Contrastive Learning for Single-Cell Classification

A Thesis Presented in Partial Fulfillment of
the Honors Bachelor's Degree

William Howard-Snyder

Approved: ______________________________

Computer Science and Engineering
Under the supervision of Sheng Wang
University of Washington
June 2022

Contents

List of Figures

List of Tables

Abstract

Recent breakthroughs in single-cell RNA sequencing technologies have improved our understanding of the molecular basis for cellular heterogeneity and dynamics. A crucial step in single-cell data analysis is to identify which cells correspond to which cell types. Traditionally, cell classification has been performed manually by experts, which is expensive and slow. In recent years, deep learning models have been proposed to automatically classify individual cells. However, many of these models yield little to no performance gains compared to simpler and more interpretable models such as logistic regression and support vector machines. In this work, we present a model-agnostic component that uses contrastive learning to improve internal model representations in deep neural network architectures for single-cell classification. We demonstrate that our contrastive component improves classification accuracy and significantly improves AUPRC of logistic regression, especially for rare cell types.

Acknowledgements

This thesis would not have been possible without the support of my friends, family, and mentors. First I'd like to thank my research advisor Professor Sheng Wang for taking the time to meet with me one-on-one and inspiring me to explore the field of computational biology.

I'd also like to thank Addie Chambers for the time she invested in meeting with me weekly, providing technical advice on the specifics of my thesis, giving tips on how to write more clearly, and offering perspective on what it takes to be a PhD student.

Thank you to the rest of the Wang Group for feedback on my senior thesis and allowing me to use your valuable time to present my work.

Thank you to my mentor Neo Wu and the rest of my research cohort at Google Computer Science Research Mentorship Program for providing valuable advice on how to conduct literature reviews, engage with supervisors, and navigate application processes.

Thank you to my girlfriend Hannah Feller for your unwavering emotional support. I would not have been able to complete this work without our frequent FaceTimes.

Thank you to Mom, Dad, and Peter for supporting me throughout my life and encouraging me to be curious, persistent, and compassionate.

1 Introduction

The cell represents a fundamental building block of all life on Earth. In multi-cellular organisms, such as humans, cells sharing the same "operating manual" DNA differentiate themselves to behave differently, taking on different functions to perform different tasks and form the various tissues that make up the human body. Deeper understanding at the cellular level has substantial biological and medical benefits, including insights into disease, development, aging, and much more. In recent years, single-cell RNA sequencing (scRNA-seq) has enabled this new and exciting line of research.

Accurate cell-type annotation for scRNA-seq data remains a great challenge in biology [3]. As the resolution and size of single-cell datasets grow, increasingly sophisticated models and techniques are required to mine biologically meaningful information from these data. This motivates the use of deep neural networks (DNNs), which are capable of modeling arbitrarily complex relationships between input features and output variables, and excel in settings where there are huge datasets of structured high dimensional data. Many DNN architectures have been proposed for single-cell annotation [4, 5, 6]; however, these models yield small performance gains compared to simpler and more interpretable models such as logistic regression and support vector machines [7].

Contrastive parallel, a parallel line of work, is a framework for representation learning, has seen lots of recent success in fields like Natural Language Processing (NLP) and Computer Vision (CV) [8, 9, 10, 11, 12, 13, 14]. These models learn high-quality, low-dimensional representations of text and image data that are robust to noise and have been shown to improve performance in downstream tasks such as image classification [8, 11, 9], sentiment analysis [14, 15, 16], and information retrieval [17, 18] in both supervised and unsupervised settings.

Inspired by this success, we propose a model-agnostic component that uses contrastive learning to improve internal model representations in DNN architectures for single-cell classification. In this work, our main contributions are:

- **Model-agnostic contrastive component**: We define a novel component that can be included in DNN single-cell classification models. This component is simple and easy to integrate into the inference and training stages of pre-existing classifiers.
- **Benchmark evaluation**: We perform experiments on the *Tabular Microcebus* [2] scRNA-seq dataset using three base models: ACTINN [4], scDAE [5], and logistic regression. We find that our contrastive component improves classification accuracy and significantly improves AUPRC of logistic regression, especially for rare cell types.

The rest of this work is structured as follows: In Section 2 we describe relevant background for our problem, such as single-cell data and representation learning. In Section 3 we describe related works, including baseline models for single-cell classification. In Section 4 we define our problem setting and model. In Section 5 we describe the datasets and experimental procedure we used to evaluate our model, as well as the results. Finally, in Section 6, we wrap up the work by summarizing the results, and discussing some potential lines for future work.

2 Background

This thesis covers topics in machine learning and biology. In this section I will provide a summary of the relevant information for this work, including Contrastive Learning, Single Cell Data, and Single Cell Classification.

2.1 Representation Learning

Representation learning consists of a set of techniques that enable a system to automatically discover data features that improve performance on downstream tasks (e.g., information retrieval, classification, visualization, etc.). Usually, these representations are in the form of a vector of real numbers that is lower-dimensional than the original datum, and that captures the semantic similarity between individual data points. There are many approaches to representation learning. Some examples of representation learning are unsupervised, such as PCA, t-SNE [19], and UMAP [20]. These methods learn features by modelling the structure present in the raw data. Other methods are supervised, such as those in metric learning [21]. These methods utilize labels to learn representations that reflect information present in the labels as well as the data itself (e.g., datapoints with the same label have similar representations). Some methods, such as self-supervised representation learning methods, are a hybrid of the two. These methods do not use curated labels, rather they treat part of the training data itself as a label and learn representations as if they were supervised. One example is BERT [14], which learns text representations by predicting removed words from a sentence.

Contrastive Learning is a discriminative form of representation learning that has seen a lot of recent success in supervised and semi-supervised representation learning for image and text data [8, 9, 11, 13]. There are many formulations of contrastive learning, but the goal remains the same in each: Learn an embedding function such that objects that represent the same thing are mapped together, and objects that represent different things are mapped far apart. The key task is to define what it means for two objects to represent the same/different things.

In self-supervised representation learning, models define transformations that they want their representations to be invariant to, such as crop and blur for images [9, 12]. An object $\mathbf{x}_i$ is defined as the same thing as its transformed version $\tilde{\mathbf{x}}_i$, while a different object $\mathbf{x}_j$ (e.g., a different image), is considered a different thing. We call $(\mathbf{x}_i, \tilde{\mathbf{x}}_i)$ a positive pair, and $(\mathbf{x}_i, \mathbf{x}_j)$ a negative pair. The set of all objects that form a positive pair with $\mathbf{x}_i$ is $\mathcal{P}(i)$, and similarly, the set of all objects that form a negative pair with $\mathbf{x}_i$ is $\mathcal{N}(i)$. Often, $\mathcal{N}(i)$ will be much larger than $\mathcal{P}(i)$. In self-supervised contrastive learning, there is no way to determine whether two objects actually represent the same concept for downstream tasks of interest. Since there are no labels, the only available knowledge is whether one object is a transformation of another. For example, there may be two distinct pictures of the same dog in a dataset; however, those images will be treated as a negative pair. Downstream performance on classification tasks can suffer because the representations for a single class may require complicated decision boundaries to separate in the learned representation space.

Supervised contrastive learning attempts to overcome this problem by using labels to define positive and negative pairs [11]. Typically, two objects $\mathbf{x}_i$ and $\mathbf{x}_j$ are a positive pair in this framework if they share a label, and otherwise they're a negative pair. Defining positive and negative pairs in this way forces the the learned embedding to group representations by label, making the representations easier to separate by class.

