapter 2

Enabling comparative analysis of new scRNA-seq datasets with large, publicly available databases.

ScRNA-seq studies profile thousands of cells in heterogeneous environments. As mentioned in Chapter 1, current methods for characterizing cells perform unsupervised analysis followed by assignment using a small set of known marker genes. Such approaches are limited to a few, well characterized cell types. We developed an automated pipeline to download, process, and annotate publicly available scRNA-seq datasets to enable large scale supervised characterization. We extend supervised neural networks to obtain efficient and accurate representations for scRNA-seq data. We apply our pipeline to analyze data from 500 different studies with 300 unique cell types and show that supervised methods outperform unsupervised methods for cell type identification. A case study highlights the usefulness of these methods for comparing cell type distributions in healthy and diseased mice. Finally, we present scQuery, a web server which uses our neural networks and fast matching methods to determine cell types, key genes, and more.

A version of this chapter was published in *Nature Communications* and presented as a talk at Regulatory and Systems Genomics conference in 2018 and is joint work with Matthew Ruffalo, Aiyappa Parvangada, Zhilin Huang, and Ziv Bar-Joseph [71]. The reprocessed data that support the findings of this study are publicly available online at `https://scquery.cs.cmu.edu/processed_data/`. Code for our preprocessing (alignment/quantification) pipeline is available at `https://github.com/mruffalo/sc-rna-seq-pipeline`. Code for training and evaluation of our neural network models is available at `https://github.com/AmirAlavi/scrna_nn`. Code for our differential expression analysis to find cell type-specific genes is available at `https://github.com/AmirAlavi/single_cell_deg`.

2.1 Introduction

ScRNA-seq has recently emerged as a major advancement in the field of transcriptomics [72]. Compared to bulk (many cells at a time) RNA-seq, scRNA-seq can achieve a higher degree of resolution, revealing many properties of subpopulations in heterogeneous groups of cells [73].

Several different cell types have now been profiled using scRNA-seq leading to the characterization of sub-types, identification of new marker genes, and analysis of cell fate and development [74–76].

While most work attempted to characterize expression profiles for specific (known) cell types, more recent work has attempted to use this technology to compare differences between different states (for example, disease vs. healthy cell distributions) or time (for example, sets of cells in different developmental time points or age) [59, 77]. For such studies, the main focus is on the characterization of the different cell types within each population being compared, and the analysis of the differences in such types. To date, such work primarily relied on known markers [53] or unsupervised (dimensionality reduction or clustering) methods [8]. Markers, while useful, are limited and are not available for several cell types. Unsupervised methods are useful to overcome this, and may allow users to observe large differences in expression profiles, but as we and others have shown, they are harder to interpret and often less accurate than supervised methods [78].

To address these problems, we have developed a framework that combines the idea of markers for cell types with the scale obtained from global analysis of all available scRNA-seq data. We developed scQuery, a web server that utilizes scRNA-seq data collected from over 500 different experiments for the analysis of new scRNA-Seq data. The web server provides users with information about the cell type predicted for each cell, overall cell type distribution, set of differentially expressed (DE) genes identified for cells, prior data that is closest to the new data, and more.

Here, we test scQuery in several cross-validation experiments. We also perform a case study in which we analyze close to 2000 cells from a neurodegeneration study [59], and demonstrate that our pipeline and web server enable coherent comparative analysis of scRNA-seq datasets. As we show, in all cases we observe good performance of the methods we use and of the overall web server for the analysis of new scRNA-seq data.

2.2 Methods

2.2.1 Data collection and preprocessing

We selected a mouse gene set of interest based on the NCBI Consensus CDS (CCDS) which contains 20,499 distinct genes. For genes with multiple isoforms, we consolidate all available coding regions (Supporting Methods). To search for scRNA-seq datasets, we queried the NCBI Gene Expression Omnibus (GEO) and the ArrayExpress database for mouse single-cell RNA-seq series (See Supporting Methods for the queries we used). We then download metadata for each series returned in this query and parse this metadata to identify the distinct samples that comprise each series. We examined the metadata for each sample (*e.g.* library strategy, library source, data processing) and exclude any samples that do not contain scRNA-seq data.

We next attempted to download each study's raw RNA-seq reads and for those studies for which this data is available we developed a pipeline that uniformly processed scRNA-seq data. We use the reference mouse genome from the UCSC genome browser [79] (build mm10), and align RNA-seq reads with HISAT2 [80] version 2.1.0. We align reads as single- or paired-end as

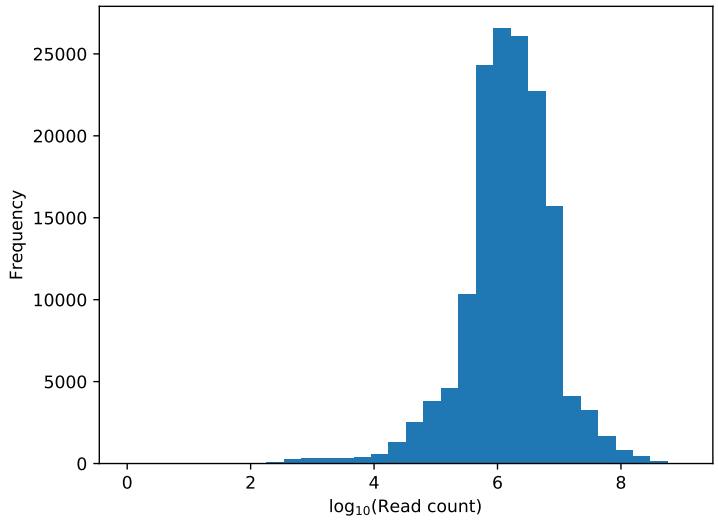


Figure 2.1: **Distribution of reads in available raw data.** Number of RNA-seq reads in each available series/cell, across all studies.

appropriate, and discard samples for which fewer than 40% of reads align to coding regions. We represent gene expression using RPKM. See Figure 2.1 for a histogram of read counts per series.

Labeling using Cell Ontology terms

We use the Cell Ontology (CL) [81] available from `http://obofoundry.org/ontology/cl.html` to identify the specific cell types which are represented in our GEO query results. We parsed the ontology terms into a directed acyclic graph structure, adding edges between terms for "`is_a`" and "`part_of`" relationships. Note that this choice of edge direction means that all edges point *toward* the root nodes in the ontology.

We use the name and any available synonyms for each ontology term to automatically identify the matching terms for each sample of interest (Supporting Methods). This produces a set of ontology term hits for each sample. We filter these ontology term hits by excluding any terms that are descendants of any other selected terms (*e.g.*, term `CL:0000000` "cell" matches many studies), producing a set of "specific" ontology terms for each sample – for any two nodes u and v in such a set, neither u nor v is a descendant of the other in the ontology.

2.2.2 Dimensionality Reduction

Most current analysis methods for scRNA-seq data use some form of dimensionality reduction to visualize and analyze the data, most notably PCA and similar methods [16, 41] and t-SNE [82]. Past work has shown that while such methods are useful, supervised methods for dimensionality reduction may improve the ability to accurately represent different cell types (see [78]).

Using neural networks for dimensionality reduction has been shown to work well as a supervised technique to learn compact, discriminative representations of data [83]. The original, unreduced dimensions form the input layer to a neural network, where each dimension is an

input unit. After training the model towards a particular objective (such as classification), the last hidden layer, which is typically much smaller in the number of units than the input layer, may be taken as a reduced dimensionality representation of the data. These learned features are referred to as *neural embeddings* in the literature, and here we tested a number of different neural network architectures which either explicitly optimize these neural embeddings (for example, siamese [65] and triplet networks [67]) or those that only optimize the label accuracy. All neural networks we used were implemented in Python using the Keras API [84].

Neural network architectures

Prior work showed that sparsely connected NN architectures based on protein interaction data can be more effective in determining cell types when compared to dense networks [78]. Here we further studied other NN networks architectures and compared their performance to the PPI and dense networks. First, we looked at another method to group genes based on the Gene Ontology (GO) [85]. To construct a hierarchical neural network architecture that mirrors the structure of GO, we associate input genes with GO nodes. Multiple genes are associated (and connected to) the same node. We use this grouping of the input genes as the first hidden layer of a neural network. Nodes in the next hidden layer will be constructed from GO nodes that are descendants of nodes in the prior layer. We continue this process until the last hidden layer has the desired number of nodes (the size of our reduced dimension). The final result is the network depicted in Figure 2.5. See also Supporting Methods.

Siamese architectures trained with contrastive loss

The NNs discussed above indirectly optimize the neural embedding layer by optimizing a classification target function (correct assignment of scRNA-seq data to cell types). A number of NN architectures have been proposed to explicitly optimize the embedding itself. For example, siamese neural networks [65, 66] (Figure 2.2) consist of two identical twin subnetworks which share the same weights. The outputs of both subnetworks are connected to a conjoined layer (sometimes referred to as the distance layer) which directly calculates a distance between the embeddings in the last layers of the twin networks. The input to a siamese network is a pair of data points and the output which is optimized is whether they are similar (same cell type) or not. The loss is computed on the output of the distance layer, and heavily penalizes large distances between items from the same class, while at the same time penalizing small distances between items from different class. Specifically the network optimizes the following loss function:

$$\text{Contrastive Loss} = \sum_{i=1}^{P} (Y^i) L_s(D^i) + (1 - Y^i) L_d(D^i) \tag{2.1}$$

Where:

$\boldsymbol{P}$ is the set of all training examples (pairs of data points)

$\boldsymbol{Y}$ is the corresponding label for each pair (1 indicates that the pair belong to the same class, 0 indicates that each sample in the pair come from different classes)

$\boldsymbol{D}$ is the Euclidean distance between the points in the pair computed by the network

$$\boldsymbol{L_s(D)} = \frac{1}{2}(D)^2 \tag{2.2}$$

$$\boldsymbol{L_d(D)} = \frac{1}{2}(\max\{0, m - D\})^2 \tag{2.3}$$

$\boldsymbol{m}$ is a margin hyperparameter, usually set to 1

Following the same motivations as siamese networks, triplet networks also seek to learn an optimal embedding but do so by looking at three samples at a time instead of just two as in a siamese network. The triplet loss used by Schroff et al. considers a point (anchor), a second point of the same class as the anchor (positive), and a third point of a different class (negative) [67]. See Supporting Methods for details.

Training and testing of neural embedding models

We conduct supervised training of our neural embedding models using stochastic gradient descent. Although our processed dataset contains many cells, each with a set of labels, we train on a subset of "high confidence" cells to account for any label noise that may have occurred in our automatic term matching process. This is done by only keeping terms that have at least 75 cells mapping to them, and then only keeping cells with a single mapping term. This led to a training set of 21,704 cells from the data we processed ourselves (36,473 cells when combined with author-processed data). We experimented with tanh, sigmoid, and ReLU activations, and found that tanh performed the best. ReLU activation is useful for helping deeper networks converge by preventing the vanishing gradient problem, but here our networks only have a few hidden layers, so the advantage of ReLU is less clear. We also experimented with different learning rates, momentums, and input normalizations (see web server for full results).

Since our goal is to optimize a discriminative embedding, we test the quality of our neural embeddings with retrieval testing, which is similar to the task of cell type inference. In retrieval testing, we query a cell (represented by the neural embedding of its gene expression vector)

against a large database of other cells (which are also represented by their embeddings) to find the query's nearest neighbors in the database.

Accounting for batch effects is a central issue in studies which integrate data from many different studies and experimental labs [86, 87]. Here, we adopt a careful training and evaluation strategy in order to account for batch effects. We separate the **studies** for each cell type when training and testing so that the test set is completely independent of the training set. We find all cell types which come from more than one study, and hold out a complete study for each such cell type to be a part of the test set (sometimes referred to as the "query" set in the context of information retrieval). Cell types that do not exist in more than one study are all kept in the training set. For our integrated dataset, our training set contained 45 cell types, while our query set was a subset of 26 of the training cell types. After training the model using the training set, the training set can then be used as the database in retrieval testing.

2.2.3 Evaluation of classification and embeddings

In both training and evaluation of our neural embedding models, we are constantly faced with the question of how similar two cell types are. A rigid (binary) distinction between cell types is not appropriate since "neuron", "hippocampus", and "brain" are all related cell types, and a model that groups these cell types together should not be penalized as much as a model that groups completely unrelated cell types together. We have thus extended the NN learning and evaluation methods to incorporate cell type similarity when learning and testing the models. See Supporting Methods for details on how these are used and how they are obtained.

2.2.4 Differential expression for cell types

We use the automated scRNA-seq annotations we recovered to identify a set of differentially expressed genes for each cell type. Unlike prior methods that often compare two specific scRNA-seq datasets, or use data from a single lab, our integrated approach allows for a much more powerful analysis. Specifically, we can both focus on genes that are present in multiple datasets (and so do not represent specific data generation biases) and those that are unique in the context of the ontology graph (i.e. for two brain related types, find genes that distinguish them rather than just distinguishing brain vs. all others).

Our strategy, presented in Algorithm 1, is DE-method agnostic, meaning that we can utilize any of the various DE tools that exist. In practice we have used Single-Cell Differential Expression (SCDE) here [19]. For this we used read counts rather than RPKM, as SCDE requires count data as input. This method builds an error model for each cell in the data, where the model is a mixture between a negative binomial and a Poisson (for dropout events) distribution, and then uses these error models to identify differentially expressed genes. We also tried another method, limma with the voom transformation [88], but the list of DE genes returned was too long for meaningful analysis (many of the reported DE genes had the same p-value). Results of our comparison of SCDE and limma-voom are shown in Table 2.2.

Algorithm 1: Cell type-specific gene list construction algorithm. The outermost loop of the algorithm is over different cell types. If there exists multiple studies of significant size ($> m$) that have profiled a particular cell type, then there is an inner loop that does a differential expression analysis for each such study. Otherwise, the algorithm will try to do a single differential expression analysis by pooling the cells of this type from the various studies. If there were multiple studies, then the final list of DE genes for a cell type is created from a meta-analysis of the DE genes from each study (Supporting Methods).

input : D = matrix of gene expression values (preprocessed as described in Supporting Methods)
m = minimum cells in a DE experiment
`DEModule(`*goup1, group2*`)` = Differential Expression analysis module

output: CellTypeGenes = Lists of cell type-specific differentially expressed genes

```
CellTypeGenes ← {∅}
foreach cell type C ∈ D do
    DEResults ← {∅}
    D' ← only cells of type C from D
    S ← set of studies in D'
    if ∃ a study s ∈ S that contains at least m cells of type C then
        foreach study s that satisfies the above condition do
            a ← SampleXFromY(m, D'_s)
            b ← SampleXFromY(m, D)
            DEResults ← DEResults ∪ DEModule(a, b)
        end
    else if |D'| >= m then
        (If there doesn't exist at least one study for C with at least m cells, then
        we try to combine all of C's studies into a single group)
        a ← SampleXFromY(m, D')
        b ← SampleXFromY(m, D)
        DEResults ← DEResults ∪ DEModule(a, b)
    else
        continue
    end

    CellTypeGenes ← CellTypeGenes ∪ MetaAnalysis(DEResults)
end
```

Another key aspect of our strategy is the use of meta-analysis of multiple DE experiments. The algorithm attempts to make the best use of the integrated dataset by doing a separate DE experiment for each study that contains cells of a particular cell type, and then combines these results into a final list of DE genes for the cell type. See Supporting Methods for the details of this meta-analysis.

2.2.5 Large scale query and retrieval

To enable users to compare new scRNA-seq data to the public data we have processed, and to determine the composition of cell types in such samples, we developed a web application. Users download a software package available on the website to process SRA/FASTQ files. The software implements a pipeline that generates RPKM values for the list of genes used in our database and can work on a PC or a server (Supporting Methods).

Once the user processes their data, the data is uploaded to the server and compared to all studies stored in the database. For this, we first use the NN to reduce the dimensions of each of the input datasets and then use approximate nearest neighbor approaches to match these to the data we have pre-processed as we discuss below.

Number of Queries	Linear Scan (secs)	NMSLib (secs)	Accuracy
100	0.457	0.001967	0.9819
500	2.417	0.005926	0.9543
1000	4.447	0.013977	0.9459

(a) NMSLB

Number of Queries	Linear Scan (secs)	ANNoY (secs)	Accuracy
100	0.457	0.028205	0.9673
500	2.417	0.137404	0.97726
1000	4.447	0.278778	0.94861

(b) ANNoY

Number of Queries	Optimal Number of probes	Linear Scan (secs)	FALCONN (secs)	Accuracy
100	16	0.457	0.028205	0.9673
500	31	2.417	0.137404	0.97726
1000	346	4.447	0.278778	0.94861

(c) FALCONN

Table 2.1: **Approximate nearest neighbor search method comparison.** Nearest neighbor search benchmarking results for **(a)** NMSLB, **(b)** ANNoY, and **(c)** FALCONN.

Since the number of unique scRNA-seq expression vectors we store is large, an exact solution obtained by a linear scan of the dataset for the nearest neighbor cell-types would be too slow. To enable efficient searches, we benchmarked three approximate nearest neighbor libraries: NMSLib [89], ANNoY[1], and FALCONN [90]. Benchmarking revealed that NMSLib was the fastest method (Table 2.1). NMSLib supports optimized implementations for cosine similarity and L2-distance based nearest neighbor retrieval. The indexing involves creation of hierarchical layers of proximity graphs. Hyperparameters for index building and query runtime were tuned to trade-off a high accuracy with reduced retrieval time. For NMSLib, these were: M = 10, efConstruction = 500, efSearch = 100, space =“cosinesimil”, method = “hnsw”, data_type = nmslib.DataType.DENSE_VECTOR, dtype = nmslib.DistType.FLOAT. Time taken to create the index: 2.6830639410000003 secs. Hyperparameters tuned for the ANNOY library were: number of trees = 50, search_k_var = 3000. Time taken to create the index: 1.3495307050000065 secs. For FALCONN, a routine to compute and set the hyperparameters at optimal values was

[1] `https://github.com/spotify/annoy`

used. This calibrates K (number of hash functions) and last cp dimensions. Time taken to create the index: 0.12065599400011706 secs.

2.2.6 Visualizing query results

We use the approximate nearest neighbors results to compute a similarity measure of each query cell to each ontology term. This is done by identifying the 100 nearest neighbors for each cell and determining the fraction of these matches that belong to a specific cell type. This generates a matrix of similarity measure entries for all query cells against all cell types which is presented as a hierarchical clustering heat-map (Figure 2.18A). All visualizations are based on this matrix.

For each query cell q_i and nearest neighbor n_k, we calculate the similarity score as:

$$s_{n_k}^{q_i} = 1/(1 + D(q_i, n_k))$$

Where $k \in [1, 100]$ and D is the euclidean distance function.

We sum (over the nearest neighbors) the similarities to a specific cell type ct to obtain a cumulative similarity score of the query to that cell-type:

$$S_{ct}^{q_i} = \sum_k (\quad s_{n_k}^{q_i} \quad * \quad \nVdash_{ct}(n_k) \quad)$$

Where

$$\nVdash_{ct}(n_k) = \begin{cases} 1 & \text{if } n_k \text{ is cell type } ct \\ 0 & \text{otherwise} \end{cases}$$

Thus, we obtain for each query a vector of similarity scores against all cell types. Finally, we normalize the vector such that the cell-type specific cumulative similarities sum to 1. Each normalized vector forms a row in the hierarchical heat-map.

We also perform further dimensionality reduction of the query via PCA to obtain a 2D nearest-neighbor style visualization against all cell types in the database and generate the ontology subgraph that matches the input cells. Users can click on any of the nodes in that graph to view the cell associated with it, DE genes related to this cell type, and their expression in the query cells.

In addition to matching cells based on the NN reduced values, we also provide users with the list of experiments in our database that contain cells that are most similar to a subset of uploaded cells the user selects. This provides another layer of analysis beyond the automated (though limited) ontology matching that is based on the cell types extracted for the nearest neighbors.

Finally, users can obtain summary information about cell type distribution in their uploaded cells and can find the set of cells matched to any of the cell types in our database.

2.3 Supporting Methods

2.3.1 Labeling of single cell experiments using Cell Ontology terms

We use all available labels in the Cell Ontology to assign each single cell experiment to a set of ontology terms. We consider all cell labels and synonyms, and adjust each label/synonym

before performing text matching against single cell experiment metadata. We strip the " cell" or " cells" suffixes, if present and if the cell descriptor consists of two or more characters without those suffixes. For example, "b cell" is used as-is, but "brain cell" is converted to "brain" before performing text searching. We build a regular expression for each adjusted label/synonym, replacing "{term}" with the appropriate label/synonym:

```
(-|\b)({term})(s?)(-|\b)
```

This allows every term, with or without a "s" suffix, to be matched with adjoining word boundaries or "–" characters.

We use the 'source,' 'cell type,' and 'cell ID' fields in GEO and ArrayExpress to assign a set of Cell Ontology terms to each single cell experiment. We collect all unique values in these fields across all single cell experiments, and test whether each value matches the regular expression for each ontology term name/synonym. We then reduce the set of terms for each experiment, removing any term that is an ancestor of another. For example, the cell label "Bone marrow cells" is assigned the following terms:

- CL:0000000 cell
- CL:0000137 osteocyte
- CL:0001035 bone cell
- CL:0002092 bone marrow cell
- UBERON:0001474 bone element
- UBERON:0002371 bone marrow
- UBERON:0002481 bone tissue

Removing redundant terms leaves only "CL:0000137 osteocyte" and "CL:0002092 bone marrow cell".

2.3.2 Triplet architectures trained with batch-hard loss

Following the same motivations as siamese networks, triplet networks also seek to learn an optimal embedding but do so by looking at three samples at a time instead of just two as in a siamese network. The triplet loss used by Schroff et al. considers an point (anchor), a second point of the same class as the anchor (positive), and a third point of a different class (negative) [67].

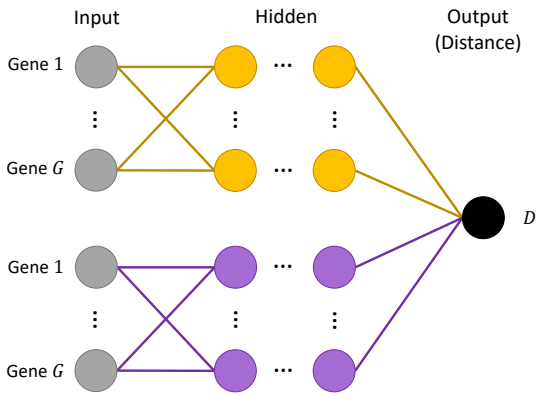


Figure 2.2: **Siamese neural network configuration.** Weights are shared between the yellow and the purple subnetworks. Data is fed in pairs with one item in the pair going through the yellow subnetwork, and the other through the purple subnetwork. Triplet networks operate in a similar manner, except that three points are fed through three subnetworks (again, shared weights).

$$\text{Triplet Loss} = \sum_{i=1}^{T} \left[(D_{a,p}^{i})^2 - (D_{a,n}^{i})^2 + \alpha \right]_+ \tag{2.4}$$

Where:

$\boldsymbol{T}$ is the set of all training examples (triplets of data points)

$\boldsymbol{D_{a,p}}, \boldsymbol{D_{a,n}}$ are the Euclidean distances computed by the network between the embeddings of the anchor point and the positive point, and the embeddings of the anchor point and the negative point, respectively

$\boldsymbol{\alpha}$ is a margin hyperparameter

When training a siamese or triplet network, it is very important to select pairs/triplets that are difficult enough so that the network learns and converges [67] . A straightforward approach to mine hard examples would involve periodically pausing training, embedding all data points using the current weights, calculating pairwise distances among all of these points in the embedded space, and selecting new examples based on this. This is computationally expensive and is prone to selecting pairs that are too difficult (outliers), so here we have opted to mine challenging pairs in an online fashion by selecting from points within a mini-batch [67, 91]. Specifically, we use Hermans et al.'s "batch hard" loss, shown in Equation 2.5 [91]. The loss is calculated over a mini-batch, which is constructed by first sampling P cell types (labels) and then sampling K cells of each type.

$$\text{Batch Hard Loss} = \sum_{i}^{P} \sum_{a}^{K} \left[m + \max_{p=1,\ldots,K} D\left(f(x_a^i), f(x_p^i)\right) - \min_{\substack{j=1,\ldots,P; j\neq i \\ n=1,\ldots,K}} D\left(f(x_a^i), f(x_n^j)\right) \right]_+ \tag{2.5}$$

Where:

$\boldsymbol{f}$ is the neural network (an embedding function)

$\boldsymbol{x_c^t}$ is the c^{th} cell of type t in the mini-batch

$\boldsymbol{D(a, b)}$ is the Euclidean distance between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$

$\boldsymbol{m}$ is a margin hyperparameter

For each anchor point a in the mini-batch, the $\max$ term in Equation 2.5 finds the hardest positive pair, while the $\min$ term finds the hardest negative pair.

2.3.3 Unsupervised neural network pretraining

While all of our neural embedding models are trained in a supervised fashion (for cell-type classification or embedding optimization), we also used unsupervised training to initialize the weights for these models, and this was shown to improve performance (Results, Figure 2.11). We have shown in prior work that using a denoising autoencoder (DAE) for unsupervised pretraining can improve performance [78]. Here we extend this to using stacks of DAEs, which are each individually trained in a process called "greedy layer-wise pretraining" to pretrain each layer of our neural network architectures [92, 93].

Here we illustrate the process with an example of how we would pretrain a neural network with an input layer, two hidden layers, and an output layer (Figure 2.3a). Figure 2.3b shows how the first hidden layer is pretrained. The first hidden layer becomes the hidden layer in a DAE, with a "reconstruction" layer of the same dimensionality as the input added on top. Gaussian noise is added to each sample and fed through this DAE, and the training objective is to minimize the mean squared error between the reconstructed output and the clean, uncorrupted samples. After training this DAE, the weights can be used to initialize the first hidden layer in our supervised neural network. This process is repeated for the other hidden layers, where all weights below the layer of interest are fixed, and the input is first fed forward through these fixed weights to get a new representation, and then the corrupting noise is added to this new representation for training the DAE of the current layer.

2.3.4 Cell-type similarity calculation

Rather than rely on distances based on the ontology graph structure, we computed cell type similarities with a data driven approach by using term co-occurrences in PubMed. For each pair of cell types in our ontology, we queried PubMed for articles which had both of the terms in the title or abstract texts.

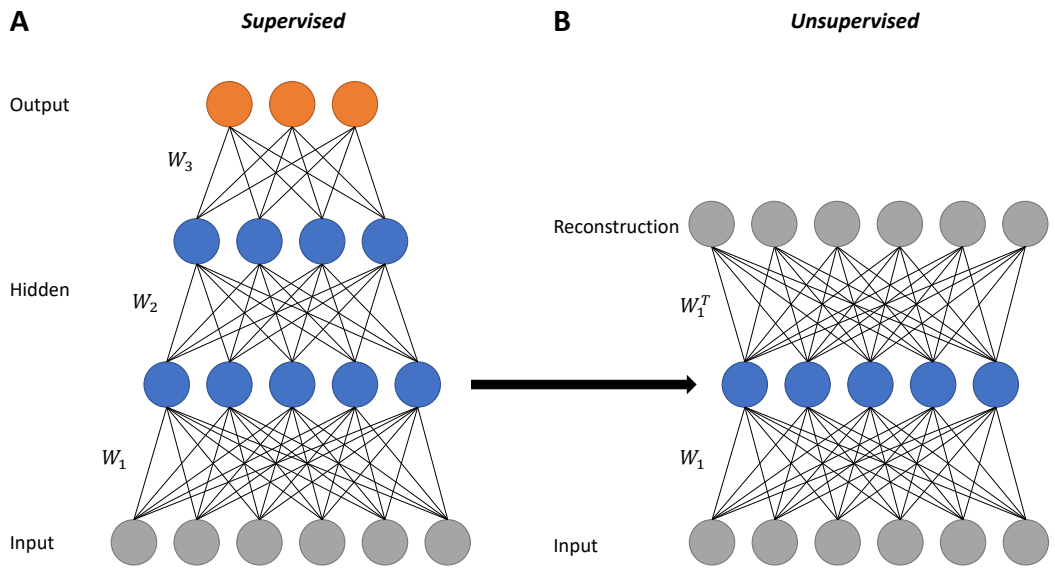


Figure 2.3: **Illustration of one stage of greedy layer-wise unsupervised pretraining of a neural network.** $W_1, W_2,$ and W_3 are weight matrices. **(A)** A supervised neural network with two hidden layers to be pretrained. **(B)** An unsupervised DAE that has the the supervised network's first hidden layer as its middle layer. This DAE is trained to minimize the mean squared error between its reconstruction and the original input data (prior to corruption with noise). After training this DAE, its weights can be used as initial values for the weights in the supervised network.

We then define S_i as the number of articles in which term i occurred in, and $P_{i,j}$ as the number of articles in which terms i and j co-occurred in. We additionally define a binary matrix $\boldsymbol{M}$ of shape (number of ontology terms, number of documents), where $M_{i,j} = 1$ if term j occurred in document i.

The similarity between two terms can then be computed as a cosine similarity:

$$\begin{aligned} D_{i,j} &= cos(\boldsymbol{M_i}, \boldsymbol{M_j}) \\ &= \frac{\boldsymbol{M_i} \cdot \boldsymbol{M_j}}{|\boldsymbol{M_i}||\boldsymbol{M_j}|} \\ &= \frac{P_{i,j}}{\sqrt{S_i S_j}} \end{aligned}$$

The above measure is not symmetric, and a symmetric version can be defined as:

$$D_{i,j} = \frac{P_{i,j}/S_i + P_{i,j}/S_j}{2}$$

For our neural embedding models that rely on computing distances between cell types (siamese architectures), we use the symmetric version of the cell type similarities. We use the non-symmetric version to evaluate all of our neural embedding models (*i.e.* to compute the MAFP values in our retrieval analysis).

2.3.5 Mean average flexible precision

We employ a modified version of mean average precision (MAP) called mean average flexible precision (MAFP) in order to evaluate the retrieval performance of our dimensionality reduction models. Traditional average precision operates on ranked lists (*e.g.* a list of nearest neighbor cells in a retrieval database) by calculating the precision at exact-match cutoffs in the list, and then taking the mean of these. This requirement is too restrictive for proper evaluation here, as a binary hit or miss criterion would not allow a list containing similar cell types to be valued higher than lists with less relevant cell types. Instead, average flexible precision is computed by summing the cell-type similarities between the query cell and each retrieved result and dividing by the length of the list at a particular depth, and finally taking the average of these values across all depths in the list.

2.3.6 Missing data imputation

Our uniform preprocessing pipeline requires the raw reads data for each study. For certain studies, this is not available, and only the RPKM values processed by the authors of the original study are present. We would still like to integrate this data into our database, which requires us to impute the values of the expression of missing genes.

We conduct missing value imputation with a k-nearest neighbors-based approach, similar to procedures in prior work [78, 94]. We ignore cells which have more than 20% of our gene set missing. After filtering those cells out, we impute the remaining values for each study independently. For each study, in order to compute nearest neighbor genes within the study we first fill the missing values with the median expression value in each cell. We then calculate the euclidean distance matrix of the genes in the study (where each gene is represented by a vector of its expression values in each cell). Finally, the value of each missing gene is imputed with the average expression of its 10 nearest neighbor genes within each cell.

2.3.7 Preprocessing strategies for differential expression analysis

Prior to conducting differential expression analysis, we filter out low-quality genes and cells. To filter out lowly-expressed genes, we first calculate the average reads of each gene within each cell type. We then calculate an overall average for each gene by taking the average of its cell-type specific average expressions. We then only keep genes that have an overall average value that is above a cutoff of 50. This resulted in the removal of 4180 out of 20499 genes.

To filter out cells, we remove cells with fewer than 1.8×10^3 detected genes, which is the default procedure in the tool we have chosen to use for DE analysis, SCDE [19]. This resulted in the removal 732 cells out of 23982 cells. We note that the 23982 cells are a subset of our total database, as we only conduct DE analysis for cell types with a minimum of 75 cells.

2.3.8 Meta-analysis of differential expression experiments

For cell types that have data from multiple studies, we conduct separate DE analysis for each of those studies, and then combine the results to obtain a single list of differentially expressed

genes with their p-values. To do this, we take the maximum FDR adjusted p-value for each gene across the studies as its final p-value. Our list of significant DE genes for a particular cell type is then taken as the set of genes with final p-values below a threshold of 0.05.

Additionally, we only keep the DE genes that had a consistent sign (positive or negative) $\log_2$ fold-change across the multiple DE analysis that were done for the particular cell type.

2.3.9 Gene Ontology enrichment analysis

We took the output of Algorithm 1 (a list of differentially expressed genes for each cell type) and filtered to keep the top 50 most significant (by FDR adjusted p-value) up-regulated genes. We then used this ordered set of genes as our query for GO enrichment analysis using g:Profiler [95]. We specified that g:Profiler should take the order of the query list into account (more significantly up-regulated genes are more informative) [96]. Other parameters of our analysis were organism="mmusculus", that Benjamini-Hochberg multiple testing correction be used, and that "GO: Biological Process" should be used as the term source. We also supplied a list of 16968 genes to be used as statistical background. These background genes are the subset of our 20499 genes that remain after we filtered out lowly expressed genes in the preprocessing step.

Cell Type	retina	
Term ID	UBERON:0000966	
# of experiments	2	
GO ID	**Name**	**p-value**
GO:0050877	nervous system process	7.63e-09
GO:0050953	sensory perception of light stimulus	7.63e-09
GO:0007601	visual perception	7.63e-09
GO:0007423	sensory organ development	8.69e-09
GO:0060041	retina development in camera-type eye	8.69e-09
GO:0007600	sensory perception	4.09e-08
GO:0001654	eye development	3.96e-07
GO:0003008	system process	6.67e-07
GO:0009584	detection of visible light	6.86e-07
GO:0043010	camera-type eye development	1.23e-06

(a) SCDE

Cell Type	retina	
Term ID	UBERON:0000966	
# of experiments	2	
GO ID	**Name**	**p-value**
GO:0045721	negative regulation of gluconeogenesis	3.03e-02
GO:0032379	positive regulation of intracellular lipid tr...	3.03e-02
GO:0032385	positive regulation of intracellular choleste...	3.03e-02
GO:0061179	negative regulation of insulin secretion invo...	3.03e-02
GO:1905600	regulation of receptor-mediated endocytosis i...	3.03e-02
GO:0010888	negative regulation of lipid storage	3.03e-02
GO:0032382	positive regulation of intracellular sterol t...	3.03e-02
GO:1905602	positive regulation of receptor-mediated endo...	3.03e-02
GO:0045475	locomotor rhythm	3.03e-02
GO:0090118	receptor-mediated endocytosis involved in cho...	3.34e-02

(b) limma-voom

Table 2.2: **Comparison of SCDE (a) and limma-voom (b) when used as "`DEModule`" in Algorithm 1**. We conducted the same process as for Table 2.5 to produce the above two panels: We used GO: Biological Process as the source term set, and used the top 50 up-regulated differentially expressed genes from our cell type-specific gene lists for retina. The top 10 GO terms are shown, sorted by FDR adjusted p-values.

2.3.10 PPI/TF architectures

Since our set of input genes is different, the architectures discussed in Lin et al. could not be used directly here. We used the same transcription factor data as Lin et al. [97]. For our protein-protein interaction data, we use a protein-protein interaction network which integrates known and predicted interactions from multiple databases [98].

2.3.11 Gene Ontology based architectures

We also tested another method to group genes based on the Gene Ontology (GO) [85]. GO is a directed acyclic graph (DAG) whose nodes are gene products (RNA or protein) and whose

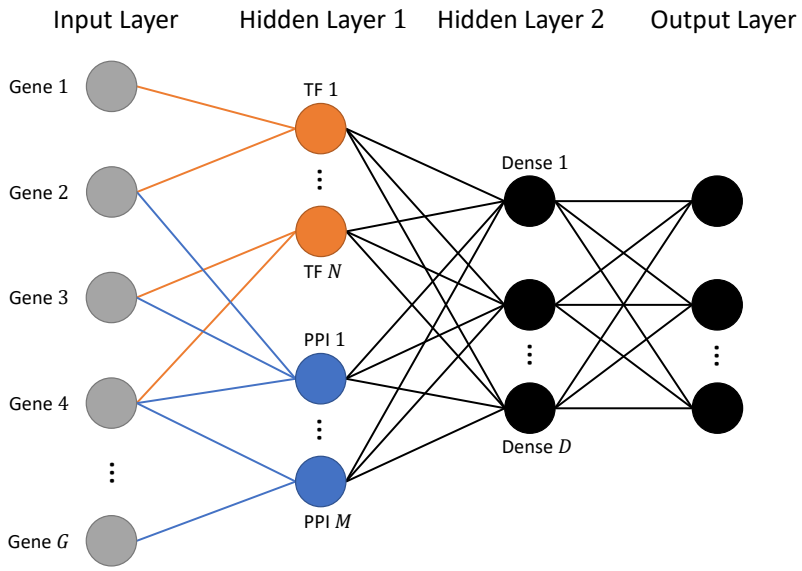


Figure 2.4: **PPI/TF neural network architecture** Our Protein-protein/protein-DNA (PPI/TF) based neural network architecture, showing sparsely connected layers based on prior biological knowledge.

edges are relationships between these gene products. Along with this graph, there exist curated annotations which associate genes from specific organisms with each node in the graph. Here we propose that GO's hierarchical structure will enable our models to recognize complicated relationships that most likely exist between genes.

To construct a hierarchical neural network architecture that mirrors the structure of the GO DAG, we first use the published GO annotations for *M. musculus* [99, 100] to associate each of our input genes with a node in GO. Specifically, we choose a particular depth d (distance from a root node) and first associate the genes with only the nodes at this depth in the GO graph. Multiple genes can be associated with the same node. Then we use this grouping of the input genes as the first hidden layer of a neural network. Each node in this hidden layer represents a GO node at depth d with its associated input genes connected to it. The nodes in the next hidden layer will be constructed from GO nodes that have depth d-1. A node in a layer is connected to a node in the layer above if a path exists between their corresponding nodes in the GO graph. We continue this process until the last hidden layer has the desired number of nodes (the size of our reduced dimension). The final result is the network depicted in Figure 2.5.

A key point is that the connections in the network are sparse: a node is only connected to a subset of the nodes in the layer below. This is analogous to the idea of convolutional neural networks (CNNs), in which neurons in a given layer are only connected to a subset of neurons in the layer below (the neuron's "receptive field") [101].

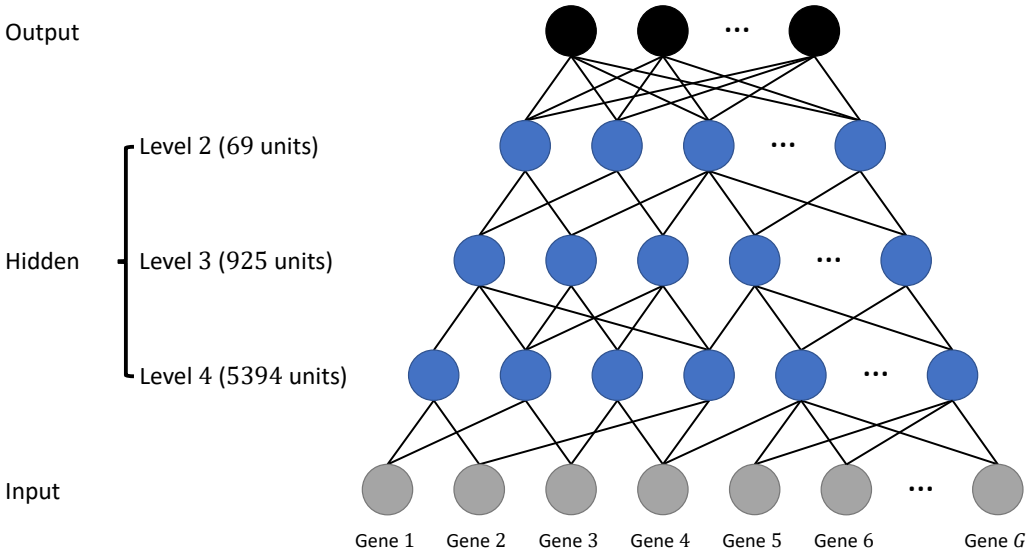


Figure 2.5: **Our Gene Ontology-based neural network architecture.** The connections between the input layer and the first hidden layer, as well a the connections between hidden layers, are sparse and follow connections in the GO DAG. The final hidden layer is fully connected (dense) to the output layer.

2.4 Results

2.4.1 Pipeline and web server overview

We developed a pipeline (Figure 2.6) for querying, downloading, aligning and quantifying scRNA-seq data. Following queries to the major repositories (Methods), we uniformly processed all datasets so that each was represented by the same set of genes and underwent the same normalization procedure (RPKM). We next attempt to assign each cell to a common ontology term using text analysis (Methods and Supporting Methods). This uniform processing allowed us to generate a combined dataset that represented expression experiments from more than 500 different scRNA-seq studies, representing 300 unique cell types, and totaling almost 150K expression profiles that passed our stringent filtering criteria for both expression quality and ontology assignment (Methods). We next used supervised neural network (NN) models to learn reduced dimension representations for each of the input profiles. We tested several different types of NNs including architectures that utilize prior biological knowledge [78] to reduce overfitting as well as architectures that directly learn a discriminatory reduced dimension profile (siamese [66] and triplet [67] architectures). Reduced dimension profiles for all data were then stored on a web server that allows users to perform queries to compare new scRNA-seq experiments to all data collected so far to determine cell types, identify similar experiments, and focus on key genes.

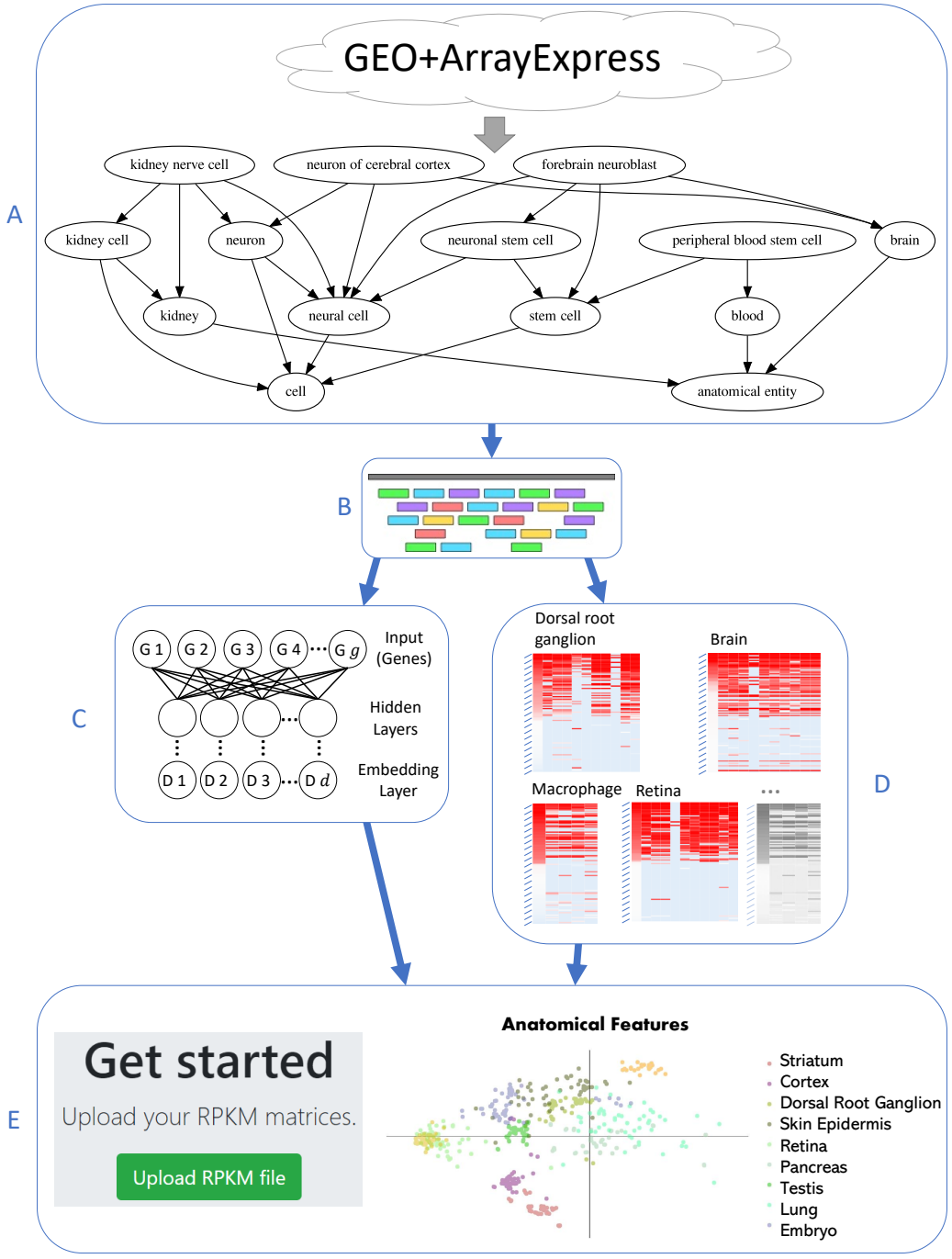


Figure 2.6: **Pipeline for large-scale, automated analysis of scRNA-seq data.** **(A)** Bi-weekly querying of GEO and ArrayExpress to download the latest data, followed by automatic label inference by mapping to the Cell Ontology. **(B)** Uniform alignment of all data sets using HISAT2, followed by quantification to obtain RPKM values. **(C)** Supervised dimensionality reduction using our neural embedding models. **(D)** Identification of cell type-specific gene lists using differential expression analysis. **(E)** Integration of data and methods into a publicly available web application.

2.4.2 Statistics for data processing and downloads

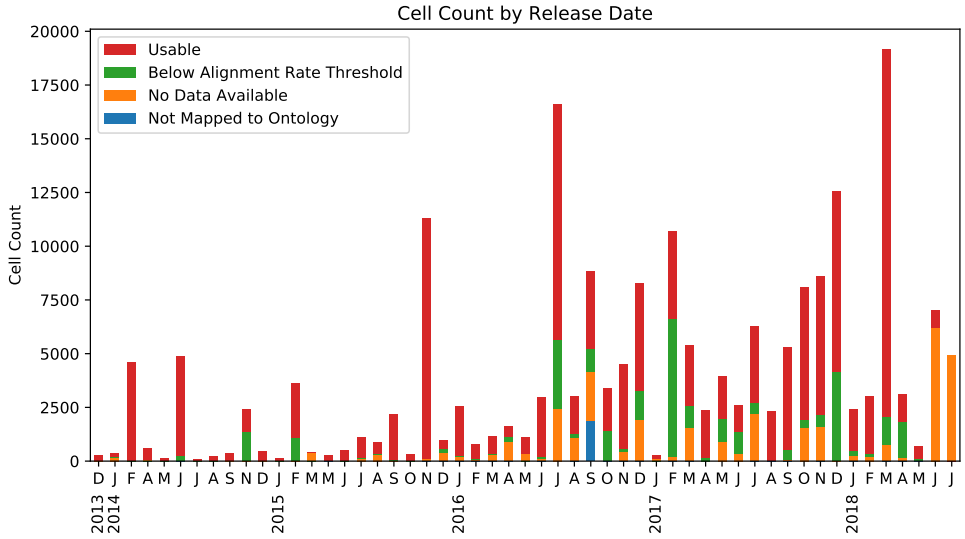


Figure 2.7: **Monthly cell count available on GEO and ArrayExpress.** Cell counts by month, separated into four categories: usable, below our alignment rate threshold, no raw or author-processed data available, and unmapped to ontology terms.

To retrieve all available scRNA-seq data we queried the two largest databases, GEO and ArrayExpress, for scRNA-seq data. Figure 2.8 presents screenshots of queries to the NCBI GEO and ArrayExpress databases similar to the ones we used here, though our queries utilized automated APIs instead of the web interfaces shown in these figures. Figure 2.9 and Figure 2.7 respectively show study and cell counts by month, with respect to the "release date" data provided by GEO and ArrayExpress. As can be seen, while cell counts increase over time, there is a lag in availability of raw data and author-processed supplementary data available through NCBI GEO and ArrayExpress systems. Since our pipeline is automated, we expect to be able to collect and analyze much more data over the next several months.

Our "mouse single-cell RNA-seq" query matched a total of 193,414 cells, of which 151,084 have raw data and 29,216 had only author-processed data. We used established ontologies to determine the cell type that was profiled (Methods) for each cell expression dataset we downloaded. Of the 2,481 unique descriptors we obtained for all cells, 1,909 map to at least one term in the cell ontology. Of the 5,010 distinct cell ontology terms (restricted to the `CL` and `UBERON` namespaces), 331 are assigned at least one cell expression profile.

Of the 151,084 cells for which raw data is available, 114,249 had alignment rates above our cutoff of 40%. Of these 114,249, we identified 2,473 raw data files that contained reads from multiple cells, but lacked any metadata that allowed us to assign reads to individual cells. This leaves 148,549 cells that are usable for building our scRNA-seq database.

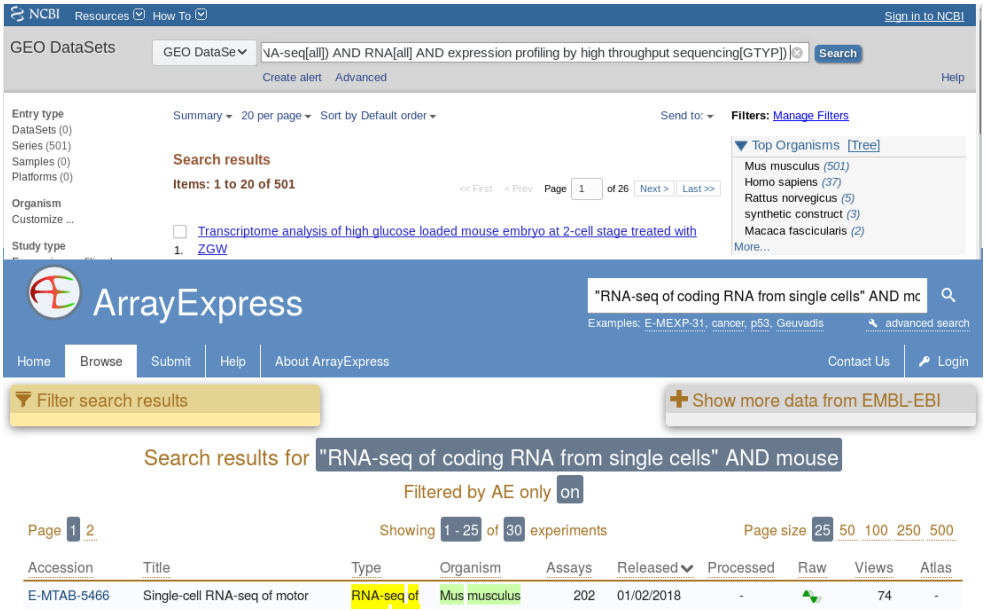


Figure 2.8: **Example data query.** Our queries to the NCBI GEO and ArrayExpress systems, selecting mouse single-cell RNA-seq data.

