Accurate Identification of Cell Type from scRNA-seq

Qixin Ye
University of Waterloo
q32ye@uwaterloo.ca

Abstract

The goal of this project is to develop a machine learning model to classify cells into different types based on their gene expression from scRNA-seq (single cell RNA) data. The model will be trained on a large dataset of over 700,000 cells with over 40,000 genes per cell, which includes cells generated using a different sequencing method. There are several challenges to this task, including the large amount of data, sparsity of the data, long-tailed distribution of the data, and the presence of an ”unknown” category in the test set. The model will be evaluated using the F1 score, which is a measure of accuracy and recall. Both supervised and unsupervised methods will be used, and the final result will be generated by taking a majority vote from both methods.

1 Introduction

The purpose of this project is to achieve accurate classification of cells. Cells are the basic structural and functional units of a living organism, and the ability to accurately identify cell types is of significant value in medicine and bioinformatics. Currently, most cell classification tasks are still performed manually, with a process similar to that shown in the figure on the right. This involves first isolating single cells from a tissue sample, followed by sequencing the RNA and finally performing manual identification of the cell type, which can be seen as a somewhat tedious and labor-intensive process.

The specific task is to automatically identify the classes of cells at four levels, with each level containing multiple classes. Because cells in levels 1-3 can be inferred from the class of the cell in level 4, the actual task is to automatically identify the class of cells in level 4. If a cell type does not have a level 3 or 4, then the level 3 and level 4 of that cell type are the same as the previous level. This project is evaluated using the F1 score, which is a measure of accuracy and recall and is proportional to the precision.

2 Related Works

As a relatively new field, there is currently a limited number of studies available for reference. However, these studies have also provided some potential avenues for exploration.

The commonly used method for cell type identification from scRNA-seq data is to first clustering to different groups using unsupervised method, and then label each cluster to correct types [1].

Another method is using supervised method, this method is also gaining popularity due to better accuracy, robustness, and computational performance. However, the supervised method relies heavily on feature selection, prediction method, and choice of the reference dataset [2].

In this project, I will utilize both supervised and unsupervised method, and generate the final result by taking majority vote from both supervised and unsupervised method.

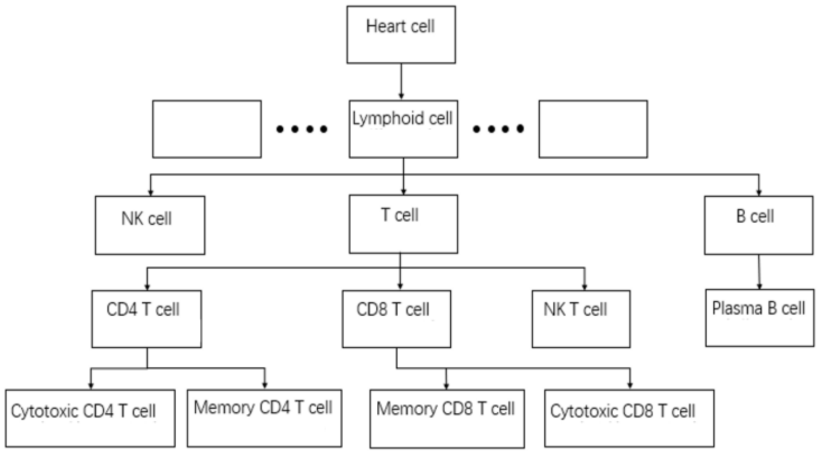

Figure 1: Tree of Heart Cell Types

3 Problem Formulation

In this study, we found that the training dataset consists of over 700,000 cell samples and the test dataset consists of over 62,000 cells, indicating that the training data is sufficient. Furthermore, the number of donors for the cells is 14. Additionally, the number of genes per single cell is over 43,000. Lastly, the sequencing method used to generate the training set cells is only 10X, but the test set includes cell data generated using another sequencing method called smart SEQ. Therefore, based on these four observations, we conclude that the challenge presents the following objective conditions.

