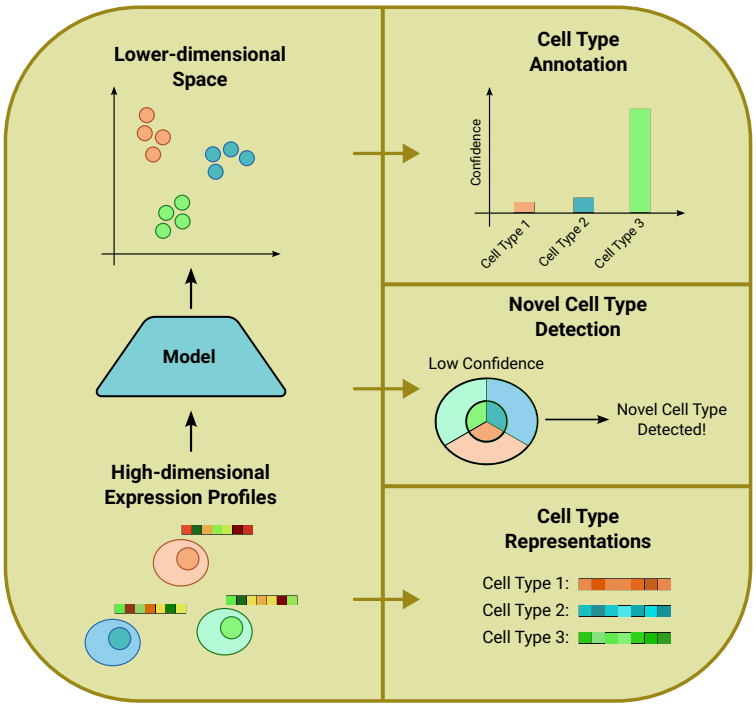
CHALMERS
UNIVERSITY OF TECHNOLOGY



Learning Meaningful Representations of Cells

A study investigating the use of machine learning, single-cell RNA sequencing data, and gene sets to produce a meaningful embedding space of cells

Master's thesis in biotechnology

Leo Andrekson

DEPARTMENT OF LIFE SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Learning Meaningful Representations of Cells

A study investigating the use of machine learning, single-cell RNA sequencing data, and gene sets to produce a meaningful embedding space of cells

Leo Andrekson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Life Sciences
Division of Data Science and AI
AI Laboratory for Biomolecular Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Learning Meaningful Representations of Cells
A study investigating the use of machine learning, single-cell RNA sequencing data, and gene sets to produce a meaningful embedding space of cells
Leo Andrekson

Supervisor: Rocío Mercado, Department of Computer Science and Engineering
Examiner: Johan Bengtsson-Palme, Department of Life Sciences

Master's Thesis 2024
Department of Life Sciences
Division of Data Science and AI
AI Laboratory for Biomolecular Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Should be a description of what the cover image shows.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Learning Meaningful Representations of Cells
A study investigating the use of machine learning, single-cell RNA sequencing data, and gene sets to produce a meaningful embedding space of cells
Leo Andrekson
Department of Life Sciences
Chalmers University of Technology

Abstract

Batch effects are a significant concern in single-cell RNA sequencing (scRNA-Seq) data analysis, where variations in the data can be attributed to factors unrelated to cell types. This can make downstream analysis a challenging task. In this study, a neural network model is designed utilizing contrastive learning and a novel loss function for learning an generalizable embedding space from scRNA-Seq data. When benchmarked against multiple established methods for scRNA-Seq integration, the model outperforms existing methods in learning a generalizable embedding space on multiple datasets. A downstream application that was investigated for the embedding space was cell type annotation. When compared against multiple well-established cell type classifiers, the model in this study displayed a performance competitive with top performing methods across multiple metrics, such as accuracy, balanced accuracy, and F1 score. These findings aim to quantify the "meaningfulness" of the embedding space learned by the model, and highlight the potential applications of these learned cellular representations. The model is currently being structured into an open-source Python package, simplifying and streamlining its usage.

Keywords: scRNA-Seq, deep learning, contrastive learning, bioinformatics, cell type annotation, novel cell type detection, cell type representations, machine learning, AI, transformer

Acknowledgements

First of all I would like to thank my supervisor Rocío Mercado. Over the course of the year, she provided continual guidance, encouragement, and patiently assisted me with various challenges. I would also like to thank all members of the AI Laboratory for Biomolecular Engineering (AIBE) research group for great discussions and feedback on various parts of this project. I am also very grateful for the advice and knowledge shared by my examiner Johan Bengtsson-Palme and also for reading through and giving feedback on my thesis work. Lastly, I would like to express my gratitude to ChatGPT for its assistance in spellchecking and rewriting text to enhance the overall reader experience.

Leo Andrekson, Gothenburg, January 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AE	Autoencoder
BP	Biological Process
CL	Contrastive Learning
CV	Coefficient of Variation
GAN	Generative Adversarial Network
GO	Gene Ontology
GSEA	Gene Set Enrichment Analysis
HVG	Highly Variable Gene
MLP	Multi-layer Perceptron
PBMC	Peripheral Blood Mononuclear Cell
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
sc-omics	Single-cell omics
scRNA-Seq	Single-cell RNA sequencing
UMAP	Uniform Manifold Approximation and Projection
VAE	Variational Autoencoder

Contents

List of Figures

1

Introduction

Cells in our body exhibit remarkable diversity, and characterizing and classifying different cell types is fundamental to understanding their functions and contributions to various biological processes. Learning meaningful representations of cell types enables us to organize and categorize cells based on their shared characteristics, such as their observed phenotypes, but in a lower dimensional space. Due to its far-reaching implications in the biomedical sciences, including disease diagnostics and drug development, learning meaningful cell representations is an active area of research in deep learning, and something which was an active topic of discussion at the recent Learning Meaningful Representations of Life (LMRL) workshop at NeurIPS 2022. These representations enable us to better comprehend cellular similarities, identify disease-specific cell populations, and design targeted therapies conditioned on the relevant cell-type.

To learn meaningful representations of cells, this study aims to leverage single-cell RNA sequencing (scRNA-Seq) data, gene sets, and combinations of the two to train neural networks to produce an efficient, lower-dimensional, generalizable embedding space of cell types. In addition, a secondary objective is to utilize the embedding space for downstream applications, such as cell type annotation and novel cell type detection, thereby validating the quality of the latent space generated.

1.1 Background

In the following section, overviews are provided on relevant topics to the study. These encompass single-cell omics, gene sets, cell type annotation, and scRNA-Seq integration.

1.1.1 Single-cell omics

In the last decade, single-cell omics (sc-omics) has emerged as a rapidly growing field of research for its applications within developmental studies, atlasing (e.g., the Human Cell Atlas), and precision medicine.[1] Popular single-cell methods include transcriptomics, epigenomics, and proteomics, where single-cell transcriptomics is currently the most used sc-omics method. [1, 2] A generalized workflow for single-cell RNA sequencing is illustrated in Figure 1.1.

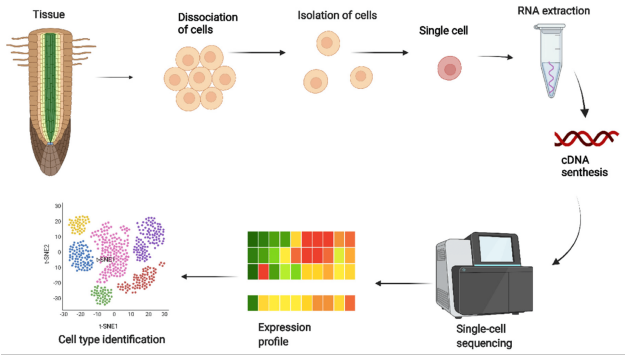


Figure 1.1: Single-cell RNA sequencing generally consists of sampling, isolation of cells, RNA extraction, reverse transcription to cDNA, sequencing, and eventual data analysis. Image from [3].

As various omics techniques become available for single-cell analysis, the field of single-cell multi-omics has gained prominence.[4] These are methods where the goal is to extract and integrate information from multiple omics methods within a single cell-type, such that complex properties or insights about that cell family emerge. Some examples of common integrated multi-omics approaches are combinations of transcriptomics+genomics and transcriptomics+epigenomics.[5] As these technologies improve and their cost is reduced, one might eventually be able to analyze how diseases develop in patients and how they react to different therapies *at single-cell resolution*, giving unprecedented insights into disease progression at a personalized level.[1] But in order for sc-omics data to be used in this way, one first has to investigate different approaches that can reduce noise and dimensionality in these large datasets, motivating the development of models capable of learning a meaningful latent space from scRNA-Seq data.

1.1.2 Gene sets

Being able to use known biological gene set information could be of great use when training a machine learning model on scRNA-Seq data. It has previously been used in the *TOSICA* [6] model developed by Chen *et al.*, where it reportedly outperformed multiple established methods, such as SingleR [7], SciBet [8], and Seurat [9]. Gene sets are collections of genes that have been shown to have similar functions or characteristics. There are many types of gene sets, but in this study a gene set derived from the Gene Ontology (GO), specifically for biological processes (BP), was used. Raw data was gathered from the Gene Set Enrichment Analysis (GSEA) human molecular signatures database. [10, 11]

1.1.3 Cell type annotation

As the acquisition of large-scale omics data for single cells becomes increasingly accessible, the interest for automated cell annotation tools grows. This is attributed to the time-consuming and subjective nature of manual cell annotation. Recently developed strategies for cell type annotation from scRNA-Seq data typically rely on either referencing marker gene databases, correlating reference expression data, or

supervised classification methods. [12] Figure 1.2 shows an overview of automatic approaches for cell type annotation.

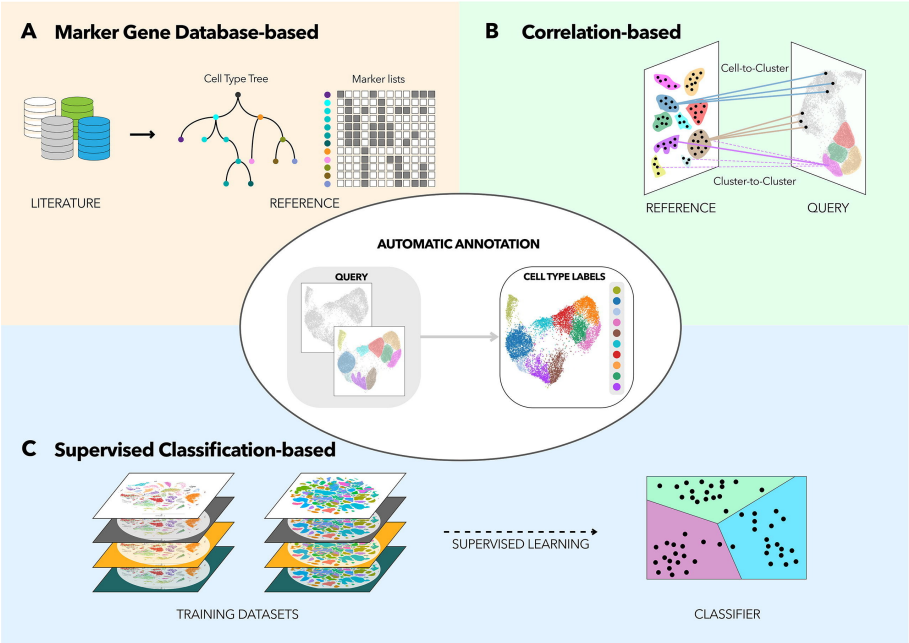


Figure 1.2: Automatic approaches for cell type annotation. **A** Marker gene database-based annotation leverages cell type atlases. These atlases compile markers derived from literature reviews and scRNA-Seq analyses into reference hierarchies and lists. **B** Correlation-based methodologies utilize various correlation measures to assess gene expression profiles between a reference dataset and a query dataset, operating at either the single-cell or cluster level. **C** Supervised annotation employs machine learning techniques to train a classifier using labeled reference scRNA-seq datasets. This classifier is then utilized to annotate the query dataset. Image from [12].

In this study, a novel method for supervised cell type annotation was developed. The model works by initially learning an embedding space from the scRNA-Seq data through contrastive learning, and subsequently tuning this space for cell type classification.

1.1.4 scRNA-Seq data integration

ScRNA-Seq datasets often encompass numerous samples, leading to the inevitable introduction of batch effects arising from various sources (e.g., diverse patients, sequencing methodology). Here, batch effects denote sources of variation within the data attributed to non-biological factors. To emphasize the challenges imposed by batch effects, we visualize some example data in Figure 1.3. Ideally, a model would integrate scRNA-Seq data into an embedding space where biological variation is maximized while minimizing variation caused by batch effects. Many methods have been developed for this purpose, such as *scANVI* [13] and *scGen* [14], and were recently benchmarked in a study by Luecken *et al.* [15] While these methods have

shown great promise in scRNA-Seq data integration, they have not been evaluated for learning a generalizable embedding space. The model developed in this study was compared to previous scRNA-Seq integration methods on generalizability to unseen data using metrics from the aforementioned benchmark. The data for testing contains the same cell types as were present during training since it is being evaluate on each test fold. Where each test fold is created by randomly splitting all data, while maintaining the same proportion of cell types. This gives an image of how the model might be able to generalize to new patient data if given enough data so to capture the space better.

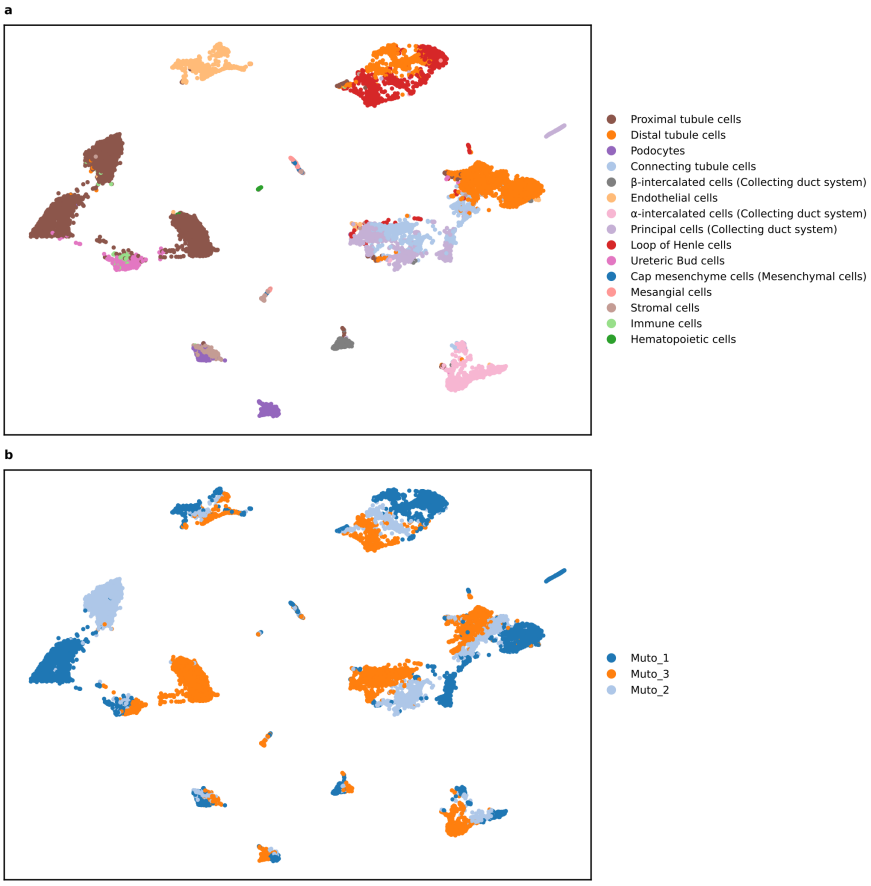


Figure 1.3: **a** UMAP visualization of cell type clusters in the kidney dataset constructed in this study. **b** UMAP highlighting batch effect cause by patient ID.

Here, it can be seen how variation caused by data coming from different patients (Fig. 1.3 b) forms separate clusters with data belonging to the same cell type (Fig. 1.3 a). This is not desirable since cells of the same type are expected to cluster together. Ideally, the data would be completely mixed in terms of patient ID, the "batch effect" in this case (Fig. 1.3 b). This implies that no variation within the dataset should be attributed to batch effects, such as patient or laboratory, from which a set of data originated. Attaining complete mitigation of batch effects poses a significant challenge, prompting the development of numerous new approaches to scRNA-seq data integration.

1.2 Previous work

1.2.1 scRNA-Seq integration

Here we list relevant previous work for scRNA-seq integration, to which we compared the models developed in this study:

- *TOSICA* [6]
- *scGen* [14]
- *scANVI* [13]
- *scVI* [16]

The generalizable scRNA-Seq integration benchmark covers approaches utilizing transformers (*TOSICA*), variational autoencoders (*scGen*), and hierarchical Bayesian models with conditional distributions parametrized by neural networks (*scVI* and *scANVI*). Contrastive learning approaches are covered by models developed in this thesis. By including multiple modeling approaches we ensure that the benchmark covers a wide range of previous research in this area, facilitating a fair and comprehensive comparison. Many other approaches exist, such as *Scanorama* [17] and *Harmony* [18], but most of these other models are not able to make predictions on new data. Many models that have been developed are designed to integrate the data you have, instead of being trained on one set of data and later used to integrate new data. The models used in the benchmark are selected for their ability to make predictions on new data, making them suitable for a generalizability benchmark.

1.2.2 Cell type annotation

Here we list relevant previous work for cell type annoation based on scRNA-seq data, to which we compared the models developed in this study:

- *TOSICA* [6]
- *scNym* [19]
- *Seurat* [9]
- *SciBet* [8]
- *CellID* [20]

These 5 models cover a wide range of approaches for solving cell type annotation tasks. In the annotation benchmark, we incorporate not only the inclusion of all genes in models that can handle this, but also the filtering of HVGs for all models, regardless of whether they inherently support this feature. This is done to make a more fair comparison, as the models developed in this study always utilizes HVG filtering.

1.3 Aim

The aims of this study are five-fold:

1. To train a semi-supervised neural network model on scRNA-Seq data with the objective of clustering cell types in latent space while minimizing batch effects.
2. To train a multi-modal model by combining the scRNA-Seq data and gene sets information, thus perhaps leading to an even better cell-type embedding space.
3. To compare the performance of developed models to other approaches in a benchmark using metrics developed by Luecken *et al.* [15].
4. To investigate if the developed model also can be used as a cell type annotator, and compare its performance with other established methods.
5. To investigate novel cell type detection as a downstream application of the generated embedding space.

2
Methods

Approaches used in this study for data gathering (2.1), preprocessing (2.2), embedding space model development (2.3), embedding space evaluation (2.4), producing cell type representation vectors (2.5), cell type annotation (2.6), and novel cell type detection (2.7) are summarized below.

2.1 Data gathering

Three types of data were gathered for this study: scRNA-Seq data, gene sets, and *gene2vec* representations.

2.1.1 scRNA-Seq data

Data collected from human participants were gathered from the following studies and tissues:

- Bone marrow: GSM3396161, GSM3396176, GSM3396184, GSM3396183 and GSM3396174. [21] Data available for 20 cell types and 5 patients.
- PBMC: GSE115189, GSM3665016, GSM3665017, GSM3665018, GSM3665019 and PBMC 10X data. [22, 23] Data available for 17 cell types and 6 patients.
- Pancreas: GSE81076, GSE85241, GSE86469 and E-MTAB-5061, gathered into one dataset by the Satija lab. Data available for 9 cell types and 4 patients/sequencing methods.
- Kidney: GSM4572192, GSM4572193 and GSM4572194. [24] Data available for 15 cell types and 3 patients.
- Merged: Bone marrow, PBMC, pancreas and kidney datasets all merged into one dataset consisting of 46 cell types and 18 patients.

These datasets were used to assess the model's ability to create a generalizable embedding space from scRNA-Seq data. For exploring the model's cell type annotation capability, the following datasets were used:

- Baron: GSE84133. [25] Data available for 14 cell types and 4 patients.
- MacParland: GSE115469. [26] Data available for 20 cell types and 5 patients.

- Segerstolpe: E-MTAB-5061. [27] Data available for 15 cell types and 10 patients.
- Zheng68k: Data from the 'Fresh 68K PBMCs' section at 10xGenomics. [28] Data available for 11 cell types and 8 unique barcode identifiers.

Ones all of the above datasets had been preprocessed they contained this number of cells:

- Bone marrow dataset: 14,779
- PBMC dataset: 22,027
- Kidney dataset: 16,370
- Pancreas dataset: 6,313
- Megred dataset: 59,489
- Baron: 8,569
- MacParland: 8,444
- Segerstolpe: 3,514
- Zheng68k: 68,579

2.1.2 Gene sets

The following gene sets were collected from GSEA:

- Canonical pathways gene sets. File name: *c2.cp.reactome.v2023.1.Hs.symbols.gmt*
- Gene sets that depict possible targets subject to regulation by transcription factors or microRNAs. File name: *c3.all.v2023.1.Hs.symbols.gmt*
- Gene sets derived from GO BP. File name: *c5.go.bp.v2023.1.Hs.symbols.gmt*
- Gene sets capturing cellular states and alterations in the immune system. File name: *c7.all.v2023.1.Hs.symbols.gmt*
- Gene sets containing markers for cell types based on single-cell sequencing studies of human tissue. File name: *c8.all.v2023.2.Hs.symbols.gmt*

The user can then select which individual gene sets are of interest to their study and use those to retrain the model. In this study, the gene set for GO BP was used.