2.4.3 Neural networks for supervised dimensionality reduction

We trained several different types of supervised NNs. These included models with the label matching a cell type as the output (with the layer before last serving as the reduced dimension) [78] and models that directly optimize a discriminatory reduced dimension layer (using as input pairs or triplets of matched and unmatched profiles). See Figure 2.2, Figure 2.4, and Figure 2.5 for details. Some of the models utilized prior biological knowledge as part of the architecture to reduce overfitting (including protein-protein and protein-DNA interaction data and hierarchical GO assignments, termed PPI and TF respectively) while others did not (dense). We experimented with various hyperparameters (Methods), and all of our neural network models were trained for up to 100 epochs, with training terminated early if the model converged. All models converged sooner than the full 100 epochs (Figure 2.10). Performance on a held out validation set was assessed after each epoch during training. The final weights chosen for each model were those at the end of the epoch with the lowest validation loss out of the 100 epochs. For triplet networks, we also monitored the fraction of "active triplets" in each batch as a selection criterion. Most models trained in minutes (Table 2.3).

Prior work [78] has shown that NN weights can be used to identify relevant groupings of genes. Here we focused on the accuracy of the learned networks. After training each of our neural embedding models, we evaluated their performance using retrieval testing as described in Methods. The training set we used contained 36,473 cells while the test ("query") set consisted of 2,330 cells. Cells used for testing are completely disjoint from the set of cells that were used for training and come from different studies so that batch effects and other experimental artifacts do not affect performance and evaluation. The results of this retrieval testing for a selection of architecture types and cell types are shown in Figure 2.11. These models have been trained

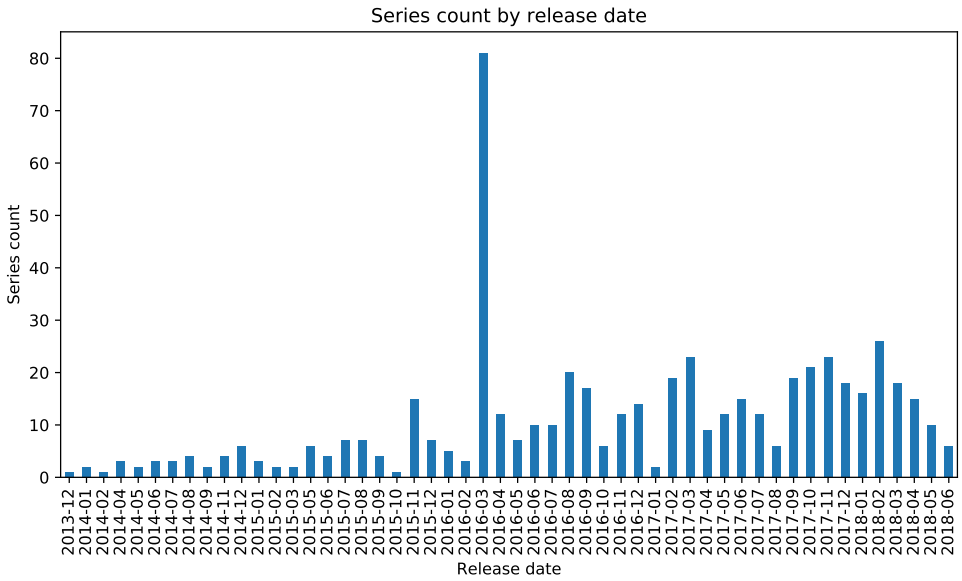


Figure 2.9: **Monthly study count available on GEO and ArrayExpress.** Number of studies released each month in GEO and ArrayExpress.

using the data that we processed ourselves in addition to data from studies that only had author-processed data available (these required missing-data imputation, see Methods) though similar results were obtained on models trained using only our own processed data (Figure 2.12). It is common to assess retrieval performance with mean average precision (MAP). Here, each value in the table represents the mean average flexible precision (MAFP, Supporting Methods), which allows for scores between 0 and 1 for matching a cell type to a similar or parent type (for example, a cortex cell matched to brain, Methods).

The best scoring model achieved a weighted average (accross all query cell types) MAFP of 0.576 which is very high when considering the fact that this was a 45-way classification problem (while our database contains over 300 cell types, only 45 types had independent data from multiple studies and these were used for the analysis discussed here). When restricting the analysis to the six cell types for which we had more than 1000 cells in our database, results for this model further improved to 0.623, which is significantly better than PCA 100 (paired t-test p-value: 1.253×10^{-41}). A paired t-test comparison on the full query set of cells is shown in Table 2.4. This top performing model (first row in Figure 2.11) employed a dense architecture (two-hidden-layer perceptron network). The next best model overall (based on the weighted average) was a similarly defined and performing dense architecture with three-hidden-layers (in Source Data), followed by a PPITF architecture with three hidden layers (the first one being sparsely connected to the input based on prior biological knowledge) trained as a triplet network (third row in Figure 2.11). We also see that for specific cell types, other neural networks perform better. Specifically, triplet networks perform best for neuron, embryo, and retina. Siamese architectures perform the best in the neural cell type. We also note that the best performing models were those that were pretrained with an unsupervised strategy (a full table with result from over 100 models, including those from models without pretraining are available on our web server). Finally, as

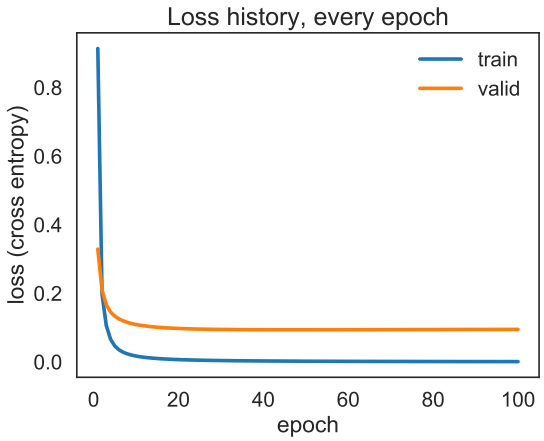


Figure 2.10: **Neural network training history.** Training and validation loss exhibiting convergence within 100 epochs for the "PT dense 1136 100" model in Figure 2.11. The weights saved are those of the first iteration after which validation loss no longer improves.

Model	# Parameters	Train set size	Mini-batch size	Training time
PT dense 1136 100	23,406,245	36,473	256	3 mins 54 secs
PT dense 1136 500 100	23,911,145	36,473	256	3 mins 51 secs
triplet PT ppitf 1136 500 100	3,116,269	72,000	72	31 mins 5 secs
triplet hierarchical GO	1,123,151	72,000	72	1 hr 57 mins
Siamese PT dense 1136 500 100	23,906,600	110,117	256	27 mins 13 secs
Siamese PT ppitf 1136 100	2,611,369	110,117	256	41 mins 18 secs

Table 2.3: **Neural network training times.** Number of parameters and time to train for each of the neural network model types from Figure 2.11. "PT" indicates that the model had been pretrained using the unsupervised strategy (Supporting Methods). Numbers after the model name indicate the hidden layer sizes. "hierarchical GO" refers to the GO-based architectures (Supporting Methods). All models were trained for 100 epochs on a machine with two Intel(R) Xeon(R) E5-2620 v3 @ 2.40GHz CPUs and an Nvidia GeForce GTX 1080 GPU.

is clear from the last two rows, supervised neural network embeddings consistently outperform PCA and the original data in the retrieval task. Even when we consider only those cell types that are rare in our training data ($< 1\%$ of total training population), we again see our neural embedding models outperforming PCA and original data (Figure 2.13).

2.4.4 Functional analysis of cell-type specific DE genes

We further used our ontology assignments to identify cell type-specific genes (Methods). The differential expression analysis we conducted is based on multiple studies for each cell type. Our procedure performs DE analysis for each study, for each cell type independently and then combines the results. This method ensures that resulting DE genes are not batch or lab related but rather real DE for the specific cell type. We used this method to identify cell type specific genes for 66 cell types. The number of significant (< 0.05 FDR adjusted p-value) DE genes for each cell type ranged from 31 for embryonic stem cells to 7576 for B cells. The full list of DE genes can be found on the supporting web server.

	# in query	91	45	162	81	45	250	678	
	# in database	734	481	421	523	250	340	1097	
Architecture type	**Model**	**hematop-oietic stem cell**	**osteocyte**	**neuron**	**neural**	**embryo**	**retina**	**skin epidermis**	**weighted average (higly rep. cell types)**
Cell type classifier	PT dense 1136 100	0.683	**1.000**	0.931	0.574	0.145	0.909	**0.621**	**0.623**
	PT dense 1136 500 100	**0.798**	0.977	0.905	0.930	0.091	0.909	0.573	0.586
Triplet	PT (frac active) ppitf 1136 500 100	0.382	0.951	0.955	0.644	0.493	0.941	0.598	0.594
	PT (val loss) ppitf 1136 500 100	0.308	0.961	**0.963**	0.516	**0.640**	**0.942**	0.552	0.550
Siamese	PT dense 1136 500 100	0.510	0.981	0.937	**0.996**	0.258	0.896	0.426	0.483
	PT ppitf 1136 100	0.113	0.767	0.385	0.952	0.082	0.870	0.159	0.224
N/A	PCA 100	0.696	0.946	0.863	0.889	0.167	0.901	0.488	0.494
	Original data	0.019	0.976	0.539	0.823	0.146	0.797	0.062	0.109

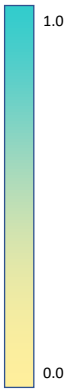


Figure 2.11: **Neural embedding retrieval testing results.** Retrieval testing results of various architectures, as well as PCA and the original (unreduced) expression data. Scores are MAFP (Mean average flexible precision) values (Supporting Methods). "PT" indicates that the model had been pretrained using the unsupervised strategy (Supporting Methods). "Ppitf" refers to architectures based on protein-protien and protein-DNA interactions (Supporting methods, Figure 2.4). Numbers after the model name indicate the hidden layer sizes. For example, "dense 1136 500 100" is an architecture with three hidden layers. The metrics in parenthesis for the triplet architectures indicate the metric used to select the best weights over the training epochs. For example, "frac active" indicates that the weights chosen for that model were the ones that had the lowest fraction of active triplets in each mini-batch. We highlight the best performing model in each cell type with a bolded value. We can see that in every column, the best model is always one of our neural embedding models. The final column shows the weighted average score over those cell types, where the weights are the number of such cells in the query set. The best neural embedding model (PT dense 1136 100, top row) outperformed PCA 100 (0.623 vs 0.494) with a p-value of 1.253×10^{-41} based on two-tailed t-test. Source data are provided as a Source Data file.

To determine the accuracy of the DE genes and to showcase the effectiveness of the automated processing and ontology assignments, we performed Gene Ontology (GO) enrichment analysis on the set of DE genes for each cell type (Supporting Methods). Results for a number of the cell types are presented in Table 2.5. As can be seen, even though each of the cell type data we used combined multiple studies from different labs, the categories identified for all of the cell types are highly specific indicating that the automated cell type assignment and processing were able to correctly group related experiments.

As an example, the top three enriched terms for "retina" are "nervous system process" ($p = 7.63 \times 10^{-9}$), "sensory perception of light stimulus" ($p = 7.63 \times 10^{-9}$), and "visual perception" ($p = 7.63 \times 10^{-9}$). Cells of dorsal root ganglion are sensory neurons as reflected in Table 2.5 with terms such as "sensory perception of pain" ($p = 3.54 \times 10^{-6}$) and "detection of temperature stimulus" ($p = 9.86 \times 10^{-6}$). For "T cell," nine of the top ten terms are related to immune response and specific aspects of the T cell-mediated immune system. Complete results are available from the supporting web server.

2.4.5 Mouse brain case study

To test the application of our pipeline we used it to study a recent scRNA-seq neurodegeneration dataset that was not included in our database [59]. This study profiled 2208 microglial cells extracted from the hippocampus of the CK-p25 mouse model of severe neurodegeneration. In the CK-p25 mouse model, the expression of p25, a calpain cleaved kinase activator, is induced

t-Test: Paired Two Sample for Means

	PT dense 1136 100	*PCA 100*
Mean	0.576379804	0.5440947
Variance	0.207874104	0.18939181
Observations	2330	2330
Pearson Correlation	0.845531936	
Hypothesized Mean Difference	0	
df	2329	
t Stat	6.272445056	
P(T<=t) one-tail	2.11189E-10	
t Critical one-tail	1.645508148	
P(T<=t) two-tail	4.22378E-10	
t Critical two-tail	1.960983084	

Table 2.4: **Paired t-test comparing PT dense 1136 to PCA 100.** The tested dataset is the full set of 2330 cells in our query set. Each observation is the Average Flexible Precision score on a query cell. We see that our best neural embedding model outperformed PCA 100 ($p = 4.224 \times 10^{-10}$.)

and results in Alzheimer's disease-like pathology. In the original study, the microglial cells were extracted from control and CK-p25 mice from four time points: before p25 induction (three months old), and one, two, and six weeks after induction (three months 1 week, three months 2 weeks old, and 4 months 2 weeks old, respectively). The goal of the study was to compare the response of microglial cells to determine distinct molecular sub-types, uncover disease-stage specific states, and further characterize the heterogeneity in microglial response. We used the raw read data to perform alignment and quantification (Methods) resulting in 1990 cells that passed our alignment thresholds and were used for the analysis that follows.

We used this data to test several aspects of the method, pipeline, and website. We performed a complete analysis of the roughly 2000 cells using the scQuery webserver (see below) which only took a few minutes. We first compared the supervised (using NN) and unsupervised dimensionality reduction. The cells were transformed to a lower dimensional space using the "PT dense 1136 100" NN followed by t-SNE to get them to 2 dimensions. We compared this to a completely unsupervised dimensionality reduction, as was done in the original paper. Figure 2.14 presents the results of this analysis. We observe that the supervised method is able to better account for the differences between the two populations of healthy and disease cells.

Cell type classifications

We next performed retrieval analysis by using the mouse brain cells as queries against our large database of labeled cells. We classified each query cell based on the most common label in its 100 nearest neighbors in the database (Methods). The results of this cell type classification can be seen in Figure 2.15.

We next compared the cell assignments for 3 different groups. An early time point (three months old) in which the healthy and disease mouse models are not expected to diverge, a later time point (four months 2 weeks old) in which differences are expected to be pronounced, and all data collected from the healthy and disease data. As can be seen in Figure 2.15a, our assign-

Cell Type	Experiments	GO ID	Name	p-value
retina	2	GO:0050877	nervous system process	7.63e-09
		GO:0050953	sensory perception of light stimulus	7.63e-09
		GO:0007601	visual perception	7.63e-09
		GO:0007423	sensory organ development	8.69e-09
		GO:0060041	retina development in camera-type eye	8.69e-09
		GO:0007600	sensory perception	4.09e-08
		GO:0001654	eye development	3.96e-07
		GO:0003008	system process	6.67e-07
		GO:0009584	detection of visible light	6.86e-07
		GO:0043010	camera-type eye development	1.23e-06
dorsal root ganglion	3	GO:0019233	sensory perception of pain	3.54e-06
		GO:0016048	detection of temperature stimulus	9.86e-06
		GO:0031175	neuron projection development	2.41e-04
		GO:0050965	detection of temperature stimulus involved in...	2.51e-04
		GO:0050877	nervous system process	2.51e-04
		GO:0050961	detection of temperature stimulus involved in...	2.51e-04
		GO:0048666	neuron development	3.49e-04
		GO:0009581	detection of external stimulus	3.79e-04
		GO:0009582	detection of abiotic stimulus	3.79e-04
		GO:0007600	sensory perception	4.31e-04
skin epidermis	2	GO:0009888	tissue development	2.99e-06
		GO:0043588	skin development	2.99e-06
		GO:0042303	molting cycle	9.57e-05
		GO:0042633	hair cycle	9.57e-05
		GO:0030177	positive regulation of Wnt signaling pathway	1.16e-04
		GO:0042476	odontogenesis	1.16e-04
		GO:0009653	anatomical structure morphogenesis	1.87e-04
		GO:0022404	molting cycle process	2.37e-04
		GO:0030111	regulation of Wnt signaling pathway	2.37e-04
		GO:0022405	hair cycle process	2.37e-04
T cell	4	GO:0002376	immune system process	4.68e-10
		GO:0006955	immune response	3.86e-08
		GO:0046649	lymphocyte activation	1.25e-07
		GO:0045321	leukocyte activation	1.25e-07
		GO:0002682	regulation of immune system process	1.25e-07
		GO:0042110	T cell activation	2.75e-07
		GO:0050776	regulation of immune response	2.84e-07
		GO:0002684	positive regulation of immune system process	3.82e-07
		GO:0001775	cell activation	4.54e-07
		GO:0002252	immune effector process	6.38e-07

Table 2.5: **Results of GO enrichment analysis**. Top 10 enriched "GO: Biological Process" terms using top 50 up-regulated DE genes for each cell type (FDR adjusted p-values). "Experiments" refers to the number of studies used to compute differentially expressed genes (Methods).

	# in query	160	45	162	81	45	250	678	
	# in database	645	481	421	523	250	340	1097	
Architecture type	**Model**	**hematop-oietic stem cell**	**osteocyte**	**neuron**	**neural**	**embryo**	**retina**	**skin epidermis**	**weighted average (higly rep. cell types)**
Cell type classifier	PT dense 1136 100	0.241	0.984	0.942	0.811	0.246	0.903	0.629	**0.649**
	PT dense 1136 500 100	0.196	0.990	0.935	0.526	0.242	0.914	0.480	0.534
Siamese	PT dense 1136 500 100	0.125	0.883	0.846	0.666	0.233	0.895	0.499	0.543
	PT ppitf 1136 100	0.053	0.695	0.242	0.672	0.267	0.828	0.307	0.335
N/A	PCA 100	0.369	0.959	0.701	0.747	0.184	0.904	0.485	0.504
	Original data	0.130	0.974	0.539	0.824	0.146	0.797	0.062	0.122

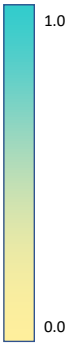


Figure 2.12: **Retrieval testing results for reprocessed data only.** Retrieval testing results of various architectures, as well as PCA and the original (unreduced) expression data, trained on only the data that we processed ourselves. "PT" indicates that the model had been pretrained using the unsupervised strategy (Supporting Methods). Numbers after the model name indicate the hidden layer sizes. For example, "dense 1136 500 100" is an architecture with three hidden layers. The final column is the weighted average score over those cell types with at least 1000 cells in the database (some not shown in table), and the weights are the number of such cells in the query set.

ment indeed reflects these stages with a much more significant difference for the later time point compared to the earlier one, with the entire dataset (which includes more intermediate points) in the middle. Focusing on the later time point, Figure 2.15b shows the cell type distribution. Several of the cell types identified by the method correspond to brain cells (brain, cortex, meningeal cluster) while others are related to blood and immune response (bone marrow, macrophage). The most common classification among the query cells was "fibroblast." Recent studies have shown that fibroblast-like cells are common in the brain [102], and that brain fibroblast cells can express neuronal markers [103].

As can be seen, the main difference observed between the disease and healthy mice is the increase in the immune system related types of "bone marrow" and "macrophage" cells in the disease model. We believe that while the method labeled these cells as macrophages, they are actually microglia cells which were indeed the cells the authors tried to isolate. To confirm this, we analyzed sets of marker genes that are distinct for macrophages and for microglia [104]. Figure 2.16 shows that indeed, for the cells identified by the method the expressed markers are primarily microglia markers.

The main reason that the method identified them as macrophages is the lack of training data for microglial cells in our database (our train data of high-confidence cell types contains no microglial cells and 273 macrophage cells; our full database contains only 44 microglial cells compared to 603 macrophage cells). Still, the result that disease samples contain more immune cells, which is only based on our analysis of scRNA-seq data (without using any known immune markers) indicates that as more scRNA-seq studies are performed and entered into our database, the accuracy of the results would increase.

Comparison to macrophage differentially expressed genes

We further characterized the gene expression within these microglial cells by comparing the gene expression in the query cells to our cell-type-specific DE genes (Methods). We uploaded the RPKM gene expression values for 256 microglial cells from late-stage neurodegenerative

	# in query	9	42	37	45	250	16	
	# in database	207	273	50	250	340	161	
Architecture type	**Model**	**fibroblast**	**macrophage**	**blastoderm**	**embryo**	**retina**	**liver**	**weighted average (lowly rep. cell types)**
Cell type classifier	PT dense 1136 100	**1.000**	**1.000**	**0.990**	0.145	0.909	0.129	0.468
	PT dense 1136 500 100	**1.000**	**1.000**	0.973	0.091	0.909	0.248	0.460
Triplet	PT (frac active) ppitf 1136 500 100	**1.000**	0.980	0.541	0.493	0.941	0.385	0.508
	PT (val loss) ppitf 1136 500 100	**1.000**	0.998	0.478	**0.640**	**0.942**	**0.481**	0.512
Siamese	PT dense 1136 500 100	0.991	0.964	0.660	0.258	0.896	0.110	**0.791**
	PT ppitf 1136 100	**1.000**	0.874	0.730	0.082	0.870	0.158	0.419
N/A	PCA 100	0.832	0.993	0.660	0.167	0.901	0.097	0.454
	Original data	0.279	0.826	0.262	0.146	0.797	0.200	0.368

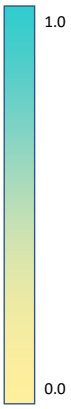


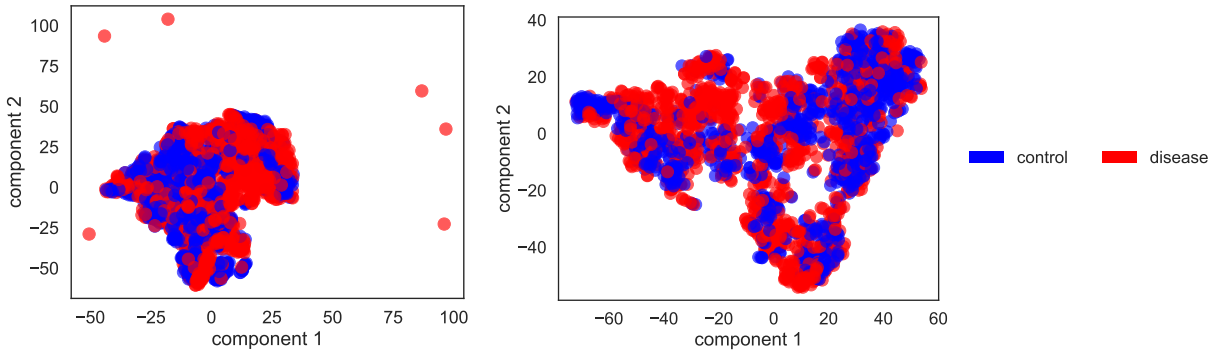
Figure 2.13: **Retrieval testing results for lowly represented cell types in our database.** Results presented in a similar manner as Figure 2.11, except that here we only show the results of those cell types with less than 1% of total training set population (36,473 cells). The final column is the weighted average score over all such "lowly represented" cell types (some not shown in table due to space), and the weights are the number of such cells in the query set.

mice (6 weeks after p25 induction) to our web server [59]. Given the hits from the retrieval database, we then selected a cluster of these that were enriched for "macrophage", and then viewed the expression (logRPKM) of our previously calculated cell type-specific marker genes as a heatmap for a subset of these cells. A cropped screenshot of the interactive heatmap provided by the web server is shown in Figure 2.17, where we see that there is rough agreement in up and down regulation trends of the set of macrophage-specific DE genes we identified between the cells in the database (leftmost column) and the query (microglial) cells. We would expect to see this expression pattern for these genes in these cells, as microglia are distinct from macrophages, but are related in function.

Among the up-regulated genes selected for "macrophage", we see genes that are also found to be up-regulated in late-response clusters including a microglial marker (*Csf1r*), a gene belonging to the chemokine superfamily of proteins (*Ccl4*), a major histocompatability complex (MHC) class II gene (*Cd74*), and other genes related to immune response (*Lilrb4a, Lgals3*), [59] (magenta highlights in Figure 2.17). We also see that a different microglial marker (*C1qa*) [105] is up-regulated. In addition to re-identifying genes of interest from the original study, our method is also able to highlight additional genes that are biologically relevant. These are highlighted in yellow in Figure 2.17 and include more chemokines (*Ccl2, Ccl6, Ccl9*), another MHC class II gene (*H2-ab1*), and other cell surface antigens (*Cd14, Cd48, Cd52, Cd53*).

2.4.6 Query and retrieval

To enable users to compare new scRNA-seq data to the data we processed, and to determine the composition of cell types in such samples we developed a web application. After processing their data (Methods) users can uploaded it to the server. Next, uploaded data is compared to all studies stored in the database. For this, we use approximate nearest neighbor approaches to match these



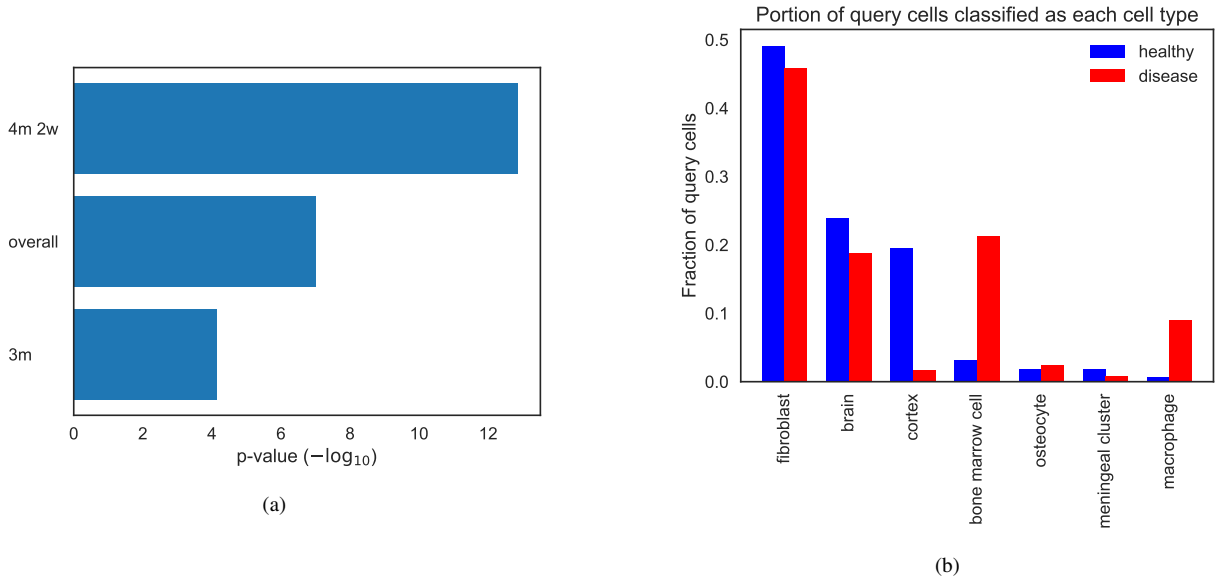
(a) original data

(b) PT dense 1136 100

Figure 2.14: **t-SNE visualizations of mouse brain query cells.** (a) Two component t-distributed stochastic neighbor embedding (t-SNE) of the query cells from their original 20,499 dimensions (genes), colored by disease status. (b) Two component t-SNE of the query cells from the 100 dimensional neural embedding via the "PT dense 1136 100" model.

to the data we have pre-processed. Embedding in the reduced dimension representation and the fast matching queries of thousands of cells against hundreds of thousands of database cells can be performed in minutes.

Figure 2.18 presents an example of a partial analysis of newly uploaded data by the web server. The web server clusters cells based on their matched types (Figure 2.18A), plots their 2D embedding with respect to all other cell types in the database (Figure 2.18B), highlights the top represented ontology terms in the uploaded cells (Figure 2.18C), and provides additional information about specific studies that it matched to the uploaded data (Figure 2.18D).



(a) (b)

Figure 2.15: **Analysis of mouse neurodegeneration dataset, late response cells. (a)** p-values of the difference in cell type classification distributions (healthy vs disease cells) for different time points. Three months was the initial time point in the study, and four months two weeks was the last time point. "Overall" is the pool of all 1990 cells. The p-values are from conducting Fisher's exact test (for "overall", the p-value was simulated based on 1e+07 replicates). **(b)** Classification distribution for late stage cells (four months two weeks), showing an increase in immune-related cell types in the disease cell population.

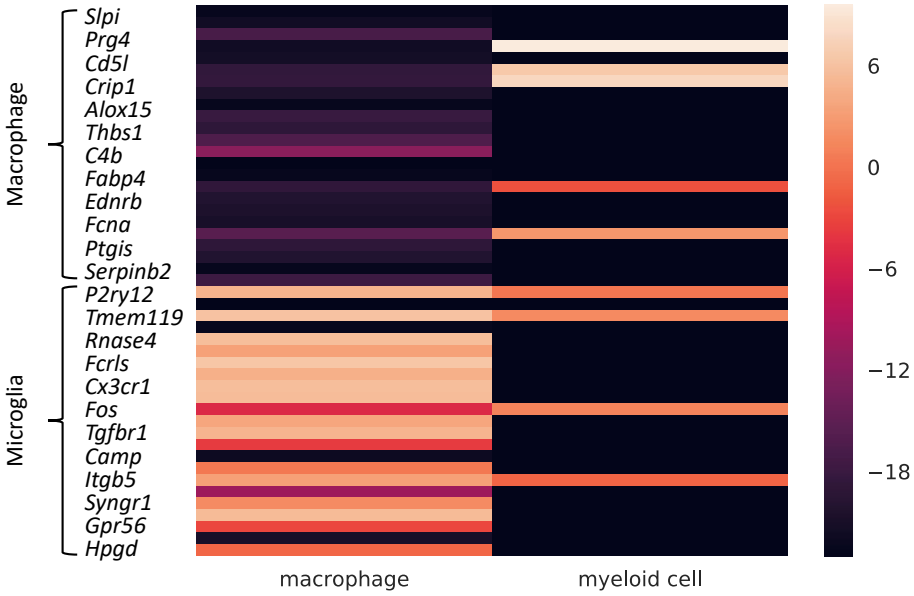


Figure 2.16: **Marker gene expression in mouse brain immune cells.** Expression ($\log_2$(RPKM)) of marker genes [104] for macrophages and microglial cells for all cells we classified as "macrophage" or "myeloid" cell.

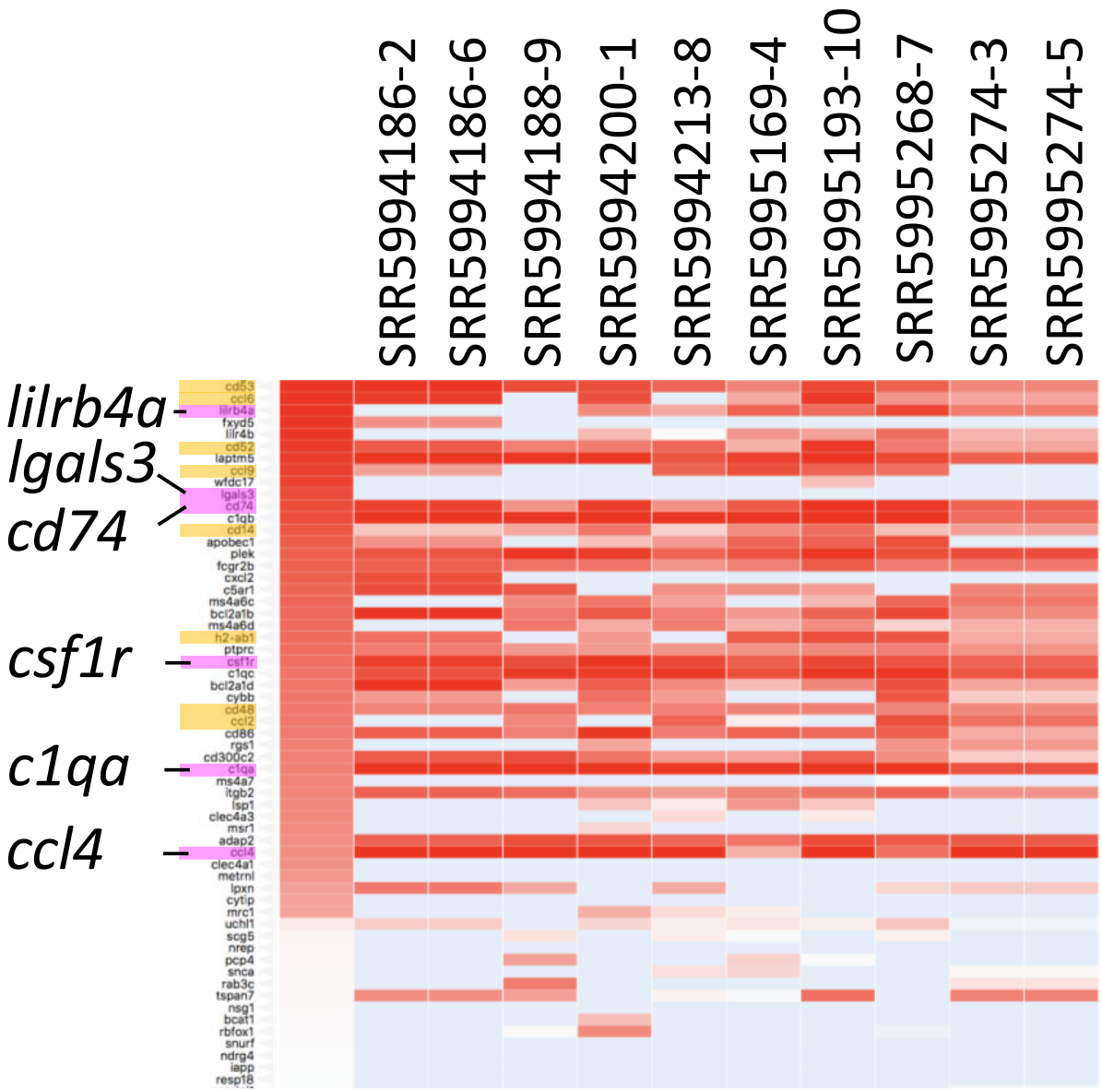
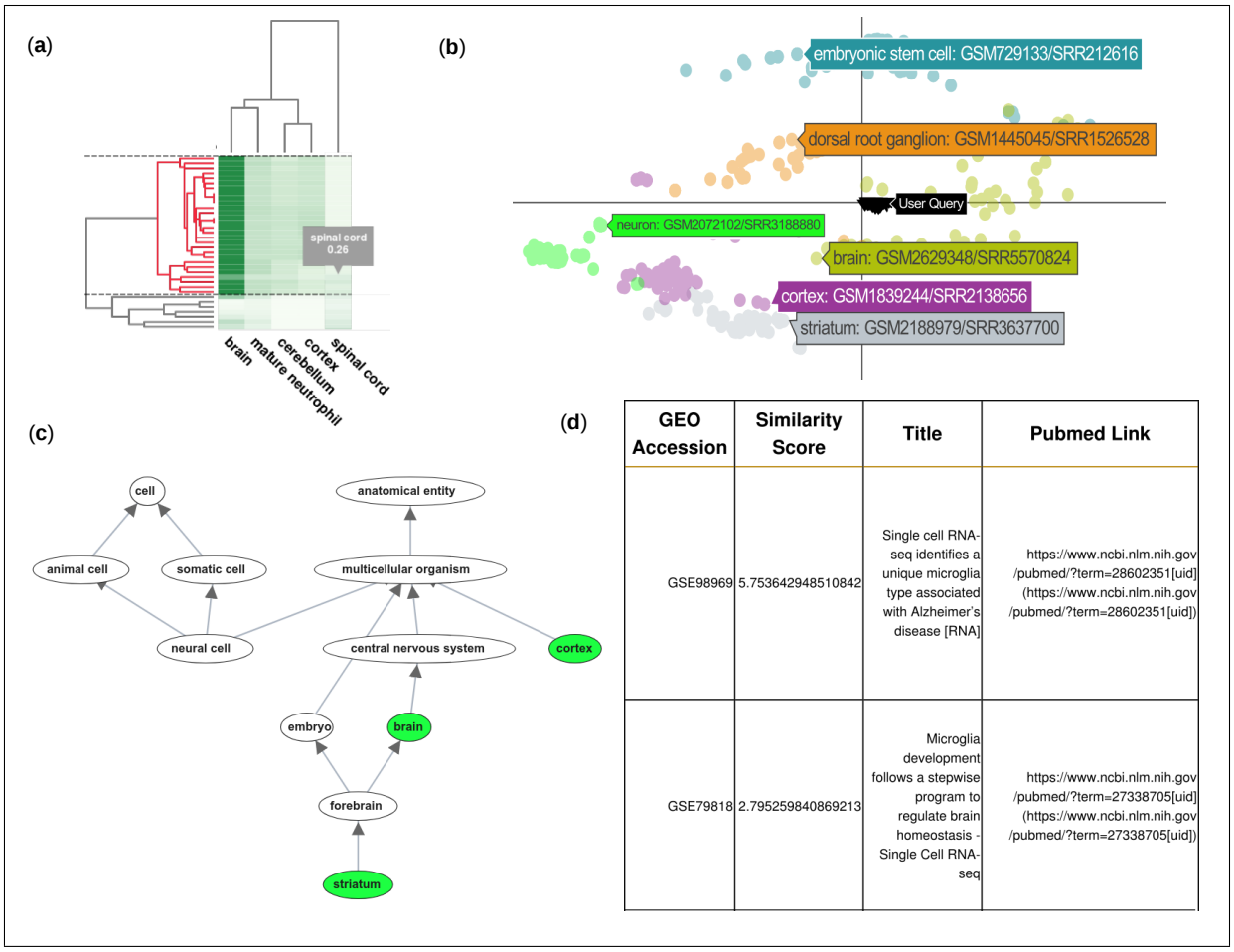


Figure 2.17: **Gene expression within mouse neurodegeneration dataset, late response cells.** This figure is a crop from a screenshot of an scQuery session. logRPKM expression of the macrophage-specific genes (Methods) within a cluster of late-stage (6 weeks after p25 induction) neurodegenerative cells enriched for "macrophage" labeling by our retrieval server. The leftmost column is the average logRPKM among our database macrophage cells. Genes highlighted in magenta were also found to be up-regulated in the original study [59]. Genes highlighted in yellow are additional genes related to immune response that are up-regulated in the query cells. Genes highlighted in yellow in order of top to bottom: *Cd53, Ccl6, Cd52, Ccl9, Cd14, H2-ab1, Cd48, Ccl2*



GEO Accession	Similarity Score	Title	Pubmed Link
GSE98969	5.753642948510842	Single cell RNA-seq identifies a unique microglia type associated with Alzheimer's disease [RNA]	https://www.ncbi.nlm.nih.gov/pubmed/?term=28602351[uid] (https://www.ncbi.nlm.nih.gov/pubmed/?term=28602351[uid])
GSE79818	2.795259840869213	Microglia development follows a stepwise program to regulate brain homeostasis - Single Cell RNA-seq	https://www.ncbi.nlm.nih.gov/pubmed/?term=27338705[uid] (https://www.ncbi.nlm.nih.gov/pubmed/?term=27338705[uid])

Figure 2.18: **The scQuery web server**. **(A)** Cluster heatmap of the nearest neighbor results for a query consisting of 40 "brain" and 10 "spinal cord" cells. The horizontal dashed lines demarcate the currently selected cluster and the corresponding dendrogram sub-cluster is highlighted in red. **(B)** 2D scatter plot of the selected sub-cluster (shown as inverted triangles and tagged as "User Query") along with a handful of other cell-types whose tags show cell-type information and GEO submission ids for a single cell from each cluster. **(C)** Ontology DAG depicting the retrieved cell-types in green while the nodes in gray visualize the path to the root nodes (which reflects paths of cellular differentiation as well as other biological relationships). **(D)** Metadata table for the retrieved hits displaying the GEO accession id, similarity score, publication titles, and their respective pubmed links.

2.5 Discussion

We developed a computational pipeline to process all scRNA-seq data deposited in public repositories. We have identified over 500 studies of scRNA-seq data. For each we attempted to download the raw data and to assign each cell to a restricted ontology of cell types. For cells for which this information existed we uniformly processed all reads, ran them through a supervised dimensionality reduction method based on NNs and created a database of cell type profiles. Using the scQuery server, users can upload new data, process it in the same way, and then compare it to all collected scRNA-seq data. We view the main goal of scQuery as a way to assist experimentalists who are performing exploratory analysis of new scRNA-Seq studies or re-analyzing existing studies. scQuery addresses several issues in the analysis of new scRNA-Seq data by automating the annotation process based on prior deposited samples.

In addition to cell assignments, the web server allows users to view the metadata on which the assignment is based, view the ontology terms that are enriched for their data and the distribution it predicts, and compare the expression of genes in the new profiled cells to genes identified as DE for the various cell types. The web server also clusters the cells and plots a 2D dimensionality reduction plot to compare the expression of the users' cells with all prior cell types it stores. Applying the method to analyze recent neurodegeneration data led to the identification of significant differences between cell distributions of healthy and diseased mouse models, with the largest observed difference being the set of immune related cell types that are more prevalent in the diseased mouse. Our method also revealed additional up-regulated immune-related genes in the late-stage neurodegenerative cells that were classified as "macrophage" cells.

While the pipeline was able to process several of the datasets we identified on public repositories, not all of them could be analyzed. Specifically, many studies lacked raw scRNA-seq reads, and thus could not be processed via our uniform expression quantification pipeline. Though we were able to find author-processed expression data for many studies, usage of this data is complicated by different gene selections, data format differences (*e.g.* RPKM vs. FPKM vs. TPM vs. read counts), and more. Additionally, several studies profiled thousands of cells but published far fewer raw data files, with each raw data file containing reads from hundreds or thousands of cells but no metadata that allows each read to be assigned to a unique cell.

In addition to issues with processing data that has already been profiled and deposited, we observed that cell type distribution in our database is still very skewed. While some cell types are very well represented ("bone marrow cell": 6,283 cells, "dendritic cell": 4,126 cells, "embryonic stem cell": 2,963 cells) others are either completely missing or were only represented with very few samples ("leukocyte": 12 cells, "B cell": 22 cells, "microglial cell": 44 cells, "cardiac muscle cell": 72 cells). Such skewed distribution can cause challenges to our method leading to cells being assigned to similar, but not the correct, types. These are still the early days of scRNA-seq analysis with several public and private efforts to characterize cell types more comprehensively.

Currently, scQuery does not account for the case when query cell types are not contained in our database. We note that simply collecting more data may not fix this issue, as there may always be the case of novel cell types appearing in queries, which are impossible to cover with publicly available data. A potential extension to the methods in scQuery to address these out-of-database cell types is to use the probabilistic output of the neural network classifiers instead of solely using the embedding. Specifically, many of our neural network embedding models

(including the best performing model overall) are models trained to classify input query cells. In the work we have presented here, we use the classification task as a proxy to learn high-quality, compact embeddings to enable efficient comparisons with query data in this space. However, because these classifiers are probabilistic models (the softmax non-linear activation function on the output nodes gives a probability from [0., 1.] of each label for an input cell), they can be used to convey the uncertainty of the prediction to users. For example, if for an input cell all of the output predicted probabilities are small (e.g. all are around $1/C$ where C is the number of possible cell types in the database), then this indicates that the model is not confident about the identity of this input cell. In other words, the input cell is not very similar to any of the cell types we have in our database. A trivial way to implement this in scQuery would be to feed forward each query cell all the way to the prediction stage of the neural network (rather than just stopping at the embedding layer) and flag those input cells which do not have a minimum threshold of predicted probability for any classes. This threshold could be tuned with training data by completely holding out a set of cell types and testing if the model can correctly flag these as "out-of-database". Another more rigorous approach may be to quantify the Shannon entropy of the output cell type distribution from the model. A more confident prediction distribution would be "peaky", and thus have low entropy, whereas a low-confidence prediction distribution would be more flat and have high entropy. We note that it the predictions from a deep neural network might not be accurate estimates of class probabilities, as they tend to overfit. In this case we should assess the calibration of the network by plotting the calibration curve, which plots the true fraction of positives against the predicted probability for samples in a predicted probability bin, for a particular class. If this curve doesn't lie near the $y = x$ diagonal curve, our model's probabilistic outputs aren't accurate estimates of class probability. In this case, we can then train a separate model which may be better suited to estimate class probabilities (e.g. logistic regression because it directly optimizes the log loss), or fit a regressor to calibrate our neural network model's outputs.

Our dataset retrieval and processing pipeline (including cell type assignments) is fully automated and we expect that once more experiments are available they would be added to the database and server. We believe that as more data accumulates the accuracy of scQuery would increase making it the tool of choice for cell type assignment and analysis.

Chapter 3

Integrating diverse scRNA-seq data batches for combined analysis.

The question of how to integrate scRNA-seq data has become increasingly relevant as the size and number of datasets has increased. This problem is also known as "dataset alignment", "dataset harmonization", or "batch correction." In Chapter 2, we did not explicitly address this issue when we were training our models on data from many diverse datasets. In this chapter we will discuss two contributions that specifically address this problem.

We look at the scRNA-seq alignment problem from both perspectives: supervised (section 3.1) and unsupervised (section 3.2). In both cases, we have the same goal of learning a new representation of the data (either in the same dimensions as the original data, or in a new reduced dimension) with desirable properties, namely: 1) Cells from different cell types are far apart, and cells from the same cell type are in close proximity, and 2) Cells from different batches are mixed together as much as possible while respecting the first property.

The crux of the matter is that in single cell transcriptomics, the cell type label distribution and batch label distribution are often correlated, and so achieving both properties simultaneously is not always possible. Instead, we may need to trade-off these two properties with each other.

In the following sections we describe our contributions to this problem. In the supervised setting, we have shown that our scDGN method can outperform the unsupervised methods in the case where you have labels and prioritize classifier accuracy. In the unsupervised setting, we have proposed a novel framing of the problem, as a point-set registration problem. We proposed a method, SCIPR, which extends ideas originally used for 3D rigid objects to make it usable on scRNA-seq data, and has a unique combination of advantages over previous methods

3.1 scDGN: a supervised approach

Dimensionality reduction is an important first step in the analysis of single cell RNA-seq (scRNA-seq) data. In addition to enabling the visualization of the profiled cells, such representations are used by many downstream analyses methods ranging from pseudo-time reconstruction to clustering to alignment of scRNA-seq data from different experiments, platforms, and labs. Both supervised and unsupervised methods have been proposed to reduce the dimension of scRNA-

seq. However, all methods to date are sensitive to batch effects. When batches correlate with cell types, as is often the case, their impact can lead to representations that are batch rather than cell type specific. To overcome this we developed a domain adversarial neural network model (scDGN) for learning a reduced dimension representation of scRNA-seq data. The adversarial model tries to simultaneously optimize two objectives. The first is the accuracy of cell type assignment and the second is the inability to distinguish the batch (domain). We tested the method by using the resulting representation to align several different datasets. As we show, by overcoming batch effects our method was able to correctly separate cell types, improving on several prior methods suggested for this task. Analysis of the top features used by the network indicates that by taking the batch impact into account, the reduced representation is much better able to focus on key genes for each cell type.

A version of this section was published in the proceedings of the Research in Computational Molecular Biology (RECOMB) conference in 2020 [106] and presented as a talk there, and later published in the *Journal of Computational Biology* [107]. This is joint work with Songwei Ge, Haohan Wang, Eric Xing, and Ziv Bar-Joseph. Code and sample data for scDGN is available at `https://github.com/SongweiGe/scDGN`.

3.1.1 Introduction

Single-cell RNA sequencing (scRNA-seq) has revolutionized the study of gene expression programs [9, 108]. The ability to profile genes at the single-cell level has revealed novel specific interactions and pathways within cells [109], differences in the proportions of cell types between samples [8, 74], and the identity and characterization of new cell types [110]. Several biological tissues, systems, and processes have recently been studied using this technology [8, 74, 109].

While studies using scRNA-seq provide many insights, they also raise new computational challenges. One of the major challenges involves the ability to integrate and compare results from multiple scRNA-seq studies. There are several different commercial platforms for performing such experiments, each with their own biases. Furthermore, similar to other high throughput genomic assays, scRNA-seq suffers from batch effects which can make cells profiled in one lab look very different from the same cells profiled at another lab [111, 112]. This is a key issue for consortium-scale analysis, such as the Human Cell Atlas (HCA) [113, 114] and HUBMaP [115], where researchers across the globe are profiling single cells in their own labs and seeking to perform large-scale analysis that integrates data across the entire consortia. Even for cells profiles in the same lab, we often cannot avoid batch effects, for example in studies where samples are collected at different times or across a large set of individuals [58, 116]. Moreover, other types of high throughput transcriptomics profiling, including microscopy-based techniques, are also generating single cell expression datasets [117, 118]. The goal of fully utilizing these spatial datasets motivates the development of methods that can combine them with scRNA-seq when studying specific biological tissues and processes.

A number of recent methods have attempted to address this challenge by developing methods for aligning scRNA-seq data from multiple studies of the same biological system. Many of these methods rely on identifying nearest neighbors between the different datasets and using them as anchors. Methods that use this approach include Mutual Nearest Neighbors (MNN) [87] and Seurat [29]. Others including scVI and scAlign first embed all datasets into a common lower

dimensional space. scVI encodes the scRNA-seq data with a deep generative model conditioning on the batch identifiers [119] while scAlign regularizes the representation between two datasets by minimizing the random walk probability differences between the original and embedding spaces. While these methods were successful for some datasets, here we show that they are not always able to correctly match all cell types. A key problem with these methods is the fact that they are unsupervised and rely on the assumption that cell types profiled by the different studies overlap. While this works for some datasets, it may fail for studies in which cells do not fully overlap or for those containing rare cell types. Unsupervised methods tend to group rare types with the larger types making it hard to identify them in a joint space.