Their are many formulations of contrastive learning optimization objectives, but they all follow a similar pattern. Formally, given a set of datapoints $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$, learn $f : \mathbb{R}^m \to \mathbb{R}^d$, $d << m$ by minimizing the contrastive loss

$$\mathcal{L}_{CL} \propto \sum_{\mathbf{x}\in\mathcal{D}} \sum_{\mathbf{x}^+\in\mathcal{P}(\mathbf{x})} \sum_{\mathbf{x}^-\in\mathcal{N}(\mathbf{x})} \mathrm{dist}(f(\mathbf{x}), f(\mathbf{x}^+)) - \mathrm{dist}(f(\mathbf{x}), f(\mathbf{x}^-)), \tag{1}$$

where $\mathcal{P}(\mathbf{x})$ and $\mathcal{N}(\mathbf{x})$ are the positive and negative set for an *anchor point* $\mathbf{x}$, respectively, and $\mathrm{dist}(\cdot,\cdot)$ is function that measures the distance between its inputs (e.g., cosine distance or euclidean distance). The use of triples (i.e., anchor point, positive example, and negative example) in this loss is reminiscent of triplet loss, which was used for face recognition in [22]. However, more recent work compares many positive and negative examples at a time [10, 23]. We will discuss one such method in depth in Section 3.2.

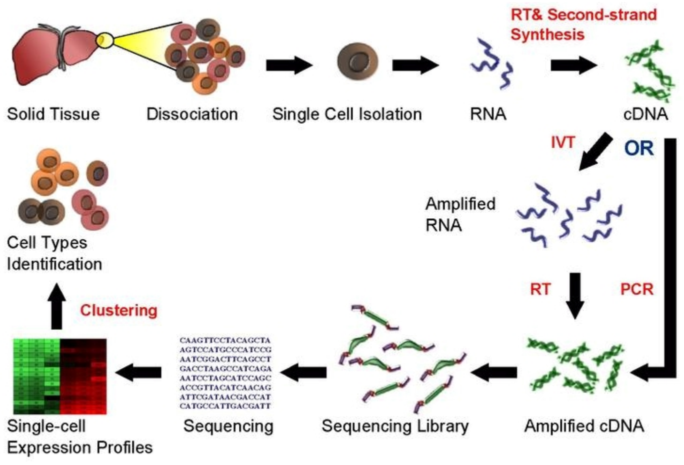

Figure 1: General scRNA-seq workflow [1]

2.2 Single Cell Data

Recent technological breakthroughs in single-cell sequencing allow an unprecedented amount of single-cell data to be generated. These data have the potential to provide biological insights such as the discovery of novel cell subtypes and exploration the cellular basis for disease [24]. However, the size, sparsity, and noisiness of gene expression data make it challenging to focus on the underlying biological signal. Thus, the development of efficient methods for mining these data plays an important role in progressing modern biomedical research. In this section I will describe how noise and sparsity enter single-cell data.

2.2.1 Acquiring the Data

Each cell contains millions of genes, which carry instructions for making proteins that can effect cellular function. Only a small portion of these genes are expressed, and the degree to which they're expressed varies across cells and over time. Measuring the expression of genes at single-cell resolution is critical for understanding the heterogeneity of cells and offers exciting biological and medical insights [25, 26]. Single-cell RNA sequencing (scRNA-seq) measures the gene expression of individual cells. The first method was introduced in 2009 [27], but it wasn't until 4 years later that a new more cost-efficient technique was developed, making scRNA-seq more broadly accessible.

There are a variety of methods for generating single-cell data from a biological sample, but four key steps appear in each [28].

1. The first is *disassociation*. In this step, the biological tissue sample is digested and broken down so that individual cells can be extracted.

2. Second is *isolation*. This step differs depending on the scRNA-seq method. Plate-based methods isolate cells into wells on a plate, while droplet-based methods suspend cells in microfluidic droplets. Regardless of the method, errors can occur when multiple cells are captured together (referred to as a doublet), or no viable cells are captured. When exactly one cell is captured, that is the ideal case.

3. Third is *library construction*, which refers to gathering the mRNA for each individual cell. This involves breaking down the membrane of captured cells, reverse transcribing their mRNA to cDNA, and amplifying the mRNA. Additionally, each cDNA library is labelled with a unique barcode that demarcates individual cells. However, note that the mRNA of two cells might be given the same barcode when a doublet occurs.

4. Fourth, these libraries are pooled together and *sequenced*, which produces a string of nucleotides for each mRNA molecule. These strings are grouped by their barcodes and subjected to quality control measures (e.g., removal of low-quality cells).

The resulting dataset for a tissue sample is a matrix of counts that has dimensions barcodes by number-of transcripts. We say *barcodes* rather than *number of cells* because in the *isolation* step a barcode may mistakenly be tagged to multiple cells or not be be tagged to any cells.

2.2.2 Data Details

The full scRNA-seq workflow produces a gene expression matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, for n the number of barcodes and m the number of genes, where each row $\mathbf{x}_i \in \mathbb{R}^m$ is a gene expression profile. For our purposes, we can think of a gene expression profile as a vector of features that represent an individual cell, where each coordinate of that vector contains the measurement of a single gene's activity for that particular cell. This provides a snapshot of global cellular function at the time the cell was sequenced. Understanding the data as capturing a biologically-relevant signal is similar to how data are viewed in fields like NLP and CV. However, what makes single-cell data uniquely challenging, compared to image or text data, are their noise and sparsity.

Single-cell data are notoriously noisy. One source of noise in single cell data is batch effects. As described in Section 2.2.1, gene expression profiles are generated through complex wet lab procedures. Batch effects occur when the transcriptome measurements for a batch of data is impacted by environmental conditions of the experiment specific to the experiment. Each experiment uses different machines, different procedures, and different technicians, resulting in significant variation in transcriptome measurement across experiments that is irrelevant to the variable of interest (e.g., cell type) and can confound downstream analysis. Another source of noise comes from the inclusion of low-quality cells. This can happen in at least two ways. Multiple cells can be captured together and treated as a single cell, or captured cells may include those whose membrane broke prematurely. For example, if a barcode has few genes and a high mitochondrial count, then it is likely the profile of a cell whose cytoplasmic mRNA has leaked out through a broken membrane [28]. In either case, low-quality cells are typically omitted from the scRNA-seq dataset for downstream application via outlier detection, but sometimes they are mistaken for healthy single cells and make it into the final gene expression matrix. When this occurs, those gene expression profiles are nonsensical and can hinder the extraction of biological insight from the data.

Sparsity is another challenge for scRNA-seq data. A vector is sparse when many of its coordinates are zero. In our setting, this corresponds to many gene expression values being measured as zero. There is a lot of biologically-true absence of expression, but there are also many artificial zeros where a gene is expressed, but not detected due to technical limitations (e.g., platform or sequencing depth) [3]. These zeros are called *dropout events*. Sparsity in general is challenging to model, and often necessitates the use of specialized algorithms and techniques. These challenges are amplified by the presence of dropout, which can severely hinder downstream applications.

2.2.3 Takeaways

The details of acquiring gene expression profiles are extremely complex, and we present only a small piece of that complexity above. For the purposes of this thesis, there are two key points to remember:

- Gene expression profiles contains signal for cellular function, including cell type. This is analogous to how a pixel vector of an image contains signal for the objects within that image.
- Gene expression profiles are incredibly noisy and sparse due to technological limitations of acquiring single-cell data. These properties present unique challenges for handling this data.

