PREDICTING PATIENT-LEVEL PHENOTYPES FROM SINGLE-CELL DATA

Mike Fu (xf2209), Chinyere Ihuegbu (coi2002), Amritha Musipatla (sm3773)

Columbia University in the City of New York
Department of Computer Science

ABSTRACT

Deep learning approaches have seen increased use in several functional genomics applications, often meeting or exceeding the performance of state-of-the-art methodologies [1]. In this project we deploy three similar deep neural network architectures to predict patient-level phenotypes from single-cell mass cytometry data. First, we use a three level DNN to classify individual cells in a group of bone marrow B-Cells as either unstimulated or stimulated with B cell receptor (BCR) cross-linking antibodies (BCR stimulated B cells). We find that the DNN is able to differentiate between between BCR stimulated vs. unstimulated B-Cells with higher accuracy than several machine learning baseline models, and with greater clarity than traditional flow cytometry gating techniques. Our second DNN uses a pared down list of protein markers, and incorporates cell features that have been extracted via the Wishbone algorithm [2]. We are able to show that combining Wishbone extracted features with the measured flow cytometry features results in a more accurate classifier than training with the flow cytometry features alone. We run this model on surface cells filtered from the the full dataset, and then train a separate model on signaling cells filtered from the full dataset. Finally, we develop a third architecture, which combines the surface cell trained model and signaling cell trained model, and find we are able to improve overall accuracy on this task compared to the separate models.

1. INTRODUCTION

High-dimensional single-cell technologies (such as single-cell RNA sequencing, single-cell ATAC-sequencing, mass cytometry, among many others) have become useful tools in understanding the heterogeneity and molecular profile of cells in both health and diseased states [3], as a result these technologies now play an important role in the development of more effective and precise therapies to diseases such as cancer [4, 5]. Single-cell technologies also enable the detection and description of diverse cell population in both healthy and diseased states - including the identification of rare cell populations which traditional biological experiments may fail to detect. This ability to map out the genomic and proteomic profile of cells in different states, such as in development, healthy and diseased states, is crucial in improving our understanding of cellular processes and the human biology .

Unsupervised clustering is a machine learning approach, which utilizes single-cell omics analysis to identify and define cell clusters (and subsequently cell types) [6, 7, 8] . While this learning approach has aided our understanding of the heterogeneity of cell types in development, health and diseased states, unsupervised clustering is still limited in directly associating identified cell clusters with disease status, as this is usually done manually with the aid of literature, wet-lab experiments and expertise knowledge. Furthermore, with unsupervised learning associating cell cluster to a disease phenotype usually requires an additional step of condition-specific differences introduced in wet-lab experiments (or differences introduced as a function of individuals having a disease and thus altered biology relative to healthy individuals),which in turn is represented in unsupervised clustering of single-cell data by the increased or decreased abundance of a given cell cluster(s) compared to healthy controls or healthy individual.

The limitation of unsupervised clusterings' inability to predict phenotype(s) (such as disease condition, expected survival, developmental stage) from single-cell data is one which is expensive and time consuming when relying on the combination of literature, expertise knowledge and wet-lab experiments to associate cell types with phenotypes.

Our project aims to address this problem using supervised learning, specifically deep convolutional neural networks with multiple layers and transfer learning. In a recent study done by Arvaniti and Claassen, a computational framework, called CellCNN, was developed to facilitate the detection of rare- disease associated subsets and their phenotype [9]. CellCNN combined multi instance learning and convolutional neural network using a single layer to identify cell clusters and also the associated disease status of clusters. Our project aims to further extend on this using additional techniques such as dimensionality reduction and saliency maps to extract information which could then be used for predictions.

2. RELATED PAPERS

2.1. Deep Learning for Phenotype Prediction

Recent applications of convolutional neural networks (CNNs) on single-cell inputs have demonstrated CNN utility for both the classification (multi-node output) and regression (single node output) task.

Specifically, Claasen et. al adapted CNN architecture for processing groups of single-cell inputs (collectively, multi-cell inputs), where each of these multi-cell inputs was associated with a phenotype [9]. The group was able to demonstrate efficacy of their CellCNN architecture in a variety of phenotype prediction tasks. Their architecture was used to classify whether peripheral blood mononuclear cells (PBMC) samples had been stimulated or unstimulated by paracrine agents, based on the samples intra-cellular markers. Additionally, the group was able to use CellCNN for the regression task, to identify T-cell subpopulations that were at higher risk of AIDS onset following infection to HIV. CellCNN was shown to predict comparably to state-of-the-art approaches, with lower computational costs.