Recent machine learning work has focused on a related problem termed "domain adaptation/generalization". Methods developed for these problems attempt to learn representations of diverse data that are invariant to technical confounders [120–122]. These methods have been used for multiple applications such as machine translation for domain specific corpus [123] and face detection [124]. Several methods proposed for domain adaptation rely on the use of adversarial methods [121, 125–127], which has been proved effective to align latent distributions. In addition to the original task such as classification, these methods apply a domain classifier upon the learned representations. The encoder network is used for both improving accurate classification while at the same time reducing the impact of the domain (by "fooling" a domain classifier). This is achieved by learning encoder weights that simultaneously perform gradient *descent* on the label classification task and gradient *ascent* on the domain classification task.

Here we extend these approaches, coupling them with Siamese network learning [66] for overcoming batch effects in scRNA-seq analysis. We define a "domain" in this paper as a standalone dataset profiled at a single lab using a single platform. We define "label" as the cell type for each cell in the dataset. Considering the specificity of the cell types in the scRNA-seq datasets, we propose a conditional pair sampling strategy that constrains input pair selection when training the adversarial network. We discuss how to formulate a domain adaptation network for scRNA-seq data, how to learn the parameters for the network, and how to train it using available data.

We tested our method on several datasets ranging in size from 10 to 39 cell types and from 4 to 155 batches. As we show, for all of the datasets our domain adversarial method improves on previous methods, in some cases significantly. Visualization of the learned representation from several different methods helps highlight the advantages of the domain adversarial framework. As we show, the framework is able to accurately mitigate the batch effects while maintaining the grouping of cells from the same type across different batches. Biological analysis of the resulting model identifies key genes that can correctly distinguish between cell types across different experiments. Such batch invariant genes are promising candidates for a cell type specific signature that can be used across different studies to annotate cells.

3.1.2 Methods

Problem Formulation

To formulate the problem we start with a few notation definitions. We assume that the single cell RNA-seq data are drawn from the input space $\mathbf{X} \in \mathbb{R}^p$ where each sample (a cell) $\mathbf{x}$ has

p features corresponding to the gene expression values. Cells are also associated with the label $y \in \mathbf{Y} = \{1, 2, ..., K\}$ which represents their cell types. We associate each sample with a specific domain/batch $d \in \mathcal{D}$ that represents any standalone dataset profiled at a single lab using a single platform. Note that we will use domain and batch interchangeably in this paper for convenience. The data are divided into a training set and a test set that are drawn from multiple studies. The domains used to collect training data are not used for the test set and so batch effects can vary between the training and test data. In practice, each of the domains only contains a small subset of the cell types. This means that the distribution of cell types is correlated with the distribution of domains. Thus, the methods that naively learn cell types based on expression profile [71, 128, 129] may instead fit domain information and not generalize well to the unobserved studies.

Domain Adversarial Training with Siamese Network

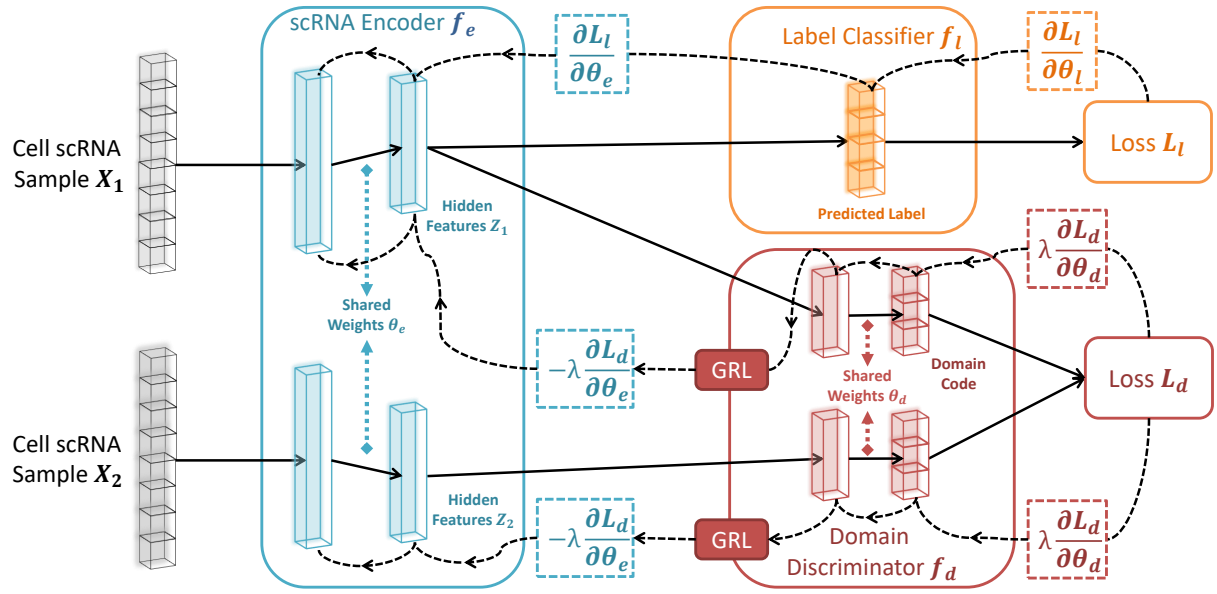


Figure 3.1: Architecture of scDGN. The network includes three modules: scRNA encoder f_e (blue), label classifier f_l (orange) and domain discriminator f_d (red). Note that the red and orange networks use the same encoding as input. Solid lines represent the forward direction of the neural network while the dashed lines represent the backpropagation direction with the corresponding gradient it passes. Gradient Reversal Layers (GRL) have no effect in forward propagation, but flip the sign of the gradients that flow through them during backpropagation. This allows the combined network to simultaneously optimize label classification and attempt to "fool" the domain discriminator. Thus, the encoder leads to representations that are invariant to the different domains while still distinguishing cell types.

To overcome this problem and remove the domain impact when learning a cell type representation we propose a neural network framework which includes three modules as shown in Figure 3.1: scRNA encoder, label classifier, and domain discriminator. The encoder module $f_e(\mathbf{x}; \theta_e)$ is used to reduce the dimensions of the data and contains fully connected layers which produce the hidden features, where θ_e represents the parameters in these layers. The label classifier $f_l(f_e; \theta_l)$ attempts to predict the label of input $\mathbf{x_1}$ whereas the goal of the domain discriminator $f_d(f_e; \theta_d)$ is to determine whether a pair of inputs $\mathbf{x_1}$ and $\mathbf{x_2}$ are from the same domain or not. Past work

for classifying scRNA-seq data only attempted to minimize the loss function for the label classifier $\mathcal{L}_l(f_l(f_e;\theta_l))$ [71, 78]. Here, we extend these methods by adding a regularization term based on the adversarial loss of the domain discriminator $\mathcal{L}_d(f_d(f_e;\theta_d))$ which we will elaborate later. The overall loss E on a pair of samples $\mathbf{x_1}$ and $\mathbf{x_2}$ is denoted by:

$$E(\theta_e,\theta_l,\theta_d) = \mathcal{L}_l\big(f_l(f_e(\mathbf{x_1};\theta_e);\theta_l)\big) \;-\; \lambda\mathcal{L}_d\big(f_d(f_e(\mathbf{x_1};\theta_e);\theta_d),\; f_d(f_e(\mathbf{x_2};\theta_e);\theta_d)\big),$$

where λ can control the trade-off between the goals of domain invariance and higher classification accuracy. For convenience, we use $\mathbf{z_1}$ and $\mathbf{z_2}$ to denote the hidden representations of $\mathbf{x_1}$ and $\mathbf{x_2}$ calculated from $f_e(\mathbf{x};\theta_e)$. Inspired by Siamese networks [66], we implement our domain discriminator by adopting a contrastive loss [130]:

$$\begin{aligned}\mathcal{L}_d\big(f_d(\mathbf{z_1};\theta_d),\; f_d(\mathbf{z_2};\theta_d)\big) = U\frac{1}{2}D\big(f_d(\mathbf{z_1}),f_d(\mathbf{z_2})\big)^2 \\ + (1-U)\frac{1}{2}\big(\max\{0, m - D(f_d(\mathbf{z_1}),f_d(\mathbf{z_2}))\}\big)^2,\end{aligned}$$

where $U = 1$ indicates that two samples are from the same domain d and $U = 0$ indicates that they are not, $D(\cdot)$ is the euclidean distance, and m is the margin that indicates the prediction boundary. The domain discriminator parameters, θ_d, are updated using back propagation to *maximize* the total loss E while the encoder and classifier parameters, θ_e and θ_l, are updated to *minimize* E. To allow all three modules to be updated together end-to-end, we use a Gradient Reversal Layer (Figure 3.1) [125, 131]. Specifically, Gradient Reversal Layers (GRL) have no effect in forward propagation, but flip the sign of the gradients that flow through them during backpropagation. The following provides the overall optimization problems solved for the network parameters:

$$\begin{aligned}(\widehat{\theta_e},\widehat{\theta_l}) &= \arg\min_{\theta_e,\theta_l} E(\theta_e,\theta_l,\widehat{\theta_d}) \\ (\widehat{\theta_d}) &= \arg\max_{\theta_d} E(\widehat{\theta_e},\widehat{\theta_l},\theta_d)\end{aligned}$$

In other words, the goal of the domain discriminator is to tell if two samples are drawn from the same or different batches. By optimizing the scRNA encoder adversarially against the domain discriminator, we attempt to make sure that the network representation cannot be used to classify based on domain knowledge. During the training, the maximization and minimization tasks compete with each other, which is achieved by adjusting the representations to improve the accuracy of the label classifier and simultaneously fool the domain discriminator.

Conditional Domain Generalization Strategy

Most prior domain adaption or generalization methods focused on the cases where the distribution of labels is independent of the domains [120, 121]. In contrast, as we show in Results, for scRNA-seq experiments different studies tend to focus on certain cell types [8, 74, 109]. Consequently, it is not reasonable to completely merge the scRNA-seq data from different batches.

To be specific, aligning the scRNA-seq data from two batches with different sets of cell types would sacrifice its biological significance and prevent the cell classifier from predicting effec-tively. To overcome this issue, instead of arbitrarily choosing positive pairs (samples from the same domain) and negative pairs (samples from different domains), we constrain the selection as follows: 1) for positive pairs, only the samples with different labels from the same domain are selected. 2) for negative pairs, only the samples with the same label from different domains are selected. Figure 3.2 provides a visual interpretation of this strategy. Formally, letting y_i and z_i represent the i-th sample's cell-type label and domain label respectively, we have the following equations to define the value of U for sample pairs:

$$U = \begin{cases} 0, & z_1 \neq z_2 \text{ and } y_1 = y_2 \\ 1, & z_1 = z_2 \text{ and } y_1 \neq y_2 \end{cases}$$

This strategy prevents the domain adversarial training from aligning samples with different labels or separating samples with same labels. For example, in order to fool the discriminator with a positive pair, the encoder must implicitly increase the distance of two samples with different cell types. Therefore, combining this strategy with domain adversarial training allows the network to learn cell type specific, focused representations. We term our model Single Cell Domain Generalization Network (scDGN).

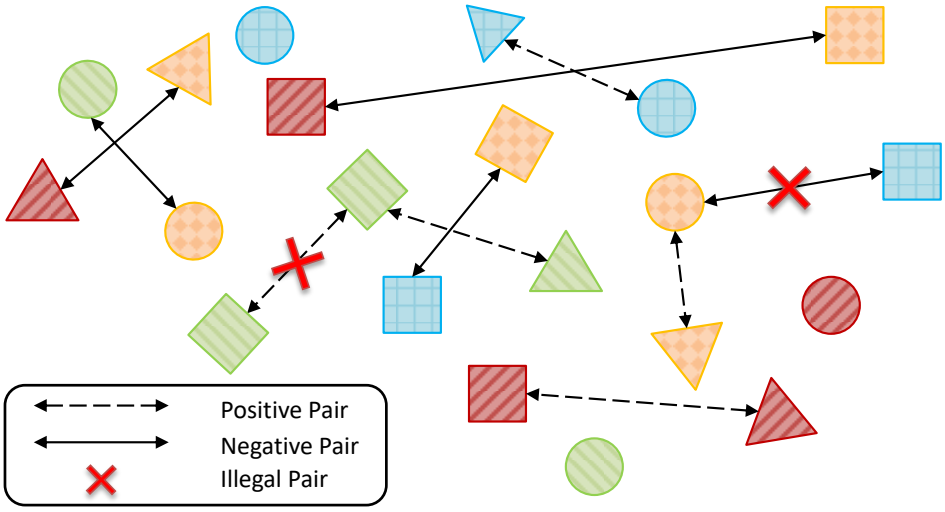


Figure 3.2: Conditional domain generalization strategy: Shapes represent different labels and colors (or patterns) represent different domains. For negative pairs from different domains, we only select those samples with the same label. For positive pairs from the same domain, we only select the samples with different labels.

3.1.3 Results

Experiment Setups

Datasets To test our method and to compare it to previous methods for aligning and classifying scRNA-seq data, we used several recent datasets. These datasets contain between 6,000

and 45,000 cells, and all include cells profiled in multiple experiments by different labs and on different platforms.

Table 3.1: Basic statistics for scQuery, Suerat pancreas, and PBMC datasets

	scQuery			Seurat pancreas			Seurat pbmc		
	data	cell type	domain	data	cell type	domain	data	cell type	domain
training	37697	39	99	6321	13	3	25977	10	8
validation	3023	19	26	-	-	-	-	-	-
test	3770	23	30	638	13	1	2992	10	1

The evaluation datasets include a subset of the data from scQuery [71], which contains 44,490 samples from 155 different experiments, including a broad range of cell types. In addition, we use a Peripheral Blood Mononuclear Cell (PBMC) dataset with 9 batches (sequencing technologies) and 28,969 cells [132]. Finally we also test on a dataset of human pancreatic islet cells from Seurat [29], which we artificially split into 6 smaller datasets to simulate cases where cell types and domains are highly correlated. See Table 3.1 and Supporting Information A.1 in [133] for details on batches, cell type distributions, train and test split information, and normalization for these three datasets.

Model Configurations We used the network of Lin et al [78] as the components for the encoder and the label classifier in our model. The encoder contains two hidden layers with 1136 and 100 units. The label classifier is directly connected to the 100 unit layer and makes predictions based on these values. The domain discriminator contains an additional hidden layer with 64 units and is also connected to the 100 unit layer of the encoder (Figure 3.1). For each layer, $tanh()$ is used as the non-linear activation function. We test several other possible configurations but did not observe improvement in performance. As is commonly done, we use a validation set to tune the hyperparameters for learning including learning rates, decay, momentum, and the adversarial weight and margin parameters λ and m. Generally, our analysis indicates that for larger datasets a lower weight λ and larger margin m for the adversarial training is preferred and vice versa. More details about the hyperparameters and training are provided in Supporting Information A.3 in [133].

Baselines We compared scDGN to several prior methods for classifying and aligning scRNA-seq data. These included the neural network (NN) model of Lin et al [78] which is developed for classifying scRNA-seq data, CaSTLe [129] which performs cell type classification based on transfer learning, and several state-of-the-art alignment methods. For alignment, we compared to MNN [87] which utilizes mutual nearest neighbors to align data from different batches, scVI [119] which trains a deep generative model on the scRNA-seq data and uses an explicit batch identifier to retain conditional independence property of the representation, and Seurat [29] which first identifies the anchors among different batches and then projects different datasets using a correction vector based on the order defined by hierarchical clustering with pairwise distances. Our comparisons include both visual projection of the learned alignment (Figure 3.5 and 3.5) and quantitative analysis of the accuracy of the predicted test cell types (Table 3.2). For the

latter, to enable comparisons of the supervised and unsupervised methods, we used the resulting aligned data from the unsupervised methods to train a neural network that has the same configuration as Lin et al [78]. For scVI, which results in a much lower dimensional representation, we used a smaller input vector and a smaller hidden layer. Note that these alignment methods actually use the scRNA-seq test data to determine the final dimensionality reduction function while our method does not utilize the test data for any model decision or parameter learning. To effectively apply Seurat to scQuery, we remove the batches which have < 100 samples. Also, for those datasets that the assumption of overlapped cell types is not guaranteed such as scQuery, we find that the performance of MNN highly depends on the order of alignment. Therefore, for MNN on the scQuery dataset, we use 10 random permutations of batch orders and report the average accuracy.

Table 3.2: Overall performances of different methods. *MI* represents the mutual information between batch and cell type in the corresponding dataset. The highest test accuracy for each dataset is bolded.

Experiments	MI	NN	CaSTLe	MNN	scVI	Seurat	scDGN
scQuery	3.025	0.255	0.156	0.200	0.257	0.144	**0.286**
PBMC	0.112	0.861	0.865	0.859	0.808	0.830	**0.868**
Pancreas 1	0.902	0.720	0.705	0.591	0.855	0.812	**0.856**
Pancreas 2	0.733	0.891	0.764	0.764	0.852	0.825	**0.918**
Pancreas 3	0.931	0.545	0.722	0.722	0.651	**0.751**	0.663
Pancreas 4	0.458	0.927	0.914	0.914	**0.925**	0.881	**0.925**
Pancreas 5	0.849	0.928	0.882	**0.932**	0.895	0.865	0.923
Pancreas 6	0.670	0.944	0.917	0.946	0.893	0.907	**0.950**
Average	-	0.826	0.817	0.842	0.845	0.840	**0.872**

Overall Performance

As mentioned above, we use the validation set to select the best model when using the scQuery dataset. For the smaller datasets, we use the model obtained after 250 epochs (all models converged after this number of epochs). Test accuracy for the different methods is presented in Table 3.2. We show both mean and standard deviation of the accuracy for 10 randomly initialized experiments. An example for the performance for all methods we tested on the 2 dataset is shown in Figure 3.3. As can be seen, on average scDGN outperforms the other methods in terms of test accuracy, though for a particular cell type we sometimes see other methods perform better. Performance comparisons for all cell types is presented in Supporting Information C (Tables C1-8) [133]. In addition, Table 3.2 presents the Mutual Information (MI) between labels and domains which corresponds to the difficulty of the dataset. A larger MI indicates that models that do not account for the domain are likely to fit the domain information rather than the cell type. For the scQuery dataset, we find the accuracy is low for all methods indicating that this dataset is relatively difficult. This is corroborated by the large MI value. For such data we see a clear advantage for the scDGN: scDGN improves by over 10% over all other methods ($p = 5.069 \times 10^{-5}$ based on Student's t-test when compared to the NN baseline which is tied

for second best). The improvements over other single cell alignment methods are even more significant. scDGN also achieves the best performance on the second largest dataset, the PBMC dataset. However, given the very low MI for this dataset the performance of the other methods, including the baseline NN, is almost as good as the performance of scDGN. The third dataset we test on is the Seurat pancreas dataset. This is the smallest dataset and so it has the least number of training samples. Still, of the 6 settings we tested (which differed in the subset of cells that were excluded from training), we find that scDGN is the top performer in 4 of them, comparable to the top performer for another 1 and in only one setting (Pancreas 3, with the highest MI) is significantly outperformed by Seurat. Note that even for the Pancreas 3 data the domain adversarial training helps: using this the scDGN is able to improve by more than 20% over the baseline NN used for the label classifier.

# train	# test	Cell Type	NN	CaSTLe	MNN	scVI	Seurat	scDGN
330	25	gamma	0.832	0.56	0.82	0.716	0.724	0.794
462	21	ductal	1	1	1	1	1	1
323	18	mast	0.95	0.833333	0.9556	0.9833	0.8833	0.8833
21	14	endothelial	0.4786	0.428571	0.4786	0.75	0.0286	0.5
6	3	beta	0	0	0.0333	0	0.3333	0.0333
15	1	quiescent_stellate	1	1	1	1	0	0.9
5	1	macrophage	0.6	1	0.6	0.7	0.1	0.4
1	1	activated_stellate	0.5	1	0.4	1	0	0
327	36	schwann	0.7806	0.805556	0.7639	0.8556	0.1667	0.866
3	5	epsilon	0.16	0	0.18	0.34	1	0
1199	239	alpha	0.8695	0.648536	0.8833	0.777	0.9063	0.9381
74	16	delta	0.975	0.625	0.9687	0.9	1	0.9812
469	258	acinar	0.9647	0.910853	0.9725	0.8	0.8725	0.9667
3235	638	Average	0.891	0.764	0.899	0.852	0.825	0.918

Figure 3.3: Test Accuracy of each model on different cell types from pancreas2 dataset.

Visualization of the Representation Learned by Alignment and Classification Methods

To further explore the effectiveness of the batch removal provided by our proposed domain adversarial training with conditional domain generalization strategy, we visualize the 100-dimensional hidden representations learned by NN and scDGN: Figure 3.4 presents both PCA and t-SNE plots for several different cell types across the three datasets. Note that we only used the top two components for visualization though the actual classification for all methods was performed using all raw feature values without any dimensionality reduction. Points are colored using their batch IDs in order to evaluate batch effects. As can be seen, using scDGN we obtain results that are much better at mixing cells from the different batches when compared to the baseline NN model. The impact is larger for the pancreas datasets which have larger MI compared to the PBMC dataset, which helps explain the large increase in performance for these two datasets.

We next extended this comparison and visualized the learned (aligned) representations for all methods using data from both the Pancreas2 and scQuery datasets (Figures 3.5 and 3.5). For the Pancreas2 dataset, we visualize the entire dataset. For scQuery, given the large number of cell types and domains, we present PCA visualization of a subset of cell types and domains. As can be seen, in addition to scDGN, Seurat is also able to successfully mix the data from

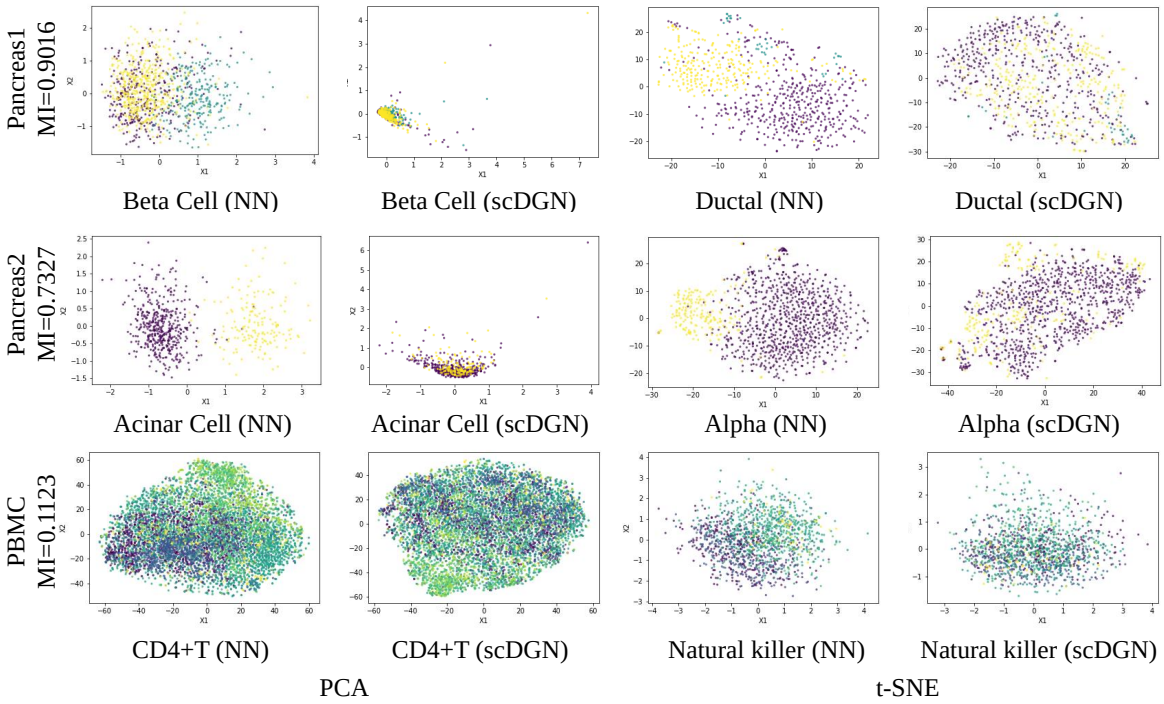


Figure 3.4: Visualization of learned representations for NN and scDGN: using PCA and t-SNE Rows: The three datasets we tested the method on. Columns: Methods and cell types. For each row, data from different batches are distinguished using different colors.

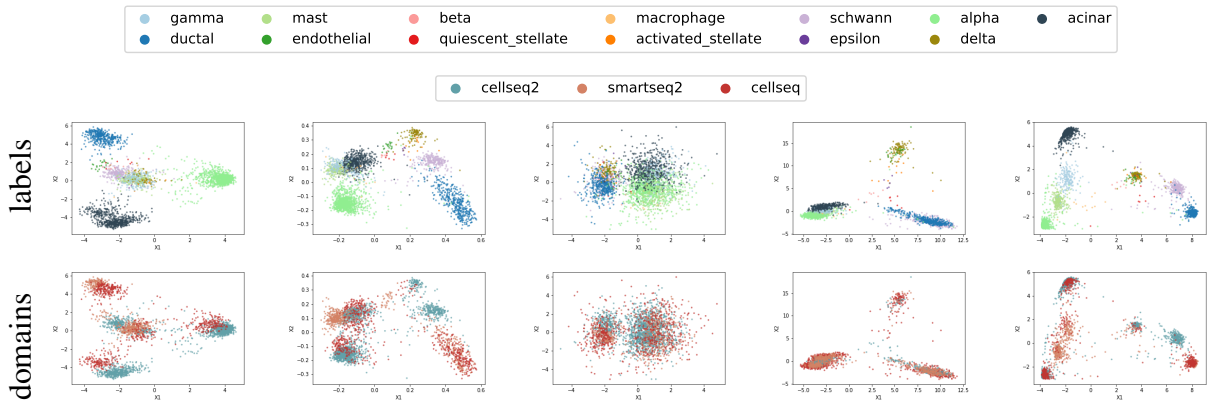
different batches. However, as the results in Table 3.2 indicate this may come at the expense of not correctly separating cell types. MNN and scVI are not always effective at removing batch effects for the cell types. In contrast, scDGN is able to do both domain mixing and cell type assignment, leading to its better performance overall. For example, for the acinar and alpha cell types in the pancreas dataset (Figure 3.5), only scDGN , MNN, and Seurat are able to align the data from different domains. However, MNN and Seurat over-correct the representation by aligning different cell types from different domains, mixing acinar and gamma cells. Additional visualizations for other cell types and domains can be found in Supporting Information D [133], where the same advantages of scDGN over other methods can be consistently observed.

Analysis of Key Genes

While NNs are often treated as black boxes, recent methods provide useful directions for making them more interpretable [134]. Here we use activation maximization, which relies on the gradient of the correct category logit with respect to the input vector to select the key inputs for each of the models [135–137]. Formally, given a particular cell type i and a trained neural network ϕ, activation maximization looks for important input genes x' by solving the following optimization problem:

$$x' = \max_{x}(\phi(x) \cdot e_i),$$

where e_i is the natural basis vector associated with the i-th category. This can be solved through

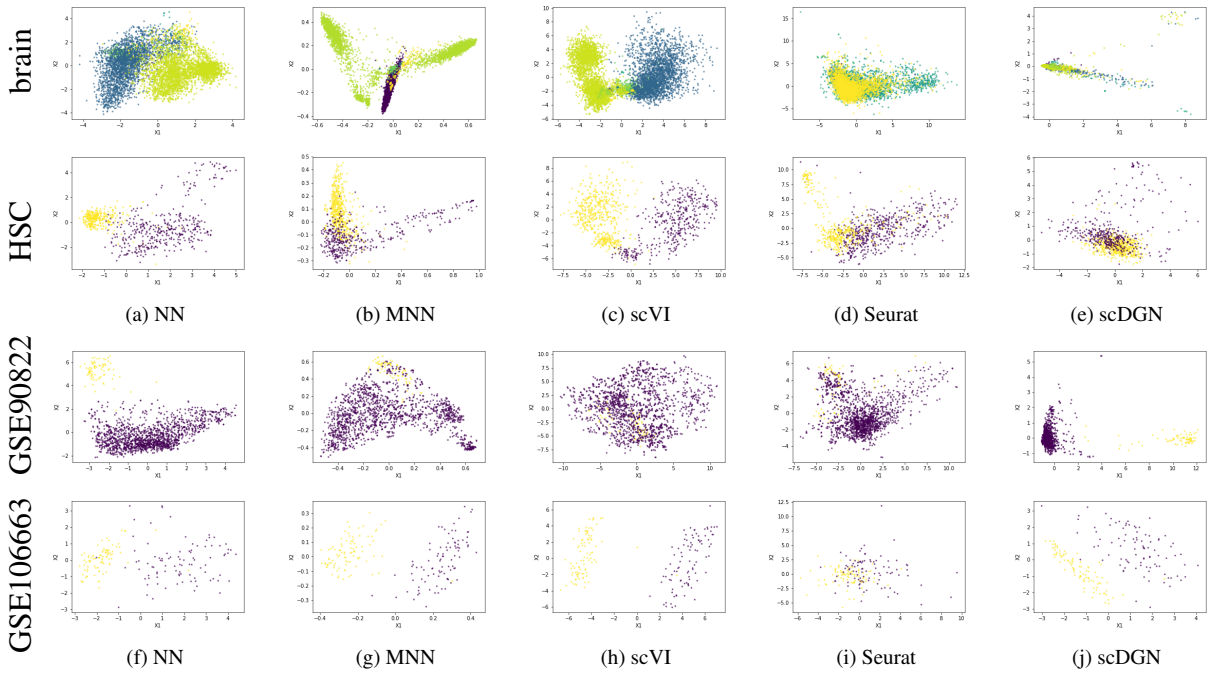


(a) NN (b) MNN (c) scVI (d) Seurat (e) scDGN

Figure 3.5: PCA visualizations of the representations learned by different models on the full Pancreas2 dataset. Colors for different cell types and domains are shown in the legend at the top.

backpropagation, where the gradient of $\phi(x)$ with respect to x, which can be viewed as the weight of the first-order Taylor expansion of the neural network, are calculated to iteratively update the input. We follow a previous method [136] and initialize the optimization with a zero vector. Given this setting, we ran the optimization for 100 iterations with learning rate set to 1. The important genes are selected as those inputs leading to the largest changes compared with the initialization values. To compare scDGN and NN for certain cell types, we select the top k genes with the largest changes and perform GO analysis on these selected genes.

As an example, consider the genes identified for the liver cell type using the scQuery dataset. We select the top 100 genes for this cell type from NN and scDGN and present the enriched GO categories on Biological Process with adjusted p-value $< 1.0 \times 10^{-4}$ in Tables 3.3 and 3.4. We also list these genes by order in Supporting Information A.3 [133]. As can be seen, while a number of significant GO categories are identified for the top 100 NN genes, these are generic and not liver specific. They include general terms related to interactions between organs and immune response categories that are active in multiple organs and cell types. In sharp contrast, the categories identified for scDGN are much more specific and highlight key pathways that are mainly utilized in the liver. For example, the top category for the scDGN genes, "chylomicron remodeling", refers to the main physiological purpose of chlyomicron remnants: to facilitate the return of bile lipoproteins and cholesterol to the liver [138]. Specifically, in this pathway chylomicrons (lipoproteins) are broken down (remodeled via hydrolysis) and converted to a form called "chlyomicron remnant" that is taken up by specific receptors that exist primarily on the surface of liver cells [139]. The second term, "pos. regulation of cholesterol esterification" refers to cholesterol esterification, a critical step in reverse cholesterol transport, the process in which excess cholesterol is sent to the liver to be removed from the body [140, 141]. Furthermore, Cholesteryl Ester Transfer Protein (CETP) is a key enzyme involved in this process and is highly expressed in liver cells, and variants of CETP are associated with increased risk of atherosclerosis [140, 142]. The fifth most significant term, "lipoprotein remodeling" is part of the two aforementioned processes. The top 100 genes identified by the scDGN include *apoa1* (main protein component of High-Density Lipoprotein cholesterol), *apoa2*, and *apoc1*, all of which encode lipoproteins



(a) NN (b) MNN (c) scVI (d) Seurat (e) scDGN

(f) NN (g) MNN (h) scVI (i) Seurat (j) scDGN

Figure 3.5: PCA visualizations of the representations of certain cell types and batches by different models for the scQuery dataset. Top two rows: Cell types. Colors represent different batches. HSC = hematopoietic stem cell. Bottom two rows: Batches. Colors represent different cell types.

that are primarily expressed in the liver [143, 144]. These genes were not included in the top 100 genes by the NN. We present the GO analysis results comparison for several additional cell types in Supporting Information E.2 [133].

3.1.4 Discussion

Single cell computational methods that do not account for batch effects are likely to fit the noise introduced by the batches. Several recent methods have been proposed for aligning scRNA-seq from multiple studies of the same tissues or processes. Most of these methods are unsupervised and assume that the cell types among different batches overlap. However, we show that these methods would fail on the studies in which cell types do not fully overlap, which is often the case when dealing with multiple datasets. To overcome this problem we extend a supervised scRNA-seq cell type assignment method based on NN and regularize its prediction to be invariant to batch effects.

Our method is based on the ideas of domain adversarial training. In such training, two competing tasks are used to optimize the representation of scRNA-seq data. The first focuses on the traditional goal of cell type identification while the second attempts to construct representations that are not affected by specific batch or experimental artifacts. This is accomplished by jointly minimizing a loss function that takes into account both goals, accounting for the weight of each of the goals using a gradient reversal layer. We also proposed a conditional strategy to avoid over-correction. We presented efficient learning methods for this setting and tested it on

Table 3.3: GO analysis results for top 100 scQuery liver genes in the NN method.

term_name	term_id	p_{adj}	$-\log_{10} p_{adj}$
symbiotic process	GO:0044403	1.16E-08	7.935246875
interspecies interaction between organisms	GO:0044419	3.14E-08	7.503093471
viral process	GO:0016032	3.69E-08	7.433145019
immune response	GO:0006955	2.5491E-06	5.593613105
multi-organism process	GO:0051704	1.40837E-05	4.851282542
immune effector process	GO:0002252	4.53533E-05	4.34339136
response to stress	GO:0006950	5.56335E-05	4.254663785
defense response	GO:0006952	6.18759E-05	4.208478308

Table 3.4: GO analysis results for top 100 scQuery liver genes in the scDGN method.

term_name	term_id	p_{adj}	$-\log_{10} p_{adj}$
chylomicron remodeling	GO:0034371	3.04042E-05	4.517066786
positive reg. of cholesterol esterification	GO:0010873	3.04042E-05	4.517066786
negative reg. of cellular component organization	GO:0051129	3.94437E-05	4.404022507
protein-lipid complex remodeling	GO:0034368	7.34551E-05	4.133978335
plasma lipoprotein particle remodeling	GO:0034369	7.34551E-05	4.133978335
protein-containing complex remodeling	GO:0034367	8.8522E-05	4.052948555

three large scale scRNA-seq datasets containing experiments from several different platforms for partially overlapping cell types.

As we show, our scDGN method is able to correctly identify cell types in the test datasets. For the largest dataset we tested on which contained close to 40 different cell types, scDGN significantly outperformed all prior methods. It also ranked first for the 2nd largest dataset and for all but 1 of the 6 tests on the third dataset. Importantly, it always outperformed the supervised learning based method indicating that batch effects should be addressed when designing such methods for cell type assignments. In addition to accurately assigning cell types, further analysis of significant genes indicates that by overcoming batch effects scDGN is better able to focus on relevant sets of genes when compared to prior supervised methods, explaining its improvement in accuracy.

While scDGN performed best on the data we analyzed, there are a number of possible issues with this approach. First, it learns a large number of parameters which require large input datasets. However, as we showed, scDGN is able to perform well even for datasets with a few

thousand cells which matches current sizes of scRNA-seq datasets. Second, scDGN is based on NNs which are often seen as a black box, making it hard to interpret the resulting model and its biological relevance. Recent work provides a number of directions that can be used to overcome this issue. As we showed, using activation maximization we were able to identify several relevant cell type specific genes in the learned network. Future work would include using additional NN interpretation methods, including LIME [134] or ROAR and KAR [145], to further identify the set of genes that play the largest role in the decisions the network makes. Third, as shown in Supporting Information D.3 [133], scDGN sometimes does not mix up the representations from different batches for all cell types. Considering the visualization results for NN in Supporting Information D.8 [133] and its competitive performance in Table 3.2 together, it may indicate that it is not always necessary to remove batch effects for the model to achieve high test accuracy. Therefore, it is worthwhile to further study when the alignment is imperative. Finally, unlike prior scRNA-seq alignment methods scDGN is supervised. While this is an advantage when it comes to accuracy, as we have shown, it may be a problem for the new data. We believe that as more scRNA-seq and other high throughput single cell data accumulate, we would have labeled data for most cell types which would enable training an scDGN for even more cell types. As we have shown with the scQuery dataset, for which scDGN significantly outperformed all other methods, when such data exists scDGN is able to correctly align experiments and platforms not seen in the training set. More generally, this paper presents a method that connects the batch effect removal problem to domain adaptation tasks in machine learning. Recent developments in this direction in the machine learning community may lead to even better results for batch removal problems. For instance, it has been recently shown that self-supervision with domain knowledge, for instance rotation prediction [146], can greatly improve the learned features and generalization to unseen data. It would be interesting to consider if similar biological information, for example knowledge about gene interactions, can be used to further improve the solution for alignment problems.

scDGN is implemented in Python with the PyTorch API [70] and users can obtain the code and sampled data from `https://github.com/SongweiGe/scDGN`.

3.2 SCIPR: an unsupervised approach

While in the previous section we described our supervised approach (scDGN) for integrating scRNA-seq batches which outperformed the unsupervised methods in our experiments, we often have to resort to unsupervised methods for scRNA-seq alignment because label information might not be available. This is especially the case for novel or rare cell types. In this setting, our algorithms only have access to the structure in the data itself, and any external assumptions we may make.

Several studies profile similar scRNA-Seq data using different technologies and platforms. A number of alignment methods have been developed to enable the integration and comparison of scRNA-Seq data from such studies. While each performs well on some of the datasets, to date no method was able to both perform the alignment using the original expression space and generalize to new data. To enable such analysis we developed Single Cell Iterative Point set Registration (SCIPR) which extends methods that were successfully applied to align image data

to scRNA-Seq. We discuss the required changes needed, the resulting optimization function, and algorithms for learning a transformation function for aligning data. We tested SCIPR on several scRNA-Seq datasets. As we show it successfully aligns data from several different cell types, improving upon prior methods proposed for this task. In addition, we show the parameters learned by SCIPR can be used to align data not used in the training and to identify key cell type-specific genes.

A version of this section was published in *PLOS Computational Biology* and is joint work with Ziv Bar-Joseph [147]. The source code, installation instructions, and user guide documentation for SCIPR is available at `https://scipr.readthedocs.io`, and our benchmarking pipeline and data used are available at `https://github.com/AmirAlavi/sc-alignment-benchmarking`.

3.2.1 Introduction

While only recently introduced, single-cell RNA-sequencing (scRNA-seq) has quickly developed into an indispensable tool for transcriptomics research. Driven by the development of droplet microfluidics-based methods [1, 3–5] and split-pool barcoding-based methods [148, 149], current experiments are able to simultaneously profile expression of genes in tens of thousands of single cells. Studies ranging from cell type and state identification [8, 9] to tracking early development [12, 13] to unveiling the spatial organization of cells [10, 11] are all utilizing scRNA-Seq data, providing new insights about the activity of genes within and between cells.

While the size and number of individual scRNA-seq datasets is large and constantly growing, the question of how to integrate scRNA-Seq data from multiple experiments or platforms has become increasingly relevant. Different labs are seeking to analyze related tissues in an organ system, such as mapping out the cell types in the human pancreas [150] or building an adult mouse brain cell atlas [151]. On on even larger scale, consortia such as the Human Cell Atlas [113, 114] or the HUBMaP [115] are organizing researchers globally with the goal of mapping cells in the entire human body.

Combining datasets, even for the same tissue, across platforms or labs is a challenging problem. This process is often referred to "dataset alignment", "dataset harmonization", or "batch correction," and is an active area of research. A number of methods have been recently suggested to address this problem. Many of these rely on nearest neighbors computations. For example, Mutual Nearest Neighbors (MNN) integrates two datasets by first identifying cells in the two datasets that are mutual nearest neighbors (in each other's set of k nearest neighbors) [87]. It then computes vector differences between these pairs and uses weighted averages of these vector differences to shift one batch onto the other. Another method, Seurat [29], extends this idea by first computing MNNs in a reduced dimension space, via canonical correlation analysis (CCA) which identifies common sources of variation between the two datasets, and then proceeding to correct the batch effects in a similar fashion as MNN. Other methods such as scVI [119] and ScAlign [152] use a neural network embedding to align the two datasets. These methods seek to encode the scRNA-seq datasets using a common reduced dimensional space in which the batch effects are reduced. While the above methods are unsupervised, there are also a few supervised methods proposed for this task. These method require as input the correct cell type labels for cells in the training data and use that to learn a function to assign cell types for the test data. An

example of such method is Single Cell Domain Generalization Network (scDGN) which uses a supervised neural network trained with adversarial objectives to improve cell type classification [106]. Another example is Moana, which uses hierarchical cell type classifiers robust to batch effects to project labels from one dataset onto another [56].

Each of the methods mentioned above offers different features and so might be appropriate for different settings. For example, some methods align the data in the given gene space and thus maintain gene semantics while others, namely the neural network-based methods, do the alignment in a new embedded space (i.e. a reduced dimensional space). On the other hand, the neural network methods typically are learning an alignment function which enables the alignment to be applied to new data (generalization). A comparison of the features of several popular methods is summarized in Table 3.5. See also [153–155] for recent reviews comparing different alignment methods.

As the table shows, none of the current methods enable both maintenance of semantics (required for analyzing genes following the alignment) and generalization (required for keeping the alignment consistent when new data arrives). Here we propose a new method, Single Cell Iterative Point set Registration (SCIPR), which achieves both using an unsupervised framework. Our method extends a well known method in image analysis termed iterative closest points (ICP), which is used for the problem of point-set or point-cloud registration [156]. In ICP, two datasets are represented as sets of points in a common coordinate system, and the method proceeds by pairing together points between the two sets and learning a transformation to move one set closer to (the corresponding points) in the other [157]. A method based on ICP can maintain semantics because it operates in the input data space, meaning that if the input features are genes, then the output features will also be those same genes. Such methods can also generalize to unseen data because they are fitting a transformation function, which can then be applied to new batches.

We tested SCIPR on three benchmark datasets and compared its performance to several prior methods suggested for the alignment task. As we show, single cell iterative point set registration outperforms prior methods for most of the tasks and is able to generalize to both unseen data in the target and in the source batch by learning a general function which can be applied to new data. Finally, since it retains the original (gene space) representation, the coefficients learned by single cell iterative point set registration can be used to identify key genes related to the cell types being analyzed.

3.2.2 Methods

Dataset selection

To evaluate SCIPR and to compare its performance to previous alignment methods we used different scRNA-Seq datasets, each profiling similar cells in multiple batches. The first is the CellBench dataset (GEO: GSE118767) [158], which profiled human lung cancer cell lines and contained three batches, each from a different platform: 10x Chromium [5], Dropseq [1], and CEL-seq2 [159] (Table 3.6a). The smallest batch had 210 cells (Dropseq) and the largest had 895 (10x Chromium) after removing cells with low reads, and we filtered the genes to the most highly variable genes across all batches leaving us with 2351 genes (Supporting Methods 3.2.3). The second was data from human pancreatic cells (GEO: GSE84133) [160], with four batches all

Method	Unsupervised?	Corrects input?	Maintains semantics?	Generalizable?	Transfers labels?
scDGN		X		X	X
Moana	X				X
ScAlign	X	X		X*	
scVI	X	X		X	
Harmony	X	X			
Scanorama	X	X	X		
MNN	X	X	X		
Seurat	X	X	X		
SCIPR	X	X	X	X	

Table 3.5: Comparison of features and properties of various scRNA-seq alignment methods. The "Corrects input?" column refers to whether the method actually aligns (transforms) the input data batches in order to integrate them. The "Maintains semantics?" column refers to whether the output of the method retains the gene semantics given as input. The "Generalizable?" column refers to whether the method learns a model which can be applied to new data. The "Transfers labels?" column refers to whether the method also explicitly aims to apply the cell type labels of one data batch onto another, unlabeled batch.
* ScAlign is theoretically able to be applied on new data, as it learns a neural network embedding model, but the ability to save and load the function in different sessions to apply it on new data was not available in software at the time of testing.

using Indrop sequencing [4] where the largest batch had 1488 cells (inDrop3), the smallest had 834 cells (inDrop4), and we used the five largest cell types and the set of 2629 highly variable genes (Table 3.6b). Finally, the third and largest dataset is a PBMC dataset (GEO: GSE132044) [161] which consisted of four different batches using 10x Chromium [5] sequencing. We used the three largest cell types and the largest batch had 2510 cells (10x Chorm. (v2)), the smallest had 2011 cells (10x Chrom. (v2) A), and we used the set of 1466 highly variable genes (Table 3.6c). See Supporting Methods 3.2.3 for complete details, and Table 3.6 for exact numbers of cells in each batch and their cell type distributions.

scRNA-seq alignment

In the scRNA-seq alignment task, our goal is to learn a new representation of the data (either in the same dimensions as the original data, or in a new reduced dimension) to accomplish the following:

Property 1 *Cell type identification: Cells from different cell types are distinct and cells from the same type are in close proximity*

Property 2 *Batch mixing: Cells from different batches are mixed together as much as possible while respecting the first property*

Point set registration for single cell alignment

Unsupervised alignment of single cell data relies on the implicit assumption that the different datasets share several of the same cell types though potentially using different representations for the same type. A similar assumption is central to much of the literature in point set registration, a well-studied problem in the robotics and computer vision fields [156]. In the point set registration problem, we wish to assign correspondences between two sets of points (two

Cell type Batch	H1975	H2228	HCC827	total
10x	310	312	273	895
CELseq2	103	67	70	240
Dropseq	79	65	66	210

(a) CellBench dataset (GEO: GSE118767) [158]

Cell type Batch	acinar	alpha	beta	delta	ductal	total
inDrop1	58	160	519	110	93	940
inDrop2	87	326	221	58	146	838
inDrop3	468	438	312	26	244	1488
inDrop4	59	184	351	71	169	834

(b) Pancreas dataset (GEO: GSE84133) [160]

Cell type Batch	B cell	CD4+ T cell	Cytotoxic T cell	total
10x Chrom. (v2) A	287	550	1174	2011
10x Chrom. (v2) B	388	905	953	2246
10x Chrom. (v3)	346	960	962	2268
10x Chrom. (v2)	861	955	694	2510

(c) PBMC dataset (GEO: GSE132044) [161]

Table 3.6: Cell type and batch distributions for three scRNA-seq datasets we use for evaluation. Each row pertains to a batch, each column pertains to a cell type, and each value is the number of cells for each row and column combination. Numbers here are *after* our preprocessing described in Supporting Methods 3.2.3. The largest batch in each dataset is bolded, which we use as our reference "target" batch in our alignment tasks.

"point clouds"), and learn a transformation that maps one set onto the other (Figure 3.10). Point sets are commonly the 2D or 3D coordinates of rigid objects, and the class of transformation function under consideration is often rigid transforms (rotations, reflections, and translations). The various point sets often originate from differing settings of sensors (viewing angle, lighting, resolution, etc) viewing the same objects or scene. Among the most widely used and classical of point-cloud registration algorithms is Bessl and McKay's Iterative Closest Point (ICP) algorithm [157]. Briefly, each iteration of ICP has two steps: 1) assigning each point in one set (A, "source") to its closest point in the other set (B, "target"), 2) update the rigid transformation function to transform the points in A as close as possible to their assigned points in set B. At the end of each iteration, the points in A are transformed via the current rigid transform and the process is repeated until convergence. Thus, each iteration of ICP can be concisely represented as minimizing the following loss function:

$$L_{ICP}(A, B, f_\theta) = \sum_{i \in A} \min_{j \in B} \frac{1}{d} ||f_\theta(A_i) - B_j||_2^2 \tag{3.1}$$

where $A, B \in \mathbb{R}^d$, and d is the number of genes

and f_θ is further constrained to rigid transformation functions

However, applying ICP as-is to align two scRNA-seq datasets could be problematic since:

- ICP assumes that every point in A corresponds to a point in set B, whereas scRNA-seq datasets may not fully overlap in cell types. For example, in studying embryonic development, we observe the transcriptome at different embryonic days, where some cell fates exist only after a certain day [58, 116].
- ICP assumes that a rigid transform relates the two sets. This may have been appropriate for 3D rigid objects, but not for the complicated, high-dimensional single cell transcriptome data.
- ICP is prone to assigning many points in A to the same point in B (collapsing a point) even when they are not fully compatible [162]. In contrast, if the same cell type exists in both datasets we can expect the number of cells to be more balanced.

Thus, while ICP has been very successful in image analysis, it requires modifications in order to accurately align scRNA-Seq data.

Adapting ICP for scRNA-seq dataset alignment

Given the discussion above, both stages of ICP need to be changed in order to align scRNA-Seq data. More formally, these two stages are:

1. Assignment stage (input: point sets A and B) - assign pair set $S \subseteq \{(i, j) \mid 1 \leq i \leq |A|,\ 1 \leq j \leq |B|\}$ (Figure 3.10 panel 2). Options for this stage may vary in the cardinality of S, and whether or not it allows points to be shared between pairs.
2. Transformation stage (input: assigned pairs S) - given S from the Assignment stage, learn a transform function that transforms points in A to reduce the mean squared error (MSE) between the assigned pairs in S (Figure 3.10 panel 3). Options for this stage vary based on the family of functions considered.

To adapt stage 1, we propose two approaches for assigning points in the next section: one based on a novel greedy algorithm and another based on Mutual Nearest Neighbors (MNNs). For stage 2, we set the family of transformation functions to be affine transformations (section titled "Learning a transformation function").

Assigning cells between datasets

First, we focus on the Assignment stage. The input to this stage is the target set B and the current state of the source set of points A (A is being updated at every iteration of the algorithm). ICP computes the pairwise distance matrix between members of these sets $\boldsymbol{D} \in \mathbb{R}^{n \times m}$ where $n = |A|$

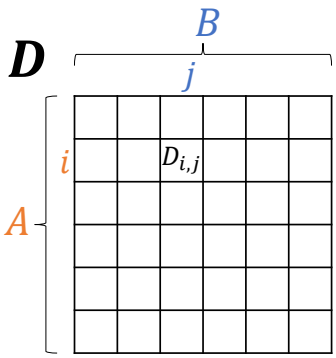


Figure 3.6: The distance matrix between cells in batches A and B. $\boldsymbol{D} \in \mathbb{R}^{n \times m}$ where $n = |A|$ and $m = |B|$ and $D_{i,j}$ is the distance between cells A_i and B_j. In our work, A is called the "source" set and B is called the "target" set. We use the euclidean distance throughout our paper.

and $m = |B|$ (Figure 3.6), and finds the element in each row with the smallest distance to match to that point in A.

