Contrastive Learning for Improved Single Cell Annotation

1 Introduction

In this work, we present a model-agnostic component that uses supervised contrastive learning to learn high-quality cell representations and improve cell type annotation from gene expression profiles. Additionally, this component is fairly simple and can easily be incorporated into any deep neural network classification model, requiring minimal changes to existing model architectures. This work is largely inspired by the success of representation learning techniques in semi-supervised image classification. In the following sections I will describe the problem setting, our approach to solving it, and our experimental results.

2 Problem Statement

We are given a gene expression matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ accompanied by each cell's label $\mathbf{Y} \in \mathbb{R}^n$. Additionally, we have a pretrained backbone model for classifying single cell data $f_\phi(g_\theta(\mathbf{x}))$. We refer to g_θ as the *encoder*, and f_ϕ as the *regressor*, 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 disconnected.

Our goal is to introduce a simple component h_ψ that improves the representations that are provided to the regressor. 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 $f_\phi(g_\theta(\mathbf{x}))$.

3 Method

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 align the model representations $\mathbf{z}$ and improve 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 regressor. 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 pretrained embeddings $\mathbf{z} = g_\theta(\mathbf{x})$ for each cell in the gene expression matrix.
3. Train $h_\psi(\mathbf{z})$ by minimizing $\mathcal{L}_{CL}$ on the training dataset.
4. Freeze the parameters ψ, and train $f_{\phi'}(h_\psi(\mathbf{z}))$ using original training procedure.

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

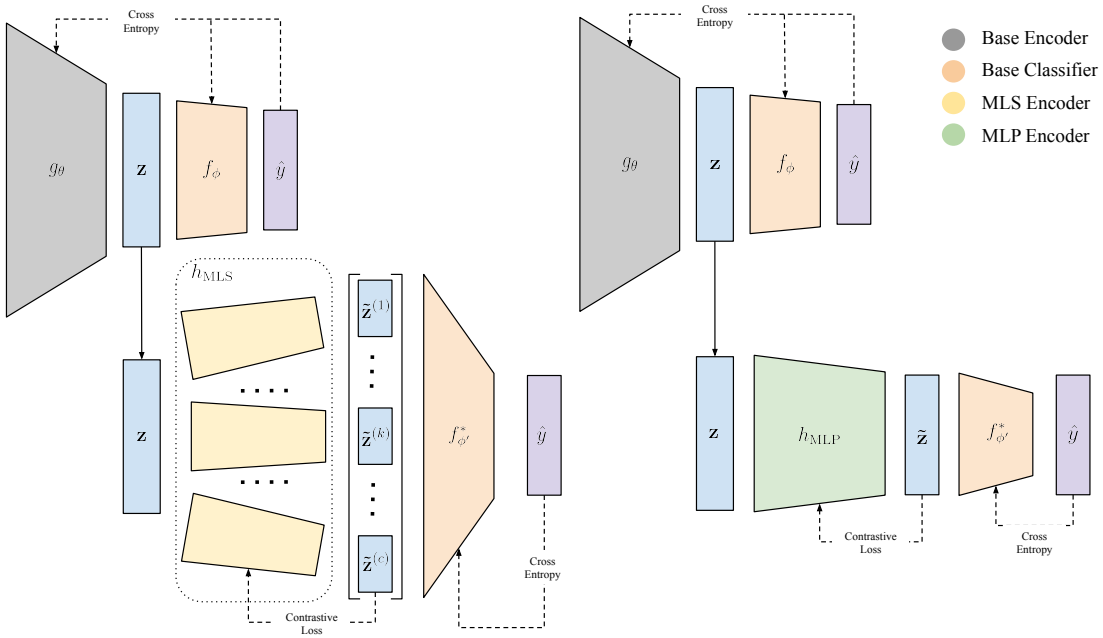

Figure 1: Base models with their contrastive components MLS-CL (left) and MLP-CL (right). Dashed line indicates the backpropogation of loss. In both versions the base model is trained by minimizing cross entropy on the training set. Then, the frozen base emebddings $\mathbf{z}$ are passed into the contrastive component, which is trained by minimizing the contrastive loss of its learned embeddings $\tilde{\mathbf{z}}$. Last, the regressor is retrained to classify those contrastive embeddings.

We tried two different models for the contrastive component h. The first model is the *multi-layer-perceptron contrastive component* (MLP-CL), which is shown in 1 (as applied to the ACTINN base model). We implemented MLP-CL using a fully-connected neurual network with two layers separated by ReLU non-linearities. We will refer to this embedding function 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-CL embedding be defined as $\tilde{\mathbf{z}} = h_\psi^{\text{MLP}}(\mathbf{z})$.

To train MLP-CL, we applied the InfoNCE loss $\mathcal{L}_{CL}$ to the unit-sphere normalized contrastive embeddings, which is defined as

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

where $\tilde{\mathbf{z}}_i = h_\psi^{\text{MLP}}(\mathbf{z}_i)$, $A(i) = \{a \in [n] : a \neq i\}$ 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.

The second model is the *multi-latent-space contrastive component* (MLS-CL), which is shown in 1 (as applied to the ACTINN base model). We refer to this embedding function as $h_\psi^{\text{MLS}} : \mathbb{R}^d \to \mathbb{R}^{2 \times c}$, where c is the number of classes. 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 $2 \times c$-dimensional representation. In other words,

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

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 that have cell type k are well separated from the cells that have a different label. 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. The idea is that representing individual cells in this way will (i) enable the classifier to make more accurate predictions on rare cell types, and (ii) enable the classifier to leverage underlying relationships between cell types for better overall predictions.