2.1.3 *Gene2vec*

In an attempt to enhance the model's ability to attend to different genes, a set of vector representations of size 200 is used for each gene. *Gene2vec* is a method that can generate gene representations based on co-expression patterns of human

genes. [29] These representations are used in this study to provide the model with additional information.

2.2 Preprocessing

2.2.1 scRNA-Seq data

ScRNA-Seq data was preprocessed as follows:

1. Cells filtered by the number of unique genes a cell must reach for the cell to be considered for downstream processing. This was achieved by calculating the shifted logarithm

$$x = log(x + 1) \tag{2.1}$$

of the number of genes (x) and filtering out cells by setting a threshold on how many median absolute deviations (MAD)

$$MAD = median(|x - median(x)|) \tag{2.2}$$

from the median a sample was allowed to be. In this study the threshold was set to 5 MAD.

2. Cells filtered by the sequencing depth to ensure high quality samples. Similarly to the previous point, the shifted logarithm was calculated for the read counts and filtering was performed using a MAD value of 5.

3. Cells filtered by the proportion of mitochondrial counts. This was done to remove cells suspected of having broken membranes, which could potentially be dying cells. [30] As before, filtering was performed using MAD. The threshold was set to 5 MAD for almost all filtering, but was occasionally changed to a more appropriate value based on a visual assessment of the distribution. The MAD value was selected from a range between 5 and 10 based on the visual assessment.

4. Cells filtered by the proportion of total counts corresponding to the top 20 genes with the highest number of counts per cell. As before, filtering was performed using MAD. The threshold was set to 5 MAD for almost all filtering, but was occasionally changed to a more appropriate value based on a visual assessment of the distribution. The MAD value was selected from a range between 3 and 5 based on the visual assessment.

5. Genes filtered by number of unique cells expressing a gene that must be reached in order for a gene to be considered for downstream processing. In this study, genes that appeared in less than 20 cells were discarded.

6. Normalization was accomplished by first normalizing the total number of counts so it becomes comparable between cells. This was achieved by first

calculating the average number of counts per cell in the data

$$L = \frac{1}{N_c} \cdot \sum_{g,c} y_{g,c} \tag{2.3}$$

where N_c is the number of cells and y is the counts matrix where g annotates the position of a gene and c the position of a cell, together specifying positions in y. Then the sum of counts of each cell was divided by L

$$S_c = \frac{1}{L} \cdot \sum_{g} y_{g,c} \tag{2.4}$$

Next it is desired to transform the data so it resembles a normal distribution as much as possible. A shifted logarithm transform was shown to perform very well in a recent benchmark and was therefore used in this study. [31] Normalization was performed by calculating

$$y_{norm_{g,c}} = log(\frac{y_{g,c}}{S_c} + 1) \tag{2.5}$$

where y_{norm} is the normalized counts matrix.

7. Cell type annotation was performed using the *ScType* package. [32]

8. As a final step, one can pick the top N most highly-variable genes (HVGs) in the processed data that will be used as input for the model to be trained. This can be achieved by utilizing the *highly_variable_genes* function from *Scanpy* using the *cell_ranger* flavor. [33] This method was also used in the Luecken *et al.* [15] benchmark to identify HVGs. *Scanpy* is an established tool for preprocessing single-cell omics data and is therefore used in this study. N was chosen to be 2000.

The entire preprocessing pipeline described above is also illustrated in Figure 2.1.

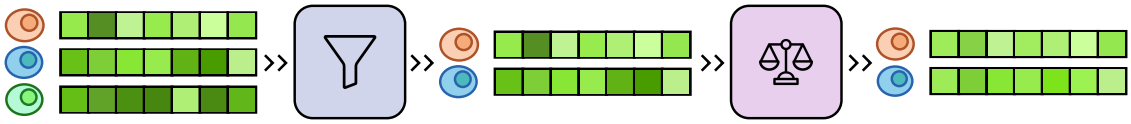


Figure 2.1: Summary of the entire preprocessing pipeline, starting with scRNA-Seq data from multiple samples. The data is then filtered to ensure sample quality and normalized to make samples comparable.

It is worth noting that for Baron, Segerstolpe, and Zheng68k, only step 5, 6 and 8 were implemented since we assume that the quality of these samples are very high due to the vast popularity of using these datasets for benchmarking cell type annotators. For the MacParland dataset, only step 5 and 8 were implemented since it comes pre-normalized. All four of these datasets have already been annotated, such that step 7 was not required.

2.2.2 Gene sets

Each gene set consists of a list of gene symbols. To make this information useful for a machine learning model, a binary mask matrix is created. For each position corresponding to an HVG that also exists in the given gene set, the value is set to 1; for all other positions, it is set to 0. Each row corresponds to a gene set, and each column represents a gene. This way, one can apply each gene set mask to a vector of expression levels for a given sample and then feed it as input to a network.

Relevant gene sets are selected by calculating the percentage of genes in a gene set that are considered to be HVGs

$$Q_i = \frac{\sum^N GeneSetsHVG_{Mask_i}}{\sum^N GeneSets_{Mask_i}} \tag{2.6}$$

where N is the number of genes, $GeneSetsHVG_{Mask_i}$ is the binary mask of gene set i, where only HVGs are assigned a value of 1 while all other values are 0. $GeneSets_{Mask_i}$ is the binary gene sets mask where all genes that belong to gene set i is given a value of 1. Q_i is the percentage of HVGs in gene set i. In order to avoid picking gene sets with a very few number of HVGs, a limit is introduced, stating that there must be more than 10 HVGs in a gene set for it to be considered. To ensure that gene sets remain valuable for discerning biological insights, it's essential to prevent them from becoming too extensive. Consequently, a limit was introduced stipulating that a gene set must contain fewer than 300 genes. In this study, the top 500 gene sets with the highest values on the Q metric were selected, while also being below 300 genes.

2.3 Development of the cellular embedding model

Several model architectures were developed in this study and investigated in their capacity to produce a generalizable cell type embedding space. Some utilize only scRNA-Seq information, while others incorporate gene set information in combination with scRNA-Seq data. Further details about the models and the chosen learning strategy are discussed below.

2.3.1 Contrastive learning

Models were trained using contrastive learning (CL). This is a technique where the model tries to produce outputs that are close to each other in space for augmented inputs of the same original input, called positive (+) input pairs, while being pushed away from all other inputs, called negative (-) input pairs. [34] A few approaches to CL are shown in Figure 2.2.

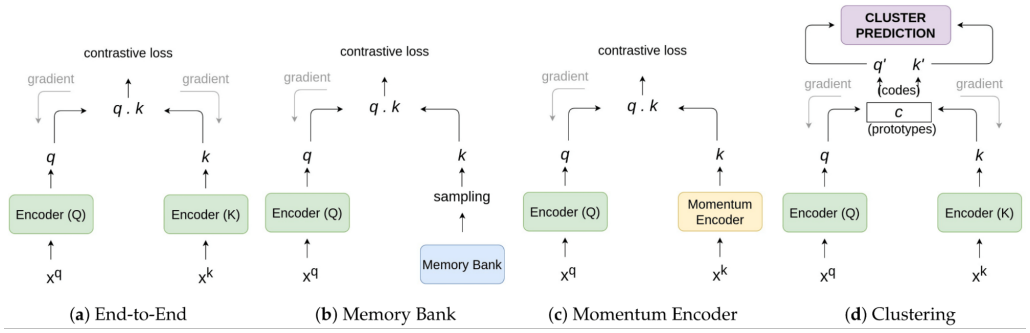


Figure 2.2: Four approaches to contrastive learning. (a) A query encoder is trained on original inputs while a key encoder is trained on augmented inputs.[35] (b) Keys are instead stored and retrieved from a memory bank [36]. (c) Keys are instead sampled from a dynamic dictionary in the form of a momentum encoder [37]. (d) Like (a) but with a clustering mechanism applied at the end [38]. Image from [34].

To apply CL, one needs an applicable loss function. *Normalized temperature-scaled cross entropy loss* (NT-Xent; Equation 2.7) is a loss function that is suitable for CL when there are multiple negative inputs. [35, 39]

$$\mathcal{L}_{NT-Xent} = -log(\frac{exp(sim(q, k_+)/\tau)}{exp(sim(q, k_+)/\tau) + \sum_{i=0}^{N} exp(sim(q, k_{-,i})/\tau)}), \tag{2.7}$$

where $sim()$ is the cosine similarity function

$$sim(A, B) = \frac{A \cdot B}{||A||||B||} \tag{2.8}$$

τ represents the temperature parameter, N is the number of negative samples, q is the original input, k are augmented inputs where all $k_{-,i}$ are negative inputs such that $q \neq k_i$, "+" indicates the augmented version of the original input, and "-" indicates all other inputs.

This learning strategy can be modified by instead stating that inputs belonging to the same label should strive to be as close to each other as possible while being far away from inputs of other labels, resulting in a semi-supervised approach. This is accomplished by modifying Equation 2.7 to be

$$\mathcal{L}_{CellType} = -\frac{1}{B}\sum_{b=0}^{B} log(\frac{\sum_{j=0}^{M} \alpha_b(k_j)}{\sum_{j=0}^{M} \alpha_b(k_j) + \sum_{i=0}^{N} \alpha_b(k_i)}) \tag{2.9}$$

where α_b is defined as

$$\alpha_b(x) = exp(sim(q_b, x)/\tau) \tag{2.10}$$

and q_b is the current reference sample in your dataset, specified by number b in the batch, consisting of B samples. k_j are inputs of the same cell type as sample q_b while $q_b \neq k_j$ and $q_b \neq k_i$. k_i are inputs of all other cell type entries. M is the number of

samples with the same label as q_b and N is the number of remaining samples. The loss calculation is performed for each sample q_b, meaning for all data points in the batch, such that $\mathcal{L}_{CellType}$ is an expectation value. This loss function is called the *soft nearest neighbor loss*. [40]

Using the loss described in Equation 2.9, one can enforce the creation of cell type clusters in latent space, but this loss alone does not necessarily reduce batch effects. To do so, we define an additional loss function

$$\mathcal{L}_{BatchEffect} = -\frac{1}{B}\sum_{b=0}^{B}(log(\frac{\sum_{j=0,y_j=y_b}^{M}\alpha_b(k_j)}{\sum_{j=0,y_j=y_b}^{M}\alpha_b(k_j)+\sum_{i=0,y_i=y_b}^{N}\alpha_b(k_i)}))^{-1} \tag{2.11}$$

where k_j are inputs of the same batch effect key (e.g., laboratory) as sample q_b while $q_b \neq k_j$ and $q_b \neq k_i$. k_i are inputs of all other batches. y_b is the cell type of q_b, y_j is the cell type of k_j, and y_i is the cell type of k_i, where $y_j = y_b$ and $y_i = y_b$ must be fulfilled. Using this loss we promote the dispersion of data points of the same batch within each cell type cluster.

ScRNA-Seq data tends to be unbalanced in terms of cell types and batch effects. Hence, it can be advantageous to weight the contributions of each label type to the loss differently. Weights can be calculated as

$$\mathcal{W}_i = \frac{\frac{1}{N_i}}{\sum_j^M \frac{1}{N_j}} \tag{2.12}$$

where $\mathcal{W}_i$ is the weight to be applied to the loss of cell type label i, N_i is the number of samples of label i and M is the total number of unique cell types present in the data. By calculating the weights in this way, we promote less common cell types to have a larger impact during training, which counters the label unbalance. Weights are also calculated and used for batch effects in the same manner by redefining N_i to be the number of samples of batch effect i and M is the number of unique batch effect keys.

Combining Equations 2.9 and 2.12, we arrive at the final cell type loss

$$\mathcal{L}_{CellType} = \sum_{i}^{M} \mathcal{W}_{CellType,i} \cdot \mathcal{L}_{CellType,i} \tag{2.13}$$

where i is a specified cell type and M is the number of unique cell types in the dataset.

Similarly, for batch effects we get

$$\mathcal{L}_{BatchEffect} = \sum_{i}^{M} \mathcal{W}_{BatchEffect,i} \cdot \mathcal{L}_{BatchEffect,i} \tag{2.14}$$

where i is a specific type of batch effect and M is the number of unique batch effect keys.

Next, the two loss terms (Equations 2.13 and 2.14) are combined to arrive at the final loss:

$$\mathcal{L}_{CL} = 0.9 \cdot \mathcal{L}_{CellType} + 0.1 \cdot \mathcal{L}_{BatchEffect} \tag{2.15}$$

Here, the loss components are added together and weighted with 90% importance on the cell type loss and 10% on the batch effect loss. Having a small contribution from the batch effect loss is necessary so that the noise promotion does not take over, but it cannot be completely neglected. It is worth noting, however, that these percentages can be adjusted to individual research needs.

2.3.2 Cell type centroid loss

Next, it is of interest to enforce a meaningful relativistic positioning of cell type clusters in the model's output latent space. To achieve this clustering behavior, an additional term, the centroid loss, is included in the loss.

First, the scRNA-Seq expression levels are transformed using principle component analysis (PCA) to a latent space of the same dimension as the intended output of the model; in this study, this dimension is set to 100. PCA is used for its ability to extract dimensions responsible for significant variation in data, providing a lower-dimensional representation with reduced noise. From this latent space, the goal is to extract a distance matrix containing the Euclidean distance between each cell type cluster's centroid (Figure 2.3).

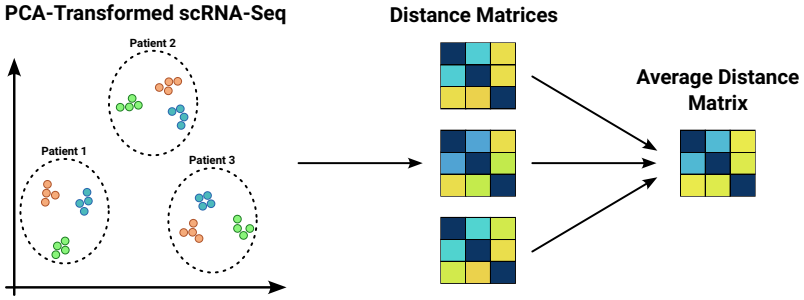


Figure 2.3: PCA is used to reduce the dimensionality of the scRNA-Seq data. The Euclidean distance matrix is calculated between cell type clusters in different *batches* (i.e., patients). From this, the average distance matrix is calculated.

Assuming the main source of variation is due to the specified batch effect (patient ID in Figure 2.3), one can expect the distances between cell type clusters within each individual *batch* of data to be similar. This assumption is further investigated in Section 3.2. Calculating the average distance matrix across patients gives an approximation of the underlying relativistic positioning of cell types in the latent space. To obtain the relativistic distance matrix, one simply divides the average centroid distance matrix by the maximum value of this matrix. This is performed both on the PCA transformed space and then later on the model produced space. Ideally, the cell type clusters in the latent output space of the model should reflect this underlying relativistic positioning, which can be enforced by introducing the aforementioned centroid loss term, defined as

$$\mathcal{L}_{Centroid} = MSE(D_{Model}, D_{PCA}) \tag{2.16}$$

where MSE stands for mean squared error and D_{Model} is the relativistic centroid distance matrix produced by the model and D_{PCA} is the same matrix but generated through dimensionality reduction of the scRNA-Seq via PCA.
The workflow for calculating the cell type centroid loss is described in Algorithm 1 below (note that the algorithm is split over two pages).

Algorithm 1 Cell Type Centroid Loss Algorithm

// Step 1: Reference Distance Matrix
Input: *adata // Annotated data object*
Define: $X = adata.X$ *// scRNA-Seq data*
Define: $X_{PCA} = PCA(X, dim = 100)$
Define: $A =$ "$cell\_type$"; $B =$ "$patientID$"
Define: $unique_A = unique(adata.obs[A])$
Define: $unique_B = unique(adata.obs[B])$
Define: $centroids = \{\}$
for β in $unique_B$ **do**
  for α in $unique_A$ **do**
    Define mask: $mask = (adata.obs[B] == \beta)$ & $(adata.obs[A] == \alpha)$
    Apply mask and calculate centroid:
    $centroids[(\beta, \alpha)] = mean(X_{PCA}[mask], axis = 0)$
  end for
end for
Initialize distance matrix:
$D_{ref} = zeros(len(unique_A), len(unique_A))$
for α_1 in $unique_A$ **do**
  for α_2 in $unique_A$ **do**
    $\delta = []$
    for β in $unique_B$ **do**
      $\delta_\beta = EuclideanDistance(centroids[(\beta, \alpha_1)], centroids[(\beta, \alpha_2)])$
      $\delta.append(\delta_\beta)$
    end for
    Compute mean Euclidean distance across batch effect elements:
    $D_{ref}[\alpha_1, \alpha_2] = mean(\delta)$
  end for
end for
Normalize: $D_{ref} = \frac{D_{ref}}{max(D_{ref})}$

Continued from previous page:

// Step 2: Calculate Loss For Mini-Batch During Training
Input mini-batch: $adata$
Define: $X = adata.X$ *// Model-generated embedding space from scRNA-Seq data*
Define: $A =$ "$cell\_type$"
Define: $B =$ "$patientID$"
Define: $unique_A = unique(adata.obs[A])$
Define: $unique_B = unique(adata.obs[B])$
Define: $centroids = \{\}$
for α in $unique_A$ **do**
    Define mask: $mask = adata.obs[A] == \alpha$
    Apply mask and calculate centroid: $centroids[\alpha] = mean(X[mask], axis = 0)$
end for
Initialize distance matrix: $D = zeros(len(unique_A), len(unique_A))$
for α_1 in $unique_A$ **do**
    for α_2 in $unique_A$ **do**
        Euclidean distance between cell type clusters: $\delta = EuclideanDistance(centroids[\alpha_1], centroids[\alpha_2])$
        $D[\alpha_1, \alpha_2] = \delta$
    end for
end for
Normalize: $D = \frac{D}{max(D)}$
Extract relevant part from reference distance matrix: $D_{relevant\_ref} = D_{ref}[unique_A, unique_A]$
Normalize: $D_{relevant\_ref} = \frac{D_{relevant\_ref}}{max(D_{relevant\_ref})}$
$\mathcal{L} = MSE(D, D_{relevant\_ref})$
return $\mathcal{L}$

It is worth noting that since not all cell types exist in each patient's data, some cell type distances may not be possible to calculate. Therefore, these positions in the distance matrix are masked during the calculation of the loss in Equation 2.16. The idea is that the existing relativistic distances from the reference data will be maintained in the latent space even after training the model.

By combining the contrastive learning loss (Equation 2.15) and the cell type centroid loss (Equation 2.16), we arrive at

$$\mathcal{L} = \mathcal{L}_{CL} + \mathcal{L}_{Centroid} \tag{2.17}$$

This loss function was utilized for training all ML models developed in this work, with the goal of learning a meaningful embedding space for cell types.

2.3.3 Transformer encoder mechanism

Some of the developed models utilize the transformer encoder architecture with the idea being that the self-attention mechanism can help the model learn how HVGs and gene sets are related to individual contributions within itself and between the two. First, the input is linearly projected into three matrices – the query (Q), key (K), and value (V) matrices:

$$Q, K, V = \mathcal{W}_{q,k,v} \cdot X \tag{2.18}$$

where X is the input and $\mathcal{W}_{q,k,v}$ are the weights for the linear projection.

Using Q, K and V the attention matrix (A) is computed

$$A = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}}) \tag{2.19}$$

where d_k is the hidden dimension of Q and K.Output from the attention mechanism is then calculated by simply taking the dot product of A and V.

$$Attention(Q, K, V) = A \cdot V \tag{2.20}$$

It is common to project Q, K, and V multiple times in a process called *multi-head attention*, refered to as MHA in the expressions below:

$$MHA(Q, K, V) = \mathcal{W}^O \cdot concat(head_1, ..., head_M) \tag{2.21}$$

where M is the number of attention heads, $\mathcal{W}^O$ are the weights for the linear projection and

$$head_i = Attention(\mathcal{W}_i^q \cdot Q, \quad \mathcal{W}_i^k \cdot K, \quad \mathcal{W}_i^v \cdot V) \tag{2.22}$$

Output from multi-head attention is then added to the input X, typically followed by layer normalization.[41]

$$O = LayerNorm(MHA + X) \tag{2.23}$$

Output from Equation 2.23 is then fed to a two/layer multi/layer perceptron (MLP) where the hidden layer is four times the size of the input, and the output has the same size as the input. Output from the MLP is added to the original input X and layer normalization is applied once again, as follows:

$$O = LayerNorm(MLP(O) + X) \tag{2.24}$$

where $MLP(O)$ is the output from feeding the two-layer MLP the output O from Equation 2.23. Output from Equation 2.24 can then be fed back to Equation 2.18 (setting $X = O$) and the entire process can be repeated. This creates a transformer encoder with a so called *depth* specified by the number of iterations through the pipeline.