In contrast, for scRNA-seq alignment we would like to require the following:

- Not too many points in A are matched with the same point in B (avoid collapsing many points in one dataset onto a single point).
- Not all points must be assigned (since the two dataset may not fully overlap in terms of cell types).

One approach for addressing the first requirement is using a bipartite matching algorithm [163] instead of picking the closest point. In such an algorithm a global optimal matching is found such that each point is only matched to a single point in the other set. However, such algorithms violate the second requirement since they result in "perfect" matchings, where all points in A are matched. An alternative is to use partial matching algorithms in which only a subset (or a fraction) of the points in A are required to be matched to points in B. Optimal partial matching is a well studied problem in the computer science literature and requires solving a min-cost flow graph problem [164]. The problem can be solved via an efficient network simplex algorithm, however, for graphs with thousands of nodes (as in single cell data) this is still rather time consuming. If we let the number of vertices be $V = n + m$ (e.g. number of cells in both batches), the number of edges be $E = n \times m$, and the largest edge weight (distance between points) be C, then the polynomial time network simplex algorithm has a run time of $O(VE \log V \log(VC))$ [165]. Given the large number of cells in each dataset such partial matching methods are too time consuming in practice (Figure 3.7).

Instead, in Algorithm 2 we propose an efficient greedy algorithm for partial assignment between A and B. The algorithm sorts all of the edges between members of the two sets based on distance. Next, we proceed along the ordered edges starting from the smallest distance. If an edge includes a point in set B that has already been selected β times by previously chosen edges, we discard it and continues down the list. Our algorithm has parameters to adjust how many times we allow each point in B to be matched to (β), and how many of the points in A must be matched (α). In our experiments, we set $\beta = 2$ to allow more flexibility than bipartite single-matching, while strictly preventing over-matching and the collapse of several cells onto

Algorithm 2: Greedy pair assignment algorithm. It takes as input the distance matrix $\boldsymbol{D}$ (Figure 3.6) and algorithm hyperparameters α and β and returns the set of pairings S between the source batch of cells A and the target batch of cells B. The rows of the matrix $\boldsymbol{D}$ correspond to cells in A, the columns correspond to cells in B, and the value at D_{ij} is the distance between cells $d(A_i, B_j)$ where $d(\cdot, \cdot)$ is the euclidean distance function. S is a set of tuples of indexers into sets A and B which will be chosen by the algorithm: $S \subseteq \{(i, j) \mid 1 \leq i \leq |A|,\ 1 \leq j \leq |B|\}$. The $countOccurences(j, S)$ function below counts the number of times the jth target cell appears in the set of pairings S.

Input: $\boldsymbol{D} \leftarrow$ distance matrix, $\alpha \leftarrow$ source set match threshold, $\beta \leftarrow$ target set node match limit
Result: $S \equiv$ pair set

$S \leftarrow \{\}$;
$P \leftarrow$ sortElementsAscending($\boldsymbol{D}$);
foreach D_{ij} *in P* **do**
    if $i \notin S$ *and countOccurences(*j, S*)* $\leq \beta$ **then** `/* assign pair if` i `hasn't been matched yet, and` j `hasn't been matched more than` β `times */`
        $S \leftarrow S \cup \{(i, j)\}$;
    end
    if $\frac{|S|}{|A|} \geq \alpha$ **then** `/* stop if more than` α `A cells have been assigned */`
        break;
    end
end

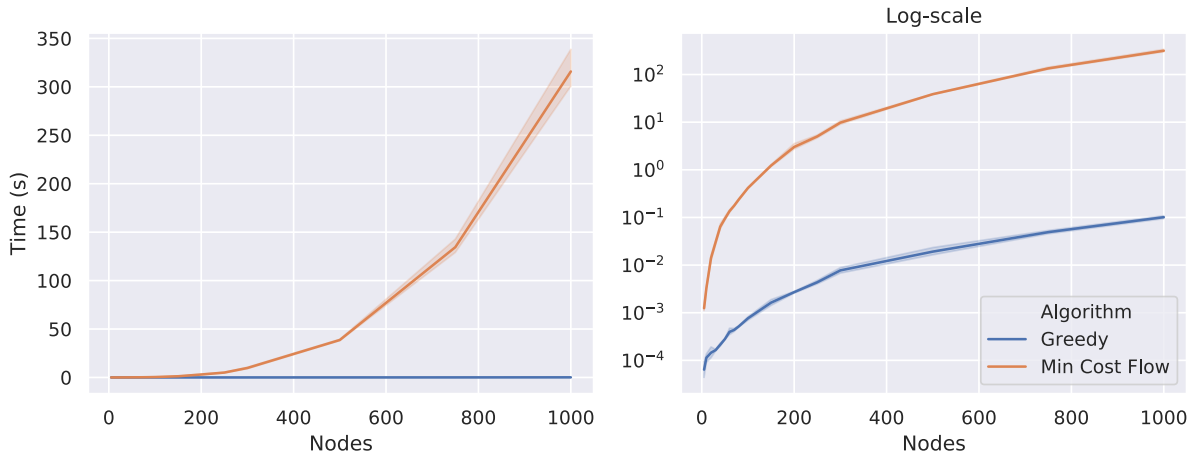


Figure 3.7: A comparison of runtimes of a greedy matching algorithm (Algorithm 2) compared to a network flow-based approach for finding an optimal partial matching (Min Cost Flow). For both algorithms, we generated random distances matrices with varying numbers of cells (also called nodes). In this simple case designed to test algorithm runtimes as a funciton of input size, the distances were integers uniformly drawn from [0, 50]. The distance matrices are square, representing the case when a source batch has the same number of cells as the output batch, where the x-axis in the above plots is the number of cells in each batch. For the greedy algorithm, we used the default parameters as discussed in the main body and in the Supporting Methods section 3.2.3. For the Min Cost Flow algorithm, we started by constructing a bipartite graph where nodes in one set represented cells in the source batch, and nodes in the other set represented cells in the target set. The directed (from source to target) edge weights (costs) were set to the distances between the nodes as given by the randomly generate distance matrix. Then a "source" node was added and connected to all of the nodes of the source cells, and a "sink" node was added and all of the nodes of the target cells were connected to it. The demand of the source node was set to -0.5 $\times$ nodes (the number of "units" of flow that this node wants to send, i.e. the number of pairings of cells we want to assign), and the demand of the sink node was set to negative of this (the number of "units" of flow that it wants to receive). Finally, the capacity of each edge in this directed network was set to 1, and the the network simplex algorithm was used to find a solution. The directed graph was constructed using the NetworkX python package and they network simplex algorithm was run via the `min_cost_flow` function [166].

the same point and $\alpha = 0.5$ because requiring 50% matching would allow for cases where significant portions of the source or target cell types do not overlap. These hyperparameters can be more precisely set based on the user's prior belief of the composition of their batches, though we show that these default settings worked well in our experiments across three datasets. The runtime of this algorithm is dominated by the $sortElementsAscending$ function which sorts the distances leading to a worst case runtime of $O(E^2)$ and a much faster $O(E \log(E))$ on average. Though not an optimal solution to the partial bipartite matching problem, we find that this works well in our related scRNA-seq cell pair assignment problem for alignment.

We also experiment with a matching procedure which follows the foundational work of using mutual nearest neighbors (MNNs) [87] to define our pair assignments between the two sets. For a point i in set A and a point j in set B, if i is in the set of k-nearest neighbors among A for point j, and j is in the set of k-nearest neighbors among B for point i, then i and j are MNNs. In our experiments, we set $k = 10$. To pick this default value for k, we used the average of the defaults in two well-established analysis frameworks for scRNA-seq, Seurat [167] (`FindIntegrationAnchors` function which uses $k = 5$) and scanpy [28]

(`scanpy.pp.neighbors` which uses $k = 15$) and we found that such assignment worked well in our experiments across all three datasets.

Learning a transformation function

So far we focused on matching points given their distance. In the next stage, we will fit a transfor-mation function to align the matched points. As discussed above, the family of rigid transforms is not well suited to compute such alignments for scRNA-seq data (Figure 3.8). Instead, we pro-pose to use the family of affine transformations to align scRNA-seq datasets. Affine transforms are of the form $f_\theta(x) = W^T x + b$ where $\theta = \{W, b\}$ is the learnable weights of the function, and include rotation, reflection, scaling, and shearing.

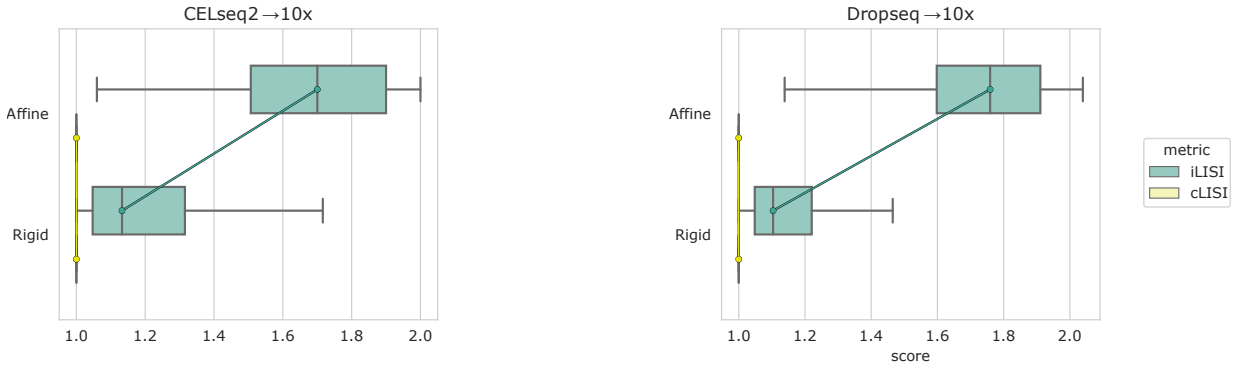


Figure 3.8: Comparison of using a rigid transformation versus an affine transformation in the SCIPR method. These alignment tasks are from the CellBench dataset, the smallest and somewhat easiest dataset. The two subplots use different source batches (both are aligned to the same largest reference batch, 10x). The scores are iLISI (green, batch integration score), and cLISI (orange, cell type mixing score). In each subplot the methods are ordered from top to bottom in order of largest difference (median iLISI - median cLISI) of scores. The center of each box is the median, and whiskers represent 1.5 times the IQR past the low and high quartiles. Circle markers are placed on the medians and connected between boxes with lines of the corresponding color to facilitate visual comparisons. We can see that even on this rather small dataset, the rigid transformation functions are not sufficient to integrate the data well (low iLISI scores), and we see a large gap in iLISI compared with the affine transformation functions.

To learn this function, we minimize an objective function that aims to move the assigned points closer to each other, via such an affine transformation. Given a pair assignment S from the previous step (section "Assigning cells between datasets") (which may be the result of the classic "closest" strategy from ICP, our greedy algorithm, or MNN matching), learning the trans-formation function (Figure 3.10 panel 3) is equivalent to minimizing the loss function given as Equation 3.2. We note that this objective function is not over all pairs of points in sets A and B; it is computed over only those pairs of points selected in S, denoted by the subscript under the sum.

$$\begin{aligned} L(A, B, f_\theta, S) &= \frac{1}{|S|} \sum_{i,j \in S} \frac{1}{d} ||f_\theta(A_i) - B_j||_2^2 \\ &= \frac{1}{|S|} \sum_{i,j \in S} \frac{1}{d} ||(W^T A_i + b) - B_j||_2^2 \end{aligned} \tag{3.2}$$

Runtimes on task *Pancreas: inDrop1→inDrop3*			
Method	Runtime		
	hr	min	sec
ScAlign	1	14	43
SCIPR-mnn	1	7	46
SCIPR-gdy	0	21	59
SCIPR-mnn (gpu)	0	4	22
SeuratV3	0	2	34
MNN	0	1	10
SCIPR-gdy (gpu)	0	0	39

Table 3.7: Runtimes of alignment methods. We ran each alignment method on the same task: aligning cells from the inDrop1 batch (940 cells) to the inDrop3 batch (1488 cells) of the Pancreas dataset, using the 2629 most variable genes. The methods are sorted from top to bottom in order of longest to shortest runtime. All methods were run on the same exact machine, with 2 cores of an Intel Xeon CPU E5-2630 v4 @ 2.20GHz with 16Gb of memory. The SCIPR methods can automatically utilize a GPU if available to accelerate training. SCIPR (gpu) methods utilized an Nvidia GeForce GTX 1080 Ti GPU card.

where $A, B \in \mathbb{R}^d$, and d is the number of genes

This is a least-squares objective function. If the system is overdetermined, this could be solved exactly. However, due to the high dimension we are working in (each point is the expression of thousands of genes), the matrix inversion for the exact solution is expensive to compute, as matrix inversion is $O(d^3)$. To avoid this, we approximate the solution using gradient descent to arrive at our transformation function $f_\theta^{(t)}$ for the current iteration t (see Supporting Methods 3.2.3 for gradient descent settings).

Iterative step

After each of the stages (assignment and transformation function update), we use our learned transformation function at the current iteration $f_\theta^{(t)}$ to transform the all points in A (not just those in the set S from the matching algorithm) (Figure 3.10 panel 4), and then repeat the stages for T iterations (Figure 3.10 panel 5). The final learned transformation of source points A to target points B is a chained series of transformations (composite function) from each iteration (Figure 3.10 panel 6). Since our function class for f_θ is affine transformations, and the composition of affine transformations is itself an affine transformation, we can combine this chain of transformations into a single affine transformation. See Supporting Methods 3.2.3 for details.

In all of our experiments, we ran the iterative point set registration for five iterations. Our experiments (Figure 3.9) indicate that distances between matched cells converge within very few iterations for all three datasets. We note that this is in line with prior work that used the ICP algorithm on image data. For that data ICP also exhibited fast convergence within the first few iterations [157].

The runtime of our iterative algorithm is slower than that of the pure matching-based methods such as MNN and Seurat (Table 3.7), and is similar to the runtime of neural network-based methods such as ScAlign. This is because while MNN and Seurat do not learn a function and must be recomputed to align any new data, our point set registration method and methods like ScAlign aim to learn an alignment function that can generalize and be applied to align new data not seen in the learning process. This comes at the cost of having to optimize an objective function via an iterative learning procedure. However, we note that these methods can utilize graphics processing units (GPUs) to greatly accelerate the process, and we see that our iterative algorithms can be even faster than MNN or Seurat when run with a GPU (Table 3.7).

Validation

A number of methods have been proposed to test the accuracy of alignment based methods [168, 169]. These evaluation metrics try to balance two, sometimes competing, attributes. The first is dataset mixing which is the goal of the alignment. The second is cell type coherence. A method that randomly mixes the two datasets would score high on the first measure and low on the second while a method that clusters each of the datasets very well but cannot match them will score high on the second and not on the first.

To track both dataset mixing and biological signal preservation, we follow [169] and use the local inverse Simpson's Index (LISI). LISI measures the amount of diversity within a small neighborhood around each point in a dataset, with respect to a particular label. The lowest value of LISI is 1 (no diversity). As in [169], we define integration LISI (iLISI) as the score computed when using the batch label for each datapoint, and cell-type LISI (cLISI) as the score when using the cell-type label. iLISI measures the effective number of datasets within the neighborhood (so the higher the better). cLISI measures the effective number cell types within the neighborhood (so the lower the better). See [169] and Supporting Methods 3.2.3 for details on how to compute LISI scores (Equation 3.3). With these two metrics in hand, we can keep track of not only the ability of our algorithms to align one dataset onto another, but also their ability to preserve original signal. In our figures which report the iLISI and cLISI scores, we rank the methods based on the difference of medians $iLISI - cLISI$ score to capture the ability of the methods to maximize and minimize these two quantities respectively.

3.2.3 Supporting Methods

scRNA-seq alignment benchmarking software and data

We created a Python software pipeline to load datasets, specify alignment tasks between batches of these datasets, preprocess data, run different methods of scRNA-seq alignment (including our own), compute iLISI and cLISI alignment scores for each, and visualize the results. We provide this pipeline and data in a separate repository from our SCIPR code. It can be found in our repository at `https://github.com/AmirAlavi/sc-alignment-benchmarking`.

Data preprocessing and filtration

For each dataset used in our experiments, we removed cells which had low detected genes and genes which were not detected in enough cells or had low number of reads. We did this for each batch within each dataset seperately. For the CellBench and Pancreas batches, we used cells which had a minimum of 1,800 non-zero genes (detected genes). For the batches in the PBMC dataset, we used a lower threshold of 250 genes, as that dataset had lower coverage in general. For filtering the genes, we required that each gene had at least 10 reads in the batch, and that it was detected (non-zero) in at least 5 cells in the batch (these same thresholds were used for all batches in all datasets). These gene filtering steps would result in different sets of genes being kept in each batch. Thus, for a dataset, the set of genes we keep is the intersection of those from each batch.

After this process of filtering out cells and genes based on read counts, for our SCIPR method and all other alignment methods we compared to, we further filtered the genes to the set of most highly variable genes for each dataset, across the batches. We used the method used by in the Seurat R package for scRNA-seq analysis [167], and implemented in the scanpy package for python[28] via the function `scanpy.pp.highly variable genes`.

Software and settings for related methods

We compared our method to Mutual Nearest Neighbors (MNN) [87], SeuratV3 [29], and ScAlign [152]. We used the python implementation of MNN (mnnpy) [170] with default settings via the `mnnpy.mnn correct` function. For SeuratV3, we followed the instructions for installation from `https://satijalab.org/seurat/` and used SeuratV3 with the default settings recommended in the "Standard Workflow" vignette. For ScAlign, we downloaded the code from the repository at `https://github.com/quon-titative-biology/scAlign` and for easier integration into our python benchmarking framework, created python wrappers to call the same tensorflow functions in python as the R tensorflow code in their repository. We used the same default parameters as specified in their original code.

Computing the final affine transformation at the end of SCIPR

Since our function class for f_θ is affine transformations, and the composition of affine transformations is itself an affine transformation, we can combine this chain of transformations into a single affine transformation. In this case, we represent $f_\theta(x) = W^T x + b$ as $f_\theta(x) = W'^T x$ where W' is the augmented matrix that adds another dimension to include the translation term b with the linear transformation in a single matrix W'. With this representation in hand, we can update our overall f_θ after each iteration by left multiplying the current f_θ by the latest f_θ learned in the current iteration. In the end after T iterations of the algorithm, our final function is:

$$\begin{aligned} f_\theta &= f_\theta^{(T)} \circ f_\theta^{(T-1)} \circ ... \circ f_\theta^{(1)} \\ &= W'^{(T)} \cdot W'^{(T-1)} \cdot ... \cdot W'^{(1)} \end{aligned}$$

Parameter settings for SCIPR experiments

In all of our experiments, we used the same default parameter settings for our SCIPR methods. We found that these settings were fairly robust to different input datasets, as we saw in our results, but users may want to experiment with different values to achieve the desired level of batch mixing:

- We first normalized the source and target batches by scaling each cell's gene expression vector to unit norm, as is also done in the MNN method [87]
- We ran the algorithm for 5 iterations, as we find fast convergence during the first few iterations (Figure 3.9). This fast early convergence behavior is also reported in the original iterative closest point publication [157]. One could run the algorithm for even longer if desired, but the decrease in mean distances between corresponding cells is usually small after the first few iterations.
- We ran the gradient descent to the learn the affine transform at each iteration for 1000 steps
- We used a learning rate of 1e-3 for the gradient descent
- If MNN was used for the matching algorithm, we used 10 neighbors for computing mutual nearest neigbhors
- If our greedy algorithm, Algorithm 2, was used for the matching, we used parameters $\alpha = 0.5$ and $\beta = 2$

Computing iLISI and cLISI scores

As described in the Methods section 4.8, we use the local inverse Simpson's Index (LISI) metric [169] to evaluate our alignments, which measures the amount of diversity within a small neighborhood around each point in a dataset, with respect to a particular label. A good alignment will mix the batches well, and result in high diversity with respect to the batch label in local neighborhoods. On the other hand, a good alignment will simultaneously preserve biological signal (cellular state), and result in low diversity with respect to the cell type label in local neighborhoods. LISI is computed on a per-cell (per data point) basis as in the expression below:

$$LISI(x_i) = \frac{1}{\sum_{y \in Y} p(y|x_i)^2} \tag{3.3}$$

Where $x_i \in \{x_1, ..., x_N\}$ is the expression vector of the i-th cell in the dataset of size N and Y is the set of unique values of the covariate with respect to which we are computing LISI. For example, when evaluating the integration LISI (iLISI), then Y could be {10x, CELseq2, Dropseq, ...}. If we are evaluating cell-type LISI (cLISI) on the other hand, then Y could be {Cytotoxic T Cell, CD4+ T Cell, B cell, ...}

The function $p(y|x_i)$ is a "relative abundance" of the covariate label y in a local neighborhood around the point x_i. As in [169], we use Gaussian kernel-based distributions of neighborhoods, and use the same fixed perplexity of 30. Our implementation is a direct translation of the R implementation in [169] to Python.

Intuitively, the summation is capturing the expected number of cells needed to be sampled before two are drawn with the same covariate label. If all the cells in the local neighborhood

have the same label, then the expression will be 1 (low diversity). If all cells can have one of two labels with equal probability, then the expression will be 2 (high diversity). In this way, LISI measures the effective number of batches in a local neighborhood [169].

Differential expression analysis

To conduct the differential expression analysis on the PBMC dataset presented in Table 3.8, we did one-vs-rest tests of each cell type vs the other cell types present in a batch. The statistical test used was a non-parameteric Wilcoxon rank sum test, as is used in the latest version of the Seurat package [29]. We used the implementation available in the diffxpy python package for differential gene expression analysis of scRNA-seq data [171] via the function `diffxpy.api.test.rank test`. After computing the adjusted p-value for each gene, via the Benjamini-Hochberg procedure for controlling false discover rate (FDR) [172], we then select the top 500 genes by adjusted p-value for enrichment analysis, as outlined below in Supporting Methods 3.2.3.

Selecting top genes from SCIPR models for enrichment analysis

After fitting a SCIPR model to align a source batch onto a given target batch, we can then analyze the model's parameters to determine which genes it is giving more weight to in its affine transformation. The model's weights are a matrix $W \in \mathbb{R}^{d \times d}$, where d is the number of genes. The rows of the matrix correspond to the input genes, and the columns correspond to the output genes. The dimensions of the matrix are the same, as SCIPR is a transformation from R^d to R^d, i.e. the input genes and the output genes are the same. We determine the importance of each input gene by first taking the absolute value of this matrix, then dividing the diagonal elements by the the sum in each column. This effectively places in each diagonal a normalized measure of the amount of influence that input gene had on computing the output of the transformation, relative to the other input genes. We then select the top 500 of these diagonal elements (where each diagonal element corresponds to a gene) for enrichment analysis as in Supporting Methods 3.2.3.

Gene set enrichment analysis

Given a set of genes of interest (e.g. either top differentially expressed genes as in Supporting Methods 3.2.3 or most highly weighted genes from the SCIPR model as in Supporting Methods 3.2.3), we would like to compare them to annotated sets of genes to see if there is significant overlap, or enrichment. We compare our sets of genes to sets in the "Biological Process" domain of the Gene Ontology (GO) [34, 173]. We conducted the hypergeometric test for over-representation to quantify the significance of overlap between the gene sets and used the Benjamini-Hochberg procedure as above [172], using the implementation in the diffxpy package [171] via the function `diffxpy.api.enrich.test`. We specified the background set to be the full set of genes after the read count filtration outlined in Supporting Methods 3.2.3. In addition, we specified that the annotated gene sets from GO first be filtered to only include genes that exist in our background set.

3.2.4 Results

Method and benchmarking overview

We developed SCIPR which aligns two batches of scRNA-seq data (termed source and target) using methods motivated by point set registration algorithms. SCIPR first identifies corresponding pairs of cells between source and target batches (Figure 3.10 panel 1). Rather than using the closest cell (as defined by euclidean distance) in the target to match a source cell, SCIPR uses either of two matching algorithms to account for the heterogeneity and noise in scRNA-seq data: Mutual Nearest Neighbors (MNN) matching [87], and a novel greedy matching algorithm (Algorithm 2, Methods). Once a pairing of cells is established (Figure 3.10 panel 2), a transformation function is learned to transform source cells so that they are closer to their matched target cell (Figure 3.10 panel 3). To allow for accurate alignment of high-dimensional scRNA-seq data, we replace the rigid transformation commonly used for point cloud registration with affine transformations. After fitting the transformation function (Methods), we apply it to the source cells (Figure 3.10 panel 4), and iteratively repeat the process until convergence. The final alignment function we learn is a composition of the transformation functions learned at each iteration (Methods) (Figure 3.10 panels 5,6).

We used three datasets to test and compare two versions of SCIPR to prior alignment methods (Methods). See Supporting Methods 3.2.3 for software settings of the related methods. These comparisons were performed by testing the methods on several "alignment tasks". An alignment task is defined by:

- A dataset (e.g. Pancreas)
- A source batch A within that dataset, which you would like to transform (e.g. inDrop1)
- A target batch B within that dataset, which you would like to transform A onto (e.g. inDrop 3)

For example, an alignment task can be summarized with the notation: *Pancreas: inDrop1* $\rightarrow$ *inDrop3*. In the comparisons we performed we fix the target within a dataset to be the largest batch in that dataset. We define these tasks as pairwise alignments, but we note that it is possible to use our method to align multiple batches using SCIPR (Figure 3.11). We scored the performance of the methods using local inverse Simpson's Index (LISI) in which higher integration LISI (iLISI) is better and lower cell-type LISI (cLISI) is better [169] (Methods).

An affine global transformation function yields well-mixed alignments

We first evaluated the ability of SCIPR and other methods to integrate pairs of batches from three different datasets. Results for 8 alignment tasks in three datasets are presented in Figure 3.12. As the figure shows, for 7 of the 8 alignment tasks the two version of SCIPR ranked at the top. SCIPR-mnn was the overall top performer ranking first on 4 tasks and 2nd on 2 whereas SCIPR-gdy ranked first on 3 tasks and 2nd on 1. The only other method that performed well is ScAlign which ranked first on 1 task and 2nd on 4. For example, for the CellBench alignment tasks (first row of Figure 3.12), we see that SCIPR-mnn, which uses the MNN matching for the cell pair assignment stage, has consistent better performance, and achieves high batch mixing (1.70 and 1.76 median iLISI scores on *CELseq2*$\rightarrow$*10x* and *Dropseq*$\rightarrow$*10x* respectively) with very little cell

type mixing (1.00 median iLISI score on both *CELseq2→10x* and *Dropseq→10x*). When looking at the same dataset, on the *CELseq2→10x* task the other methods such as ScAlign (iLISI: 1.00, cLISI: 1.00) or SeuratV3 (iLISI: 1.51, cLISI: 1.00) are also able to avoid cell type mixing, but are not able to mix the batches as much as SCIPR (Figure 3.12). Full alignment quantitative scores for these tasks and all others in the paper are listed in Table S7. These quantitative metrics are also corroborated by a qualitative assessment of the resulting t-SNE embeddings (Figure 3.13). There we can see that both SCIPR-gdy and SCIPR-mnn (top two rows) mix the batches well (1st and 3rd columns) compared to methods like MNN and ScAlign while successfully keeping cell types separate (2nd and 4th columns). SeuratV3 also performs well. The embeddings on the Pancreas dataset (Figure 3.14) show that most methods result in embeddings with these desirable properties on this dataset, including SCIPR.

SCIPR robustly mixes batches with non-overlapping cell types

The comparisons presented above involved sources and target batches with the same set of cells. However, in practice it is often unknown if both source and target indeed contain the same cell types. To test the robustness of SCIPR and other methods for such realistic scenarios we hold-out a complete cell type from the target set B in each of the alignment tasks from the section "An affine global transformation function yields well-mixed alignments". As the figures show, for these alignment tasks SCIPR is able to mix batches well, while keeping the median cell type mixing (iLISI) score low, though with a longer tail (Figures 3.15, 3.16, 3.17). For example, for the *CellBench: CELseq2→10x (H1975 cell type held-out from target)* task, SCIPR-mnn had median iLISI and cLISI scores of 1.63 and 1.02 respectively while the second best method, SeuratV3, had iLIS and cLISI scores of 1.49 and 1.01 respectively (Figure 3.15). On the other hand, for the task *Pancreas: inDrop1→inDrop3 (acinar cell type held-out from target)*, SCIPR-mnn achieves a higher median batch mixing score of iLISI=1.69 compared to ScAlign's score of 1.57, but also mixes the cell types slightly more (SCIPR-mnn median cLISI score: 1.28, ScAlign median cLISI score:1.00) (Figure 3.16).

SCIPR generalizes to unseen data

One of the advantages of SCIPR compared to most previous methods is the fact that it learns a general transformation function that can be applied to additional data when it becomes available (Table 3.5). Such a function allows researchers to "fix" a specific setting rather than have all results completely change when new data is introduced. To test the use of the learned transformation function for unseen cell types in the *source dataset* we repeated our analysis, this time holding out a complete cell type from the source set in each alignment task. We next learned the transformation based on the available data and then applied the learned function to the held out data to evaluate the batch and cell type mixing. Results are presented in Figures 3.18, 3.19, 3.20, and 3.21. As the figures show, the transformation learned by SCIPR allows it to keep cell types distinct, even for the unseen source cell type, while also being able to mix the batches of unseen cell types. This is evident in the high median iLISI (1.69) and low median cLISI (1.04) scores of SCIPR-gdy on the task *PBMC:10x Chrom. (v2) A→10x Chrom. (v2) (CD4+ T cell held-out from source)*, where the model is fit without seeing CD4+ T cells in the source set, but is then

GO term	Corrected p-val	intersection	reference	enquiry	background
COTRANSLATIONAL PROTEIN TARGETING TO MEMBRANE	0.000000e+00*	76	96	500	10518
CELLULAR AMIDE METABOLIC PROCESS	0.000000e+00*	122	743	500	10518
ORGANIC CYCLIC COMPOUND CATABOLIC PROCESS	0.000000e+00*	99	451	500	10518
TRANSLATIONAL INITIATION	9.709302e-10	91	174	500	10518
INTRACELLULAR PROTEIN TRANSPORT	3.188832e-09	107	885	500	10518
DEFENSE RESPONSE	3.188832e-09	92	906	500	10518
PROTEIN TARGETING	3.188832e-09	91	340	500	10518
LYMPHOCYTE ACTIVATION	3.188832e-09	72	473	500	10518
T CELL ACTIVATION	3.188832e-09	51	317	500	10518
ADAPTIVE IMMUNE RESPONSE	3.188832e-09	44	273	500	10518
INTRACELLULAR TRANSPORT	3.188832e-09	123	1306	500	10518
PEPTIDE BIOSYNTHETIC PROCESS	3.188832e-09	114	553	500	10518
REGULATION OF CELL ACTIVATION	3.411453e-09	54	356	500	10518
RIBOSOME BIOGENESIS	3.411453e-09	43	252	500	10518
CELLULAR MACROMOLECULE LOCALIZATION	3.411453e-09	125	1331	500	10518
POSITIVE REGULATION OF IMMUNE SYSTEM PROCESS	3.411453e-09	76	701	500	10518
PROTEIN LOCALIZATION TO MEMBRANE	3.411453e-09	83	418	500	10518
CELLULAR MACROMOLECULE CATABOLIC PROCESS	3.411453e-09	111	887	500	10518
REGULATION OF T CELL ACTIVATION	3.411453e-09	37	213	500	10518
POSITIVE REGULATION OF CELL ACTIVATION	3.411453e-09	41	231	500	10518

Table 3.8: Gene enrichment analysis of Differential Expression results. Differential expression results are from testing for differentially expressed genes in CD4+ T cells in the "10x Chromium (v2) A" batch of the PBMC dataset (CD4+ T cells vs. all other cell types). GO terms are from the "Biological Process" domain.
* The p-value was non-zero but very small and within floating point precision equal to zero.

used to transform the full source set in evaluation (Figure 3.21). Figure 3.18 displays the aligned results for the cell type not used in the learning. As the figure shows, for CD4+ and Cytotoxic T cells SCIPR-gdy is able to mix the two batches even though it had never seen these in fitting.

SCIPR identifies biologically relevant genes

The above results demonstrate SCIPR's ability to integrate batches quantitatively and qualitatively. Since SCIPR achieves these results by learning a transformation function that places different weights on different genes, we next asked whether the learned weights provide information on the importance of specific genes for the set of cells being studied. Since SCIPR aims to align specific pairs of cells (one from each batch), when it is successful it tends to focus more strongly on cell type-specific genes. As we showed, for several datasets the method is indeed correct in the assignments it identifies and for these, the set of genes it uses may be of relevance for the cell types it aligns (Sections "An affine global transformation function yields well-mixed alignments" and "SCIPR robustly mixes batches with non-overlapping cell types").

To evaluate this idea we compared ranking genes based on their SCIPR coefficients to a baseline that ranks them based on differential expression (DE) (Supporting Methods 3.2.3, 3.2.3). Next we performed gene set enrichment analysis using the Gene Ontology to identify the significant functions associated with top genes and test their relevance (Supporting Methods 3.2.3). The PBMC dataset, which is the largest, was also the one with the most number of significant categories identified (Table 3.8). When comparing top ranked genes by SCIPR and DE for the "PBMC: 10x Chrom. (v2) A $\rightarrow$ 10x Chrom. (v2)" alignment task we observed that SCIPR genes significantly overlapped with much more relevant terms when compared to DE genes for the same dataset (Tables 3.9 and 3.10 for SCIPR-mnn and SCIPR-gdy respectively). For exam-

GO term	Corrected p-val	intersection	reference	enquiry	background
DEFENSE RESPONSE	9.743408e-09	95	906	500	10518
REGULATION OF IMMUNE RESPONSE	9.743408e-09	75	676	500	10518
HUMORAL IMMUNE RESPONSE	9.743408e-09	26	95	500	10518
LYMPHOCYTE ACTIVATION	9.743408e-09	61	473	500	10518
REGULATION OF CELL ACTIVATION	2.599720e-08	49	356	500	10518
REGULATION OF IMMUNE SYSTEM PROCESS	2.599720e-08	99	981	500	10518
B CELL MEDIATED IMMUNITY	2.599720e-08	22	80	500	10518
POSITIVE REGULATION OF IMMUNE SYSTEM PROCESS	3.605960e-08	74	701	500	10518
ADAPTIVE IMMUNE RESPONSE	3.605960e-08	41	273	500	10518
INNATE IMMUNE RESPONSE	4.770427e-08	65	580	500	10518
REGULATION OF LYMPHOCYTE ACTIVATION	5.137285e-08	43	299	500	10518
HUMORAL IMMUNE RESPONSE MEDIATED BY CIRCULATIN...	1.173927e-07	13	29	500	10518
B CELL RECEPTOR SIGNALING PATHWAY	4.539188e-07	17	58	500	10518
IMMUNE RESPONSE REGULATING CELL SURFACE RECEPT...	7.247366e-07	42	318	500	10518
LYMPHOCYTE MEDIATED IMMUNITY	8.827507e-07	29	171	500	10518
B CELL ACTIVATION	1.007833e-06	30	183	500	10518
CELL ACTIVATION	2.547541e-06	83	922	500	10518
IMMUNE EFFECTOR PROCESS	3.600901e-06	76	822	500	10518
REGULATION OF B CELL ACTIVATION	3.600901e-06	21	102	500	10518
LEUKOCYTE MIGRATION	3.600901e-06	33	229	500	10518

Table 3.9: Gene enrichment analysis of model weights from SCIPR-mnn. Model weights are fit to align cells (unsupervised) from the "10x Chromium (v2) A" batch to the "10x Chromium (v2)" batch.

ple, the top three categories for top ranked SCIPR-mnn genes are "Defense response", "Regulation of immune response", and "Humoral immune response" (all with adj. p-value 9.743e-9, Table 3.9). These categories are very relevant for blood cells given their immune system function. On the other hand, the top three categories recovered by top DE genes are much more generic and include "Cotranslational protein targeting to membrane", "Cellular amide metabolic process", and "Organic cyclic compound catabolic process" (Table 3.8). We compared the enrichment results obtained from using top ranked genes based on SCIPR to results obtained using highly variable genes (HVG). While HVG's do result in enrichment of relevant GO terms (Table 3.11), the significance of enriched relevant categories is lower when compared to genes ranked by SCIPR-mnn.

3.2.5 Discussion

We presented SCIPR which extends point set registration for the alignment of scRNA-Seq data. SCIPR combines many of the desirable features of previous methods including the fact that its unsupervised, generalizable, and keeps the original (gene space) representation. Analysis of several datasets show that SCIPR successfully aligns scRNA-Seq data improving upon other methods proposed for this task. When data is missing from either the source or the target the transformation function learned by SCIPR can be used to accurately align it when it becomes available. Finally, the coefficients learned by SCIPR provide valuable information on the key genes related to the cells being analyzed.

Framing scRNA-seq alignment as a point set registration problem opens the door to applying many of the developments and advancements in that area to scRNA-seq alignment. Point set registration is a mature area that has been widely used for more than two decades. As part of this researchers looked at several different types of transformation functions, data filtration, outlier

GO term	Corrected p-val	intersection	reference	enquiry	background
INNATE IMMUNE RESPONSE	0.000010	61	580	500	10518
HUMORAL IMMUNE RESPONSE MEDIATED BY CIRCULATIN...	0.000010	12	29	500	10518
B CELL MEDIATED IMMUNITY	0.000010	19	80	500	10518
COMPLEMENT ACTIVATION	0.000013	12	31	500	10518
DEFENSE RESPONSE	0.000018	81	906	500	10518
HUMORAL IMMUNE RESPONSE	0.000018	20	95	500	10518
REGULATION OF IMMUNE SYSTEM PROCESS	0.000022	85	981	500	10518
REGULATION OF LYMPHOCYTE ACTIVATION	0.000022	38	299	500	10518
LYMPHOCYTE ACTIVATION	0.000022	51	473	500	10518
LYMPHOCYTE MEDIATED IMMUNITY	0.000023	27	171	500	10518
REGULATION OF CELL ACTIVATION	0.000027	42	356	500	10518
NEUTROPHIL MIGRATION	0.000084	14	54	500	10518
POSITIVE REGULATION OF IMMUNE SYSTEM PROCESS	0.000142	64	701	500	10518
RESPONSE TO BIOTIC STIMULUS	0.000361	52	539	500	10518
REGULATION OF IMMUNE RESPONSE	0.000361	61	676	500	10518
LEUKOCYTE CELL CELL ADHESION	0.000365	29	224	500	10518
DEFENSE RESPONSE TO BACTERIUM	0.000390	18	100	500	10518
PHAGOCYTOSIS RECOGNITION	0.000468	8	19	500	10518
GRANULOCYTE MIGRATION	0.000501	14	64	500	10518
POSITIVE REGULATION OF LYMPHOCYTE ACTIVATION	0.000501	27	205	500	10518

Table 3.10: Gene enrichment analysis of model weights from SCIPR-gdy. Model weights are fit to align cells (unsupervised) from the "10x Chromium (v2) A" batch to the "10x Chromium (v2)" batch.

GO term	Corrected p-val	intersection	reference	enquiry	background
B CELL MEDIATED IMMUNITY	0.000004	20	80	500	10518
LYMPHOCYTE ACTIVATION	0.000005	54	473	500	10518
LYMPHOCYTE MEDIATED IMMUNITY	0.000019	28	171	500	10518
CELL ACTIVATION	0.000022	82	922	500	10518
RESPONSE TO BACTERIUM	0.000022	39	306	500	10518
REGULATION OF LYMPHOCYTE ACTIVATION	0.000029	38	299	500	10518
DEFENSE RESPONSE	0.000029	80	906	500	10518
B CELL ACTIVATION	0.000033	28	183	500	10518
RESPONSE TO BIOTIC STIMULUS	0.000042	55	539	500	10518
ADAPTIVE IMMUNE RESPONSE BASED ON SOMATIC RECO...	0.000048	27	177	500	10518
ADAPTIVE IMMUNE RESPONSE	0.000048	35	273	500	10518
INNATE IMMUNE RESPONSE	0.000061	57	580	500	10518
LEUKOCYTE PROLIFERATION	0.000061	27	181	500	10518
IMMUNOGLOBULIN PRODUCTION INVOLVED IN IMMUNOGL...	0.000099	12	40	500	10518
REGULATION OF CELL ACTIVATION	0.000155	40	356	500	10518
HUMORAL IMMUNE RESPONSE	0.000177	18	95	500	10518
POSITIVE REGULATION OF LYMPHOCYTE ACTIVATION	0.000177	28	205	500	10518
IMMUNE EFFECTOR PROCESS	0.000181	71	822	500	10518
LYMPHOCYTE ACTIVATION INVOLVED IN IMMUNE RESPONSE	0.000360	20	121	500	10518
B CELL ACTIVATION INVOLVED IN IMMUNE RESPONSE	0.000562	13	56	500	10518

Table 3.11: Gene enrichment analysis of highly variable genes. Enrichment of random subset of 500 of the 1466 highly variable genes in the PBMC dataset, versus a background of all genes assayed in the dataset. This experiment is a baseline, showing the enrichment present simply due to filtering to highly variable genes. The number of genes (500) was chosen to be the same as the number chosen in Supporting Methods 3.2.3 and 3.2.3 to allow for fair comparison.

handling, and association mapping, all of which may find applications in scRNA-seq analysis.

When evaluating SCIPR and prior methods we used the local inverse Simpson's Index (LISI) to quantify both cell type mixing and batch mixing. This leads to two values for each alignment task which can be combined for ranking the different methods by computing the difference of the medians $iLISI - cLISI$. Such ranking places equal weight on both issues. However, this score may not tell the whole story since some methods may be much better at one task vs. the other. For example, while SCIPR was ranked as the top method for most of the comparisons we performed, it has a tendency of to sacrifice some cell type separation in order to achieve greater batch mixing. Thus, depending on the user priorities between cell type and batch mixing, different methods may be more attractive even if the combined score is lower when compared to other methods.

With regards to choosing between SCIPR-mnn and SCIPR-gdy, we generally recommend the use of the MNN strategy which performs well and does not rely on hyper-parameters related to the expected matching between the datasets (thus, requires less assumptions). However, in cases where the user has prior beliefs or expectations on what portion of the cells should be matched between the two batches (for example, if s/he knows that some types are missing in one sample and so the match proportion is not expected to be high) then we recommend using the greedy algorithm, since such information can be incorporated using its hyper-parameters and will thus lead to better results. We tested SCIPR using a set of highly variable genes (see "Dataset selection" in Methods and Supporting Methods 3.2.3). Previous methods also recommend using highly variable genes [29]. Using such set limits the application of our alignment transformation to a rather small subset of genes, and so may have implications for downstream analysis. We note that SCIPR can be run with more genes, though performance can suffer a bit non variable genes are included (Figure 3.22), only showing a slight decrease in alignment performance in some cases. While the set of highly variable genes alone does result in enrichment for relevant terms (Table 3.11), the significance of this enrichment is not as strong as the results from our SCIPR-mnn model weights analysis.

While SCIPR performed best in our analysis, there are a number of ways in which it can be further improved. As mentioned above, SCIPR tends to weight batch mixing higher than cell type separation. A possible way to overcome this would be to add a regularization term to the transformation function to increase the weight of high scoring matches. Another option is to explore the use of non-linear transformations with strong customized regularization. A potential drawback of SCIPR is the fact that it relies on affine transformations that do not handle non-linear differences between batches. While this leads to a reduction in the number of parameters that the method uses, and so reduces overfitting, it also means that the method cannot handle non-linear changes which may occur between different batches.

SCIPR is implemented as a Python package, with documentation, installation instructions, and source code available at `https://scipr.readthedocs.io`, and our benchmarking pipeline and data used are available at `https://github.com/AmirAlavi/sc-alignment-benchmarking` (see Supporting Methods 3.2.3).

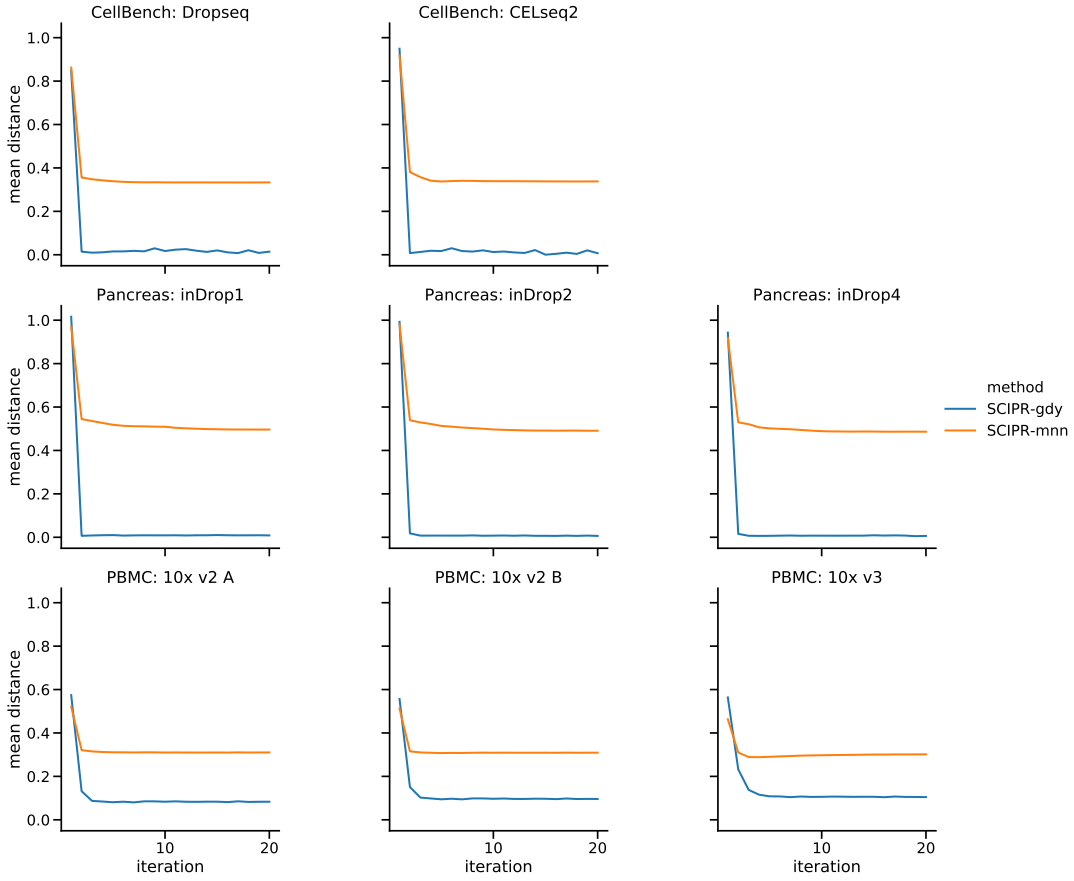


Figure 3.9: Convergence during fitting of SCIPR models. Each row of subplots are tasks from the same dataset, where each column uses a different source batch (all are aligned to the same largest reference batch, 10x for CellBench, inDrop3 for Pancreas, and 10x v2 for PBMC). The values plotted are the mean distances between the selected pairs of points after each iteration of the algorithm. We can see that both SCIPR-gdy and SCIPR-mnn do indeed converge to a local optimum within the first few iterations. Fast convergence within the first few iterations is expected for Iterative Closest Points-based algorithms [157]. This supports our choice to run the SCIPR methods for 5 iterations in the experiments we present in our work.

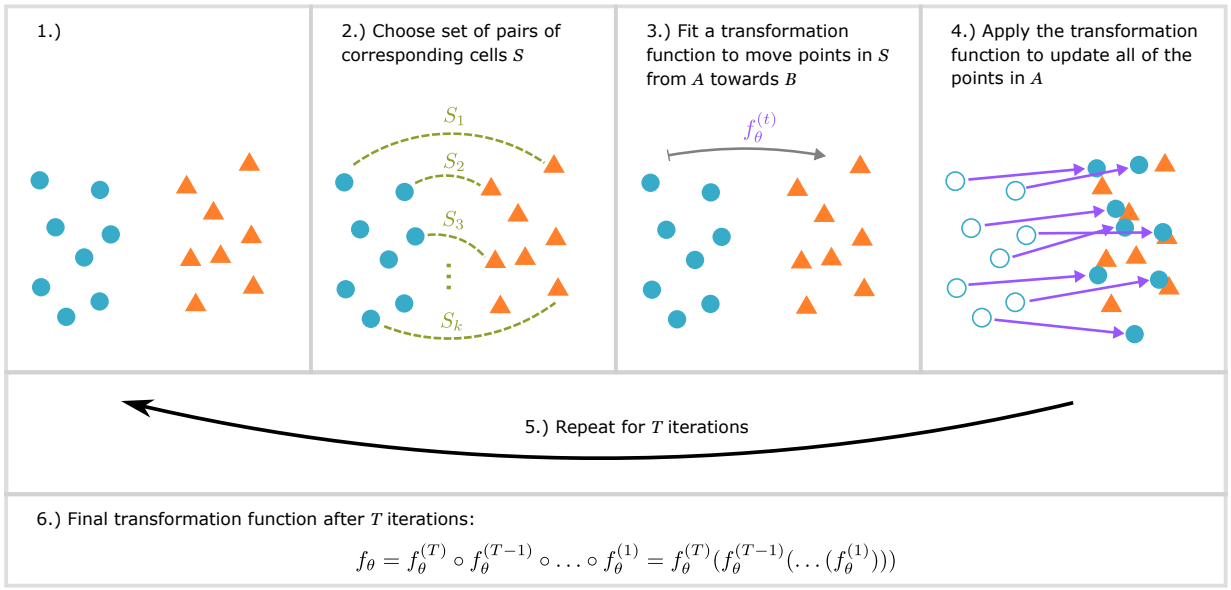


Figure 3.10: Summary of steps in iterative point set registration for scRNA-seq data. Each cell in an scRNA-seq dataset can be viewed as a point in high dimensional space. 1) We start with two unaligned batches (sources, blue and targets, orange). 2) A matching algorithm (e.g. picking the closest corresponding point, or using mutual nearest neighbors) is used to pair source cells from A with a corresponding target cell in B. The number of source and/or target cells matched can vary for different matching strategies. 3) Based on the selected pairs, a global transformation function is learned so that source cells in A become closer to their paired cell in B. 4) The learned transformation is next applied to all points in A. 5) This process (steps 2-4) is repeated, iteratively aligning set A onto B until the mean distance between the assigned pairs of cells no longer improves. 6) The final global transformation function is the composition of the functions learned in each iteration at step 3.