Chen et. al. were able to use a deep learning approach to predict gene expression patterns of cells [12]. Their model, dubbed D-GEX, predicts the expression of a "target" gene based on known expressions of a group of "landmark" genes. Comparing their RNN based models to other state-of-the-art methods (linear regression), resulted in D-GEX significantly outperforming other methodologies.

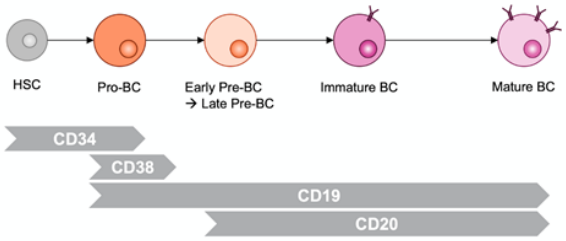


Figure 1: Schematic Representation of B cell development

3. METHODS

Our Python code and instructions to train and classify can be found on GitHub, along with notebooks for our exploratory data analysis and baseline models:

https://github.com/a-musipatla/predicting-patient-level-phenotypes-from-single-cell-data.

3.1. Datasets

A study done by the Pe'er and Nolan lab groups employed mass cytometry to study the development trajectory of B cells, as well as the behaviour of B cells in response to stimuli [10, 11]. As part of their experiment, B cell centric mass cytometry data were collected from human bone marrow. Their dataset includes records of multiple immune and B cell cellular features, such as activation of regulatory signaling molecules, and phenotypic protein markers of B cell development in both unstimulated and BCR stimulated conditions [10]. We will be using their publicly available data of mass cytometry of unstimulated and stimulated conditions from the same group to reduce technical variations such as batch effects.

We develop and train our model using mass cytometry data of B cells under unstimulated and stimulated conditions. Since the development of B cells and its response to stimuli results in differential expression of specific B cell proteins and phenotypes (Fig 2), we would test the sensitivity and efficiency of our model in detecting phenotype(s) associated with observed cell types. This intended data set is already publicly available on https://reports.cytobank.org/1/v1.

The structure of the cytometry data is laid out in Figure 2. Each cell measurement consists of 33 float value features, indicating measurements of 33 different protein markers in the cell. Each cell is labeled as BCR stimulated or unstimulated.

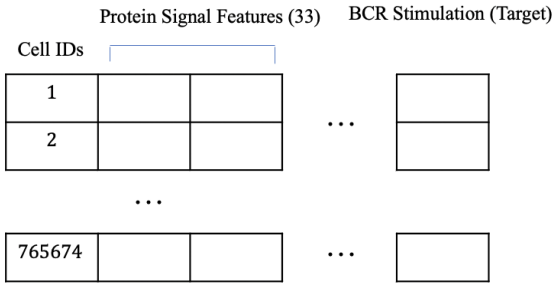
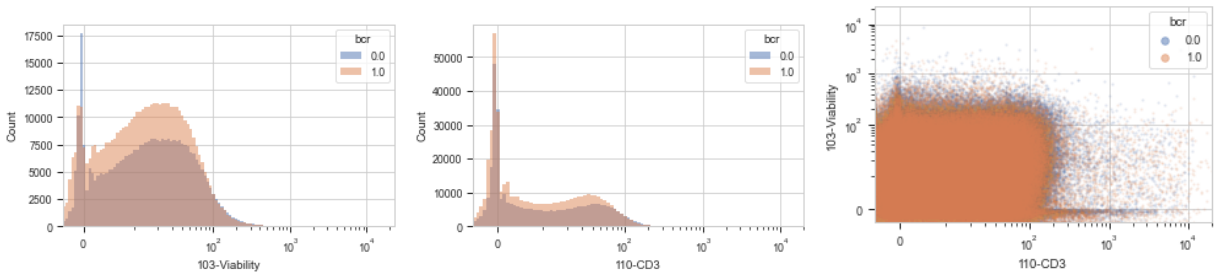


Figure 2: Flow Cytometry Data Structure

Our networks will take in single cell data as input and attempt to identify the phenotype-associated cell subpopulations. We will be using supervised learning via variety of different model architectures.