2.3 Single Cell Type Classification

One step in many scRNA-seq analysis pipelines is to identify which cell populations are present in an experiment. This problem is known as *single cell classification* (or annotation).

Traditional methods for cell-type annotation involve dimensionality reduction of gene expression profiles using techniques such as PCA, t-SNE [19], or UMAP [20]. The resulting representations are clustered using techniques like hierarchical clustering or k-means clustering. Then, each cluster is manually annotated by a biologist using a reference database of marker genes. While these methods have been very useful for extending our knowledge of cell populations, the manual annotation step is slow and expensive, as it requires experts to inspect individual marker genes. Moreover, manual annotations are less reproducible, as the annotations are not typically based on standardized vocabularies of cell types. Another drawback to these techniques is that they rely heavily on the availability of high-quality marker genes for each cell type. A cell's marker genes are a small set of genes whose expression determine what the type of the cell is. Discovering marker genes for each cell is itself a significant challenge in biology, and so developing methods that do not rely on curated high quality marker genes is preferred [5].

These challenges with traditional annotation techniques have motivated the use of supervised machine learning, which automatically learns relationships between features and labels. Viewed in this way, single cell annotation is just an instance of multi-class classification. Formally, we are given a gene expression matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ and a vector of each cell's cell type $\mathbf{Y}$. We want to learn a mapping f from gene expression vectors to cell types $f : \mathbf{X} \to \mathbf{Y}$.

A standard pipeline for these methods is to first perform traditional unsupervised feature selection, and then apply simple statistical classifiers. For example, scPred uses PCA to reduce the dimension of the gene expression vectors, and then learns an SVM with radial basis kernel on top of those representations [29]. CaSTLe uses similar data preprocessing steps followed by XGBoost [30], which is a classifier based on gradient boosted decision trees [31].

These conventional techniques leverage hand-engineered feature extraction methods to transform the gene expression vectors into a representation that is more suitable for statistical methods. Recently, more sophisticated models are being explored that learn their own feature extractors. In particular, there has been a recent push for DNN models, which can learn arbitrarily complex relationships between the data and label space. Examples include ACTINN [4] and scDAE [5], which are described in Section 3.1.

3 Related Works

3.1 Deep Learning in Single Cell Classification

Deep Learning models are capable of learning the arbitrarily complex relationships between gene expression and cell type. As such, DNNs are excellent candidates for single-cell classification. In the following section, I describe two DNN models that are used for single-cell classification.

3.1.1 ACTINN

Automated Cell Type Identification using Neural Networks (ACTINN) [4] implements a multi-layer perceptron for single-cell classification. The model uses four layers consisting of a linear function followed by a non-linear activation, defined as

$$l_i(\mathbf{x}) = \sigma\left(\mathbf{W}^{[i]}\mathbf{x} + \mathbf{b}^{[i]}\right) \qquad\qquad \hat{y} = l_4 \circ l_3 \circ l_2 \circ l_1(\mathbf{x}),$$

where the non-linear activation function $\sigma(\mathbf{x})$ is the element-wise Rectified Linear Unit (ReLU) function for the first three layers, and softmax for the final layer, defined as,

$$\text{ReLU}(\mathbf{x}) = [\max(\mathbf{x}_i, 0)]_i \qquad\qquad \text{softmax}(\mathbf{z}) = \left[\frac{\exp(\mathbf{z}_i)}{\sum_{j=1}^{n}\exp(\mathbf{z_j})}\right]_i .$$

This model is trained to minimize the L_2-regularized cross entropy loss $\mathcal{L}_{\text{ACTINN}} = \mathcal{L}_{CE} + \lambda\,\mathcal{L}_{\text{reg}}$ where

$$\mathcal{L}_{CE} = -\sum_{i=1}^{n}\sum_{j=1}^{c} y_j \cdot \log(\hat{y}_j) + (1 - y_j) \cdot \log(1 - \hat{y}_j), \tag{2}$$

and $\mathcal{L}_{\text{reg}} = \sum_{\theta\in\Theta} ||\theta||_2$. Here, Θ is the set of model parameters, which are $\mathbf{W}_{1,2,3,4}$ and $\mathbf{b}_{1,2,3,4}$.

ACTINN is a simple model that has relatively few parameters compared to other DNNs. When classifying gene expression profiles with 25,000 genes to 50 cell types the model weights contain $25,000 \times 100 + 100 \times 50 + 50 \times 25 + 25 \times 50 \approx 2.5$ million parameters. This means that it can be trained quickly and requires little memory to store. However, the simplicity of this model comes at a cost. ACTINN's performance suffers when trying to distinguish many cell subtypes [4]. This is undesirable, especially as new scRNA-seq technologies are growing more powerful and larger datasets with more cell subtypes are becoming available.

3.1.2 scDAE

scDAE is another DNN-based model that performs single-cell subtype classification [5]. The model consists of a denoising autoencoder (DAE), which extracts gene expression profile representations that are robust to noise, and a multilayer perceptron that classifies those representations.

The DAE is comprised of an encoder $enc(\cdot) : \mathbb{R}^m \to \mathbb{R}^{125}$ and a symmetrical decoder $dec(\cdot) : \mathbb{R}^{125} \to \mathbb{R}^d$, where m is the dimension of gene expression vectors (usually $m \approx 10^4$). The encoder maps gene expression vectors to representations, and the decoder tries to reconstruct the original vector from that representation. The DAE is trained to minimize the reconstruction error on the training gene expression matrix $\mathbf{X}$ with added Gaussian noise

$$\mathcal{L}_{DAE}(\mathbf{X}) = \frac{1}{2n}\sum_{i=1}^{n} ||\mathbf{x}_i - dec(enc(\mathbf{x}_i + \mathbf{e}_i))||_2^2, \tag{3}$$

where $\mathbf{e}_i \sim \mathcal{N}(0, 1)$. The purpose of corrupting the input is to make the encoder's representations robust to noise in the data since the model is incentivized to minimize the difference between representations for noisy versions of the same expression vector.

Once the representations are learned, they are fixed and passed to the classification component f, which is a multilayer perceptron that was trained to minimize cross entropy loss in Equation (2) on top of the representations generated by the encoder. Inference uses both the DAE and the classification component and runs as

$$\hat{y} = \text{argmax}_j f(enc(\mathbf{x}))_j.$$

The training of the model is split into three stages. First, the DAE was trained to minimize reconstruction error $\mathcal{L}_{DAE}$. Second, with the parameters of *enc* frozen, the classifier was trained to minimize the cross entropy $\mathcal{L}_{CE}$. Last, both components were trained jointly by summing the individual loss terms.

scDAE achieves higher accuracy than ACTINN, especially when classifying cell subtypes. The authors argue that this success is due to scDAE's representation learning component, which makes the model robust to corruption of the gene expression values. Due to the complexities in generating gene expression profiles, this sort of noise is common in single-cell data. One potential downside to scDAE's approach to representation learning is that it could confuse the model and cause representation collapse. For instance, if individual cells from two different classes are similar enough, it's possible that the perturbation of the input will make the representations indistinguishable. Then, the classifier would have a difficult time discerning which cells belong to which class.

3.2 Supervised Contrastive Learning

As discussed in Section 2.1, contrastive learning is a form of representation learning that has seen a lot of recent success in learning image representations [8, 13, 9, 11]. Inspired by this success, we employ contrastive learning as well.