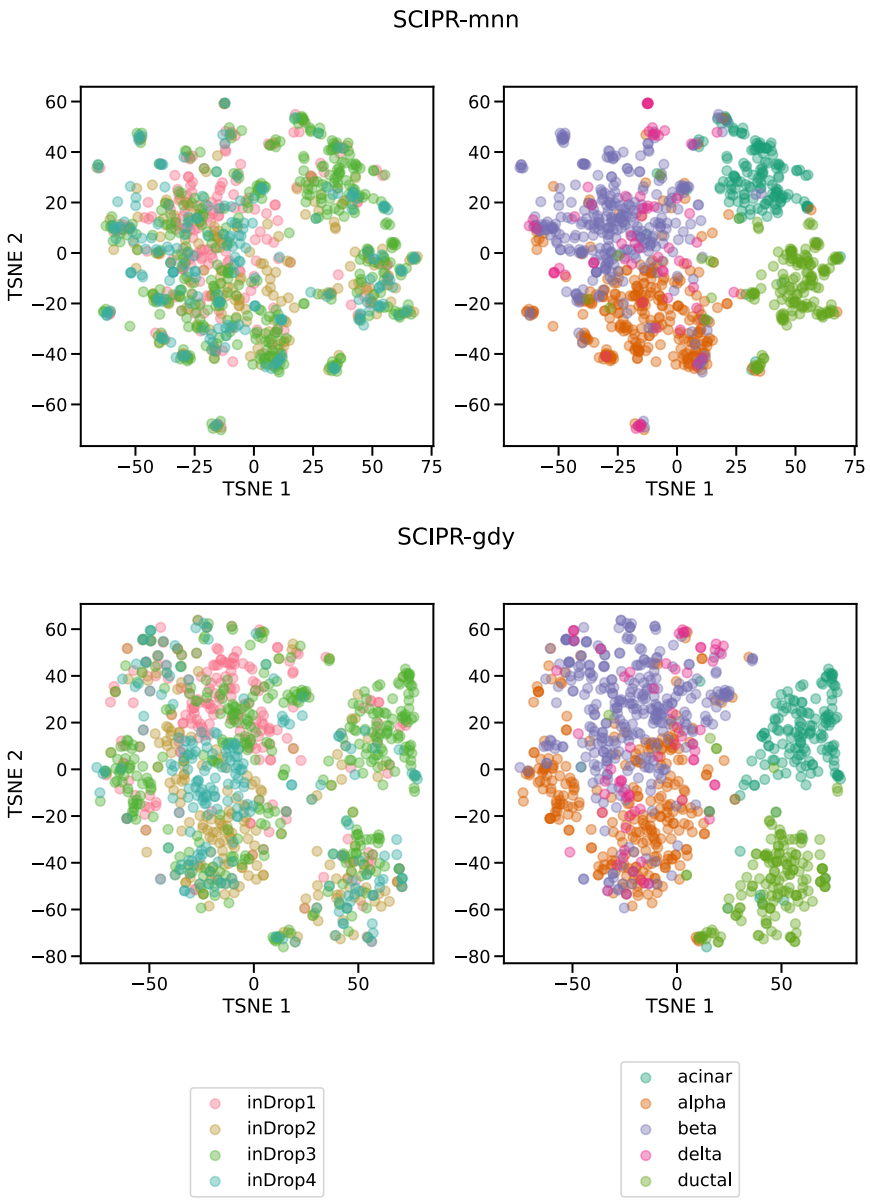


Figure 3.11: Example of aligning multiple batches to a reference batch using SCIPR. To see how SCIPR can be used to align multiple batches to a reference batch, we aligned each of the inDrop1, inDrop2, and inDrop4 batches to the inDrop3 batch (the largest batch) in the Pancreas dataset. These alignments were done independently, as pairwise alignments, and visualized together in the figure. In the subplots on the left, each point (cells) is colored by batch, and on the right they are colored by cell type. This straightforward multiple alignment strategy shows that it is possible to align many different batches to a single reference batch using SCIPR which results in coherent cell type representations while mixing the batches well.

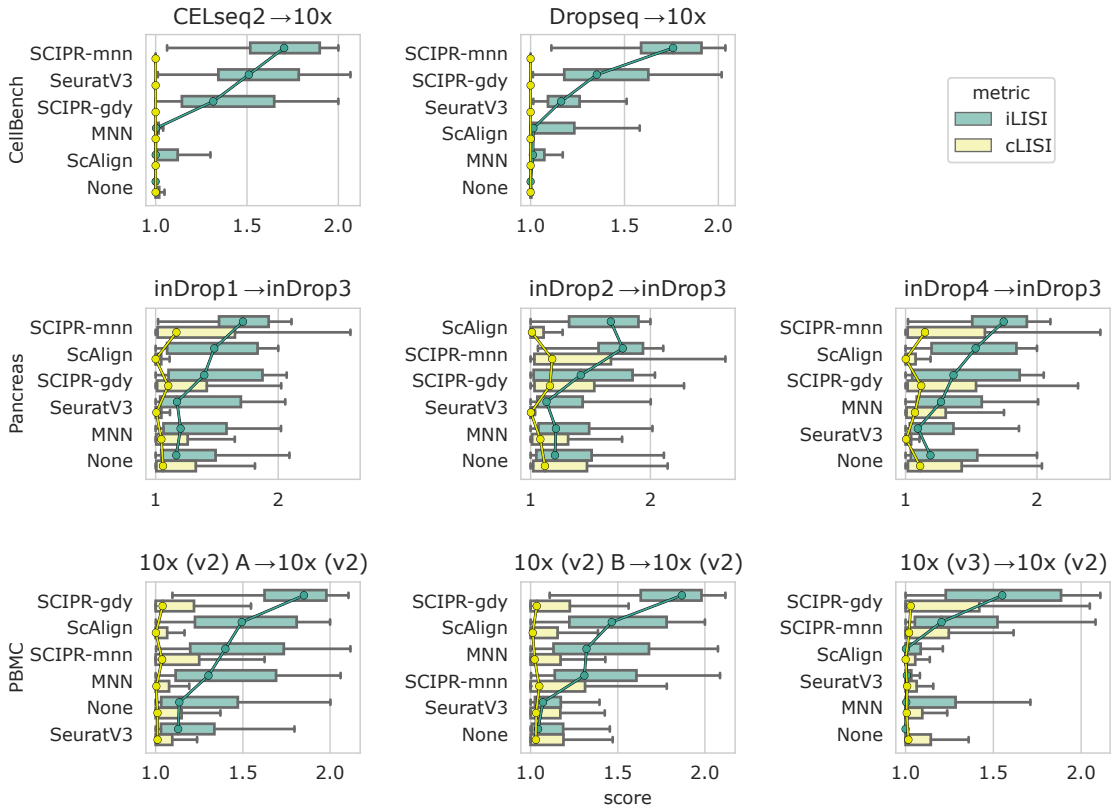


Figure 3.12: Quantitative scoring of alignment methods on benchmark datasets. Each row of subplots are tasks from the same dataset, where each column uses a different source batch (all are aligned to the same largest reference batch). The scores are iLISI (green, batch integration score), and cLISI (yellow, cell type mixing score). In each subplot the methods are ordered from top to bottom in order of largest difference (median iLISI - median cLISI) of scores. "None" means no alignment method is applied to the data. The center of each box is the median, and whiskers represent 1.5 times the IQR past the low and high quartiles. Circle markers are placed on the medians and connected between boxes with lines of the corresponding color to facilitate visual comparisons.

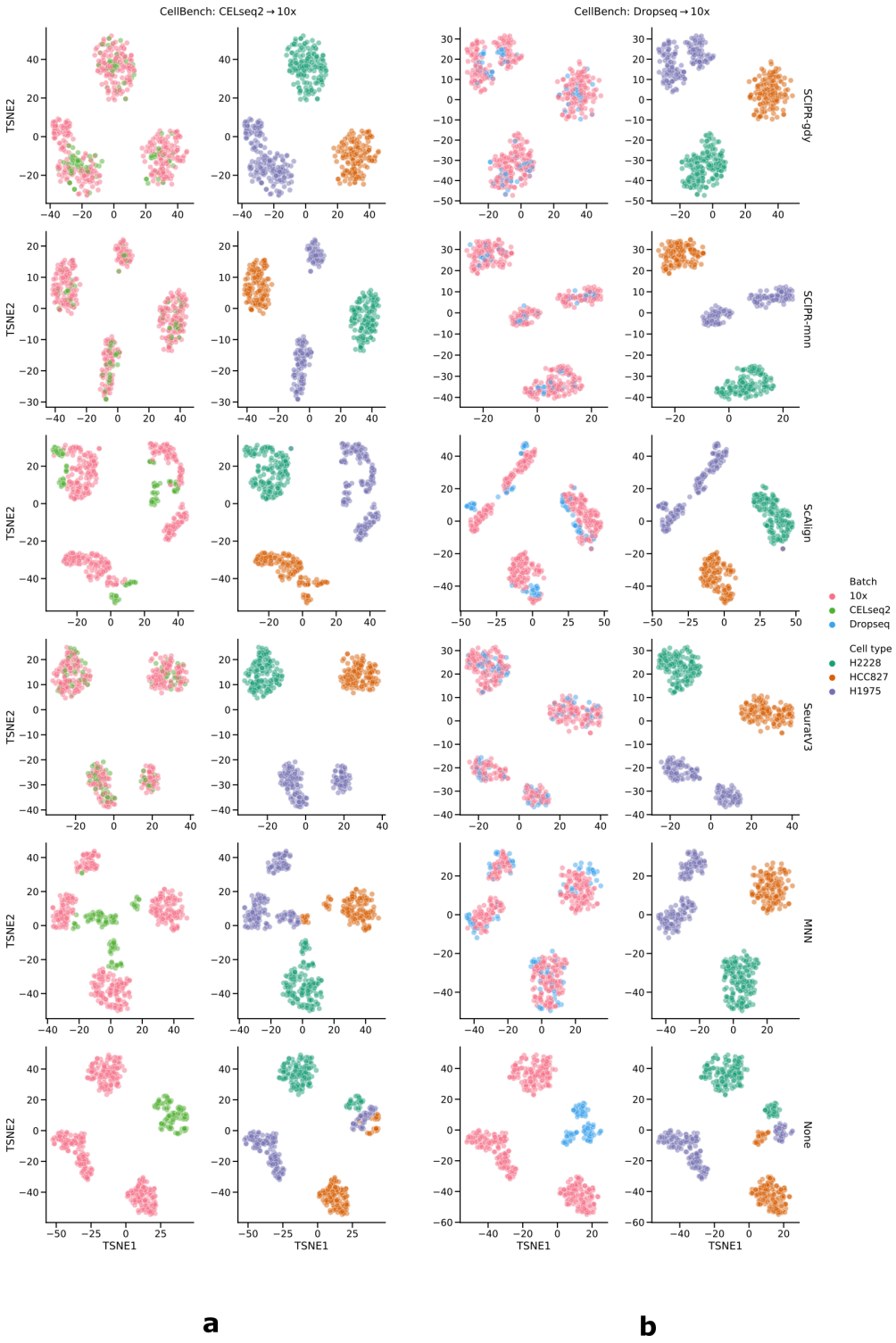


Figure 3.13: Embedding (t-SNE) visualization from alignment tasks on the CellBench dataset using various alignment methods. Each row is a different alignment method (the bottom row ,"None", is with no alignment). The columns are in two groups based on alignment task: the left two columns (a) pertain to aligning the CELseq2 batch onto the 10x batch, the right two columns (b) are for aligning the Dropseq batch onto the 10x batch. The first and third columns are colored by batch, and the second and fourth columns are colored by cell type.

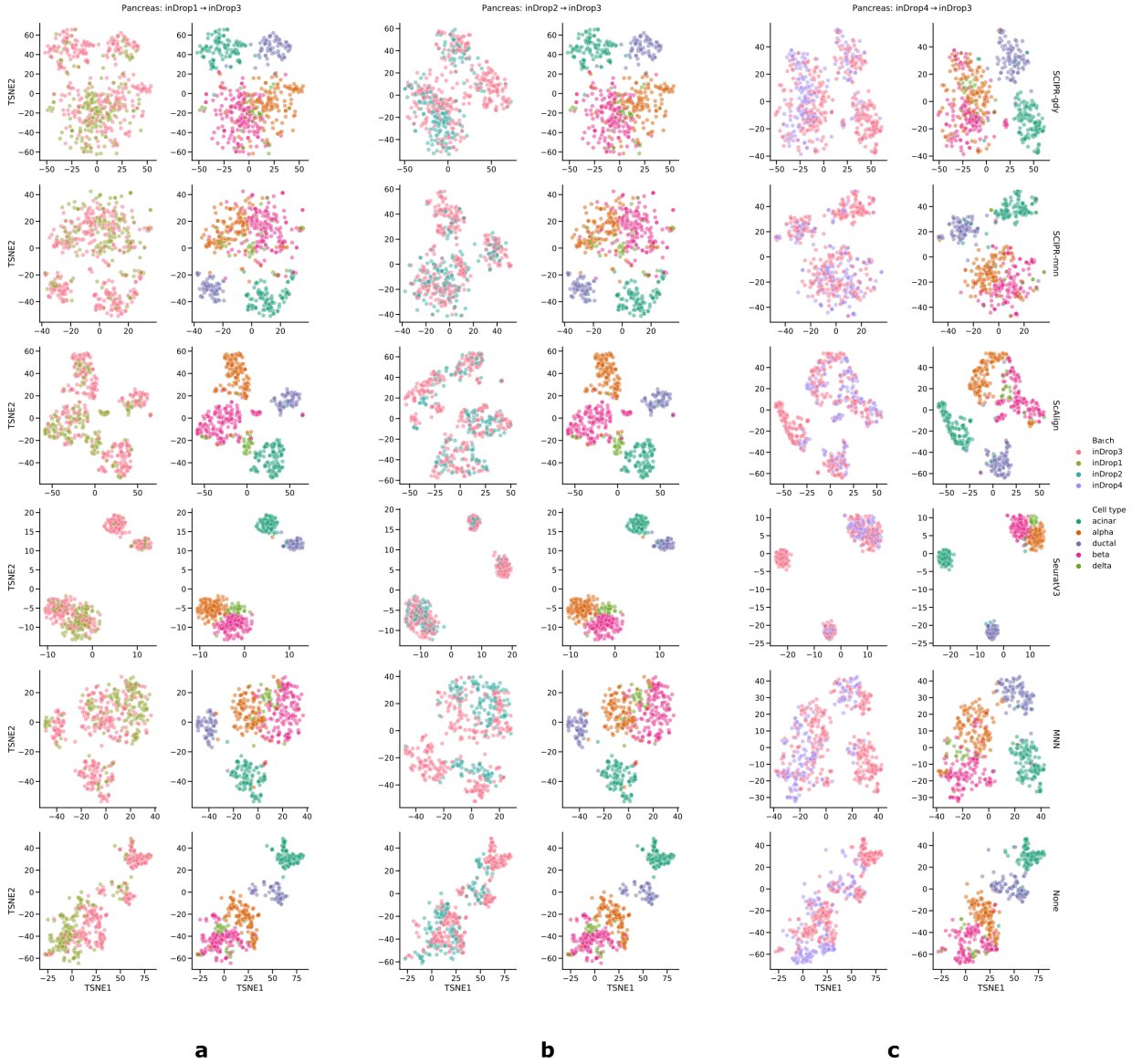


Figure 3.14: Embedding (t-SNE) visualization from alignment tasks on the Pancreas dataset using various alignment methods. Each row is a different alignment method (the bottom row,"None", is with no alignment). The columns are in three groups based on alignment task: the left two columns (a) pertain to aligning the inDrop1 batch onto the inDrop3 batch, the middle two columns (b) are for aligning the inDrop2 batch onto the inDrop3 batch, and the right two columns (c) are for aligning the inDrop4 batch onto the inDrop3 batch. The first, third, and fifth columns are colored by batch, and the second, fourth, and sixth columns are colored by cell type.

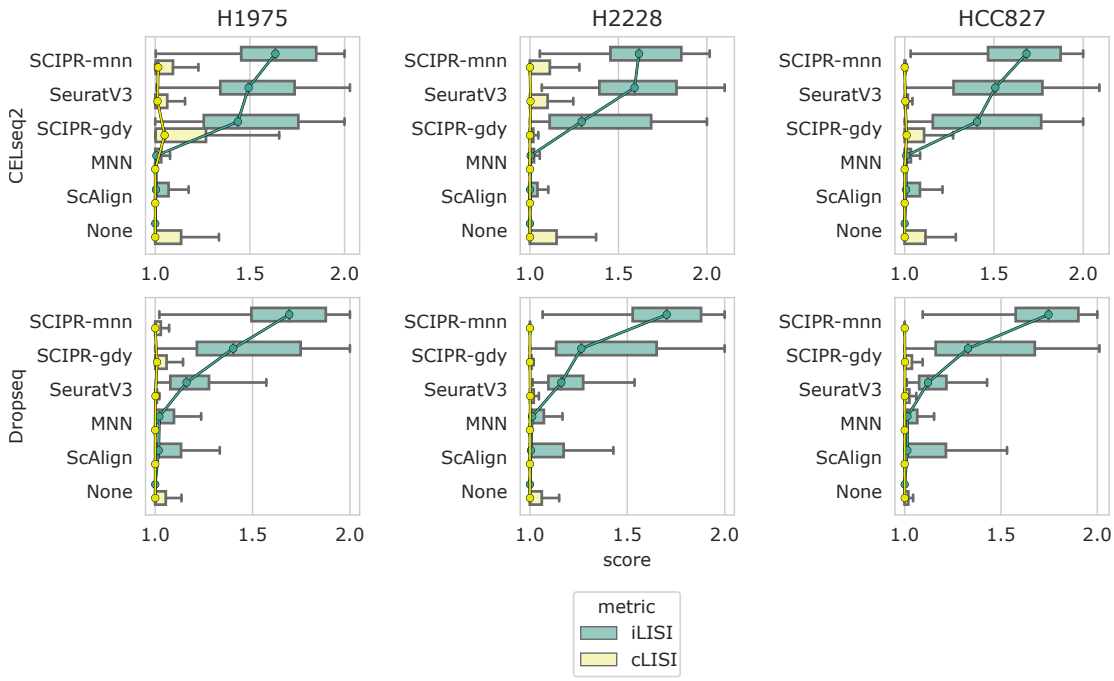


Figure 3.15: Quantitative scoring of alignment methods on the CellBench dataset with a cell type held out from the target set. Each row of subplots are alignment tasks with the same source batch, where each column uses a different cell type as a hold-out from the target set (the 10x batch). Box plot computation and ordering of methods in each subplot is determined in the same fashion as in Figure 3.12.

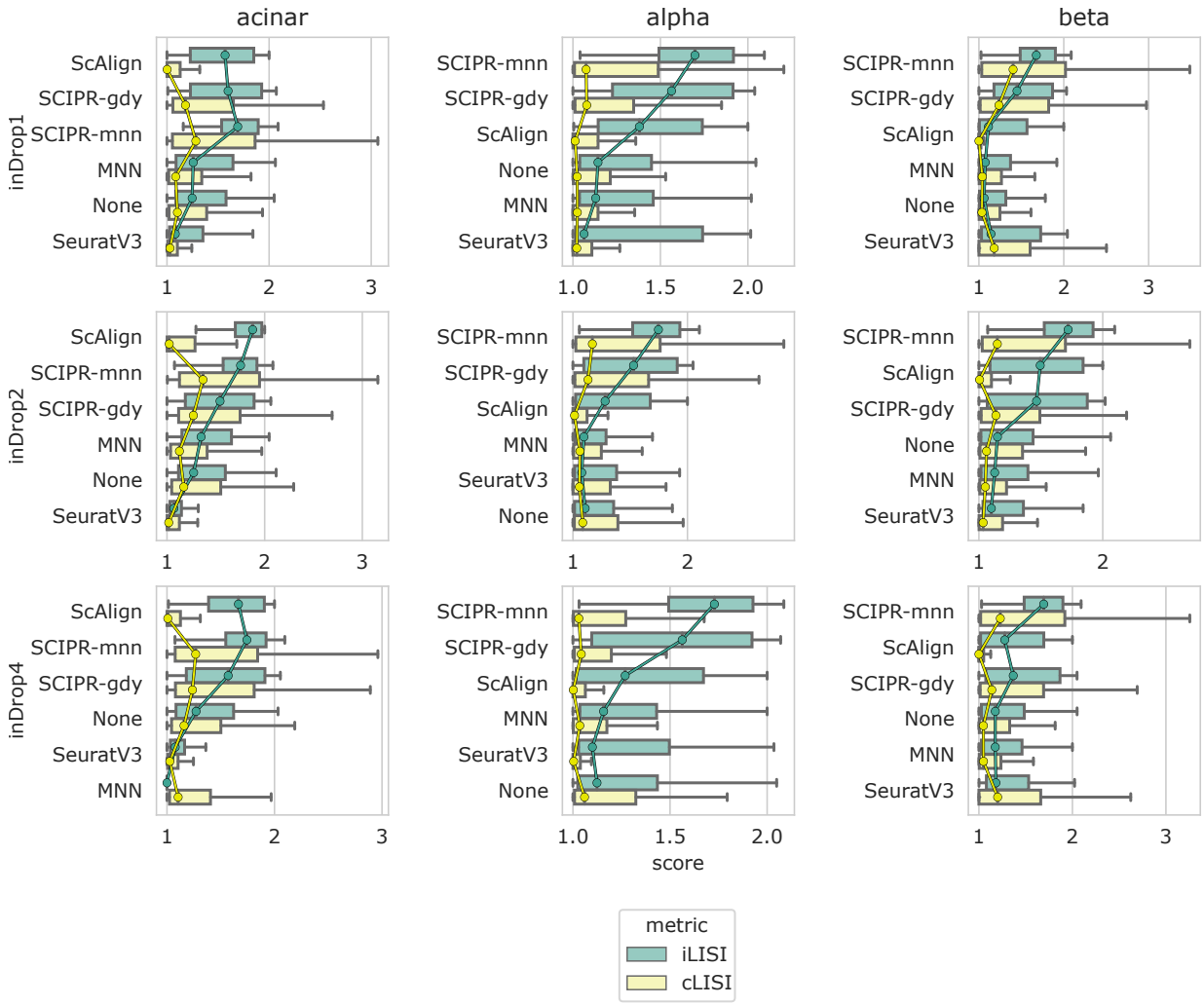


Figure 3.16: Quantitative scoring of alignment methods on the Pancreas dataset with a cell type held out from the target set. Each row of subplots are alignment tasks with the same source batch, where each column uses a different cell type as a hold-out from the target set (the inDrop3 batch). Box plot computation and ordering of methods in each subplot is determined in the same fashion as in Figure 3.8.

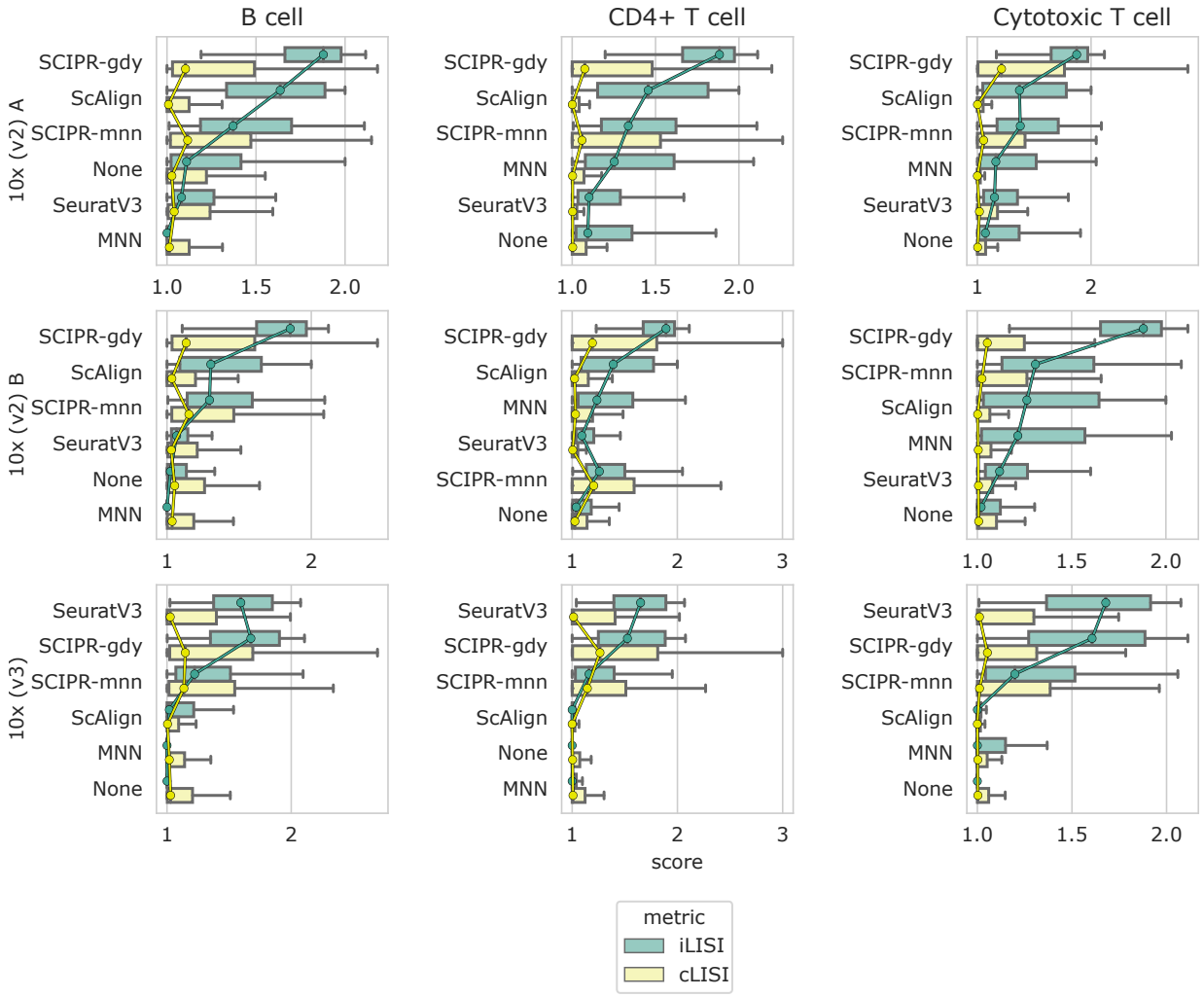


Figure 3.17: Quantitative scoring of alignment methods on the PBMC dataset with a cell type held out from the target set. Each row of subplots are alignment tasks with the same source batch, where each column uses a different cell type as a hold-out from the target set (the 10x Chrom. (v2) batch). Box plot computation and ordering of methods in each subplot is determined in the same fashion as in Figure 3.8.

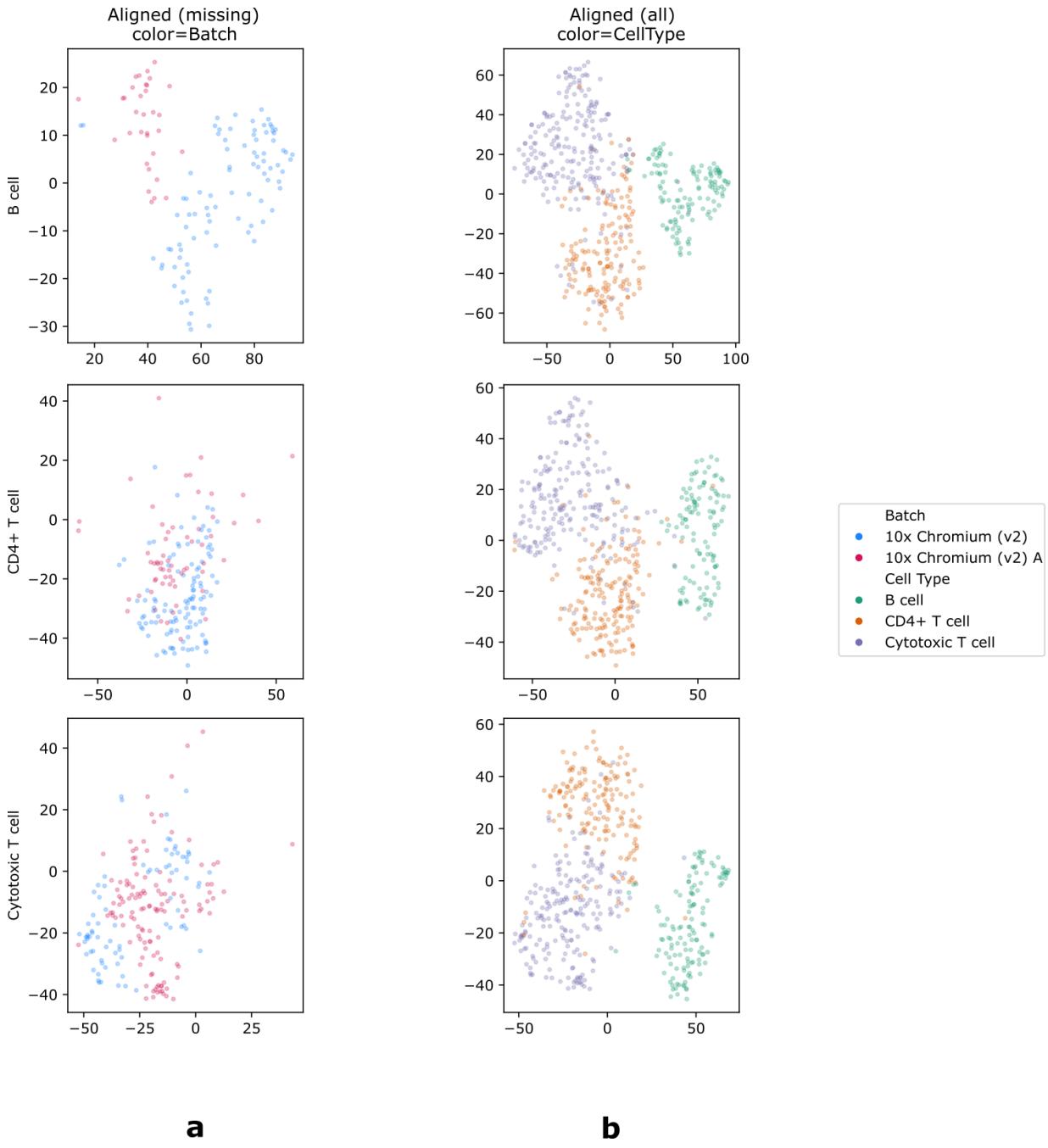


Figure 3.18: Embedding (t-SNE) visualization from the *PBMC:10x Chrom. (v2) A→10x Chrom. (v2)* task using SCIPR-gdy showing generalizability to new cells. In each alignment task (rows), a different cell type is completely held-out from the *source* set. The model is then fitted to align the source and the target, and this fitted model is then used to transform the full source set, including the held-out cell type which the model did not see in the source set used for fitting. The first column (a) shows just the held-out cell type colored by batch, after applying the fitted SCIPR-gdy model to align it. The second column (b) shows all of the data after applying the fitted model, colored by cell type.

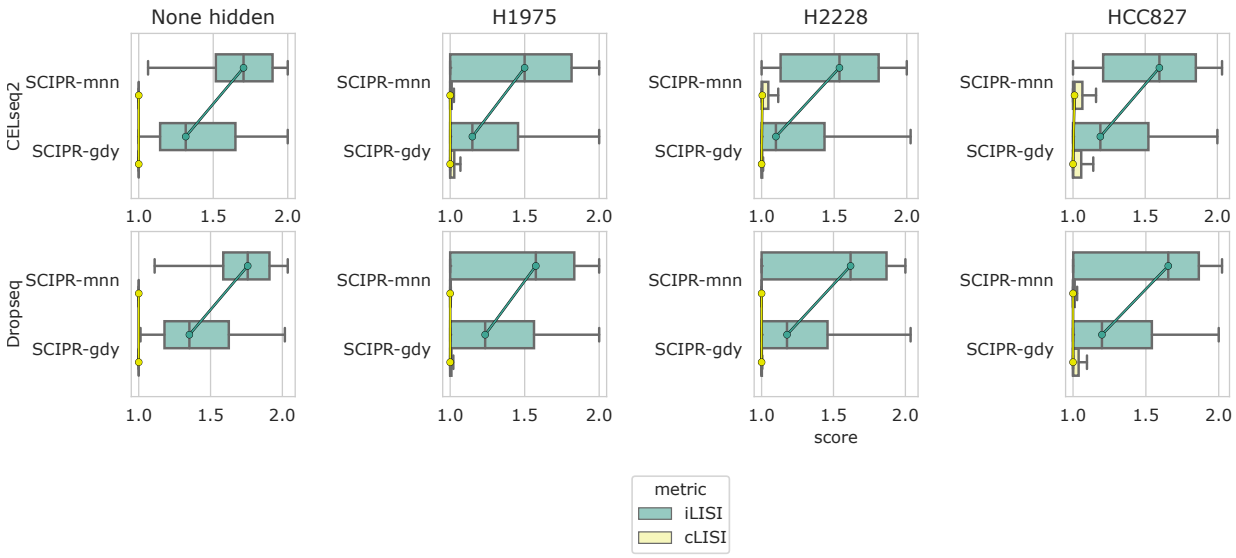


Figure 3.19: Quantitative scoring of alignment methods on the CellBench dataset with a cell type held out from the source set. Each row of subplots are alignment tasks with the same source batch, where each column uses a different cell type as a hold-out from the source set. The target set is 10x for all. In the first column, "None hidden", no cells were hidden from the source set. Box plot computation and ordering of methods in each subplot is determined in the same fashion as in Figure 3.8.

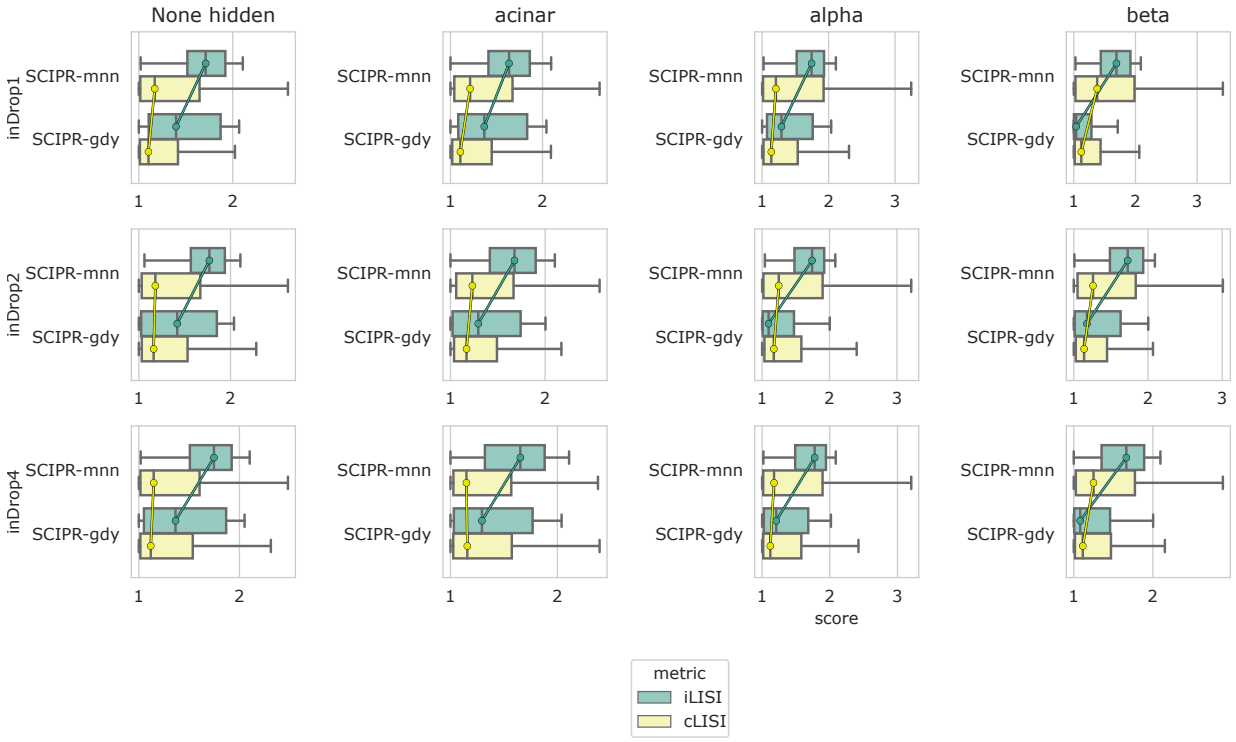


Figure 3.20: Quantitative scoring of alignment methods on the Pancreas dataset with a cell type held out from the source set. Each row of subplots are alignment tasks with the same source batch, where each column uses a different cell type as a hold-out from the source set. The target set is inDrop3 for all. In the first column, "None hidden", no cells were hidden from the source set. Box plot computation and ordering of methods in each subplot is determined in the same fashion as in Figure 3.8.

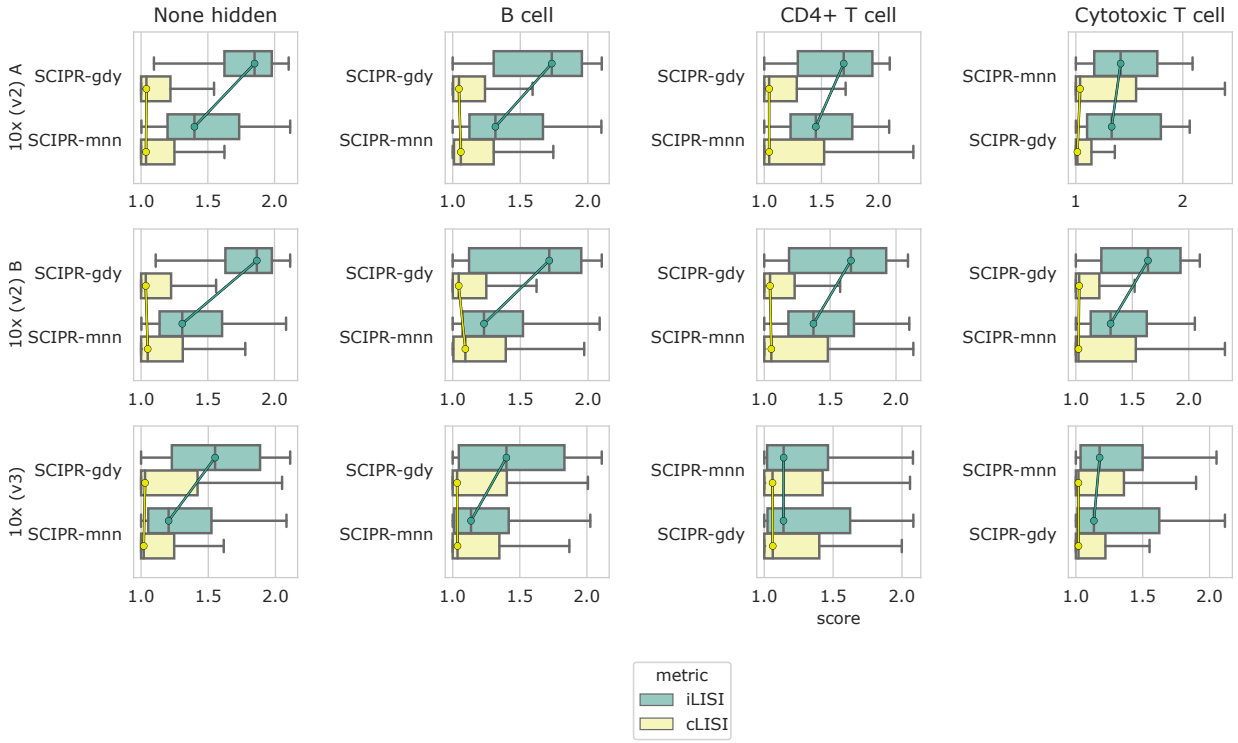


Figure 3.21: Quantitative scoring of alignment methods on the PBMC dataset with a cell type held out from the source set. Each row of subplots are alignment tasks with the same source batch, where each column uses a different cell type as a hold-out from the source set. The target set is 10x (v2) for all. In the first column, "None hidden", no cells were hidden from the source set. Box plot computation and ordering of methods in each subplot is determined in the same fashion as in Figure 3.8.

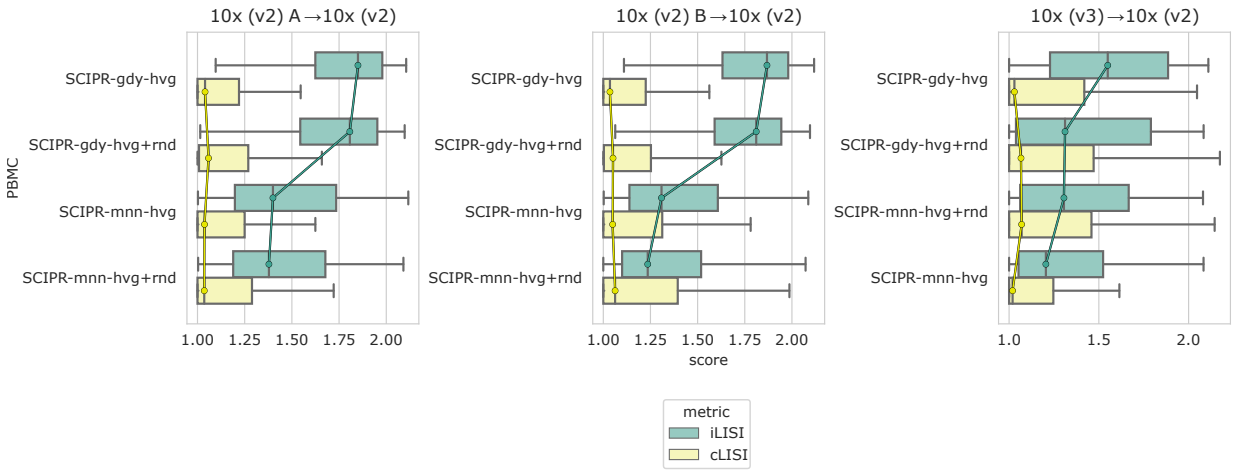


Figure 3.22: Comparison of using highly variable genes versus additional random genes in SCIPR. We compared using highly variable genes (as described in Supporting Methods 3.2.3), versus using additional genes, for SCIPR models on the PBMC dataset (the largest dataset). Both our SCIPR-gdy and SCIPR-mnn models were run with either just the highly variable genes ("-hvg" suffix) (there are 1466 in the PBMC dataset) or with the highly variable genes and an equal number of randomly selected other genes ("-hvg+rnd" suffix). The three subplots correspond to the three different alignment tasks (aligning a source batch to a target batch) within the PBMC dataset. These quantitative scores show that SCIPR still performs well, and is robust to the inclusion of even more genes that are not necessarily the most informative genes. Box plot computation and ordering of methods in each subplot is determined in the same fashion as in Figure 3.8.

Chapter 4

Identifying markers for cell type and tissue type combinations in HuBMAP data

Finding marker genes that can be used to identify phenotypes is a common task in scRNA-seq data analysis. These phenotypes are usually cell types, disease states, or developmental stages. In this chapter however, we are concerned with finding markers that are indicative of not only a particular cell type, but are unique to that cell type within a particular tissue. In other words, we want to find marker genes that are specific to the joint (cell type, tissue type). While in Chapter 2 and Chapter 3.1 cell type classification was used as a proxy to learn a lower dimensional representation (a neural embedding with some desirable properties), here classification serves as a proxy for learning important features of the input data (finding marker genes).

In contrast to the traditional statistical differential expression analysis, we propose a machine learning approach, based on feature selection in a classification framework. We use a logistic regression-based classifier tasked with classifying the (cell type, tissue type) phenotype of each cell, and experiment with different regularization strategies to identify unique markers for such phenotypic pairs. We apply our approach to a previously studied mouse dataset, and also to a new human scRNA-seq dataset that includes data from the HuBMAP [115]. We show that using an exclusive lasso penalty recovers more and higher quality markers of cell types and tissues compared to the traditional lasso penalty. We conduct qualitative assessments of the marker lists with visualizations of gene expression in our large datasets, and we also use quantitative scores to show that the marker lists from the exclusive lasso model are significantly better. We also investigate the biological relevance of the markers for their respective phenotypes.

4.1 Introduction

Several recent international efforts focus on the characterization of gene expression in different tissues, organs, disease states and more. Examples include HuBMAP, a large NIH effort to reconstruct a 3D map of the human body at the single cell resolution [115], the Human Cell Atlas [113, 114], the Cancer Cell Atlas [174], the Brain Atlas [175], and many more.

One of the first steps of studies at the single cell level is to characterize cell states or cell types. In many cases this is done by using marker genes whose expression, or co-expression with

other such markers, is indicative of a cell type [53, 176, 177]. To find such markers researchers often rely on differential expression (DE) testing, which poses a statistical hypothesis test in which expression of genes in one group of cells is compared the expression in another group. These groups are usually defined by cluster, cell-type, or condition labels. However, with the increase in large multi-organ studies we can now answer another important though unstudied question: are there markers that are specific for a cell type and simultaneously specific to a tissue? Beyond their ability to characterize specific cell types Such genes can be important for clinical applications. Consider T cells which mature in the thymus [178, 179]. While T cells later migrate and reside in tissues throughout the body, the identification of T cells that have recently left the thymus (recent thymic emigrants, or RTEs) can impact treatment decisions. Such T cells are an indicator of thymic output [180] and provide relevant information on the activity of the adaptive immune system which can impact clinical decisions, for example for prognosis and treatment of HIV patients [181]. Similarly, the role of resident and infiltrating immune cell types is still an active area of research for neurodegenerative diseases. A key challenge is the current inability to distinguish the resident central nervous system (CNS) immune cells and the bone marrow-derived immune cells [182]. Better signatures of CNS-specific immune cells and signatures of infiltrating immune cells are needed to understand the immune responses to therapies.

Identifying such cell type / tissue unique genes is a challenging problem. Unlike standard DE analysis in which one cluster or cell type is compared to all other clusters in the same tissue (or batch) to identify unique genes, here the focus is not just a single label for a cell (i.e. cluster) but instead on genes unique to both, tissue and cell type. Due to batch effects that are often associated with the data (for example, when tissue data is from different labs or technologies) and the relatively small number of cells in a tissue / cell type cluster, simply using one vs. all for the combined label is not likely to work.

Given the importance of this problem, recent efforts have attempted to characterize differences in cell type expression profiles among different tissues. Tabula Muris [183] is a collection scRNA-seq profiles of over 100,000 cells from over 20 different organs and tissues in *Mus musculus*. Great care was taken to collect scRNA-seq profiles while controlling for batch variables such as environment, gender, age, and identity of individual mice so as to produce a dataset that would allow for direct, controlled comparison of cell types across organs or tissues [183]. However, on the methodological side the study uses traditional clustering and DE analysis methods which are not directly tailored to the question of finding markers unique to combinations of cell types and tissues. An unsupervised graph-based clustering approach was used to identify sub-populations, which were then labeled using known markers, and traditional statistical DE analysis was applied in a one-versus-rest scheme to determine marker genes for the identified clusters [183]. While Tabula Muris represents a rich resource for inspecting the variation of gene expression profiles across cell types and organs, there is still a need for an automated procedure for finding distinctive signatures of (cell type, tissue type) phenotypes.

Here, we propose an automated pipeline for establishing signatures of (cell type, tissue type) phenotypes. We use a machine learning-based approach to take advantage of the large (and growing) sample size in multi-tissue human scRNA-seq datasets. Prior work has shown that given a large collection of scRNA-seq data discriminative approaches such as logistic regression can be used to improve both differential expression and differential transcript usage results, by inspecting the fitted model weights [184]. We extend these methods by using a multi-label softmax

regression classifier, tasked with predicting the joint cell type and the tissue type label of each cell. We use an exclusive lasso regularization to recover ranked lists of important genes for each (cell type, tissue type) combination.

We applied our method to both mouse and human data from multiple tissues and cell types. As We show, our softmax exclusive lasso expression model can correctly identify distinct gene modules that serve as signatures for cell type / tissue combinations.

4.2 Methods

4.2.1 Problem definition

First we formally define the problem that we are interested in. Our goal is to identify a set of genes (features) $G_{c,t}$ which are markers for a (cell type, tissue type) label combination. Specifically, based on the expression of this group of genes we should be able to determine for a cell if its from both, the cell type and tissue, or not. Ideally, the individual members of $G_{c,t}$ are also specific to this combination, and not any of its components (i.e. specific to the cell type only or specific to the tissue type only). We aim to find this set G for each label combination by using a large number of examples from labeled scRNA-seq data, where we can observe many instances of scRNA-seq expression of each label combination.

4.2.2 Data

We tested our method on both mouse and human data. For the mouse data we use scRNA-Seq data from the Tabula Muris mouse dataset. Specifically, we use the subset of their data that is from their fluorescence-activated cell sorting (FACS) based scRNA-seq protocol, which provides higher sensitivity and coverage than the droplet-based portion of their data [183]. For this data we use a subset of the tissues which had overlapping cell types. While this dataset is small in terms of the number of cells when compared to the human data discussed below (Table 4.2), it benefits from uniform processing and a standard cell type labeling scheme applied across cells from all organs / tissues.

We have also collected human scRNA-seq data from six tissues for our analysis (Table 4.1). The spleen, thymus, lymph node, and kidney datasets are obtained via the HuBMAP consortium [115]. The lung data is from a large scRNA-seq study on idiopathic pulmonary fibrosis (IPF), from which we used healthy control cells [185]. The PBMC data is from a large cohort study that collected scRNA-seq data to study celltype-specific effects of genetic variation on genome-wide gene expression [186]. The accessions for all data are listed in Table 4.3. The data from the spleen, kidney, lung, and PBMC datasets included cell type annotations. The data from the thymus and lymph node were from the same lab as the labeled spleen data. In order to annotate the cell types in thymus and lymph node as well, we performed label transfer using Seurat [29], with the spleen data serving as the reference.

All tissues with the exception of PBMC were processed using the same pipeline from the HuBMAP, available from the HuBMAP data portal at `https://portal.hubmapconsortium.org`. This pipeline quantifies transcripts from raw reads using Salmon [15] and uses scanpy [28]

Tissue	Number of cells
lung	76,178
spleen	34,515
PBMC	25,185
lymph node	24,311
thymus	22,367
kidney	10,353
Total	192,909

Table 4.1: Tissue distribution of our human scRNA-seq data.