3.2. Exploratory Data Analysis

The Pe'er Lab's B cell data was processed into single-cell inputs with corresponding phenotype classification (in this case, B cell response to stimuli). Preliminary exploratory data analysis of the bone marrow B-cell data shows that 38 cellular features were measured via flow cytometry. We use the open source python library Cytoflow [13] to extract and plot the cytometry data. We examine histograms of measurements of specific phenotypes for both the BCR stimulated and unstimulated cells, Figure 3 (a) and 3 (b) plot measurements of two example measurements for both the stimulated and unstimulated populations. We also plot cells'



(a) (b) (c)

Fig. 3. We plot the flow cytometry measurements of protein markers for single-cell measurements out of the B-cell bone marrow data [10, 11] (a) Cell count for phenotype : 103-Viability (b) Cell count for phenotype : 110-CD3 (c) Scatter plot showing single cells' measurements of Protein Marker 110-CD3 and 103-Viability

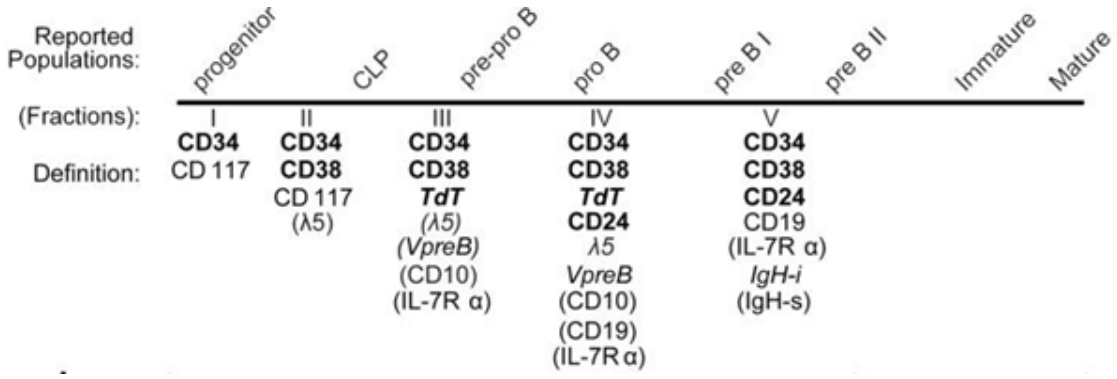


Fig. 4. Significant Protein Markers used to Categorize B-Cell Development Stage [11]

measurements of selected phenotypes against their measurements of other phenotypes in scatter plots, Figure 3 (c) records an example scatter plot featuring cells' 103-Viability measurement against their 110-CD3 measurements.

3.3. Preprocessing and Feature Engineering

We attempt feature engineering techniques to both improve performance and reduce training time. Dimensionality reduction techniques such as linear discriminant analysis and principal component analysis (PCA) have been used to reduce the raw data size (and by extension the computational cost) of genomics data sets [14]. Outside of functional genomics, dimensionality reduction of input data has been shown to improve performance of image classification models. We implement principal component analysis to transform our data to a lower dimension (20 features) before training.

3.4. Extracting Wishbone Features

Another area of interest was including the original findings of the Pe'er Lab group as features in our model. The Pe'er Lab's Wishbone [2] is a trajectory-detection algorithm able to order cells along their trajectory during development, identify branching points cells undergo during development, and associate cells with their respective developmental branches. Wishbone takes multiparameter single-cell events as input, and maps them onto a cell developmental trajectory to study how functionally competent mature cells [2, 11].

In this project, we employed the Wishbone algorithm in performing the trajectory analysis of B cells, because of its ability to accurately identify and associate cell branches during development. As input to Wishbone, the data was first pre-processed. The ion count matrix stores relevant metadata where each column represents a distinct cell feature labelled with a distinct isotope-antibody measured (such as cell surface protein, or signalling protein) and each row represents a single mass scan of the detector, resulting in a single-cell resolution data.

Conventionally the ion count matrix is Arcsinh transformed. Following Arcsinh transformation, the data in both the untreated sample and BCR treated sample were filtered for cells in the B cell lineage by removing other immune cell types based on the expression of their surface marker proteins (such as myeloid cells and macrophages), and then positively selecting for CD38+, CD19+ and CD20+ cells in the data sets, which are surface marker proteins corresponding to the B-cell lineage. Following this the B cell surface proteins were separated from the B cell signalling proteins in both untreated and BCR treated samples, generating four B cell sample groups.