The first challenge is the large amount of data, with over 700,000 cells in the training set and over 40,000 genes per cell. This results in a matrix of over 700,000 x 40,000 cells, which requires over 200GB of memory and may impact the speed of the algorithm if used for training.

The second challenge is the sparsity of the data distribution, as we found that approximately 20,000 genes have a value of 0 in all cells. This high frequency of zeros may affect the performance of algorithms such as neural networks or decision trees.

The third challenge is the long-tailed distribution of the data, as seen in the pie chart on the right. The number of cells in different cell types is not uniform, and this uneven distribution may affect the generalization performance of the model. Additionally, the test set includes cell data generated using another sequencing method called smart SEQ, in addition to 10X. The distribution of gene data generated by this new sequencing method is likely to be different from that of other methods, and this difference in distribution may affect the generalization performance of the model.

The fourth challenge is the presence of an ”unknown” category in the test set that does not appear in the training set. This mimics what would happen in the real world. This presents a challenge for supervised learning algorithms such as neural networks.

To address the challenge of the large amount of data, we applied data preprocessing techniques. To address the impact of the long-tailed distribution and the new sequencing method, we added RELU and dropout layers to the neural network architecture. To address the new cell categories, we also used unsupervised methods to optimize the performance of the neural network, which resulted in significant improvements. Let me first introduce our preprocessing techniques for the cells.

4 Experiments and Main Results

4.1 Data Preprocessing

We found that the number of genes with no expression in the dataset is quite high. For example, there are genes that have no expression in any of the cells in the entire dataset. In the test set of over

40,000 genes, this situation occurred for over 20,000 genes, so we performed gene filtering. We then removed genes that had fewer than one expression in both the training and test sets. This left us with over 21,000 genes, which we then normalized based on the total number of molecular tags (UMIs) per cell. We then selected genes with a higher mean and variance, and we were left with 4,400 genes. Finally, we standardized and scaled the data to unit variance and zero mean before proceeding with training.

It is worth noting that we were left with only 4,400 genes out of the initial 48,000+ genes, reducing the amount of data by approximately 1/10.

4.2 The Supervised Model

We initially designed the first version of the neural network architecture, which consisted of five fully connected layers and a softmax output layer that outputs the probability of each class. However, this initial neural network suffered from severe overfitting. For example, the training set had an accuracy of 95%, but the test set only had an accuracy of 82%. Therefore, we made some modifications to this simple neural network architecture to prevent overfitting.

We added a ReLU activation layer and dropout operation after every fully connected layer. The computation for the ReLU activation layer is as follows: when a parameter comes in, if it is less than zero, it is set to zero; if it is greater than zero, it is set to the original value. Compared to other commonly used activation functions, such as sigmoid, the ReLU activation function has fewer problems with gradient disappearance, and it only involves comparisons, addition, and multiplication, making it more computationally efficient. We also adopted the dropout operation, which randomly removes a certain number of units from the neural network during training. The advantage of this is that it reduces overfitting, and it is more effective than other regularization methods, such as L2 or LASSO.

As can be seen, after incorporating the packet capturing operation, the accuracy of our test instrument has improved by two percentage points. However, we can also observe that the validation machine has a 93% accuracy test and 84% accuracy, indicating that there are still certain challenges and issues to be addressed.

Table 1: Accuracy

	Train Accuracy	Val Accuracy	Test Accuracy
Fully-Connected	95	87	82
FC with ReLU, Dropout	95	93	84

Next, we implemented a five-fold cross-validation scheme for further optimization. In five-fold cross-validation, the data is divided into five folds, and the model is trained and evaluated five times, each time using a different fold as the validation set and the remaining folds as the training set. The final class is the mass of the classes obtained from each of the five iterations. Ultimately, we observed a significant improvement of approximately five points on the test set, indicating enhanced generalization performance. Five-fold cross-validation has the following advantages:

1. It maximizes data utilization by using all the data for training.
2. It creates a validation set of equal size to the training set, improving the statistical reference significance of the validation set during hyperparameter tuning and ensuring generalization performance.
3. Each model receives independent weights for model fusion, and during inference, voting inference is used to ensure the generalization performance of the model.