Tissue	Number of cells
Fat	4,865
Spleen	1,697
Lung	1,676
Limb Muscle	1,090
Total	9,328

Table 4.2: Tissue distribution of the subset of Tabula Muris [183] scRNA-seq data we use.

for secondary analysis. It is available via the consortium's repository at `https://github.com/hubmapconsortium/salmon-rnaseq`. For the PBMC data we used the read-count matrices that were published with the accompanying paper [186]. For each tissue, we additionally remove any cells that did not express at least 200 genes. The resulting final cell counts per tissue in our combined dataset are listed in Table 4.1.

4.2.3 Preprocessing

Cell type label harmonization

Because our aim is to find marker genes for cell types and tissues, we focus on cell types that are common to multiple tissues. Immune cells are one example of a such cell types because they infiltrate multiple tissues, and some become tissue-resident.

Since cell type annotations are based on work from different groups using slightly different naming conventions, the same cell types may appear under different names in different tissue. We this first harmonize similar cell type labels for cell types that are present in multiple studies. The synonymous labels are shown in Table 4.4. After harmonizing the cell type labels we generated a dataset for analysis by keeping only the (cell type,tissue) combinations which had at least 100 cells. The resulting cell type and tissue type distribution is shown in Table 4.5. For the Tabula Muris mouse data, we used a minimum threshold of just 50 cells for each combination because this dataset was much smaller (Table 4.6).

Tissue	Accession codes
lung	GSE136831
spleen	HBM336.FWTN.636, HBM472.NTNN.543, HBM984.GRBB.858, HBM556.QMSM.776, HBM396.RPRR.624
PBMC	EGAS00001002560
lymph node	HBM226.LBVC.946, HBM252.HMBK.543
thymus	HBM373.RTKK.586, HBM724.ZKSM.924, HBM895.WHGJ.263, HBM457.SQKR.279
kidney	HBM684.ZPCL.638, HBM595.QDQD.996, HBM476.NNFJ.275, HBM327.JDHF.334

Table 4.3: Data access codes for our human scRNA-seq data. Accession prefix legend: EGA, European Genome-phenome Archive (EGA); HBM, HuBMAP data portal; GSE, Gene Expression Omnibus (GEO).

Chosen label	Synonyms
B Cell	B, B cell, B_Plasma
T Cell	T, alpha-beta T cell, gamma-delta T cell, CD4+ T, CD8+ T, T_Cytotoxic, T_regulatory
NK Cell	NK, natural killer cell, CD56(bright) NK, CD56(dim) NK
Macrophage	macrophage, Macrophage_Alveolar
Fibroblast	splenic fibroblast

Table 4.4: Cell type synonyms in our human scRNA-seq data.

Cell type and tissue type combination	Number of cells
T Cell,PBMC	18,234
B Cell,spleen	16,341
T Cell,thymus	14,009
B Cell,lymph node	13,283
T Cell,lymph node	10,727
T Cell,spleen	5,523
B Cell,thymus	4,814
T Cell,lung	4,342
NK Cell,PBMC	2,881
NK Cell,spleen	2,625
NK Cell,lung	1,328
B Cell,lung	863
B Cell,PBMC	838
Fibroblast,lung	561
NK Cell,thymus	533
Fibroblast,thymus	446
NK Cell,lymph node	136
Fibroblast,kidney	117
Total	97,601

Table 4.5: Tissue and cell type distribution of our human scRNA-seq data.

Cell type and tissue type combination	Number of cells
B cell,Spleen	1,297
B cell,Fat	519
T cell,Fat	353
T cell,Spleen	352
natural killer cell,Fat	95
B cell,Limb Muscle	71
B cell,Lung	57
T cell,Lung	53
Total	2,797

Table 4.6: Tissue and cell type distribution of the Tabula Muris mouse scRNA-seq data.

Filtering to highly variable genes

We followed prior methods and filter the features (genes) to obtain a set of variable genes[29, 30, 187, 188]. We use the method in [187] and scanpy's `scanpy.pp.highly variable genes` function [28]. Our read-count scRNA-seq data across the six human tissues contained 12,115 genes in common. After filtering to the highly variable genes (across the full dataset of 192,909 cells), we are left with 2,007 genes for further downstream analysis. For the Tabula Muris mouse data, there were 23,433 genes initially, after filtering to the highly variable genes we are left with 5,127 genes.

4.2.4 Multi-label classification

In order to find joint markers for two variables (e.g. cell type and tissue type), we require a model which couples classification of both variables. To this end, we transform our classification problem to the multi-label paradigm, in which our model is tasked with predicting both variables for each instance. Formally, we have our scRNA-seq data $\mathcal{D} = \{(x_i, c_i, t_i), i = 1, ..., n\}$ where $x_i \in \mathbb{R}^G$ is the read count vector for cell i (for G genes), $c_i \in \{$T cell, B cell, NK cell, ...$\}$ is the cell type label for cell i, and $t_i \in \{$Spleen, Lymph node, Lung, ...$\}$ is the tissue type label for cell i. The cell type label set is $\mathcal{C}$ with cardinality $C = |\mathcal{C}|$, and the tissue type label set is $\mathcal{T}$ with cardinality $T = |\mathcal{T}|$. Whereas single-label classification would aim to learn the mapping $f_{celltype}: \mathcal{X} \to \mathcal{C}$ or $f_{tissuetype}: \mathcal{X} \to \mathcal{T}$, here we aim to learn the joint classification function $f: \mathcal{X} \to \mathcal{J}$.

Thus, we transform our training data for multi-label classification as $\mathcal{D}' = \{(x_i, y_i), i = 1, ..., n\}$ where the label $y_i \in \mathcal{J}$ is now a joint label set such that

$$\mathcal{J} = \{\text{(T cell,Spleen)}, \text{(T cell,Lymph node)}, \text{(T cell,Lung)}, ...,$$

$$\text{(B cell,Spleen)}, \text{(B cell,Lymph node)}, \text{(B cell,Lung)}, ...,$$
$$...\}$$

In other words, $\mathcal{J} = \mathcal{C} \times \mathcal{T}$ is the Cartesian product of the cell type label and tissue type label sets. Thus its cardinality is $J = |\mathcal{J}| = C \times T$. By transforming the problem in this way, we are able to treat the multi-label classification problem as an instance of a multiclass classification problem.

The optimization problem for fitting the model with weights W is:

$$\min_W \sum_{i=1}^{n} \ell(y_i, f_W(x_i)) \tag{4.1}$$

We use logistic regression as our modeling approach because it is an interpretable model. Specifically, we would like to inspect the model weights W after fitting to determine important features for each class (cell type and tissue type combinations). This interpretation is only informative if the model weights are regularized during training. For example, using an L2 norm penalty on the model weights encourages the model to only assign high coefficient weights to the most important input features [189]. We discuss this vital point about regularization in more

depth in the next section. The generalization of logistic regression to multiclass problems is also called multinomial logistic regression, or softmax regression. See Supporting Methods 4.3.1 for an overview of softmax regression and how we formulate it for our problem, and see Supporting Methods 4.3.2 for implementation details.

4.2.5 Exclusive lasso penalty

After fitting the *regularized* model, the weights W can be interpreted as the importance of each feature for each class [190, 191]. Unfortunately, L2 regression models do not produce sparse or distinct weights, and so many features will appear to be important for multiple classes, or even all classes. Such weights would not result in marker genes that could be used to identify a phenotype (e.g. a cell type-tissue type pair) amongst others.

A classical approach to induce sparsity in logistic regression models is to apply an regularization L1 penalty resulting in many feature weights being set to zero. The functional form of L1 regularization (also known as lasso regularization) is given below:

$$R_{L1}(W) = \sum_{ij} |W_{ij}|$$

However, while the L1 penalty tends to produce sparse solutions, it does not encourage non-zero features to be distinct to particular classes. Here, we need to create competition between the classes for access to features. Thus, here we extend the "Exclusive Lasso" penalty [192] instead of the L1 penalty:

$$R_{EL1}(W) = \sqrt{\sum_{g \in \{1,..,G\}} \left(\sum_{j \in \{1,...,J\}} |W_{jg}| \right)^2}$$

In contrast to the L1 regularization, here for each feature the L1 norm is applied across the different classes, inducing competition between the classes for use of the same feature. The L2 norm is then applied to enforce this competition across all features, by encouraging small weights.

4.2.6 Learning model parameters

The loss function in the multiclass classification optimization problem (Equation 4.1) is multi-class cross-entropy:

$$\ell(y_{true}, \overrightarrow{y_{pred}}) = - \sum_{j \in \{1,...,J\}} \mathbb{1}_{y_{true}}(j) \log y_{pred_j}$$

Where $\overrightarrow{y_{pred}} \in \mathbb{R}^J$ is the predicted probability distribution over the J classes (the output of the softmax regression model) and $\mathbb{1}_{y_{true}}(j)$ is the indicator function which takes value 1 if $j = y_{true}$ and 0 otherwise.

The overall objective function is then:

Dataset	Model	LR	λ_{reg}	Max epochs	Mean score	Std score	Test score
Human	L1	0.001	0.001	20	0.795071	0.004300	0.805
Human	Excl. L1	0.001	0.01	10	0.768890	0.001570	0.810
Mouse	L1	0.001	0.01	10	0.502260	0.006013	0.503
Mouse	Excl. L1	0.001	0.1	10	0.481559	0.003838	0.463

Table 4.7: Hyperparameter tuning results for our softmax regression models. See Supporting Methods 4.3.3 for details on how we conduct the hyperparameter search. The scores reported are class balanced accuracies. The best settings amongst these are shown in each row of the table, for the different dataset and model type (choice of regularization) combinations, along with the mean and standard deviation of the score across the three folds. The test score is reported on a portion of the data that was held-out from the hyperparameter tuning and training stage.

$$J(W) = \sum_{i=1}^{n} \ell(y_i, f_W(x_i)) + \lambda R_{EL1}(W) \tag{4.2}$$

We fit the parameters W using stochastic gradient descent with the Adam optimizer over the training data. We use a learning rate schedule to reduce the learning rate from its initial value. Specifically we reduce the learning rate by a factor of $1/2$ after 10 epochs of no improvement on validation loss. To fit the model hyperparameters such as the initial optimizer learning rate, number of training epochs, and the amount of regularization, we used three-fold cross validation (Supporting Methods 4.3.3). The final model was fitted using the best parameters for 200 epochs, keeping the weights from the epoch with the lowest validation loss as the final model. Hyperparameter tuning results are shown in Table 4.7.

4.3 Supporting Methods

4.3.1 Softmax regression model formulation

Multinomial logistic regression is a generalization of logistic regression from the binary classification setting to more than two classes. The output of the multinomial logistic regression model is a probability distribution over the J labels for a sample x_i, as given by using the softmax activation function after the linearity $Wx^T + b$. Thus, multinomial logistic regression is also known as softmax regression. The linearity can be rewritten as Wx^T without loss of generality by appending a constant 1 feature to every data example and adding an additional column to the weight matrix W. The functional form of our softmax regression model is then:

$$f_W(x) = \frac{1}{Z}\left[\exp(W^{(1)}x^T),\ \ldots,\ \exp(W^{(J)}x^T)\right]^T$$

The predicted probability that a sample x belongs to class j is then:

$$f_W(x)_j = \frac{\exp(W^{(j)}x^T)}{Z}$$

Where $W^{(j)}$ is the j-th row of the weight matrix W (for the j-th class) and $Z = \sum_{j=1}^{J} \exp(W^{(j)} x^T)$ is a normalization factor so that the class probabilities sum to 1. Figure 4.1 illustrates the softmax regession model as a single-layer neural network. We highlight the fact that each node in the output layer is joint node, representing the combination of a cell type and a tissue. This is in contrast to traditional single-label classifiers for cell types or tissue types, where each output node represents just a particular cell type, or just a particular tissue type.

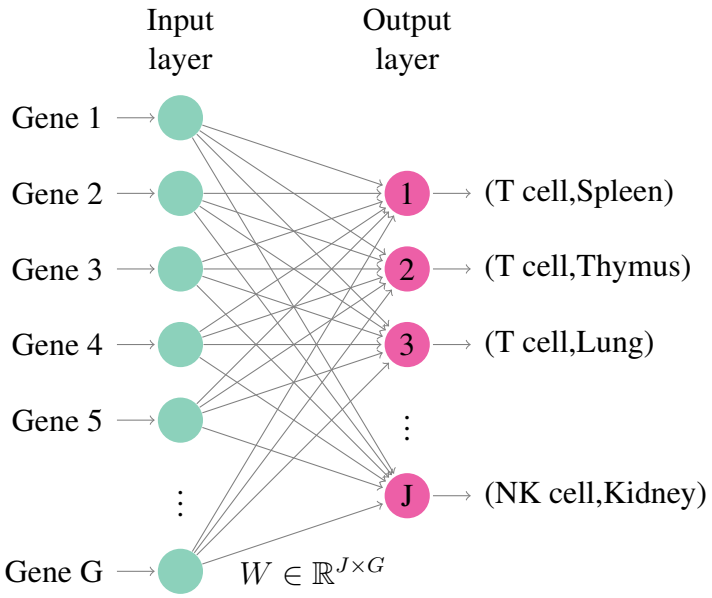


Figure 4.1: Multi-label classifier architecture. The classifier is a softmax regression model (also known as multinomial logistic regression). Each output node represents a combination of a cell type and a tissue type.

4.3.2 Model implementation details

We implemented our softmax regression models as single-layer PyTorch neural network classifiers [193]. We utilized skorch, scikit-learn compatible neural network library that wraps PyTorch [194, 195]. We extended skorch's `NeuralNetClassifier` class to add the Exclusive L1 penalty to the training objective. We experiment with both L1 and Exclusive L1 penalty in this work, which are both sparsity-inducing penalties. However, when a gradient descent optimization method is used to fit the weights of regularized models, it is extremely unlikely that gradient descent will arrive at exact zero for the parameter values. Rather, stochastic gradient descent (SGD) will usually achieve very near zero parameter values, alternating above and below zero [196]. This destroys the sparsity that we hope to achieve, so we implement a thresholding method which at the end of every epoch, sets any parameters less than ϵ=1e-4 to be exactly zero.

4.3.3 Hyperparameter tuning procedure

We conducted three fold cross validation to determine the best parameter settings for the learning rate (LR), regularization coefficient (λ_{reg}), and number of epochs to train for. The candidate

values in our search grid were LR = $\{0.01, 0.001\}$, λ_{reg} = $\{0, 0.001, 0.01, 0.1\}$, and Max epochs = $\{10, 20, 100\}$, resulting in 24 different combinations (three-fold cross validation results in 72 total models trained). We used class balanced accuracy as our evaluation metric, which is the mean of the accuracy on each of the separate classes. Once we choose the highest scoring setting, a final model with those settings was trained on the entire training set for 200 epochs, taking the final weights as those from the epoch that achieved the lowest validation loss. The training set (also used for the three-fold cross validation) was taken to be 75% of each dataset, with the remaining 25% being held-out for the test set. The samples were stratified by class label between these two splits. We used scikit-learn's `model selection.GridSearchCV` API to run the hyperparamter search [195].

4.3.4 Gene set enrichment analysis

Given a set of genes of interest (e.g. the set of exclusive marker genes for a (cell type, tissue type) class from our softmax regression models), we would like to compare them to annotated sets of genes to see if there is significant overlap, or enrichment. We compare our sets of genes to sets in the "Biological Process" domain of the Gene Ontology (GO) [34, 173]. We use the hypergeometric test for over-representation to quantify the significance of overlap between the gene sets and compute the adjusted p-value for each gene via the Benjamini-Hochberg procedure for controlling false discover rate (FDR) [172], using the implementation in the diffxpy package [171] via the function `diffxpy.api.enrich.test`. We specified the background set to be the full set of genes which was 12,115 genes for the human scRNA-seq dataset. In addition, we specified that the annotated gene sets from GO first be filtered to only include genes that exist in our background set.

4.4 Results

Our goal is to identify markers for (cell type,tissue) pairs. For this, we use an exclusive lasso classification approach in order to infer coefficients which would lend themselves to interpretation as gene importance scores. An overview of our method is presented in Figure 4.2. In the first step, we fit an exclusive lasso regularized softmax regression model to our large training data, with the multiclass classification object. In the second step, we select the top k most positive coefficients for each class for further interpretation (k is a user-defined hyperparameter, here we use $k = 20$). Then we further refine these coefficients in the third step, by comparing these top ranked genes between all of the classes and keeping only those that are unique for each class.

4.4.1 Classification performance

While we are interested in the markers and gene sets for each (cell type, tissue) pair, the validity of these sets is still subject to the accuracy of the classifier that we use, and so we start by inspecting the performance of our models. As others have found, the use of logistic regression-based models on scRNA-seq data is dependent on having sufficient sample sizes to properly fit the model [184]. We have also observed that the size of the dataset impacts the performance of the

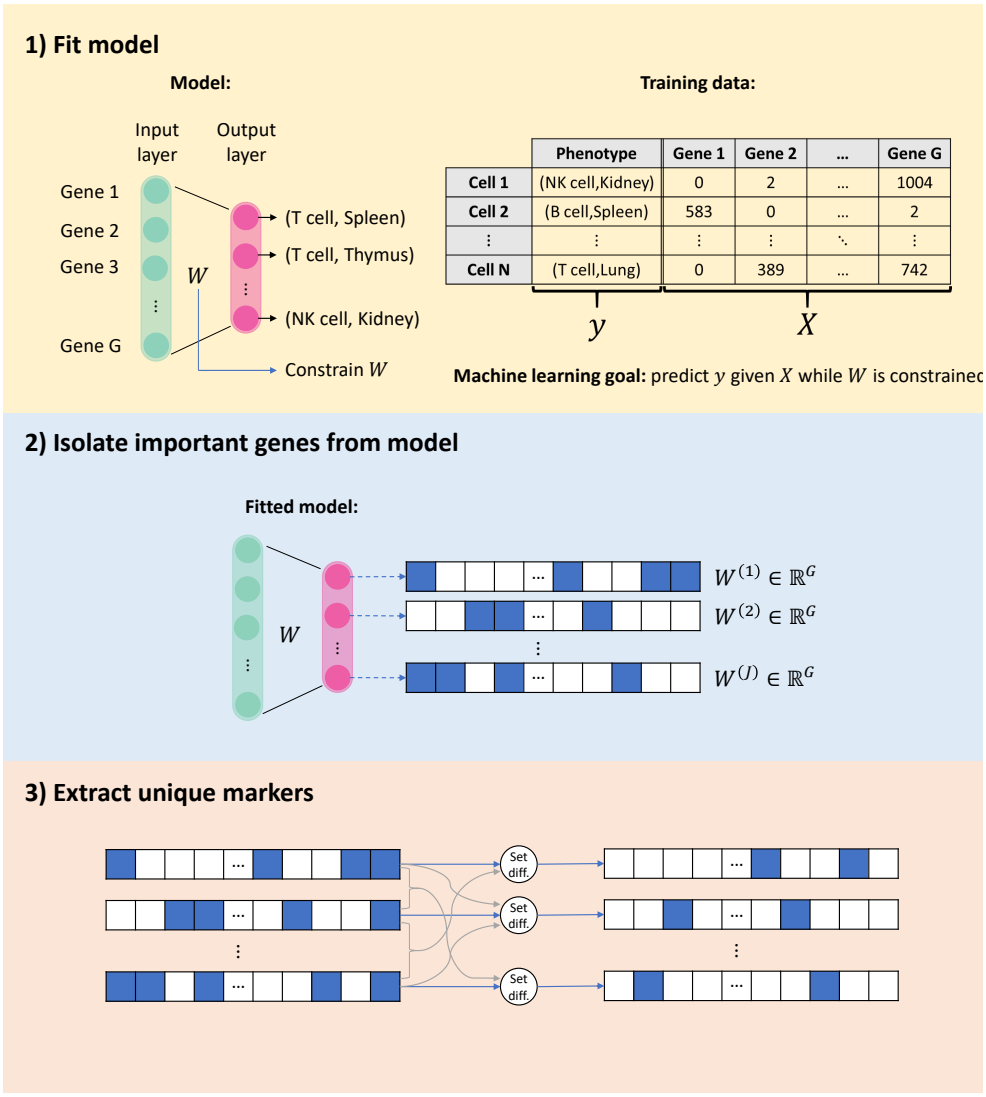


Figure 4.2: Summary of our feature selection-based marker finding method. 1) We fit a regularized softmax regression classifier to predict the joint (cell type, tissue type) phenotype label of each cell from a large training dataset. Regularization is crucial during training. 2) For each phenotype, we isolate the top k positively weighted genes, where k is a user-selected parameter, and is set based on how many markers they would like to recover. 3) Finally, we extract *unique* marker lists for each phenotype by removing any of the markers that also appear in the top k for other phenotypes.

models we train. When we fit and evaluate both the L1 and Exclusive L1 regularized models on the smaller Tabula Muris dataset, we see that they achieve a test balanced accuracy of just 50.3% and 46.3% respectively (for eight possible classes) (Table 4.7 and Figure 4.3a). However, when we apply our models to our larger human scRNA-seq dataset, we obtain test balanced accuracies of 80.5% and 81.0% for the L1 and Exclusive L1 regularized models respectively (18 possible classes) (Table 4.7 and Figure 4.4a). This strong performance, even when the number of classes is higher, shows that logistic regression leads to accurate results when applied to large datasets. It also shows that despite its more complicated regularization (which is required for identifying unique genes in our case) the Exclusive L1 penalty can improve classification performance when compared to traditional L1 penalty when the datasets are large, as we expect from the new atlas efforts.

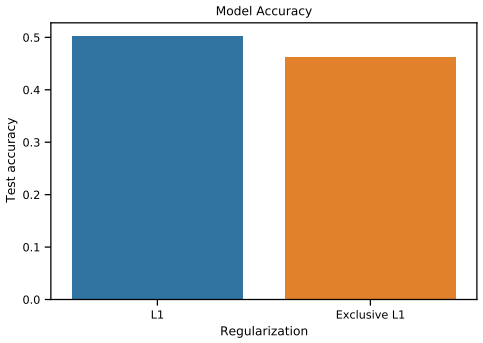
4.4.2 Sparsity and exclusivity of model weights

Feature selection in regularized logistic regression uses sparsity of model coefficients to select the most important genes. The sparsity (or number of features selected) depends on the amount of regularization that we apply to the coefficients (encoded using the λ parameter in Equation 4.2 and similarly defined for L1 regularization). To select the most appropriate value for λ we performed cross-validation experiments. For our human scRNA-seq data, the regularization parameters that achieved the best validation accuracy was $\lambda_{L1} = 0.001$ and $\lambda_{EL1} = 0.01$ for the L1 and Exclusive L1 regularized models respectively. For the Tabula Muris dataset, the values that led to the best performance were $\lambda_{L1} = 0.01$ and $\lambda_{EL1} = 0.1$.

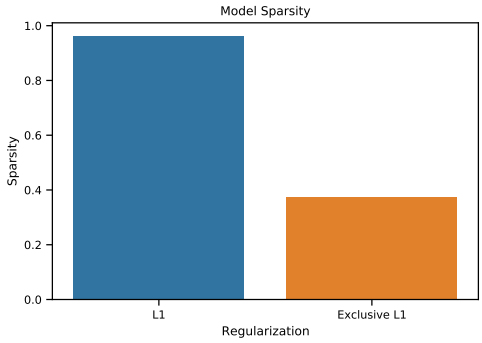
When inspecting the learned coefficients from these models, we observe that while both models can learn to accurately predict the correct class, the sparsity characteristics show great differences. For the models trained on the human data, when looking at the portion of model coefficients that are set to zero (Figure 4.4b) we see that the best setting for λ results in much sparser coefficients for the L1 model compared to the best setting for the Exclusive L1 model. However, when we look at the exclusivity of these coefficients (how disjoint the set of top features are for a phenotype compared to the others), we can see that, as expected, the Exclusive L1 regularized models identifies more exclusive markers (Figure 4.4c). Figure 4.3 shows a similar story for the models trained on the mouse data. We can also observe this visually when we inspect the heatmap of the model weights. Figure 4.5a shows the heatmap of the model coefficients from the L1 regularized model for the human data, and Figure 4.5b shows the same for the Exclusive L1 regularized model. We can observe that the large coefficients for the L1 regularized model often occur for the same gene (columns) across multiple classes (rows), while for the Exclusive L1 regularized model the large coefficients are reserved only for the most relevant label (cell type, tissue) pair for each gene.

4.4.3 Inspection of markers for cell type and tissue types

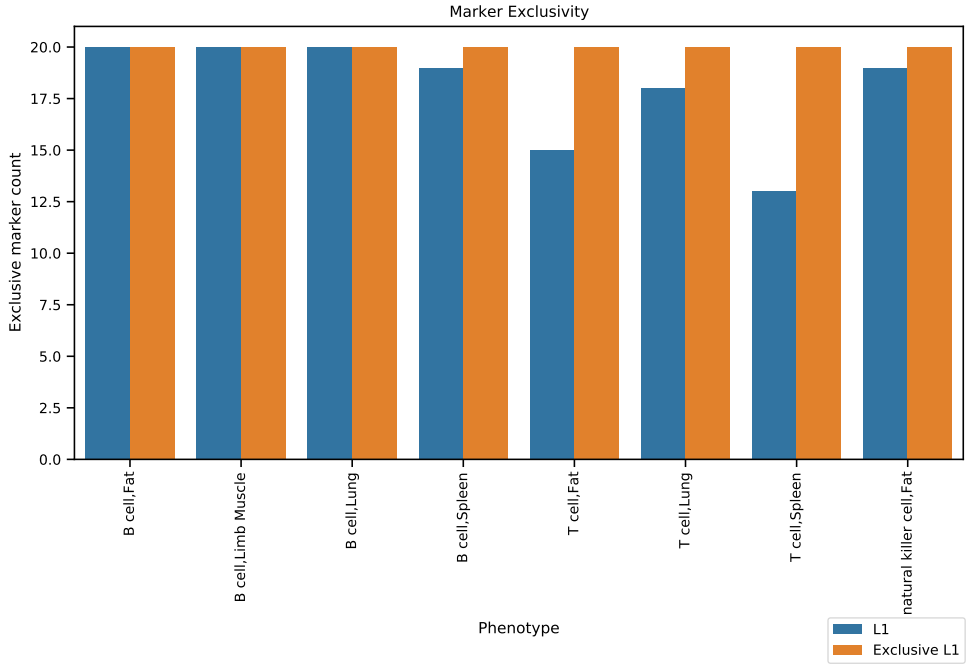
We next studied the marker lists identified by the exclusive lasso method. For the mouse data, we can see that in many cases, both the L1 and Exclusive L1 regularized models recover equal numbers of unique markers (Figure 4.3). However, in some phenotypes, such as (T cell,Spleen) the Exclusive L1 model finds many more markers. When we inspect these markers, we see that



(a) Balanced accuracy on test data

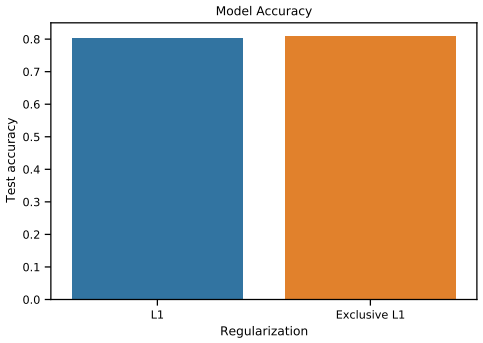


(b) Model weight sparsity

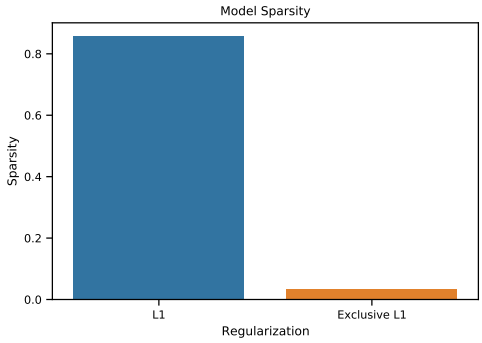


(c) Model weight exclusivity by class

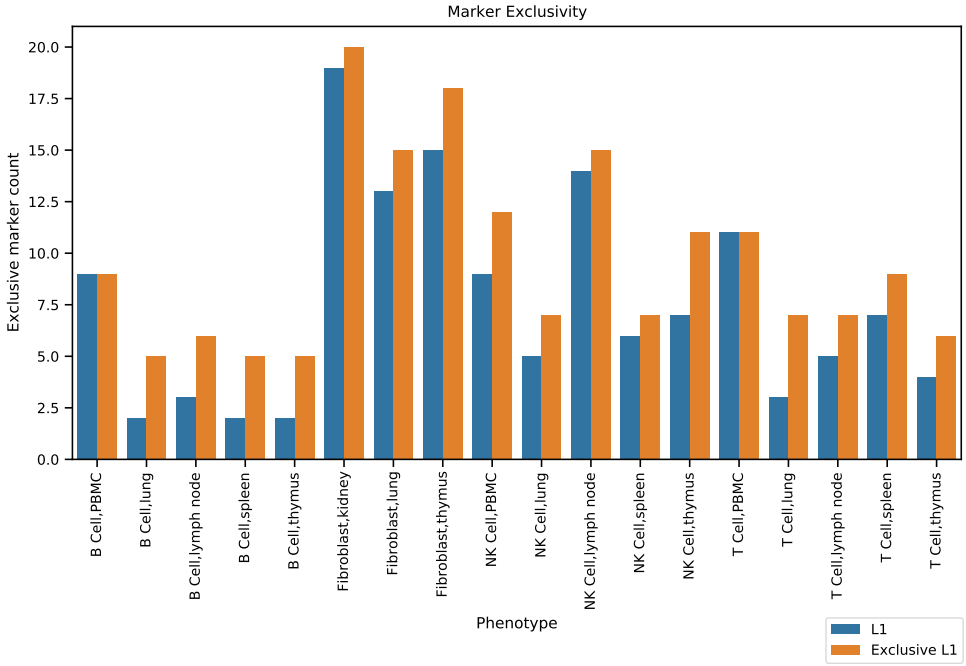
Figure 4.3: Model performance and sparsity on Tabula Muris data. (a) Balanced accuracy of L1 regularized model and Exclusive L1 regularized model on test data. (b) Sparsity of model weights, computed as the portion of model weights that are zero. (c) Plot of counts of exclusive markers for each cell type and tissue type combination, computed as follows: for each class (output node of the softmax regression model), take the top 20 highest weighted features. Then remove any of these features that are found in the top 20 lists for any other classes, and count how many remain.



(a) Balanced accuracy on test data

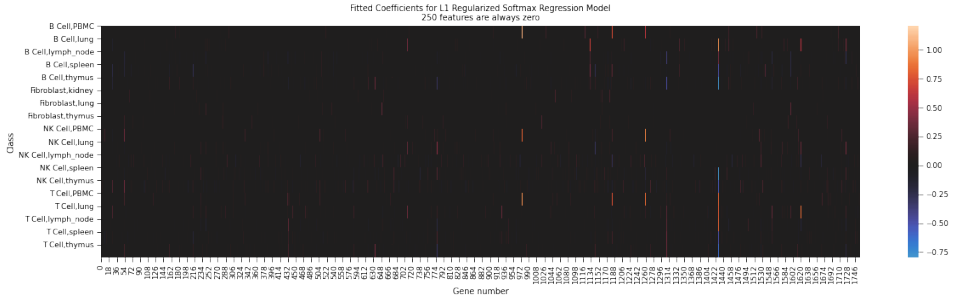


(b) Model weight sparsity

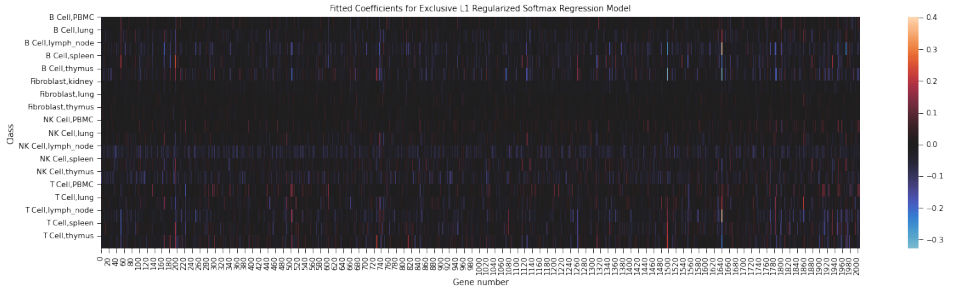


(c) Model weight exclusivity by class

Figure 4.4: Model performance and sparsity on large human data. (a) Class-balanced accuracy of L1 regularized model and Exclusive L1 regularized model on test data. (b) Sparsity of model weights, computed as the portion of model weights that are zero. (c) Plot of counts of exclusive markers for each cell type and tissue type combination, computed as follows: for each class (output node of the softmax regression model), take the top 20 highest weighted features. Then remove any of these features that are found in the top 20 lists for any other classes, and count how many remain.



(a) Learnt model coefficients for L1 regularized model



(b) Learnt model coefficients for Exclusive L1 regularized model

Figure 4.5: Model coefficient heatmaps from human data. (a) and (b) Visualizations of the coefficent sizes after fitting for the human data for the L1 and Exclusive L1 regularization respectively. In the weights from the L1 regularized model, 250 of the features were always zero (across all classes) and are removed from the heatmap.

they are of similar quality between the two models, both qualitatively through visual inspection of expression in UMAP plots (Figures 4.6 and 4.7), and quantitatively through statistical testing of the marker gene scores (Figure 4.8).

When we look at the results from the larger human scRNA-seq data, we see that the recovered markers differ much more between the L1 and Exclusive L1 regularized models, with the Exclusive L1 model always recovering at least as many unique markers as the L1 model, and usually more (Figure 4.4c). For example, when we inspect the module of marker genes for the (B Cell,lymph node) phenotype, we can see that Exclusive L1 model finds six unique markers (Figure 4.9) while the L1 model only finds three (Figure 4.10). Among the genes found by the Exclusive L1 model are *POU2F2*, *HLA-DMB*, and *LYN*, which are known to be overexpressed in lymph node tissue [197]. All three are also known to be involved in functions that are biologically relevant to B cells found in the lymph node including B cell development [198, 199], antigen presentation [200], and receptor signaling [201, 202]. Visual inspection of the expression of these genes reveals that many of them are indeed most highly expressed in the cells that are both B cells and from lymph node tissue (Figure 4.9). In addition to finding more markers, the quality of the markers from the Exclusive L1 model is higher compared to the L1 model. We can quantify this by scoring each cell by its the average relative expression level of the genes in the marker set, and subtract the average relative expression of control gene sets [82, 188]. We used the `scanpy.tl.score_genes` function in scanpy [28] to do this. This score is visualized on the UMAP dimensions in the second row, left plot of Figures 4.9 and 4.10. By comparing the difference in this mean score between (B Cell,lymph node) cells and all other cells, we can see

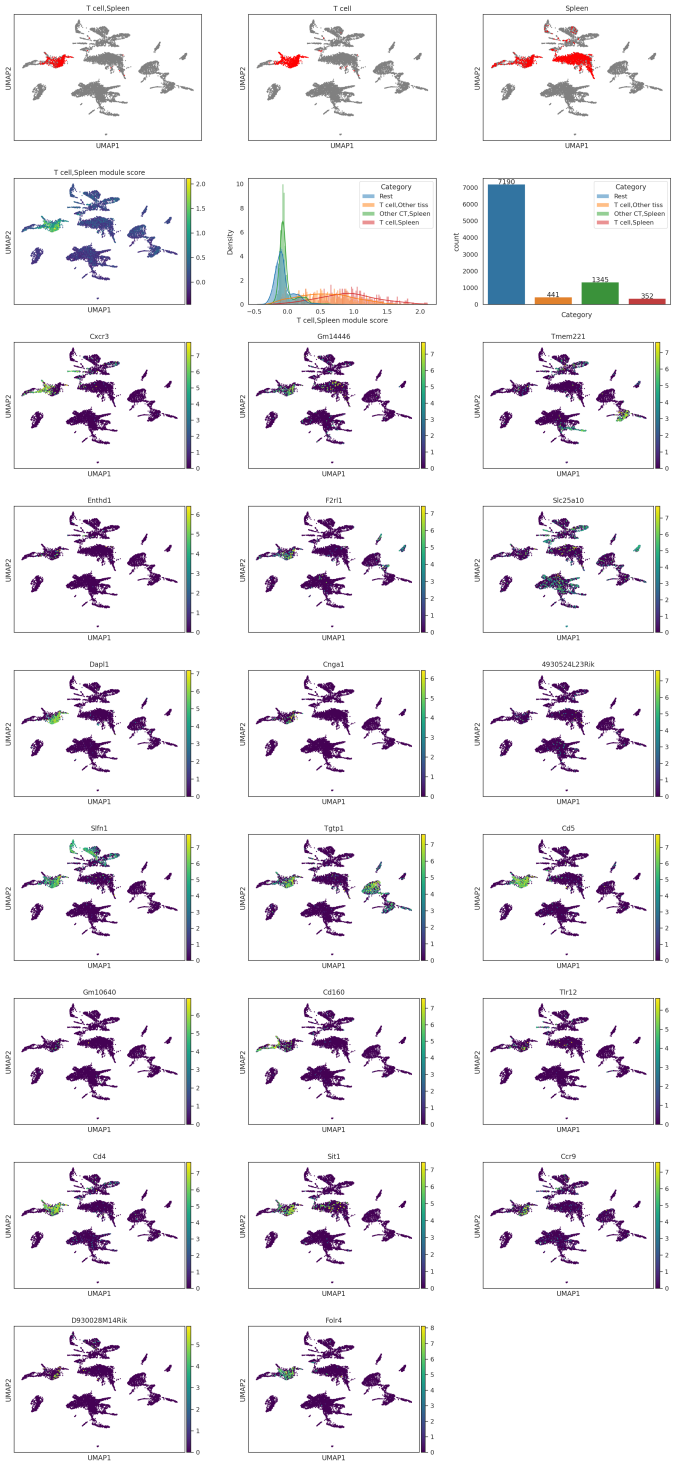


Figure 4.6: Visualization of the recovered exclusive markers for the mouse (T Cell,Spleen) phenotype from the Exclusive L1 regularized model. The subplots are drawn in the same fashion as in Figure 4.9.

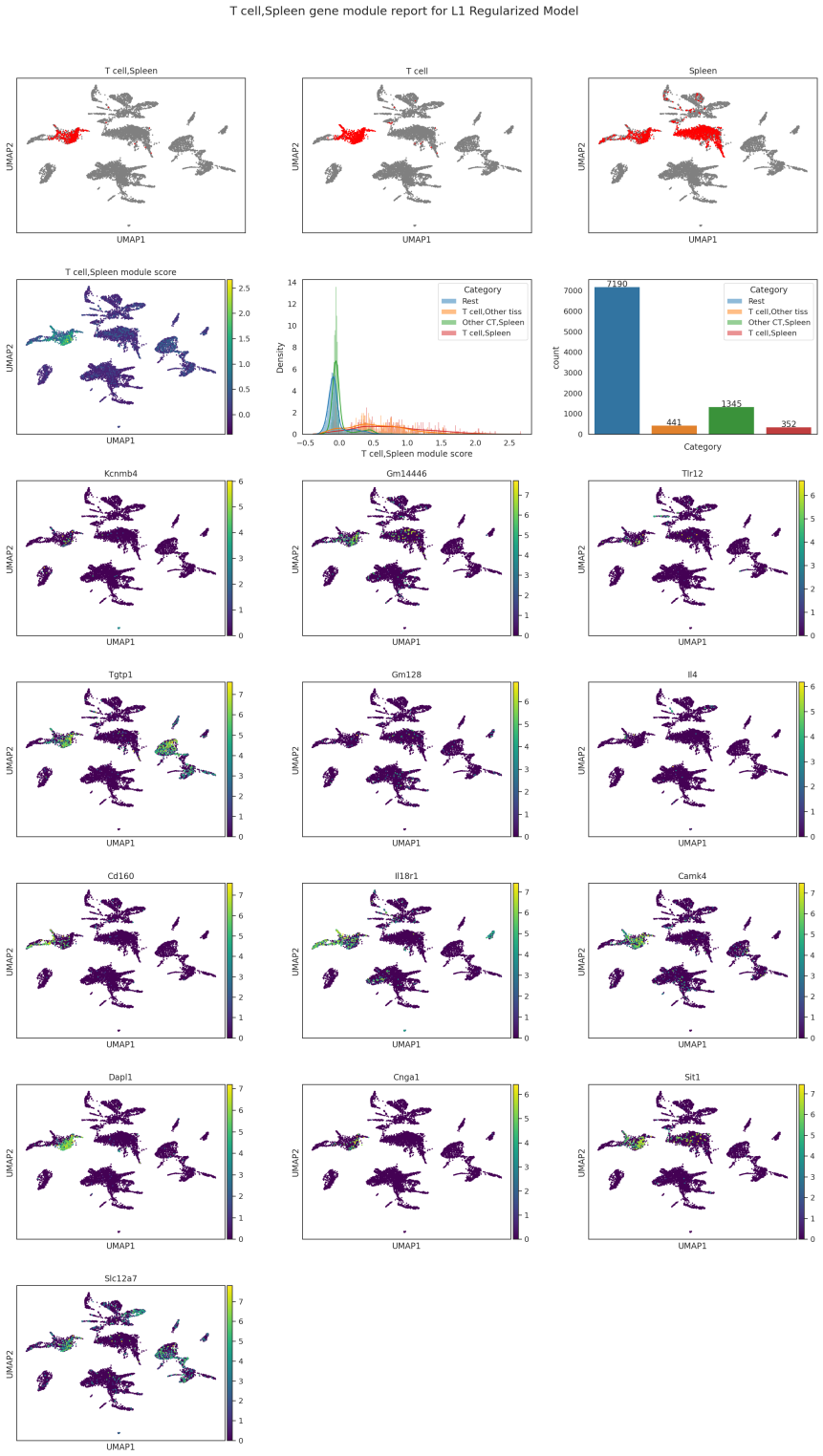
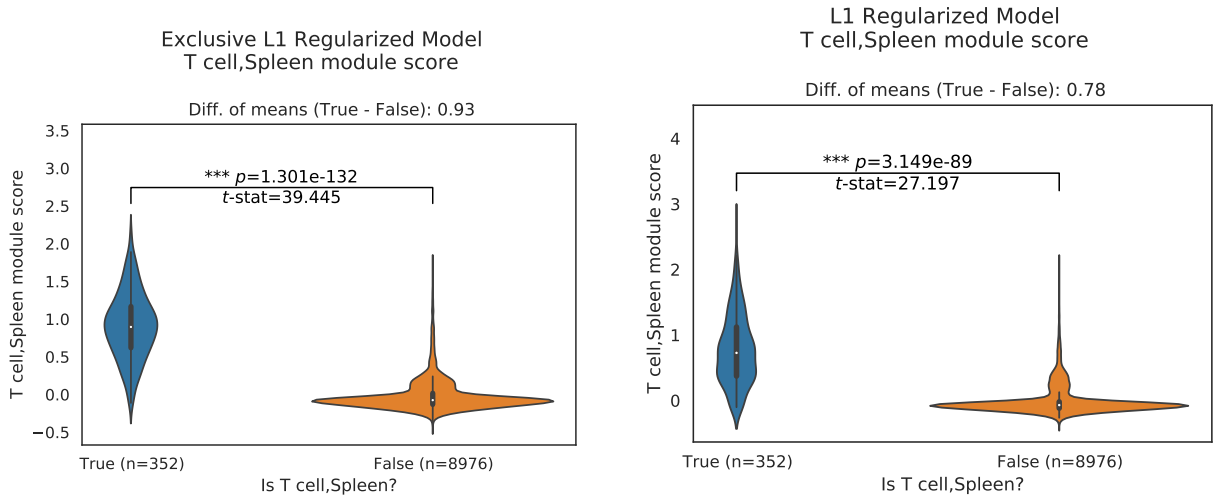


Figure 4.7: Visualization of the recovered exclusive markers for the mouse (T Cell,Spleen) phenotype from the L1 regularized model. The subplots are drawn in the same fashion as in Figure 4.9.



(a) Exclusive L1, (T Cell,Spleen)

(b) L1, (T Cell,Spleen)

Figure 4.8: Module score distribution of the exclusive marker list for (T Cell,Spleen) across cell identities in the mouse dataset. Plotting and statistical analysis is done in the same fashion as in Figure 4.13.

that scores are significantly greater in the (B Cell,lymph node) cells (Welch's t-test), and they are more significant in the case of the Exclusive L1 markers (Figures 4.13a and 4.13b).

We observed similar results for other labels. For example, for (T Cell,Thymus) phenotype marker set we again see that there are more and better unique markers from the Exclusive L1 model (Figures 4.11 and 4.12). Markers from the L1 regularized model include *TMSB10* (Figure 4.12, last row) which is highly expressed in many cells in the data, not just the (T Cell,Thymus) phenotype. The Exclusive L1 model on the other hand finds the *ITM2A* gene, which is preferentially expressed in T cells and is actually a T cell-specific transcription factor, observed to be expressed in thymus tissue as well [203]. We also see that the difference in mean score between the expression of this gene set in (T Cell,Thymus) cells and all other (cells,tissue) is statistically significant for the Exclusive L1 marker set, but is not significant for the L1 marker set (Figures 4.13c and 4.13d).

4.4.4 Functional analysis of identified marker genes

We conducted gene set enrichment analysis (GSEA) of our recovered gene lists using the Gene Ontology's Biological Process branch (Supporting Methods 4.3.4) [32]. Due to the small size of our marker lists (by design, see Figure 4.9 caption), many marker lists do not return significantly enriched results, and due to differences in sizes of list between the L1 and Exclusive L1 regularized models (Exclusive L1 tends to return more exclusive markers), there are only a handful of phenotypes for which we have significant results from both models. An example of this is shown in Tables 4.8 and 4.9. We see that both models recover marker lists that are relevant to the "natural killer cell" cell type. For example, the Exclusive L1 marker set is enriched for "Positive regulation of immune response" ($p = 0.000289$), "Lymphocyte mediated immunity" ($p = 0.005974$), "Cell activation involved in immune response" ($p = 0.041207$), and "Innate immune response" (0.041630). All of these terms are highly related to the specific function of

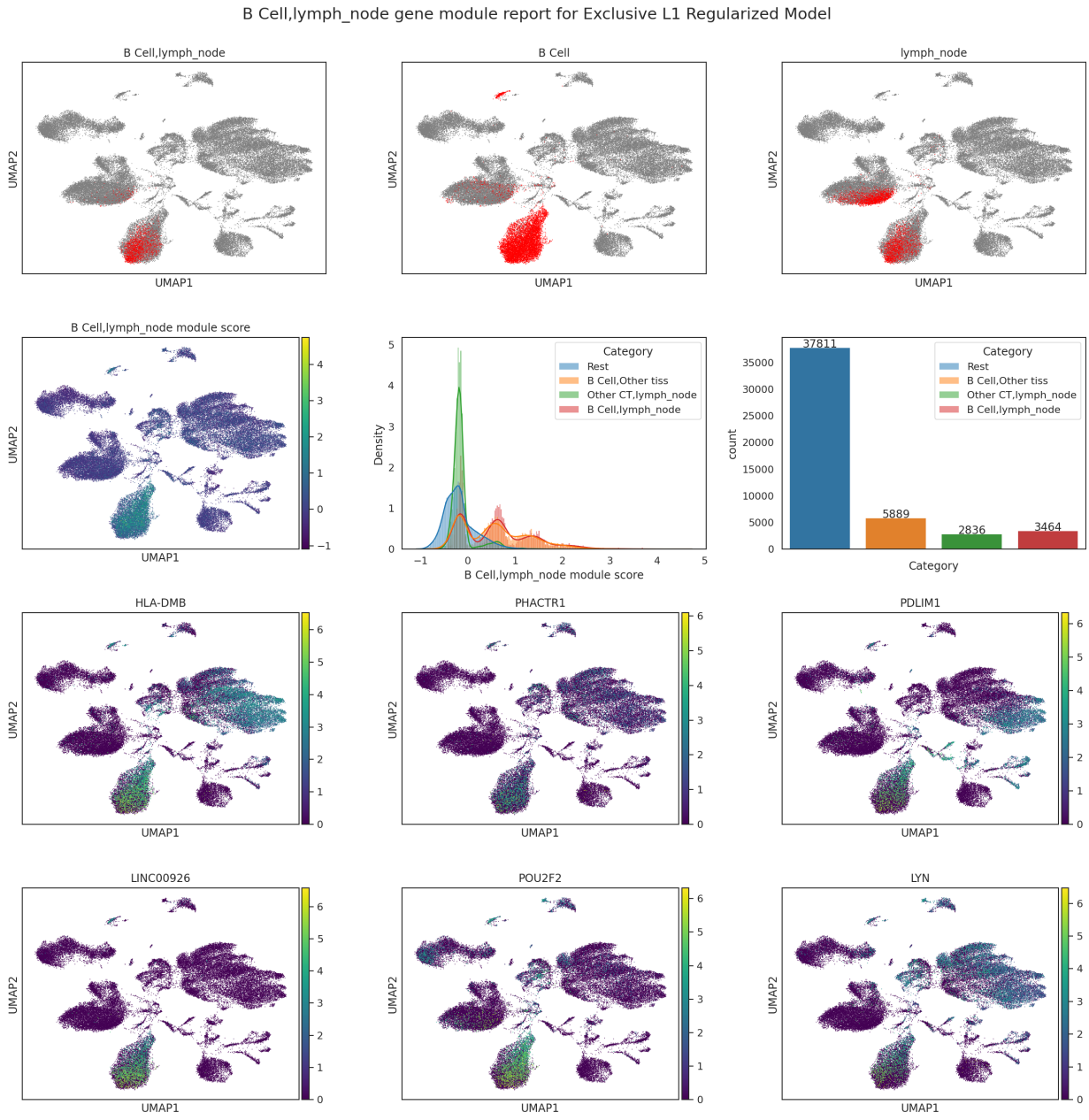


Figure 4.9: Visualization of the recovered exclusive markers for the human (B Cell,Lymph node) phenotype from the Exclusive L1 regularized model. The data was subsetted to 50,000 cells for visualization purposes only. The top row of three subplots highlight the cells in UMAP dimensions which are both B Cell and Lymph node cells, the cells that are B Cell (from any tissue), and the cells that are from Lymph node tissue (of any cell type) respectively. In the second row, the left subplot shows the UMAP embedding of the cells colored by their relative score of this set of marker genes. The middle subplot shows a histogram of these scores for the different combinations of cell types and tissues. The right subplot shows the number of cells for each combination in this subset of the data. The remaining subplots in the figure are UMAP embeddings of the cells colored by the expression of each of the individual genes that make up the recovered exclusive marker list.