A Wishbone object was created by reading the expression ion count matrix of the pre-processed B cell data, after which a tSNE dimensionality reduction was further applied to the data. The diffusion map was then implemented, and relevant Eigen values calculated. Following this, the start cells and waypoints for each B cell group were manually assigned based on the expression of markers such as CD38 and Ki67, after which the trajectory of branch points and branch associations were calculated. This allowed for the change in the normalized expression of B cell surface markers and signalling markers between untreated and BCR treated samples to be compared, as well as the developmental trajectory and branch associated with each cell. *(Special thanks to Paula Josefine Schultheiss for her kind assistance in pre-processing the data and analyzing the data using Wishbone).*

We take each cell's developmental category as classified by the Wishbone algorithm and include this in training our model.

Another aim we had was to try to independently develop a NN model to classify cells into their developmental category, using the Pe'er Lab classifications as truth data. As shown in Figure 4, the protein markers '103-Viability', '148-CD34', '167-CD38', '142-CD19', and '147-CD20' show a correlation to a B cell's developmental stage.

3.5. Baseline DNN Model Architecture

For our baseline network, we use a deep neural network (DNN) with a series of dense layers. In the case of CellCNN, the architecture consisted of max-pooling and mean-pooling layers following a convolutional layer. For our architecture, we expanded the number of layers, and implemented the NN using the TensorFlow Keras library. We include a dropout of 10% to prevent overfitting. Figure 5 shows the baseline architecture used.

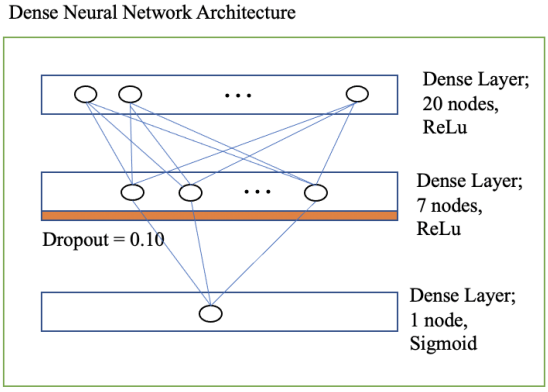


Figure 5: High Level DNN Architecture used for BCR Classification

To train this baseline model, we implement a training/validation split, with a default 10% test set held out for performance evaluation. Our initial data is loaded into a

Pandas dataframe using the Cytoflow [13] library for flow cytometry data. This dataframe consists of protein signal features, and a stimulation level target. We preprocess and convert the dataframe into a TensorFlow dataset, and batch data into our DNN model. The model is trained for 15 epochs.

3.6. Hyperparameter tuning

We tuned our model on three hyperparameters: the dropout percentage, the number of layers in the neural netowrk, and the number of dimensions to reduce to for PCA. Due to the size of the data, each hyperparameter optimization is only ran on 15 epochs. with a batch size of 128. For each hyperparameter, the loss is calculated over 3 random starts to reduce variance.

3.6.1. Dropout

For dropout, we tested dropout percentages from 0 to 0.5.

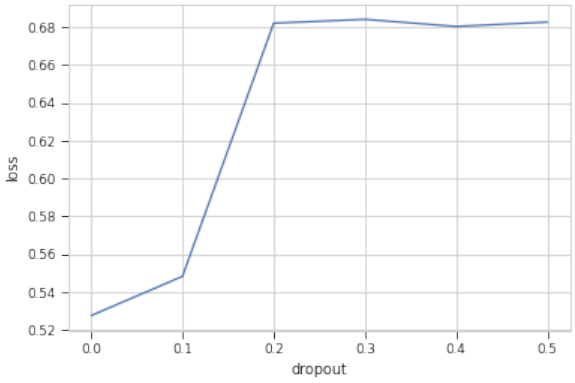


Figure 6: Having a dropout above 0.1 significantly lowers the network's ability to learn

We found that having a dropout of 0.1 is the most ideal, as it prevents overtraining without losing too much information in the nodes.