2.3.4 Models

Many methods have been developed to perform dimensionality reduction, and can be divided into three main groups: statistical methods, manifold methods, and neural networks [42]. This study focuses on developing a neural network architecture to produce meaningful cell type representations while (1) minimizing batch effects and (2) promoting biological variation. Previously developed neural network algorithms for batch correction have used methods such as autoencoders (AEs) [43, 44], variational autoencoders (VAEs) [14], generative adversarial networks (GANs) [45], AE combined with recurrent neural networks (RNNs) [46], contrastive learning [47, 48], and transformers [6].

The models developed in this study are novel due to the combination of a re-engineered contrastive loss with the cell type centroid loss function developed in this study. The hypothesis is that contrastive learning makes the model cluster scRNA-Seq data of the same cell type together while the centroid loss enforces a more meaningful positioning of cell type clusters.

Model 1 was trained on an NVIDIA A100 GPU while Model 2 and Model 3 were trained using four NVIDIA A100 GPUs. Models were trained using mini batches of size 256. Each mini batch was created by randomly dividing the dataset into N mini batches of the specified size. Mini batches were randomly created each epoch.

2.3.4.1 Simple encoder (Model 1)

Attention mechanisms are known for being computationally expensive to train, particularly when dealing with a large number of tokens, such as 2000 HVGs in this case. Therefore, it is of interest to explore simpler model architectures that are more cost-effective to train. First, a feed-forward encoder was trained on the scRNA-Seq data. The Model 1 architecture is visualized in Figure 2.4.

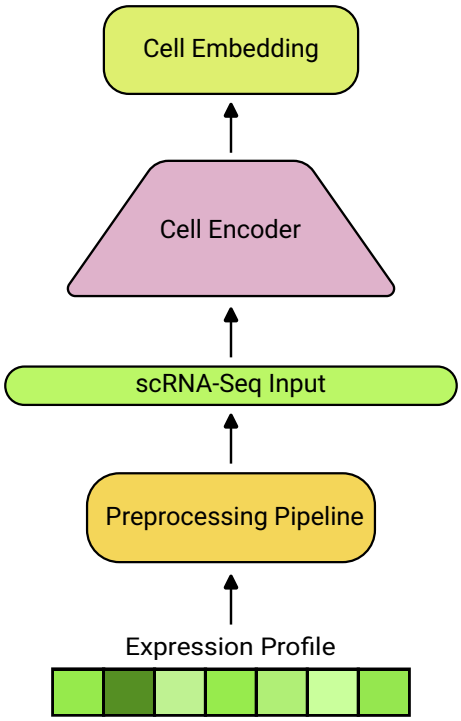


Figure 2.4: Overview of Model 1. ScRNA-Seq data is fed through a feed-forward encoder that returns a learned embedding for each sample (cell). *Expression Profile* indicates raw scRNA-seq data.

2.3.4.2 Gene2vec transformer (Model 2)

Tokenizing genes into buckets is a concept that can be directly applied to the scRNA-Seq data, making it suitable as input to a transformer encoder. This concept was explored in Model 2 by tokenizing the scRNA-Seq data into 1000 buckets equally spaced between the minimum and maximum value of each gene. In addition, gene2vec representations are added to the embedding of the buckets with the idea being the embeddings would contain additional information about how genes are related. A so-called learnable *class token* was also introduced (for details, see Section 2.3.5) and concatenated to the input; this approach was suggested by Chen *et al.* [6] and implemented in their model *TOSICA* as it allows the model learn attention weights between the sample itself and the individual genes. The Model 2 architecture is illustrated in Figure 2.5.

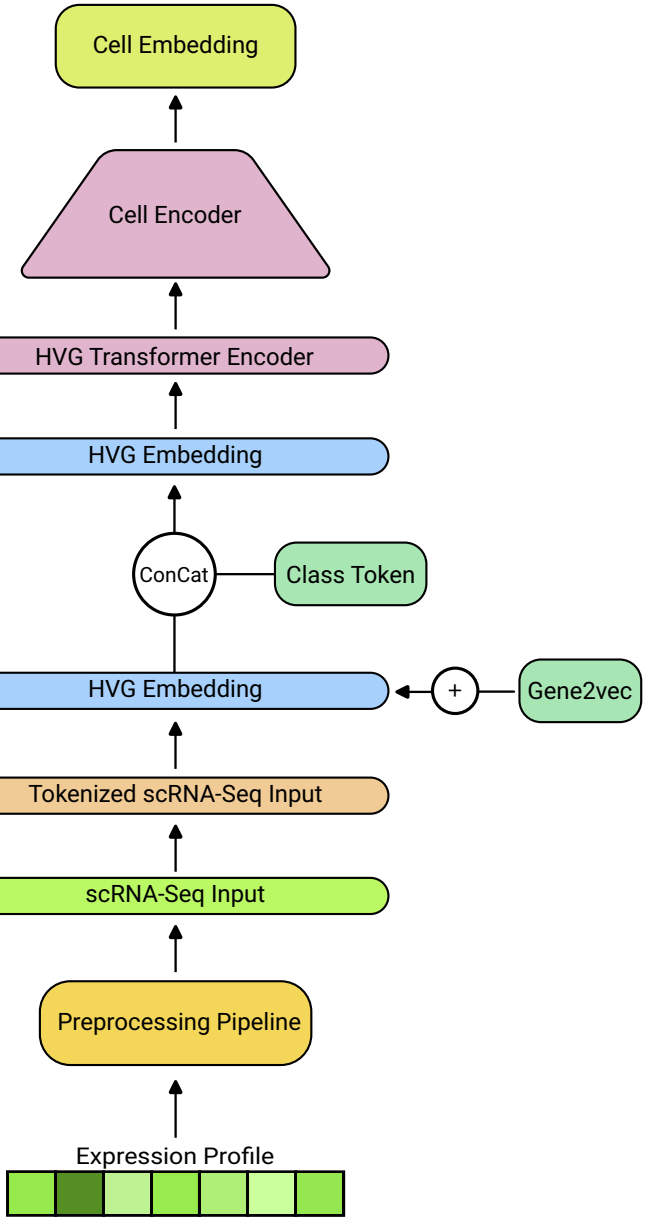


Figure 2.5: Overview of Model 2. ScRNA-Seq data is tokenized, gene2vec representations are added to the resulting embeddings, and a class token is concatenated to the input. These are then fed to a transformer encoder, producing a vector representation of attention between the input sample and each gene. Next, in order to convert the representation to a desirable dimension, it is fed to a feed-forward encoder to produce a learned embedding for each sample (cell).

2.3.4.3 Embedded gene set transformer (Model 3)

Other than scRNA-Seq data, one can imagine that selecting relevant gene sets can contribute additional information about cell types to a machine learning model. This model investigates this by taking scRNA-Seq data as input and applying a binary gene set mask on top of it. The binary gene set mask is created based on the Gene Ontology Biological Process (GO BP) gene set from Gene Set Enrichment Analysis (GSEA). Next, a simple linear transform is applied to create an embedding space and a *class token* is introduced by concatenating a trainable token to the input. This creates the input representation for the transformer encoder. Finally, the outputs from the gene sets transformer encoder and the HVGs transformer encoder are fed into a feed-forward encoder, which returns a learned representation for each sampled (cell). The Model 3 architecture is illustrated in Figure 2.6.

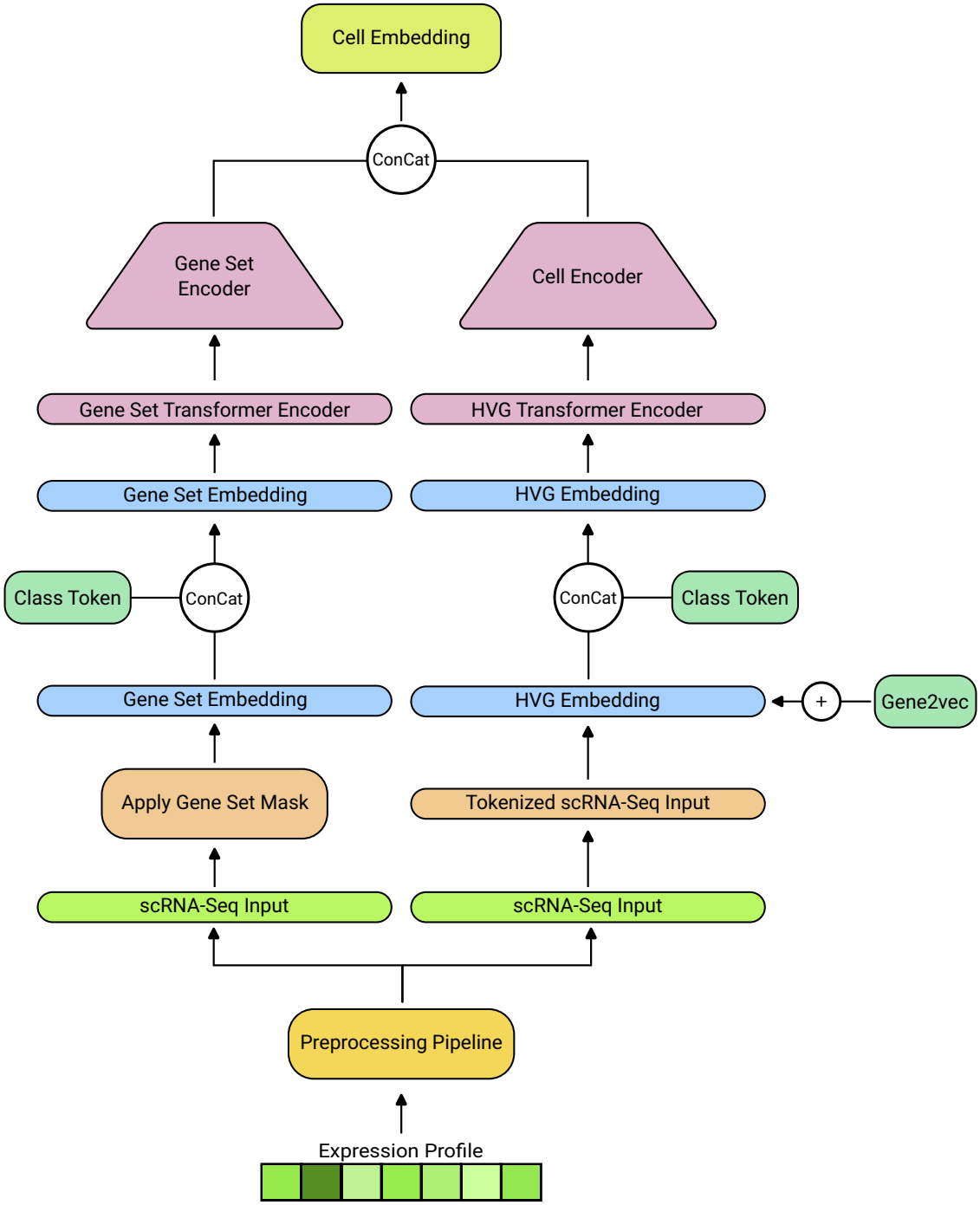


Figure 2.6: Overview of Model 3. ScRNA-Seq data undergoes multiplication with a binary gene set mask and is then mapped with weights to generate an embedding space. A class token is concatenated to the embedded data that is then input to a transformer encoder. Simultaneously, the scRNA-Seq data is tokenized, and an embedding space is generated. Gene2vec representations are added to these representations, and a class token is concatenated to the input before being fed into a separate transformer encoder. The output from these transformer encoders are then passed to one feed-forward encoder each and the resulting outputs are concatenated and finally linearly mapped to a desirable output dimension. This process results in the creation of a latent space.

2.3.5 The role of attention in an interpretable transformer

One way of interpreting how a transformer model attends to input information is to look at the attention weights. However, since the cell type is not an input, it is not straightforward how to relate the attention applied to inputs, such as HVGs, to the cell type. One approach suggested by Chen *et al.* [6] and implemented in their model *TOSICA* is the introduction of a so-called *class token*. The token is concatenated to the input before being fed to the transformer model. It has the same embedding size as the input tokens, but is defined in such a way that the token itself is updated through the training process. This molds the token's embedding values, indirectly creating a token whose attention weights correspond to the model's attention to input data in relation to the current sample (a specific cell). By extracting the attention value to the token for each sample and grouping them together by cell type, we end up with distributions of attention weights corresponding to each cell type. This information can then be used to identify inputs that the model pays significant attention to when clustering a given cell type, making the model interpretable.

It is typical to have transformers with more than one head and a depth greater than one. This introduces another challenge when interpreting attention weights, as it results in multiple attention weight matrices. One approach, as suggested by Chen *et al.* [6], was employed in this study:

1. Start by calculating the mean attention weight matrices across all attention heads.

2. Account for residual connections by adding a identity matrix to the attention weight matrices, and re-normalize by dividing by the new row sum.

3. Recursively multiply all attention weight matrices together.

4. Extract the row corresponding to the attention weights of the class token, excluding the class token itself. This is the final attention latent space representation used to interpret the attention of the transformer.

Using this approach, one can both interpret how Model 3 attends to both individual HVGs and gene sets, giving the user biological insight into what differentiate cells. This application is discussed in Section 4.3.2.

2.4 Embedding space evaluation

To evaluate the scRNA-Seq integration performance of the models in this study, they are compared to each other and to other established methods in a benchmark by Luecken *et al.* [15].

2.4.1 Benchmark

Performance of the models developed herein is evaluated in terms of (1) batch correction and (2) preservation of biological variation. This is compared to the performance of models developed in previous work using the benchmark of Luecken *et al.* [15]. Metrics such as *Normalized Mutual Information* (NMI), *Adjusted Rand Index* (ARI), *Average Silhouette Width* (ASW), *Isolated label Silhouette*, *Principal Component Regression* (PCR), *Graph Connectivity* (GC), *Isolated label* F_1 *score*, and *Cell-cycle Conservation* (CC), are used to evaluate batch correction and the conservation of biological information in the benchmarked models. All metrics are designed to fall within the value range 0 to 1, with 1 indicating better performance. Implementations of all metrics exist in the *scIB* package developed by Luecken *et al.* [15] and were used in this study.

2.4.1.1 *Normalized Mutual Information*

NMI is a metric that assesses the overlap between two clustering schemes. In this study, NMI is utilized to evaluate the alignment of cell-type labels with Louvain clusters derived from the integrated dataset.

2.4.1.2 *Adjusted Rand Index*

ARI provides a measure of the agreement between two clusterings, taking into account both correct clustering overlaps and correct disagreements between the two clusterings while adjusting to correct for randomly occurring correct labels. Cell-type labels are compared with the Louvain clustering optimized for NMI on the integrated dataset.

2.4.1.3 *Average Silhouette Width*

The silhouette width quantifies the association between a cell's within-cluster distances and its between-cluster distances to the nearest cluster. ASW is obtained by averaging over all silhouette widths for a set of cells, with values ranging from -1 to 1. ASW can be used both as a batch effect metric and bioconservation metric, as outlined below.

For bioconservation, ASW is calculated as

$$ASW_{CellType} = \frac{ASW_C + 1}{2} \tag{2.25}$$

where C is the set of all cell identity labels.

For batch effects the metric is instead calculated as

$$ASW_{BatchEffect} = \frac{1}{M} \sum_{j \in M} \frac{1}{|C_j|} \sum_{i \in C_j} 1 - |s(i)| \tag{2.26}$$

where $s(i)$ is the absolute silhouette width on batch labels per cell i, C_j is the set of samples with cell label j, $|C_j|$ is the number of cells of cell label j, and M is the number of unique cell types.

2.4.1.4 *Isolated label Average Silhouette Width*

Isolated label ASW applies ASW on samples with isolated labels shared by few batches to see how well they are distinguished compared to all other labels. Isolated cell types are identified by finding labels occuring in the fewest number of batch effects. The final metric is calculated as the mean of ASW over all isolated labels.

2.4.1.5 *Principal Component Regression*

PCR can be used for estimating the extent of batch effect removal, and is derived as

$$Var(C|B) = \sum_{i}^{G} Var(C|PC_i) \cdot \mathcal{R}^2(PC_i|B) \tag{2.27}$$

where C is a input data matrix, B is a specified batch effect variable (e.g., patient ID), PC stands for principal component, and G is the number of principle components to be used. Here $\mathcal{R}^2(PC_i|B)$ is calculated as the $\mathcal{R}^2$ value from a linear regression of B onto the ith PC and $Var(C|PC_i)$ is the variance of C explained by PC_i.

2.4.1.6 *Graph Connectivity*

GC is calculated by first deriving the k-NN graph $G(N_y; E_y)$ for a given cell type label y, and repeating for all cell types. Next, the GC metric itself is calculated as

$$GC = \frac{1}{|C|} \sum_{y \in C} \frac{|LCC(G(N_y; E_y))|}{|N_y|} \tag{2.28}$$

where C is the set of all cell identity labels, $|C|$ is the number of cells in set C, $|LCC()|$ is the node count of the largest linked component in the graph $G(N_y; E_y)$, and $|N_y|$ is the number of nodes with cell type y.

GC gives the user a measure of how well-integrated the data is by assessing whether the k-NN graph is connecting every cell sharing the same cell type.

2.4.1.7 *Isolated label F_1 score*

We can optimize the Louvain clustering assignment for an isolated label by assessing the F_1 score across various resolutions, ranging from a resolution of 0.1 to 2, with increments of 0.1. The final metric for evaluation is the optimal F_1 score obtained for the isolated label. F_1 is calculated as

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{2.29}$$

where P stands for precision, calculated as

$$P = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{2.30}$$

and R for recall, derived as

$$R = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2.31}$$

2.4.1.8 *Cell-cycle Conservation*

The CC score assesses the effectiveness of capturing the cell-cycle effect both before and after integration. CC is calculated by employing Scanpy's *score_cell_cycle* function, utilizing a reference gene set from Tirosh *et al.* [49] specific to the respective cell-cycle phases. Variance contributions are computed for the *S* and *G2/M* phase scores from *score_cell_cycle* using principle component regression, and the final CC metric is given by

$$CC = 1 - \frac{|Var_{post} - Var_{init}|}{Var_{init}} \tag{2.32}$$

where Var_{init} is the variance before integration and Var_{post} is the variance after.

2.4.1.9 Overall metric

To get an estimate of how well a model performs batch effect removal while conserving biological information, the final step of the evaluation and benchmarking process is to calculate the average over all metrics reflecting each aspect of interest. The batch effect removal metric is calculated as

$$S_{BatchEffect} = \frac{1}{|N_{BatchEffect}|} \sum_{n_i \in N_{BatchEffect}} n_i(X) \tag{2.33}$$

where $N_{BatchEffect}$ is the set of batch effect removal metrics ($\text{ASW}_{BatchEffect}$, PCR, and GC) and n_i is metric function i applied to input data X.

Bio conservation metric is calculated as

$$S_{CellType} = \frac{1}{|N_{CellType}|} \sum_{n_i \in N_{CellType}} n_i(X) \tag{2.34}$$

where $N_{CellType}$ is the set of bioconservation metrics ($\text{ASW}_{CellType}$, NMI, ARI, CC, isolated label F1, and isolated label silhouette) and n_i is metric function i applied to input data X.

To calculate the overall performance, we take a weighted sum of the output from Equations 2.33 and 2.34

$$S_{Overall} = 0.6 \cdot S_{CellType} + 0.4 \cdot S_{BatchEffect} \tag{2.35}$$

Bioconservation is weighted slightly more than batch effect removal to highlight the importance of conserving the biological context.

Before computing $S_{Overall}$, $S_{BatchEffect}$, and $S_{CellType}$, we first need to normalize each metric, from NMI to CC, such that they are comparable. This is achieved by min-max normalizing each metric as follows

$$g(x) = \frac{x - min(x)}{max(x) - min(x)} \tag{2.36}$$

where x is a vector containing the metric score for all models of a given metric.

2.5 Producing cell type representation vectors

A promising application of the embedding space produced by the models is to create cell type representations. Since the model will return different outputs for different inputs of the same cell type, a method is required to aggregate learned embeddings corresponding to the same cell type, such that there is a single embedding for each label. Some options are:

- Calculating the centroid of each cell type cluster.
- Calculating the median of each output dimension for each cell type.
- Calculating the medoid of each cell type cluster.

All of these approaches were implemented with the intention of enabling future users of the model to effortlessly generate desirable cell type representations from scRNA-Seq data.

2.6 Cell type annotation

Another application for the generated embedding space is to use it for classification tasks, such as cell type annotation. To explore this application, a feed-forward neural network was implemented. This classifier accepts the embeddings learned from scRNA-Seq data as input and predicts cell types using a softmax function. The output from the softmax is fed to the cross-entropy loss function *torch.nn.CrossEntropyLoss* implemented in the *Pytorch* library, and the cross-entropy loss is minimized during training [50].