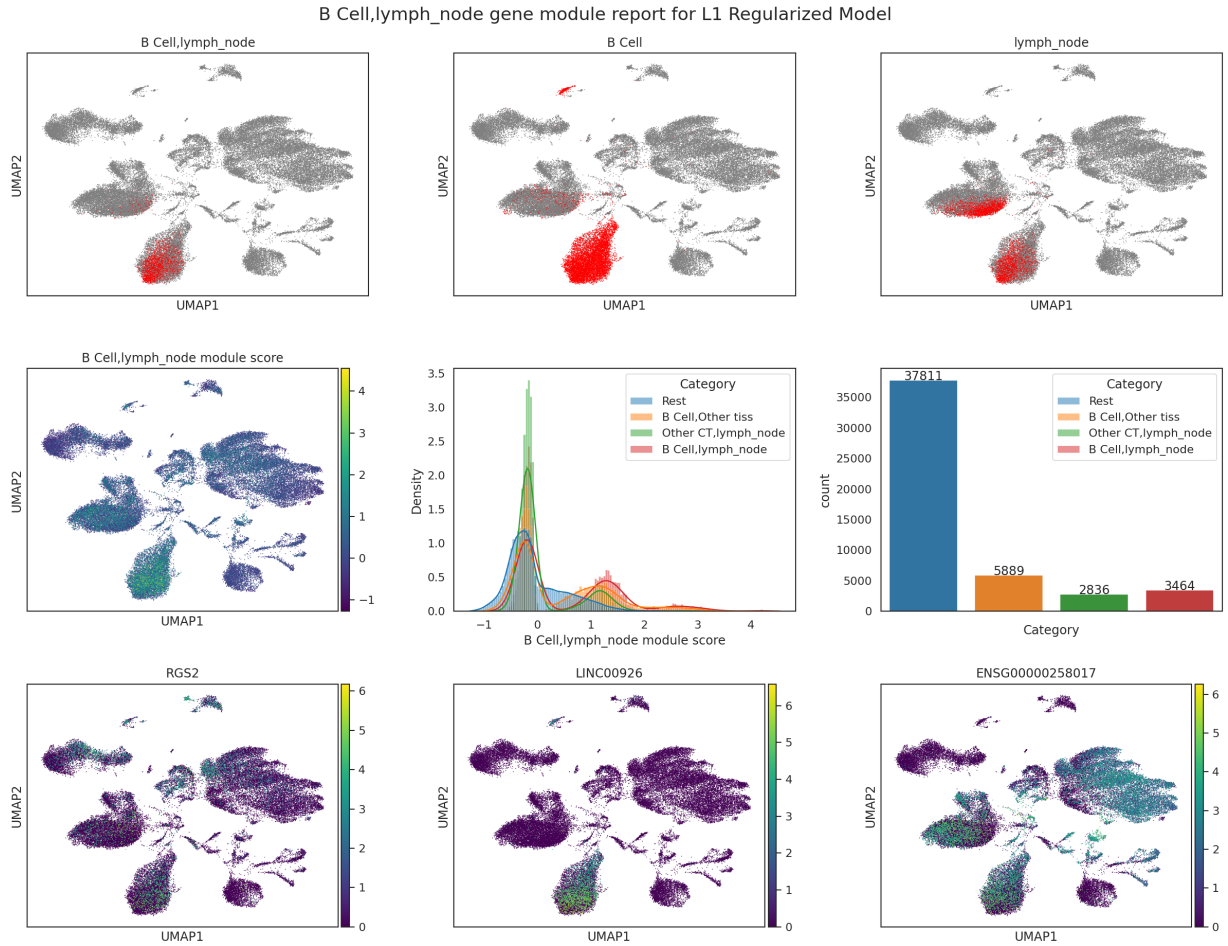


Figure 4.10: Visualization of the recovered exclusive markers for the human (B Cell,Lymph node) phenotype from the L1 regularized model. The subplots are drawn in the same fashion as in Figure 4.9.

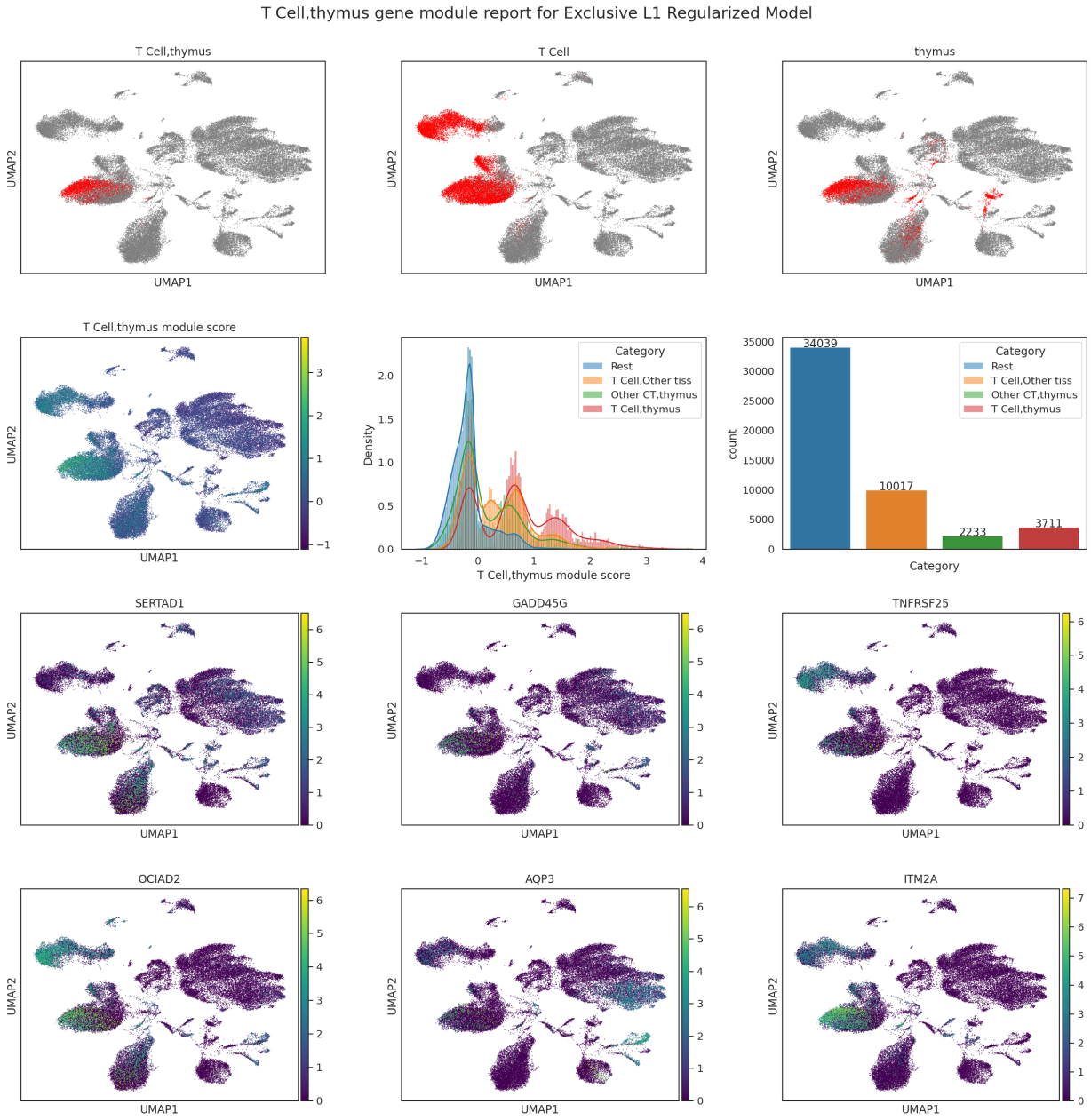


Figure 4.11: Visualization of the recovered exclusive markers for the human (T Cell,Thymus) phenotype from the Exclusive L1 regularized model. The subplots are drawn in the same fashion as in Figure 4.9.

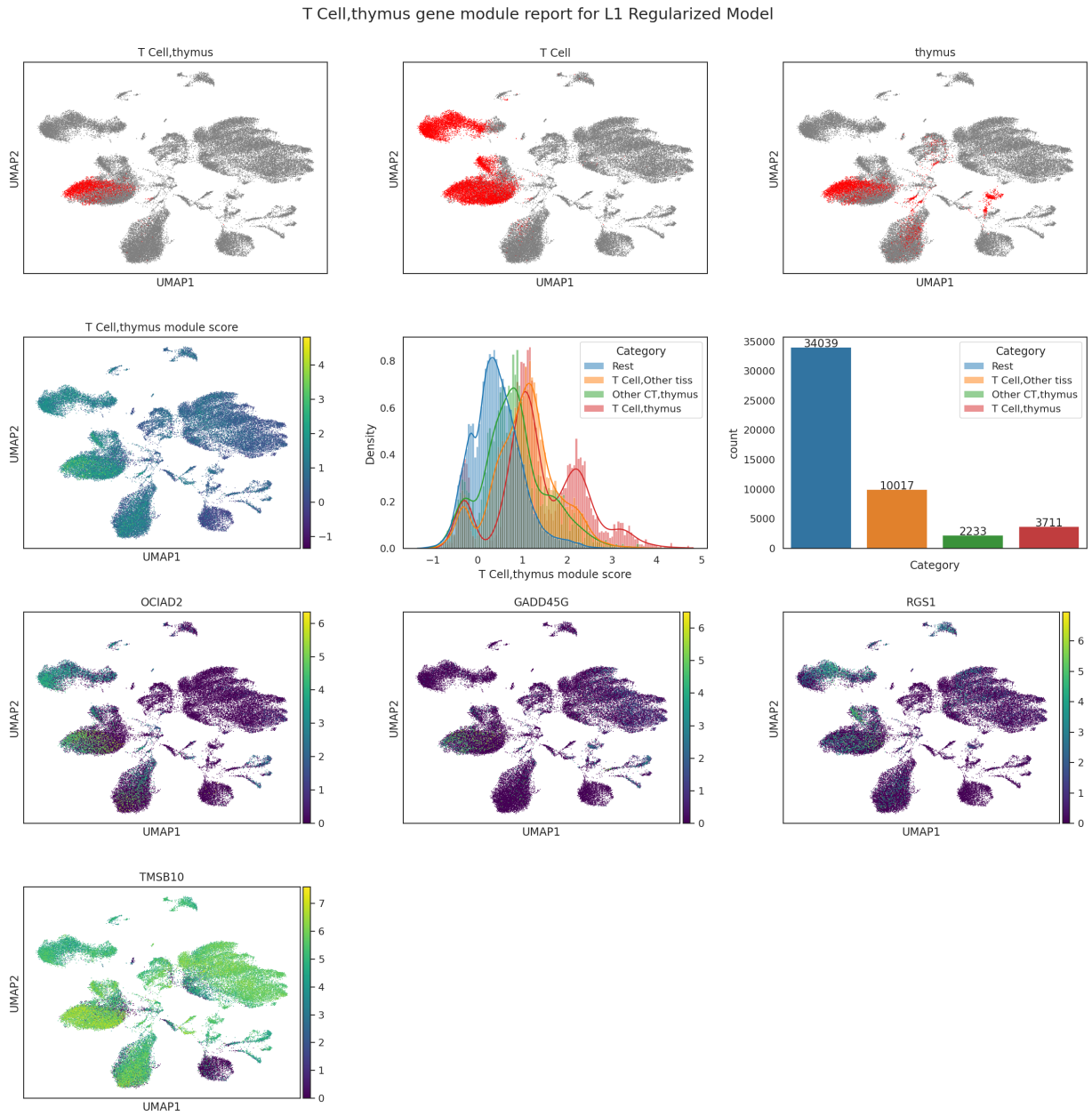
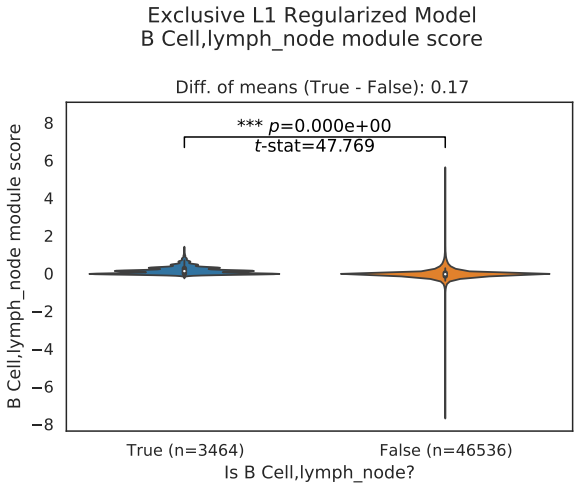
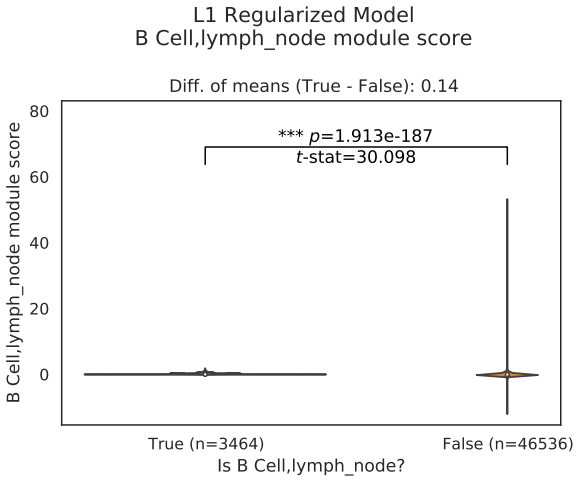


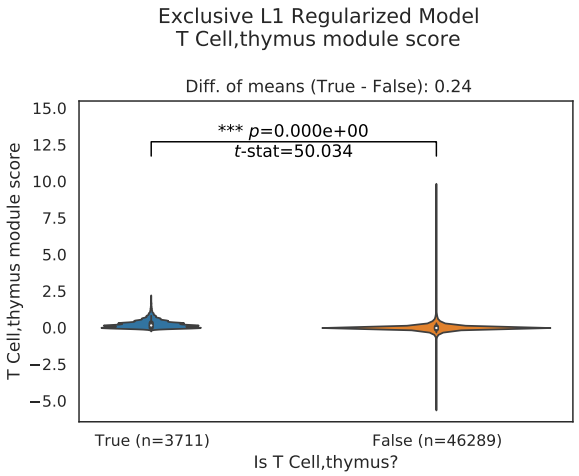
Figure 4.12: Visualization of the recovered exclusive markers for the human (T Cell,Thymus) phenotype from the L1 regularized model. The subplots are drawn in the same fashion as in Figure 4.9.



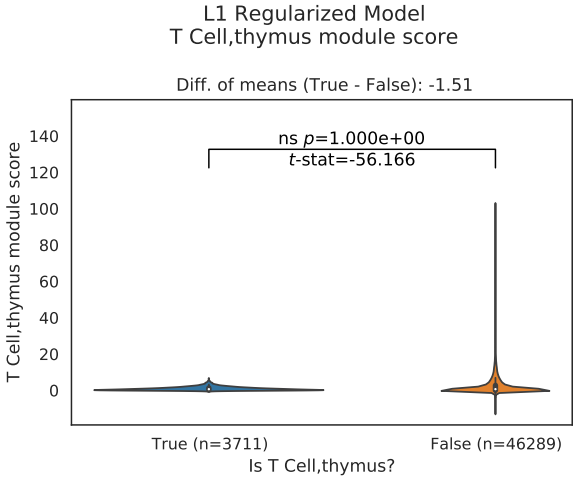
(a) Exclusive L1, (B Cell,Lymph node)



(b) L1, (B Cell,Lymph node)



(c) Exclusive L1, (T Cell, Thymus)



(d) L1, (T Cell, Thymus)

Figure 4.13: Module score distribution of exclusive marker lists across cell identities in the human dataset. In each subplot, we plot the distribution of the module score (the average expression of the genes in this marker list, relative to control gene sets, see section 4.4.3) for both the cells that have the same (cell type, tissue type) identity as the marker set ("True", left violin plot) and for all other cells ("False", right violin plot). We also report the number of cells in each category on the x-axis of each subplot. We then conduct a Welch's t-test (unequal variances) to determine if the mean score in the "True" cells is significantly higher than the mean score in the "False" cells, and report the p-value (one-sided) and the t-statistic from this test.

natural killer cells, which are a type of lymphocyte that are the chief immune effector cells of the innate immune system, and become activated in order to kill virus-infected cells in the body [204].

GO term	Corrected p-val
POSITIVE REGULATION OF IMMUNE RESPONSE	0.000289
IMMUNE EFFECTOR PROCESS	0.000670
REGULATION OF IMMUNE RESPONSE	0.000670
POSITIVE REGULATION OF IMMUNE SYSTEM PROCESS	0.000670
REGULATION OF IMMUNE SYSTEM PROCESS	0.000670
REGULATION OF LYMPHOCYTE MEDIATED IMMUNITY	0.002407
REGULATION OF IMMUNE EFFECTOR PROCESS	0.002667
ADAPTIVE IMMUNE RESPONSE	0.002667
RESPONSE TO BIOTIC STIMULUS	0.004559
LEUKOCYTE MEDIATED IMMUNITY	0.004559
REGULATION OF LEUKOCYTE MEDIATED IMMUNITY	0.004737
LYMPHOCYTE MEDIATED IMMUNITY	0.005974
CELL ACTIVATION	0.005974
REGULATION OF HYALURONAN BIOSYNTHETIC PROCESS	0.007679
HYALURONAN BIOSYNTHETIC PROCESS	0.010028
TOLL LIKE RECEPTOR 7 SIGNALING PATHWAY	0.016098
POSITIVE REGULATION OF LEUKOCYTE MEDIATED IMMU...	0.023503
CELL ACTIVATION INVOLVED IN IMMUNE RESPONSE	0.041207
INNATE IMMUNE RESPONSE	0.041630
DEFENSE RESPONSE	0.043680

Table 4.8: GO enrichment results of the exclusive marker list for the (NK Cell, PBMC) phenotype from the Exclusive L1 regularized model. The "Biological Process" domain of GO was used as the reference set.

4.4.5 Relaxing the exclusivity requirement in marker gene assignment

So far, the gene marker lists for each phenotype presented in sections 4.4.2, 4.4.3, and 4.4.4 have been constructed in an overly-restricted way. Specifically, we took the top k positively weighted features for each phenotype, and removed any genes that were shared between any of these top k lists. This can easily be achieved by repeated set subtractions. The result of this post-processing procedure is marker gene lists of length at most k which do not have any genes in common.

While this approach is useful for evaluating the ability of various models to identify phenotype-specific features, we may benefit from more relaxed post-processing procedures for practical application of these marker lists. This is because many of the phenotypes we consider (which are a combination of a cell type and tissue type) are very similar. For example, two phenotypes that share a cell type but are from different tissues. In these cases, we expect that their strongest markers would have significant overlap. Thus the post-processing procedure described above would remove these strong markers, resulting in less useful marker lists.

GO term	Corrected p-val
POSITIVE REGULATION OF IMMUNE RESPONSE	0.000953
ADAPTIVE IMMUNE RESPONSE	0.001041
REGULATION OF LYMPHOCYTE MEDIATED IMMUNITY	0.001041
CELL ACTIVATION	0.001041
REGULATION OF IMMUNE RESPONSE	0.001041
POSITIVE REGULATION OF IMMUNE SYSTEM PROCESS	0.001311
REGULATION OF LEUKOCYTE MEDIATED IMMUNITY	0.001945
LYMPHOCYTE MEDIATED IMMUNITY	0.002416
IMMUNE EFFECTOR PROCESS	0.006571
LYMPHOCYTE ACTIVATION	0.006596
REGULATION OF IMMUNE SYSTEM PROCESS	0.009519
TOLL LIKE RECEPTOR 7 SIGNALING PATHWAY	0.010313
LEUKOCYTE MEDIATED IMMUNITY	0.010313
REGULATION OF IMMUNE EFFECTOR PROCESS	0.010313
POSITIVE REGULATION OF LEUKOCYTE MEDIATED IMMU...	0.010313
RESPONSE TO KETONE	0.030485
REGULATION OF ADAPTIVE IMMUNE RESPONSE	0.033617
TOLL LIKE RECEPTOR 9 SIGNALING PATHWAY	0.036970
POSITIVE REGULATION OF IMMUNE EFFECTOR PROCESS	0.042226

Table 4.9: GO enrichment results of the exclusive marker list for the (NK Cell, PBMC) phenotype from the L1 regularized model. The "Biological Process" domain of GO was used as the reference set.

We are interested in exploring other post-processing procedures that would allow this overlap. Let us formally define our gene set assignment problem with a relaxed exclusivity requirement.

The gene set assignment problem. Given m phenotypes (groups) $P_1, P_2, ..., P_m$; g genes; a phenotype-specific value for each gene w_i (represented as a weight matrix $W \in \mathbb{R}^{m \times g}$, where w_i is a row that contains the values for all genes for a specific phenotype P_i); a user defined gene set size n; and a user defined maximum overlap size $0 \leq k \leq n$, the goal is to assign n-subsets of genes to each phenotype such to maximize the total value of the gene sets such that no two gene sets have an overlap larger than k genes.

The problem of assigning discrete objects (gene IDs in this case) to another set of objects (phenotypes in this case), while minimizing or maximizing other criterion (weights or set overlaps) is a combinatorial optimization problem. There are a number of well-studied combinatorial optimization problems that deal with assigning one type of object to another, but to the best of our knowledge, none are the same as our gene set assignment problem. These include the assignment problem [205], the lottery problem [206], set cover problems [207], and interval scheduling problems [208]. In this work we have not found a polynomial time reduction from one of these problems to our gene set assignment problem above, while one may exist.

Nonetheless, here we propose a greedy algorithm for the gene set assignment problem stated

above. Our algorithm is described by pseudocode in Algorithm 3. The basic idea is to sort all gene weights (where a gene has multiple weights, one for each phenotype) from largest to smallest, and simply proceed down the list, adding each weighted gene to its corresponding phenotype. At the same time, we keep track of the size of each phenotype gene set as it grows, as well as the maximum overlap that results from adding each gene. If a gene set has n, we consider it complete and stop adding to it. If adding a gene to a particular gene set increases the maximum overlap beyond k, we do not add it. We proceed down the list until all gene sets are complete. The complexity of this algorithm is $O(mg \log(mg))$, because it is dominated by the sorting operation on the elements of the weight matrix W.

We implemented Algorithm 3 and applied it to the fitted model weights of our Exclusive L1 regularized model trained on the human dataset. The result is visualized in Figure 4.14. Here we can clearly see that the phenotypes with the maximum overlap are always those that are from the same cell type or the same tissue, as expected (the maximum overlap in this example was set to $k = 5$). Furthermore, we can observe that the instances of relatively higher overlaps occur in diagonal strips in Figure 4.14, showing that there is stronger similarity in gene sets between phenotypes of the same cell type (the rows and columns are sorted by cell type). This result shows that this greedy algorithm is more flexible than the post processing algorithm we presented in the prior sections, and behaves as expected in terms of allowing gene sharing amongst similar phenotypes. We also analyzed the gene sets assigned by this algorithm to our phenotypes by scoring each cell by their relative expression of the genes in the gene sets compared to baseline genes, shown in Figure 4.15. This figure compares the scores from this algorithm to the scores for gene sets given via the original post-processing procedure we describe in Figure 4.2 panel 3, which did not allow any overlap between gene sets. We can see that across the selected phenotypes, the gene sets are specific to the phenotype of interest, clearly highlighting the subpopulation of interest with higher scores compared to other cells. Furthermore, we can see that in the case of "B Cell, spleen" and "T Cell, thymus", the greedy (relaxed) algorithm's gene sets are better able to uniquely identify the phenotype of interest, highlighting that subpopulation more clearly in the projections. On the other hand, in the case of "NK Cell, PBMC" and "T Cell, lung", the relaxed algorithm's gene sets are slightly less precise compared to the old no-overlap gene sets.

Algorithm 3: Greedy gene set assignment algorithm. It takes as input the weight matrix $\boldsymbol{W}$ and user-defined parameters n and k and returns the set of gene sets (one for each of the m phenotypes). The rows of the matrix $\boldsymbol{W}$ correspond to the m phenotypes, the columns correspond to g genes, and the value at W_{ij} is the weight (or importance score) of gene j for phenotype i. The sortElementsDescending($\boldsymbol{W}$) function below returns a sorted list of all the elements in the weight matrix $\boldsymbol{W}$ from largest to smallest (each element in the returned list retains its gene and phenotype IDs, accessed by "dot" accessors, see lines 4 and 5). The getMaxOverlap(A, B) function returns the max intersection size between the set B and all sets in A.

Input: $\boldsymbol{W} \in \mathbb{R}^{m \times g} \leftarrow$ Phenotype $\times$ gene weight matrix, $n \leftarrow$ gene set size, $k \leftarrow$ max overlap size

Result: $G = \{G_1, G_2, ..., G_m\} \equiv$ gene sets for each phenotype

1 $G \leftarrow \{\}$;
2 $S \leftarrow$ sortElementsDescending($\boldsymbol{W}$);
3 **foreach** S_r *in* S **do**
4 $\quad p \leftarrow S_r.phenotype$;
5 $\quad g \leftarrow S_r.gene$;
6 $\quad$ **if** $|G_p| == n$ **then**
$\quad\quad$ `/* Skip if phenotype's gene list done */`
7 $\quad\quad$ continue;
8 $\quad$ **end**
9 $\quad overlap \leftarrow$ getMaxOverlap($G \setminus G_p,\ G_p \cup g$);
10 $\quad$ **if** $overlap > k$ **then**
$\quad\quad$ `/* Skip if adding gene would result in too much overlap */`
11 $\quad\quad$ continue;
12 $\quad$ **end**
13 $\quad G_p \leftarrow G_p \cup g$
14 $\quad$ **if** $\min_{i=1,...,m} |G_i| == n$ **then**
$\quad\quad$ `/* Stop if every phenotype has been assigned` n `genes */`
15 $\quad\quad$ break;
16 $\quad$ **end**
17 **end**

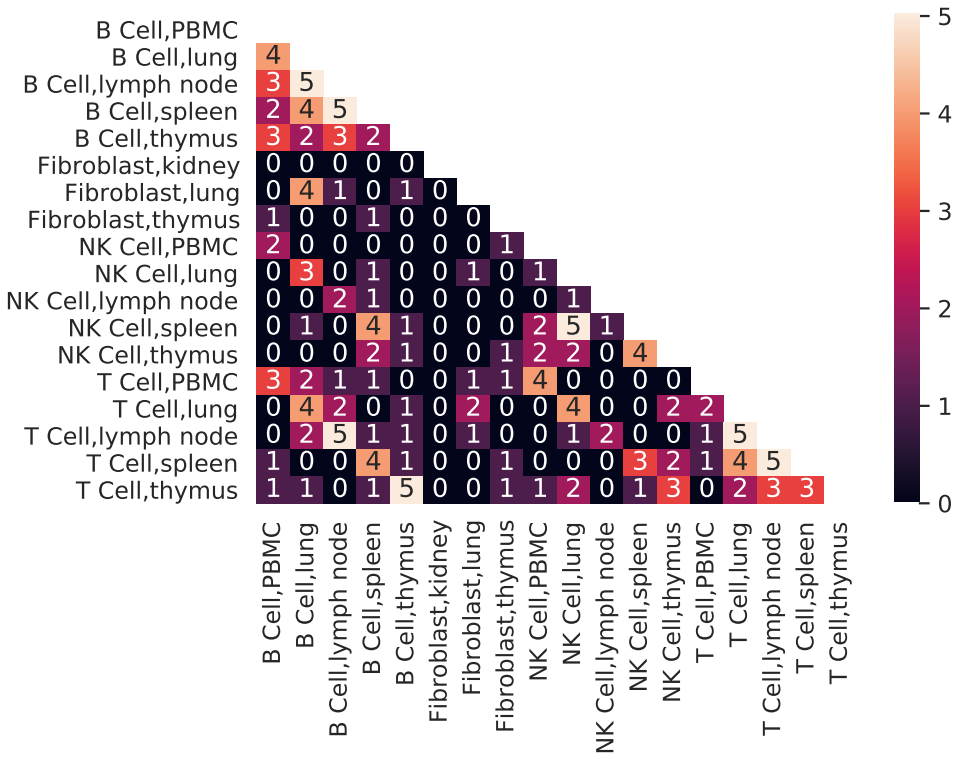
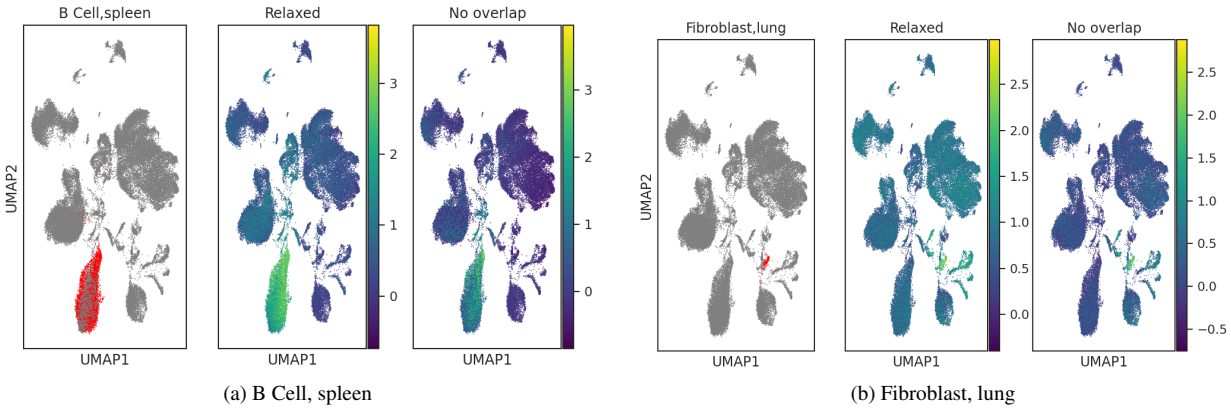
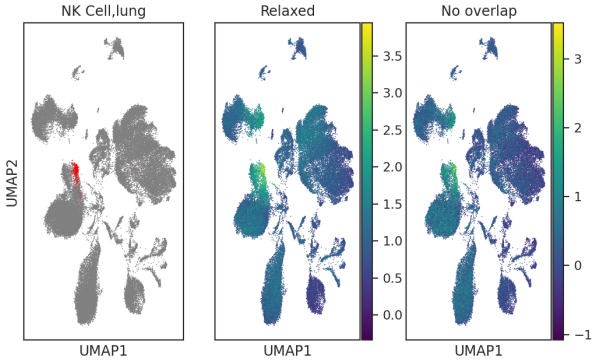


Figure 4.14: Overlaps among gene sets assigned via our greedy gene set assignment algorithm. We ran Algorithm 3 on the fitted model weights of our Exclusive L1 model from our human dataset to get marker lists ("gene sets") for our 18 phenotypes. We used parameters $n = 20$ as before, and $k = 5$ in this example. Each row and column represents the gene set for a particular phenotype. The values in the heatmap are the sizes of the overlap between the phenotypes.

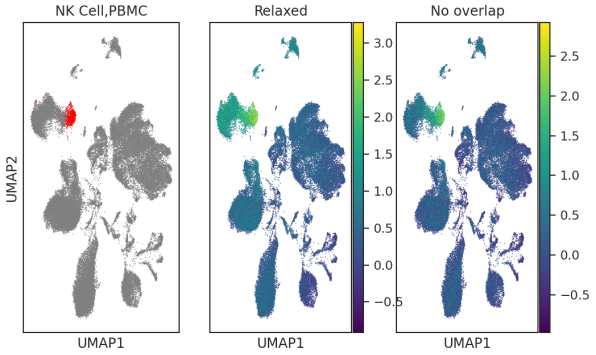


(a) B Cell, spleen

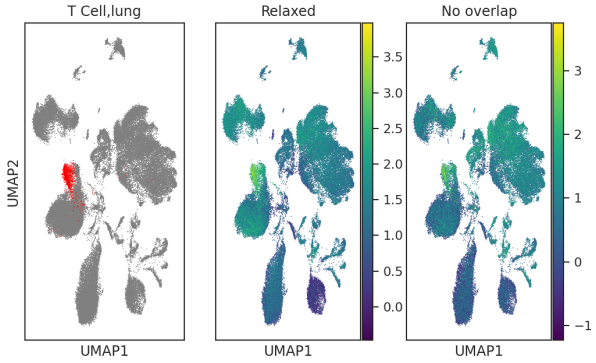
(b) Fibroblast, lung



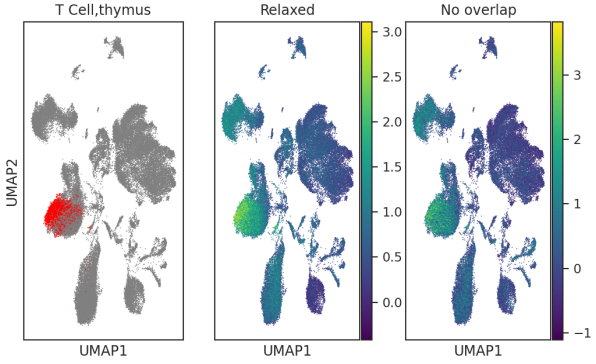
(c) NK Cell, lung



(d) NK Cell, PBMC



(e) T Cell, lung



(f) T Cell, thymus

Figure 4.15: Visualization of the recovered gene sets for the human dataset from the Exclusive L1 regularized model, comparing the greedy Algorithm 3 (relaxed) post-processing to the original no-overlaps-allowed post-processing described in Figure 4.2 panel 3. The data was subsetted to 50,000 cells for visualization purposes only. The left subplot in each subfigure highlights the cells in UMAP dimensions of a particular (cell type, tissue type) phenotype. The middle subplot shows the same UMAP embedding of the cells colored by their relative score of the gene set assigned for that phenotype via the greedy gene set assignment algorithm (Algorithm 3) which allows overlaps ("Relaxed"). The right subplot is the cells colored by their relative score of the gene set found via the original post-processing procedure that does not allow any overlap between gene sets ("No overlap").

4.5 Discussion

Over the last two decades several methods have been developed for identifying differential expressed genes between clusters, cell types, tissues and other phenotypes [19, 24–26, 209]. With the rise of atlas based efforts, more refined analysis is needed to identify genes that are unique to an intersection of phenotypes, for example for a specific cell type within a tissue. While such identification can be treated as a new label, such approach ignores the fact that these labels are no longer independent since several labels share the same cell type or tissue (though not both). Thus, new methods are required to find unique markers for such phenotype intersections

A few recent studies have looked at this problem, though to date these have mostly focused on the data collection aspect, and used traditional clustering and differential expression approaches to identify marker lists. Here we propose a different approach based on feature selection in a classification framework. Specifically, we propose a logistic regression method that uses exclusive lasso regularization to identify unique markers for such phenotypic pairs. This approach more directly addresses the task of finding unique cell type and tissue type-specific marker genes, compared to traditional differential expression-based approaches.

We tested our method on both human and mouse data from multiple tissues and cell types. We show that the Exclusive L1 regularization penalty can recover more distinct (exclusive) markers for each phenotype when compared to the L1 penalty, while retaining similar classification performance. We saw that the performance on the human data was better than on the mouse data. This is mostly due to the difference in dataset sizes, as there were over 30 times more cells in the human data compared to the mouse data. We also observed that in general, our approach performed the best on immune cell types compared to fibroblasts. This is likely because our human scRNA-seq dataset contained many more examples of immune cell types in multiple tissues.

As for the specific feature selection method, we show that the quality of the marker sets from the Exclusive L1 model are better than those from the L1 regularized model. For the (B Cell,lymph node) phenotype the list of markers from the Exclusive L1 model includes *POU2F2* (alias: *OCT-2*) which is a transcription factor that regulates the transcription of immunogloblin, an important gene for B lymphocytes [198] which has been found to be critical for B cell maturation [199]. Furthermore, it has been shown that *POU2F2* is overexpressed in lymph node tissue (`https://www.proteinatlas.org/ENSG00000028277-POU2F2/tissue`) [210]. The Exclusive L1 model's marker list for (B Cell,lymph node) also includes *HLA-DMB*, which is involved in peptide loading of MHC class II molecules that are expressed in antigen presenting cells (B cells are a main type of professional antigen presenting cells) [200] and is also found to be overexpressed in lymph node tissue (`https://www.genecards.org/cgi-bin/carddisp.pl?gene=HLA-DMB`) [197]. Another gene from the Exclusive L1 model's marker list that is relevant to this phenotype is *LYN*, which is involved in the B cell receptor signaling pathway and plays an important role in B cell activation and proliferation [201, 202], and is also found to be overexpressed in lymph node tissues [197]. Another example phenotype for which we observe more and better markers from our Exclusive L1 model is (T Cell,Thymus), where our model recovers the *ITM2A* gene, which is preferentially expressed in T cells and is actually a T cell-specific transcription factor, observed to be expressed in thymus tissue as well [203].

As the various atlas studies continue to collect more data, we expect that data for many more

tissues and cell types common to tissues will become available. This will likely require even more exclusive regularization since the number of (cell type, tissue) pairs will greatly increase (a product of the increase in cell types and tissues). On the other hand, when more cells are profiled we observe that the marker-finding ability of the Exclusive L1 and traditional L1 models becomes more similar. Thus, the tradeoff between the classification performance and the distinctness or "exclusivity" of the model coefficients is an issue that should be further studied and analyzed when more data is collected.

An important caveat of the method proposed here is that it relies on a user defined parameter k to determine how many of the top coefficients to consider when isolating the exclusive markers for each phenotype. We set this value to 20 in our experiments, as we believe that it is stringent enough that models that don't find enough exclusive markers will not produce significant enrichment results. This limitation is analogous to the choice of an arbitrary p-value cutoff in GSEA methods [211]. In the case of GSEA, studies have shown that the choice of this cutoff parameter can greatly effect the outcome of the enrichment analysis [212]. The question of how to determine a clear-cut boundary between included and excluded genes is another area for further study.

Here we focus on identifying markers for every cell type and tissue type combination. However, as datasets grow in size and coverage of cell types and tissues, the number of these cell type and tissue combinations also becomes large, and the large number of classes could make it difficult for accurate classification with a logistic regression-based model. To address this, future work could explore alternative modeling strategies where markers are identified for sets of cell types or tissues, rather than for each pair individually. For example, fitting specialized models for each cell type, to identify markers across tissues for that cell type. Another alternative would be to utilize prior knowledge about the similarities of phenotypes to reduce the number of unique cell types and tissues that our model must simultaneously consider. Specifically, we have hierarchical prior knowledge about the relationships between cell types and tissues in the form of the Cell Ontology (CL) [81] which we discussed and used in Chapter 2. Using this DAG, we can assign a cut-off depth on the tree, and merge labels below this depth to their closest parent nodes, significantly reducing both the number of cell types and the number of tissues that our model must consider. This is a reasonable thing to do because the cell types that are far from the root of the ontology are very specific cell types that occur in small numbers and are phenotypically very similar to their ancestor cell types.

Another consideration for future work is how to identify markers for sets rather than unique cell type and tissue combinations. Whereas above we propose approaches for pruning the number of these unique combinations and still seek to find markers for each combination, here we seek to identify a set of genes that are common to a number of tissues for the same cell type, but not to other tissues, or vice versa. Future work could consider hierarchical approaches to address this issue. For example, scGeneFit is a recent method which is "label hierarchy-aware" in that in addition to the scRNA-seq data, the method takes as input a hierarchical taxonomy of cell labels [213]. This could be expert provided (e.g. the Cell Ontology [81]), or inferred from hierarchical clustering of the data. The method seeks to find a projection to a lower dimensional space wherein samples with the same labels are closer together and samples with different labels are further apart. At the same time, the projection is constrained so that each coordinate in the lower dimensional space corresponds to a gene in the original representation. In other words, we

seek to find the lowest-dimensional subspace for which samples with different labels are further apart than samples with the same label [213]. Thus, finding a sparse projection of a cell in this subspace is akin to selecting marker genes. Label hierarchies are incorporated into the marker assignment by an additional constraint that encourages the distance between labeled cells in the lower dimensional space to reflect their distance in the hierarchy.

The mouse data we used in this study is publicly available from the Tabula Muris consortium [183], and the accession codes for the human scRNA-seq data we used can be found in Table 4.3. The recovered marker lists for all phenotypes we studied here, as well as the code for running our analyses are available in our code repository at `https://github.com/AmirAlavi/scRNA-seq-celltype-and-tissue-markers`.

Chapter 5

Conclusion and future work

In this thesis we presented computational methods for the analysis of scRNA-seq data. Our contributions have been aimed at advancing our ability to perform comparative analyses in order to gain biological insights from high-throughput transcriptomics data.

5.1 Summary of contributions

Across the different projects and chapters of this thesis, an overarching theme has been to develop or improve representation learning methods for scRNA-seq data to uncover functional cell states. The high dimensionality (tens of thousands of genes) and the high sample sizes (up to hundreds of thousands or even millions of single cells) in this data necessitate the application of representation learning methods that would make it more amenable to downstream tasks, but also to consumption by the human computational biologist. Here, "representation" means a set of features that may or may not be genes, which characterize a single cell (e.g. read counts from scRNA-seq, or an embedding of it) or a cell type. Throughout the thesis, the cellular states we choose to analyze are almost always the canonical cell types. In the first project (scQuery, Chapter 2), the representation we were concerned with was some lower dimensional embedding of a single cell's gene expression profile that highlighted differences in cell types. In the third chapter, sought to further refine these representations to make them invariant to batch effects. In the fourth chapter, we were seeking a representation of a cell type and tissue combination. Rather than learning a per-cell representation, here we can think of a list of marker genes as representing an entire cell state. Thus, the methods in this dissertation, and much of the computational research around scRNA-seq in the community, can be seen as developing and applying representation learning techniques to transcriptomics data.

5.1.1 scQuery

In Chapter 2, we developed scQuery [71], a tool that enabled comparative analysis of scRNA-seq data without the reliance on prior marker knowledge. At its heart, this method relies on three main pieces: a standardized (labeled) data collection pipeline, a dimensionality reduction step, and database interface using fast nearest neighbors search. Our labeling pipeline proved effec-

tive in automatically querying and processing data from over 500 different scRNA-seq studies. The simple text-matching labeling system was able to annotate over 300 different cell types, and our differential expression analysis results of these subpopulations returned biologically-relevant terms for each population. Analysis of this large data (almost 150K expression profiles with over 20,000 genes each) in its original dimensions is computationally expensive. For k-nearest neighbor (kNN) approaches, the assumption that the number of samples n is much greater than the number of dimensions d does not hold in this setting, and kNN is subject to the curse of dimensionality [214]. A key piece of scQuery is the dimensionality reduction method it uses to avoid these issues and enable efficient comparisons of scRNA-seq profiles. Unlike many prior approaches [16, 41, 215] which use unsupervised dimensionality reduction, we follow in the steps of Lin *et al.* [78] and develop supervised neural network classifiers to learn a neural embedding of the scRNA-seq data. With the large labeled data from our pipeline, we experimented with various architectures (including some that incorporate prior biological knowledge), and found that our supervised neural network embedders performed better than common unsupervised baselines. We also developed a web server that embedded both queries from users, and reference data from our large database into the same reduced dimensions using our neural embedding models, and did fast nearest neighbors-based retreival analysis in this space. We conducted our own case study where we used our approach to query the database with a held-out mouse neurodegeneration dataset, and found that we could find differences between the healthy and disease sub-populations in this data. Namely, scQuery highlighted that there was an immune response to the disease progression in the disease cells. By highlighting relevant biological differences, without relying on any prior biological knowledge of marker genes, we show that scQuery can be used as an effective tool for exploratory analysis of large scRNA-seq data.

5.1.2 scDGN

While scQuery utilized large heterogeneous data from multiple sources, it did not explicitly address the important issue of accounting for technical differences between these datasets. In Chapter 3, we developed two approaches to tackle this issue, which is also known as the "alignment problem" or "data integration" problem in the context of scRNA-seq data. Our first approach was scDGN [106], where we again appealed to a supervised approach that uses a neural network to learn an embedding. However, this time our objective is to not just learn an embedding that is discriminative for cell types, but one that is simultaneously invariant to batch effects that may come from using data from different labs, different sequencing technologies, and other sources of technical effects. We extended our neural networks from scQuery [71] with an additional adversarial loss function to discourage learning embedding vectors that are discriminative for batch effects. We then compared the use of our embeddings with those from other batch-correction methods for scRNA-seq, in the context of classifying cell types across batches. We saw that although the classification was significantly more accurate based on our embedding compared to the other approaches, scDGN requires many labeled training examples to learn an embedding that correctly aligns cell types across batches.

5.1.3 SCIPR

To alleviate the need for large labeled data, the second approach we discussed in Chapter 3 was SCIPR [147], an unsupervised method for scRNA-seq alignment. While other unsupervised approaches exist for this problem [29, 87, 152], we highlight that none exist that can both do the alignment in the original gene space, and learn a function to do the transformation. Maintaining the original gene space is important for interpretation of many downstream analysis, as gene identifiers allow use to apply common semantics to our findings. The ability to learn a transformation function is also important as it allows us to apply an alignment learned from training data to new data, instead of recomputing the alignment from scratch (which may be expensive on large data). Here, we took inspiration from point-cloud registration methods, namely the Iterative Closest Point (ICP) algorithm developed in the 1990s for robotics applications [157]. We adapted this approach to account for the particular challenges of scRNA-seq data. Specifically, we proposed new matching algorithms (one based on MNN [87] and another greedy algorithm that we developed ourselves) that would account for missing cell types across different scRNA-seq batches. We also used more complicated transformation functions to account for the higher dimensions and complexity of scRNA-seq data compared to 3D point clouds of rigid objects. We showed that SCIPR can successfully align scRNA-seq data on several datasets, improving over other methods used for this task. We also showed through model-inspection that SCIPR uses biologically-relevant genes in the transformation function to align the batches.

5.1.4 Regularized multi-label classification for identifying markers for cell type and tissue type combinations

Once cell types are assigned for different tissues, we can use these to identify unique markers for the same cell type in different tissues. In Chapter 4 we aim to find marker genes for a cell type that are also specific to its presence in a particular tissue. In other words, we want to find marker genes that are specific to the joint (cell type, tissue type) label, which are not also markers of the separate cell type or tissue type alone. While in the prior chapters cell type classification was used as a proxy to learn a lower dimensional representation (a neural embedding with some desirable properties), here classification serves as a proxy for learning important features of the input data (finding marker genes). In contrast to the traditional statistical differential expression analysis, we propose a machine learning approach, based on feature selection in a classification framework. We use a logistic regression-based classifier tasked with classifying the (cell type, tissue type) phenotype of each cell, and experiment with different regularization strategies to identify unique markers for such phenotypic pairs. We apply our approach to a previously studied mouse dataset, and also to a new human scRNA-seq dataset that includes data from the HuBMAP [115]. We show that our approach with the exclusive lasso regularizaton can recover more biologically relevant (cell type, tissue type) markers compared with the traditional lasso regularization.

5.2 Limitations

We have discussed the specific limitations of each project in this dissertation in their respective chapters. Nevertheless, it is important to reiterate these. Here we discuss them in a unified fashion, putting them in the context of not only the entire dissertation, but scRNA-seq analysis in general.

5.2.1 Access to labeled data

A defining characteristic of scQuery (Chapter 2), scDGN (Chapter 3), and the classifiers used in Chapter 4, was the fact that they were all based on supervised machine learning methods. This necessitates access to large, *labeled* training datasets. In other contexts, we may think of simple classifiers like logistic regression as not being very "data hungry". However, in the case of scRNA-seq, the high dimensionsionality and the large amount of variability (both biological and technical) mean that even the simplest classifiers require lots of training examples to accurately classify cell types. This is even more so the case when we use complicated models like multi-layered neural networks, as in scQuery [71]. While it's true that scRNA-seq has become very popular, and the amount of studies (and cells per study) has dramatically increased since 2009 [216], the portion of this data that has cell type labels is much smaller. For example, in scQuery we collected almost 150K scRNA-seq profiles from publicly available data, but only less than 40K cells remained when we filtered to just those cells for which we had high-confidence cell type labels. While we encountered this limitation in our work, others in the community are also facing this challenge. The Human Cell Atlas [113] and the HuBMAP [115] consortia are currently working on gathering cell type annotations for their diverse datasets. Recent work has focused on evaluating various methods for automatic cell type labeling in scRNA-seq data, including those based on marker genes, correlation with reference expression, and transfer learning via supervised classification [217].

Label noise and standardized labeling protocols

The challenge of acquiring labeled scRNA-seq data does not stop at gathering enough examples; it is critical that we are highly confident in the assigned labels, because otherwise the label noise will make it difficult to accurately classify cells. This is especially the case when we are dealing with cell sub-types, which could easily be mislabeled as the broader cell type and vice versa. We had to deal with this specific issue in scQuery and scDGN [71, 106]. There, we removed any cells for which we did not have a singular, high-confidence label. This reduced our training set size significantly.

Even when high-confidence, author-provided labels are available, we then encounter the problem of non-standard label sets. Each study may have a different name for the same cell type, and harmonizing the label schemas between them is a necessary step prior to combining them into a training set. In scQuery and scDGN, we used a standard label set, derived from the Cell Ontology [81]. For the classifiers used in Chapter 4, we were given datasets from various tissues that were independently labeled, and we manually harmonized the label sets by inspecting them for synonymous terms. This is an imperfect process, and is prone to incurring a labeling

bias from not only the different expert-labelers, but also the individual harmonizing the different datasets.

5.2.2 Defining ground truth of cell type assignments

While related to the issues discussed in section 5.2.1, the issue of defining ground truth cell type assignments in the first place is critical in scRNA-seq studies, and is highlighted in this section. While the advent of single cell resolution transcriptomics technologies has facilitated the precise characterization of cell types, there are still many challenges that makes defining "gold standard" cell type annotations somewhat of a moving target. For example, purifying a cell type for scRNA-seq analysis is itself a challenge, as many cell types do not have reliable surface markers [218]. Even in the cases where we do have markers for purification, there may be significant biological variation in our population that we are unable to resolve (e.g. sub-types). Sub-types may share expression of markers used to purify them. In an ideal world, we would be able to isolate sub-types accurately, and yet we would still find it difficult to distinguish them from each other based on gene expression profiles, as there is significant variablility in expression within a sub-type [218]. Proper characterization will require not only reliable markers, but also a sense of how much variation we can reasonably expect within a cell type. For these reasons, defining ground truth cell type assignments remains very challenging, and as a result, there is a scarcity of reference cell type signatures [219]. We recommend that any analyses based on cell type-annotated scRNA-seq data be treated as hypothesis generation tools to guide exploratory research and further experimental validation.