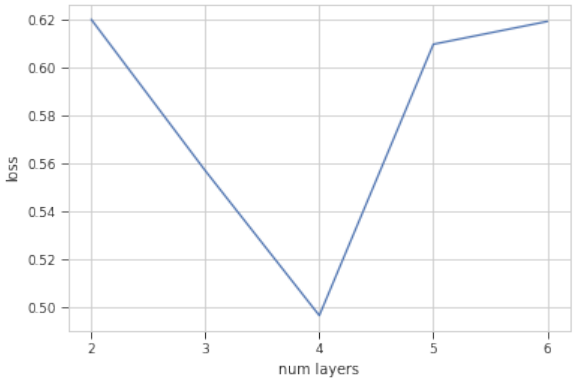


Figure 7: Having a layer of 4 is a good compromise between learning the data and overfitting

We decided to use four layers in our final model as it decreases loss by a significant margin.

3.6.2. PCA

For PCA dimensions, we tested between 5 to 25 dimensions. We ultimately decided on using 20 dimensions using the Elbow Method.

Figure 8: A low dimension reduction does not capture the data properly

3.7. DNN with Wishbone Features Integrated

In addition to the baseline model described in section, we also develop two 4-layer neural network models that we train exclusively a surface cell and signaling cell subset of the total data. We tested for whether adding the wishbone features improved overall model performance. Figure 9 shows the DNN architecture used for both the signaling cell trained model, and the surface cell trained model.

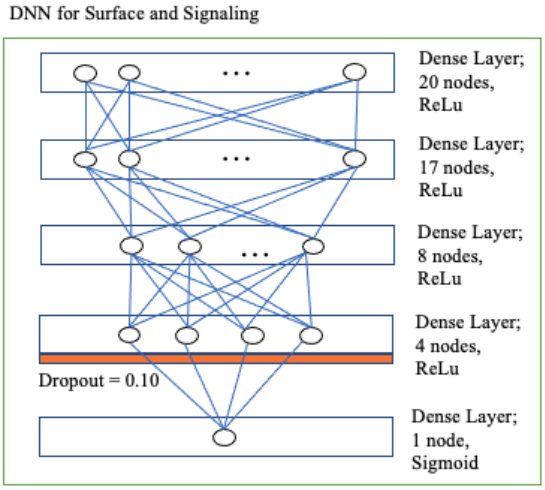


Figure 9: Model Architecture used to incorporate the Signal and Surface Cell data extracted by Wishbone

3.8. Transfer Learning through a Combination of Models

Using the two networks trained on signalling and surface cells separately, we constructed a concatenated network to differentiate between stimulated and unstimulated cells. To construct this network, we took the dropout layers of the signalling and surface cells, and performed a concatenation. Following this, we appended two more dense layers of 16 and 4 nodes, in that order. Lastly, we added a sigmoid node to differentiate the boundary.We theorized that splitting the cells beforehand depending on if they're surface cells or signalling cells may improve model performance.

The model architecture can be seen in Figure 10.

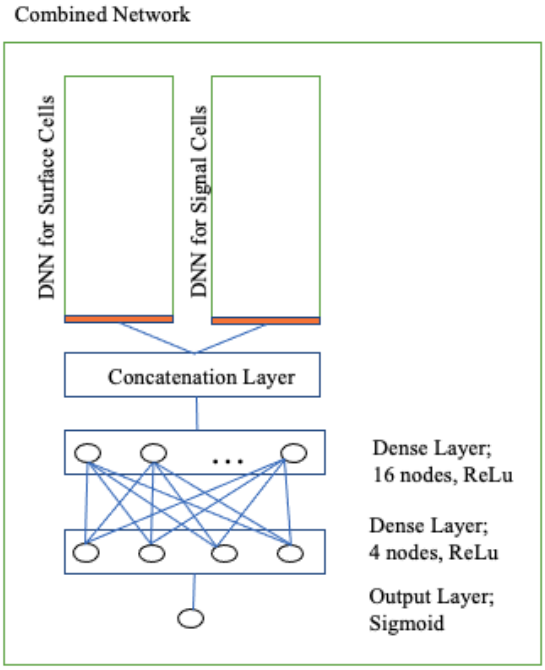


Figure 10: Model Architecture for our Combined DNN Models. The DNN architecture for signaling and surface cells are shown in Fig. 9.

4. RESULTS

4.1. Baseline Regression Model Results

From our preliminary data analysis, we noted that the spread of measurements cannot easily be correlated to whether the cell was stimulated or not, ie. the measurements for these cells are not well-separated, for that reason we chose to forego using a Gaussian Mixture Model as our baseline. Instead we developed and ran a baseline logistic regression model, which achieved an accuracy of 82% in categorizing cells as stimulated and unstimulated. Our baseline random forest scored an accuracy of 86%. Our baseline SVM model scored a baseline of 75%. Table 1 records our best accuracy for each of these baseline models. Figure 10 shows our training curve for our logistic regression model.