Khosla et. al introduce *SupCon* [11], a framework for training supervised models with contrastive loss. In this framework, model training consists of two stages. First, an embedding function g is trained using contrastive loss. Then, the embedding function parameters are frozen, and a small classification head is trained on top of the learned representations.

The authors argue that models trained with this contrastive objective are capable of utilizing label information more effectively than cross entropy alternatives. Their experiments confirm the benefits of SupCon for image data by showing that their model has higher accuracy on ImageNet-1k, is more robust to noise (e.g., blur and contrast) using the ImageNet-C benchmark, and is less sensitive to hyperparameter changes. The model is trained with a version of InfoNCE loss [10], which is defined as

$$\mathcal{L}_{CL}^{\text{SupCon}} = \sum_{i=1}^{n} \frac{-1}{|\mathcal{P}(i)|} \sum_{p \in P(i)} \log \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_p / \tau)}{\sum_{a \in A(i)} \exp(\mathbf{z}_i \cdot \mathbf{z}_a / \tau)}, \tag{4}$$

where $\mathbf{z}_i = g(\mathbf{x}_i)$, $\mathcal{P}(i) = \{p \in A(i) : \mathbf{y}_p = \mathbf{y}_i\}$ is the set of indices of the positive pairs of $\mathbf{x}_i$ within our minibatch, $A(i) = \{a \in [n] : a \neq i\}$ is the set of all indices in our minibatch other than that of our current anchor point $\mathbf{x}_i$, and τ is a temperature parameter that controls the degree of separation for learned representations.

Although the specifics differ from Equation (1), the main idea is the same. The loss penalizes positive pairs that are far apart (e.g., $\mathbf{z}_i \cdot \mathbf{z}_p$ in the numerator will be small), and penalizes negative pairs that are close together (e.g., $\mathbf{z}_i \cdot \mathbf{z}_n$ will be large in the denominator).

Our approach is inspired by SupCon, and we use $\mathcal{L}_{CL}^{\text{SupCon}}$ in the contrastive learning stage of our model. However, we use a different type of classification head. We also do not use random transformations, whereas they augment their image dataset with random transformations (e.g., crop and blur). Finally, we apply our model to single-cell classification rather than to image data.

4 Model

In this section we first present the problem setting, which involves augmenting pretrained models in a way that improves their internal representations. Then, we propose a novel solution to that problem using contrastive learning.

4.1 Problem Statement

We are given a gene expression matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ accompanied by each cell's cell-type label $\mathbf{Y} \in \mathbb{R}^n$. Additionally, we have a pretrained backbone model for classifying single cell data $\hat{y} = f_\phi(g_\theta(\mathbf{x}))$. We refer to g_θ as the *encoder* and to f_ϕ as the *classification head*, and they are parameterized by θ and ϕ, respectively. Typically, f_ϕ will be relatively simple (e.g., a one- or two-layer neural network), while g_θ is more complex (e.g., a multi-layer VAE or GNN). We also note that g_θ and f_ϕ can consist of components that are trained jointly or separately.

Our goal is to introduce a simple component h_ψ that improves the representations that are provided to the classification head. Ideally, these representations will enable the augmented model $f_{\phi'}(h_\psi(g_\theta(\mathbf{x}))$ to achieve higher accuracy, be more robust to noise, and be less sensitive to hyperparameter selection than its base model $f_\phi(g_\theta(\mathbf{x}))$.

4.2 Details of our Approach

Let $\mathbf{z} = g_\theta(\mathbf{x})$ be the base model's embedding for single cell $\mathbf{x}$. We propose a simple component h_ψ that uses contrastive learning to improve model representations $\mathbf{z}$ and boost performance on downstream tasks. First, we define g_θ and f_ϕ of the base model such that inference runs as

$$Pr(\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x}) = f_\phi(g_\theta(\mathbf{x})).$$

For example, if the base model is a fully-connected multi-layer perceptron with three hidden layers (and four total layers), then we can take the first three layers as the encoder, and the last layer as the classification head. Then, the augmented model can be trained by following the procedure below.

1. Train the model $f_\phi(g_\theta(\mathbf{x}))$ according to its given training procedure.
2. Aquire the pretrained embeddings $\mathbf{z} = g_\theta(\mathbf{x})$ for each cell in the gene expression matrix.
3. Train $h_\psi(\mathbf{z})$ by minimizing $\mathcal{L}_{CL}$, as given in Equation (5) and Equation (6) on the training dataset.
4. Freeze the parameters ψ, and train $f_{\phi'}(h_\psi(\mathbf{z}))$ using the original training procedure.

Note that we can skip step 1 if we have access to a pretrained model.

We tried two different model formulations for the contrastive component h. The first model is the *multi-layer-perceptron contrastive component* (MLP-CC), which is shown in Figure 2 (as applied to the ACTINN base model [4]). We implemented MLP-CC using a fully-connected neurual network with two layers separated by ReLU non-linearities. We will refer to this component as $h_\psi^{\text{MLP}} : \mathbb{R}^d \to \mathbb{R}^d$. For a given expression vector $\mathbf{x}$, recall that its base embedding is denoted $\mathbf{z} = g(\mathbf{x})$. Let the MLP-CC embedding be defined as $\tilde{\mathbf{z}} = h_\psi^{\text{MLP}}(\mathbf{z})$.

To train MLP-CC, we applied the InfoNCE loss to the unit-hypersphere-normalized contrastive embeddings, which is defined as

$$\mathcal{L}_{CL}^{\text{MLP}} = \sum_{i=1}^{n} \frac{-1}{|\mathcal{P}(i)|} \sum_{p \in \mathcal{P}(i)} \log \frac{\exp(\tilde{\mathbf{z}}_i \cdot \tilde{\mathbf{z}}_p / \tau)}{\sum_{a \in A(i)} \exp(\tilde{\mathbf{z}}_i \cdot \tilde{\mathbf{z}}_a / \tau)}, \tag{5}$$

where $\tilde{\mathbf{z}}_i = \frac{h_\psi^{\text{MLP}}(\mathbf{z}_i)}{||h_\psi^{\text{MLP}}(\mathbf{z}_i)||_2}$, $A(i) = \{a \in [n] : a \neq i\}$ is the set of all indices in our minibatch other than that of our current anchor cell i, $\mathcal{P}(i) = \{p \in A(i) : y_p = y_i\}$ is the set of indices that have the same label as cell i, and τ is a temperature parameter that controls the degree of separation for learned representations.