5.2.3 Ground truth and evaluation in batch effect correction

In Chapter 3 we evaluated the quality of our SCIPR alignment method by using the iLISI and cLISI metrics, as in Korsunsky *et al.* [169], which measured how well the batches were integrated and how much the cell types were mixed, respectively. We mentioned that good integration is indicated by a high iLISI score while simultaneously keeping a low cLISI score. However, a drawback of this approach is that it is not clear how much the right trade-off between these two quantities is. In many multi-study analyses, the cell type and batch label distributions are tightly coupled [71, 106], and mixing one comes at the cost of potentially over-mixing the other. We currently do not have clear metrics on what "over-mixing" is, because we do not have ground-truth examples of what the right batch correction should look like. In addition to LISI scores, several other metrics have been developed to assess batch effect correction, and this is still an active area of research [168]. In fact, there are several recent reviews that compare and evaluate many different scRNA-seq batch effect correction methods [153, 220]. They also encounter the ground-truth issue, in that they do not settle on a single metric to evaluate the quality of the batch correction. Instead, they compute the scores from multiple batch-effect correction metrics and report all of them. Interestingly, they find that the various metrics did not always agree with each other. Furthermore, they also did not always agree with their qualitative observations from t-SNE and UMAP visualizations [220].

5.2.4 Access to matched cell type populations across many conditions

For several of our methods, it was not only important to have many examples of labeled scRNA-seq profiles, but it was also critical that we observe enough examples of each cell type across some other condition variable. For example, in scDGN [106] and SCIPR [147] it was critical that we had enough examples of each cell type in each of the various batches we were seeking to integrate. This was necessary for our models to be able to learn the differences of each cell type across diverse batches in order to correctly integrate them.

In the case of the classifiers used in Chapter 4, we required many examples of each cell type for each of the different tissues we were considering in our analysis. This requirement meant that we were not able to use all cell types that we had in our data, and only used the subset that occurred in large enough numbers in multiple tissues. We observed how this hindered learning in the case of the Tabula Muris mouse data, which was a much smaller dataset, compared to the larger human dataset we gathered. We hope that others continue in the direction of the Tabula Muris consortium in collecting many labeled scRNA-seq samples across diverse organs and tissues, in a controlled and standardized environment [183].

5.3 Future work

5.3.1 Learning representations that combine gene expression data and spatial information

In addition to collecting high resolution scRNA-seq data, recent advances in *in situ* sequencing technology have also allowed us to collect spatial information about the expression of genes within complex tissues [118, 221, 222]. This additional layer of information can help further characterize cellular relationships and gene regulatory patterns in heterogeneous tissues, for example descendent-precursor relationships in developing tissues, or metastasis in tumors. Integrating this data with the vast collection of scRNA-seq data for tissues could lead to richer representations for these tasks. However, in integrating this data with prior scRNA-seq data we will inherently face the batch effect problem. In addition, current limitations with spatial technology (such as profiling fewer genes than traditional scRNA-seq and with higher variance) mean that the set of features in this data may not have enough overlap with other datasets.

The scRNA-seq alignment and dimensionality reduction methods discussed in this dissertation could be extended to include spatial genomics data, and recent frameworks such as Pamona [223] look to be promising starting points for extending this line of work. Pamona is a manifold alignment method using an optimal transport framework. Because it is a partial manifold alignment method, it can handle heterogeneity among datasets, and can even handle different data modalities (i.e. transcriptomics and epigenomic data). For example, to integrate spatial location data with scRNA-seq data using Pamona, one could treat the spatial data as a separate modality. This would result in a final embedding that incorporates cell-cell similarities based not only on gene expression profiles, but also on x, y, and z locations.

5.3.2 Experimentation with signal-preserving penalties in batch-correction methods

In Chapter 3, we presented SCIPR [147] and scDGN [106] for aligning scRNA-seq data (batch correction). While the supervised scDGN had cell type labels to rely on for preserving the cell type identities, our unsupervised SCIPR did not have an explicit label it could use to ensure it preserved biological signal during its transformation. SCIPR is not alone in this aspect, as many of the popular scRNA-seq alignment methods are also unsupervised [29, 87, 152]. A future direction of research is to explore the use of various penalties during the fitting procedure, which seek to preserve input characteristics of the data. ScAlign did this by using the same penalty as in t-SNE [224]. The basic intuition is that pairwise distances between cells in the original data space should be preserved as much as possible after transformation, while still allowing for batch alignment. We could similarly apply this as a penalty term to our fitting procedure in SCIPR, as well as explore the use of other such penalties.

5.3.3 Collection of larger and more diverse scRNA-seq data

The biggest limitation of the methods we have proposed in this thesis has been lack of enough data. Future directions of research involve gathering more data, from many more conditions. For example, through collaboration with researchers in the HuBMAP consortium [115], we can collect more data from other tissue types. We currently have data from four HuBMAP tissues in the analysis discussed in Chapter 4 (spleen, thymus, lymph node, and kidney). The HuBMAP has samples from seven tissues so far, and we could gather scRNA-seq profiles from these remaining organs. As the goal of the HuBMAP is to build a comprehensive cellular atlas of the human body, this and similar efforts [113] represent the best opportunity to gather the large multi-tissue scRNA-seq dataset that we require for answering the pressing research questions such as the one we pose in Chapter 4.

We will also need to work with our collaborators to make sure we collect high-confidence labels for this data, from an agreed-upon label set that would be suitable for integrated downstream analysis. With this larger, and higher-quality dataset in hand, we could reapply and reassess the methods proposed in this dissertation on this data.

5.4 Final remarks

The work in this thesis has addressed important challenges in the analysis of scRNA-seq data. While none of these problems are considered "solved", we hope that the contributions put forth here represent important steps forward in computational methods development for scRNA-seq data. It is an exciting time to be involved in this sub-field of computational biology research, and all indications point to many more years of excitement, attention, and investment in this area.

Bibliography

1. Macosko, E. Z. *et al.* Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell* **161,** 1202–1214 (2015) (cit. on pp. 1, 2, 65, 66).
2. Tang, F. *et al.* mRNA-Seq whole-transcriptome analysis of a single cell. *Nature methods* **6,** 377 (2009) (cit. on p. 1).
3. Agresti, J. J. *et al.* Ultrahigh-throughput screening in drop-based microfluidics for directed evolution. *Proceedings of the National Academy of Sciences* **107,** 4004–4009 (2010) (cit. on pp. 1, 65).
4. Klein, A. M. *et al.* Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell* **161,** 1187–1201 (2015) (cit. on pp. 1, 65, 67).
5. Zheng, G. X. *et al.* Massively parallel digital transcriptional profiling of single cells. *Nature communications* **8,** 1–12 (2017) (cit. on pp. 1, 65–67).
6. Rosenberg, A. B. *et al.* Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science* **360,** 176–182 (2018) (cit. on p. 1).
7. Cao, J. *et al.* Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science* **357,** 661–667 (2017) (cit. on p. 1).
8. Jaitin, D. A. *et al.* Massively parallel single-cell RNA-seq for marker-free decomposition of tissues into cell types. *Science* **343,** 776–779 (2014) (cit. on pp. 1, 10, 18, 52, 55, 65).
9. Papalexi, E. & Satija, R. Single-cell RNA sequencing to explore immune cell heterogeneity. *Nature Reviews Immunology* **18,** 35 (2018) (cit. on pp. 1, 52, 65).
10. Zhu, Q., Shah, S., Dries, R., Cai, L. & Yuan, G.-C. Identification of spatially associated subpopulations by combining scRNAseq and sequential fluorescence in situ hybridization data. *Nature biotechnology* **36,** 1183–1190 (2018) (cit. on pp. 1, 65).
11. Medaglia, C. *et al.* Spatial reconstruction of immune niches by combining photoactivatable reporters and scRNA-seq. *Science* **358,** 1622–1626 (2017) (cit. on pp. 1, 65).
12. Raj, B. *et al.* Simultaneous single-cell profiling of lineages and cell types in the vertebrate brain. *Nature biotechnology* **36,** 442–450 (2018) (cit. on pp. 1, 65).
13. Spanjaard, B. *et al.* Simultaneous lineage tracing and cell-type identification using CRISPR–Cas9-induced genetic scars. *Nature biotechnology* **36,** 469–473 (2018) (cit. on pp. 1, 65).
14. Islam, S. *et al.* Quantitative single-cell RNA-seq with unique molecular identifiers. *Nature methods* **11,** 163 (2014) (cit. on p. 2).
15. Patro, R., Duggal, G., Love, M. I., Irizarry, R. A. & Kingsford, C. Salmon provides fast and bias-aware quantification of transcript expression. *Nature methods* **14,** 417–419 (2017) (cit. on pp. 2, 101).

16. Pierson, E. & Yau, C. ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome biology* **16,** 241 (2015) (cit. on pp. 4, 8, 19, 134).
17. Gong, W., Kwak, I.-Y., Pota, P., Koyano-Nakagawa, N. & Garry, D. J. DrImpute: imputing dropout events in single cell RNA sequencing data. *BMC bioinformatics* **19,** 1–10 (2018) (cit. on p. 4).
18. Qiu, P. Embracing the dropouts in single-cell RNA-seq analysis. *Nature communications* **11,** 1–9 (2020) (cit. on p. 4).
19. Kharchenko, P. V., Silberstein, L. & Scadden, D. T. Bayesian approach to single-cell differential expression analysis. *Nature methods* **11,** 740 (2014) (cit. on pp. 4, 6, 22, 30, 129).
20. Li, W. V. & Li, J. J. An accurate and robust imputation method scImpute for single-cell RNA-seq data. *Nature communications* **9,** 1–9 (2018) (cit. on p. 4).
21. Van Dijk, D. *et al.* Recovering gene interactions from single-cell data using data diffusion. *Cell* **174,** 716–729 (2018) (cit. on p. 4).
22. Rao, M. S. *et al.* Comparison of RNA-Seq and microarray gene expression platforms for the toxicogenomic evaluation of liver from short-term rat toxicity studies. *Frontiers in genetics* **9,** 636 (2019) (cit. on p. 4).
23. Zeng, Y. *et al.* Single-cell RNA sequencing resolves spatiotemporal development of pre-thymic lymphoid progenitors and thymus organogenesis in human embryos. *Immunity* **51,** 930–948 (2019) (cit. on p. 5).
24. Love, M. I., Huber, W. & Anders, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome biology* **15,** 1–21 (2014) (cit. on pp. 5, 129).
25. Robinson, M. D., McCarthy, D. J. & Smyth, G. K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26,** 139–140 (2010) (cit. on pp. 5, 129).
26. McDavid, A. *et al.* Data exploration, quality control and testing in single-cell qPCR-based gene expression experiments. *Bioinformatics* **29,** 461–467 (2013) (cit. on pp. 5, 129).
27. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* **1,** 80–83. ISSN: 00994987. http://www.jstor.org/stable/3001968 (1945) (cit. on p. 6).
28. Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome biology* **19,** 1–5 (2018) (cit. on pp. 6, 10, 72, 76, 101, 105, 114).
29. Stuart, T. *et al.* Comprehensive Integration of Single-Cell Data. *Cell* **177,** 1888–1902. https://doi.org/10.1016/j.cell.2019.05.031 (2019) (cit. on pp. 6, 10, 52, 57, 65, 76, 78, 84, 101, 105, 135, 139).
30. Hao, Y. *et al.* Integrated analysis of multimodal single-cell data. *bioRxiv*. https://doi.org/10.1101/2020.10.12.335331 (2020) (cit. on pp. 6, 10, 105).
31. Soneson, C. & Robinson, M. D. Bias, robustness and scalability in single-cell differential expression analysis. *Nature methods* **15,** 255 (2018) (cit. on p. 6).
32. Subramanian, A. *et al.* Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences* **102,** 15545–15550 (2005) (cit. on pp. 6, 117).

33. Chen, L. & Wong, G. in *Encyclopedia of Bioinformatics and Computational Biology* (eds Ranganathan, S., Gribskov, M., Nakai, K. & Schönbach, C.) 324–340 (Academic Press, Oxford, 2019). ISBN: 978-0-12-811432-2. https://www.sciencedirect.com/science/article/pii/B9780128096338202045 (cit. on p. 6).
34. Ashburner, M. *et al.* Gene ontology: tool for the unification of biology. *Nature genetics* **25,** 25–29 (2000) (cit. on pp. 6, 78, 109).
35. Consortium, T. G. O. The Gene Ontology resource: enriching a GOld mine. *Nucleic Acids Research* **49,** D325–D334. ISSN: 0305-1048. https://doi.org/10.1093/nar/gkaa1113 (Dec. 2020) (cit. on p. 6).
36. Kanehisa, M. & Goto, S. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research* **28,** 27–30. ISSN: 0305-1048. https://doi.org/10.1093/nar/28.1.27 (Jan. 2000) (cit. on p. 6).
37. Kanehisa, M. Toward understanding the origin and evolution of cellular organisms. *Protein Science* **28,** 1947–1951. https://onlinelibrary.wiley.com/doi/abs/10.1002/pro.3715 (2019) (cit. on p. 6).
38. Kanehisa, M., Furumichi, M., Sato, Y., Ishiguro-Watanabe, M. & Tanabe, M. KEGG: integrating viruses and cellular organisms. *Nucleic Acids Research* **49,** D545–D551. ISSN: 0305-1048. https://doi.org/10.1093/nar/gkaa970 (Oct. 2020) (cit. on p. 6).
39. Tarca, A. L., Bhatti, G. & Romero, R. A comparison of gene set analysis methods in terms of sensitivity, prioritization and specificity. *PloS one* **8,** e79217 (2013) (cit. on p. 6).
40. Pearson, K. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2,** 559–572 (1901) (cit. on p. 8).
41. Yau, C. *et al.* pcaReduce: hierarchical clustering of single cell transcriptional profiles. *BMC bioinformatics* **17,** 140 (2016) (cit. on pp. 8, 19, 134).
42. Van der Maaten, L. & Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research* **9** (2008) (cit. on p. 8).
43. McInnes, L., Healy, J. & Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018) (cit. on p. 8).
44. Becht, E. *et al.* Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology* **37,** 38–44 (2019) (cit. on pp. 8, 9).
45. Wong, M. T. *et al.* A high-dimensional atlas of human T cell diversity reveals tissue-specific trafficking and cytokine signatures. *Immunity* **45,** 442–456 (2016) (cit. on p. 9).
46. Kiselev, V. Y., Andrews, T. S. & Hemberg, M. Challenges in unsupervised clustering of single-cell RNA-seq data. *Nature Reviews Genetics* **20,** 273–282 (2019) (cit. on p. 9).
47. Lloyd, S. Least squares quantization in PCM. *IEEE transactions on information theory* **28,** 129–137 (1982) (cit. on p. 9).
48. Kiselev, V. Y. *et al.* SC3: consensus clustering of single-cell RNA-seq data. *Nature methods* **14,** 483–486 (2017) (cit. on p. 9).
49. Grün, D. *et al.* Single-cell messenger RNA sequencing reveals rare intestinal cell types. *Nature* **525,** 251–255 (2015) (cit. on p. 9).

50. Levine, J. H. *et al.* Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. *Cell* **162,** 184–197 (2015) (cit. on p. 10).
51. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* **2008,** P10008 (2008) (cit. on p. 10).
52. Traag, V. A., Waltman, L. & Van Eck, N. J. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports* **9,** 1–12 (2019) (cit. on p. 10).
53. Usoskin, D. *et al.* Unbiased classification of sensory neuron types by large-scale single-cell RNA sequencing. *Nature neuroscience* **18,** 145–153 (2015) (cit. on pp. 10, 18, 100).
54. Zhang, X. *et al.* CellMarker: a manually curated resource of cell markers in human and mouse. *Nucleic acids research* **47,** D721–D728 (2019) (cit. on p. 10).
55. Alquicira-Hernandez, J., Sathe, A., Ji, H. P., Nguyen, Q. & Powell, J. E. scPred: accurate supervised method for cell-type classification from single-cell RNA-seq data. *Genome biology* **20,** 1–17 (2019) (cit. on p. 10).
56. Wagner, F. & Yanai, I. Moana: A robust and scalable cell type classification framework for single-cell RNA-Seq data. *BioRxiv,* 456129 (2018) (cit. on pp. 10, 66).
57. Trapnell, C. *et al.* The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature biotechnology* **32,** 381–386 (2014) (cit. on p. 11).
58. Nowotschin, S. *et al.* The emergent landscape of the mouse gut endoderm at single-cell resolution. *Nature* **569,** 361–367 (2019) (cit. on pp. 11, 52, 69).
59. Mathys, H. *et al.* Temporal tracking of microglia activation in neurodegeneration at single-cell resolution. *Cell reports* **21,** 366–380 (2017) (cit. on pp. 11, 18, 39, 43, 46).
60. Achim, K. *et al.* High-throughput spatial mapping of single-cell RNA-seq data to tissue of origin. *Nature biotechnology* **33,** 503–509 (2015) (cit. on pp. 11, 12).
61. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **5,** 115–133 (1943) (cit. on p. 12).
62. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *nature* **323,** 533–536 (1986) (cit. on pp. 12, 14).
63. LeCun, Y. *et al.* Backpropagation applied to handwritten zip code recognition. *Neural computation* **1,** 541–551 (1989) (cit. on p. 12).
64. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* **25,** 1097–1105 (2012) (cit. on p. 12).
65. Chopra, S., Hadsell, R. & LeCun, Y. *Learning a similarity metric discriminatively, with application to face verification* in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* **1** (2005), 539–546 (cit. on pp. 14, 20).
66. Koch, G., Zemel, R. & Salakhutdinov, R. *Siamese neural networks for one-shot image recognition* in *ICML deep learning workshop* **2** (2015) (cit. on pp. 14, 20, 33, 53, 55).
67. Schroff, F., Kalenichenko, D. & Philbin, J. *Facenet: A unified embedding for face recognition and clustering* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), 815–823 (cit. on pp. 14, 20, 21, 26, 27, 33).

68. Ustinova, E. & Lempitsky, V. *Learning Deep Embeddings with Histogram Loss* in *Advances in Neural Information Processing Systems* (eds Lee, D., Sugiyama, M., Luxburg, U., Guyon, I. & Garnett, R.) **29** (Curran Associates, Inc., 2016). https://proceedings.neurips.cc/paper/2016/file/325995af77a0e8b06d1204a171010b3a-Paper.pdf (cit. on p. 14).
69. Martín Abadi *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015. https://www.tensorflow.org/ (cit. on p. 14).
70. Paszke, A. *et al.* in *Advances in Neural Information Processing Systems 32* (eds Wallach, H. *et al.*) 8024–8035 (Curran Associates, Inc., 2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on pp. 14, 64).
71. Alavi, A., Ruffalo, M., Parvangada, A., Huang, Z. & Bar-Joseph, Z. A web server for comparative analysis of single-cell RNA-seq data. *Nature communications* **9,** 4768 (2018) (cit. on pp. 17, 54, 55, 57, 133, 134, 136, 137).
72. Kolodziejczyk, A., Kim, J. K., Svensson, V., Marioni, J. & Teichmann, S. The Technology and Biology of Single-Cell RNA Sequencing. *Molecular Cell* **58,** 610–620. ISSN: 1097-2765. http://www.sciencedirect.com/science/article/pii/S1097276515002610 (2015) (cit. on p. 17).
73. Wills, Q. F. *et al.* Single-cell gene expression analysis reveals genetic associations masked in whole-tissue experiments. *Nature biotechnology* **31,** 748–752 (2013) (cit. on p. 17).
74. Zeisel, A. *et al.* Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science* **347,** 1138–1142 (2015) (cit. on pp. 18, 52, 55).
75. Patel, A. P. *et al.* Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science,* 1254257 (2014) (cit. on p. 18).
76. Lescroart, F. *et al.* Defining the earliest step of cardiovascular lineage segregation by single-cell RNA-seq. *Science,* eaao4174 (2018) (cit. on p. 18).
77. Rizvi, A. H. *et al.* Single-cell topological RNA-seq analysis reveals insights into cellular differentiation and development. *Nature biotechnology* **35,** 551 (2017) (cit. on p. 18).
78. Lin, C., Jain, S., Kim, H. & Bar-Joseph, Z. Using neural networks for reducing the dimensions of single-cell RNA-Seq data. *Nucleic Acids Research* **45,** e156. http://dx.doi.org/10.1093/nar/gkx681 (2017) (cit. on pp. 18–20, 28, 30, 33, 36, 55, 57, 58, 134).
79. Rosenbloom, K. R. *et al.* The UCSC genome browser database: 2015 update. *Nucleic acids research* **43,** D670–D681 (2014) (cit. on p. 18).
80. Kim, D., Langmead, B. & Salzberg, S. L. HISAT: a fast spliced aligner with low memory requirements. *Nature methods* **12,** 357–360 (2015) (cit. on p. 18).
81. Bard, J., Rhee, S. Y. & Ashburner, M. An ontology for cell types. *Genome biology* **6,** R21 (2005) (cit. on pp. 19, 130, 136).
82. Tirosh, I. *et al.* Dissecting the multicellular ecosystem of metastatic melanoma by single-cell RNA-seq. *Science* **352,** 189–196 (2016) (cit. on pp. 19, 114).

83. Hinton, G. E. & Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science* **313,** 504–507. ISSN: 0036-8075. eprint: http://science.sciencemag.org/content/313/5786/504.full.pdf.http://science.sciencemag.org/content/313/5786/504 (2006) (cit. on p. 19).
84. Chollet, F. *et al. Keras* https://github.com/fchollet/keras. 2015 (cit. on p. 20).
85. Consortium, G. O. *et al.* Expansion of the Gene Ontology knowledgebase and resources. *Nucleic acids research* **45,** D331–D338 (2017) (cit. on pp. 20, 31).
86. Butler, A., Hoffman, P., Smibert, P., Papalexi, E. & Satija, R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology* **36,** 411 (2018) (cit. on p. 22).
87. Haghverdi, L., Lun, A. T., Morgan, M. D. & Marioni, J. C. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nature biotechnology* **36,** 421–427 (2018) (cit. on pp. 22, 52, 57, 65, 72, 76, 77, 79, 135, 139).
88. Law, C. W., Chen, Y., Shi, W. & Smyth, G. K. voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome biology* **15,** R29 (2014) (cit. on p. 22).
89. Boytsov, L. & Naidan, B. *Engineering Efficient and Effective Non-metric Space Library* in *Similarity Search and Applications - 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings* (2013), 280–293. https://doi.org/10.1007/978-3-642-41062-8_28 (cit. on p. 24).
90. Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I. & Schmidt, L. *Practical and optimal LSH for angular distance* in *Advances in Neural Information Processing Systems* (2015), 1225–1233 (cit. on p. 24).
91. Hermans, A., Beyer, L. & Leibe, B. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017) (cit. on p. 27).
92. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* **11,** 3371–3408 (2010) (cit. on p. 28).
93. Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. *Greedy layer-wise training of deep networks* in *Advances in neural information processing systems* (2007), 153–160 (cit. on p. 28).
94. Troyanskaya, O. *et al.* Missing value estimation methods for DNA microarrays. *Bioinformatics* **17,** 520–525 (2001) (cit. on p. 30).
95. Reimand, J. *et al.* g: Profiler—a web server for functional interpretation of gene lists (2016 update). *Nucleic acids research* **44,** W83–W89 (2016) (cit. on p. 31).
96. Eden, E., Navon, R., Steinfeld, I., Lipson, D. & Yakhini, Z. GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists. *BMC bioinformatics* **10,** 48 (2009) (cit. on p. 31).
97. Schulz, M. H. *et al.* DREM 2.0: Improved reconstruction of dynamic regulatory networks from time-series expression data. *BMC systems biology* **6,** 104 (2012) (cit. on p. 31).

98. Szklarczyk, D. *et al.* STRING v10: protein–protein interaction networks, integrated over the tree of life. *Nucleic acids research* **43,** D447–D452 (2014) (cit. on p. 31).
99. Smith, C. L. *et al.* Mouse Genome Database (MGD)-2018: knowledgebase for the laboratory mouse. *Nucleic acids research* **46,** D836–D842 (2017) (cit. on p. 32).
100. Mouse Genome Informatics, T. J. L. *Mouse Genome Database (MGD)* `http://www.informatics.jax.org` (2017) (cit. on p. 32).
101. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86,** 2278–2324 (1998) (cit. on p. 32).
102. Park, T. I.-H. *et al.* Adult human brain neural progenitor cells (NPCs) and fibroblast-like cells have similar properties in vitro but only NPCs differentiate into neurons. *PloS one* **7,** e37742. `https://doi.org/10.1371/journal.pone.0037742` (2012) (cit. on p. 42).
103. Vanlandewijck, M. *et al.* A molecular atlas of cell types and zonation in the brain vasculature. *Nature* **554,** 475 (2018) (cit. on p. 42).
104. Hickman, S. E. *et al.* The microglial sensome revealed by direct RNA sequencing. *Nature neuroscience* **16,** 1896 (2013) (cit. on pp. 42, 45).
105. Zhang, Y. *et al.* An RNA-sequencing transcriptome and splicing database of glia, neurons, and vascular cells of the cerebral cortex. *Journal of Neuroscience* **34,** 11929–11947 (2014) (cit. on p. 43).
106. Ge, S., Wang, H., Alavi, A., Xing, E. & Bar-Joseph, Z. *Supervised Adversarial Alignment of Single-Cell RNA-seq Data* in *Research in Computational Molecular Biology* (ed Schwartz, R.) (Springer International Publishing, Cham, 2020), 72–87. ISBN: 978-3-030-45257-5 (cit. on pp. 52, 66, 134, 136–139).
107. Ge, S., Wang, H., Alavi, A., Xing, E. & Bar-joseph, Z. Supervised Adversarial Alignment of Single-Cell RNA-seq Data. *Journal of Computational Biology* **28.** PMID: 33470876, 501–513. eprint: `https://doi.org/10.1089/cmb.2020.0439`. `https://doi.org/10.1089/cmb.2020.0439` (2021) (cit. on p. 52).
108. Hwang, B., Lee, J. H. & Bang, D. Single-cell RNA sequencing technologies and bioinformatics pipelines. *Experimental & molecular medicine* **50,** 1–14 (2018) (cit. on p. 52).
109. Yu, Y. *et al.* Single-cell RNA-seq identifies a PD-1 hi ILC progenitor and defines its development pathway. *Nature* **539,** 102 (2016) (cit. on pp. 52, 55).
110. Villani, A.-C. *et al.* Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. *Science* **356,** eaah4573 (2017) (cit. on p. 52).
111. Tung, P.-Y. *et al.* Batch effects and the effective design of single-cell gene expression studies. *Scientific reports* **7,** 39921 (2017) (cit. on p. 52).
112. Stuart, T. & Satija, R. Integrative single-cell analysis. *Nature Reviews Genetics,* 1 (2019) (cit. on p. 52).
113. Rozenblatt-Rosen, O., Stubbington, M. J., Regev, A. & Teichmann, S. A. The Human Cell Atlas: from vision to reality. *Nature News* **550,** 451 (2017) (cit. on pp. 52, 65, 99, 136, 139).
114. Regev, A. *et al.* Science forum: the human cell atlas. *Elife* **6,** e27041 (2017) (cit. on pp. 52, 65, 99).

115. Consortium, H. *et al.* The human body at cellular resolution: the NIH Human Biomolecular Atlas Program. *Nature* **574,** 187 (2019) (cit. on pp. 52, 65, 99, 101, 135, 136, 139).
116. Pijuan-Sala, B. *et al.* A single-cell molecular map of mouse gastrulation and early organogenesis. *Nature* **566,** 490–495 (2019) (cit. on pp. 52, 69).
117. Wang, G., Moffitt, J. R. & Zhuang, X. Multiplexed imaging of high-density libraries of RNAs with MERFISH and expansion microscopy. *Scientific reports* **8,** 4847 (2018) (cit. on p. 52).
118. Eng, C.-H. L. *et al.* Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+. *Nature* **568,** 235 (2019) (cit. on pp. 52, 138).
119. Lopez, R., Regier, J., Cole, M. B., Jordan, M. I. & Yosef, N. Deep generative modeling for single-cell transcriptomics. *Nature methods* **15,** 1053–1058 (2018) (cit. on pp. 53, 57, 65).
120. Motiian, S., Piccirilli, M., Adjeroh, D. A. & Doretto, G. *Unified deep supervised domain adaptation and generalization* in *ICCV* **2** (2017), 3 (cit. on pp. 53, 55).
121. Csurka, G. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374* (2017) (cit. on pp. 53, 55).
122. Wang, H., He, Z., Lipton, Z. C. & Xing, E. P. Learning robust representations by projecting superficial statistics out. *arXiv preprint arXiv:1903.06256* (2019) (cit. on p. 53).
123. Chu, C. & Wang, R. A survey of domain adaptation for neural machine translation. *arXiv preprint arXiv:1806.00258* (2018) (cit. on p. 53).
124. Patel, V. M., Gopalan, R., Li, R. & Chellappa, R. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine* **32,** 53–69 (2015) (cit. on p. 53).
125. Ganin, Y. *et al.* Domain-adversarial training of neural networks. *The Journal of Machine Learning Research* **17,** 2096–2030 (2016) (cit. on pp. 53, 55).
126. Li, H., Pan, S. J., Wang, S. & Kot, A. C. *Domain generalization with adversarial feature learning* in *CVPR* (2018) (cit. on p. 53).
127. Wang, H., Ge, S., Xing, E. P. & Lipton, Z. C. Learning Robust Global Representations by Penalizing Local Predictive Power. *arXiv preprint arXiv:1905.13549* (2019) (cit. on p. 53).
128. Kiselev, V. Y., Yiu, A. & Hemberg, M. scmap: projection of single-cell RNA-seq data across data sets. *Nature methods* **15,** 359 (2018) (cit. on p. 54).
129. Lieberman, Y., Rokach, L. & Shay, T. Castle–classification of single cells by transfer learning: Harnessing the power of publicly available single cell rna sequencing experiments to annotate new experiments. *PloS one* **13** (2018) (cit. on pp. 54, 57).
130. Hadsell, R., Chopra, S. & LeCun, Y. *Dimensionality reduction by learning an invariant mapping* in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* **2** (2006), 1735–1742 (cit. on p. 55).
131. Pei, Z., Cao, Z., Long, M. & Wang, J. *Multi-adversarial domain adaptation* in *AAAI Conference on Artificial Intelligence* (2018) (cit. on p. 55).
132. Ding, J. *et al.* Systematic comparative analysis of single cell RNA-sequencing methods. *BioRxiv,* 632216 (2019) (cit. on p. 57).

133. Ge, S., Wang, H., Alavi, A., Xing, E. & Bar-Joseph, Z. Supporting Information for: Supervised Adversarial Alignment of scRNA-seq Data. *bioRxiv*. eprint: https://www.biorxiv.org/content/early/2020/01/07/2020.01.06.896621.full.pdf. https://www.biorxiv.org/content/early/2020/01/07/2020.01.06.896621 (2020) (cit. on pp. 57, 58, 60–62, 64).
134. Ribeiro, M. T., Singh, S. & Guestrin, C. *Why should i trust you?: Explaining the predictions of any classifier* in *SIGKDD* (2016), 1135–1144 (cit. on pp. 60, 64).
135. Erhan, D., Bengio, Y., Courville, A. & Vincent, P. Visualizing higher-layer features of a deep network. *University of Montreal* **1341,** 1 (2009) (cit. on p. 60).
136. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013) (cit. on pp. 60, 61).
137. Springenberg, J. T., Dosovitskiy, A., Brox, T. & Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014) (cit. on p. 60).
138. Redgrave, T. Chylomicron metabolism. *Biochemical Society Transactions* **32,** 79–82. ISSN: 0300-5127 (Feb. 2004) (cit. on p. 61).
139. Hara, T., Tan, Y. & Huang, L. In vivo gene delivery to the liver using reconstituted chylomicron remnants as a novel nonviral vector. *Proceedings of the National Academy of Sciences* **94,** 14547–14552 (1997) (cit. on p. 61).
140. in (ed Komoda, T.) 35–59 (Academic Press, Boston, 2010) (cit. on p. 61).
141. Murakami, T. *et al.* Triglycerides are major determinants of cholesterol esterification/transfer and HDL remodeling in human plasma. *Arteriosclerosis, thrombosis, and vascular biology* **15,** 1819–1828 (1995) (cit. on p. 61).
142. Seidman, M. A., Mitchell, R. N. & Stone, J. R. in *Cellular and Molecular Pathobiology of Cardiovascular Disease* (eds Willis, M. S., Homeister, J. W. & Stone, J. R.) 221–237 (Academic Press, San Diego, 2014). ISBN: 978-0-12-405206-2 (cit. on p. 61).
143. Domingo-Espín, J., Nilsson, O., Bernfur, K., Giudice, R. D. & Lagerstedt, J. O. Site-specific glycations of apolipoprotein A-I lead to differentiated functional effects on lipid-binding and on glucose metabolism. *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease* **1864,** 2822–2834. ISSN: 0925-4439 (2018) (cit. on p. 62).
144. Ko, H.-L. *et al.* Apolipoprotein C1 (APOC 1) as a novel diagnostic and prognostic biomarker for lung cancer: A marker phase I trial. *Thoracic cancer* **5,** 500–508 (2014) (cit. on p. 62).
145. Hooker, S., Erhan, D., Kindermans, P.-J. & Kim, B. Evaluating feature importance estimates. *arXiv preprint arXiv:1806.10758* (2018) (cit. on p. 64).
146. Sun, Y. *et al. Test-time training with self-supervision for generalization under distribution shifts* in *International Conference on Machine Learning* (2020), 9229–9248 (cit. on p. 64).
147. Alavi, A. & Bar-Joseph, Z. Iterative point set registration for aligning scRNA-seq data. *PLOS Computational Biology* **16,** 1–21. https://doi.org/10.1371/journal.pcbi.1007939 (Oct. 2020) (cit. on pp. 65, 135, 138, 139).
148. Rosenberg, A. B. *et al.* Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science* **360,** 176–182. ISSN: 0036-8075. eprint: https://

science.sciencemag.org/content/360/6385/176.full.pdf. https://science.sciencemag.org/content/360/6385/176 (2018) (cit. on p. 65).
149. Cao, J. *et al.* Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science* **357,** 661–667. ISSN: 0036-8075. eprint: https://science.sciencemag.org/content/357/6352/661.full.pdf. https://science.sciencemag.org/content/357/6352/661 (2017) (cit. on p. 65).
150. Muraro, M. J. *et al.* A single-cell transcriptome atlas of the human pancreas. *Cell systems* **3,** 385–394 (2016) (cit. on p. 65).
151. Saunders, A. *et al.* Molecular diversity and specializations among the cells of the adult mouse brain. *Cell* **174,** 1015–1030 (2018) (cit. on p. 65).
152. Johansen, N. & Quon, G. scAlign: a tool for alignment, integration, and rare cell identification from scRNA-seq data. *Genome biology* **20,** 1–21 (2019) (cit. on pp. 65, 76, 135, 139).
153. Tian, L. *et al.* Benchmarking single cell RNA-sequencing analysis pipelines using mixture control experiments. *Nature methods* **16,** 479–487 (2019) (cit. on pp. 66, 137).
154. Luecken, M. D. *et al.* Benchmarking atlas-level data integration in single-cell genomics. *BioRxiv* (2020) (cit. on p. 66).
155. Chazarra-Gil, R., van Dongen, S., Kiselev, V. Y. & Hemberg, M. Flexible comparison of batch correction methods for single-cell RNA-seq using BatchBench. *bioRxiv* (2020) (cit. on p. 66).
156. Pomerleau, F., Colas, F., Siegwart, R., *et al.* A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics* **4,** 1–104 (2015) (cit. on pp. 66, 67).
157. Besl, P. & McKay, N. D. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14,** 239–256 (1992) (cit. on pp. 66, 68, 74, 77, 85, 135).
158. Tian, L. *et al.* scPipe: A flexible R/Bioconductor preprocessing pipeline for single-cell RNA-sequencing data. *PLOS Computational Biology* **14,** 1–15. https://doi.org/10.1371/journal.pcbi.1006361 (Aug. 2018) (cit. on pp. 66, 68).
159. Hashimshony, T. *et al.* CEL-Seq2: sensitive highly-multiplexed single-cell RNA-Seq. *Genome biology* **17,** 77 (2016) (cit. on p. 66).
160. Baron, M. *et al.* A single-cell transcriptomic map of the human and mouse pancreas reveals inter-and intra-cell population structure. *Cell systems* **3,** 346–360 (2016) (cit. on pp. 66, 68).
161. Ding, J. *et al.* Systematic comparison of single-cell and single-nucleus RNA-sequencing methods. *Nature biotechnology,* 1–10 (2020) (cit. on pp. 67, 68).
162. Godin, G., Rioux, M. & Baribeau, R. *Three-dimensional registration using range and intensity information* in *Videometrics III* **2350** (1994), 279–290 (cit. on p. 69).
163. Ahuja, R. K., Magnanti, T. L. & Orlin, J. B. in *Network Flows: Theory, Algorithms, and Applications* 470–473 (Prentice Hall, Englewood Cliffs, N.J, 1993) (cit. on p. 70).

164. Jungnickel, D. in *Graphs, Networks and Algorithms* 321–339 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005). ISBN: 978-3-540-26908-3. https://doi.org/10.1007/3-540-26908-8_11 (cit. on p. 70).
165. Tarjan, R. E. Dynamic trees as search trees via euler tours, applied to the network simplex algorithm. *Mathematical Programming* **78,** 169–177 (1997) (cit. on p. 70).
166. Hagberg, A. A., Schult, D. A. & Swart, P. J. *Exploring Network Structure, Dynamics, and Function using NetworkX* in *Proceedings of the 7th Python in Science Conference* (eds Varoquaux, G., Vaught, T. & Millman, J.) (Pasadena, CA USA, 2008), 11–15 (cit. on p. 72).
167. Satija, R., Farrell, J. A., Gennert, D., Schier, A. F. & Regev, A. Spatial reconstruction of single-cell gene expression data. *Nature biotechnology* **33,** 495–502 (2015) (cit. on pp. 72, 76).
168. Büttner, M., Miao, Z., Wolf, F. A., Teichmann, S. A. & Theis, F. J. A test metric for assessing single-cell RNA-seq batch correction. *Nature methods* **16,** 43 (2019) (cit. on pp. 75, 137).
169. Korsunsky, I. *et al.* Fast, sensitive and accurate integration of single-cell data with Harmony. *Nature methods,* 1–8 (2019) (cit. on pp. 75, 77–79, 137).
170. Kang, C. *An implementation of MNN (Mutual Nearest Neighbors) correct in python* https://github.com/chriscainx/mnnpy. 2019 (cit. on p. 76).
171. Fischer, D. S., Hölzlwimmer, F. & Theis, F. J. *Fast and scalable differential expression analysis on single-cell RNA-seq data* https://github.com/theislab/diffxpy. 2020 (cit. on pp. 78, 109).
172. Benjamini, Y. & Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* **57,** 289–300 (1995) (cit. on pp. 78, 109).
173. Consortium, G. O. The gene ontology resource: 20 years and still GOing strong. *Nucleic acids research* **47,** D330–D338 (2019) (cit. on pp. 78, 109).
174. Weinstein, J. N. *et al.* The cancer genome atlas pan-cancer analysis project. *Nature genetics* **45,** 1113–1120 (2013) (cit. on p. 99).
175. Hawrylycz, M. J. *et al.* An anatomically comprehensive atlas of the adult human brain transcriptome. *Nature* **489,** 391–399 (2012) (cit. on p. 99).
176. Giudice, Q. L., Leleu, M., La Manno, G. & Fabre, P. J. Single-cell transcriptional logic of cell-fate specification and axon guidance in early-born retinal neurons. *Development* **146** (2019) (cit. on p. 100).
177. Bassett, E. A. & Wallace, V. A. Cell fate determination in the vertebrate retina. *Trends in neurosciences* **35,** 565–573 (2012) (cit. on p. 100).
178. Janeway Jr, C. A., Travers, P., Walport, M. & Shlomchik, M. J. in *Immunobiology: The Immune System in Health and Disease. 5th edition* (Garland Science, 2001) (cit. on p. 100).
179. Heath, W. R. in *Encyclopedia of Immunology (Second Edition)* (ed Delves, P. J.) Second Edition, 2341–2343 (Elsevier, Oxford, 1998). ISBN: 978-0-12-226765-9. https://www.sciencedirect.com/science/article/pii/B0122267656006058 (cit. on p. 100).

180. Ravkov, E., Slev, P. & Heikal, N. Thymic output: Assessment of CD4+ recent thymic emigrants and T-Cell receptor excision circles in infants. *Cytometry Part B: Clinical Cytometry* **92,** 249–257. https://onlinelibrary.wiley.com/doi/abs/10.1002/cyto.b.21341 (2017) (cit. on p. 100).
181. Zhang, L. *et al.* Measuring recent thymic emigrants in blood of normal and HIV-1–infected individuals before and after effective therapy. *The Journal of experimental medicine* **190,** 725–732 (1999) (cit. on p. 100).
182. Ronning, K. E., Karlen, S. J., Miller, E. B. & Burns, M. E. Molecular profiling of resident and infiltrating mononuclear phagocytes during rapid adult retinal degeneration using single-cell RNA sequencing. *Scientific reports* **9,** 1–12 (2019) (cit. on p. 100).
183. Consortium, T. M. *et al.* Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. *Nature* **562,** 367–372 (2018) (cit. on pp. 100–102, 131, 138).
184. Ntranos, V., Yi, L., Melsted, P. & Pachter, L. A discriminative learning approach to differential expression analysis for single-cell RNA-seq. *Nature methods* **16,** 163–166 (2019) (cit. on pp. 100, 109).
185. Adams, T. S. *et al.* Single-cell RNA-seq reveals ectopic and aberrant lung-resident cell populations in idiopathic pulmonary fibrosis. *Science Advances* **6.** eprint: https://advances.sciencemag.org/content/6/28/eaba1983.full.pdf. https://advances.sciencemag.org/content/6/28/eaba1983 (2020) (cit. on p. 101).
186. Van Der Wijst, M. G. *et al.* Single-cell RNA sequencing identifies celltype-specific cis-eQTLs and co-expression QTLs. *Nature genetics* **50,** 493–497 (2018) (cit. on pp. 101, 102).
187. Satija, R., Farrell, J. A., Gennert, D., Schier, A. F. & Regev, A. Spatial reconstruction of single-cell gene expression data. *Nature Biotechnology* **33,** 495–502. https://doi.org/10.1038/nbt.3192 (2015) (cit. on p. 105).
188. Butler, A., Hoffman, P., Smibert, P., Papalexi, E. & Satija, R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology* **36,** 411–420. https://doi.org/10.1038/nbt.4096 (2018) (cit. on pp. 105, 114).
189. Le Cessie, S. & van Houwelingen, J. C. Ridge Estimators in Logistic Regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **41,** 191–201. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.2307/2347628. https://rss.onlinelibrary.wiley.com/doi/abs/10.2307/2347628 (1992) (cit. on p. 105).
190. La Cava, W., Bauer, C., Moore, J. H. & Pendergrass, S. A. *Interpretation of machine learning predictions for patient outcomes in electronic health records* in *AMIA Annual Symposium Proceedings* **2019** (2019), 572 (cit. on p. 106).
191. Saarela, M. & Jauhiainen, S. Comparison of feature importance measures as explanations for classification models. *SN Applied Sciences* **3,** 1–12 (2021) (cit. on p. 106).
192. Zhou, Y., Jin, R. & Hoi, S. C.-.-H. *Exclusive lasso for multi-task feature selection* in *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), 988–995 (cit. on p. 106).

193. Paszke, A. *et al.* in *Advances in Neural Information Processing Systems 32* (eds Wallach, H. *et al.*) 8024–8035 (Curran Associates, Inc., 2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on p. 108).
194. Tietz, M., Fan, T. J., Nouri, D., Bossan, B. & skorch Developers. *skorch: A scikit-learn compatible neural network library that wraps PyTorch* (July 2017). https://skorch.readthedocs.io/en/stable/ (cit. on p. 108).
195. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011) (cit. on pp. 108, 109).
196. Carpenter, B. Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. *Alias-i, Inc., Tech. Rep,* 1–20 (2008) (cit. on p. 108).
197. Fishilevich, S. *et al.* Genic insights from integrated human proteomics in GeneCards. *Database* **2016.** baw030. ISSN: 1758-0463. eprint: https://academic.oup.com/database/article-pdf/doi/10.1093/database/baw030/17473276/baw030.pdf. https://doi.org/10.1093/database/baw030 (Mar. 2016) (cit. on pp. 114, 129).
198. García-Cosío, M. *et al.* Analysis of transcription factor OCT. 1, OCT. 2 and BOB. 1 expression using tissue arrays in classical Hodgkin's lymphoma. *Modern pathology* **17,** 1531–1538 (2004) (cit. on pp. 114, 129).
199. Corcoran, L. M. *et al.* Oct-2, although not required for early B-cell development, is critical for later B-cell maturation and for postnatal survival. *Genes & development* **7,** 570–582 (1993) (cit. on pp. 114, 129).
200. Yin, L., Maben, Z. J., Becerra, A. & Stern, L. J. Evaluating the role of HLA-DM in MHC class II–peptide association reactions. *The Journal of Immunology* **195,** 706–716 (2015) (cit. on pp. 114, 129).
201. Harwood, N. E. & Batista, F. D. Early Events in B Cell Activation. *Annual Review of Immunology* **28.** PMID: 20192804, 185–210. eprint: https://doi.org/10.1146/annurev-immunol-030409-101216. https://doi.org/10.1146/annurev-immunol-030409-101216 (2010) (cit. on pp. 114, 129).
202. Gauld, S. B. & Cambier, J. C. Src-family kinases in B-cell development and signaling. *Oncogene* **23,** 8001–8006 (2004) (cit. on pp. 114, 129).
203. Tai, T.-S., Pai, S.-Y. & Ho, I.-C. Itm2a, a Target Gene of GATA-3, Plays a Minimal Role in Regulating the Development and Function of T Cells. *PLOS ONE* **9,** 1–9. https://doi.org/10.1371/journal.pone.0096535 (May 2014) (cit. on pp. 117, 129).
204. Vivier, E., Tomasello, E., Baratin, M., Walzer, T. & Ugolini, S. Functions of natural killer cells. *Nature immunology* **9,** 503–510 (2008) (cit. on p. 123).
205. Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics* **5,** 32–38 (1957) (cit. on p. 124).
206. Grundlingh, W. R. *Two new combinatorial problems involving dominating sets for lottery schemes* PhD thesis (Stellenbosch: University of Stellenbosch, 2004) (cit. on p. 124).
207. Karp, R. M. in *Complexity of computer computations* 85–103 (Springer, 1972) (cit. on p. 124).

208. Gamzu, I. & Koutsopoulos, I. *Advertisement allocation and mechanism design in native stream advertising* in *International conference on complex networks and their applications* (2018), 197–210 (cit. on p. 124).
209. Smyth, G. K. in *Bioinformatics and computational biology solutions using R and Bioconductor* 397–420 (Springer, 2005) (cit. on p. 129).
210. Uhlén, M. *et al.* Tissue-based map of the human proteome. *Science* **347** (2015) (cit. on p. 129).
211. Simillion, C., Liechti, R., Lischer, H. E., Ioannidis, V. & Bruggmann, R. Avoiding the pitfalls of gene set enrichment analysis with SetRank. *BMC bioinformatics* **18,** 1–14 (2017) (cit. on p. 130).
212. Pan, K.-H., Lih, C.-J. & Cohen, S. N. Effects of threshold choice on biological conclusions reached during analysis of gene expression by DNA microarrays. *Proceedings of the National Academy of Sciences* **102,** 8961–8965 (2005) (cit. on p. 130).
213. Dumitrascu, B., Villar, S., Mixon, D. G. & Engelhardt, B. E. Optimal marker gene selection for cell type discrimination in single cell analyses. *Nature communications* **12,** 1–8 (2021) (cit. on pp. 130, 131).
214. Kouiroukidis, N. & Evangelidis, G. *The effects of dimensionality curse in high dimensional knn search* in *2011 15th Panhellenic Conference on Informatics* (2011), 41–45 (cit. on p. 134).
215. Wang, B., Zhu, J., Pierson, E., Ramazzotti, D. & Batzoglou, S. Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nature methods* **14,** 414 (2017) (cit. on p. 134).
216. Angerer, P. *et al.* Single cells make big data: New challenges and opportunities in transcriptomics. *Current Opinion in Systems Biology* **4,** 85–91 (2017) (cit. on p. 136).
217. Pasquini, G., Rojo Arias, J. E., Schäfer, P. & Busskamp, V. Automated methods for cell type annotation on scRNA-seq data. *Computational and Structural Biotechnology Journal* **19,** 961–969. ISSN: 2001-0370. `https://www.sciencedirect.com/science/article/pii/S2001037021000192` (2021) (cit. on p. 136).
218. Trapnell, C. Defining cell types and states with single-cell genomics. *Genome research* **25,** 1491–1498 (2015) (cit. on p. 137).
219. Diaz-Mejia, J. J. *et al.* Evaluation of methods to assign cell type labels to cell clusters from single-cell RNA-sequencing data. *F1000Research* **8** (2019) (cit. on p. 137).
220. Tran, H. T. N. *et al.* A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome biology* **21,** 1–32 (2020) (cit. on p. 137).
221. Ståhl, P. L. *et al.* Visualization and analysis of gene expression in tissue sections by spatial transcriptomics. *Science* **353,** 78–82 (2016) (cit. on p. 138).
222. Lubeck, E., Coskun, A. F., Zhiyentayev, T., Ahmad, M. & Cai, L. Single-cell in situ RNA profiling by sequential hybridization. *Nature methods* **11,** 360 (2014) (cit. on p. 138).
223. Cao, K., Hong, Y. & Wan, L. Manifold alignment for heterogeneous single-cell multi-omics data integration using Pamona. *bioRxiv* (2020) (cit. on p. 138).
224. Maaten, L. v. d. & Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research* **9,** 2579–2605 (2008) (cit. on p. 139).