We start by training the cell encoder to learn a latent space from transcriptomics data. The learned embeddings can then be used as input to the classifier model. When training the classifier, we only update the parameters of the classifier, and not the embedding producing model, as this can have undesirable affects on the embedding space.

The model architecture of the final cell type classifier is illustrated in Figure 2.7.

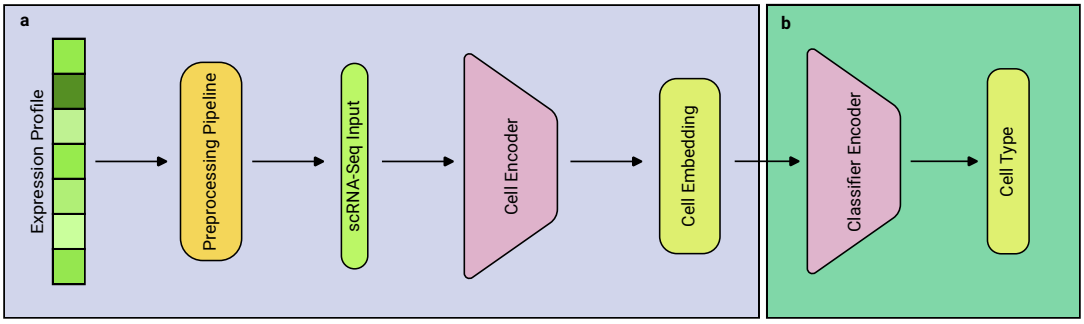


Figure 2.7: An overview of the classifier architecture, conditioned in this illustration on the output from Model 1. **a** ScRNA-Seq data is fed through a pre-trained feed-forward encoder that learns a latent space. **b** Embeddings from this space can be used as input to another feed-forward neural network to classify samples into potential cell types.

Benchmarking of our models' cell type annotation capabilities was done by comparing them to the following models:

- *scNym* [19]
- *Seurat* [9]
- *TOSICA* [6]
- *SciBet* [8]
- and *CellID* [20].

Three metrics were used to assess the performance of all models: accuracy, balanced accuracy, and F1 score. All of these metric have already been implemented in the *scikit-learn* package, *sklearn.metrics.accuracy_score*, *sklearn.metrics.balanced_accuracy_score*, and *sklearn.metrics.f1_score*, and were used in this study to evaluate model performance [51]. For *sklearn.metrics.f1_score* the *average* parameter was set to "*weighted*".

The models developed in this study have a built-in flag for automatically optimizing hyperparameters associated with the classifier part using *Optuna* [52]. The model takes the data, splits it 80 : 20 for training and validation, and runs 100 trials to find optimum hyperparameters using *Optuna*. This has been implemented as a default feature of the model since training the classifier is not computationally demanding. The following hyperparameters are being optimized:

- Number of neurons in the first hidden layer
- Number of neurons in the second hidden layer
- Initial learning rate
- Dropout percentage

2.7 Novel cell type detection

Since a cell type annotator is trained using specific data, it can only recognize and predict the cell types present in its training dataset. If this model is later used to analyze data containing unfamiliar cell types, it will not be able to accurately identify these new types. Instead, it will mistakenly classify them as one of the known types it was trained on, leading to incorrect predictions. However, it can be possible to identify whether a set of samples contains a previously unidentified cell type. One approach for detecting novel cell types is as follows:

1. Make predictions on data and retrieve the likelihood of each sample corresponding to a given cell type.
2. For each sample, min-max normalize the likelihoods across all possible cell types, where the maximum is a likelihood of 1.0, and the minimum is the inverse of the number of unique labels (cell types).
3. Extract the minimum likelihood out of the set of all predictions.
4. Compare this value to a pre-determined threshold (0.XX). A value below this threshold suggest that the sample could correspond to a novel cell type previously unseen in the data.

The intuition behind this approach is that a model which is not confident in any of the known labels (cell types) for a given sample implies that the sample is possibly a novel cell type, previously unseen.

2.8 Data and Code Availability

The reproducibility code for this work is available at: `https://github.com/LeoAnd00/Masters_Thesis`.
The data for this work is available at: `https://doi.org/10.5281/zenodo.10959788`.

The final model is currently being encapsulated into a Python package, facilitating its seamless integration and utilization by researchers. The following functions will be included:

- Training function for the embedding space model.
- Training function for the classifier model.
- Predict function for generating an embedding space.
- Predict function for performing cell type annotation.
- Function for novel cell type detection.
- Function for creating cell type representation vectors.

- Function for applying the same normalization strategy as was used in this study, giving the end user the option of using the same strategy or implementing their own.
- Function for automatic preprocessing, although it is still recommended for end users to use their own preprocessing pipeline to make sure it is appropriate for their data.

3
Results

In this section, we present the results of this thesis, providing a comprehensive analysis of key findings and insights in cell representation learning.

3.1 Preprocessing of data

After preprocessing all the scRNA-Seq data one can see how batch effects have a visible impact on the data when visualized using Uniform Manifold Approximation and Projection (UMAP).

A UMAP of the bone marrow dataset was computed and colored according to the 20 cell types (Fig. 3.1a) and 5 patient IDs (Fig. 3.1b). The severity of the batch effect appears to be relatively mild as some mixing is observed between patient data.

The UMAP of the PBMC dataset shows all 17 cell types (Fig. 3.2a) and 6 patient IDs (Fig. 3.2b). When compared to the bone marrow dataset, one can see how the batch effect is much more severe for the PBMC data, showing the importance of being able to adjust and correct for such effects.

Visualizations of the pancreas, kidney and merged dataset can be seen in Appendix A.1.1, A.1.2, and A.1.3. The Baron, MacParland, Segerstolpe, and Zheng68k datasets used for classification tasks are visualized in Appendix A.1.4, A.1.5, A.1.6, and A.1.7.

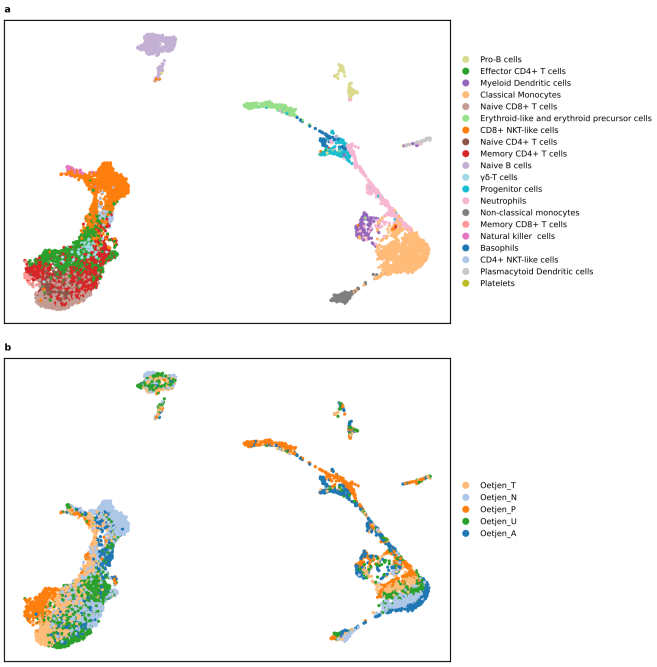


Figure 3.1: **a** UMAP visualization of cell type clusters in the bone marrow dataset. **b** UMAP highlighting batch effect caused by patient ID (different patients).

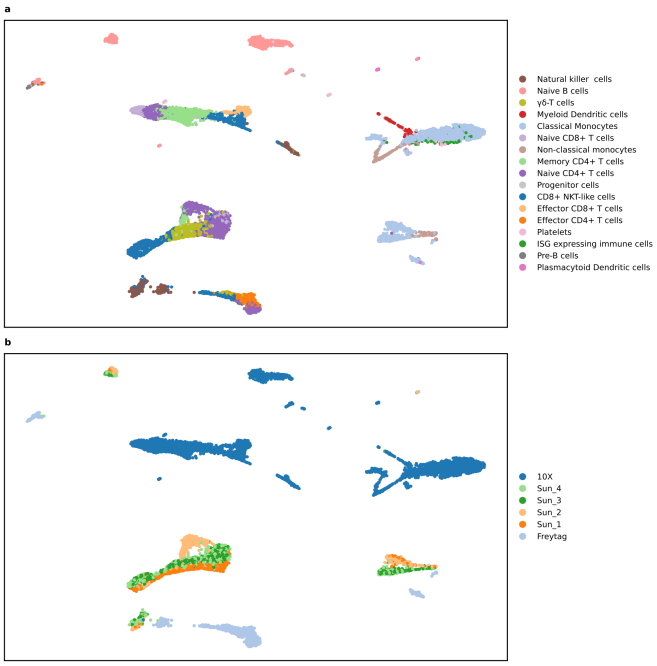


Figure 3.2: **a** UMAP visualization of cell type clusters in the PBMC dataset. **b** UMAP highlighting batch effects caused by patient ID.

3.2 Investigation of patient ID as main cause of unwanted variation

During the formulation of the loss function described in Equation 2.16, it is assumed that patient ID is the main source of unwanted variation. To investigate whether or not this is a reasonable assumption, one can evaluate the coefficient of variation (CV). It is derived as

$$CV = \frac{\sigma}{\mu} \tag{3.1}$$

where μ is the mean Euclidean distance across patients between two cell type clusters and σ is the standard deviation of the Euclidean distance. The CV score metric can vary between a value of 0 and infinity, where a lower value is preferable. If you for instance have a CV score of 1, then the standard deviation would be equal to the mean, indicating fairly large variability in your data.

A heatmap of the average normalized Euclidean distance between cell type clusters across patients in the PCA transformed latent space (Fig. 3.3a) and the corresponding CV scores (Fig. 3.3b) for the merged dataset can be seen below. Ideally one would observe that the CV scores would be 0 for all Euclidean distance between cell type clusters. This would mean that the normalized Euclidean distance between cell types within each patient is the same. However, in reality there exists many sources of errors that can affect the CV calculation and is therefor unlikely to be 0, even if the underlying assumption is true.

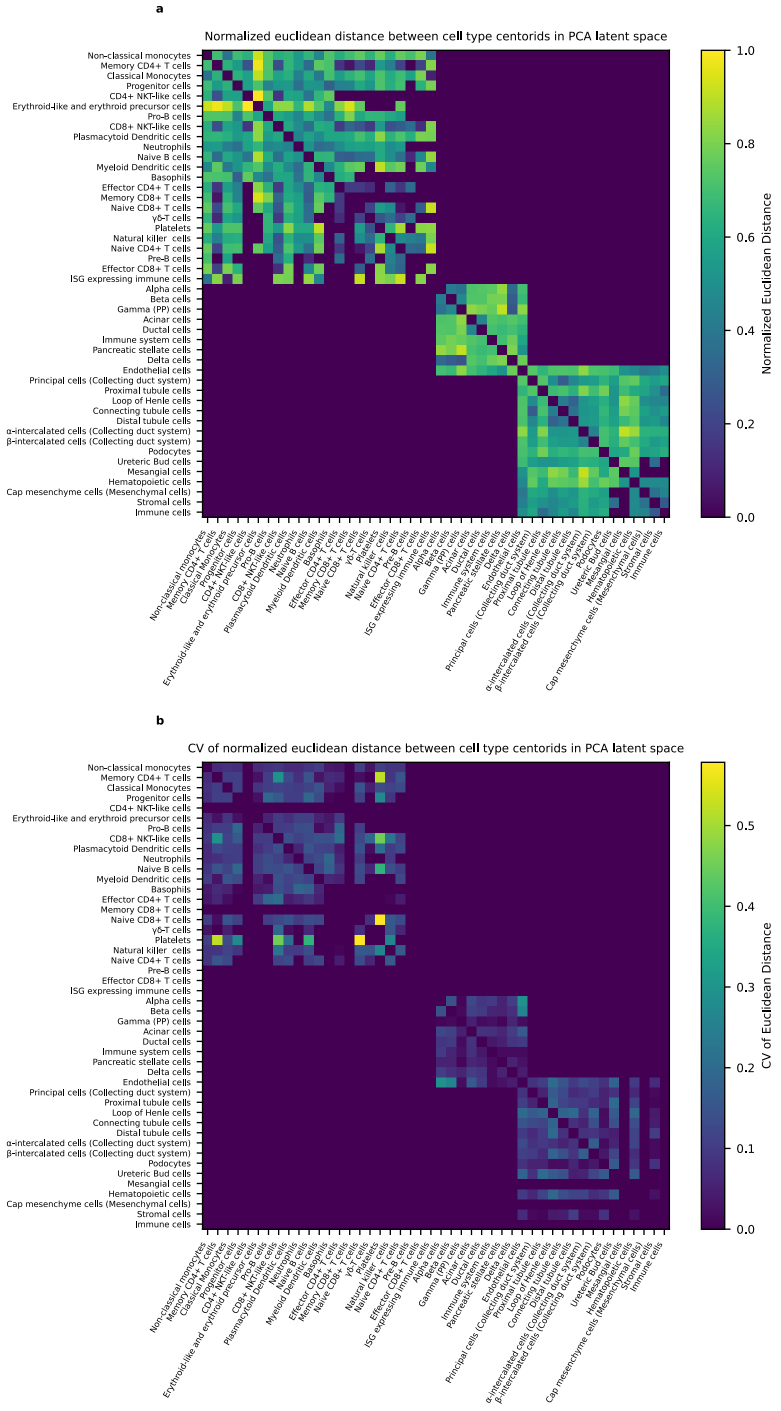


Figure 3.3: **a** Normalized mean Euclidean distance across patient IDs between centroids of cell type clusters in PCA transformed latent space of the merged dataset. **b** CV of Euclidean distance between centroids of cell type clusters in PCA transformed latent space of the merged dataset.

It is worth noting that there are many pairs with a value of zero. A position is assigned a zero if there does not exist scRNA-Seq data for any patient ID where the two cell types exists. These pairs are excluded from this calculation. There are more

zero elements than non-zero elements in the CV heatmap due to the fact that there must exist at least two patient IDs containing the two cell types for an element to be non-zero, which is not commonly occurring. Another point worth discussing is that the merged dataset consists of data from bone marrow, PBMC, pancreas and kidney. This causes the formation of the three "squares" in the distance matrix; from left to right (Fig. 3.3a) we have bone marrow and PBMC forming one (combined) square from their entries since they share a lot of cell types in common. Next there is a square corresponding to entries from the pancreas cell types, and finally a square in the bottom-right corner due to kidney cells.

The mean CV score is 0.1068 with a standard deviation of 0.0764, indicating a fairly low CV score overall (Fig. 3.4). This supports the assumption that the major source of unwanted variability is patient ID, since CV is low when one assumes that patient ID is the main source of unwanted variation. The remaining variation in the data is likely due to less prominent batch effects.

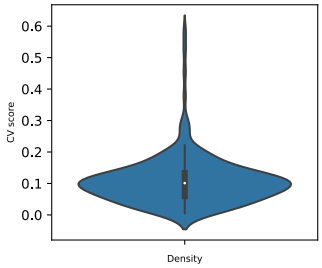


Figure 3.4: Density of CV scores calculated from Euclidean distance between centroids of cell type clusters in the PCA transformed latent space of the merged dataset.

3.3 Benchmark on generalizable scRNA-Seq embedding space

An ideal data integration algorithm would be able to enforce the biological variation in the data while minimizing variability caused by additional factors, such as patient ID, tissue, sequencing technology, and so on. In order to investigate how well a model performs on this task, the *scIB* package was used to calculate suitable metrics to evaluate the performance [15]. Metrics were min-max normalized across methods for each metric and at each fold. Multiple previously developed methods are compared to the models created in this study for a better understanding of how well they perform on creating a generalizable embedding space. In addition, the input data was used directly to compute metrics in the benchmark as a control. This calculation is named Unint (Unintegrated). Benchmarking was performed for the bone marrow, PBMC, pancreas, kidney and merged dataset (Fig. 3.5), where patientID is considered to be the batch effect.

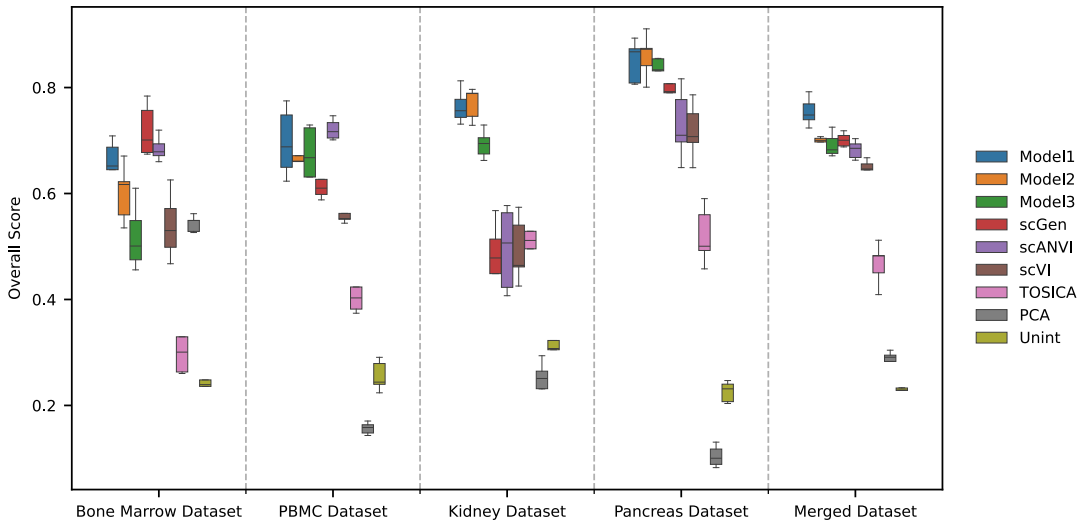


Figure 3.5: Five-fold cross-testing on the bone marrow, PBMC, kidney, pancreas, and merged dataset, where the overall scores were calculated as described in Equation 2.35 from min-max normalized *scIB* metrics. The following models are used in the benchmark: Unint (Unintegrated), PCA, TOSICA [6], scVI [16], scANVI [13], scGen [14], Model 1 (this work), Model 2 (this work), and Model 3 (this work). The input to the methods are the top 2000 HVGs. Box plots show the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range.

Here we see how *scIB* ranks the models developed in this study (Models 1-3) among the best performing models for each dataset, and particularly Model 1 for the merged dataset. We observe that Model 1 is the best performing model for the four datasets tested out of the three models developed in this study. This indicates that, for these datasets, the attention mechanism does not aid the model during training, and feeding the expression levels directly into a feed-forward encoder achieves better

performance when combined with the engineered learning approach. Performance of all models, on all metrics, and for each dataset can be seen in Appendices A.2.1, A.2.2, A.2.3, A.2.4, and A.2.5.

To get a more visual interpretation of how Model 1 performs at making a generalizable embedding space, UMAP is used on one of the test folds of the kidney dataset when the data is PCA transformed (Fig. 3.6a,b) and generated using Model 1 (Fig. 3.6c,d).

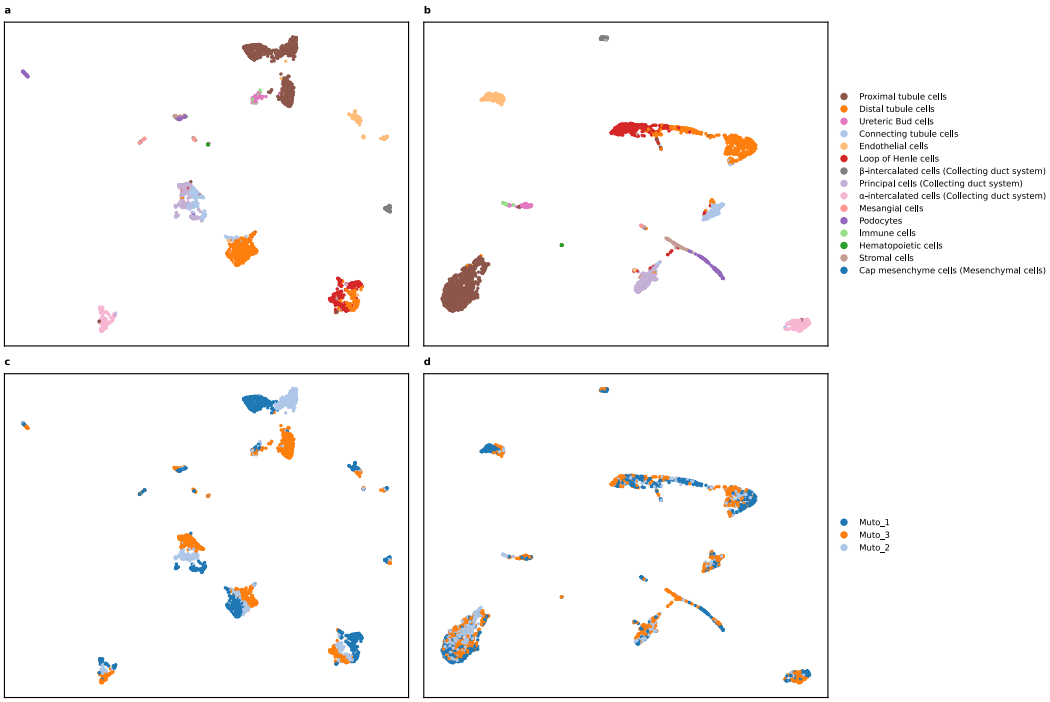


Figure 3.6: **a** UMAP of the first test fold for the kidney dataset when transformed using PCA and colored by cell type. **b** UMAP of the first test fold for the kidney dataset when transformed using PCA and colored by patient ID. **c** UMAP of the first test fold for the kidney dataset when transformed using Model 1 and colored by cell type. **d** UMAP of the first test fold for the kidney dataset when transformed using Model 1 and colored by patient ID.