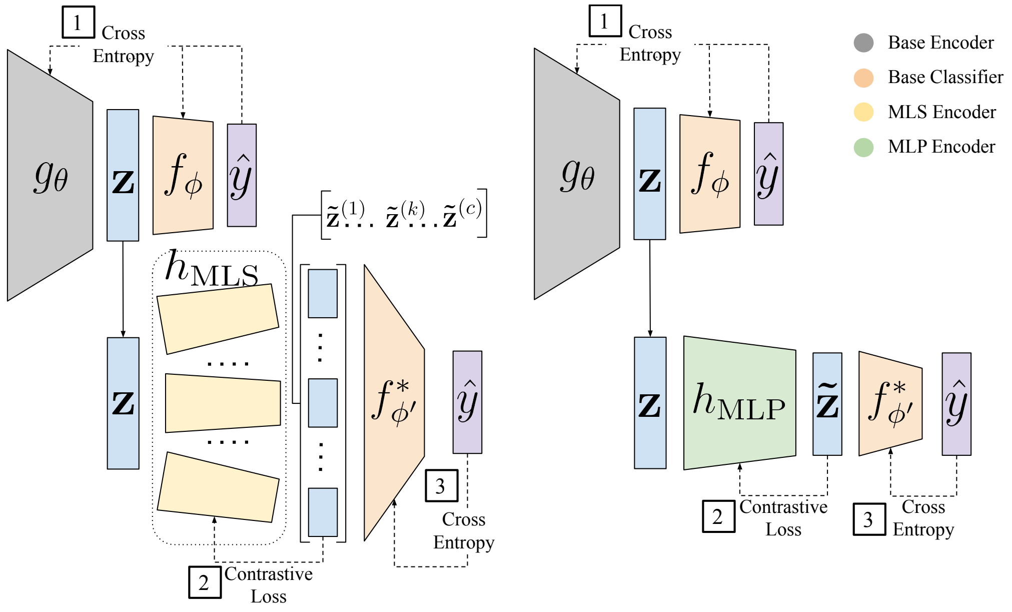

Figure 2: Base models with their contrastive components MLS-CC (left) and MLP-CC (right). Dashed line indicates the backpropogation of loss. In both versions the base model is trained by minimizing cross entropy on the training set. Then, the frozen base emebddings $\mathbf{z}$ are passed into the contrastive component, which is trained by minimizing the contrastive loss of its learned embeddings $\tilde{\mathbf{z}}$. Last, the classification head $f^*_{\phi'}$ is retrained to classify those contrastive embeddings. We call the classification head f^* to denote that it can take a different form than the original f (e.g., the input layer for the classification head after the MLS contrastive component will often be a different size than that of the original classifier).

The second model is the *multi-latent-space contrastive component* (MLS-CC), which is shown in Figure 2 (as applied to the ACTINN base model [4]). We refer to this embedding function as $h_\psi^{\text{MLS}} : \mathbb{R}^d \to \mathbb{R}^{c\times 2}$, where c is the number of cell types. For a given base embedding $\mathbf{z}$, a two dimensional embedding is created for each class $h_k(\mathbf{z}) = \tilde{\mathbf{z}}^{(k)} \in \mathbb{R}^2$. These representations are normalized to the unit hypersphere such that $||\tilde{\mathbf{z}}^{(k)}||_2 = 1$. Then these representations are concatenated together into a $c \times 2$-dimensional representation. In other words,

$$h_\psi^{(\text{MLS})}(\mathbf{z}) = \begin{bmatrix} h_1(\mathbf{z}) \ / \ ||h_1(\mathbf{z})||_2 \\ \vdots \\ h_k(\mathbf{z}) \ / \ ||h_k(\mathbf{z})||_2 \\ \vdots \\ h_c(\mathbf{z}) \ / \ ||h_c(\mathbf{z})||_2 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{z}}^{(1)} \\ \vdots \\ \tilde{\mathbf{z}}^{(k)} \\ \vdots \\ \tilde{\mathbf{z}}^{(c)} \end{bmatrix} = \tilde{\mathbf{z}}.$$

We refer to the range of the learned function h_k as *cell k's latent space*. The goal for each h_k is to embed cells into a latent space such that embeddings labelled with cell type k are well separated from those of the cells from all other cell types. This representation learning scheme forces the model to devote a fixed number of dimensions to learning embeddings for each cell type, including those that are under-represented in the training set. Representing individual cells in this way will hopefully (i) enable the classification head to make more accurate predictions on rare cell types and (ii) enable the classification head to leverage underlying relationships between cell types for better overall predictions.

We use a slightly modified version of InfoNCE for our contrastive loss $\mathcal{L}_{CL}^{\text{MLS}}$, which is defined as

$$\mathcal{L}_{CL}^{\text{MLS}} = \sum_{k=1}^{c} \sum_{i=1}^{n} \frac{-1}{|\mathcal{P}(i)|} \sum_{p\in\mathcal{P}(i)} \log \frac{\exp(\tilde{\mathbf{z}}_i^{(k)} \cdot \tilde{\mathbf{z}}_p^{(k)}/\tau)}{\sum_{a\in A(i)} \exp(\tilde{\mathbf{z}}_i^{(k)} \cdot \tilde{\mathbf{z}}_a^{(k)}/\tau)}, \tag{6}$$

where $\tilde{\mathbf{z}}_i^{(k)} = h_k(\mathbf{z}_i)$, and $A(i)$, $P(i)$, and τ are defined the same way as in Equation (5).

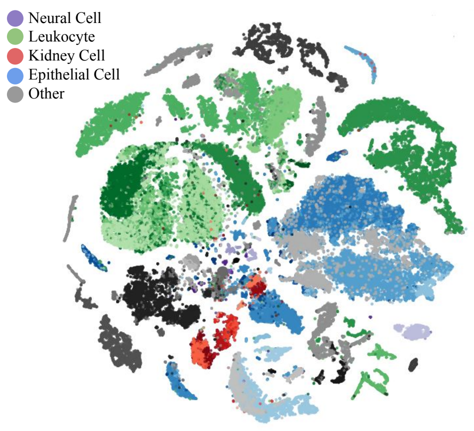

Figure 3: t-SNE visualization of unbalanced dataset on gene expression vectors. Broad cell types are assigned a hue, and specific cell sutypes within that supertype are assigned different shades of that color.

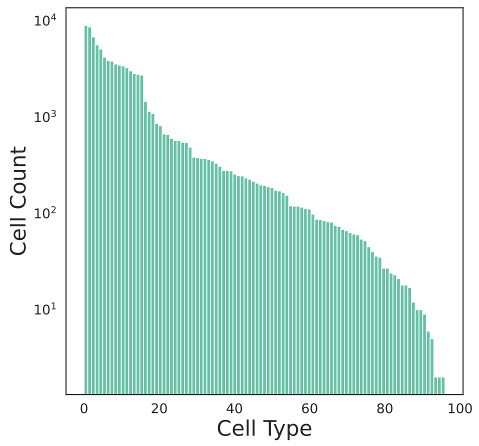

Figure 4: Histogram of counts for each cell-type for the entire *Tabula Microcebus* dataset [2].

5 Experiments

In this section we describe the datasets and experiments used to evaluate our model. We also present our results.

5.1 Datasets

In our experiments, we analyze the dataset of *Tabula Microcebus* Lemur 3 (10×) [2], which is publicly available on figshare. This dataset contains over 90,000 individual cell gene expression vectors and consists of 97 cell types. From this dataset we created three data settings.

1. *Simple*, which consists of the 25-most-common cell types from *Tabula Microcebus* and randomly downsamples the cells in each type until there are approximately 500 cells per type in the train dataset.

2. *Data Scarce*, which consists of the 50-most-common cell types and randomly downsamples the cells in each type until there are approximately 60 cells per type in the train dataset.

3. *Unbalanced*, which consists of the 80-most-common cell types, with no downsampling. The training data set for this data setting contains some cell types with more than 5,000 samples, and some cell types with fewer than 50 samples. As shown in Figure 4, there is a large imbalance in the number of cells per cell types in the full *Tabula Microcebus* [2] dataset.

For each data setting we used the raw gene expression counts and used the preprocessing techniques that the base model used. For all base models this involved removing all genes whose expression was zero in every cell. For ACTINN [4] and logistic regression, we also normalized gene expressions by the total sum for that gene in the expression matrix, multiplied by 10,000, and took the log-plus-1 of the resulting values. For scDAE [5], we divided each gene expression by the largest value in the gene expression matrix and took the log-plus-1 of the resulting values.