Table 2: Accuracy

	Train Accuracy	Val Accuracy	Test Accuracy
FC with ReLU, Dropout	95	93	84
5-fold cross validation	95	93	89

Subsequent analysis revealed that gradients and parameters within the neural network were sparse, prompting the need for optimization strategies specific to the neural network as well as other components. An important concept in neural network training is the optimization function, which serves to optimize network parameters. In order to effectively optimize network parameters, it is necessary to choose an appropriate optimization function.

Adam (Adaptive Moment Estimation) is an optimization algorithm that is widely used in deep learning to train neural networks. It combines the strengths of the momentum and RMSprop optimization algorithms, and can often outperform these algorithms in practice.

One potential issue with Adam is that it can get stuck in local minima when the learning rate is set too low. This is because Adam uses a moving average of the past gradients to compute the update to the model's weights, which can dampen the magnitude of the updates and slow down the convergence of the optimization process. [3]

AdamW (Adam with Weight Decay) is an optimization algorithm that is similar to the Adam algorithm, with the addition of weight decay, which is a form of regularization that helps prevent overfitting by adding a penalty to the model's loss function for large weights.

There are several advantages to using AdamW compared to Adam:

1. Weight decay: AdamW includes weight decay, which can help improve the generalization performance of the model by preventing overfitting.
2. Better convergence: AdamW has been shown to converge faster and to better optima than Adam in some cases.
3. Stabilization: AdamW can be more stable and produce better results than Adam in some cases, particularly when training large and complex models.
4. Robustness: AdamW is more robust to the choice of initial learning rate and is less sensitive to the choice of hyperparameters.

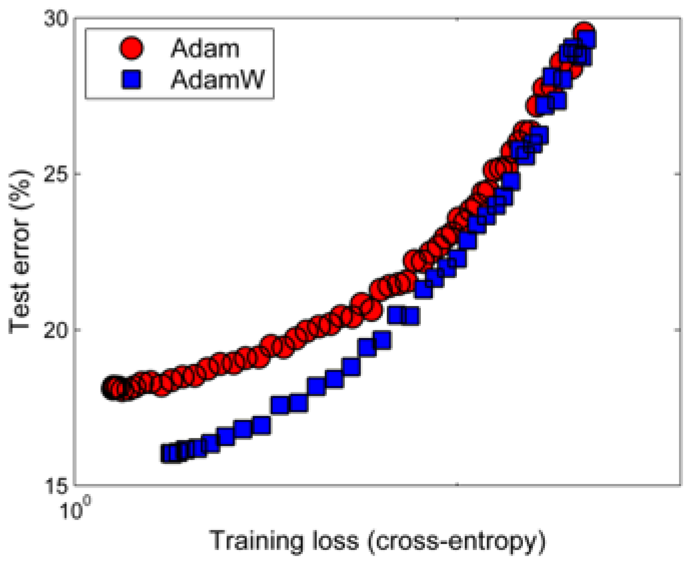

Figure 2: Adam vs AdamW

4.3 The Unsupervised Model

Clustering is an unsupervised learning method that can divide data into different categories without prior knowledge of the labels for these categories. One advantage of clustering is that it allows for automatic grouping of data and does not require human intervention. This makes clustering particularly suitable for datasets with missing or unknown labels, as it can help identify similarities and relationships among the data.

In addition, clustering has the following advantages:

1. Clustering is fast, as it does not require pre-labeling of the data or extensive computation.

2. Clustering can discover latent structures in the data and can be applied to various data types.

3. Clustering can handle high-dimensional data, as it does not require linear partitioning of the data.

4. Clustering can adapt to changes in the distribution of the data, as it does not require human intervention.

In summary, clustering is a very useful unsupervised learning method, particularly in cases where data labels are missing or unknown. It can quickly discover latent structures in the data and can be applied to various data types.

Our proposed unsupervised method involves the following process: we have a portion of raw data, which is passed through a supervised neural network for inference training, resulting in an inference result. We then use unsupervised clustering to obtain a clustering result. For example (in Figure 3), we use clustering to obtain two clusters, red and blue. We then use the cell identifiers to find the cell classification that the supervised neural network has already made from the inference result, such as in the blue cluster, four cells, three of which were actually inferred as T cells and one as an NKT cell. In the red cluster, one was inferred as an NK cell, one as a B cell, and one as an NKT cell. We then observe that 3/4 of the cells in the blue cluster are T cells, so we define it as a T cell, and in the red cluster, the proportions are roughly equal, so we define it as a new cell type because the neural network may not accurately infer which type of cell it is. The new cell does not appear in the training set, so it must perform the worst in the neural network.

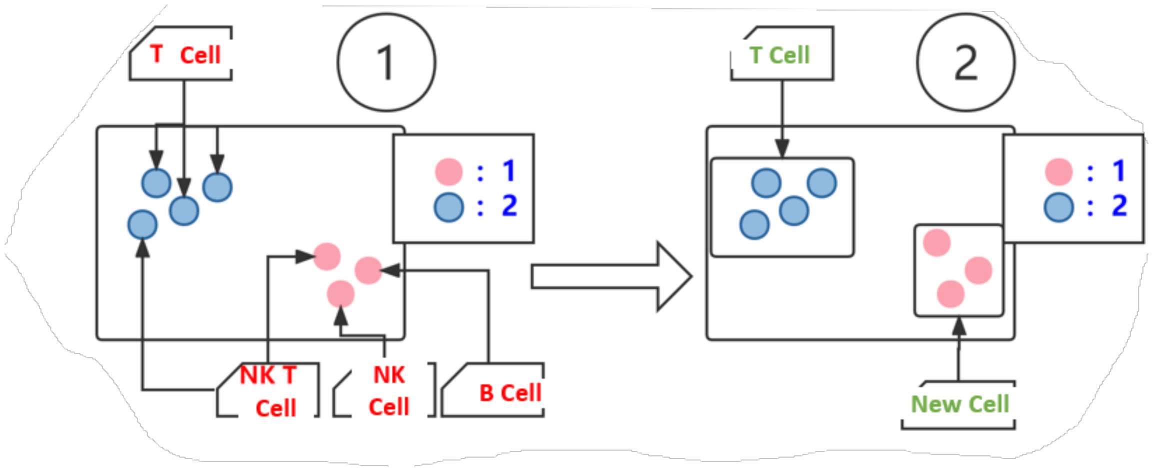

Figure 3: Clustering

In unsupervised clustering, our initial approach was to use unsupervised methods to classify and optimize all data, but this approach did not consider batch effects, resulting in disorganized classification outcomes as shown in the left figure. To address this issue, we proposed a solution that involves performing unsupervised classification on each donor and then allocating optimization. As demonstrated in the right figure, each class is relatively well-separated, indicating the success of our unsupervised clustering. Two additional examples, shown in the left and right figures, also display well-separated classes for different donors, resulting in approximately 0.2% improvement with this approach.

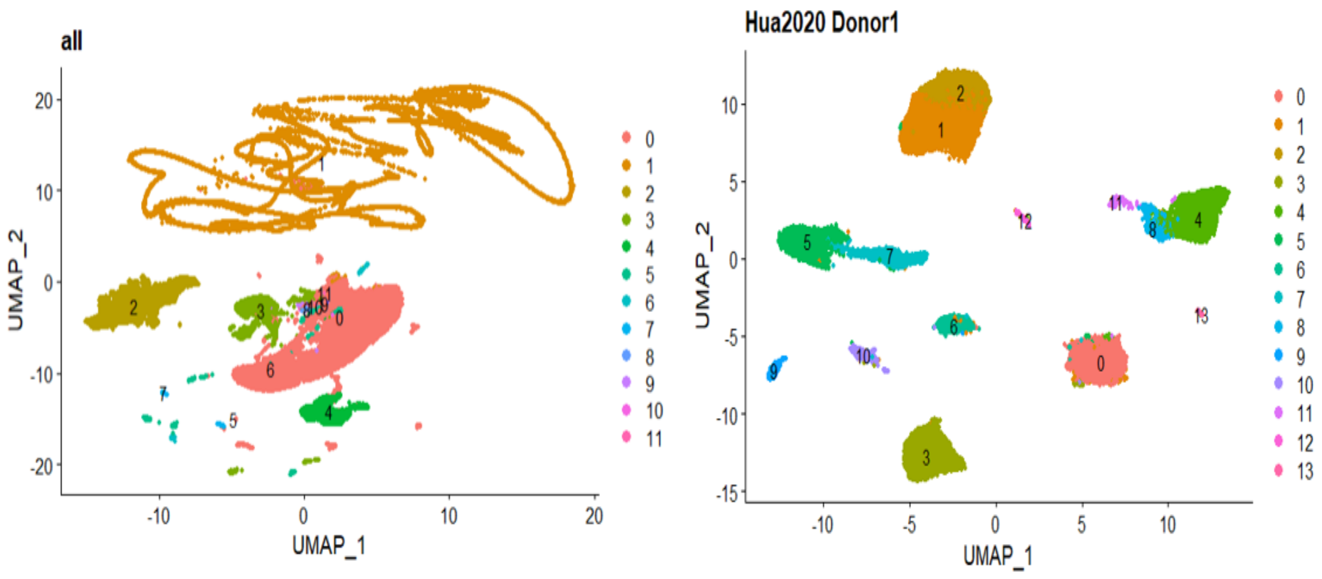

Figure 4: Clustering2

5 Conclusion

Table 3: Accuracy

ReLU, Dropout	5-fold	AdamW	Unsupervised Optimization	Test Accuracy
				82.3%
✓				84.2%
✓	✓			88.7%
✓	✓	✓		90.1%
✓	✓	✓	✓	92.4%

This is the result of our ablation experiment. ReLU and DropOut resulted in a 1.9 improvement. Cross-validation resulted in a 4.5 improvement. The AdamW optimizer resulted in a 1.4 improvement, and finally, unsupervised optimization resulted in a 2.3 improvement, resulting in an overall accuracy of 92.4.

In summary, our approach combines supervised and unsupervised methods based on machine learning, taking into consideration the strong fitting capabilities of deep learning and the interpretability of clustering methods. We also preprocessed the data to reduce computational requirements. Finally, we optimized the neural network algorithms to achieve better generalization. One limitation of our approach is that we still observed overfitting, such as a training accuracy of 95% but a test accuracy in the 90s.

Acknowledgement

Special thanks to my friend Bizhe Bai on providing thoughts and computing resources in this project.

References

[1] B. et al. Xie. “Automatic cell type identification methods for single-cell RNA sequencing”. *Computational and Structural Biotechnology Journal* (2021) (cit. on p. 1).

[2] W. et al. Ma. “Evaluation of some aspects in supervised cell type identification for single-cell RNA-seq: Classifier, feature selection, and Reference Construction”. *Genome Biology. BioMed Central* (2021) (cit. on p. 1).

[3] Ricardo Llugsi, Samira El Yacoubi, Allyx Fontaine, and Pablo Lupera. “Comparison between Adam, AdaMax and Adam W optimizers to implement a Weather Forecast based on Neural Networks for the Andean city of Quito”. In: *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*. 2021, pp. 1–6 (cit. on p. 4).