We observe a significant reduction in batch effects when utilizing Model 1 compared to performing PCA transformation (Fig. 3.6b,d), as the datapoints are better integrated within and across clusters. This underscores the effectiveness of Model 1 in generating a meaningful embedding space.

3.4 Cell type annotation benchmark

In order to demonstrate a downstream use case for the learned cellular embedding space, it was used as input to a classifier. Using four different datasets the performance of this cell type annotator is compared to multiple other established models (Fig. 3.7).

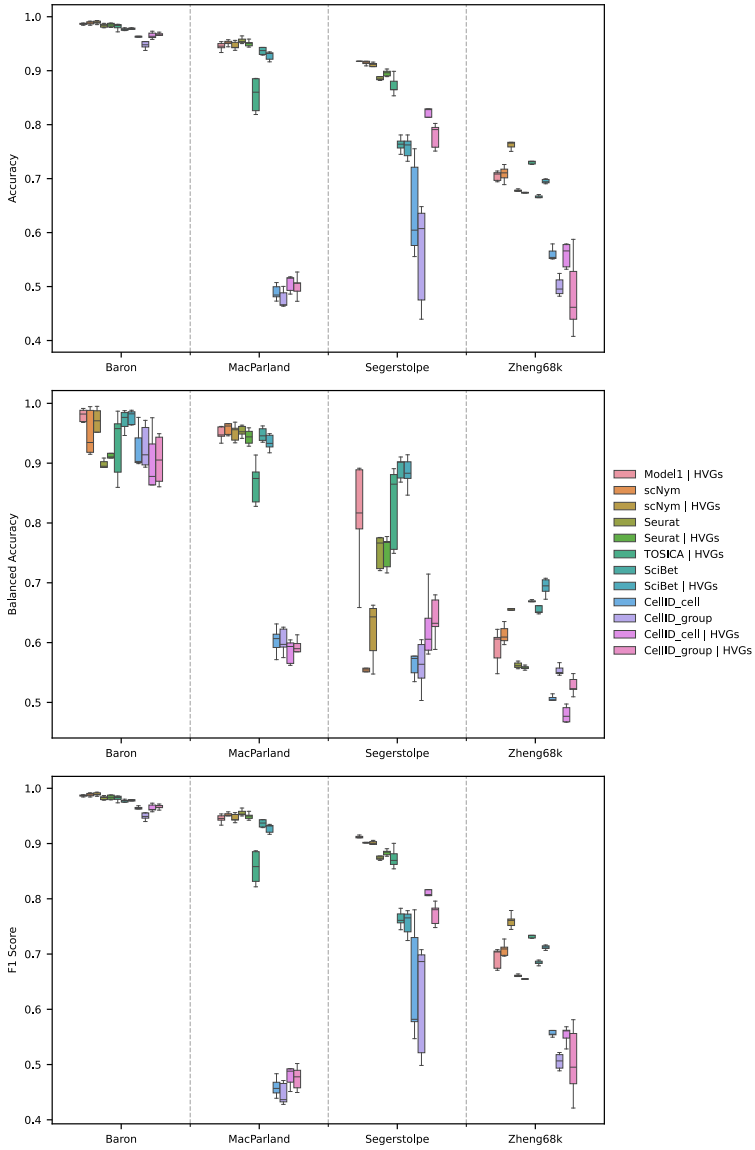


Figure 3.7: Five-fold cross-testing on the Baron, MacParland, Segerstolpe, and Zheng68k dataset, where accuracy, balanced accuracy and F1 score are used as metrics. The following models are used in the benchmark: Model 1, scNym [19], Seurat [9], TOSICA [6], SciBet [8], and CellID [20]. The input to the methods are either all genes or the top 2000 HVGs, specified by the " | HVGs" appended to the end of each label in the legend. Box plots show the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range.

We see here that for cell type annotation as well, Model 1 tends to be among the top performing methods for all datasets and metrics. Nevertheless, it is difficult to determine which model performs the best overall. To attempt evaluating this, the mean metric of each model and dataset was calculated. Then the mean values of each metric were min-max normalized across models for each dataset to make the metrics comparable between datasets. This is visualized (Fig. 3.8a) using a box plot. The benefit of using balanced accuracy for evaluating annotation performance of scRNA-Seq datasets is that they tend to be unbalanced. Balanced accuracy is better at taking the unbalanced nature of the data into account compared to calculating ordinary accuracy. Hence one could argue that balanced accuracy is a more informative metric in this instance. But during the calculation of the average metric score it is not weighted heavier than any other metric.

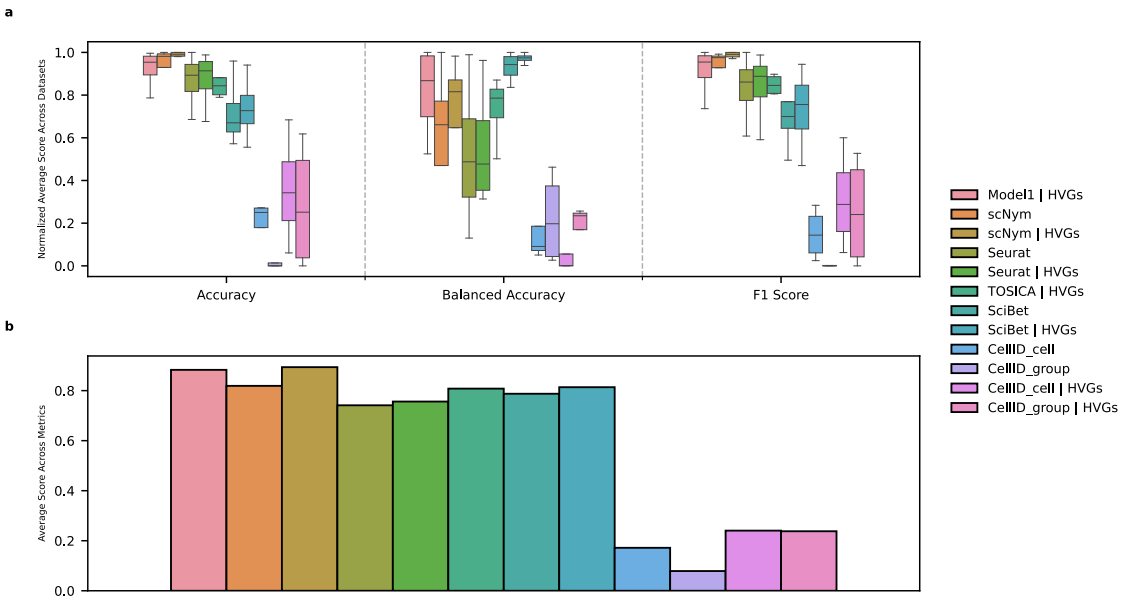


Figure 3.8: **a** Min-max normalized metrics averaged across the Baron, MacParland, Segerstolpe, and Zheng68k datasets. Box plots show the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range. **b** Bar plot of the mean value across the average values of the metric scores.

Next, the mean value of each model for each metric was calculated and the average across these values was visualized as a bar plot (Fig. 3.8b). Here, the overall performance of all models becomes clearer. It can be seen that scNym emerges as the top-performing model when utilizing highly variable genes (HVGs), with Model 1 closely following in second place. Notably, in the original scNym paper, the authors did not explore the use of HVGs, leading so a sub-par performance for the original model. However, it's important to note that Model 1's utilization of HVGs is the default approach, integrated within its workflow. This distinction makes Model 1 the best performing model under the default settings. Part of the noteworthy contributions of this study, furthermore, include an approach to improving existing methodologies, such as scNym via the use of HVGs.

3.5 Loss function comparison

In this section, the engineered loss function utilized for creating an embedding space from scRNA-Seq data, as described by Equation 2.17, is compared to employing only its constituent parts, namely the contrastive loss and centroid loss, or Equations 2.15 and 2.16, respectively. This comparative analysis serves to underscore the significance of employing the complete loss function (Fig. 3.9).

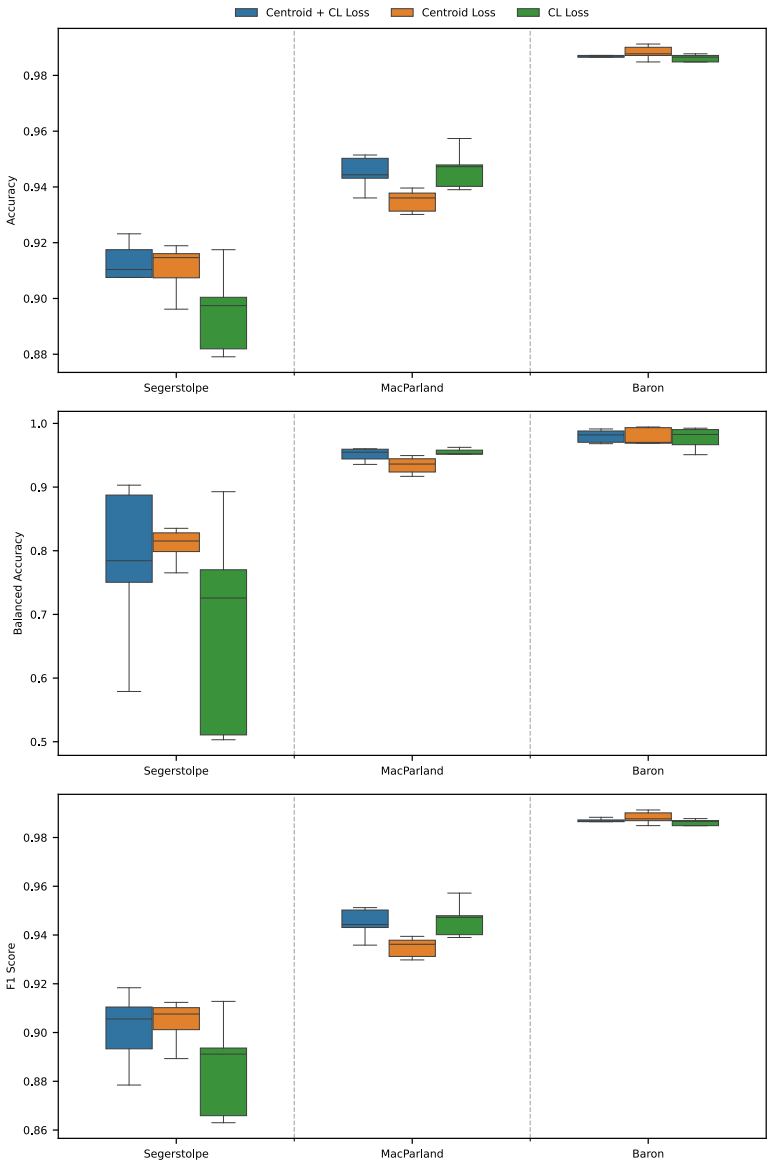


Figure 3.9: Five-fold cross-testing of Model 1 with different losses on the Baron, MacParland, and Segerstolpe, datasets. Here, accuracy, balanced accuracy, and F1 score are shown. The following losses are used in the benchmark: Centroid + CL Loss, Centroid Loss, and CL Loss. The input to the models are the top 2000 HVGs. Box plots show the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range.

Worth noting is how the cell type centroid loss (Equation 2.16) under-performed on the MacParland dataset, while the contrastive learning loss (Equation 2.15) under-performed on the Segerstolpe dataset. We observe that the use of the complete loss function consistently leads to the highest performance across datasets.

We also evaluated the embedding space in terms of its performance in the benchmarking metrics from Luecken *et al.* [15]. These results are visualized in Figure 3.10.

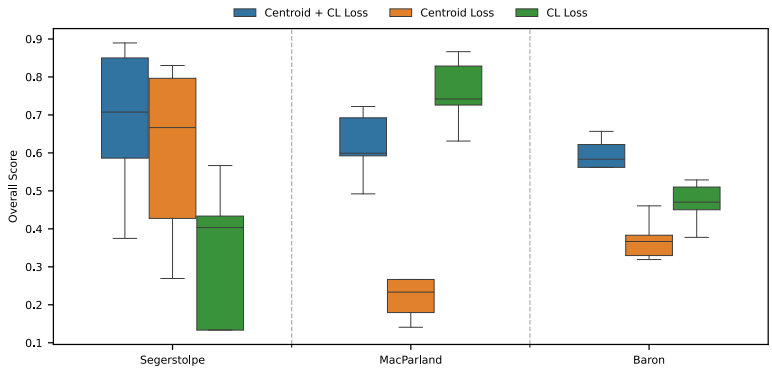


Figure 3.10: Five-fold cross-testing of Model 1 on the Baron, MacParland, and Segerstolpe datasets, where the overall score was calculated as described in Equation 2.35 from min-max normalized *scIB* metrics. The following models are used in the benchmark: Centroid + CL Loss, Centroid Loss, and CL Loss. The input to the models are the top 2000 HVGs. Box plots show the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range.

We again observe how the complete loss function leads to the best and most consistent performance across the three datasets, further justifying the its use in this work.

3.6 Novel cell type detection

As seen in the previous sections, Model 1 demonstrates promising performance for creating a generalizable embedding space for cell types, as well as good performance in cell type classification. Another application of the model would be to use it for novel cell type detection. When working with neural network approaches for classification, it is challenging for neural networks to generalize beyond classes seen during training, as one typically does not have a class for "novel" items. Instead, we can judge the novelty of new datapoints by setting a threshold on the likelihood of sampling known classes for a new datapoint. If a prediction has a likelihood lower than this threshold for all classes, then it can be considered novel. To investigate whether Model 1 can be used for this purpose, we carried out an experiment where one cell type was left out of training at a time from each of the MacParland, Baron, and Zheng68k datasets. The data of the removed cell type is then concatenated to the test fold, so that one gets predictions both on known cell types and novel ones. For each test fold, dataset, and cell type left out during training, the minimum likelihood of any non-novel prediction was calculated along with the minimum likelihood of any novel guess (Fig. 3.11).

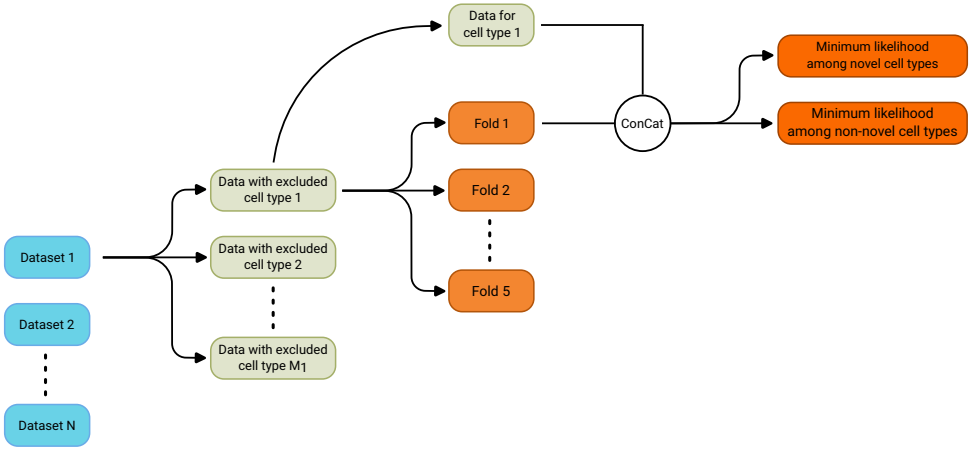


Figure 3.11: For each dataset we exclude one cell type at a time. Then, For each instance of excluding a cell type we use a five-fold split, where each fold is used for testing. The data of the excluded cell type is then concatenated to the test fold, acting as our novel cell type. Finally, the minimum likelihood produced by Model 1 is then gathered for non-novel cell type and novel cell type samples separately. This eventually creates $5 \cdot \sum_{i=1}^{N} \mathrm{M}_i$ total folds, where M_i is the number of cell types in dataset i.

The results from utilizing this workflow are visualized in Figure 3.12.

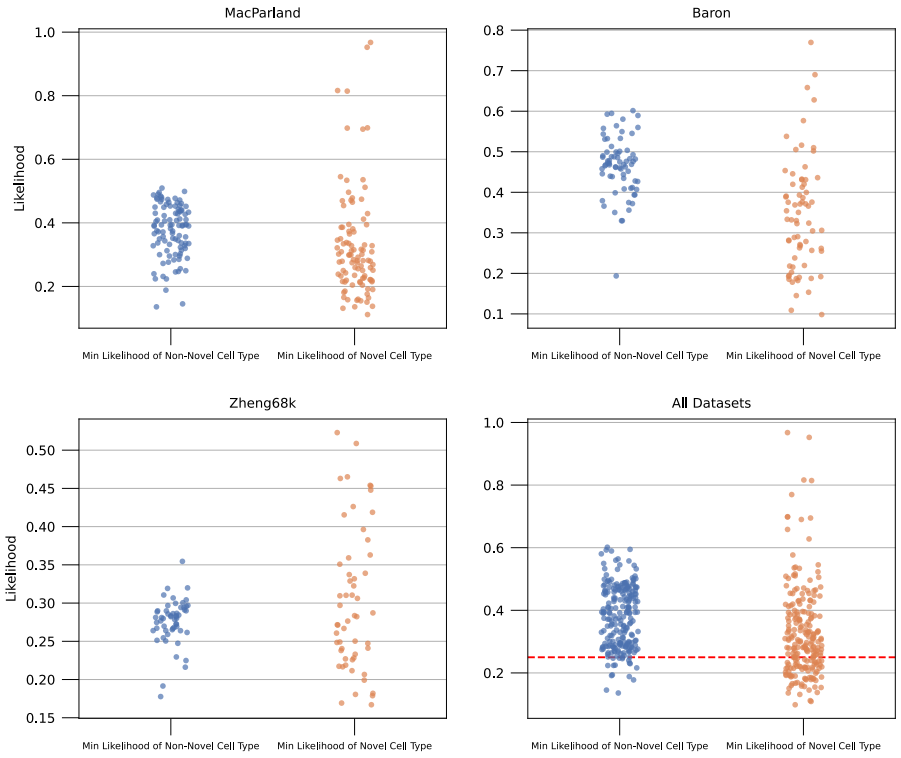


Figure 3.12: Jitter plot visualizing the distribution of minimum likelihood levels attained by predictions made using Model 1 for both non-novel and novel cell types. Each point represents the minimum likelihood observed for predictions of non-novel (known) and novel (unknown) cell types when one cell type was omitted from training across various test folds and datasets (MacParland, Baron, Zheng68k, and All Datasets). The red doted line represent a likelihood threshold of 0.25; here, all data points falling below the line would be considered to contain novel cell types according to the model. The All Datasets panel consists of all data points from the three other datasets combined into a single plot. Data points from MacParland, Baron, and Zheng68k are min-max normalized in relation to the maximum and minimum possible likelihood of each dataset, making datasets comparable in the All Datasets panel.

We investigated how picking different threshold values influenced the predictions made by the novel cell type detection model. A likelihood threshold value of 0.25 is illustrated in the *All Datasets* panel of Fig. 3.12. This value achieves a high precision of 0.81 while maintaining a reasonable coverage of 0.33. The impact of the threshold value on precision and coverage is demonstrated in Figure 3.13.