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

$$\mathcal{L}_{MLS} = \sum_{k=1}^{c} \sum_{i=1}^{n} \frac{-1}{|\mathcal{P}(i)|} \sum_{p \in \mathcal{P}(i)} \log \frac{\exp(\tilde{\mathbf{z}}_i^{(k)} \cdot \tilde{\mathbf{z}}_p^{(k)} / \tau)}{\sum_{a \in A(i)} \exp(\tilde{\mathbf{z}}_i^{(k)} \cdot \tilde{\mathbf{z}}_a^{(k)} / \tau)}, \tag{2}$$

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 1.

4 Experiments

4.1 Datasets

In our experiments, we used the dataset of *Tabula Microcebus* Lemur 3 (10×), which contains over 90,000 individual cells and consists of 97 cell types. We created experiment datasets by using 80% of the data for training, 10% for testing, and 10% for validation. Additionally, from this dataset we created two data settings. The first is *Default*, which consists of the 25 most common cell types from *Tabula Microcebus* where the cells in each type where randomly downsampled until each cell type was equally represented in the training, validation and test sets. For a given trial, this resulted in an average of 500 cells per type. The second setting is *Data Scarce*, which consists of the 50 most common cell types. Again, the cells in each type where randomly downsampled until each cell type was equally represented in the training, validation and test sets. For a given trial, this resulted in an average of 60 cells per type.

4.2 Experiment

For each combination of models (logistic regression, ACTINN, and scDAE), CL type (Base, MLS, and MLP), and settings (Default and Data Scarce) we 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 for the given setting.

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

4.3 Results

The performance in accuracy and macro F1 score indicates that both contrastive components do not uniformly improve model performance as hoped. However, there are of positive results to note in the table below

- MLS-CL improves logistic regression across all metrics in all settings.
- In both settings, the highest performing model in all metrics is MLS-CL on logistic regression.
- MLS-CL improves ACTINN across all metrics in the default setting.

The log fold change bar plot highlights the comparisons between model performance in terms of top-1 accuracy.

Table 1: Model performance in Default (left) and Data Scarce (right) settings in terms of top-1 and top-3 accuracy (Top-1 and Top-3), and macro F1 score (F1). The best performance for each metric and model is bolded. For each metric, a asterisk is placed next to the best performing model in that setting.

Model	Top-1	Top-3	F1
ACTINN	0.9174	0.9864	0.8165
LR	0.9285	**0.9935**	0.8633
scDAE	0.9238	0.9868	0.8660
ACTINN + MLP	0.9111	0.9826	0.7886
LR + MLP	0.9267	0.9915	0.8445
sDAE + MLP	0.9221	0.9793	0.8615
ACTINN + MLS	0.9135	0.9824	0.7879
LR + MLS	**0.9303**	0.9926	**0.8690**
scDAE + MLS	0.9222	0.9796	0.8633

Model	Top-1	Top-3	F1
ACTINN			
Base	**0.6285**	**0.7761**	**0.5771**
MLP	0.3849	0.5881	0.3177
MLS	0.4416	0.6434	0.3754
LR			
Base	0.8188	0.9570	0.8148
MLP	0.8349	0.9516	0.8302
MLS	**0.8908***	**0.9750***	**0.8880***
scDAE			
Base	**0.8848**	**0.9732**	**0.8803**
MLP	0.8787	0.9688	0.8739
MLS	0.8777	0.9665	0.8734

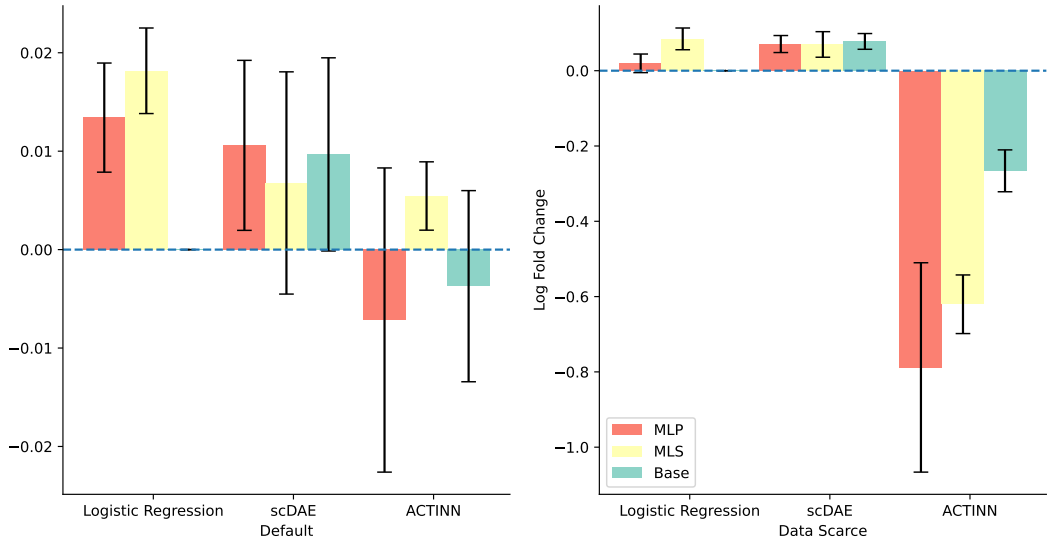

Figure 2: Log fold change of base models and their augmented versions for the metrics top-1 accuracy in Default and Data Scarce settings. Log fold is comparing each model to vanilla logistic regression baseline. E.g. if acc_{LR} is the average accuracy for logistic regression base model, and $\text{acc}_{\text{scDAE-MLP}}$ is the average accuracy for scDAE with an MLP component, then the height of the orange bar in the scDAE column is computed as $\log\left(\frac{acc_{\text{scDAE-MLP}}}{acc_{\text{LR}}}\right)$. Bars indicate the standard deviation of the given models performance across 5 different random data splits and model seeds.