Logistic Regression	Random Forests	Support Vector Machine
82.5%	86.7%	75.6%

Table 1. Measured classification accuracy of baseline models, trained on the full B-Cell Bone Marrow dataset.

4.2. DNN Results

4.2.1. Classifying Cells by BCR Stimulation

Our DNN exceeds performance over our baseline models in determining whether or not a cell has been stimulated with BCR or unstimulated. We achieve 91.1% accuracy on a 10% test set hold out.

4.2.2. Classifying Cells by Wishbone Trajectory

We found that incorporating the Wishbone filters as additional features improved the accuracy of our modified Wishbone model, we achieved an additional accuracy improvement of 10% on a held out test set.

Surprisingly, using no Wishbone with PCA started off with a lot lower mean squared error compared to the other two models. As expected, using wishbone features decreases the starting loss, as well as produces the best performing model. A comparison of training curves is shown in Figure 11.

4.2.3. Classifying Cells using Transfer Learning

Unsurprisingly, using transfer learning lowers the starting loss. Moreover, it is able to achieve to achieve a higher validation accuracy compared to the baseline model. Even more surprisingly, the accuracy outperforms the accuracy of both component models, the signalling and surface models, at 0.892% and 0.884%, respectively. Figures 12 and 13 show the improved performance of our transfer (combined) model compared to our baseline model.

4.3. Protein Marker Importance

Our baseline models also give us an insight on the relative importance of each protein marker. The top 4 prominent cell surface protein markers we found were the 114-CD3, 112-CD3, 150-pSTAT5, 154-pSHP2, 110-CD3.

However, using only these features did not produce a robust model.

As expected, including the wishbone features increased model performance for surface cells. Surprisingly, using

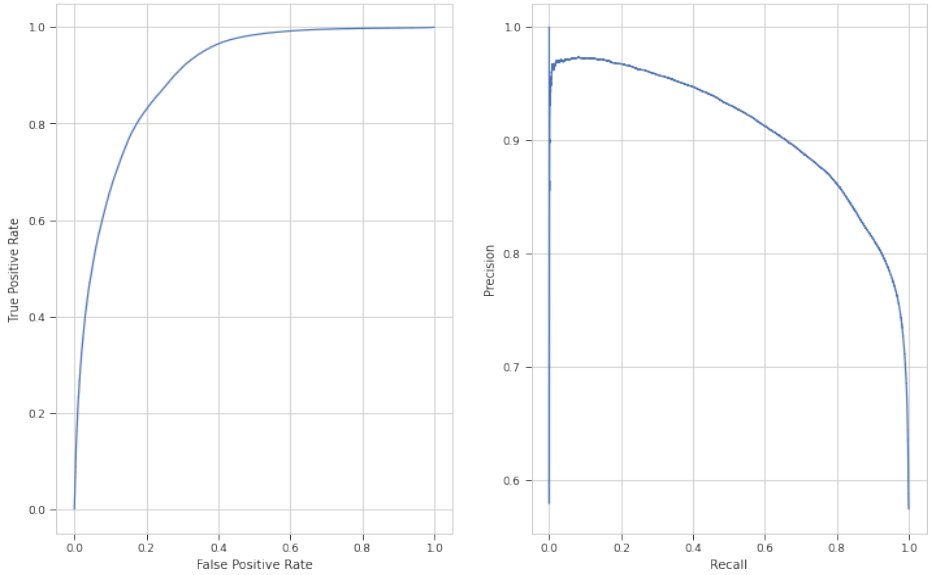


Fig. 10. Precision Recall curve and ROC curves for logistic regression. Our baseline achieved an AUC of 0.78 and 0.83 respectively.

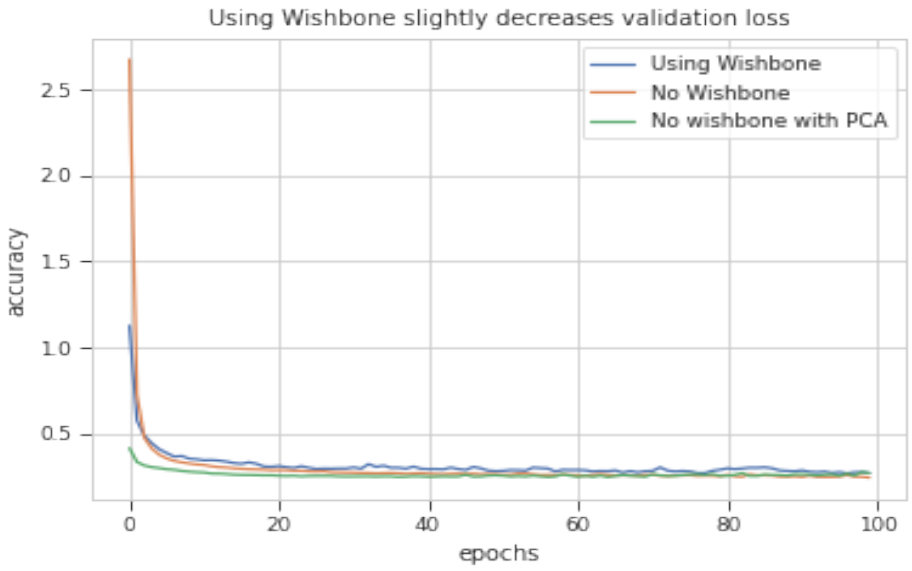


Fig. 11. Using wishbone slightly improves model performance compared to using the baseline model and performing a dimension reduction

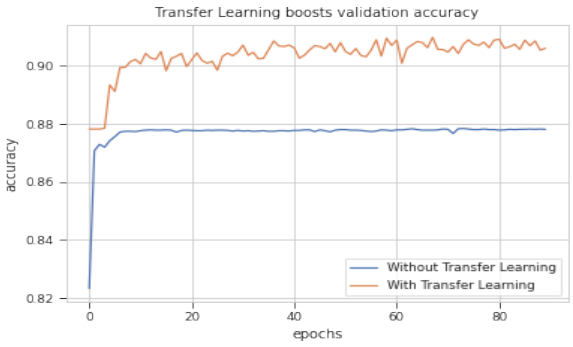


Fig. 12. Transfer outperforms the baseline model in loss

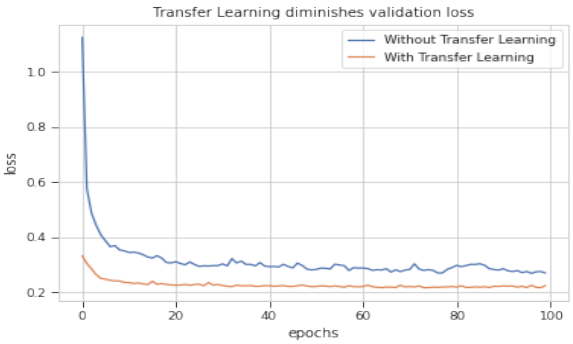


Fig. 13. Transfer learning outperforms the baseline model in accuracy

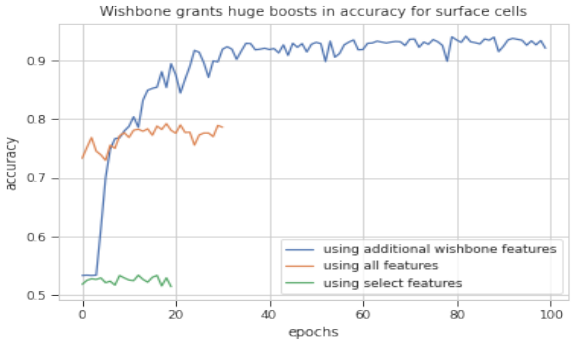


Fig. 14. Validation accuracy for using the wishbone features, original data, and only relative important markers. The difference in epochs trained is due to early stop loss

only the prominent cell surface proteins was not able to successfully differentiate between stimulated and unstimulated cells.

5. DISCUSSION

We have developed neural networks that are able to classify flow cytometry cell measurements by phenotype, based on protein marker features. We have also shown that a DNN based classification model can be more effective at this task than simpler machine learning models. We have also shown that we can effectively incorporate the Wishbone algorithm's extracted features into our training data to produce a more accurate model.

5.1. Model Improvements

In addition to the hyperparameter tuning as described in Preliminary Results, we could further improve the performances of our NN model by using ablation, where specific features/nodes are removed and the change in accuracy is measured to see how the model performs in the presence vs absence of selected features/nodes.

5.2. Training Methods for Genomics Problems

One potential issue we ran into was the sheer amount of the data available. Training our baseline logistic regression took over an hour on CPU, with other baseline methods, such as random forests and support vector machines taking even longer. One potential fix we came up with is to train only on 50% of the available training data to fine tune our model architecture and hyperparameters.

5.3. Potential Applications

Further application of neural networks to flow cytometry measurements may yield classifier models that are capable of differentiating between cell groups better than possible with flow cytometry gating techniques alone. As we noted with the B-cell bone marrow dataset explored here; a dataset with cytometry measurements that are not easily separable can still be classified with a DNN. This technique may lend itself to differentiating between phenotype groups with similar protein signal markers.

6. REFERENCES

[1] Lefteris Koumakis, "Deep learning models in genomics; are we there yet?," *Computational and Structural Biotechnology Journal*, vol. 18, 6 2020.

[2] Reich-Zeliger S Angel O Salame TM Kathail P Choi K Bendall SC Friedman N Pe'er D Setty M*, Tadmor MD*, "Wishbone identifies bifurcating developmental trajectories from single-cell data," *Nature Biotechnology*, 2016.

[3] Edoardo Galli, Ekaterina Friebel, Florian Ingelfinger, Susanne Unger, Nicolás Gonzalo Núñez, and Burkhard Becher, "The end of omics? high dimensional single cell analysis in precision medicine," *European Journal of Immunology*, vol. 49, no. 2, pp. 212–220, 2019.

[4] Tian Q.-Price N. D. Hood L. Yurkovich, J. T., "A systems approach to clinical oncology uses deep phenotyping to deliver personalized care.," *Nature reviews. Clinical oncology, 17(3), 183–194.*, 2020.

[5] Iorgulescu J. B. Braun D. A. Keskin D. B. Livak K. J. Gohil, S. H., "Applying high-dimensional single-cell technologies to the analysis of cancer immunotherapy.," *Nature reviews. Clinical oncology, 10.1038/s41571-020-00449-x. Advance online publication.*, 2020.

[6] Xiaoshu Zhu, Hong-Dong Li, Lilu Guo, Fang-Xiang Wu, and Jianxin Wang, "Clustering of single-cell rna-seq data by unsupervised learning methods," *Current Bioinformatics*, vol. 14, 11 2018.

[7] Andrews T. S. Hemberg M. Kiselev, V. Y., "Challenges in unsupervised clustering of single-cell rna-seq data.," *Nature reviews. Genetics, 20(5), 273–282.*, 2019.

[8] Hockley J. Görnitz N. Vidovic M. M. Müller K. R. Gutteridge A. Ziemek D. Mieth, B., "Using transfer learning from prior reference knowledge to improve the clustering of single-cell rna-seq data.," *Scientific reports, 9(1), 20353.*, 2019.

[9] Claassen M. Arvaniti, E., "Sensitive detection of rare disease-associated cell subsets via representation learning.," *Nature communications, 8, 14825.*, 2017.

[10] Kara L. Davis El-ad David Amir Michelle D. Tadmor Erin F. Simonds Tiffany J. Chen Daniel K. Shenfeld Garry P. Nolan Bendall, Sean C. and Dana Pe'er., "Single-cell trajectory detection uncovers progression and regulatory coordination in human b cell development," *Cell 157 (3): 714–25.*, 2014.

[11] Davis K. L. Amir-e. Tadmor M. D. Simonds E. F. Chen T. J. Shenfeld D. K. Nolan G. P. Pe'er D. Bendall, S. C., "Single-cell trajectory detection uncovers progression and regulatory coordination in human b cell development.," *Cell, 157(3), 714–725.*, 2014.

[12] Rajiv Narayan Aravind Subramanian Xiaohui Xie Yifei Chen, Yi Li, "Gene expression inference with deep learning," *Bioinformatics*, vol. 32, 2 2016.

[13] Brian Teague, "Cytoflow," https://github.com/cytoflow/cytoflow/, 2018, [Online; accessed 04-April-2021].

[14] Seiya Imoto Satoru Miyano Alok Sharma, Kuldip K. Paliwal, "A feature selection method using improved regularized linear discriminant analysis," *Machine Vision and Application*, vol. 25, 11 2013.