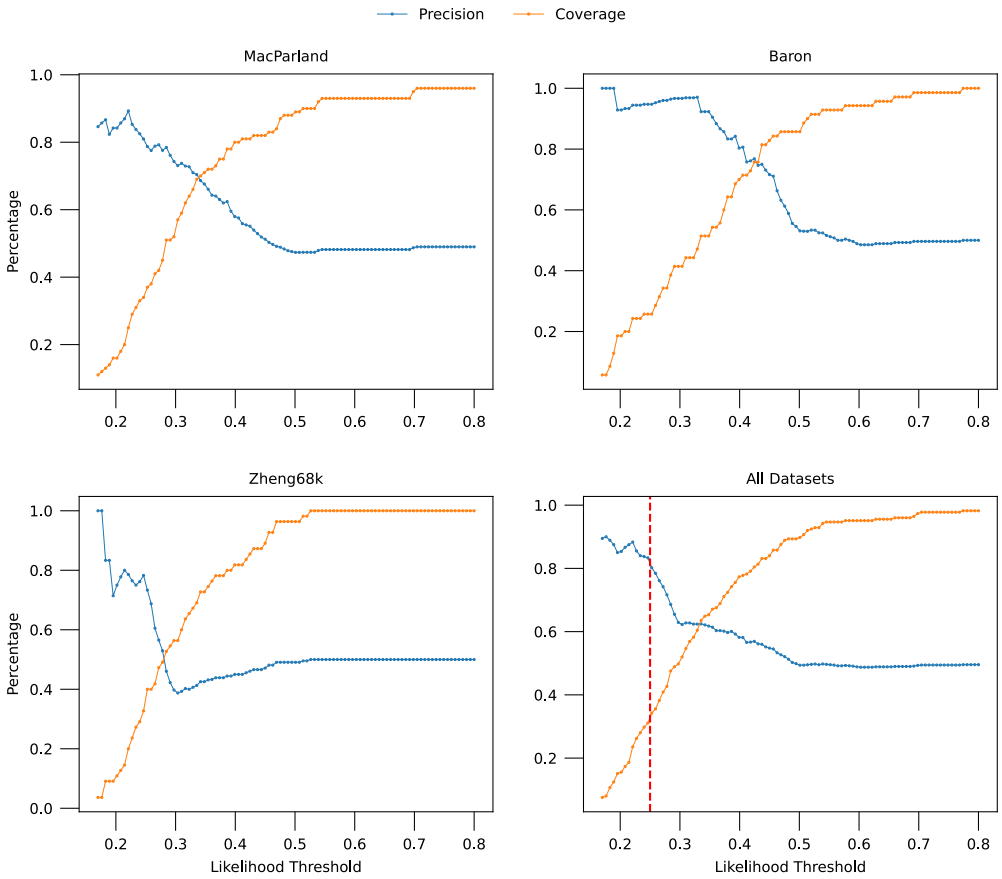


Figure 3.13: The relationship between the likelihood threshold used and the precision and coverage for detecting novel cell types using Model 1. The red doted line represents a likelihood threshold of 0.25, which leads to a precision of 0.81 and a coverage of 0.33 for novel cell-type detection across all datasets. The trade-off between precision and coverage is evident, allowing researchers to choose threshold values based on their desired model behavior for novel cell type detection.

This shows how one can vary the threshold value for novel cell-type detection depending on which metric one chooses to prioritize: precision, or coverage. For instance, the choice of 0.25 is used as an example, reflecting a reasonable compromise between coverage and precision, and favoring the attainment of high precision.

4

Discussion

The model developed in this thesis have displayed impressive performance on two benchmarks. It was overall the best performing model for making a generalizable scRNA-Seq embedding space, and the second best for cell type classification. On top of this it also has the ability to identify the presence of novel cell types in data. Due to its multiple functionalities, the developed model becomes a simple and useful tool for end users.

4.1 ScRNA-Seq data augmentations

Some augmentations one can try for scRNA-Seq data are:

- Randomly mask a specified percentage of gene expressions (i.e., make them zero).
- Add Gaussian noise by randomly replacing a specified percentage of gene expressions with a sample from a Gaussian distribution fitted to the expression distribution of the given gene for the given cell type.

The concept of artificially masking gene expressions aims to simulate the occurrence of errors during sequencing, leading to the dropout of gene expression. However, since gene expressions are typically dependent on expression of other genes one can imagine that randomly removing expression levels, essentially removing known information, can make the model less prone to discover these connections. Hence one could argue that this augmentation of sc-omics data might not be desirable. But it could also make the model better at generalizing, in which case this augmentation could be desirable. In summary, this can be a worthwhile augmentation strategy worth investigating.

Another potent augmentation strategy is fitting a Gaussian distribution to the gene expression levels, then sampling noise from this Gaussian and adding it to the training data. This can be a valid strategy to generate more artificial data for the training process, potentially making the model better at generalizing. Assuming the noise of each gene expression within each cell type is minor, this augmentation strategy can be valid and worth implementing.

4.2 Novel cell type detection

Detecting novel cell types in data is a difficult task. When looking at the precision and coverage of the Zheng68k dataset (Fig. 3.13) one can see how we would only achieve a high precision by sacrificing a lot of coverage. As this is not generally a worthwhile trade-off, this means the model would most often miss the presence of a novel cell type in the data. In contrast, if one looks at the Baron dataset (Fig. 3.13) one can achieve both a fairly high precision and coverage, making the model useful for novel cell type detection on this dataset. The key insight here is that the novel cell type detection capability of the model is highly reliant on the characteristics of the data. Consequently, selecting an optimal likelihood threshold that yields the best overall performance becomes challenging. An attempt at this was made by combining data points from all datasets and picking a likelihood threshold of 0.25 based on all data. This gave a high precision and a acceptable coverage. Nonetheless, the model can serve as a valuable additional verification step within novel cell type detection pipelines. In scenarios where the model detects something novel, it provides an extra layer of confidence in the presence of genuinely novel cell types. Conversely, if the model fails to detect anything novel, it should not be interpreted as an absence of novel cell types. Instead, it indicates a limitation in drawing conclusions based solely on this approach.

4.3 Future applications

4.3.1 Cell type representations

Model 1 developed in this study can be used for producing a generalizable embedding space. By generalizable, we mean that this embedding space can then go on to be used in a variety of downstream tasks, such as cell type annotation, or for novel cell type detection, both of which we demonstrated in this study. Another application which was implemented but not explored is the use of the embedding space to create cell type representation vectors. Three options were implemented to achieve this, calculating the centroid, median, or medoid of each cell type cluster to make a vector. However, the performance of using these representations as input to other machine learning applications was not investigated and is left for future research. One possible application for these representations would be to provide molecular generative models with cell type information in drug discovery efforts; in this way, generative models would propose molecules conditioned on the target cell type.

4.3.2 Interpretable attention space

Another interesting application that was investigated involves the utilization of the method described in Section 2.3.5 to interpret the attention latent space of sample input. It is worth mentioning that, from this analysis, it appears that the model learns to attend to the input in a semi-random manner. That is, we were not able to extract much meaning from the learned attention weights. Sometimes, the way the model attends to the input is logical and supported by the literature, but at

other times, it seems random. This indicates that further investigations are needed to ensure that attention-based models, such as Model 2 and Model 3, establish connections that are consistently meaningful and supported by the existing literature.

4.3.3 Multi-modal model

4.3.3.1 Combining scRNA-Seq data with *Cell Painting* images

Due to its ability to cluster inputs in a latent space, contrastive learning could be used to combine scRNA-Seq information and *Cell Painting* images to create a cell type embedding space. *Cell Painting* images have been used in many ways to discern information about cellular properties. For instance, it is been shown that *Cell Painting* images can be used to discern cell health when exposed to different CRISPR perturbations which target genes related to known biological pathways and inducing morphological changes [53]. *Cell Painting* images are utilized as an unbiased representation of quantitative information regarding the condition or status of cells, making its utilization advantageous [54]. This has the potential to enhance the meaningfulness of the latent space by exposing Model 1 to more information related to cell types, making the model better at downstream applications. However, for this to work, one would need many *Cell Painting* images of the same cell types for which scRNA-Seq data is available. Due to the lack of a Cell Painting image database with diverse enough cell types, this option was not further investigated but is recognized as an interesting approach for future research.

4.3.3.2 Integrated multi-omics

Yet another way of modifying Model 1 would be to introduce other forms of omics data, such as single-cell assay for transposase-accessible chromatin using sequencing (scATAC-Seq) data into the model. The benefit of using contrastive learning in this case would be that one version of Model 1 takes scRNA-Seq data as input while another version takes scATAC-Seq. By using multiple different types of data, such as scRNA-Seq, scATAC-Seq, and Cell Painting images, one encourages the model to capture more information describing cells. This should make it possible to create a space that more accurately represents how cell types are related to each other, motivating the use of a multi-modal model.

4.4 Limitations

As with most models, there are limitations. One such limitation of the model developed in this study is the time needed for training. This can make it problematic if one for instance have a large number of cells. The biggest dataset used for training in this study was the Zheng68k dataset, containing around 68k cells. It was manageable to train with this amount of data. But lets say you have 1 million cells. Then the computational time could become an issue depending on the computational power of the used GPU. The reason for the relatively slow training is due to the loss function. Large amount of computations are needed and creates a bottle neck. Trying

to make this part more efficient would therefor be a great improvement to the model.

Another point worth discussing is that Model 1 outperformed models utilizing attention, such as Model 2 and Model 3. There could be many reasons for why this occurred, but one point which seems likely to cause issues for the attention based models is that the expression levels need to be tokenized. This increases the risk of missing key connections between genes at their very specific expression levels. Since Model 1 is capable to take the data directly without tokenizaition, it does not suffer form this disadvantage. It could be that having this information is more valuable than having attention for scRNA-Seq data. Further research is therefor needed to try and make attention beneficial for models using scRNA-Seq data, and also to make the attention interpretably meaningful.

5
Conclusion

In this study, a robust tool for leveraging scRNA-Seq data to generate an generalizable embedding space, leading to the mitigation of batch effects while enforcing biologically relevant information, was developed. Multiple downstream applications were integrated into the package. This comprehensive approach empowers other researchers to explore this method for learning a meaningful cellular embedding space.

The work conducted in this thesis project has advanced the field of single-cell transcriptomics via the following key contributions:

- We introduced a novel model architecture for cell representation learning from scRNA-Seq data using representation learning.
- By using a carefully crafted contrastive loss, we demonstrated that our model is robust to batch effects while maintaining biologically relevant information.
- We demonstrated robust performance of our model on cell type annotation and novel cell type detection, two distinct tasks.

By leveraging the power of contrastive learning and the cell type centroid loss, we successfully crafted a framework that is capable of generating a highly generalizable embedding space for cells from scRNA-Seq data. We visually demonstrated how it is able to mitigate batch effects (Fig. 3.6) and showed that the embedding space learned by the model is biologically meaningful by demonstrating its effectiveness in multiple downstream applications, such as cell type annotation and novel cell type detection.

The cell type annotator, trained on the learned embedding space, demonstrates remarkable performance in classifying known cell types. Furthermore, the novel cell type detection workflow provides a valuable tool for identifying the presence of novel cell types based on the likelihood of predictions made by the annotator. These capabilities demonstrate the model's versatility for various potential applications within scRNA-Seq data analysis.

The model is currently being structured into an open-source Python package, making it easily accessible to researchers. In addition, an article is being written that encapsulates its accomplishments.

Bibliography

1. Stein CM, Weiskirchen R, Damm F, and Strzelecka PM. Single-cell omics: Overview, analysis, and application in biomedical science. Journal of cellular biochemistry 2021; 122:1571–8. DOI: 10.1002/jcb.30134
2. Cuomo AS, Nathan A, Raychaudhuri S, MacArthur DG, and Powell JE. Single-cell genomics meets human genetics. Nature Reviews Genetics 2023 :1–15. DOI: 10.1038/s41576-023-00599-5
3. Bawa G, Liu Z, Yu X, Qin A, and Sun X. Single-cell RNA sequencing for plant research: insights and possible benefits. International Journal of Molecular Sciences 2022; 23:4497. DOI: 10.3390/ijms23094497
4. Lee J, Hyeon DY, and Hwang D. Single-cell multiomics: technologies and data analysis methods. Experimental & Molecular Medicine 2020; 52:1428–42. DOI: 10.1038/s12276-020-0420-2
5. Vandereyken K, Sifrim A, Thienpont B, and Voet T. Methods and applications for single-cell and spatial multi-omics. Nature Reviews Genetics 2023 :1–22. DOI: 10.1038/s41576-023-00580-2
6. Chen J, Xu H, Tao W, Chen Z, Zhao Y, and Han JDJ. Transformer for one stop interpretable cell type annotation. Nature Communications 2023; 14:223. DOI: 10.1038/s41467-023-35923-4
7. Aran D, Looney AP, Liu L, Wu E, Fong V, Hsu A, Chak S, Naikawadi RP, Wolters PJ, Abate AR, et al. Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. Nature immunology 2019; 20:163–72. DOI: 10.1038/s41590-018-0276-y
8. Li C, Liu B, Kang B, Liu Z, Liu Y, Chen C, Ren X, and Zhang Z. SciBet as a portable and fast single cell type identifier. Nature communications 2020; 11:1818. DOI: 10.1038/s41467-020-15523-2
9. Hao Y, Stuart T, Kowalski MH, Choudhary S, Hoffman P, Hartman A, Srivastava A, Molla G, Madad S, Fernandez-Granda C, et al. Dictionary learning for integrative, multimodal and scalable single-cell analysis. Nature biotechnology 2024; 42:293–304. DOI: 10.1038/s41587-023-01767-y
10. Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences 2005; 102:15545–50. DOI: 10.1073/pnas.0506580102

11. Mootha VK, Lindgren CM, Eriksson KF, Subramanian A, Sihag S, Lehar J, Puigserver P, Carlsson E, Ridderstråle M, Laurila E, et al. PGC-1α-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. Nature genetics 2003; 34:267–73. DOI: 10.1038/ng1180

12. Pasquini G, Arias JER, Schäfer P, and Busskamp V. Automated methods for cell type annotation on scRNA-seq data. Computational and Structural Biotechnology Journal 2021; 19:961–9. DOI: 10.1016/j.csbj.2021.01.015

13. Xu C, Lopez R, Mehlman E, Regier J, Jordan MI, and Yosef N. Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. Molecular systems biology 2021; 17:e9620. DOI: 10.15252/msb.20209620

14. Lotfollahi M, Wolf FA, and Theis FJ. scGen predicts single-cell perturbation responses. Nature methods 2019; 16:715–21. DOI: 10.1038/s41592-019-0494-8

15. Luecken MD, Büttner M, Chaichoompu K, Danese A, Interlandi M, Müller MF, Strobl DC, Zappia L, Dugas M, Colomé-Tatché M, et al. Benchmarking atlas-level data integration in single-cell genomics. Nature methods 2022; 19:41–50. DOI: 10.1038/s41592-021-01336-8

16. Lopez R, Regier J, Cole MB, Jordan MI, and Yosef N. Deep generative modeling for single-cell transcriptomics. Nature methods 2018; 15:1053–8. DOI: 10.1038/s41592-018-0229-2

17. Hie B, Bryson B, and Berger B. Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. Nature biotechnology 2019; 37:685–91. DOI: 10.1038/s41587-019-0113-3

18. Korsunsky I, Millard N, Fan J, Slowikowski K, Zhang F, Wei K, Baglaenko Y, Brenner M, Loh Pr, and Raychaudhuri S. Fast, sensitive and accurate integration of single-cell data with Harmony. Nature methods 2019; 16:1289–96. DOI: 10.1038/s41592-019-0619-0

19. Kimmel JC and Kelley DR. scNym: Semi-supervised adversarial neural networks for single cell classification. bioRxiv 2020 :2020–6. DOI: 10.1101/gr.268581.120.

20. Cortal A, Martignetti L, Six E, and Rausell A. Gene signature extraction and cell identity recognition at the single-cell level with Cell-ID. Nature biotechnology 2021; 39:1095–102. DOI: 10.1038/s41587-021-00896-6

21. Oetjen KA, Lindblad KE, Goswami M, Gui G, Dagur PK, Lai C, Dillon LW, McCoy JP, and Hourigan CS. Human bone marrow assessment by single-cell RNA sequencing, mass cytometry, and flow cytometry. JCI insight 2018; 3. DOI: 10.1172/jci.insight.124928

22. Freytag S, Tian L, Lönnstedt I, Ng M, and Bahlo M. Comparison of clustering tools in R for medium-sized 10x Genomics single-cell RNA-sequencing data. F1000Research 2018; 7:1297. DOI: 10.12688/f1000research.15809.2

23. Sun Z, Chen L, Xin H, Jiang Y, Huang Q, Cillo AR, Tabib T, Kolls JK, Bruno TC, Lafyatis R, et al. A Bayesian mixture model for clustering droplet-based single-cell transcriptomic data from population studies. Nature communications 2019; 10:1649. DOI: 10.1038/s41467-019-09639-3

24. Muto Y, Wilson PC, Ledru N, Wu H, Dimke H, Waikar SS, and Humphreys BD. Single cell transcriptional and chromatin accessibility profiling redefine cellular heterogeneity in the adult human kidney. Nature communications 2021; 12:2190. DOI: 10.1038/s41467-021-22368-w

25. Baron M, Veres A, Wolock SL, Faust AL, Gaujoux R, Vetere A, Ryu JH, Wagner BK, Shen-Orr SS, Klein AM, et al. A single-cell transcriptomic map of the human and mouse pancreas reveals inter-and intra-cell population structure. Cell systems 2016; 3:346–60. DOI: 10.1016/j.cels.2016.08.011

26. MacParland SA, Liu JC, Ma XZ, Innes BT, Bartczak AM, Gage BK, Manuel J, Khuu N, Echeverri J, Linares I, et al. Single cell RNA sequencing of human liver reveals distinct intrahepatic macrophage populations. Nature communications 2018; 9:4383. DOI: 10.1038/s41467-018-06318-7

27. Segerstolpe Å, Palasantza A, Eliasson P, Andersson EM, Andréasson AC, Sun X, Picelli S, Sabirsh A, Clausen M, Bjursell MK, et al. Single-cell transcriptome profiling of human pancreatic islets in health and type 2 diabetes. Cell metabolism 2016; 24:593–607. DOI: 10.1016/j.cmet.2016.08.020

28. Zheng GX, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, Ziraldo SB, Wheeler TD, McDermott GP, Zhu J, et al. Massively parallel digital transcriptional profiling of single cells. Nature communications 2017; 8:14049. DOI: 10.1038/ncomms14049

29. Du J, Jia P, Dai Y, Tao C, Zhao Z, and Zhi D. Gene2vec: distributed representation of genes based on co-expression. BMC genomics 2019; 20:7–15. DOI: 10.1186/s12864-018-5370-x

30. Heumos L, Schaar AC, Lance C, Litinetskaya A, Drost F, Zappia L, Lücken MD, Strobl DC, Henao J, Curion F, et al. Best practices for single-cell analysis across modalities. Nature Reviews Genetics 2023 :1–23. DOI: 10.1038/s41576-023-00586-w

31. Ahlmann-Eltze C and Huber W. Comparison of transformations for single-cell RNA-seq data. Nature Methods 2023 :1–8. DOI: 10.1038/s41592-023-01814-1

32. Ianevski A, Giri AK, and Aittokallio T. Fully-automated and ultra-fast cell-type identification using specific marker combinations from single-cell transcriptomic data. Nature communications 2022; 13:1246. DOI: 10.1038/s41467-022-28803-w

33. Wolf FA, Angerer P, and Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome biology 2018; 19:1–5. DOI: 10.1186/s13059-017-1382-0

34. Jaiswal A, Babu AR, Zadeh MZ, Banerjee D, and Makedon F. A survey on contrastive self-supervised learning. Technologies 2020; 9:2. DOI: 10.3390/technologies9010002

35. Chen T, Kornblith S, Norouzi M, and Hinton G. A simple framework for contrastive learning of visual representations. *International conference on machine learning*. PMLR. 2020 :1597–607. DOI: 10.48550/arXiv.2002.05709

36. Misra I and Maaten Lvd. Self-supervised learning of pretext-invariant representations. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020 :6707–17. DOI: 10.48550/arXiv.1912.01991

37. He K, Fan H, Wu Y, Xie S, and Girshick R. Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020 :9729–38. DOI: 10.48550/arXiv.1911.05722

38. Caron M, Misra I, Mairal J, Goyal P, Bojanowski P, and Joulin A. Unsupervised learning of visual features by contrasting cluster assignments. Advances in neural information processing systems 2020; 33:9912–24. DOI: 10.48550/arXiv.2006.09882

39. Sohn K. Improved deep metric learning with multi-class n-pair loss objective. Advances in neural information processing systems 2016; 29. Available from: https://proceedings.neurips.cc/paper_files/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf

40. Frosst N, Papernot N, and Hinton G. Analyzing and improving representations with the soft nearest neighbor loss. *International conference on machine learning*. PMLR. 2019 :2012–20. DOI: 10.48550/arXiv.1902.01889

41. Ba JL, Kiros JR, and Hinton GE. Layer normalization. arXiv preprint arXiv:1607.06450 2016. DOI: 10.48550/arXiv.1607.06450

42. Gunawan I, Vafaee F, Meijering E, and Lock JG. An introduction to representation learning for single-cell data analysis. Cell Reports Methods 2023; 3. DOI: 10.1016/j.crmeth.2023.100547

43. Li X, Wang K, Lyu Y, Pan H, Zhang J, Stambolian D, Susztak K, Reilly MP, Hu G, and Li M. Deep learning enables accurate clustering with batch effect removal in single-cell RNA-seq analysis. Nature communications 2020; 11:2338. DOI: 10.1038/s41467-020-15851-3

44. Amodio M, Van Dijk D, Srinivasan K, Chen WS, Mohsen H, Moon KR, Campbell A, Zhao Y, Wang X, Venkataswamy M, et al. Exploring single-cell data with deep multitasking neural networks. Nature methods 2019; 16:1139–45. DOI: 10.1038/s41592-019-0576-7

45. Xu Z, Luo J, and Xiong Z. scSemiGAN: a single-cell semi-supervised annotation and dimensionality reduction framework based on generative adversarial network. Bioinformatics 2022; 38:5042–8. DOI: 10.1093/bioinformatics/btac652

46. Deng Y, Bao F, Dai Q, Wu LF, and Altschuler SJ. Scalable analysis of cell-type composition from single-cell transcriptomics using deep recurrent learning. Nature methods 2019; 16:311–4. DOI: 10.1038/s41592-019-0353-7

47. Han W, Cheng Y, Chen J, Zhong H, Hu Z, Chen S, Zong L, Hong L, Chan TF, King I, et al. Self-supervised contrastive learning for integrative single cell RNA-seq data analysis. Briefings in Bioinformatics 2022; 23:bbac377. DOI: 10.1093/bib/bbac377

48. Xu Y, Das P, and McCord RP. SMILE: mutual information learning for integration of single-cell omics data. Bioinformatics 2022; 38:476–86. DOI: 10.1093/bioinformatics/btab706

49. Tirosh I, Izar B, Prakadan SM, Wadsworth MH, Treacy D, Trombetta JJ, Rotem A, Rodman C, Lian C, Murphy G, et al. Dissecting the multicellular ecosystem of metastatic melanoma by single-cell RNA-seq. Science 2016; 352:189–96. DOI: 10.1126/science.aad0501

50. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, and Chintala S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019 :8024–35. Available from: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

51. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, and Duchesnay E. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 2011; 12:2825–30

52. Akiba T, Sano S, Yanase T, Ohta T, and Koyama M. Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019 :2623–31. DOI: 10.1145/3292500.3330701

53. Way GP, Kost-Alimova M, Shibue T, Harrington WF, Gill S, Piccioni F, Becker T, Shafqat-Abbasi H, Hahn WC, Carpenter AE, et al. Predicting cell health phenotypes using image-based morphology profiling. Molecular biology of the cell 2021; 32:995–1005. DOI: 10.1091/mbc.E20-12-0784

54. Caicedo JC, Cooper S, Heigwer F, Warchal S, Qiu P, Molnar C, Vasilevich AS, Barry JD, Bansal HS, Kraus O, et al. Data-analysis strategies for image-based cell profiling. Nature methods 2017; 14:849–63. DOI: 10.1038/nmeth.4397

A

Appendix 1

A.1 Preprocessing of data

A.1.1 Pancreas dataset

UMAP of the pancreas dataset shows all 9 cell types (Fig. A.1a) and 4 patient IDs / sequencing methods (Fig. A.1b). It is worth noting that the batch effects are very severe for this dataset.