5.2 Experimental Procedure

In our experiments, we applied our contrastive component to three base models: ACTINN [4], scDAE [5], and logistic regression. For each model we defined an encoder component and a classification head as

1. ACTINN: The encoder consisted of the first two fully-connected layers, while the classification head consisted of the last two layers, followed by a softmax;
2. scDAE: The encoder consisted of the Denoising Auto Encoder, while the classification head consisted of two fully-connected layers, followed by a softmax;
3. Logistic Regression: The encoder consisted of the identity function, while the classification head consisted of a single fully-connected layer followed by a softmax. Note that this means the contrastive component was applied directly to the gene expression vectors in the data space.

For each combination of model (logistic regression, ACTINN, and scDAE), CL type (Base, MLS, and MLP), and setting (Simple, Data Scarce, and Unbalanced) we created a training dataset, validation dataset, and test dataset that consisted of 80%, 10%, and 10%, of the data, respectively. We then trained the appropriate model on the train set using the given base models preprocessing scheme, and selected the best model during training according to performance on the validation set. Then, each model was evaluated on the test set.

This procedure was repeated 5 times, using different random seeds for each trial, which produced a different train, test, and validation set each time, as well as different model initializations. Performance metrics were computed by averaging the metrics across the five trials for each model, CL type, and setting triplet.

5.3 Overall Results

In Figure 5, we present bar plots comparing the performance between base models with and without a contrastive component. The height of each bar is the difference in performance (e.g., macro F1 score or top-1 accuracy) between the given base model *with* a contrastive component and the given base model *without* a contrastive component. For example, the height of the bar with the MLS-CC component would be

$$y_{\text{bar}}^{\text{MLS}} = \text{performance}(f_{\phi'}(h_{\psi}^{\text{MLS}}(g_\theta(\mathbf{x})))) - \text{performance}(f_\phi(g_\theta(\mathbf{x}))).$$

From these plots, we can see that MLP-CC boosts top-1 accuracy and macro F1 score for logistic regression in the Simple setting, while it hurts performance in six model-setting combinations. In the Data Scarce setting, MLP-CC has little effect when applied to scDAE and logistic regression. We also show that MLS-CC boosts top-1 accuracy and macro F1 score for four model-setting combinations, hurts them in three, and has little effect in one. This indicates that both MLP-CC and MLS-CC are not as model-agnostic as we had hoped. However, MLS-CC does appear to be a stronger model than MLP-CC, and improves logistic regression classification accuracy and macro f1 score across all settings.

In Table 1, we show the top-1 accuracy, top-3 accuracy, and macro F1 score for each combination of models in all three settings. In each of the settings, logistic regression with MLS-CC (LR+MLS) outperforms all other models across all metrics except for top-3 accuracy in the Unbalanced setting.

5.4 Per Cell-Type Results

We also measured the area under the receiver operating characteristic (AUROC) and the area under the precision recall curve (AURPC) for each individual cell type in each trial in the Unbalanced setting. With 5 trials and 80 cell types, this produced 400 performance samples for each model. We computed a paired t-test on these samples between each base model and its MLS-augmented version. For ACTINN [4], we found that the MLS component decreased average AUROC and AUPRC performance, with a p-values of 5.51×10^{-5} and 1.31×10^{-7}, respectively. For scDAE [5], we found that the MLS component decreased average AUROC and AUPRC performance, with a p-values of 2.46×10^{-5} and 0.0454, respectively. For logistic regression, however, the MLS component *increased* average AUPRC performance, with a p-value of 0.0115; the increase in AUROC was not statistically significant.

In Figure 6, we examine more closely how cell-type classification performance is impacted by number of cells of that type in the training set. Each data point is the difference in AUPRC for a cell-type between the base model and its MLS-augmented version. We plot the performance against the number of samples in the training dataset. Note that for cell types that have high representation in the training dataset (e.g., $> 10^5$

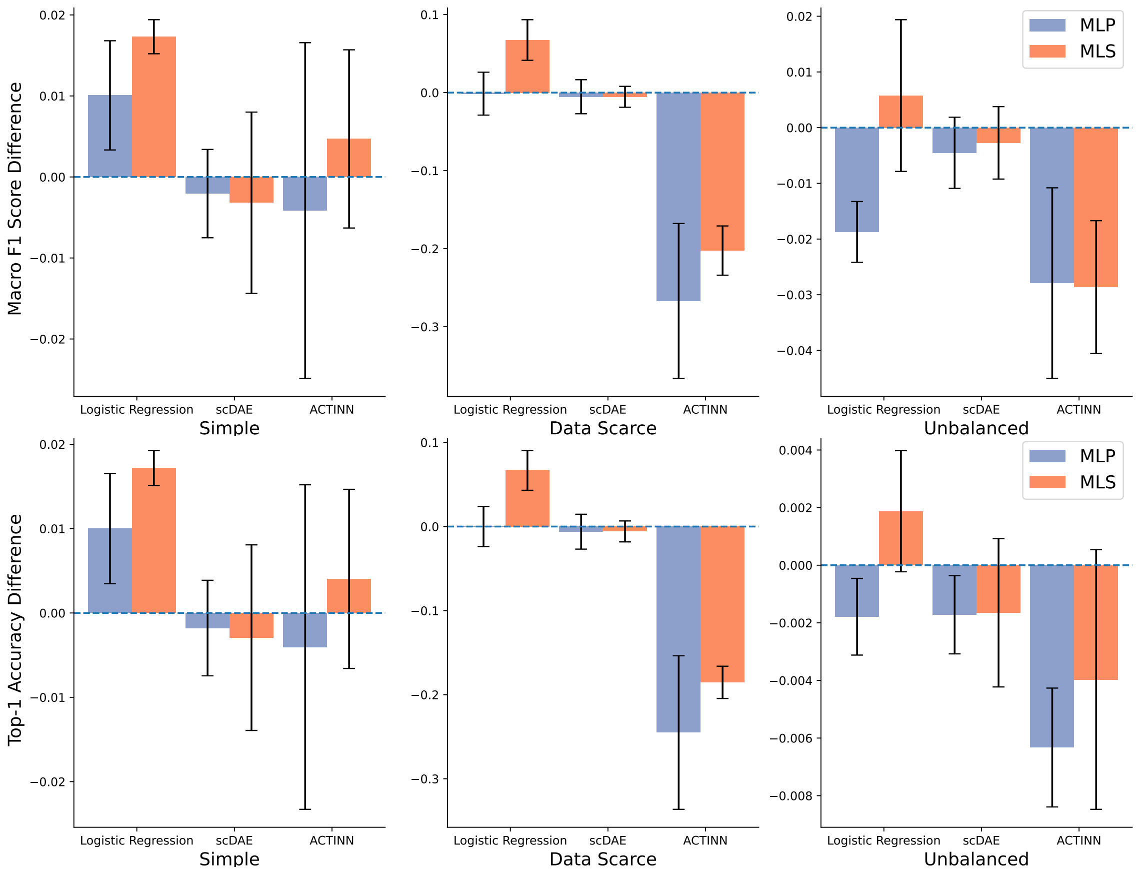

Figure 5: Bar plot where height of each bar is computed as the base model performance with contrastive component minus base model performance without contrastive component. The two contrastive formulations, MLP vs MLS, are shown in gray and orange, respectively. The performance in macro F1 score is shown on top, and the performance in top-1 accuracy is shown on the bottom. Negative height indicates the contrastive component hurts performance, while positive height indicates it improves performance. Black capped bars indicate the standard deviation of the difference.

samples) there is very little difference in performance. However, for cell types with lower representation, there is large variation in performance.

One of the hopes for MLS-CC is that it would improve performance on rare cell types. For logistic regression (middle), the MLS-CC version appears to have more positive scores (i.e., CL version performs better than the base model), especially for cell types with lower representation. This does not appear to be the case for the other two models (left and right), where relative performance is more negative for smaller cell-types. This indicates that when MLS-CC is applied directly to the gene expression profiles, it can learn representations that enable classification of rare cell types, although this does not extend to pretrained model embeddings.

5.5 MLS Embeddings

In Figure 7, we present visualizations of the latent space embeddings for three cell-types. Each plot corresponds to all $\mathbf{z}^{(k)} \in \mathbb{R}^2$ for cell k in the MLS-CC model applied directly to the gene expression vectors. The embeddings correspond to three different qualities of representation. In the Vein Endothelial Cell plot (left) the red points, which represent Vein Endothelial cells, are well separated from the blue points, which represent all the other cell types. Due to the separation of cells by cell-type, we expect the classification component to be able to easily distinguish Vein Endothelial cells by looking at this latent space. The second plot (middle), shows the Keratinocyte latent space, which has slightly worse representations. This is because, although the red points are relatively well separated from the blues, they are not as far apart as they could be, and the red class is fairly spread out. The Cardiocyte representations (right) appear to be the worst. The red points are spread out all over the circle and overlap with many blue points. When the classification component is applied to these representations, it will be difficult to discern the Cardiocyte cells from the other cell types.

Table 1: Model performance in Simple (left), Data Scarce (middle), and Unbalanced settings in terms of top-1 and top-3 classification accuracies (Top-1 and Top-3) and macro F1 score (F1) for each model and CL type combination. LR stands for logistic regression, and the "+MLS/MLP" indicates the model was augmented with its respective contrastive component. The best performance for each metric in the given setting is bolded.

	Top-1	Top-3	F1
ACTINN	91.54	97.88	91.46
LR	91.88	99.19	91.83
scDAE	92.77	98.71	92.72
ACTINN+MLP	91.14	97.11	91.05
LR+MLP	92.88	99.30	92.84
scDAE+MLP	92.60	98.54	92.52
ACTINN+MLS	91.95	98.72	91.93
LR+MLS	**93.60**	**99.42**	**93.56**
scDAE+MLS	92.48	98.63	92.40

(a) Simple

Top-1	Top-3	F1
62.85	77.61	57.71
81.88	95.70	81.48
88.48	97.32	88.03
38.34	59.07	31.01
81.89	95.48	81.35
87.87	96.39	87.50
44.31	64.58	37.47
88.55	**97.79**	**88.23**
87.90	96.85	87.49

(b) Data Scarce

Top-1	Top-3	F1
91.74	98.64	81.65
92.85	**99.35**	86.33
92.38	98.68	86.60
91.11	98.26	78.86
92.67	99.15	84.45
92.21	97.93	86.15
91.35	98.24	78.79
93.03	99.26	**86.90**
92.22	97.96	86.33

(c) Unbalanced

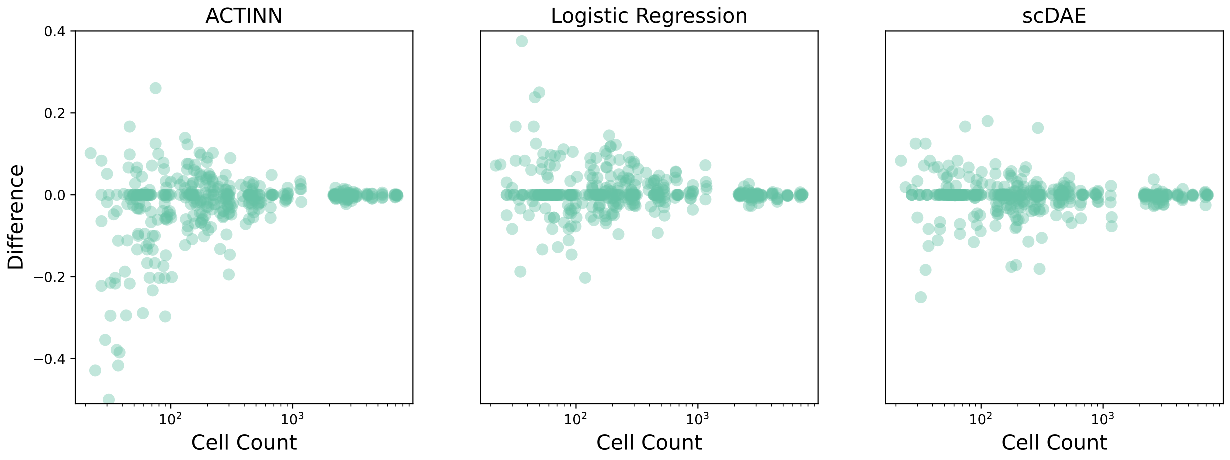

Figure 6: The difference in AUPRC for the MLS version of the base model and the vanilla base model for each cell type. The x-axis shows the number of samples that particular cell type has in the training dataset. A positive difference along the y-axis indicates that the MLS version was better at classifying that cell type than its base.

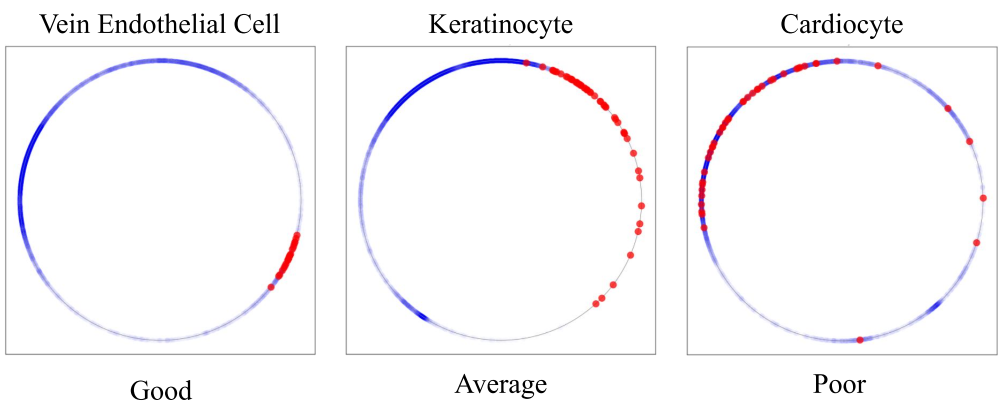

Figure 7: Logistic regression with MLS-CC embeddings for three cell types. Each point is the 2-dimensional embedding of a cell in the Unbalanced test set. The red points are embeddings of cells that are of that cell type, and the blue points are emebddings of cells that are not of that cell type. All are shown projected onto the unit hypersphere, which is traced in gray.

6 Discussion

In this work, we developed a model-agnostic component that uses contrastive learning to create internal model representations with the goal of improving single-cell classification. We found that MLS-CC was more successful at increasing classification performance than MLP-CC, but that neither contrastive component improved performance across all models. However, we also found that our MLS contrastive component improved the performance of logistic regression and it was the best model combination that we tested. Logistic Regression with MLS-CC achieved the highest top-1 accuracy and macro F1 score. The MLS-CC component also significantly increased the average AUPRC over the base model, especially for rare cell types with a paired t-test p-value of 0.0115. This indicates that the MLS formulation of contrastive learning is a promising way of representing raw gene expression data, although this does not extend to pretrained model embeddings.

6.1 Future Work

Although our results indicate that the contrastive components do not boost model performance universally, there are several directions for future work that we think are worth pursing.

First, we would like to evaluate more datasets and models. Our initial hypothesis was that MLS-CC is a model-agnostic method for improving internal model representations. However, we only evaluated on three models and one dataset due to time constraints. To draw further conclusions about what our method does well and where it fails, we must experiment with more base models and datasets.

Second, we would like to incorporate domain knowledge into the representation learning or classification stages of our model. We took initial steps in this direction by creating a classification component (to replace the base model's) that would work on top of the MLS-CC representations. The classifier would make a probability prediction for whether the cell belonged to each MLS latent space, and the probabilities would be propogated along a graphical model representing the Cell Ontology. Unfortunately, we did not have time to fully investigate this classifier in this work, but it shows potential.

Third, we would like to include biologically inspired data augmentations in the contrastive learning component. One of the reasons contrastive learning is so successful in NLP and CV is because it is easy to define transformations of the original input that don't change our human-level understanding of the input (e.g., blurring an image of a dog). By defining the original and transformed inputs as positive pairs, contrastive learning models can learn that these transformations don't affect the meaning of the image or text. In biology, it is not obvious how to define such transformations because we do not have as intuitive of an understanding of the data. For instance, applying blur to a gene expression vector might result in a profile that represents a completely different cell type. Once we gain a better understanding of how genes work together to produce cells with particular functions, we can define transformations that make our classifiers more aware of representation invariance.

By further incorporating experimental settings, domain knowledge, and biology-inspired data augmentations, we believe that we could build upon the contrastive component presented in this work to improve its accuracy, robustness, and sensitivity to rare cell types.

References

[1] V. Kiselev and S. Bell, “Introduction to single-cell rna-seq,” Oct 2019.

[2] T. T. M. Consortium, C. Ezran, S. Liu, S. Chang, J. Ming, O. Botvinnik, L. Penland, A. Tarashansky, A. de Morree, K. J. Travaglini, K. Hasegawa, H. Sin, R. Sit, J. Okamoto, R. Sinha, Y. Zhang, C. J. Karanewsky, J. L. Pendleton, M. Morri, M. Perret, F. Aujard, L. Stryer, S. Artandi, M. Fuller, I. L. Weissman, T. A. Rando, J. E. Ferrell, B. Wang, I. De Vlaminck, C. Yang, K. M. Casey, M. A. Albertelli, A. O. Pisco, J. Karkanias, N. Neff, A. Wu, S. R. Quake, and M. A. Krasnow, “Tabula microcebus: A transcriptomic cell atlas of mouse lemur, an emerging primate model organism,” *bioRxiv*, 2021.

[3] D. Lähnemann, J. Köster, E. Szczurek, D. J. McCarthy, S. C. Hicks, M. D. Robinson, C. A. Vallejos, K. R. Campbell, N. Beerenwinkel, A. Mahfouz, *et al.*, “Eleven grand challenges in single-cell data science,” *Genome biology*, vol. 21, no. 1, pp. 1–35, 2020.

[4] F. Ma and M. Pellegrini, “ACTINN: automated identification of cell types in single cell RNA sequencing,” *Bioinformatics*, vol. 36, pp. 533–538, 2020.

[5] J. Choi, “Cell subtype classification via representation learning based on a denoising autoencoder for single-cell rna sequencing,” *IEEE*, 2021.

[6] D. M. Ciortan M., “Contrastive self-supervised clustering of scRNA-seq data,” *BMC Bioinformatics*, vol. 22, p. 280, 2021.

[7] T. Abdelaal, L. Michielsen, D. Cats, D. Hoogduin, H. Mei, M. Reinders, and A. Mahfouz, “A comparison of automatic cell identification methods for single-cell rna sequencing data.,” *Genome Biology*, vol. 20, 2019.

[8] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, “Momentum contrast for unsupervised visual representation learning,” *CoRR*, vol. abs/1911.05722, 2019.

[9] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607, PMLR, 13–18 Jul 2020.

[10] A. van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *CoRR*, vol. abs/1807.03748, 2018.

[11] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, “Supervised contrastive learning,” *CoRR*, vol. abs/2004.11362, 2020.

[12] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” *CoRR*, vol. abs/2006.09882, 2020.

[13] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” *CoRR*, vol. abs/2103.00020, 2021.

[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018.

[15] Y. Zhang, R. He, Z. Liu, K. H. Lim, and L. Bing, “An unsupervised sentence embedding method bymutual information maximization,” *CoRR*, vol. abs/2009.12061, 2020.

[16] L. Logeswaran and H. Lee, “An efficient framework for learning sentence representations,” *CoRR*, vol. abs/1803.02893, 2018.

[17] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, “Towards unsupervised dense information retrieval with contrastive learning,” *CoRR*, vol. abs/2112.09118, 2021.

[18] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," in *International Conference on Learning Representations (ICLR)*, April 2021.

[19] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[20] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2018. cite arxiv:1802.03426Comment: Reference implementation available at http://github.com/lmcinnes/umap.

[21] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546 vol. 1, 2005.

[22] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015.

[23] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[24] S. Namura, "Single-cell genomics to understand disease pathogenesis," *Nature Journal of Human Genetics*, vol. 66, pp. 75–84, 2020.

[25] R. A. . Y. N. Wagner A., "Revealing the vectors of cellular identity with single-cell genomics," *Nature Biotechnology*, vol. 34, p. 1145–1160, 2016.

[26] A. Kolodziejczyk, J. Kim, J. Tsang, T. Ilicic, J. Henriksson, K. Natarajan, A. Tuck, X. Gao, M. Bühler, P. Liu, J. Marioni, and S. Teichmann, "Single cell rna-sequencing of pluripotent states unlocks modular transcriptional variation," *Cell Stem Cell*, vol. 17, no. 4, pp. 471–485, 2015.

[27] F. Tang, C. Barbacioru, and Y. Wang, "mRNA-Seq whole-transcriptome analysis of a single cell.," *Nature Methods*, vol. 6, pp. 377–382, 2009.

[28] M. D. Luecken and F. J. Theis, "Current best practices in single-cell rna-seq analysis: a tutorial," *Molecular Systems Biology*, vol. 15, no. 6, p. e8746, 2019.

[29] J. H. Alquicira-Hernandez J., Sathe A., "scPred: accurate supervised method for cell-type classification from single-cell rna-seq data," *Genome Biology*, vol. 20, p. 264, 2019.

[30] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.

[31] S. T. Lieberman Y, Rokach L, "Castle–classification of single cells by transfer learning: Harnessing the power of publicly available single cell rna sequencing experiments to annotate new experiments.," *PLoS ONE*, vol. 13, 2018.

[32] "Method of the year 2013.," *Nature Methods*, vol. 11, p. 1, 2014.

[33] L. Weng, "Contrastive representation learning," *lilianweng.github.io*, 2021.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems* (C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds.), vol. 26, Curran Associates, Inc., 2013.

[35] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *CoRR*, vol. abs/1502.03167, 2015.