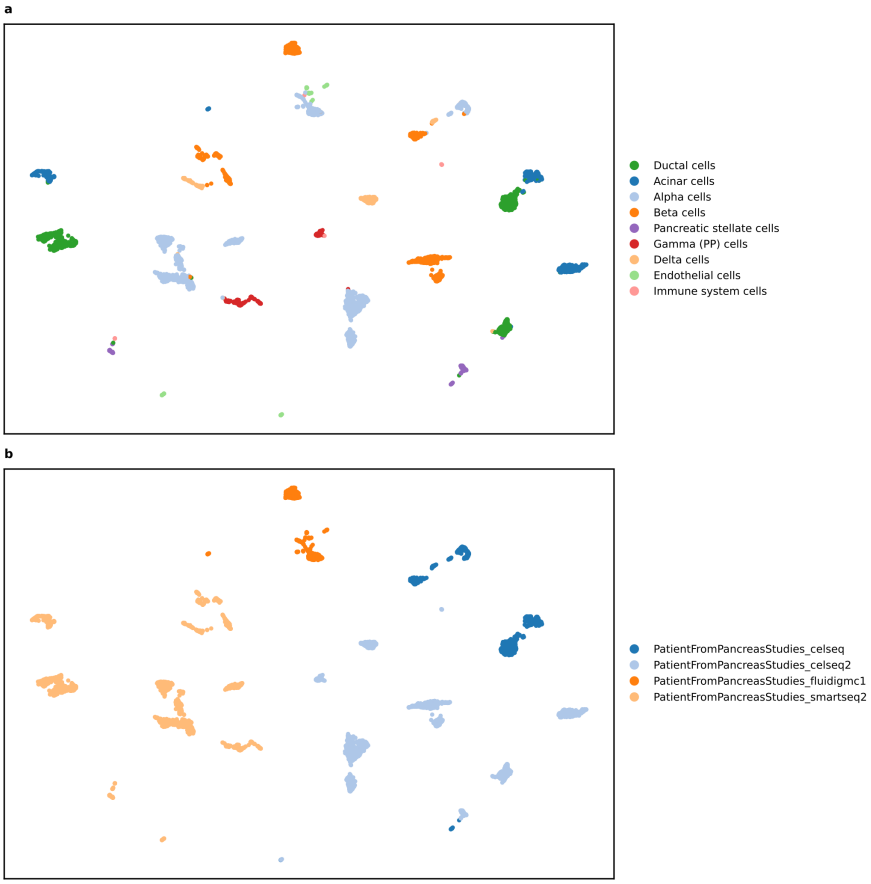


Figure A.1: a UMAP visualization of cell type clusters in the pancreas dataset. **b** UMAP highlighting batch effect caused by patient ID / sequencing method.

A.1.2 Kidney dataset

UMAP of the kidney dataset shows all 15 cell types (Fig. A.2a) and 3 patient IDs (Fig. A.2b). Once again there are visible batch effects.

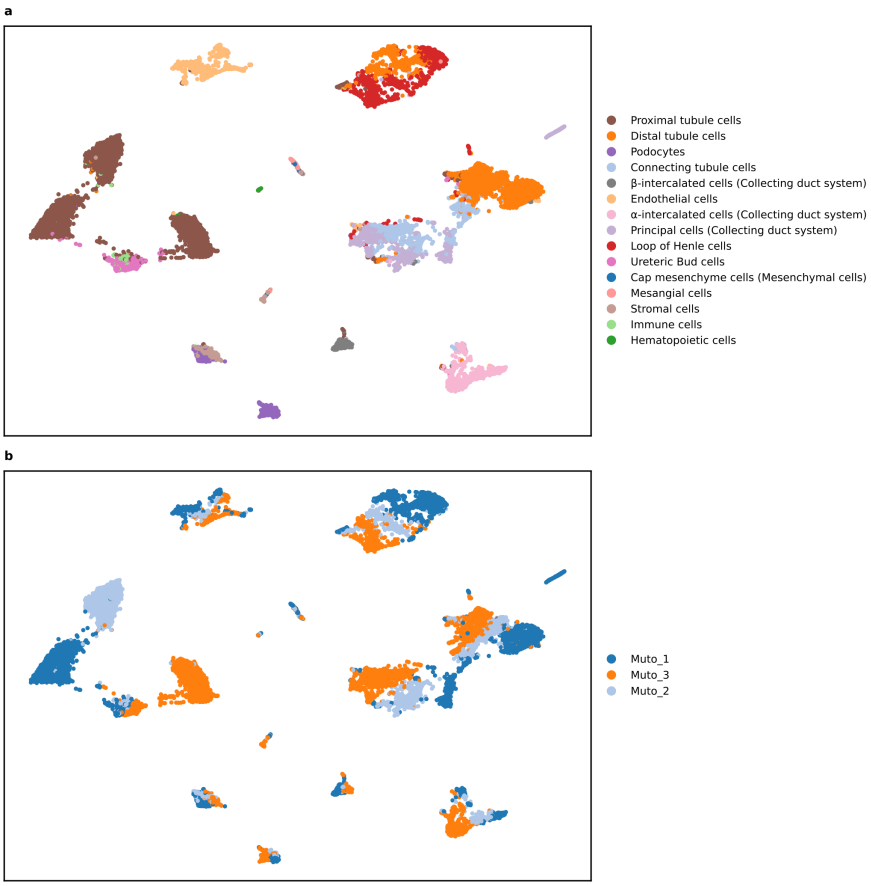


Figure A.2: **a** UMAP visualization of cell type clusters in the kidney dataset. **b** UMAP highlighting batch effect cause by patient ID.

A.1.3 Merged dataset

The entire merged dataset was visualized via UMAP and colored according to all 46 cell types (Fig. A.3a) and 18 patient IDs (Fig. A.3b).

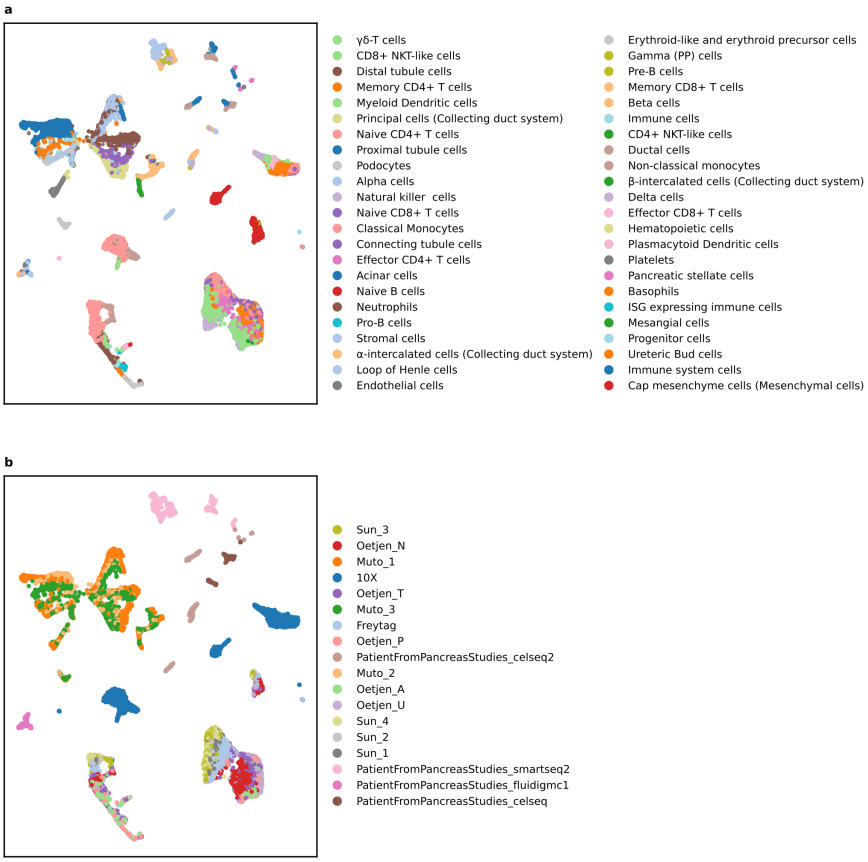


Figure A.3: **a** UMAP visualization of cell type clusters in the full merged dataset. **b** UMAP highlighting batch effect caused by patient ID.

A.1.4 Baron dataset

UMAP of the Baron dataset shows all 14 cell types (Fig. A.4a) and 4 patient IDs (Fig. A.4b). There are visible batch effects.

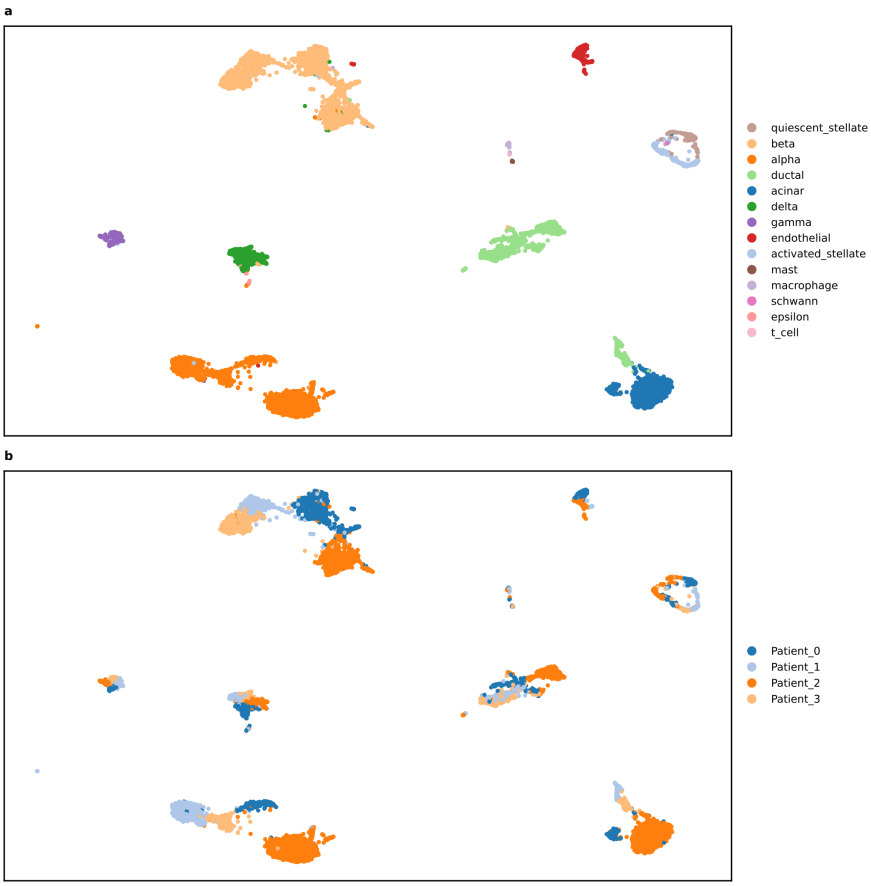


Figure A.4: **a** UMAP visualization of cell type clusters in the Baron dataset. **b** UMAP highlighting batch effect cause by patient ID.

A.1.5 MacParland dataset

UMAP of the MacParland dataset shows all 20 cell types (Fig. A.5a) and 5 patient IDs (Fig. A.5b). There are visible batch effects.

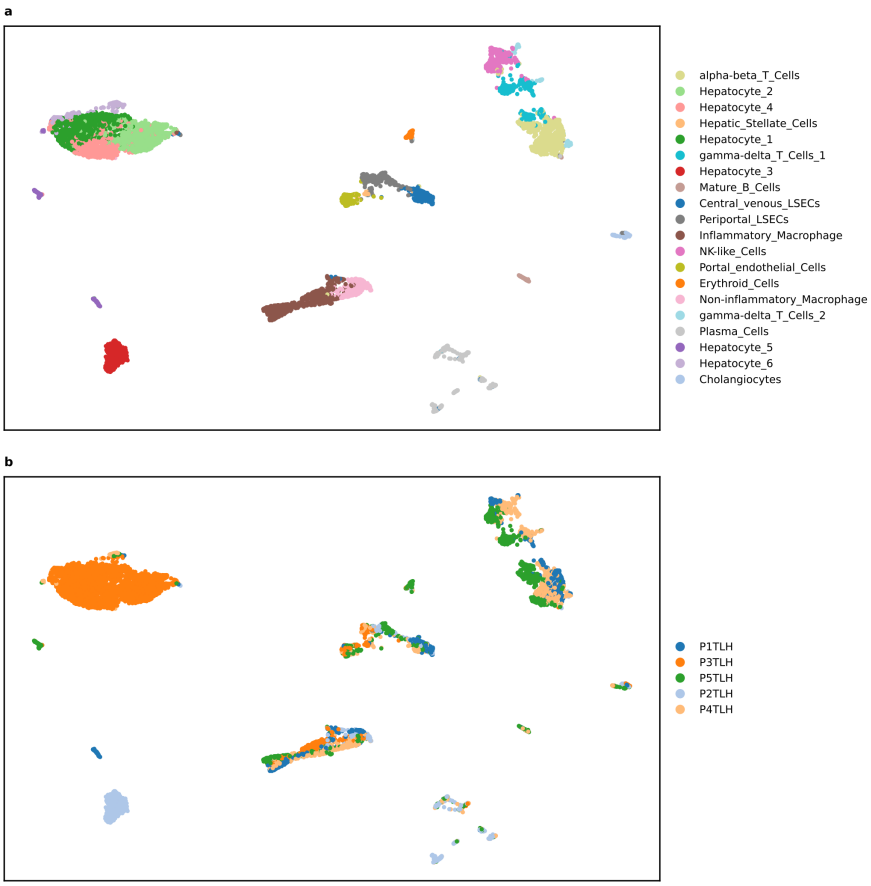


Figure A.5: **a** UMAP visualization of cell type clusters in the MacParland dataset. **b** UMAP highlighting batch effect cause by patient ID.

A.1.6 Segerstolpe dataset

UMAP of the Segerstolpe dataset shows all 15 cell types (Fig. A.6a) and 10 patient IDs (Fig. A.6b). There are visible batch effects in the dataset.

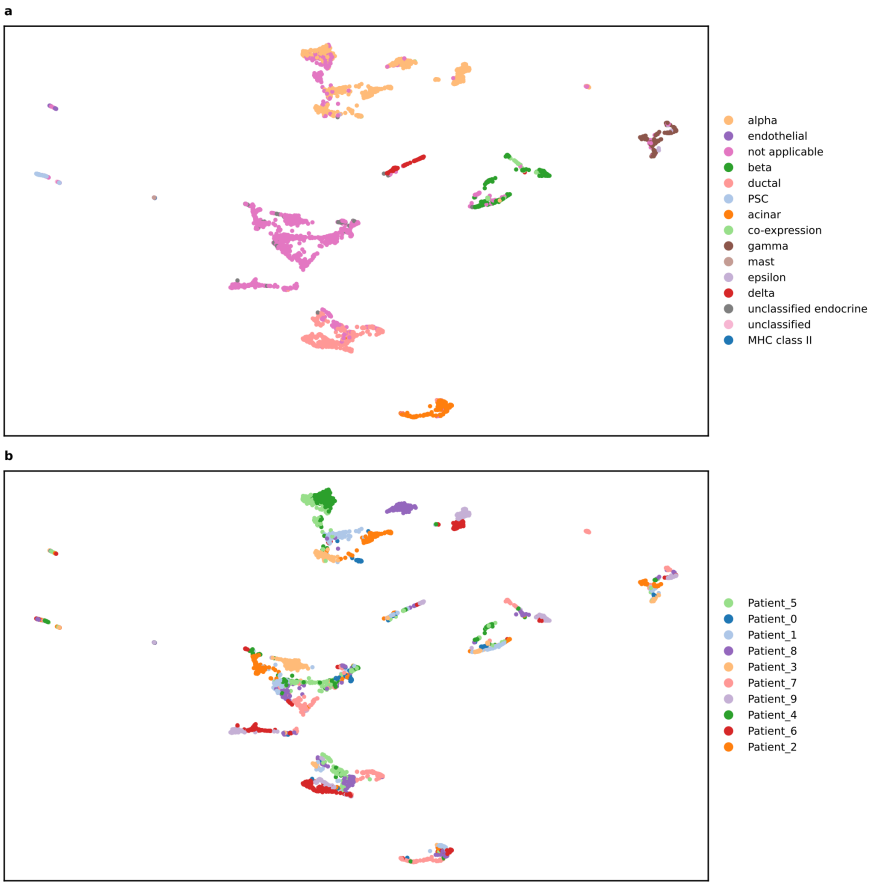


Figure A.6: **a** UMAP visualization of cell type clusters in the Segerstolpe dataset. **b** UMAP highlighting batch effect cause by patient ID.

A.1.7 Zheng68k dataset

UMAP of the Zheng68k dataset shows all 11 cell types (Fig. A.7a) and 8 unique barcode IDs (Fig. A.7b). There are visible batch effects in the dataset.

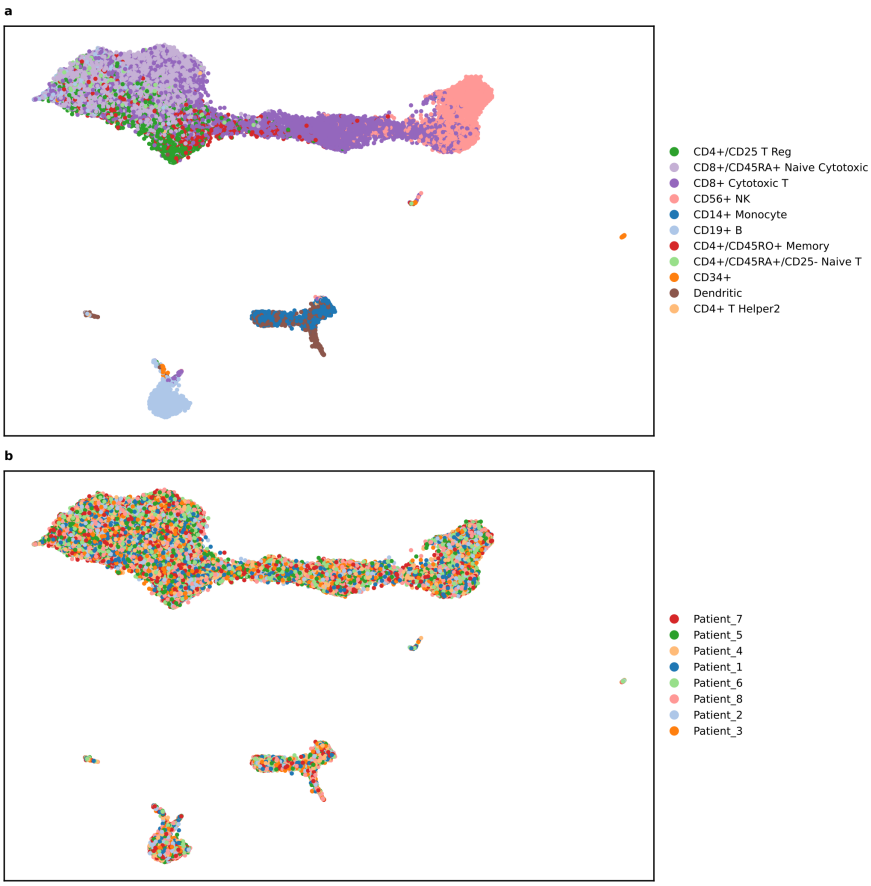


Figure A.7: **a** UMAP visualization of cell type clusters in the Zheng68k dataset. **b** UMAP highlighting batch effect cause by patient ID.

A.2 Benchmark on generalizable scRNA-Seq embedding space

A.2.1 Bone marrow dataset

All 12 metrics, including overall bio, overall batch, and overall scores for all models when trained on the bone marrow dataset.

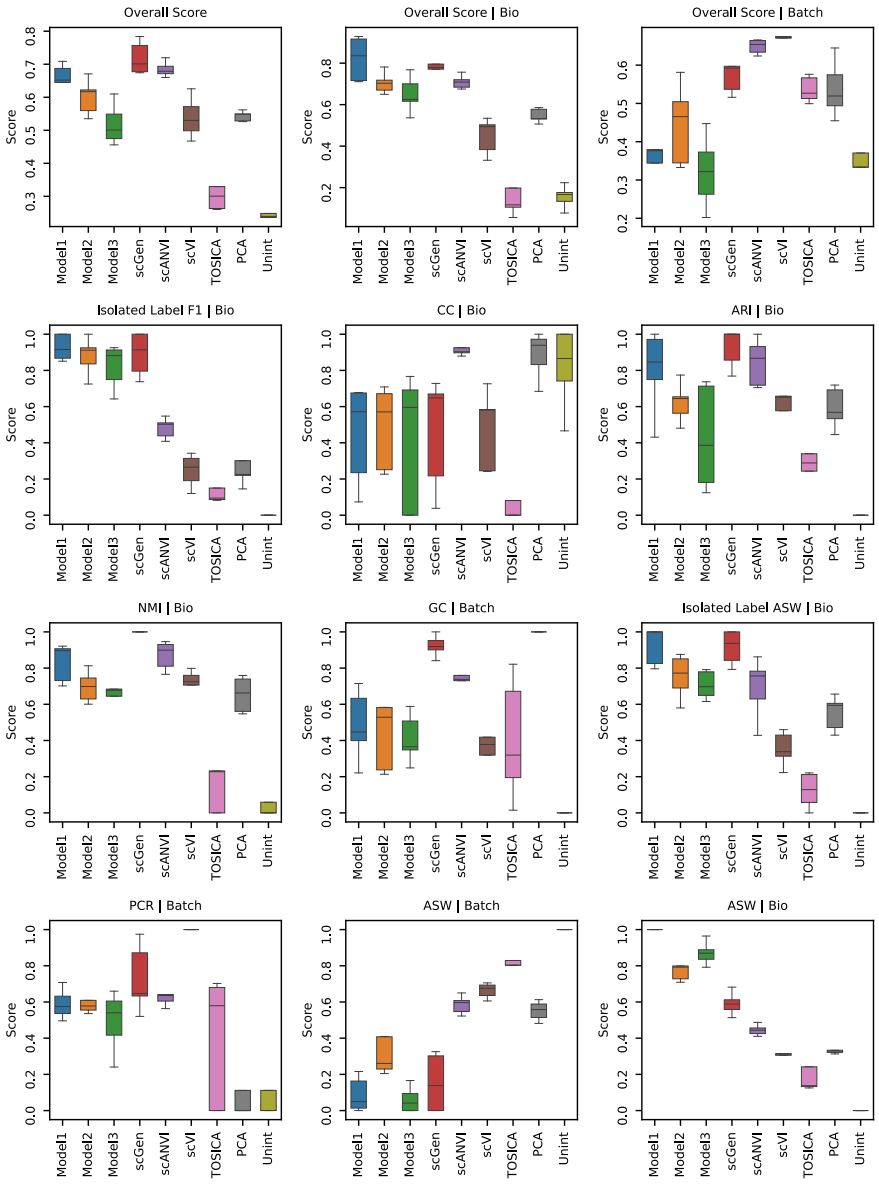


Figure A.8: Five-fold cross-testing on the bone marrow dataset, where all metrics are calculated. The following models are used in the benchmark: Unint (Unintegrated), PCA, TOSICA [6], scVI [16], scANVI [13], scGen [14], Model 1, Model 2, and Model 3. The input to the methods are the top 2000 HVGs. Box plots show the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range.

A.2.2 PBMC dataset

All 12 metrics, including overall bio, overall batch, and overall scores for all models when trained on the PBMC dataset.

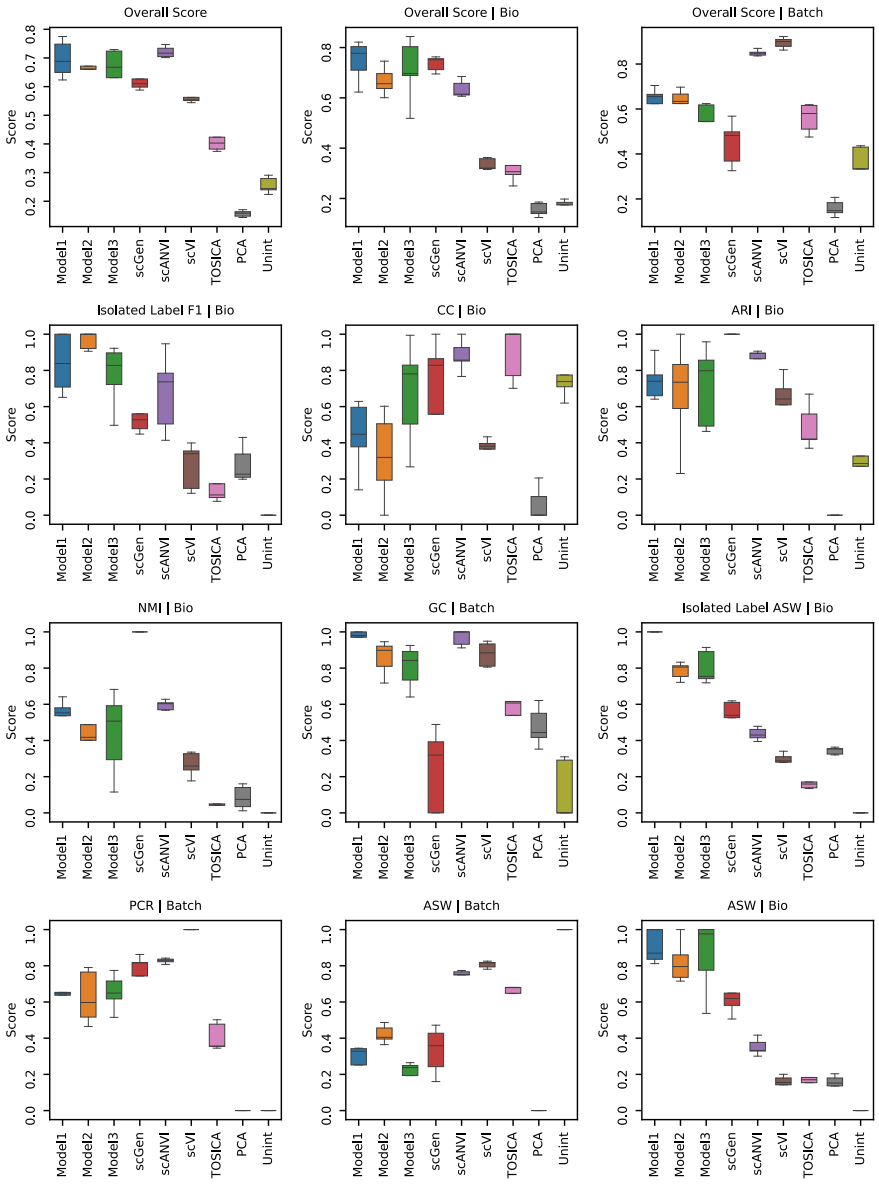


Figure A.9: Five-fold cross-testing on the PBMC dataset, where all metrics are calculated. The following models are used in the benchmark: Unint (Unintegrated), PCA, TOSICA [6], scVI [16], scANVI [13], scGen [14], Model 1, Model 2, and Model 3. The input to the methods are the top 2000 HVGs. Box plots shows the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range.

A.2.3 Kidney dataset

All 12 metrics, including overall bio, overall batch, and overall scores for all models when trained on the kidney dataset.

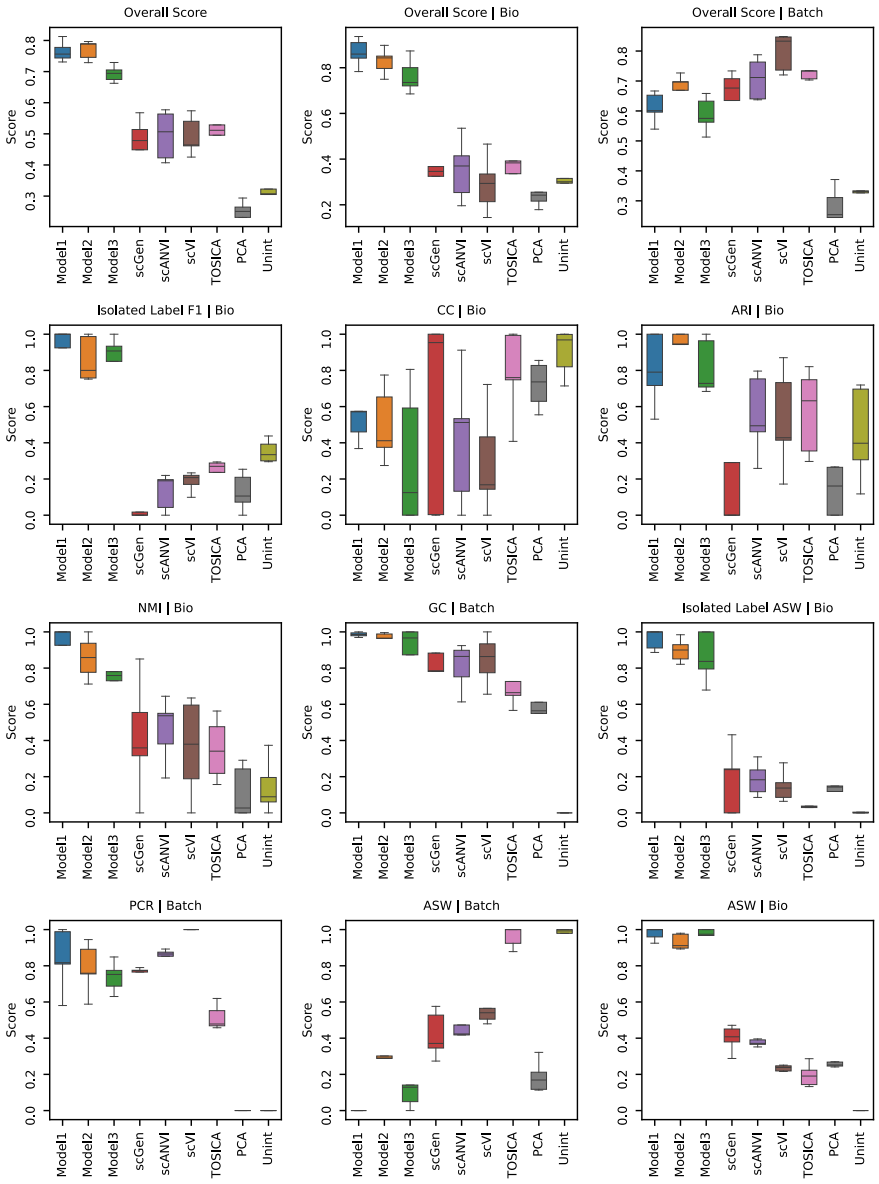


Figure A.10: Five-fold cross-testing on the kidney dataset, where all metrics are calculated. The following models are used in the benchmark: Unint (Unintegrated), PCA, TOSICA [6], scVI [16], scANVI [13], scGen [14], Model 1, Model 2, and Model 3. The input to the methods are the top 2000 HVGs. Box plots shows the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range.

A.2.4 Pancreas dataset

All 12 metrics, including overall bio, overall batch, and overall scores for all models when trained on the pancreas dataset.

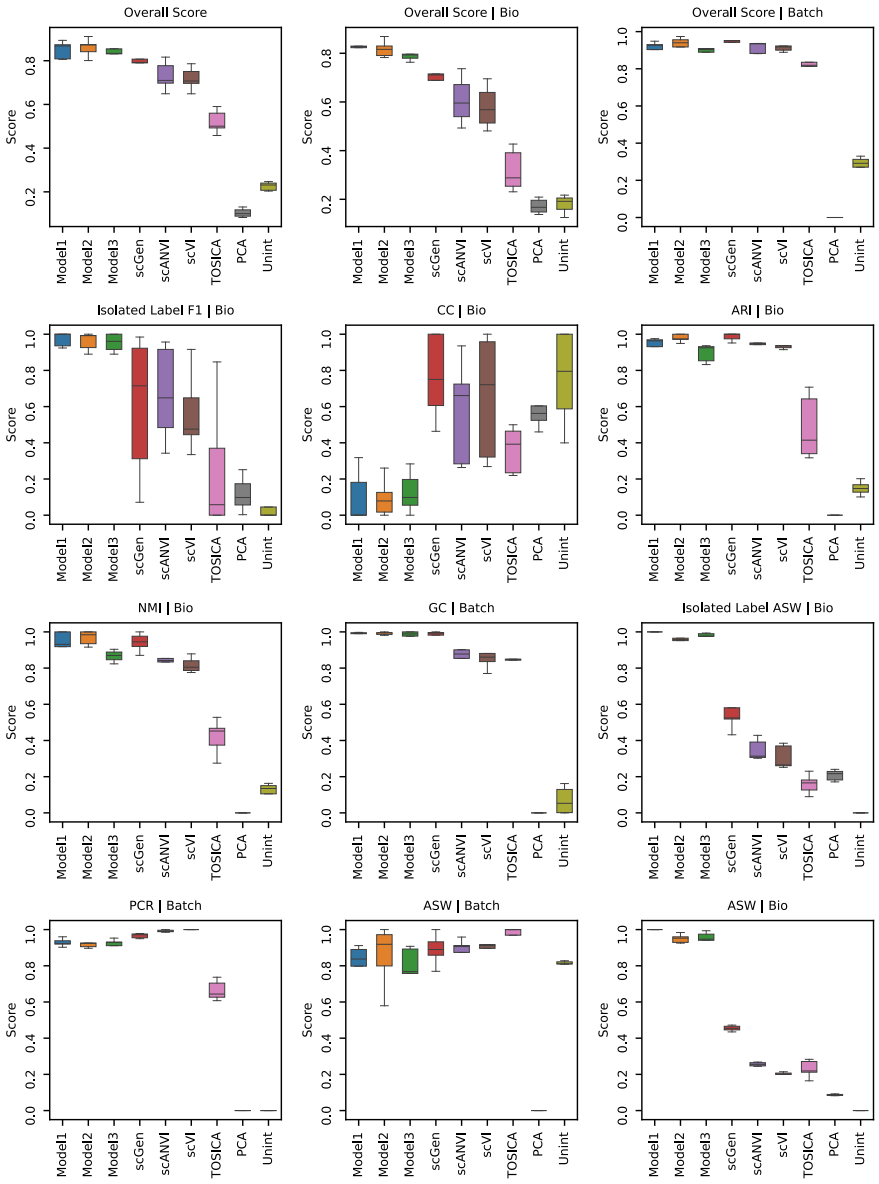


Figure A.11: Five-fold cross-testing on the pancreas dataset, where all metrics are calculated. The following models are used in the benchmark: Unint (Unintegrated), PCA, TOSICA [6], scVI [16], scANVI [13], scGen [14], Model 1, Model 2, and Model 3. The input to the methods are the top 2000 HVGs. Box plots shows the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range.

A.2.5 Merged dataset

All 12 metrics, including overall bio, overall batch, and overall scores for all models when trained on the merged dataset.

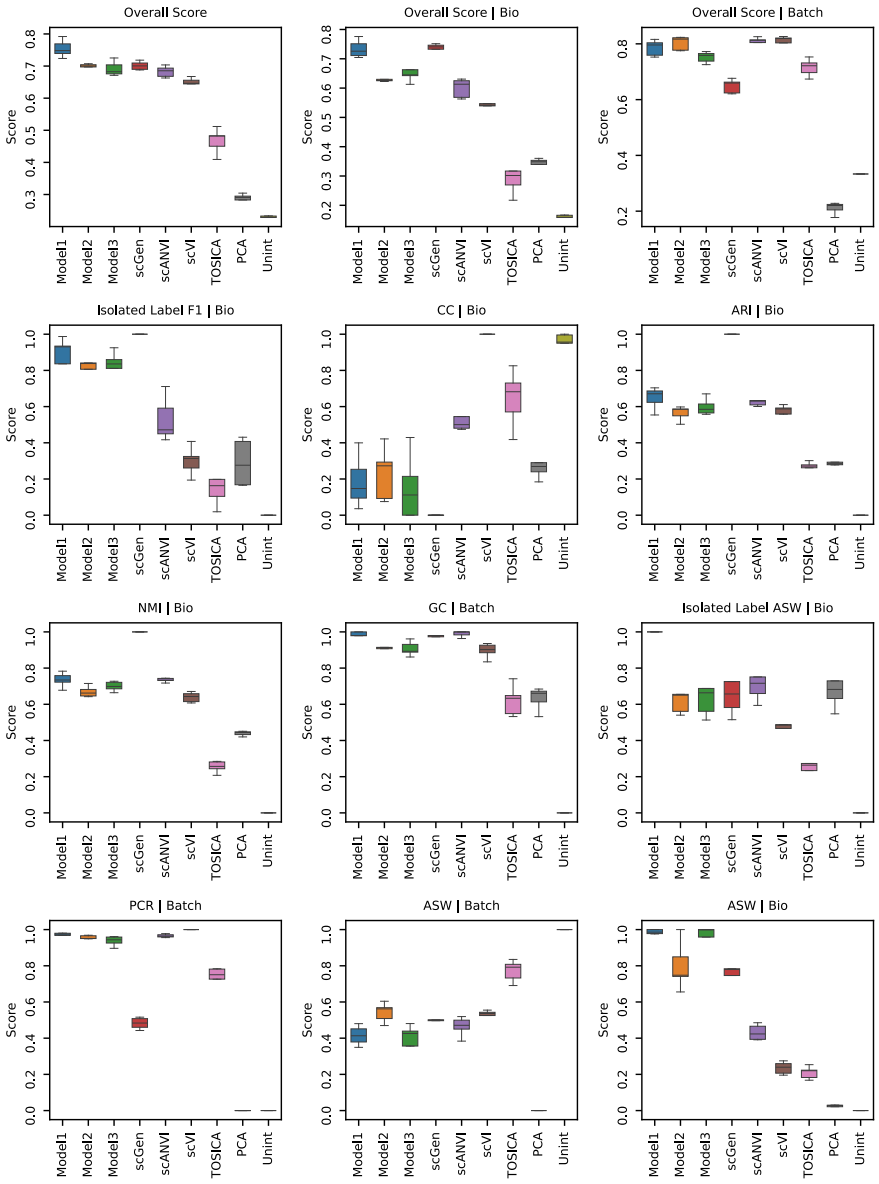


Figure A.12: Five-fold cross-testing on the merged dataset, where all metrics are calculated. The following models are used in the benchmark: Unint (Unintegrated), PCA, TOSICA [6], scVI [16], scANVI [13], scGen [14], Model 1, Model 2, and Model 3. The input to the methods are the top 2000 HVGs. Box plots shows the median displayed as the center line, the interquartile range as the box hinges, and the whiskers extending up to 1.5 times the interquartile range.

A.3 Attributions

The filter and scale icon from Figure 2.1 were taken from `https://flaticon.com`

DEPARTMENT OF LIFE SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY