UNIVERSITY OF CALIFORNIA,

IRVINE

Machine Learning Assisted Single Cell Mechanotyping with Deformability Cytometry

DISSERTATION

submitted in partial satisfaction of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in Chemistry

by

Daniel D. Seith

Dissertation Committee:
Professor Zuzanna S. Siwy, Chair
Professor James S. Nowick
Professor Andrej Lupták

2022

DEDICATION

To my parents, family members, friends, and all others who made this work possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I want to express my deepest appreciation to my faculty advisor, Prof. Zuzanna Siwy, for her continued guidance and support in completing the projects described herein. I also acknowledge my committee members, Prof. James Nowick and Prof. Andrej Luptak, for their input, conversations, and ongoing support throughout my graduate studies. Furthermore, I would like to thank everyone in the administrative office for their help in navigating the administrative aspects of graduate school and obtaining this degree. I also want to give a special thank you to all of my coworkers, notably my group members, including Cody Combs, who I have had the privilege to work with during my time at UC Irvine and who have helped make my projects successful. Moreover, I want to acknowledge AIP Publishing for permission to include previously published material from my manuscript in Chapter 3. The research in Chapter 3 was supported by UC Cancer Research Coordinating Committee, C21CR2129. Chapter 3 was based upon work supported by the National Science Foundation under grant number 1633631. This work was supported by an opportunity award from the UCI Center for Complex Biological Systems, through NIH-NCI U54-CA217378.

VITA

DANIEL D. SEITH

EDUCATION:

- Doctor of Philosophy in Chemistry, University of California, Irvine (June 2022)
- Bachelor of Science in Chemistry, Pennsylvania State University (May 2017)

POSITIONS:

- Graduate Student Researcher, Department of Chemistry, University of California, Irvine. Research Advisor: Zuzanna S. Siwy (November 2019-June 2022)
- Graduate Student Teaching Assistant, Department of Chemistry, University of California, Irvine. (September 2017-June 2022)
- Undergraduate Student Researcher, Department of Chemistry, Pennsylvania State University.
  Research Advisor: Philip C. Bevilacqua (August 2013-August 2017)

RESEARCH PUBLICATIONS:

- C. T. Combs, Daniel D. Seith, M. J. Boyvn, S. P. Gross, X. Xie, Z. S. Siwy, Deep Learning Assisted Mechanotyping of Individual Cells Through Repeated Deformations and Relaxations in Undulating Channels. *Biomicrofluidics* (2022)
- William J. Howitz and 12 others including Daniel D. Seith; Online in No Time: Design and Implementation of a Remote Learning First Quarter General Chemistry Laboratory and Second Quarter Organic Chemistry Laboratory, *J. Chem. Educ.*, (2020).
- Daniel D. Seith Jamie L. Bingaman, Andrew J. Veenis, Aileen C. Button, Philip C. Bevilacqua; Elucidation of Catalytic Strategies of Small Nucleolytic Ribozymes from Comparative Analysis of Active Sites; *ACS Catalysis* (2017).

FELLOWSHIPS AND AWARDS:

- Graduate Dissertation Fellowship, University of California, Irvine (Spring 2022)
- Machine Learning in Physical Sciences Fellowship, National Science Foundation (Fall 2020)
- Division of Teaching Excellence and Innovation Fellowship, University of California, Irvine (Summer 2020)
- Research Experience for Undergraduates Fellowship, National Science Foundation (Summer 2015)

LICENSES:

- FAA Certified Private Pilot (Winter 2022)

ABSTRACT OF THE DISSERTATION

Machine Learning Assisted Single Cell Mechanotyping with Deformability Cytometry

by

Daniel D. Seith

Doctor of Philosophy in Chemistry

University of California, Irvine, 2022

Professor Zuzanna S. Siwy, Chair

In this dissertation, methods for characterizing cells based on their mechanical phenotypes are described. A novel microfluidic channel design is presented and data are gathered as cells pass through undulations in the channel. Deep learning methods are applied to the data in a new approach for classifying cells solely based on their mechanical properties. First, in a supervised deep learning approach, a highly interpretable random forest was created and trained to extract the most influential features for cell classification. Feature attributions of the random forest were uncovered using Shapley values. Analysis of the most influential features revealed by the Shapley values highlighted the importance of temporal features, such as the change in aspect ratio over time, in classifying cells. This led to the development of a powerful convolutional recurrent neural network, which dramatically improved classification accuracy to more than 90% when using five-fold cross-validation. Next, an unsupervised deep learning approach for cell classification was explored for problems where classes of cells are unknown a priori. Unsupervised clustering was first tested using manually extracted features and traditional clustering algorithms. However, performance was significantly improved with the development of a variational autoencoder (VAE), which extracted higher-dimensional features. The encoder for the VAE was turned into a classifier using a clustering loss function. This trained network exhibited an accuracy of up to 80% when thresholding the top $\sim$10% of the predictions.

Chapter 1

Introduction to Machine Learning

Machine learning has gained attention over the past several decades due to its potential to recognize patterns in a variety of fields and aid in tasks inaccessible to conventional computer programs. There are two very broad subtypes of machine learning: supervised and unsupervised learning. In supervised learning, a model is trained to predict known quantities given the input data. An example of a supervised learning task is training a model to identify handwritten digits using the MNIST dataset,[1] which contains a large number of handwritten digits and associated labels. In unsupervised learning, the model learns patterns in the data without supervision (i.e., without ground-truth labels). Therefore, in an unsupervised learning approach, a model can be trained to group handwritten digits into clusters using only the images of the handwritten digits in the MNIST dataset, not the corresponding numbers.

Both learning approaches require optimizing the model parameters and hyperparameters. The parameters of a model are internal and directly define how the model handles the data. Additionally, the parameters are learned from the training data. Hyperparameters, however, refer to how a model is structured (e.g., shallow vs. deep neural network) and how it is trained (e.g., high learning rate). Both supervised and unsupervised learning require parameter and hyperparameter optimization since the values are task-dependent. The parameters of a model are optimized by a computer, while the hyperparameters of a model are chosen by a human.

This chapter describes the various machine learning methods employed throughout this dissertation.

1.1 Traditional Machine Learning for Classification

Traditional machine learning algorithms, defined as learning algorithms other than deep learning, generally rely on manually curated features to form decision boundaries for use in

classification tasks. Commonly used algorithms include K-nearest neighbors (KNNs), support vector machines (SVMs), and random forests (RF). While traditional machine learning methods generally do not offer state-of-the-art (SOTA) performance, they are simple to implement, lightweight and are often interpretable. Traditional machine learning algorithms are generally used where interpretability is required. These algorithms are also often used in unsupervised clustering and are utilized in Chapters 3 and 4.

1.1.1 K-Nearest Neighbors

K-nearest neighbors (KNN) is one of the simplest clustering algorithms, and while it does not contain any parameters, it does have a single hyperparameter: a value for the 'k'-nearest neighbors. KNN is generally used for simple supervised classification. At inference time, a given point of an unknown class is compared to its k-nearest neighbors in the training set with known classes, and the algorithm assigns the class of the new point to be the mean of its surroundings. This allows the algorithm to form very complex decision boundaries. However, this does not scale well for large datasets, since the entire dataset need to be held in memory to form the decision boundary.

1.1.2 Support Vector Machine (SVM)

The support vector machine (SVM) is quite powerful and can be used for classification, regression, and clustering. Given its broad capabilities, the SVM has been applied to a variety of classification tasks, such as image classification,[2] protein classification,[3] and speech recognition.[4] Unlike a KNN algorithm, the SVM is not parameter-free. The SVM finds the largest margin between the training examples and the decision boundary. To achieve this, Boser et al.[5] created a technique where input data are non-linearly mapped to a higher-dimensional space. In this higher-dimensional space, a linear decision boundary is drawn. Critical to the success of the SVM is the kernel trick described by Boser et. al., which allows for an efficient mapping of inputs to a new space. In this new higher dimension, a decision

boundary is more easily formed.

1.1.3 Random Forest

The random forest model was first conceived in 1995.[6] Although random forests may not offer SOTA on many benchmarks, they are simple to interpret and are used where transparency is required. To build a random forest model, an ensemble of individual decision trees is constructed. The output of the random forest model is defined as an average of the predictions of the individual decision trees. In order for the ensemble to form a diverse group of models with diverse predictions, the trees must be uncorrelated. A technique known as bootstrap aggregation or bagging is used to ensure that the trees are not correlated.[7] In this technique, each decision tree will randomly sample the dataset with replacement. Since the dataset is sampled with replacement, some features will be overrepresented, encouraging each decision tree to form different models.

1.2 Traditional Machine Learning for Clustering

Clustering is the process of discovering homogeneous groups in a dataset. This is one of the most important tasks in unsupervised learning, since it can be used to uncover previously unknown groups and structure in the data. For example, types of blood cells may be classified without existing knowledge of cell types using unsupervised learning. There are a wide variety of clustering algorithms that have been developed, each of which builds clusters through a different process. Similar to classification, choosing a model is largely dependent on the dataset. These algorithms are used in Chapter 4.

1.2.1 K-Means Clustering

K-means clustering is a simple clustering algorithm that belongs to the broad family of expectation-maximization algorithms. K-means clustering is generally a baseline method because of its simplicity. To implement, the first step is to create centroids of clusters defined

by n at random locations. Then the classes of the points are assigned based on their proximity to the closest centroid, which is known as the 'M step'. After this step, the positions of the centroids are updated based on the positions of their associated points, known as the 'E step'. The E and M steps are then completed until a predetermined endpoint (e.g., max steps).

1.2.2 Gaussian Mixture Model (GMM)

GMMs also utilize the expectation-maximization framework, although they employ a more general form of the framework. GMMs have found many practical uses including aiding speech recognition,[8] and bioinformatic clustering.[9] Similar to K-means optimization, the in the E step points are assigned to classes. However, in this algorithm points are given probabilitistic assignments to each cluster instead of an assignment to exactly one cluster, as is done with K-means. This soft assignment is done by calculating the expectation of the class given the parameters ϕ, μ, and σ. Next, the M step is performed, in which the expectations calculated in the E step are maximized. Here, the values ϕ, μ, and σ are updated. Similarly to K-means clustering, the algorithm continues until it reaches a predetermined endpoint.

1.2.3 Spectral Clustering

Spectral clustering follows a different methodology compared to the previous clustering techniques. Although spectral clustering is quite costly, it is very powerful and has been used in understanding HIV mutations[10] and flow cytometry clustering.[11] Spectral clustering begins with the creation of a graph of the data, also known as an adjacency matrix. The edges of the graph are then constructed by calculating a similarity matrix through a KNN approach. Nodes are connected if they fall within each other's k-nearest neighbors. With the graph constructed, the data can now be projected to a lower dimension using the graph Laplacian. This allows the calculation of eigenvalues, which are then used to create clusters. For binary classification, nodes are assigned to clusters based on the sign of their eigenvalues: positive eigenvalues designate the node to belong to one class, and negative values designate

the other.

1.3 Deep Learning

Deep learning methods have become increasingly popular over the past decade. Although the timing of this popularity was largely spurred from hardware improvements, deep learning is advantageous due to its unique ability to 1) extract features, 2) form decision boundaries on its own, and 3) process vast amounts of data. This ability provides novel information that would likely be overlooked by a traditional machine learning algorithm or even a human. Deep learning has now expanded to dominate nearly all tasks in machine learning, such as computer vision, natural language processing, and drug discovery.[12]

1.3.1 Multi-Layer Perceptron (MLP)

Multi-layer perceptrons (MLPs) are a simple type of dense neural network. They were first theorized in 1943.[13] The MLP was originally developed for image recognition[14] but has since been generalized to function in a range of tasks when used in concert with more complex networks. The MLP comprises a single input layer, a hidden layer, and an output layer. They are termed 'dense' networks, since all nodes of a given layer are fully connected to all nodes of the next layer. An exemplary MLP is provided in Figure 1.1. Each layer contains nodes which employ non-linear activation functions that take all outputs from the previous layer as inputs. The network is trained using gradient-based optimization. The gradients of the loss function flow from the output nodes of the network to the input nodes according to the chain rule. Data can be grouped into batches that can be processed simultaneously. Here, the loss calculated is the average loss over the batch. Batching has been shown to increase training stability and decrease training time.[15]

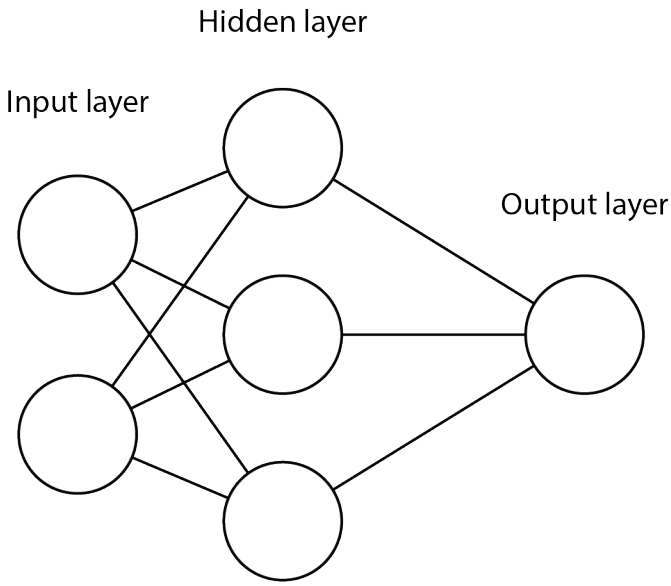

Figure 1.1: The multi-layer perceptron (MLP) architecture is shown with fully connected layers. Data flows from the input layer to the output layer.

1.3.2 Recurrent Neural Networks (RNN)

Deep learning methods are often incompatible with variably sized data. This can be an issue when dealing with data such as natural languages, which rarely have a uniform size. To address this, recurrent neural networks (RNNs) were first proposed in 1986.[16] RNNs are able to overcome this with a recurrent processing unit also known as a 'cell'. RNNs have been applied to sequential datasets, such as language translation[17] and speech recognition.[18]

RNNs sequentially process inputs and maintain an internal memory to extract temporal features from an input (Figure 1.2). Each successive input can change the internal memory and influence the output of the network. While the RNN in its most basic form is rarely used due to problems with vanishing/exploding gradients,[19] two variants: long short-term memory (LSTM)[20] and gated recurrent unit (GRU)[17] have gained popularity.

LSTMs and GRUs contain gates that help control how internal memory is maintained, as well as how gradients are propagated backward. In both variants, there is a pathway for uninterrupted gradient flow. The first variant, the LSTM, proposes the use of two states: the cell state and the hidden state. The cell state is essentially the internal memory, whereas the hidden state is the working memory. The transfer of information into and out of these states is controlled by gates. Distinct from the LSTM, the GRU employs a simpler design and contains only a hidden state but utilizes similar techniques for updating its memory. Both the LSTM and the GRU are used for similar tasks, although the GRU is generally preferred for shorter time-series.

1.3.3 Convolutional Neural Network (CNN)

While neural networks were originally developed largely for tasks in image recognition, the MLP is not translation invariant. This means that a feature in an image could be shifted or translated, and dramatically influence the network prediction despite the ground-truth label remaining unchanged. As an answer to this flaw in the MLP, convolutional neural networks (CNNs) were proposed in 1989[21] and later refined in 1998 for the recognition of

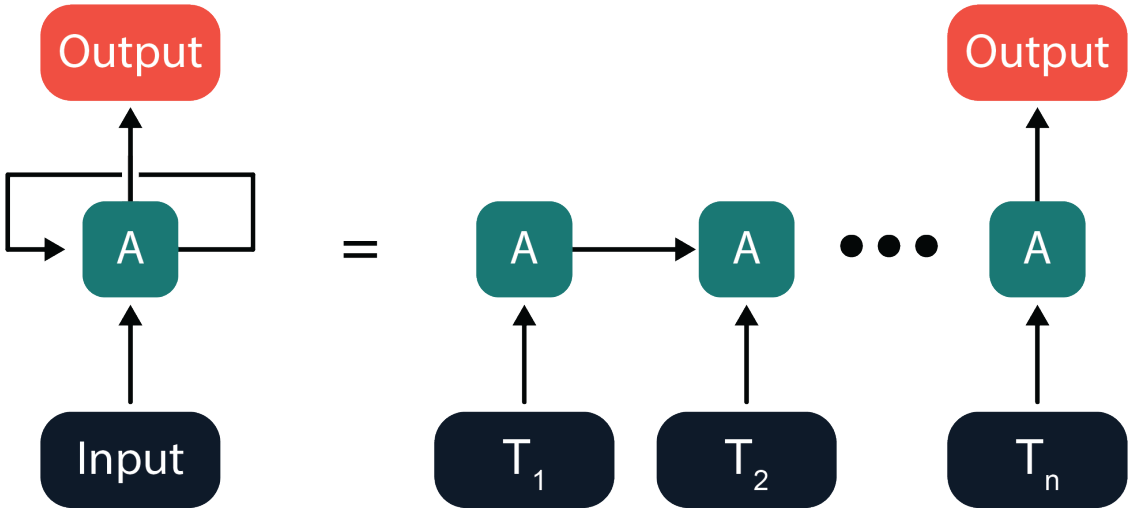

Figure 1.2: RNN architecture where 'A' represents an RNN cell and timesteps of the input are 'unrolled' and shown with T_n on the lower right.

handwritten digits.[22] The convolution drew inspiration from a finding in neuroscience that showed that two types of cells are critical for pattern recognition in cats.[23] The first cell type was responsible for extracting patterns and the second cell type was responsible for distilling information from the first. These two operations were translated into the convolution and max-pooling operations, respectively.

Since their conception, CNNs have found widespread use in image classification,[24] image segmentation,[25] medical image analysis[26] and even natural language processing.[27] CNNs work by making use of the hierarchical structure in the input data and extracting progressively finer patterns in the data as the depth of convolutional layers increases. A key highlight of convolutions is that they are translation invariant, meaning a given pattern can be translated anywhere in the image and a convolutional filter will recognize the pattern equally well. This property makes CNNs uniquely suited for image recognition tasks, where patterns can appear at various locations in an image.[28] A convolutional filter, also known as a kernel, is used to extract patterns from the image. At each position on the image, the filter (generally a 3×3 array) and corresponding patch of the image are multiplied element-wise. The maximum value of this result is then stored as a point in the output matrix. This operation is known as max-pooling.

1.3.4 Variational Autoencoder (VAE)

The variational autoencoder (VAE) is a type of generative neural network architecture that was introduced in 2013.[29] VAEs have expanded the power of neural networks to unsupervised problems and have been very successful in molecular design,[30] multi-omics analysis,[31] and medical image analysis.[32] The VAE consists of an encoder and a decoder that are connected via a latent space that has a significantly lower dimension than the input. They are trained with a two-part loss function (e.g., Equation 1.3). The VAE takes an input matrix x and compresses it down to a lower-dimensional latent space as a vector z using the encoder, which can be a neural network. This latent space is then sampled, and the decoder, which can be

another neural network, attempts to reconstruct the original input. A reconstruction loss is calculated by comparing the original input and the reconstruction ($\mathbb{E}_{q_\phi(z|x)}$ in Equation 1.3). Gradients of this loss function flow from the decoder to the encoder. This encourages the two networks to work together. The encoder learns more salient latent representations, and the decoder learns how to best utilize these representations. The reconstruction loss can take the form of a binary cross-entropy loss (Equation 1.1) if the outputs are binary valued or a mean squared error (Equation 1.2) if the outputs are continuously valued. For example, if the input is an image in which pixel values range from 0 to 255, the mean squared error loss would be the most appropriate.

$$\text{Binary cross-entropy} = -\frac{1}{m}\sum_{i=1}^{m} \hat{y}_i \log\,(y_i) + (1-\hat{y}_i)\log\,(1-y_i) \tag{1.1}$$

$$\text{Mean Squared Error} = -\frac{1}{m}\sum_{i=1}^{m} ||\hat{y}_i - y_i||^2 \tag{1.2}$$

$$\text{VAE loss} = \mathbb{E}_{q_\phi(z|x)}[\log\, p_\theta(x|z)] - D_{\text{KL}}(q_\phi(z|x)||p(z)) \tag{1.3}$$

The second part of the two-part loss is the Kullback-Leibler (KL) divergence (D_{KL} in Equation 1.3). The KL divergence is a way to quantify the difference between two probability distributions. Here we use it to measure the difference between the distribution created by the encoder and the normal Gaussian distribution. This ensures that similar inputs are encoded to similar latent spaces. For example, if a VAE is trained on pictures of dogs, two dogs of the same breed would be encoded in similar spaces.

1.4 Model Interpretability

Model interpretability is nearly as critical to a task as the model itself. Explainable machine learning models are often sought after because many applications require a clear

reasoning of the prediction. For example, tumor cells from a biopsy may be passed through a cytometer and a model may recommend no treatment since it confidently classified the cells as benign. However, we may not weight its recommendation significantly or at all unless we can determine how the model came to this conclusion. There are many different interpretability methods. They can be model-specific and give either local or global interpretability. Therefore, choosing a technique depends on the desired information about the model's predictions.

1.4.1 Shapley Values

One popular method for interpreting machine learning models is the calculation of Shapley values. Shapley values were originally described in the context of cooperative game theory.[33] In this game-theoretic approach, one could determine the individual contributions of players in a theoretical cooperative game upon receiving a reward, such as scoring a point. In the context of machine learning, the players of a game are substituted with a given feature and the reward is substituted with the prediction. To determine the contribution that each feature has on the prediction, different feature values are sampled from the dataset, and the effects on the prediction are measured. This technique has previously been applied to understand single cell classification,[34] prediction of peptide properties,[34] and even particle physics.[35]

Chapter 2

Introduction to Cytometry

Cytometry has existed in different forms since the late nineteenth century with numerous devices and data acquisition types. The term is most commonly associated with the flow cytometer, which was developed only recently in 1953.[36] Cell separation techniques in flow cytometry are non-destructive and high-throughput in nature. Most often cells are characterized by electrical or optical signals. However, image characterization has recently gained popularity and is approaching mainstream adoption.

2.1 Impedance

Impedance cytometry, also known as the Coulter counter, was developed based on the Coulter principle and measures the change in impedance as cells pass through a small orifice.[36] Devices in impedance cytometry work on the simple idea that since the cells are less conductive than the electrolyte solution in which they are found, as they pass through the orifice and displace the electrolyte solution, the channel will become momentarily less conductive. If a current is passed through the channel, its amplitude will decrease in response to the passage of a cell, creating a 'pulse' in the recorded current. The amplitude and duration of the pulse can inform one about the volume of particles and the surface characteristics.[37,38] These measurements allow for the detection and quantification of particles of a wide range of sizes, from blood cells to particles as small as viruses[39] and single molecules. This additionally allows for the sequencing of nucleic acids[40] and proteins[41] since the diameter of the orifice can be altered with great precision.

2.2 Optical

The second method of characterization uses the absorption, fluorescence, or both of laser light to analyze cells. The scattering of laser light allows for the determination of a cell's size and shape. Cell fluorescence is generally achieved through endogenous production of fluorescent labels (e.g., green fluorescent protein (GFP)) or through the binding of a fluorescent antibody tag. Antibody tags require antigens on the surface of cells for binding. A common type of antibody tags are known as 'clusters of differentiation' and are used for phenotyping. Although up to 28 different tags can be used at once, generally only 12-15 are used simultaneously.[42]

2.3 Imaging

While imaging flow cytometry (IFC) might sound similar to optical cytometry, in practical application it is quite different. The devices in IFC utilize photomicroscopes and can be used with or without fluorescent labels. Despite this similarity, IFC is rich in data and can contain information largely inaccessible with other techniques.[43–46] Imaging flow cytometry generates multiple images of cells with high resolution. Common features extracted are area, shape, morphology, texture, granularity and intensity. The power of the generated data is largely dictated by the analysis techniques employed, and there is often a content gap between the data provided and the features extracted.[46] Despite this, much progress has been made in the application of IFC and it has shown potential in the diagnosis of acute leukemia.[47]

2.4 Deformability

Mechanical properties have long been probed by atomic force microscopy,[48,49] optical tweezers,[50] or micropipette aspiration.[51] These techniques all offer different perspectives on the mechanical properties of a cell via different forces and readouts for the cell's response. Deformability cytometry (DC), unlike the methods mentioned above, was only recently

developed in 2015.[52] Since then, this technique has evolved into different forms that are a variant of either impedance or imaging flow cytometry.

Both characterization methods using impedance or imaging flow cytometry have been shown to probe deformation induced by hydrodynamic forces or narrow channel widths. To probe a cell's deformability, impedance-based methods calculate the change in current induced by a cell's passage.[53–56] Impedance measurements also allow for the determination of other biophysical parameters, such as membrane capacitance, that are unavailable to image-based techniques. Imaging-based platforms rely on a high-speed camera to capture one or more images of the cell's deformation. Similar to IFC, imaging-based deformability cytometry uses high-speed cameras and captures a large volume of information. This form of deformability cytometry has been demonstrated with single images[52] as well as a time-series of images (Figure 2.1).[57,58]

Two key deformability cytometry methods are dynamic real-time deformability cytometry (dRT-DC)[57] and extensional deformability cytometry (xDC).[59] The dRT-DC method deforms cells with a constant shear rate and the physical constants of the cells can be extracted by observing their steady-state deformation. Conversely, xDC rapidly deforms cells using a cross-channel design and ultra-fast flow rates that enable high-throughput characterization. Overall deformability cytometry has been shown to be sensitive to viscosity,[60] as well as actin and microtubule networks.[61,62] These capabilities have allowed deformability cytometry to find many applications, including the study of anticancer drug assays,[63] as well as the classification of pluripotent and differentiated cells.[64]

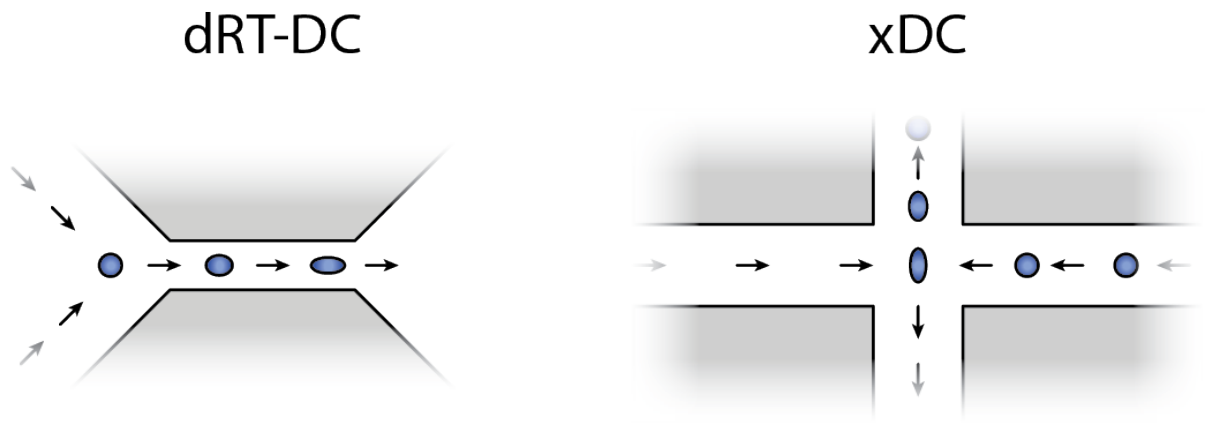

Figure 2.1: Channel architectures for two deformability cytometry devices are shown. Dynamic real-time deformability cytometry (dRT-DC) is shown on the left. Extensional deformability cytometry (xDC) shown on the right.

Chapter 3

Deep Learning Assisted Mechanotyping of Individual Cells Through Repeated Deformations and Relaxations in Undulating Channels

3.1 Introduction

Mechanical properties of cells, such as the ability to deform when an external force is applied, are directly linked to the structure of a cell's cytoskeleton. Changes in the cytoskeleton have been correlated with cell differentiation,[65] malignant transformation,[66] biofilm formation[67] and COVID-19 pathology.[68] Therefore, probing mechanical properties offers a label-free method to learn about a cell's state, such as its homeostasis or pathological conditions. This information could be useful in clinical settings such as chimeric antigen receptor therapy where the potency of the treatment might be linked to a cell's mechanical properties.

A multitude of techniques have been reported to probe the mechanical properties of cells. Conventional methods include atomic force microscopy,[48,49] optical tweezers[50] and micropipette aspiration.[51] While these approaches are capable of measuring the mechanical properties of individual cells with high resolution, their throughput of roughly 1-10 cell(s) per minute is significantly slower than the ∼10,000 cells per second achieved by most flow cytometers, which is required for characterization of cell populations that can easily reach millions of cells.

In recent years, several microfluidic methods have been developed with significantly improved throughput. In one class of methods, cells are squeezed through constrictions smaller than the cell's diameter.[53–56] The passage time of these strongly deformed cells has been shown to depend on the cells' mechanical properties, with a faster passage corresponding

to a more deformable cell. This approach was applied to different cell lines, including cancer cells, stem cells, and red blood cells.[55,69–77] The transit time through the channel is measured by recording electrical impedance or optical signals that correlate with deformability, and in certain cases measure rheological parameters such as Young's modulus.[78] However, the surface friction between the walls and the cell can make it challenging to extract deformability without including other phenomena.

In a different class of microfluidic approaches, the channels are wider than the cells to be analyzed and the passing cells are subjected to hydrodynamic forces. Again, in these methods deformations can be probed electrically or optically. Two notable approaches are extensional deformability cytometry (xDC)[59] and real-time deformability cytometry (RT-DC).[52] In xDC, a cross-channel design is used to deform cells at ultra-fast flow rates reaching $\sim$1000 µL min^{-1}. xDC has been used to classify malignant pleural effusions, discriminate stem cells, and identify transitions in the cell cycle. The authors showed that the deformation kinetics are important in the classification of induced pluripotent stem cells (iPSCs).[58,64] Using both rheological and morphological features, Masaeli et al. demonstrated the ability to classify between iPSCs and differentiated iPSCs using support vector machines (SVM).[64] The ultra fast flow rates of xDC come at the cost of being unable to probe more sensitive cytoskeletal structures like actin networks. RT-DC, which is based on a straight narrow microfluidic channel, operates at significantly lower flow rates than xDC ($\sim$1 µL min^{-1}), and induces constant shear stresses allowing cells to reach steady state deformation. RT-DC quantifies deformability using steady-state images of deformed cells captured at the end of the microfluidic constriction and is linked to a physical model to calculate Young's modulus.[79] RT-DC has also been recently extended to probe the cell deformation kinetics as they approach steady state.[57] RT-DC has also been shown to classify reticulocytes from mature red blood cells with an unsupervised approach.[80]

While these approaches have shown great promise and success for a variety of problems, the classifications rely heavily on the size and morphology of cellular populations. While

these features are useful in practical applications, existing techniques such as imaging flow cytometry are also able to discriminate based on size and morphology. The discriminative power of DC on cell populations which differ solely in mechanical properties alone has therefore not been fully explored. Additionally, the use of deep learning models, particularly convolutional and sequence-based models, may provide higher classification potential than the traditional machine learning models that many works employ.[81,82] In this manuscript, we propose to improve classification accuracy by maximizing mechanotyping information by subjecting cells to repeated deformations and relaxations with hydrodynamic forces. In addition, we will show the application and potential of deep learning methods. Given that the videos are recorded with the time resolution of 11,000 frames per second, we can probe the deformation kinetics with high precision. These observations provide quantitative insights into the deformation/relaxation processes and provide a comprehensive mechanical fingerprint of each cell. Our channel contains a cavity flanked by two narrower regions,[83] with widths that at any position are wider than the cells to be analyzed. Our technique operates at a throughput comparable to the throughput of RT-DC that enables observations of hundreds of cells per minute and yet probe the cells sufficiently slowly to observe cytoskeletal changes.[60] Inducing repeated deformations and relaxations by hydrodynamic forces is the natural progression towards improving cellular characterization and classification based solely on mechanical features.[84,85]

A model dataset was constructed with human leukemia 60 (HL60) cells before and after treatment with either Cytochalasin D (CytoD) or Nocodazole (Noco) to test the classification potential of our method. Both chemicals were found to perturb HL60 cell deformability such that CytoD-treated HL60 cells (HL60d) were more and Noco-treated HL60 cells (HL60n) were less deformable than untreated HL60. Most importantly, cells belonging to the three subpopulations, HL60, HL60d, and HL60n, have the same average size, thus enabling us to test our classification strategy based on mechanical properties alone. Using these sub-populations subjected our method to a very challenging test, since classification is often aided by cells'

size.[58,86]

The analysis of the recordings was supported by machine learning approaches with gradually increased levels of expressive power. We show that the dynamic mechanotyping features our method provides enable a significant increase in the classification accuracy of the sub-populations when compared to using any single feature alone. Additionally a Random Forest (RF) model was trained using hand-picked features. This enabled the use of Shapley values application, a technique known from economic game theory, to investigate which deformability parameters contributed the most to classification accuracy.[33,87] A significant improvement in classification accuracy was observed when the time-series of deformations was used as an input into deep learning models such as recurrent neural networks (RNNs). Furthermore, we found the combination of convolutional and recurrent layers utilizing the sequence of masks showed an increase in classification accuracy to 90%, up from the 75% accuracy we observed with the RF model. This increase in accuracy translates to an order of magnitude increase in the potential ability to enrich a sample for a rare population of cells (Supplementary Material Note 1). We discuss the trade-off between the interpretability achieved with more traditional machine learning approaches and the high accuracy often associated with deep learning.

3.2 Methods

3.2.1 Cell Culture

HL60 (ATCC CCL-240) cells were cultured in suspension with Iscove's Modified Dulbecco's Medium (IMDM) supplemented with 10%(v/v) FBS (Fisher brand) and 1% (v/v) penicillin-streptomycin and maintained in an incubator at 37°C with 5% CO_2. Cells were passaged through dilution every 2-3 days to maintain a density between 10^5 and 10^6 cells mL^{-1}. Before the experiments, cells were centrifuged at 112 relative centrifugal force (RCF) for 5 minutes and resuspended to concentrations of 2-3x10^6 cells mL^{-1} in PBS with 1% (w/v) methylcellulose (Spectrum 4,000 CP).

To create subpopulations of HL60 cells with perturbed actin and microtubule networks, cells were incubated for 10 minutes in 1 µM CytoD (Sigma) or for 1 hour in 10 µM Noco which had been diluted 10x from the stock solution with dimethylsulfoxide (DMSO).[88] Cells were spun at 112 RCF for 5 minutes and resuspended in 1% (w/v) methylcellulose solution.

3.2.2 Channel Preparation and Imaging Setup

Microfluidic channels were prepared with a master mold using standard photolithography with negative SU-8 photoresist (Kayaku Advanced Materials Inc.). The channel used in the experiments was 150 µm long, with three equal length sub-regions (50 µm) with widths of 25 µm, 50 µm, and 25 µm. The height of the device was 20 µm along the whole length. The full geometry of the chip is shown in Figure 3.1. 184-Sylgard polydimethyl siloxane (PDMS) was pipetted over the SU-8 master and baked for ∼4 hours at 75°C. The channel inlets and outlets were created with a 1.5 mm biopsy punch. PDMS channels were cleaned and dried with isopropyl alcohol, methanol and water before being bonded to a glass cover slip using a corona discharge wand (ETP). Methanol was used last before drying because of its low boiling point to ensure that no residual alcohol remained in the device. Bonded PDMS/glass samples were heated for an additional hour at 90 °C to promote further adhesion.

The cells were suspended in a methylcellulose solution to prevent the suspension to settle at the bottom of the container. The solution was then pumped through a microfluidic channel using a Genie-plus syringe pump (Kent Scientific) at a rate of 1 µL min^{-1}. The cells were laterally focused in the channel using a sheath flow geometry with a flow rate of 2 µL min^{-1}. The sheath and core flows were allowed to equilibrate for 10 minutes before data was taken. Cells were illuminated using a high-powered Red Amber (613 nm) 36 watt LED array (PT-121-RAX Luminus, Inc.) and imaged at 10x magnification in brightfield on an inverted microscope. A Chronos 1.4 high-speed camera (Krontech) imaged passing cells at a frame rate of ∼11,000 frames per second (fps) with 1 µs exposure time and a resolution of 880x140 to encapsulate the full channel length. The size of each pixel is 0.26 µm/pixel. The maximum

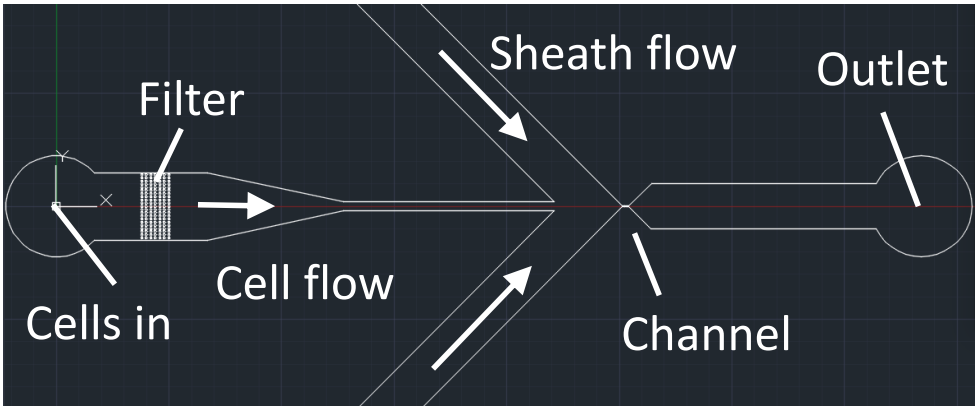

Figure 3.1: Microfluidic chip layout. Cells enter through the core and are focused by two sheath flows. Cells are imaged in the channel region and then collected from outlet.

blurring induced by cell movement is $\sim 0.1\,\mu m$. 8 second videos were recorded and saved which require ∼15 minutes to offload from camera memory. To ensure the generalizability of the model, in total, ∼ 3,500 cell trajectories were recorded and analyzed. The data presented were collected over 10 technical replicates of HL60 (4 biological replicates), 8 technical replicates of HL60d (4 biological replicates), and 6 technical replicas of HL60n (2 biological replicates).

3.2.3 Calculation of Features

Detected events with impossible trajectories (i.e., no event will begin in the middle of the channel) were discarded. To ensure the fits are accurate, a convex hull was fit to the cell mask and events were filtered out where a single frame had a ratio >1.1 between the original and convex hull fit. Detected particles with radii equal to three standard deviations from the mean were not included in the analysis, as they often contained cell/channel debris or clumps of multiple cells. Ellipses were fit to the detected masks and the aspect ratio of the ellipse was used to describe the cell deformation, AR. A full description of deformation parameters can be found in Table 3.1.

3.2.4 Comsol Simulation

A finite element simulation was conducted using Comsol Multiphysics 5.3 to simulate the velocities and stresses experienced in the undulating channel. A simplified 3-D model of the channel was modeled in the laminar flow module using the Navier-Stokes equation with creep flow in steady state. An extra fine meshing was chosen for the area near the narrow section of the channel and normal meshing was chosen for the reservoir.

3.2.5 Machine Learning Model Training

Of all the cell trajectories recorded, 3,552 were suitable for classification. They were distributed as 1114 HL60 trajectories, 1122 HL60d trajectories, and 1316 HL60n trajectories.

The RF model and SVM were created and trained using python 3.6 and scikit-learn.

Table 3.1: Parameter table

Feature	Description	Hl60n	HL60d
Radius	Radius of cell in cavity where deformation is minimal. Average radius of circle of equivalent perimeter and circle of equivalent area.	59%	56%
rD1	Maximum relative aspect ratio of fitted ellipse measured in first narrow constriction.	57%	72%
rD2	Maximum aspect ratio of fitted ellipse measured in second narrow constriction	55%	71%
R2 Slope	Slope of linear fit between position of maximum deformation in first narrow constriction and position of least deformation in cavity. Represents cell relaxation.	58%	68%
R3 Slope	Slope of linear fit between position of least deformation in cavity and position of maximum deformation in second narrow constriction. Represents cell compression.	52%	70%
R1-R3 AR	Difference between AR1 and AR2	58%	59%
R1 Per	Average perimeter of mask in first narrow constriction.	53%	55%
R3 Per	Average perimeter of mask in second narrow constriction	51%	54%
R1 Area	Average area of mask in first narrow constriction	55%	57%
R3 Area	Average area of mask in second narrow constriction	54%	56%

Table 3.2: Expanded Parameter Table

Feature	Description (Calculated at R1 max deformation, R2 minimum deformation, and R3 max deformation.)
I_{xx}	Central moment of inertia along channel in direction of fluid flow.
I_{yy}	Central moment of inertia perpendicular to direction of fluid flow.
I_{xx}/I_{yy}	Ratio of diagonal elements of central inertia tensor.
H[0],H[1],... H[6]	Hu's set of image moments

The data were standardized, shuffled, and split according to the following ratio: 70:15:15 for train, validation, and test, respectively. Unless specified otherwise, all convolutional and dense activation functions are ReLU. Unless otherwise specified, RMSProp was used with a base learning rate of 1e-3.

The GRU and CNN-GRU models were created and trained in Python 3.6 using the Keras and Tensorflow 2.3.1 libraries. The hyperparameters were optimized over 175 epochs using the validation set. Model performance was assessed using a hold-out test set, in addition to 5-fold cross-validation.

3.2.6 CNN Prefilter Training Architecture

The CNN prefilter was trained with a number of random augmentations. The following augmentations from Tensorflow image were used: per image standardization, random brightness, random flip up-down, random flip left-right, random contrast. Model weights were chosen based on validation set performance.

1. Conv2D - Filters: 16 - Kernel: 3×3
2. MaxPooling2D
3. Conv2D - Filters: 16 - Kernel: 3×3
4. MaxPooling2D
5. Conv2D - Filters: 16 - Kernel: 3×3
6. MaxPooling2D
7. Flatten
8. Dense - Dimension 1 - Activation: sigmoid
9. Loss Function: Binary Cross-Entropy

3.2.7 GRU Network Architecture

1. Masking
2. GRU - Dimension 50
3. GRU - Dimension 50
4. Dense - Dimension 24
5. Dense - Dimension 1 - Activation: sigmoid
6. Loss Function: Binary cross-entropy

3.2.8 CNN-GRU Training and Architecture

Data were filtered, aligned, and partitioned using the same methods described above for the GRU. The masks were cropped from the frame and padded to a size of 96x96. An ellipse was fitted to the cell, and the two channels were concatenated.

1. Conv2D - Filters: 8 - Kernel: 3×3
2. MaxPooling2D
3. Conv2D - Filters: 16 - Kernel: 3×3
4. MaxPooling2D
5. Conv2D - Filters: 16 - Kernel: 3×3
6. MaxPooling2D
7. Flatten
8. GRU - Hidden Dimension: 50
9. GRU - Hidden Dimension: 50

10. Dense - Dimension 1 - Activation: sigmoid

11. Loss Function: Binary cross-entropy

5-Fold Cross-Validation Results HL60 v HL60d

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean
0.917	0.922	0.908	0.906	0.917	0.914 ± 0.006

5-Fold Cross-Validation Results HL60 v HL60n

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean
0.833	0.821	0.840	0.848	0.842	0.837 ± 0.009

3.2.9 Data Preprocessing for Sequential Models

Sequences of aspect ratio, perimeter, deformability, and area were selected as inputs for the GRU model. Data previously filtered were first aligned so that all sequences started and ended at a true x-position of -30 µm and 170 µm respectively. Data were then padded to length 50 and were shuffled and split according to the following ratio: 70:15:15 for train, validation, and test, respectively.

For sequential models using image data, CNN-GRU, masks were first cropped to 96x96. An ellipse fitted to the mask shape was created using scikit-image and added to the second image channel. Each sequence of these two-channel images were padded to length 50. The data were shuffled and split according to the following ratio: 70:15:15 for train, validation, and test, respectively.

3.2.10 Code Availability

The python scripts implemented for data processing and training machine learning models are available at: https://github.com/siwylab/time-series-dc.

3.3 Results and Discussion

3.3.1 Channel Design and Data Acquisition

We designed a microfluidic channel that subjects individual cells to repeated compressions and relaxations. The multiple deformations of the cells are induced by the channel shape, specifically the presence of a cavity and two narrow zones. Literature suggests that the varying channel width will create non-homogeneous velocity profiles leading to temporal changes in the cells' shapes.[89] To test this channel design for mechanotyping, we first selected HL60 cells for study, which have been previously shown to be resistant to changes in radius and morphology in response to cytoskeletal-altering drugs.[90] We then pumped a suspension of HL60 cells (ATCC-240) in methylcellulose (1% w/v) solution through a channel of consecutive constrictions of length equal to 50 µm (total length of 150 µm) and widths of 25 µm, 50 µm, and 25 µm at a flow rate of 1 µL min^{-1}. The viscous methylcellulose solution induces larger deformations than the lower viscosity buffer solution and prevents cell sedimentation. Sheath flow was used to focus and ensure that all cells passed through the channel along its center axis and experienced the same forces. The microfluidic chip was placed on a 10x magnification inverted microscope. To enable probing dynamics of cells' shape at this high flow rate, we customized the microscope to allow high-speed imaging by replacing the stock light source with a high-powered LED array and accompanying 3D printed mount (Figure 3.2a). A high-speed camera was modified to fit the microscope, and videos of translocating cells were recorded at 11,000 frames per second, resulting in $\sim$ 30 data points on the 150 µm channel and revealing the shape dynamics exhibited by a cell as it moves along the channel axis.

Upon successful capture of video data, a python pipeline was used to process the data. The first step in the pipeline encompasses a lightweight CNN that filters out empty frames without cells, and reduces the amount of data for the next step by $\sim$80%. This 4-layer CNN was created with each Conv2D layer containing a ReLU activation function, 3x3 kernel, and 16 filters. The final layer was a single dense node with the sigmoid activation function.

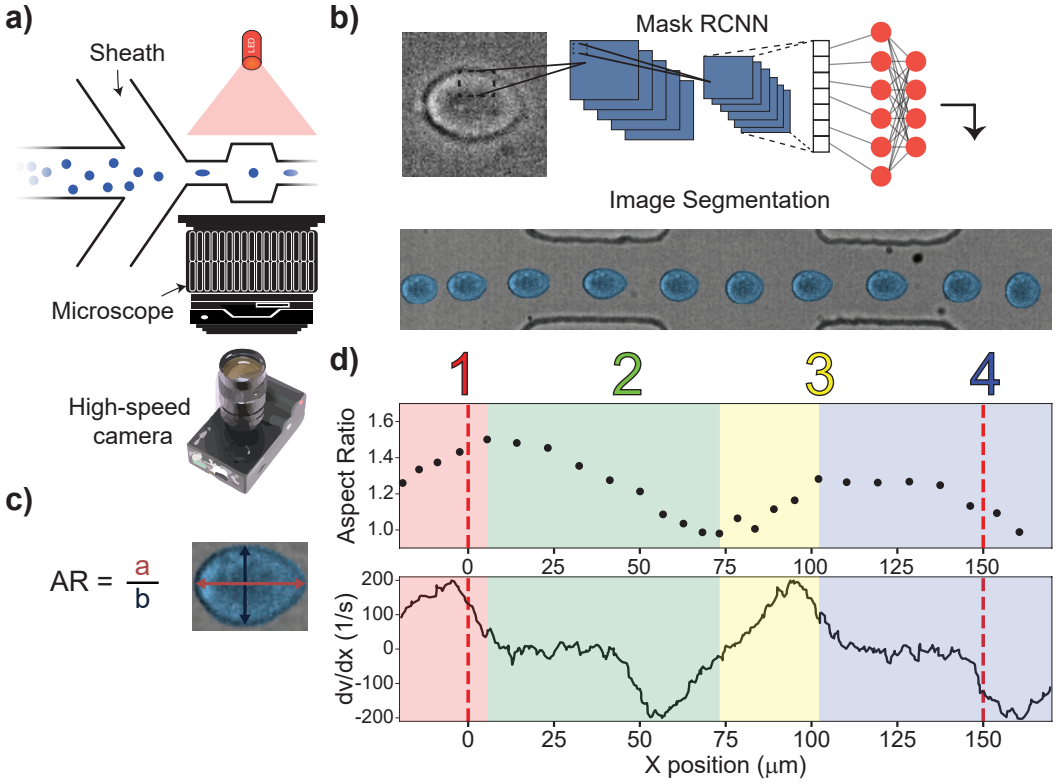

Figure 3.2: Principles of repeated mechanotyping. a) Channel design utilizing sheath flow, a high-powered LED and a microscope. The microfluidic channel that enables characterization and classification contains a cavity placed between two narrow zones. b) Data is captured by a high-speed camera, creating videos at 11k fps. Cell borders are detected and fit using Mask-RCNN. c) The cell deformation, AR, was quantified as the ratio of two axes of an ellipse that approximates the cell's shape. (d) (Top) The aspect ratio versus position relative to channel entrance of a single cell as it passes through the channel. (Bottom) COMSOL simulation showing the derivative of velocity vs.channel position, which is proportional to the shear stress. Region 1 (R1) and Region 3 (R3), denoted by red and yellow regions, are where the cells undergo deformation. Region 2 (R2) and Region 4 (R2), denoted by green and blue regions, are where the cells undergo relaxation.

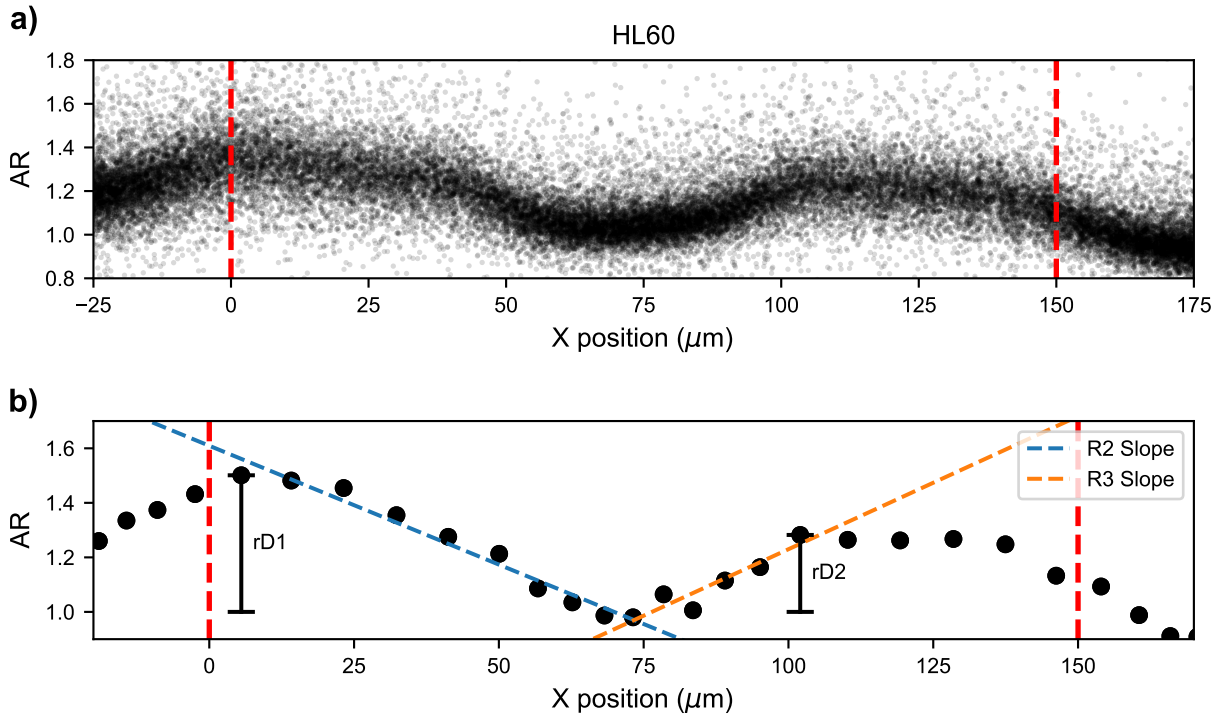

Figure 3.3: Single cell deformation traces. Deformation dynamics are shown for single cells translocating through the channel. The aspect ratio is determined by the best fit ellipse to the cell mask. Deformation is calculated by the difference between the aspect ratio at a given point and the minimum aspect ratio in the cavity. The x-position along the channel axis is determined by the centroid of the mask. Cells experience a smaller maximum deformation in the second narrow region, as compared to the first narrow zone. Channel inlets and outlets are marked by red dotted line. Deformation and relaxation occur twice within the channel. a) Full population of single cell traces of aspect ratio versus position for untreated HL60 cells. b) Single cell example of parameters that are determined: relative maximum deformations (rD1 and rD2) in the two narrow zones as well as relaxation and deformation slopes (R2 slope and R3 slope).

RMSprop was used to optimize the binary cross-entropy loss with a base learning rate of 0.0001. Roughly 9000 total images were used as inputs with 4000 background images and 5000 cell images. The input images were resized with padding down to 256x256. Pixel values were scaled down from 16 bits to 8 bits using the formula below.

$$\text{new frame} = 255 \times (\text{frame - min(frame)})/\text{range(frame)}$$

Frames labeled as containing cells were then passed to a Mask-RCNN network that was used to segment cells from images and fit masks. Mask-RCNN can learn to accurately segment cells with differing focusing conditions and is able to segment multiple cells in a single frame. The Matterport implementation[91] was used with Tensorflow 2.2 due to its code availability and its depth of support in the forums. The network was trained using a NVIDIA 1070TI on a hand-labeled dataset of ∼300 images of HL60 cells across multiple independent experiments and cell populations using VGG Image annotator 1.0. The network was initialized with weights pretrained on the COCO dataset. The COCO dataset is a large-scale object detection and segmentation dataset with images of common items such as people, bicycles, and airplanes. The architecture was modified in order to increase the output resolution of the predicted masks since all downstream tasks depend on them. Our fine-tuned Mask-RCNN network is able to accurately fit masks under various focusing and lighting conditions with mean mean average precision (mAP) and mean intersection over union (mIOU) > 0.9, reducing bias in the overall mask fitting between experiments. The training schedule consisted of training head layers for 20 epochs at a learning rate (LR) of 10^{-3}, 50 epochs training 4+ layers at LR/10, and 50 epochs training all layers at LR/10. Training curves and additional details can be found in Figure 3.4.

3.3.2 Custom Tracking Algorithm

After images are processed by Mask-RCNN, the custom tracking algorithm begins by iterating over the list of positive frames output by the CNN prefilter. The frames are then loaded from the raw video, and the frames are normalized. The trained Mask-RCNN is run

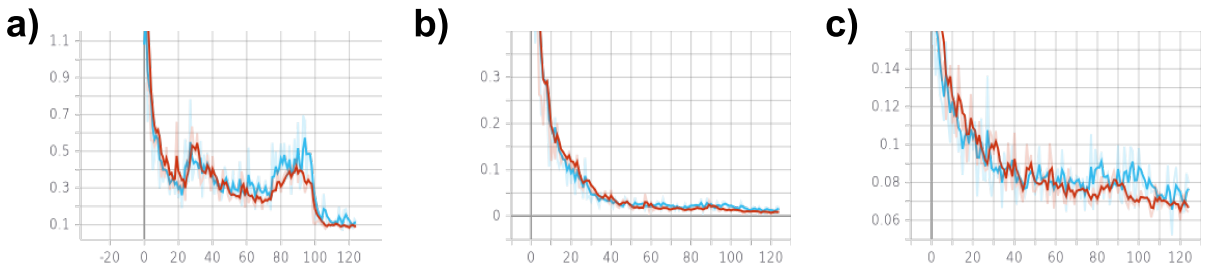

Figure 3.4: Training curves for Mask-RCNN. a) Total loss per epoch. b) Bounding box loss per epoch. c) Mask loss per epoch.

in inference mode and the normalized frames are passed through the network. The resulting list of binary masks is then iterated over. If the area is erroneously small or large (less than 600 pixels or greater than 2500 pixels), the cell is skipped. Since Mask-RCNN has a tendency to hallucinate cells that do not exist, the algorithm also checks to see if the cell is outside the channel. Assuming the cell passes these tests, the mask is then added to a list along with the coordinates of the center of the cell. The algorithm then checks to see if there are any cells with overlapping x-axis positions. If there are overlapping positions, the algorithm selects the cell closest to the center of the channel. The newly identified masks are then sorted from right to left based on their x-axis position. The algorithm then determines whether we have a new 'event' based on the number of incoming frames. If there are more frames than currently tracked events, there is a new event, and the model expands the dictionary of currently tracked events. The dictionary of currently tracked events is then screened for inactive events (current time in video and last observed frame are greater than two frames). Next, a distance matrix is computed between all the positions on the x-axis of the incoming cells and the last observed position of the current events. The incoming events are then assigned to the entries of the current events according to the smallest distance on a given vector of the distance matrix. The current events are then iterated over and the matched frames are checked to ensure the matched cell is within the minimum and maximum distance to the assigned cell. This ensures that events do not have stationary points nor do they have jumps to later points in the channel.

Figure 3.2b shows subsequent snapshots of one cell as it passes through the channel. Mask-RCNN enables us to track multiple cells in the same frame, allowing a high density of cells ($5x10^6$ cells mL^{-1}) to be used in the experiments. We found that the undulating channel design results in complex dynamics of the cells' shapes, and led to regions of differing deformation. Specifically, the cells deformed strongly at the entrance of the channel and in the first narrow constriction; the cells then relaxed to a spherical shape in the cavity, and started to deform again when approaching the second narrow constriction. As seen in Figure

3.2d, we define the regions in which cells undergo deformation as region 1 (R1) and region 3 (R3). The regions where the cell undergoes relaxation are denoted as region 2 (R2) and region 4 (R4).

We find that the shape of the deformed cells is best approximated as an ellipse where the deformation (D) is defined as the aspect ratio (AR) of the two axes of the ellipse: the axes parallel and perpendicular to the channel axis (Figure 3.2c). We define relative deformability (rD) as the difference between D at a given point in the trajectory and D in the cavity, where the cells undergo no deformation. An rD of zero corresponds to lack of deformation relative to unperturbed cell shape, whereas rD>0 corresponds to an extension along the channel axis. This is important since cells may be elliptical before forces are applied. Figure 3.2d shows how the AR evolves as the cell passes through the channel. In order to gain insight into the deformation trace, we performed a computational fluid dynamics simulation with COMSOL multiphysics. In a cell-free undulating channel, we simulated fluid flow with the Navier-Stokes equations and creep flow at experimental flow rates. Shear stress in different parts of the channel can be analyzed by taking the derivative of velocity in the center of the channel with respect to the axial position (Figure 3.2d). We find that cells experience a large velocity gradient at the entrance of the channel, leading to large stresses and deformations.

In the example cell trace shown in Figure 3.2d, we observe a peak deformation of AR = 1.49 in the first region (R1), marked in red. The velocity gradient then reaches a steady state value within the first narrow constriction, (10 µm - 50 µm) and the cell deformation decreases. Before the cell's shape can reach equilibrium, it enters the cavity where the velocity gradient begins to decrease and then again rapidly increases. The cell returns to a spherical shape, AR = 1, where dv/dx = 0, which occurs at ∼75 µm. The cell then begins to deform again due to the velocity gradient at the entrance of the second narrow constriction, reaching a second peak deformation of AR = 1.30 in region three (R3). The maximum deformation observed in the second constriction is lower than the maximum deformation measured in the first constriction. We believe this is because the cell is subjected to positive shear stresses over a relatively

short time and distance when transitioning from the cavity to the second narrow region, as compared to the transition from the bulk channel to first narrow constriction entrance. The distance from the middle cavity, where the shear stress is zero, to the position of peak deformation (and maximum positive shear stress) in the second narrow region is only 25 µm, whereas at the initial inlet the cell begins deforming from positive shear stresses ∼50 µm away from the entrance. Figure 3.3a shows values of AR for ∼1200 HL60 cells examined in the same conditions. The same trend for all cells has been observed: the cells reached the maximum deformations in the first narrow zone, relaxed to a sphere in the cavity, and underwent another deformation in the second constriction.

Taking advantage of the time-series of cells' positions and shapes, we also quantified the dynamics of deformation and relaxation. To this end, we used a linear model to fit the trace from the peak deformation in the first narrow zone to the position where the cell relaxes to a spherical shape in the cavity (R2), resulting in a slope of -8.71×10^{-3} µm^{-1}. This slope describes relaxation dynamics, and we refer to it as the R2 slope. A similar analysis can be performed for the cell entering the second narrow zone by fitting a line between the spherical shape in the cavity and the maximum deformation in the second narrow zone, obtaining a value of 9.75×10^{-3} µm^{-1}. This slope, called the R3 slope, describes the deformation dynamics. Figure 3.3b shows example fits of rD, R2, and R3 slopes to a trace of AR evolution for a single cell.

3.3.3 Application of the Undulating Channel to Probe Perturbation of Actin and Microtubule Networks

In order to evaluate the sensitivity of our method to detect cytoskeletal perturbations, we continued the experiments with HL60 cells and treated them with CytoD and Noco. These two chemicals have previously been used to create model populations of HL60 cells that allowed researchers to evaluate the performance of different mechanotyping techniques.[52,59] CytoD disrupts actin polymerization,[92] and has been shown to increase deformability of HL60.[52,78,88]

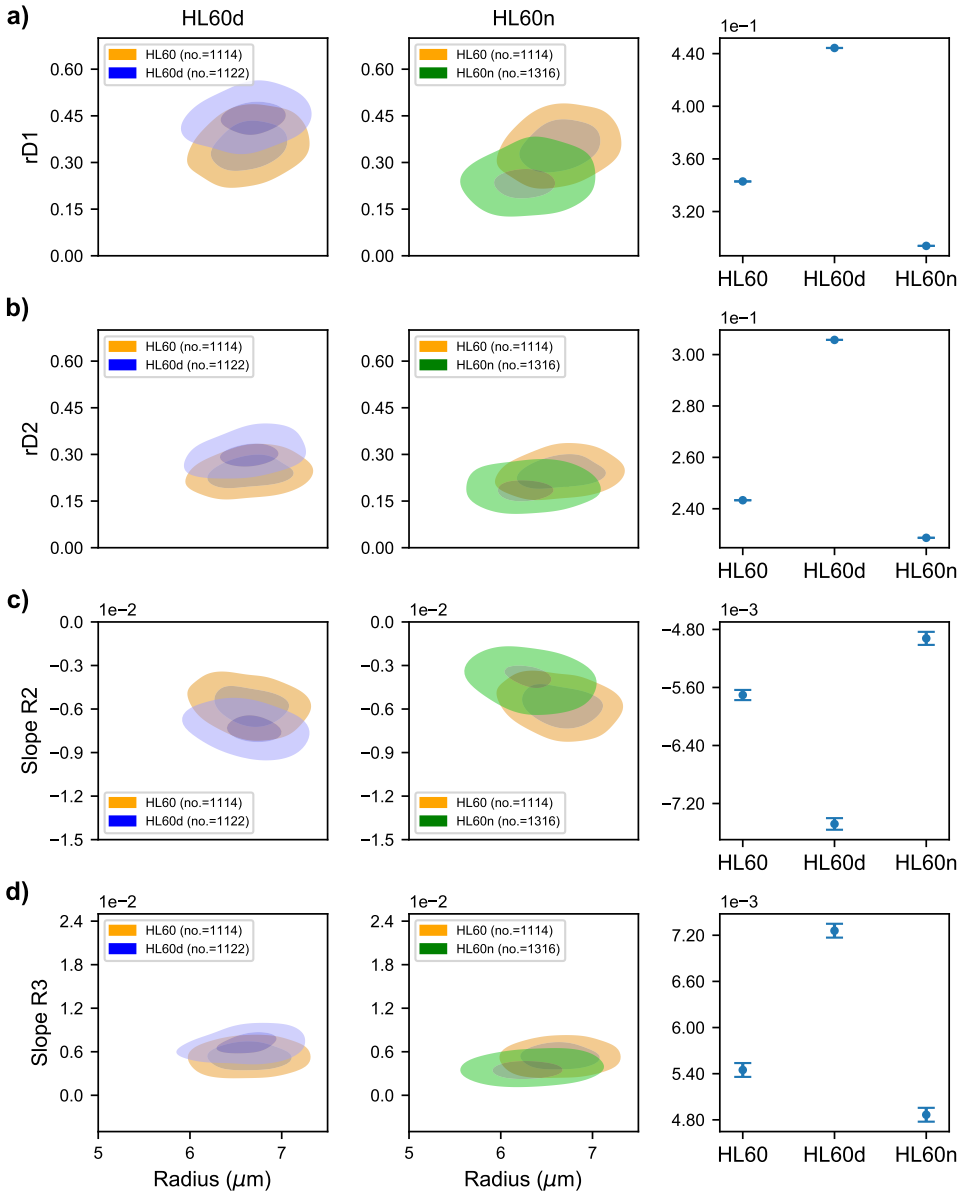

Figure 3.5: Comparison of measured deformability features between untreated and treated HL60 cells. a) Contour plot of maximum rD in first narrow zone (R1) for untreated, CytoD treated, and HL60n cells. The outer contour represent 50% density and center contour represents 90% density. Mean of each population is reported where the reported error is standard error of the mean. b) Contour plots of maximum rD in second narrow region (R3). c) Linear fit slope from maximum deformation in first narrow region (R1) to relaxation to minimum deformation in cavity (R2). d) Linear fit slope from relaxed state in cavity (R2) until maximum deformation in second narrow region (R3).

Noco targets microtubules causing rapid filament decomposition. For HL60 cells, Noco has previously been found to decrease the ability to deform.[88] Perturbing different components of the cytoskeleton allowed us to test whether the deformation and relaxation processes from our channels and the chosen flow rate are capable of measuring changes in whole cell deformation after actin and microtubule disturbance.[60] The three populations of HL60 cells, untreated, CytoD treated, and Noco treated, were suspended in methylcellulose solution, and separately passed through our microfluidic channel. After post-processing, passages of more than 1000 cells of each sub-population were captured. The repeated experiments and the amount of cells measured show consistent trends and enable statistical analysis of the various measured parameters across the populations.

Figure 3.5a,b shows the rD in the two narrow zones and the two slopes for the three populations (Figure 3.5c,d) of HL60 cells. The magnitudes of rD are consistently the highest for the population treated with CytoD, which confirms that the method is sensitive to actin network perturbations. The Noco-treated populations exhibit the lowest mean rD, although with a larger variability than the other two populations. These findings are in agreement with previous reports using the same cell line which also observed increased (decreased) deformability of the CytoD (Noco) treated HL60 populations.[88]

Interestingly, the R2 and R3 slopes that respectively represent relaxation and deformation dynamics, are also the greatest for the CytoD-treated cells, suggesting that these cells are most responsive to external forces. In contrast, the slopes for the less deformable, Noco treated populations are characterized by significantly smaller magnitudes of both slopes than CytoD treated and untreated HL60 cells. Though the HL60n had a slower response to the shear stress, they relaxed to a spherical shape in the cavity, and deformed again in the second narrow region. One important area to control was the radii of cells entering the channel since

It is important to note that the mean radius of the three populations is nearly identical (Figure 3.6), confirming that the cells experienced similar forces. We measure the radii of cells by measuring the area and perimeter of cell masks where the shear stress is zero, which

occurs at 75 µm. The radii is the average of fitting a circle with equivalent area and perimeter respectively. We find a small discrepancy between the mean radii of HL60 cells and HL60 cells treated with CytoD, while HL60 cells treated with Noco have a slightly larger variation in size. Cells with radii larger than 3σ from the mean were removed from analysis, as they were often small cell or PDMS debris, or clumps of two or more cells.

Although the mean values of deformation are significantly different (Figure 3.5, third column), there is significant overlap between the control HL60 cells and the treated cell distributions (Figure 3.5, contour plots). This overlap makes it difficult to accurately classify a given cell. We find that no single feature alone can provide an accuracy significantly higher than 70% for single cell classification (Table 3.1). For example, rD1 alone is able to classify between HL60 and HL60d populations at 72% accuracy using a logistic regression model, while the same parameter classifies HL60n with only 57% accuracy. The radius alone can be used to classify up to 56% and 59% for HL60d and HL60n, respectively, as expected from the nearly identical size of the populations. In the following sections we will show that in order to achieve discrimination between the two populations on a single cells basis, multiple features describing the cell deformation (Figure 3.5) must be considered simultaneously.

3.3.4 Feature Extraction and Machine Learning Model Comparison

Our next goal was to utilize the complete mechanotyping fingerprint our method provides to maximize classification accuracy. As mentioned above, the three populations chosen were nearly identical in size thus the differences in mechanical properties provide the only basis for classification. We first investigated the ability of traditional machine learning models to distinguish the two pairs of sub-populations, HL60 vs. HL60d, and HL60 vs. HL60n. To provide a baseline for classification potential, features were first manually extracted from the individual raw time-series data as seen in Figure 3.3. These derived features were used to train a random forest classification model[93–95] (RF), which has been previously employed for classifying cytometry data.[96] The random forest models were implemented using scikit-

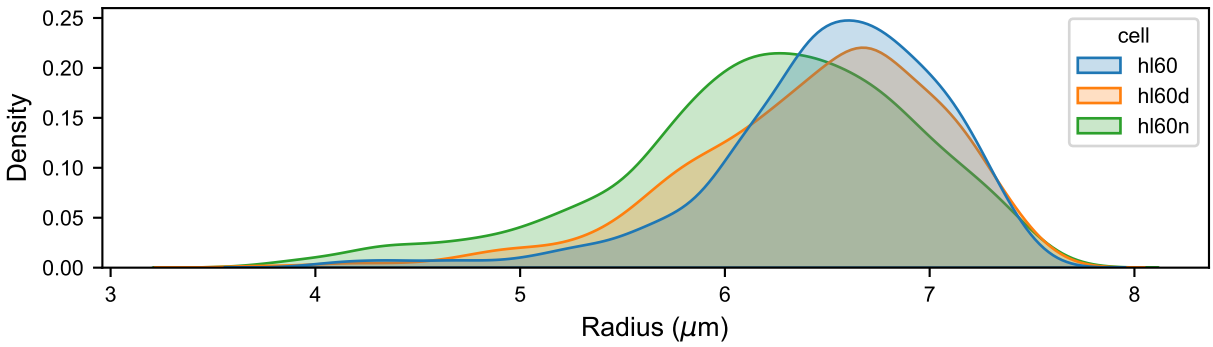

Figure 3.6: HL60 radii comparison. Distribution of radii for three cell populations.

learn and all relevant hyperparameters were optimized. Other models such as support vector machines (SVM) were tested and showed similar or less performance compared to RF (Figure 3.8). All models were trained across a mixture of multiple biological and technical replicates, and accuracy results were reported on a held on test set, in order to minimize bias.

We found that the trained RF model resulted in 75% accuracy for HL60 vs. HL60d, and 71% accuracy for HL60 vs. HL60n classification (Figure 3.7a,c). The lower accuracy of the model trained on HL60n can be attributed to the high rD1 overlap between the two classes. In the next step, we wanted to understand how the RF model weighted each of the different mechanotyping features in making its predictions. To this end, we utilized Shapley values, which were developed for coalitional game theory to inform one how to fairly attribute success to the constituent parts.[33] Thus, Shapley values can help us understand which features are most informative for a single cell classification. In order to make use of this method, the SHAP python library[97] was employed in conjunction with the trained RF model to attribute overall model performance to individual features (Figure 3.7b). The values reported in Figure 4b and 4d represent the mean impact for an individual feature in determining cell type. From the Shapley values we find that the R2 slope and rD1 have the most weight in deciding classification between HL60 vs. HL60d. For the HL60 vs. HL60n task, we find the R2 slope to have the highest impact, followed by the radius. The introduction of Shapley values provides interpretability of the mechanotyping data and demonstrates that the region from peak AR to the relaxed state contains the most information for classification. The analysis also revealed the importance of another temporal feature, the R2 slope, in the classification (Figure 3.3b). Based on these observations, we hypothesized that classification could be further improved by incorporating the sequential nature of a cell's deformation into the model design. We then decided to explore deep learning approaches to create a model that can extract shape dynamics.

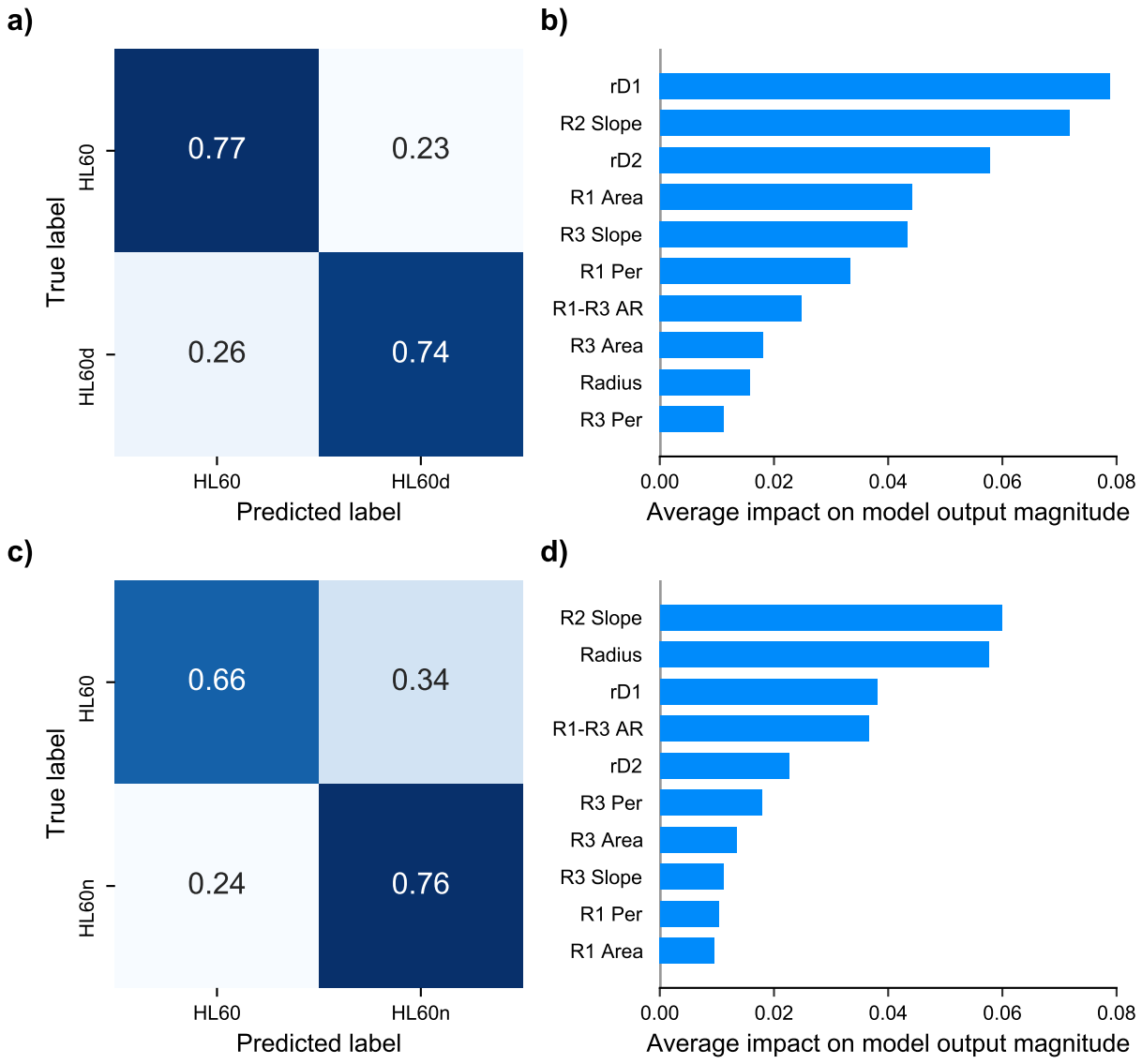

Figure 3.7: Prediction results of random forest using derived mechanotyping features. a) Confusion matrix of trained random forest predicting HL60 vs. HL60d. The values are normalized by the true label count. Accuracy is equal to the average of diagonal. b) SHAP feature importance plot obtained using trained RF model for the HL60 vs. HL60d classification. c) Confusion matrix for random forest trained on HL60 vs. HL60n prediction. d) SHAP feature importance for HL60 vs. HL60n.

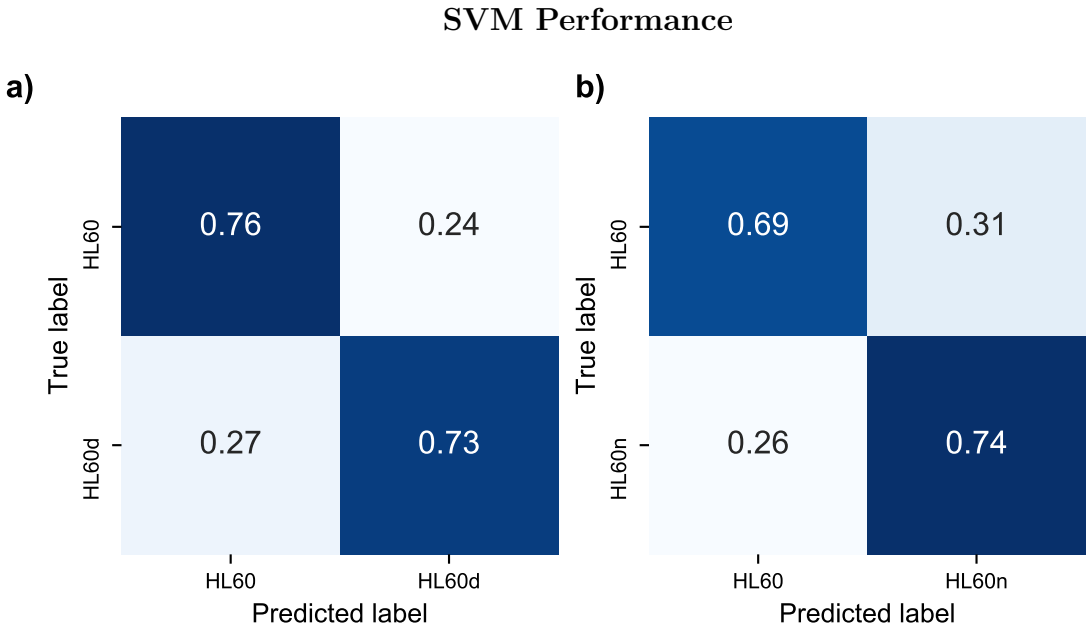

Figure 3.8: Confusion matrix for SVM. Accuracy equals average of diagonal elements.

3.3.5 Deep Learning for Enhanced Classification

RNNs are considered the optimal tools for handling sequential data[98] and are often used for translation and sequential prediction tasks. We chose RNNs since they can utilize the temporal nature of the shape dynamics, which in our case is complex due to the repeated deformations and relaxations caused by the non-linear shear force (Figure 3.2c). In short, RNNs function by receiving a single input from the full sequence, processing it and feeding the output into a copy of itself along with the next time step. We implemented a variant of RNNs called the gated recurrent unit (GRU) and used the sequential time-series features such as aspect ratio, perimeter, and area as inputs to these models. The trained GRU displayed a moderate increase in classification performance, as seen in the confusion matrices in Figure 3.9b,c, resulting in a 4% accuracy increase for HL60 vs. HL60d, and an 8% increase for HL60 vs. HL60n. Another RNN variant, the Long Short-Term Memory (LSTM) network, was tested but did not perform as well (Figure 3.10).

Since the application of RNNs yielded improved performance (Figure 3.9) compared to the RF models with manually derived features (Figure 3.7), we hypothesized that further improvement would require an approach that is not based on hand-selected features (or their combination). We therefore sought to improve accuracy by using binary masks as inputs, obtained using our video processing algorithm described above (see the sequence of blue-shaded regions in Figure 1). Note that the sequence of the masks are the only inputs; the derived deformation values from the masks are not used here. To utilize the time-series of masks, we added a number of convolutional layers to the GRU model (CNN-GRU). Convolutional neural networks are known to be very well suited for image based classification tasks. A schematic diagram of the model is shown in Figure 3.11a.

While CNN-RNNs traditionally use raw frames as inputs, here our input complexity is drastically reduced since segmentation had already been performed. Consequently, the network only learned the shapes of the deformed cells as opposed to differences in internal morphology or brightness contained in the raw images. Our CNN-GRU architecture was

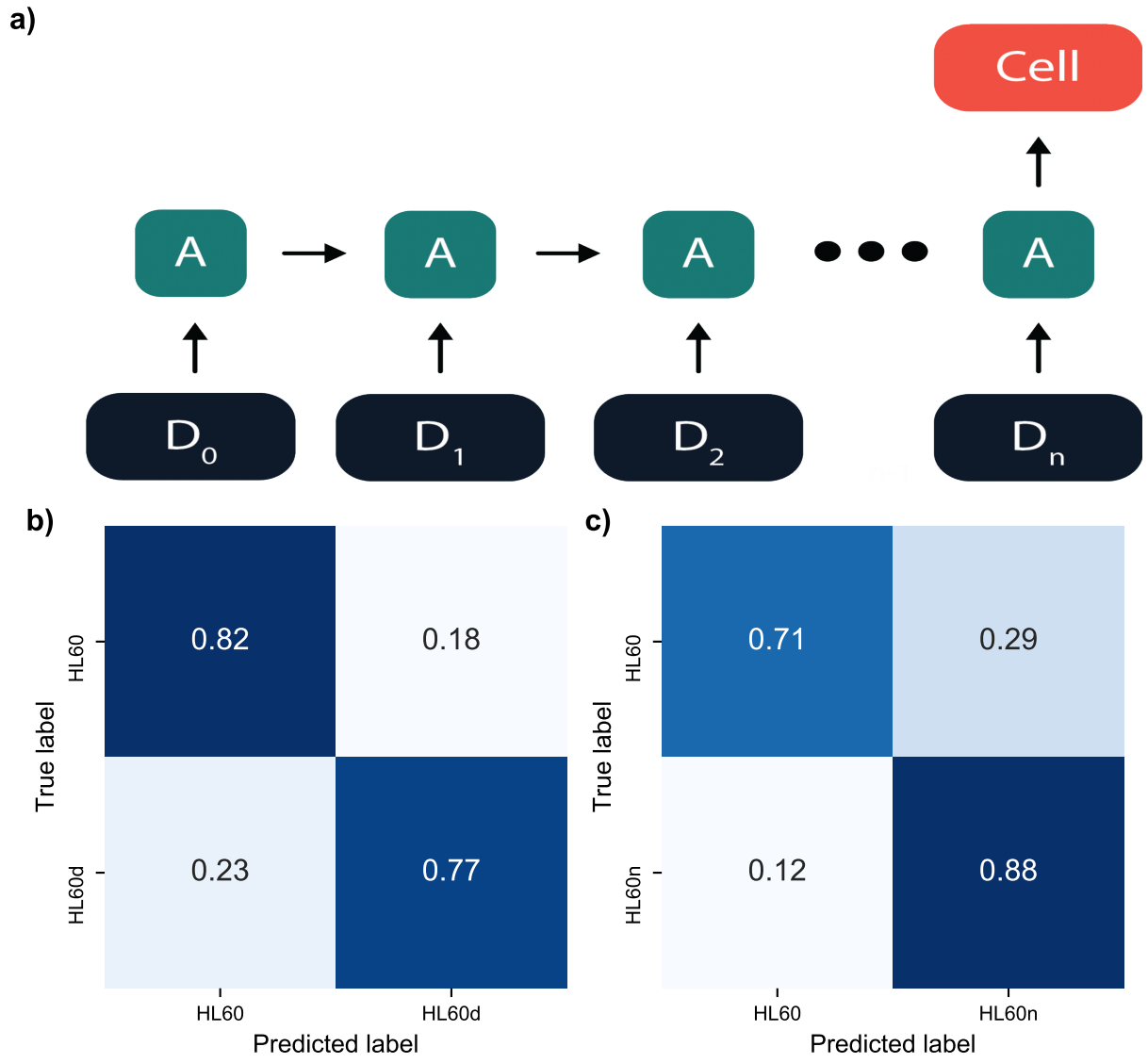

Figure 3.9: Time-series neural networks applied to mechanotyping features. a) Outline of recurrent neural network. Time-series deformation data is used as input into GRUs. The output of the network predicts cell phenotype. b) Confusion matrix for RNN trained on HL60 vs. HL60d. c) Confusion matrix for RNN trained on HL60 vs. HL60n.

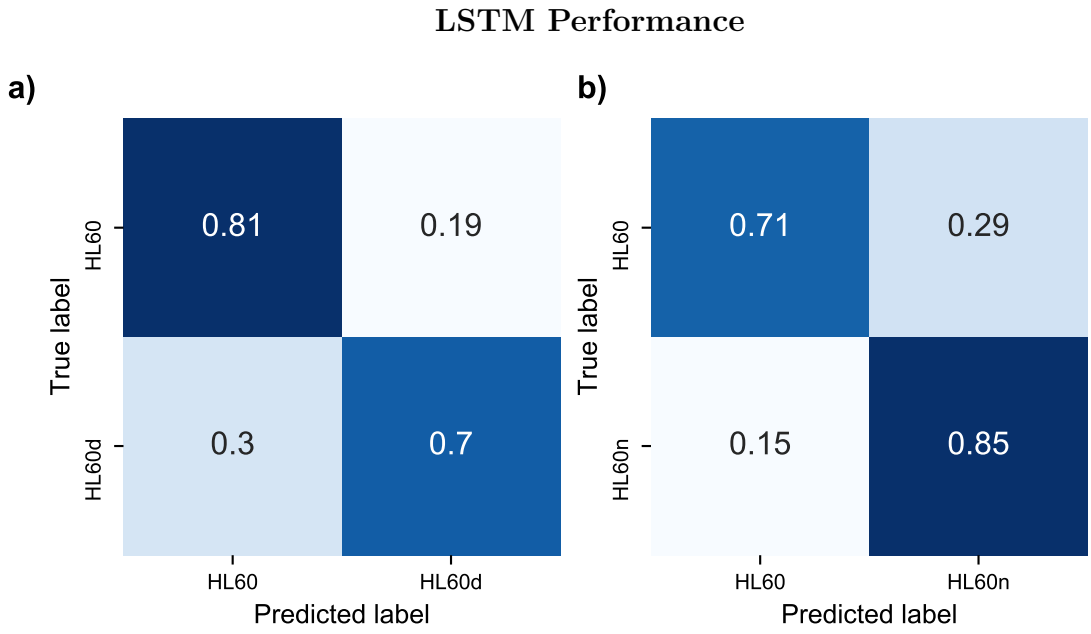

Figure 3.10: Confusion matrix for LSTM. Accuracy equals average of diagonal elements. a) Confusion matrix for HL60 vs. HL60d. b) Confusion matrix for HL60 vs.HL60n

refined by searching different filter sizes for the CNN layers, changing the number of dense and GRU layers, as well as adjusting the dropout rates. The optimized model architecture (Section 3.2.8) shows superior performance of the CNN-GRU to both the RF and GRU models (Figure 3.11b,c). Namely, the CNN-GRU enabled an increase of accuracy by 11% to HL60 vs. HL60d, and an increase of 6% to HL60 vs. HL60n, resulting in final classification accuracies of 90% and 85%, respectively. To investigate the potential of over-fitting, 5-fold cross-validation was performed in addition to the use of a train, validation, and test split. The 5-fold cross-validation resulted in accuracies of 91.4±0.6% and 83.7±0.9%, with similar results obtained for each training regime. The high validation set performance shows the richness of the deformability dynamics and their potential to aid in classification.

Finally, the importance of mechanotyping in classification of HL60 sub-populations was investigated. More specifically, whether these cells could be distinguished only through their morphological features, such as shape, prior to deformations was probed. To this end, a CNN was trained to classify HL60 vs. HL60d and HL60 vs. HL60n sub-populations using masks from the channel cavity, where no induced deformations are present. The best test accuracy attainable was 65% for HL60 vs. HL60d, indicating the poor classification potential of the undeformed shape (Figure 3.12b). However, when the same analysis was performed for HL60 and HL60n (Figure 3.12), a classification accuracy of 70% was observed, indicating that the Noco treatment affected the cells' undeformed shape, relative to the control HL60 population. The modified morphology of HL60n could have also contributed to the large increase in accuracy when using the CNN-GRU. Combining mechanotyping and deformation dynamics with morphology of cells could lead to improved accuracy of cells classification in cases where the initial morphology differs.

3.3.6 Traditional ML with Added Morphological Features VS. Deep Learning

To investigate the gain in performance obtained by deep learning over traditional machine learning methods, we extended the feature set of the manually derived features to include

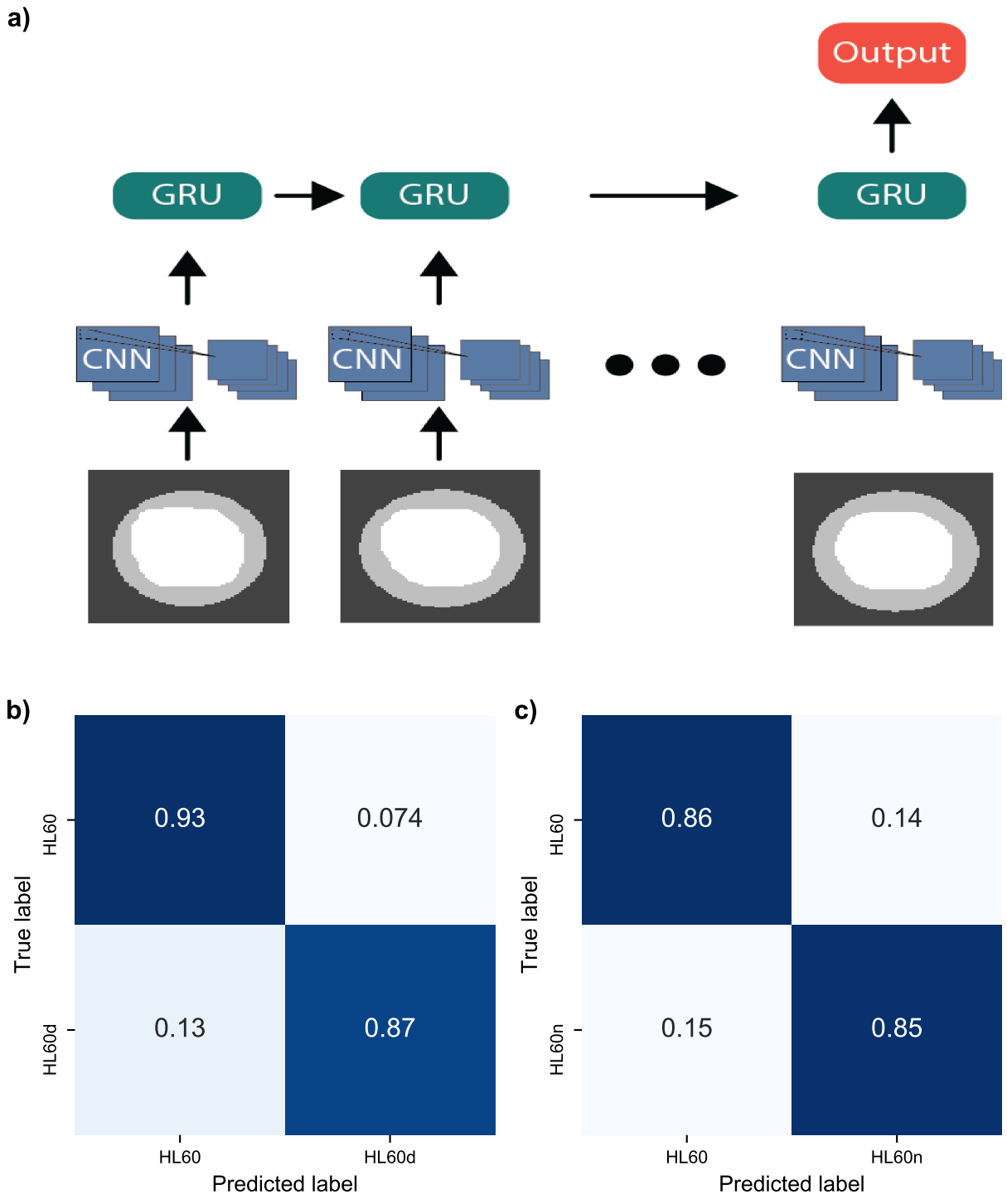

Figure 3.11: Classification comparison using sequence of cell masks. a) General flow of CNN-GRU. Sequences of masks are padded and used as inputs. CNN and GRU layers use identical weights for each time step. b) Confusion matrix for HL60 vs. HL60d. c) Confusion matrix for HL60 vs. HL60n.

Untreated Morphology vs. Treated Morphology

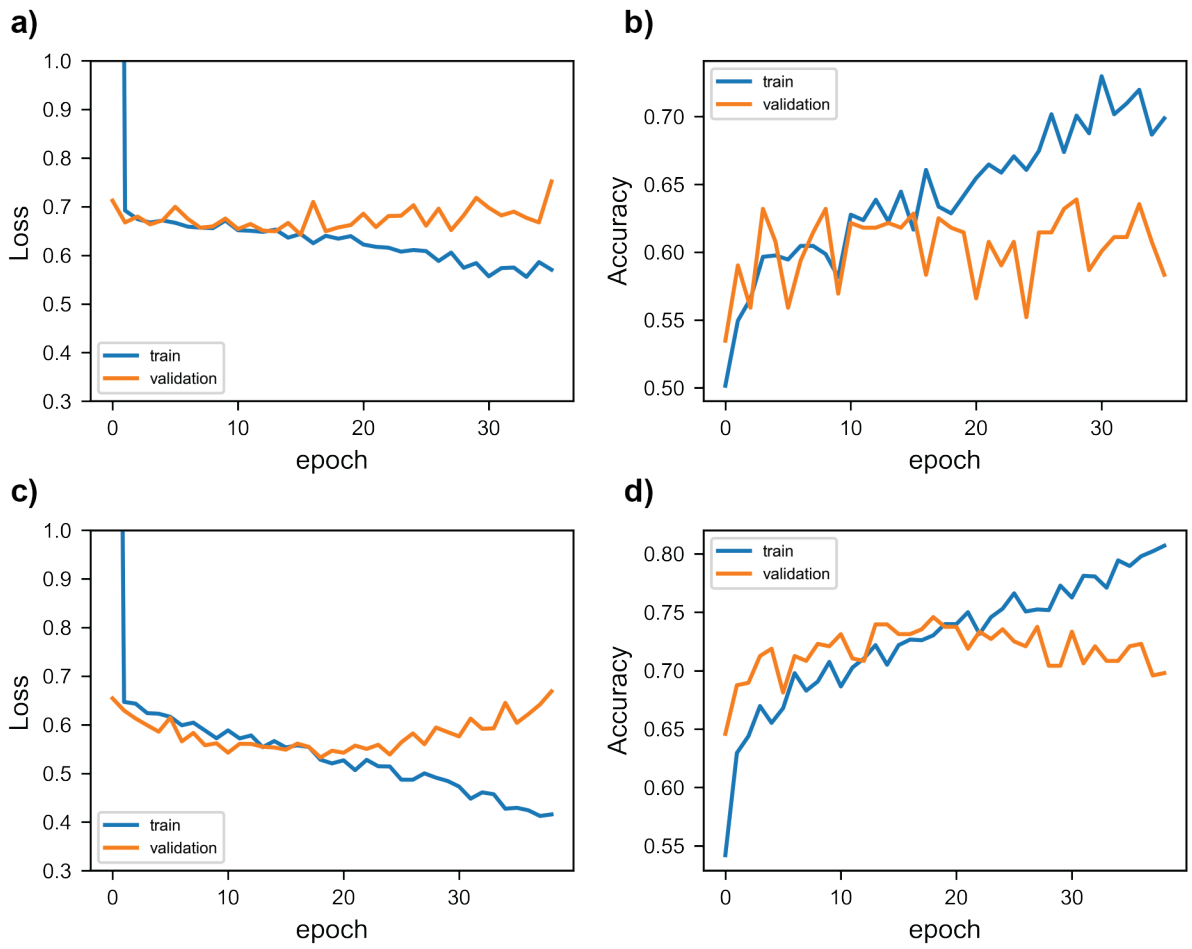

Figure 3.12: Training curve and validation set accuracy for CNN to detect morphology differences. a) HL60 vs. HL60d binary cross-entropy loss. b) HL60 vs. HL60d accuracy metric as a function of training. Final accuracy was measured using a held out test set. c) HL60 vs. HL60n loss. d) HL60 vs. HL60n accuracy.

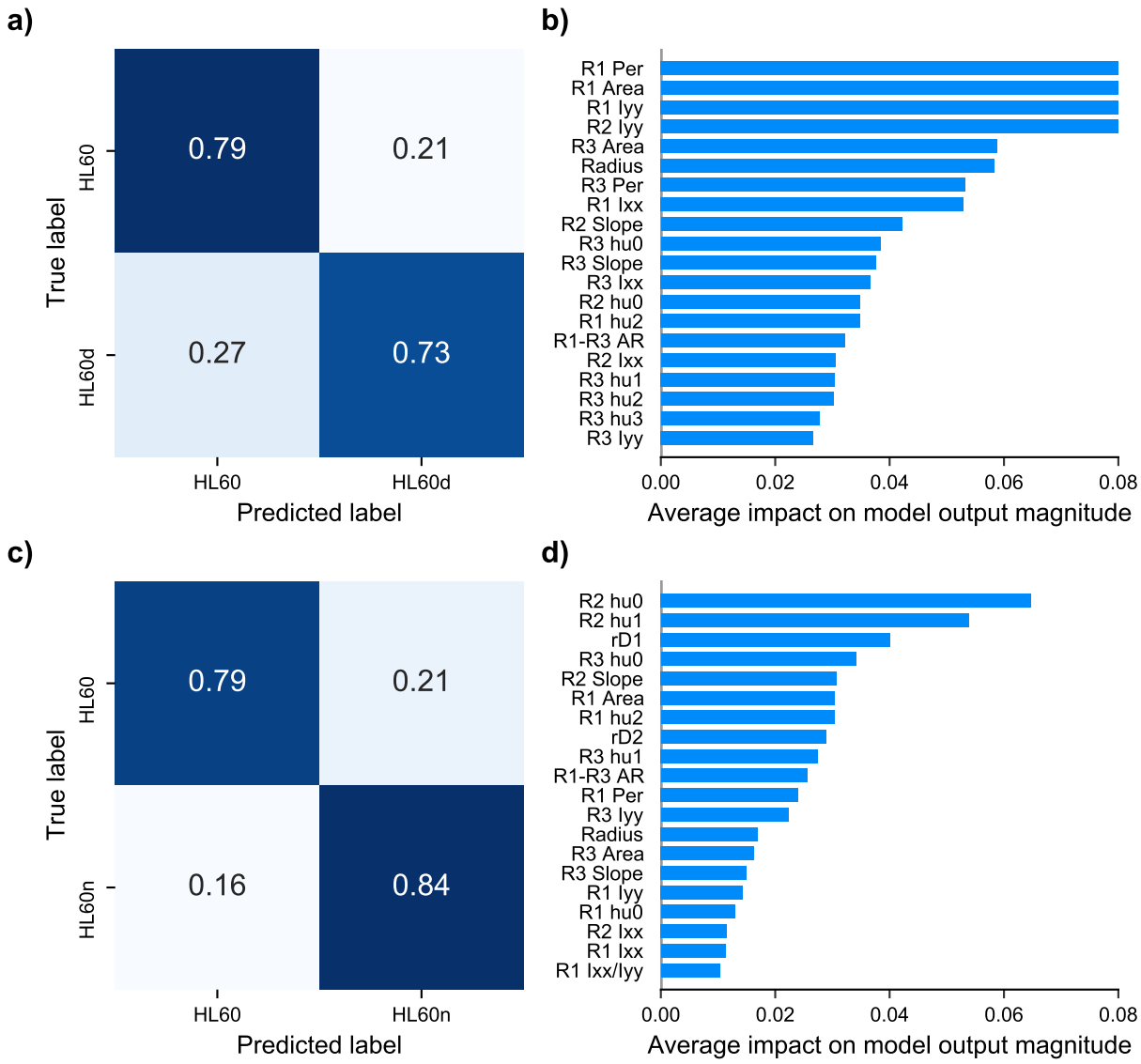

Figure 3.13: Performance for optimized SVM on set of full mechanical and morphologically derived features. a) Confusion matrix for HL60 vs. HL60d. b) Corresponding SHAP plot for HL60 vs. HL60d. c) Confusion matrix for HL60 vs. HL60n. d) Corresponding SHAP plot for HL60 vs. HL60n

more descriptors of morphology. The added features were calculated in regions R1, R2, and R3 (Figure 3.2) and included the central moments of inertia along with Hu moments,[99] which are a set of seven two-dimensional moments that are invariant to rotations, scaling, and translation (Figure 3.13). Using these new morphological descriptors, along with the previous derived features (Table 3.1), a new SVM was trained and optimized through a grid search of relevant parameters. The resulting confusion matrix on a held-out test set and SHAP plots were calculated to understand the importance of these added features for classification (Figure 3.13). The classification accuracy between HL60 and HL60d showed a marginal increase in performance (1%) compared to the SVM trained without the additional feature set. This, along with the poor classification of the undeformed shapes (65%) (Figure 3.12), suggest that the HL60 and HL60d cells are very similar morphologically, and that deformation-based features are the most informative for classification. The classification of HL60 versus HL60n was moderately improved by the newly added morphological features ($\sim$10%). The corresponding SHAP plot (Figure 3.13d) shows that the Hu moments in R2, where the cells are most relaxed, had the highest average impact on classification. The increase in classification accuracy, along with the importance of features in R2, suggests that both morphological and deformation-based features contributed to the classification of HL60 and HL60n.

While the addition of 30 new morphological features demonstrated an increase in SVM classification accuracy, it is impossible to ensure that our set of derived features forms a complete basis to describe all possible shapes the cells assume throughout their deformation and relaxation. Furthermore, we argue that features like the Hu moments, although useful for increasing classification, offer little interpretability and cannot be used to inform future physical models. In cases where one is most concerned with classification accuracy over interpretability, we found that deep learning, in particular a network with both convolutional and recurrent layers, outperforms traditional methods such as an SVM. The classification accuracy of the CNN-GRU was $\sim$15% higher for HL60 vs. HL60d, and $\sim$4% for HL60 vs.

HL60n. We assert this increase in performance is due to the ability of convolutional layers to generate a more complete feature set, along with the recurrent layers to learn the temporal dependence of the same features. It is important to highlight that the use of the deep learning models, like the CNN-GRU described here, currently presents almost no interpretability. However, this comes with a trade-off of improved classification accuracy. The significance of classification is for the eventual sorting of cells.

3.3.7 Calculation of Enrichment

To understand how classification accuracy is mathematically related to sorting efficiency we imagine a use case in which we have a mixed sample of several cell populations and seek to characterize one rare population of cells within. We begin with n_{sample} cells from the biological sample of interest, perhaps from culture or dissociated tissue. The optimal strategy to analyze the most rare cells would be to characterize all cells in the sample, from which one would recover $r\ n_{\text{sample}}$ rare cells, where r is the fraction of target cells in the sample. However, characterizing every cell to find the rare ones could be prohibitive or even untenable. For example, if one sought a rare population with a frequency of 1/1000, a typical single cell RNA sequencing run of 10,000 cells would yield data on only 10 of the rare cells. If the initial population of cells only numbered 10,000, this would be the best one could do without more information. However, if one started with millions of cells but only had the resources to sequence 10,000 of them, many more of the target rare cells could be characterized if the sample were first sorted for the rare cells.

We imagine a future version of our device which could enrich a sample by mechanically characterizing the cells, then sorting the output into positives and negatives for a defined profile of the mechanical characteristics of the rare cells. The positive subsample would be composed of true positives (TP) and false positives (FP), where true positives correspond to correct model predictions of the rare cell population to be enriched. False positives would therefore correspond to incorrect predictions of the rare cell population. TP and FP are

characterized by the true positive rate (TPR) and the false positive rate (FPR) respectively.

The resource limitations dictate that the subsequent analysis can accept a limited number of cells, $n_{\text{downstream}} < n_{\text{sample}}$. We seek an enriched sample of size $n_{\text{downstream}}$ that contains the maximum possible number of true positives. To do so, we mechanically characterize n_{RDRUC} cells. Therefore, we maximize the number of true positives, $n_{\text{RDRUC}}\, r$ TPR, subject to the constraint $n_{\text{RDRUC}} \leq n_{\text{sample}}$. We find the number of true positives recovered.

$$n_{\text{enriched}} = \begin{cases} n_{\text{sample}} r\text{TPR}, & 0 < n_{\text{sample}} \leq \frac{n_{\text{downstream}}}{\text{FPR}-r\text{FPR}+r\text{TPR}} \\ \frac{n_{\text{downstream}} r\text{TPR}}{\text{FPR}-r\text{FPR}+r\text{TPR}}, & n_{\text{sample}} > \frac{n_{\text{downstream}}}{\text{FPR}-r\text{FPR}+r\text{TPR}} \end{cases}. \tag{3.1}$$

The number of true positives recovered increases with the sample size, up to the maximum value given by the second case. If an experimenter knows the number of cells possible to analyze downstream and can estimate both the rarity of the cells they seek and the classification performance, the sample size threshold could be useful in designing experiments.

Using this result, we now seek to quantify the impact of the classification performance on the quality of the sample passed on from mechanical characterization. We define the maximum enrichment,

$$\text{enrichment} = \frac{n_{\text{enriched}}}{n_{\text{downstream}} r} = \frac{\text{TPR}}{\text{FPR} - r\text{FPR} + r\text{TPR}}. \tag{3.2}$$

We find that the maximum enrichment increases faster than linearly with decreasing FPR, as shown in figure 3.14. Therefore, seemingly small increases in classification performance can have large effects on the usefulness of a sorter.

Each machine learning model is capable of a range of true positive and false positive rates, visualized as a receiver operating curve (ROC) in Figure 3.15, left. Because the enrichment is much more sensitive to false positive rate than true positive rate, the maximum enrichment does not necessarily occur at the same point along this curve as the maximum accuracy, where the model is evaluated for the confusion matrices in the main text. To find the best

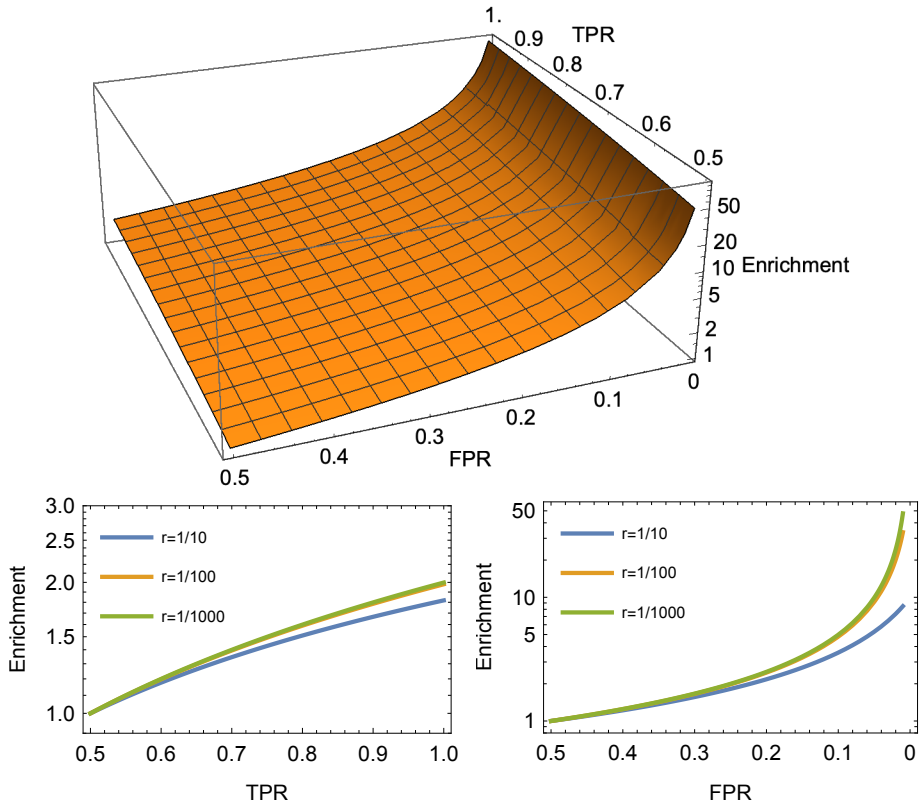

Figure 3.14: Maximum enrichment.
Top: Enrichment as a function of both true positive rate (TPR) and false positive rate (FPR) for target cell rarity $r = 1/1000$.
Bottom left: Maximum enrichment as a function of TPR shown for FPR fixed at .5 and a range of target cell rarities, r.
Bottom right: Maximum enrichment as a function of FPR shown for TPR fixed at 0.5 and a range of target cell rarities, r.
Enrichment shown on a log scale in all three subfigures.

possible max enrichment for a given model, we evaluate max enrichment at each point along the ROC, as shown in Figure 3.15, right. The greatest enrichment occurs near the edge of the ROC, where the number of samples tends to be lower than further along the curve because of high rejection rates. Therefore, we evaluate the best enrichment for a given model by averaging the three highest max enrichment values generated along the ROC curve.

We find that for a sorter with the performance of the random forest classifier, the maximum enrichment is 17 for HL60d cells (Figure 3.7), while a sorter with the accuracy of the CNN-GRU classifier for the same cells (Figure 3.11) has a best maximum enrichment of 63 (r=1/1000). This corresponds to an increase from 166 rare cells found to 635 rare cells found for a case where $n_{\text{sample}} = 1,000,000$, $n_{\text{downstream}} = 10,000$, and $r = 1/1000$. Note that only $n_{\text{downstream}}\, r = 10$ rare cells would be found without sorting.

3.3.8 Increased Accuracy Leads to Greater Enrichment

To explore the implications of an increased classification accuracy, we considered a hypothetical case in which a researcher has a heterogeneous sample of cells with many mechanical phenotypes and desires to characterize a sub-population within the sample. The sub-population is at most a few percent of the total sample, making conventional analyses, such as single cell RNA sequencing, microscopy, etc., difficult. We envision our method being implemented to identify the population of interest by combining deformability characterization, classification, and in the future sorting. In this case, the classification accuracy determines how effectively rare cells can be sorted out of the mixed sample. We define enrichment as the increase in the concentration of target cells in the sorted sample. We find that enrichment depends on classifier performance, which increases exponentially as the false positive rate decreases below 0.5, then super-exponentially for false positive rate less than ~ 0.2, as shown in Figure 3.14. This dependence gives context to the importance of the increased classification performance we achieve. Using only morphological features, the classifier performance would allow enrichment of $\sim$9 times. Using the mechanotyping data

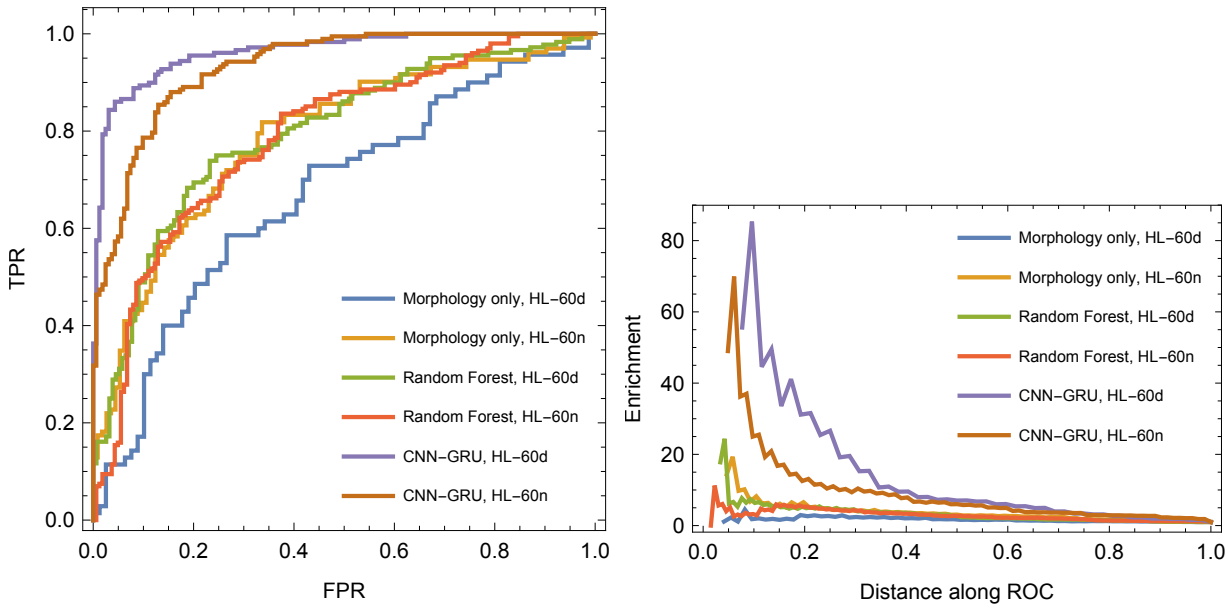

Figure 3.15: Evaluation of best enrichment.
Left: Receiver operating curves (ROCs) for several of the models presented in the text.
Right: Max enrichment evaluated for each point along the ROC. Data is excluded for points where true positive rate (TPR) or false positive rate (FPR) were 0. Shown for rarity r=1/1000.

but with the inferior random forest classifier would allow an enrichment of ∼12 times. Finally, using the CNN-GRU model would allow an enrichment of ∼58 times (each averaging over prediction performance numbers for HL60d and HL60n, and assuming a rarity of 1/1000). Thus, the CNN-GRU classifier has about five times the potential effectiveness of the random forest classifier for future sorting applications.

3.4 Conclusions

Here we show the application of a channel with an undulating width that induces nonlinear forces to cells at a high throughput. The cells undergo multiple deformations and relaxations which reveal a multitude of information on cellular mechanics. We explore how much information is revealed from this dynamic and non-linear deformation process by comparing the classification accuracy of traditional machine learning models, with derived features, to deep learning models with automatic feature extraction. For the traditional machine learning models, the use of SHAP values revealed the mechanotyping parameters that contributed the most to classification, which can be useful in gaining insight on how cell populations differ mechanically. At the cost of interpretability, the deep learning models showed an appreciable increase in classification accuracy, particularly in the case where the two cell populations were very similar morphologically.

While deep learning characterization and classification are currently performed post hoc, similar deep learning models with proper hardware can currently process ∼2,000 fps, leading to the possibility of real-time sorting. We envision this work being incorporated into existing technologies, such as imaging flow cytometry, and to be especially important for the segregation of rare cells, including circulating tumor cells[100] and even cancer stem cells.[101]

Future work will include optimizing channel designs with different widths, lengths, and shapes, as well as extending the deep learning models to include unsupervised clustering.

Chapter 4

Deformability Cytometry Clustering with Variational Autoencoders

4.1 Introduction

In this chapter, deep learning models for unsupervised cell classification are investigated. The dataset for this study was HL60 cells before and after treatment with CytoD,[102] as described in Chapter 3. Unsupervised cell classification can be investigated using two routes (Figure 4.1). In a first approach, features, such as region deformation, relaxation rate, and peak deformation, can be extracted manually or by a feature extraction algorithm, and subsequently utilized by a clustering algorithm. In a second approach, deep learning methods can be used to better access the high dimensional latent space for clustering.

Unsupervised clustering is first investigated using traditional clustering algorithms: a Gaussian mixture model (GMM) and spectral clustering algorithm, with principle component analysis (PCA) extracted features. However, these scalar features contained too much class overlap and the clustering task was too challenging for the clustering algorithms to accurately separate the two populations of cells. This provided motivation to look to deep learning models for clustering, namely a variational autoencoder (VAE), which has shown success in clustering cell data.[103,104]

Here, a novel unsupervised clustering algorithm using a VAE for deformability cytometry was developed, which is referred to as DeepDeform. A VAE alone was tested first as a benchmark of clustering performance. However, the VAE was limited in its ability to cluster cell populations, since it is only capable of compressing data and is not inherently a classifier. The performance of the VAE was then assessed in tandem with the previously mentioned traditional clustering algorithms, which showed poor clustering performance likely due to

improperly balanced clusters.[105] In DeepDeform, a semantic clustering by adopting nearest neighbors (SCAN) loss is integrated to abrogate the deficiencies of traditional VAEs. The implementation of SCAN enabled better access to the high dimensional latent space for clustering cell populations, since it did not require compression of the latent space. Using DeepDeform, the clustering accuracy of the two cell populations rivaled that of several supervised methods.[106] Additionally, the accuracy approached that of supervised models when selecting confident samples.

This chapter is organized as follows. Section 4.2 provides an overview of the methods, such as the dataset and training procedures, similar to that described in Chapter 3. Section 4.3 describes initial efforts to investigate cell populations using conventional clustering and dimensionality reduction techniques, such as principal component analysis (PCA). Section 4.4 presents the development, optimization, and performance of unsupervised clustering using DeepDeform on the HL60 model system.

4.2 Methods

Channel Design

The dataset for unsupervised clustering was obtained using the same microfluidic channel design as described in Section 3.2.2.

4.2.1 Dataset

The dataset for unsupervised clustering included 2239 human leukemia 60 (HL60) cells. The two classes are comprised of untreated (HL60) and cytochalasin D (CytoD) treated HL60 cells (HL60d). There is a roughly even split (51% - 49%) between the two classes: HL60 and HL60d. CytoD, a fungal toxin, disrupts actin polymerization, resulting in a decrease in cell stiffness, while leaving morphology and radius relatively unaffected. This means that any clustering algorithm needs to rely heavily on mechanical properties to separate the populations rather than more accessible information such as a cell's morphology or radius.

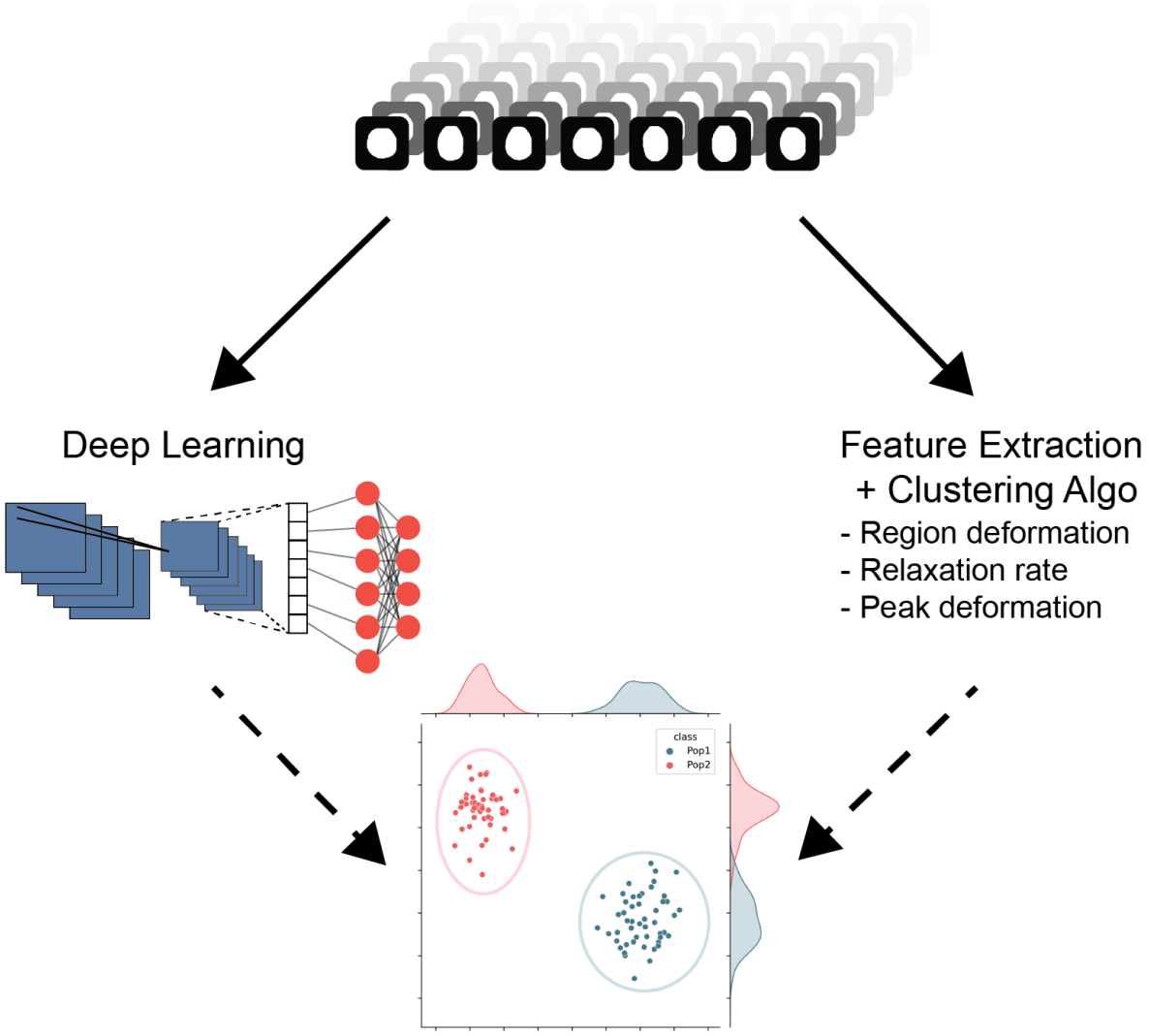

Figure 4.1: Two potential routes to clustering deformability cytometry data exist.

Since the class labels are known, an unsupervised model's ability to extract these known labels can be evaluated without the model training on them directly. There are 4 different biological replicates of HL60 cells and 4 biological replicates of HL60d cells. Multiple technical replicates were also gathered with the same density of cells being used for each session. The data gathered consist of spatially synchronized 35 frame sequences of 96×96 binary masks of cells passing through the channel. The video data can be represented in fewer dimensions by identifying important features such as peak deformation, change in deformation in the first and second regions, as described previously.[102]

Features

The features (Table 3.1) for the unsupervised clustering were extracted from the raw binary masks as explained in Section 3.2.3.

4.2.2 VAE Architecture

Provided below is the VAE architecture of DeepDeform. Raw binary mask data were used from the previous work (Section 3.3.2). Adam was used as an optimizer with a learning rate of 1e-3. Gradients were clipped at a norm of 3.0. Experiments were tracked using wandb.[107] The loss function was the sum of the reconstruction (Figure 4.2) loss and the KL divergence (Figure 4.4). Both training and validation losses were monitored during training in addition to the average class of the five nearest neighbors (Figure 4.3). The batch sizes were 32. Data were shuffled each iteration. Unless specified, all dense and convolutional activation functions were the leaky rectified linear activation unit.

Encoder Network:

1. Conv2D - Filters: 8 - Kernel: 3×3
2. MaxPooling2D
3. Conv2D - Filters: 16 - Kernel: 3×3

4. MaxPooling2D

5. Conv2D - Filters: 32 - Kernel: 3×3

6. MaxPooling2D

7. Flatten

8. GRU - Dimension 32

μ Network

1. Dense - Dimension 8

σ Network

1. Dense - Dimension 8

Decoder Network:

1. Repeat Vector - Number of repeats: Number of input timesteps

2. GRU - Dimension 32

3. Dense - Dimension 24×24×64

4. Reshape - 24×24×64

5. Conv2D Transpose - Filters: 64 - Kernel: 3×3

6. Conv2D Transpose - Filters: 32 - Kernel: 3×3

7. Conv2D Transpose - Filters: 16 - Kernel: 3×3

4.2.3 Implementation of Semantic Clustering by Adopting Nearest Neighbors (SCAN)

Once encoder training was completed, the model weights were frozen and a single two dimensional, softmax-activated, dense projection layer was added. The SCAN loss was implemented according to the work of Gansbeke et al.[108] with two separate Adam optimizers for the consistency (Figure 4.6) and entropy (Figure 4.5) components respectively. Both optimizers had a beta_1 of 0.9, a beta_2 of 0.999 and an epsilon of 1e-7. The learning rate of the consistency optimizer was set to 1e-3 while the entropy optimizer was 5e-3.

$$\Phi'^{c}_{\eta} = \frac{1}{m} \sum_{X \in D} \Phi'^{c}_{\eta}(X) \tag{4.1}$$

$$\text{entropy component} = \sum_{c \in C} \Phi'^{c}_{\eta} \log \Phi'^{c}_{\eta} \tag{4.2}$$

$$\text{consistency component} = \frac{1}{m} \sum_{X \in D} \sum_{k \in N_x} \log \left(\Phi_{\eta}(X) \cdot \Phi_{\eta}(k) \right) \tag{4.3}$$

$$\text{SCAN loss} = \lambda * \text{entropy component} + \text{consistency component} \tag{4.4}$$

4.2.4 Code Availability

The python scripts implemented for data processing and training machine learning models are available at: https://github.com/siwylab/vae_dc

4.3 Unsupervised Clustering Using Manually Extracted Features

In previous work, we observed that for a supervised task, machine learning was required to extract the most useful information out of the data (Section 3.3.4). Given that many problems in science are open-ended and do not have known labels, it is necessary to extend

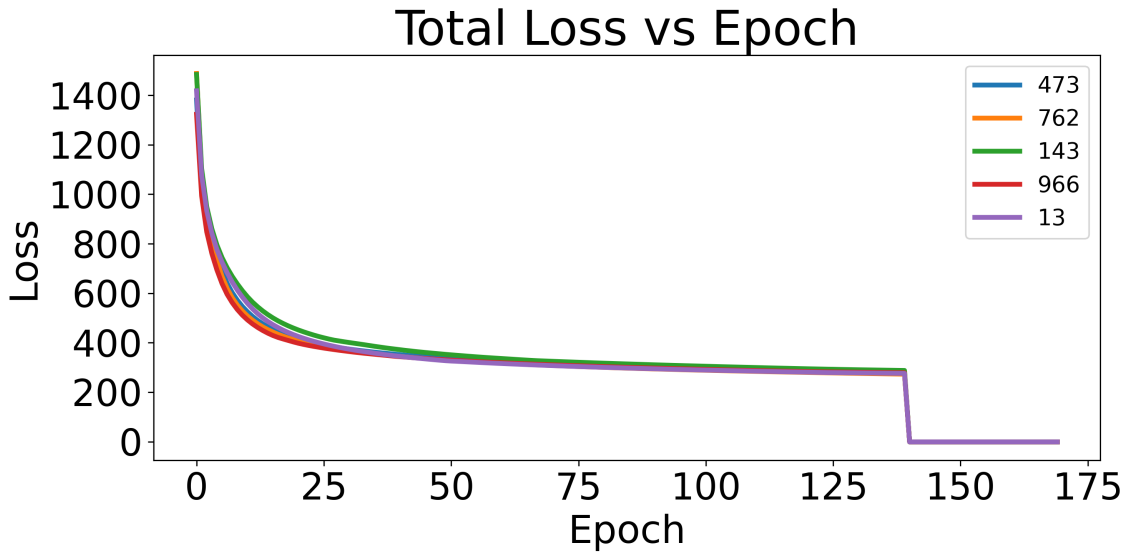

Figure 4.2: Total training loss curves are shown for HL60 vs. HL60d. Each line represents a different random seed. At 140 epochs the model is trained using the SCAN loss.

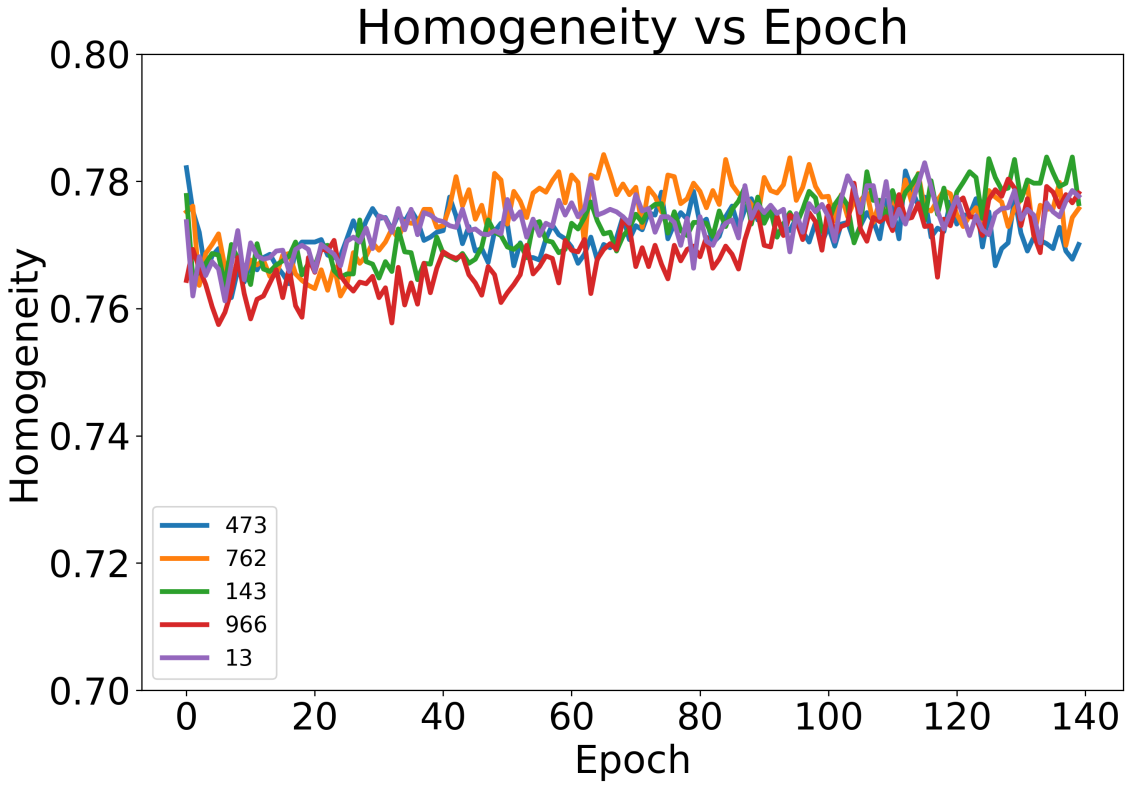

Figure 4.3: Total average homogeneity of each point's nearest 5 neighbors is plotted during VAE training.

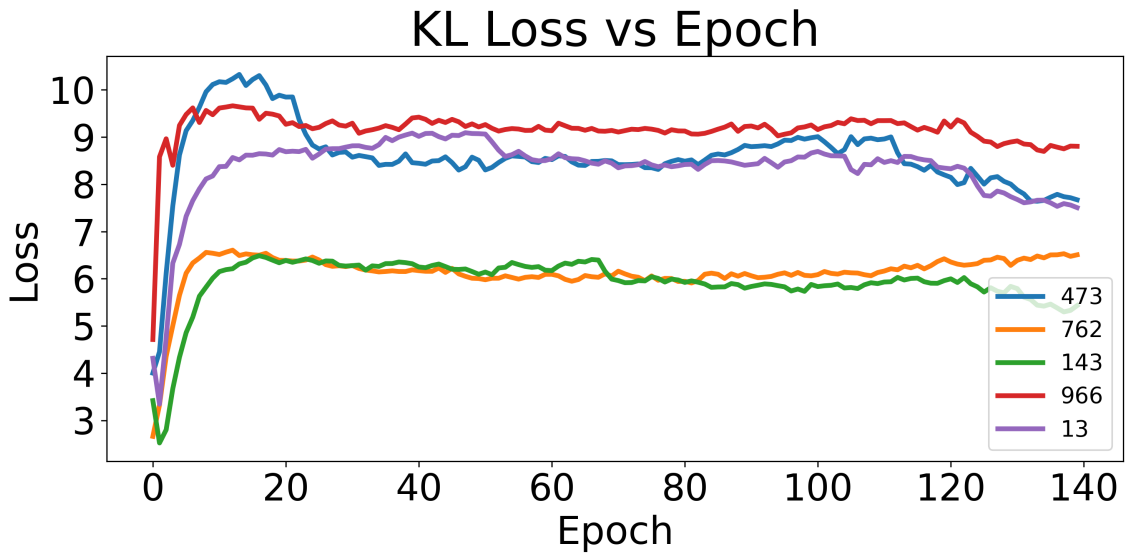

Figure 4.4: KL loss curves are shown for HL60 vs. HL60d. Each line represents a different random seed.

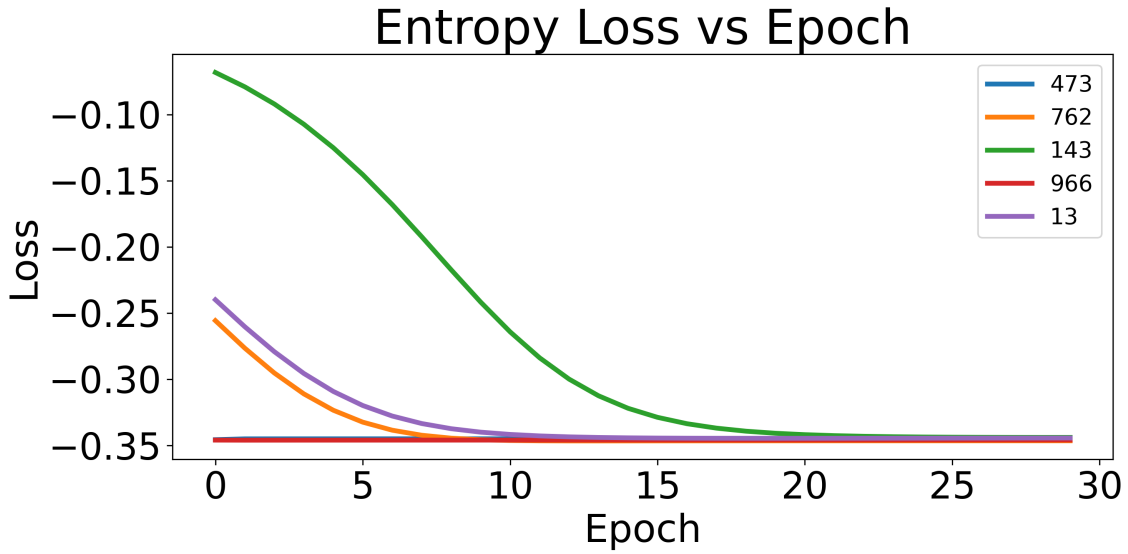

Figure 4.5: Entropy component of SCAN loss vs. epoch.

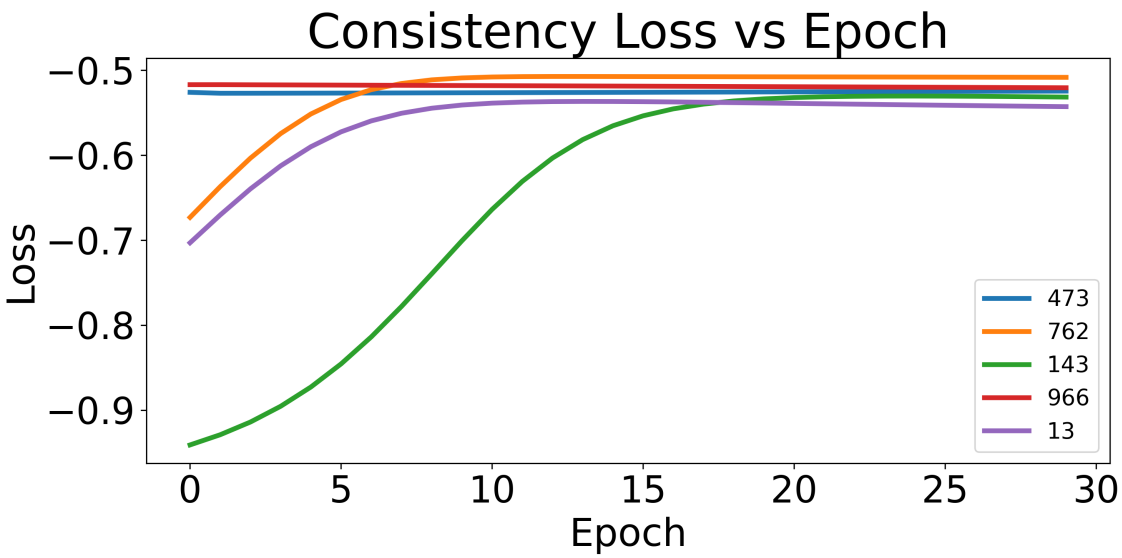

Figure 4.6: Consistency component of SCAN loss vs. epoch.

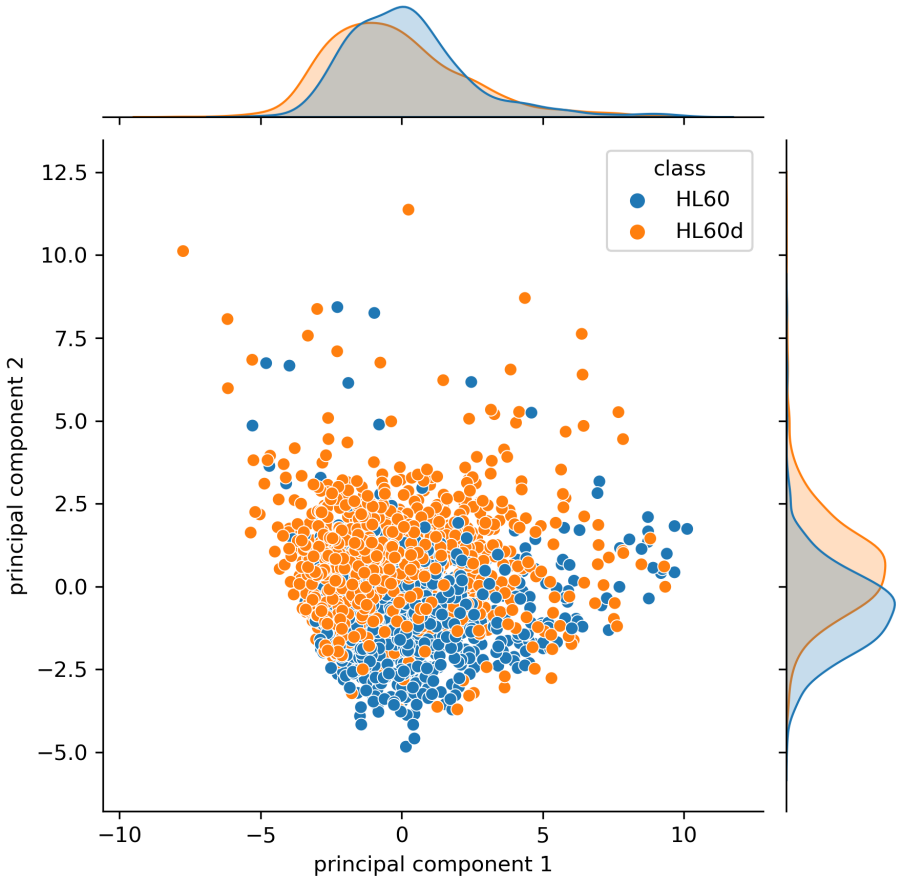

Figure 4.7: First two principal axes from PCA, ground truth labels are shown.

our deformability platform to unsupervised problems. It is not, however, clear what the ideal route towards creating a clustering algorithm is (Figure 4.1). One route, utilizing manually extracted features, could offer a simple and interpretable model for classification. Meanwhile, a deep learning could utilize the raw binary masks and could offer improved clustering accuracy.

To investigate the first route, the ten manually extracted features (Table 3.1) previously found useful for supervised classification were first considered. These data were a good starting point since they have a much lower dimensionality than the sequence of raw masks (10 vs. 35×96×96). However, the data were first condensed further since most clustering algorithms work best with two or three dimensions. The dimensionality was further reduced down to two dimensions with PCA, which is a lightweight and accessible technique for dimensionality reduction. With a two-dimensional input space, the practicality of existing clustering algorithms: the Gaussian mixture model (GMM) and the spectral clustering model were assessed with the data. Both algorithms are easily implemented and have been previously used for clustering biological data.[11,109] The desired task for these models was to learn patterns inherent to the data that would allow them to correctly assign class labels (HL60/HL60d) without training with the labels directly. The compressed features are shown in Figure 4.7. Both the GMM and the spectral clustering model showed poor performance with a clustering accuracy of 55.4 % and 50.0 %, respectively. This poor performance was due to the large overlap between the two classes in the lower dimensional space created by PCA. Additionally, a large amount of class overlap exists in the hand-picked features even before compression by PCA.

4.4 Unsupervised Clustering Using Deep Learning

It was hypothesized that a more powerful dimensionality reduction technique would compress the input data more effectively than manually extracting features. This would enable the use of richer data, such as the raw binary masks. A variational autoencoder (VAE) was therefore used as an algorithm to access the rich latent space created by efficient

compression of binary masks. VAEs reduces the dimensionality of input data to arbitrary dimensions[29] and has been previously used for clustering cell data.[103,104]

4.4.1 VAE with Traditional Clustering

To maximize the potential of the model, the raw sequence of binary masks are used as inputs, which contained more data than the manually extracted features. The VAE was created using a convolutional recurrent encoder and a corresponding decoder. The decoder network also contained a recurrent layer, like the encoder, though it used convolutional transpose layers, in order to scale up the decoder input data to create an output size equal to the raw binary masks. The full VAE architecture is provided in Figure 4.8. The VAE was trained on the sum of the reconstruction loss and the Kullback-Leibler divergence (Equation 1.3) with an encoder output size of 32 dimensions. This size was chosen through a hyperparameter sweep that optimized the loss of the VAE.

The GMM and spectral clustering models were implemented in order to utilize this latent space created by the encoder. The results are provided in Figure 4.9. Since the latent space created by the encoder was still too high-dimensional (32-dimensional) for the clustering algorithms to be used on their own, PCA was used to further reduce the dimensionality to two. Both clustering algorithms tested showed an improved mean accuracy but noisy performance ($\pm 3.5\%$). The unstable performance could be caused by an insufficient balance of class assignments.

To seek improvements, the data were carefully studied. Interestingly, each point was observed to contain a high nearest-neighbor homogeneity, meaning that a point was likely to be surrounded by other points belonging to the same class. We then looked to the literature to see how this structure in the latent space can be harnessed to improve clustering accuracy. A technique designed by Gansbeke et al.[108] takes advantage of such a structure, but in the context of unsupervised image clustering. Their method, among other things, uses a loss function (Equation 1.3) that encourages balanced and confident class predictions. It was

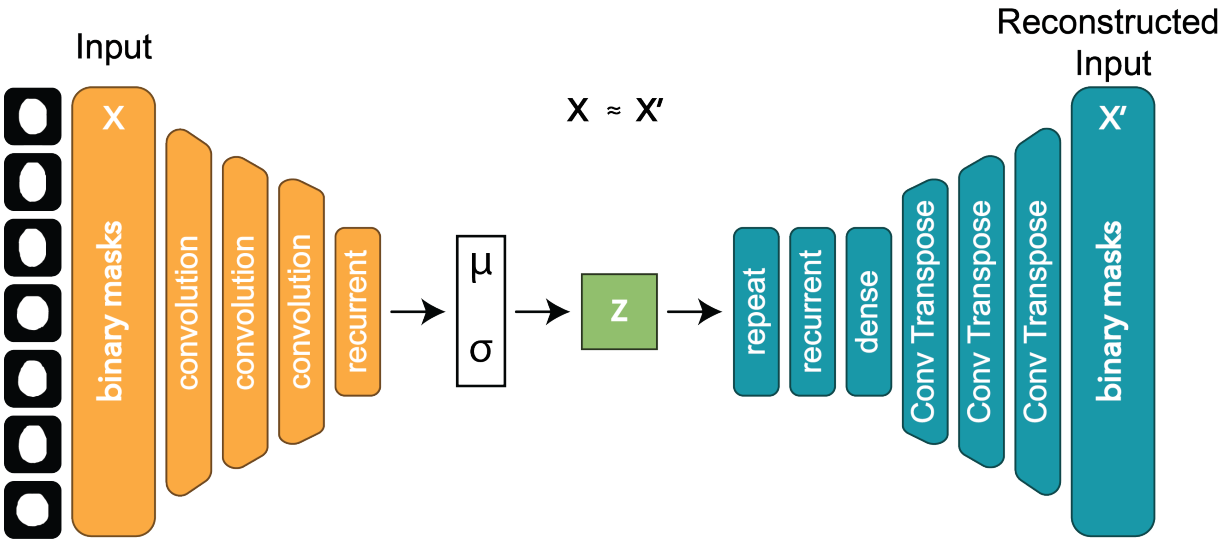

Figure 4.8: The VAE architecture is used for reconstructing the sequence of binary masks. The encoder network is colored orange, and the decoder network is colored blue.

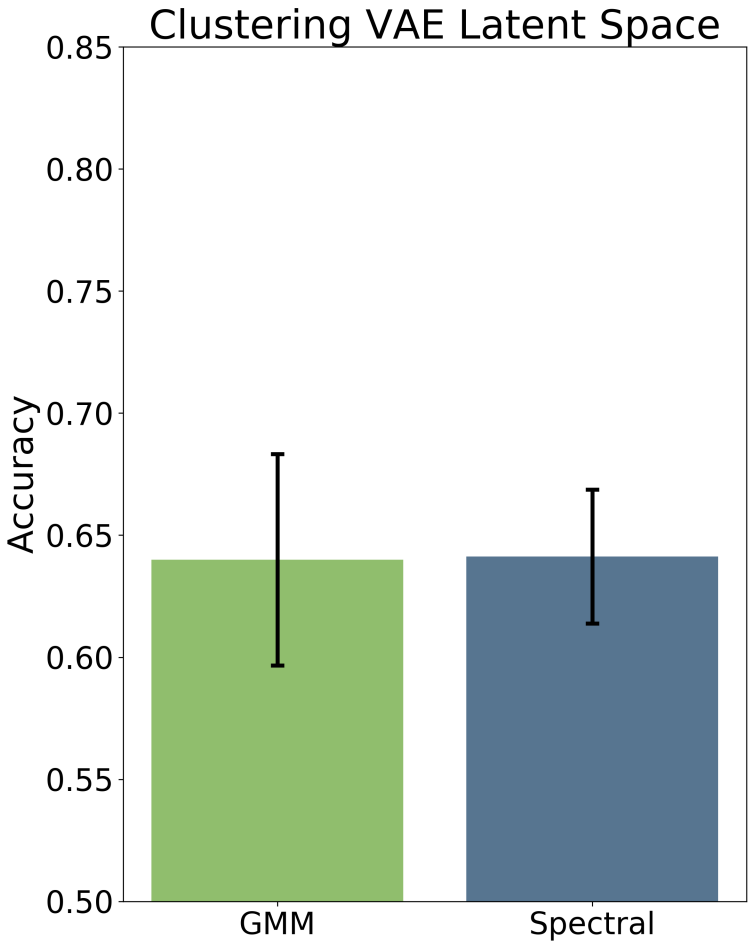

Figure 4.9: Accuracy of GMM and spectral clustering on trained VAE latent space. Five different random seeds are tested.

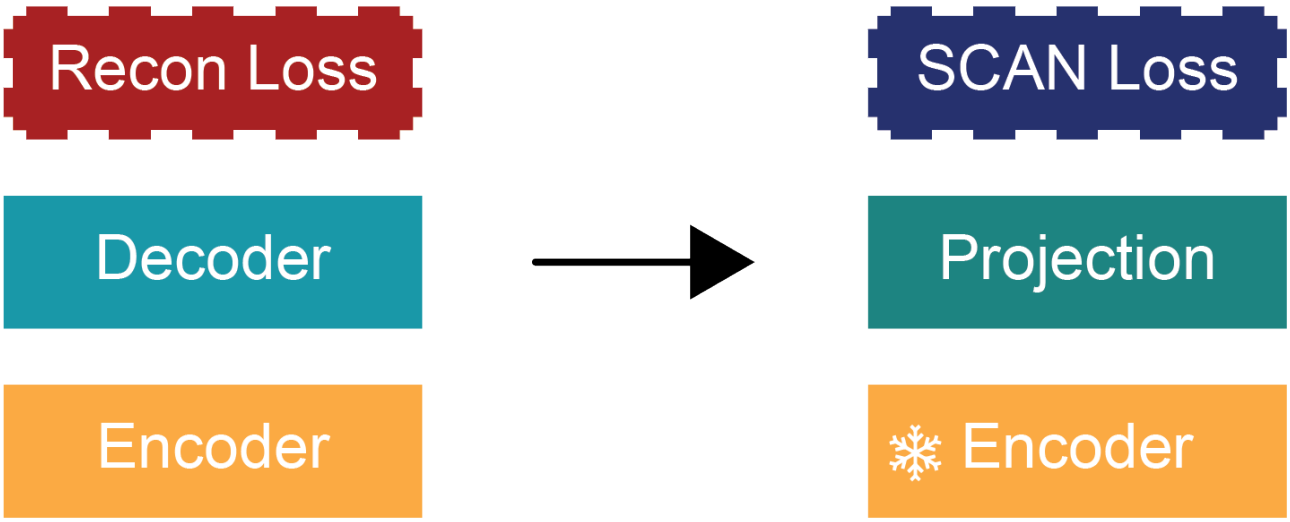

Figure 4.10: The encoder model is first trained in conjunction with a decoder using standard VAE losses. The encoder weights are then frozen and a projection layer is appended. This projection layer is then trained using the SCAN loss.

hypothesized that this could allow the encoder to cluster the data with greater accuracy. The VAE model was then trained and the encoder weights were frozen. A projection layer was then appended to the encoder and the network was trained on the SCAN loss function (Figure 4.10). The balance between the two components of the SCAN loss (entropy and consistency) was then optimized.

4.4.2 Initialization Sensitivity Tests

The next step was to characterize the performance of this trained clustering algorithm. In many previous works, deep unsupervised clustering techniques have been observed to be sensitive to initialization.[110,111] Therefore, initialization was the first component of the performance to be tested. The model's sensitivity to initialization was tested with five random seeds. The sweep over the five random initializations for the total training, VAE loss and then SCAN loss, resulted in an average accuracy of 61.9%±4.2%. As shown in Figure 4.11, the model was still quite sensitive to initialization.

It was hypothesized that the poor performance could be due to a global structure rather than a local. Specifically, while the five nearest neighbors generally belonged the same class, the global structure may not be conducive to clustering which may not be reflected in the average nearest neighbor homogeneity. It was observed that increasing the VAE training time from 30 to 140 epochs was very beneficial. This decreased both the standard deviation in accuracy from 4.2% to 1.3% (Figure 4.11) and increased the mean accuracy from 61.9% to 67.2%.

To attempt to further mitigate noise and improve training stability, we probed a correlation between each of the losses (entropy, consistency loss) and clustering accuracy. If it was observed that one of the losses (whose value does not require knowledge of the ground truth label) could be used as a proxy for clustering accuracy, multiple random seeds could be tested and a high-performing model could be selected, thus improving performance. To investigate a possible correlation, the encoder was trained with five different random seeds (Figure 4.12).

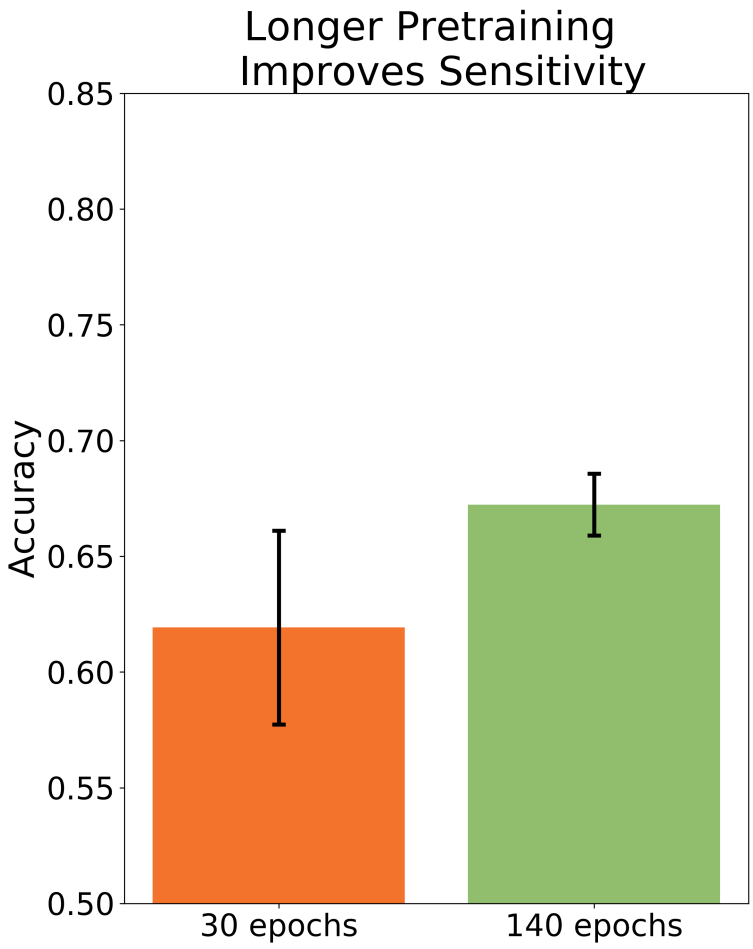

Figure 4.11: Two sweeps over the same number of random seeds are shown.

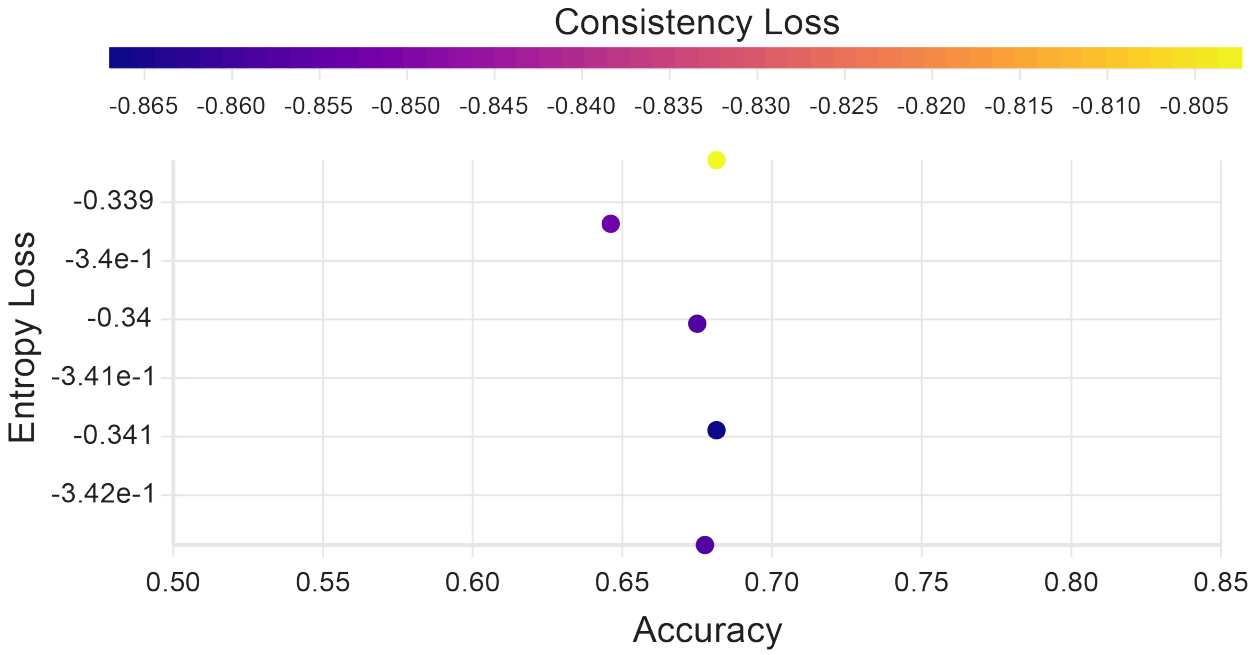

Figure 4.12: Classification accuracy plotted against entropy and consistency losses.

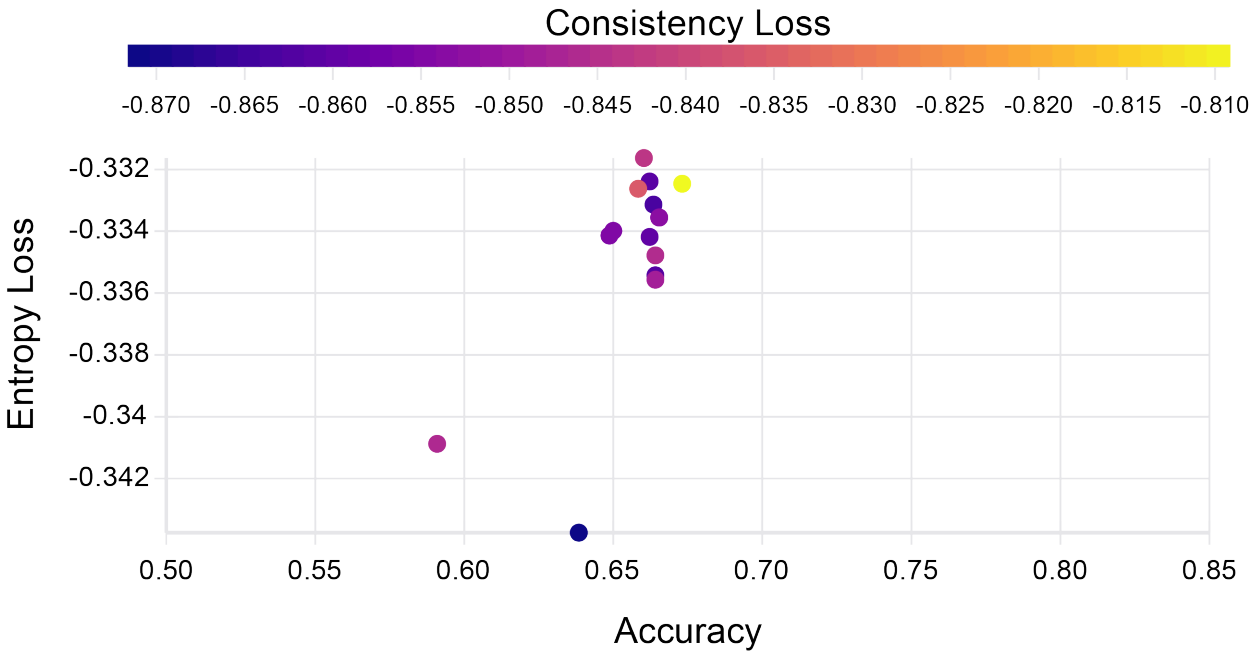

Figure 4.13: Each of the encoder models from Figure 4.12 is used to initialize the clustering model. For a single encoder, three different random seeds were used in an effort to find an optimal clustering.

Although no clear correlation was observed, it was hypothesized that clustering losses should be compared within a given encoder model, not necessarily between them. Next, for each encoder model, 3 different random seeds were used to train on the SCAN loss. None of the loss functions substantially correlated with the accuracy of the clustering, which prevented the selection of a performant model (Figure 4.13).

4.4.3 Augmentations and Selecting Confident Samples

In an effort to improve the model's clustering accuracy and encourage the encoder to extract semantically relevant features, the strategy of augmenting data was considered. This strategy seeks to modify the data while leaving the underlying class of the data easily discernible, forcing the model to rely on higher level features. Since the choice of augmentations is generally task-dependent[112] augmentations were chosen that destroyed low-level information while preserving high-level information, such as the overall order of timesteps. To test the potential of this approach, we used several augmentations during training (Figure 4.14). The four augmentations provided in Figure 4.14 include a randomly placed box of zeros of size 30×30 (style 1), complete subtraction of a given image (style 2), zeroing out 33% of the pixels (style 3), setting 33% of pixels set to one (style 4), and shuffling of a given image with the next two images in the sequence (style 5). Samples were chosen at random to be augmented by any one of the listed styles. If a sample was chosen for augmentation, 10% of the timesteps were augmented with the randomly chosen augmentation style. It was observed that the augmentations actually decreased the performance significantly (Figure 4.4.2).

$$\text{confidence} = |\Phi'_\eta(X^{(i)}) - 0.5| \tag{4.5}$$

After inspecting model performance more closely, it was observed that confidently clustered samples were often accurate (prediction confidence defined in equation 4.5). Although one generally needs to work with the whole dataset, it can be valuable to select representative

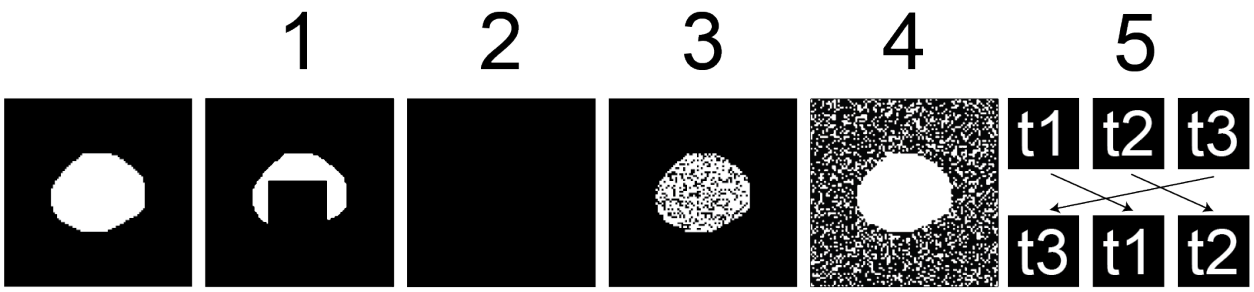

Figure 4.14: Five different augmentation styles are illustrated. The first image on the left shows a sample image before augmentation. Style one shows a randomly placed box of zeros of size 30×30. Style two shows the complete subtraction of a given image. Style three shows 33% of the pixels zeroed out. Style four shows 33% of pixels set to one. Style five shows the shuffling of a given image with the next two images in the sequence.

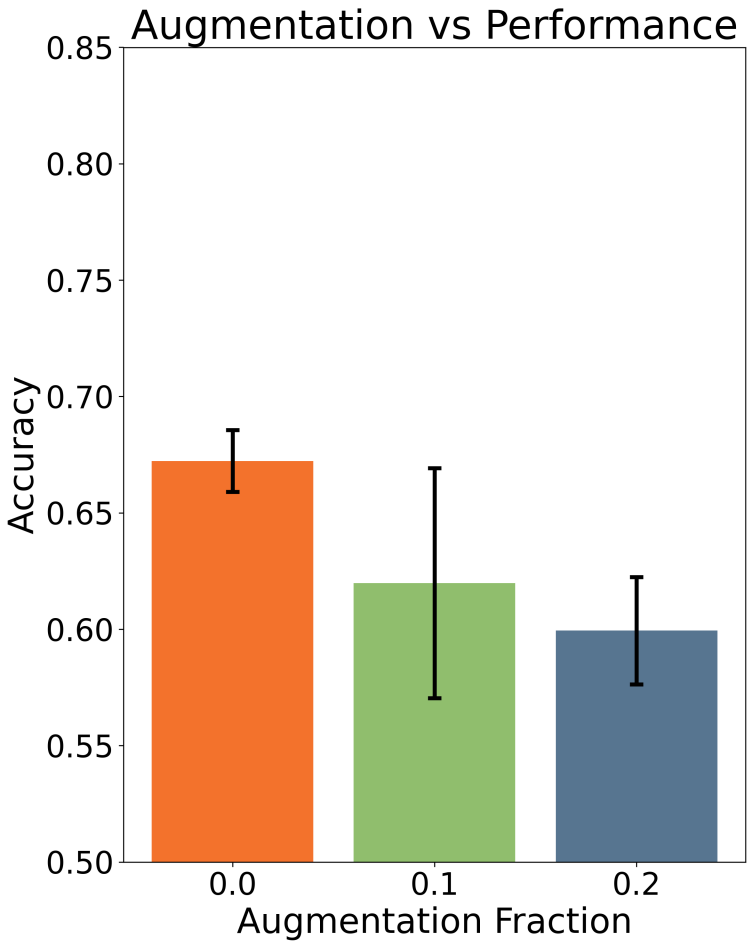

Figure 4.15: The fraction of augmented samples plotted against the classification accuracy. If a given sample is chosen to be augmented, on average 30% of the time steps will be augmented. Each of the five augmentations is selected at random for each timestep.

examples with a high likelihood of being a true positive. For example, in enriching a cell population for downstream processing, a low false-positive rate is the most important metric. Therefore, we sought to identify examples to represent each cluster by selecting only confident samples (> 0.495). The top predictions made up $\sim$10% of all predictions with an even class distribution (Figure 4.16). Setting a threshold for classified samples improved the accuracy to $\sim$ 80% (Figure 4.17). For comparison, the accuracy of a supervised deep learning model trained on binary masks is 91% (Section 3.2.8) and the accuracy is 74.5% (Figure 3.8) for a supervised SVM trained on hand-picked features.

4.5 Conclusions

This work illustrates that deep learning is required for efficient clustering in deformability cytometry, and provides a new technique for unsupervised clustering. Although simple dimensionality reduction techniques work well in a variety of areas, the work here shows that they are insufficient for the compression of complex data obtained from deformability cytometry. In addition, this work highlights the need for extensive VAE training for the purpose of feature learning. The potential of the SCAN loss in optimizing an encoder to classify unknown populations of cells is also explored using a novel unsupervised algorithm, DeepDeform. These improvements resulted in a model with moderate unsupervised clustering accuracy. Furthermore, when the predictions are thresholded, the accuracy of this model increases to 80%. A similar model could find applications in selecting rare, unknown groups of cells for further scrutiny, such as single cell RNA sequencing. Future work will entail the improvement of the interpretability of the DeepDeform model.

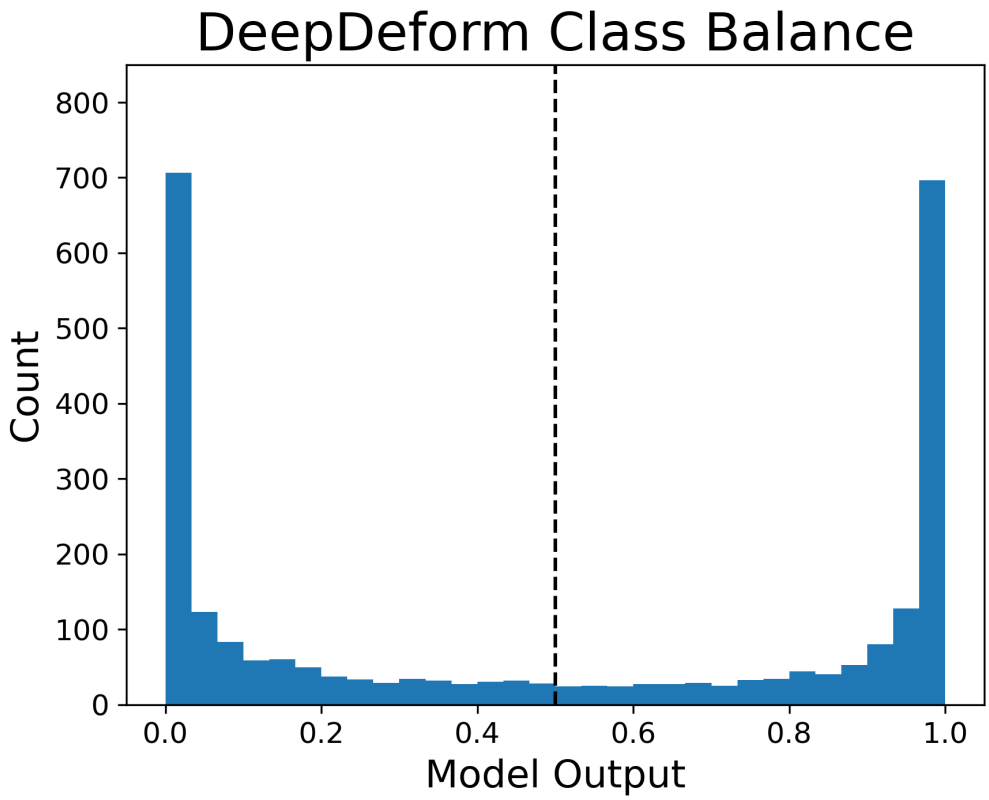

Figure 4.16: Class balance was averaged over five random seeds. A model output of zero indicates a confident prediction of HL60 while a prediction of one indicates a confident prediction of HL60d. A prediction of 0.5 indicates complete uncertainty, which is illustrated with a dotted black line.

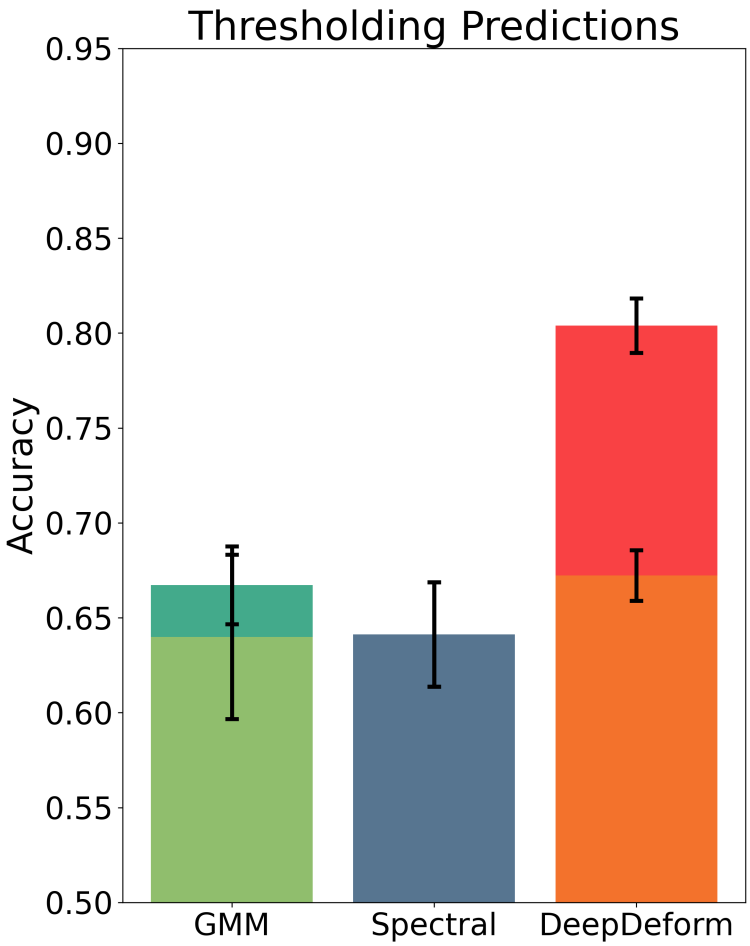

Figure 4.17: Predictions with a confidence over 0.495 are selected for both the GMM and DeepDeform. The dark green bar and red bars reflect the selection of confident samples for the GMM and DeepDeform, respectively.

Bibliography

(1) Deng, L. *IEEE Signal Proc. Mag.* **2012**, *29*, 141–142.

(2) Chapelle, O.; Haffner, P.; Vapnik, V. N. *IEEE Trans. on Neural Netw.* **1999**, *10*, 1055–1064.

(3) Leslie, C.; Eskin, E.; Noble, W. S. In *Biocomputing 2002*; World Scientific: 2001, pp 564–575.

(4) Smith, N.; Gales, M. *Adv. Neural Inf. Process. Syst.* **2001**, *14*.

(5) Boser, B. E.; Guyon, I. M.; Vapnik, V. N. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, Association for Computing Machinery: Pittsburgh, Pennsylvania, USA, 1992, pp 144–152.

(6) Kam, H. T. et al. In *Proceedings of the 3rd international conference on document analysis and recognition*, 1995; Vol. 1416, p 278282.

(7) Breiman, L. *Mach. Learn.* **1996**, *24*, 123–140.

(8) Zhang, Y.; Alder, M.; Togneri, R. In *Proc. - ICASSP IEEE EEE Int. Conf. Acoust. Speech Signal Process.* 1994; Vol. 1, pp I–613.

(9) Rau, A.; Maugis-Rabusseau, C. *Brief. Bioinform.* **2018**, *19*, 425–436.

(10) Liu, Y.; Eyal, E.; Bahar, I. *Bioinform.* **2008**, *24*, 1243–1250.

(11) Zare, H.; Shooshtari, P.; Gupta, A.; Brinkman, R. R. *BMC Bioinform.* **2010**, *11*, 1–16.

(12) LeCun, Y.; Bengio, Y.; Hinton, G. *Nature* **2015**, *521*, 436–444.

(13) McCulloch, W. S.; Pitts, W. *Bull. Math. Biol.* **1943**, *5*, 115–133.

(14) Wasserman, P.; Schwartz, T. *IEEE Expert* **1988**, *3*, 10–15.

(15) Bengio, Y. In *Neural networks: Tricks of the trade*; Springer: 2012, pp 437–478.

(16) Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. *Nature* **1986**, *323*, 533–536.

(17) Cho, K.; Van Merriënboer, B.; Bahdanau, D.; Bengio, Y. *arXiv preprint arXiv:1409.1259* **2014**.

(18) Ravanelli, M.; Brakel, P.; Omologo, M.; Bengio, Y. In *Proc. - ICASSP IEEE Int. Conf. Acoust. Speech Signal Process.* 2017, pp 4880–4884.

(19) Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J., et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

(20) Hochreiter, S.; Schmidhuber, J. *Neural Comput.* **1997**, *9*, 1735–1780.

(21) LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. *Neural Comput.* **1989**, *1*, 541–551.

(22) LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. *Proc. of the IEEE* **1998**, *86*, 2278–2324.

(23) Hubel, D. H.; Wiesel, T. N. *J. Physiol.* **1962**, *160*, 106.

(24) Rawat, W.; Wang, Z. *Neural Comput.* **2017**, *29*, 2352–2449.

(25) Aloysius, N.; Geetha, M. In *Proc. ICCSP IEEE Int. Conf. Commun. Signal Process.* 2017, pp 0588–0592.

(26) Ker, J.; Wang, L.; Rao, J.; Lim, T. *IEEE Access* **2018**, *6*, 9375–9389.

(27) Collins, M.; Duffy, N. *Adv. Neural Inf. Process. Syst.* **2001**, *14*.

(28) Goodfellow, I.; Lee, H.; Le, Q.; Saxe, A.; Ng, A. *Adv. Neural Inf. Process. Syst.* **2009**, *22*.

(29) Kingma, D. P.; Welling, M. *arXiv preprint arXiv:1312.6114* **2013**.

(30) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. *ACS Cent. Sci.* **2018**, *4*, 268–276.

(31) Simidjievski, N.; Bodnar, C.; Tariq, I.; Scherer, P.; Andres Terre, H.; Shams, Z.; Jamnik, M.; Liò, P. *Frontiers Genet.* **2019**, *10*, 1205.

(32) Berenguer, A. D.; Sahli, H.; Joukovsky, B.; Kvasnytsia, M.; Dirks, I.; Alioscha-Perez, M.; Deligiannis, N.; Gonidakis, P.; Sánchez, S. A.; Brahimetaj, R., et al. *arXiv preprint arXiv:2011.11719* **2020**.

(33) Shapley, L. S. A value for n-person games, Contributions to the Theory of Games, 2, 307–317, 1953.

(34) Xie, Y. R.; Castro, D. C.; Bell, S. E.; Rubakhin, S. S.; Sweedler, J. V. *Anal. Chem.* **2020**, *92*, 9338–9347.

(35) Grojean, C.; Paul, A.; Qian, Z.; Strümke, I. *Nature Reviews Physics* **2022**, 1–3.

(36) Coulter, W. H. Means for counting particles suspended in a fluid, U.S. Patent 2656508, 1953.

(37) Anderson, J. L.; Quinn, J. A. *Rev. Sci. Instrum.* **1971**, *42*, 1257–1258.

(38) Yee, J. P.; Mel, H. C. *Biorheology* **1978**, *15*, 321–339.

(39) Yang, L.; Yamamoto, T. *Front. Microbiol.* **2016**, *7*, 1500.

(40) Wanunu, M. *Phys. Life Rev.* **2012**, *9*, 125–158.

(41) Ouldali, H.; Sarthak, K.; Ensslen, T.; Piguet, F.; Manivet, P.; Pelta, J.; Behrends, J. C.; Aksimentiev, A.; Oukhaled, A. *Nat. Biotechnol.* **2020**, *38*, 176–181.

(42) McKinnon, K. M. *Curr. Protoc. Immunol.* **2018**, *120*, 5–1.

(43) Lugli, E.; Roederer, M.; Cossarizza, A. *Cytom. A* **2010**, *77*, 705–713.

(44) Aghaeepour, N.; Finak, G.; Hoos, H.; Mosmann, T. R.; Brinkman, R.; Gottardo, R.; Scheuermann, R. H. *Nat. Methods* **2013**, *10*, 228–238.

(45) Pedreira, C. E.; Costa, E. S.; Lecrevisse, Q.; van Dongen, J. J.; Orfao, A.; Consortium, E., et al. *Trends in Biotechnol.* **2013**, *31*, 415–425.

(46) Doan, M.; Vorobjev, I.; Rees, P.; Filby, A.; Wolkenhauer, O.; Goldfeld, A. E.; Lieberman, J.; Barteneva, N.; Carpenter, A. E.; Hennig, H. *Trends in Biotechnol.* **2018**, *36*, 649–652.

(47) Grimwade, L. F.; Fuller, K. A.; Erber, W. N. *Methods* **2017**, *112*, 39–45.

(48) Weisenhorn, A. L.; Khorsandi, M.; Kasas, S.; Gotzos, V.; Butt, H.-J. *Nanotechnology* **1993**, *4*, 106.

(49) Radmacher, M.; Fritz, M.; Kacher, C. M.; Cleveland, J. P.; Hansma, P. K. *Biophys. J.* **1996**.

(50) Dai, J.; Sheetz, M. P. *Biophys. J.* **1995**, *68*, 988–996.

(51) Evans, E. *Biophys. J.* **1973**, *13*, 941–954.

(52) Otto, O.; Rosendahl, P.; Mietke, A.; Golfier, S.; Herold, C.; Klaue, D.; Girardo, S.; Pagliara, S.; Ekpenyong, A.; Jacobi, A., et al. *Nat. Methods* **2015**, *12*, 199–202.

(53) Asghar, W.; Wan, Y.; Ilyas, A.; Bachoo, R.; Kim, Y.-t.; Iqbal, S. M. *Lab Chip* **2012**, *12*, 2345–2352.

(54) Kim, J.; Han, S.; Lei, A.; Miyano, M.; Bloom, J.; Srivastava, V.; Stampfer, M. R.; Gartner, Z. J.; LaBarge, M. A.; Sohn, L. L. *Microsyst. & Nanoeng.* **2018**, *4*, 1–12.

(55) Byun, S.; Son, S.; Amodei, D.; Cermak, N.; Shaw, J.; Kang, J. H.; Hecht, V. C.; Winslow, M. M.; Jacks, T.; Mallick, P., et al. *Proc. Natl. Acad. Sci. U.S.A.* **2013**, *110*, 7580–7585.

(56) Zhou, Y.; Yang, D.; Zhou, Y.; Khoo, B. L.; Han, J.; Ai, Y. *Anal. Chem.* **2018**, *90*, 912–919.

(57) Fregin, B.; Czerwinski, F.; Biedenweg, D.; Girardo, S.; Gross, S.; Aurich, K.; Otto, O. *Nat. Commun.* **2019**, *10*, 1–11.

(58) Lin, J.; Kim, D.; Henry, T. T.; Tseng, P.; Peng, L.; Dhar, M.; Karumbayaram, S.; Di Carlo, D. *Microsyst. & Nanoeng.* **2017**, *3*, 1–7.

(59) Gossett, D. R.; Henry, T.; Lee, S. A.; Ying, Y.; Lindgren, A. G.; Yang, O. O.; Rao, J.; Clark, A. T.; Di Carlo, D. *Proc. Natl. Acad. Sci. U.S.A.* **2012**, *109*, 7630–7635.

(60) Urbanska, M.; Muñoz, H. E.; Bagnall, J. S.; Otto, O.; Manalis, S. R.; Di Carlo, D.; Guck, J. *Nat. Methods* **2020**, *17*, 587–593.

(61) Ahmmed, S. M.; Bithi, S. S.; Pore, A. A.; Mubtasim, N.; Schuster, C.; Gollahon, L. S.; Vanapalli, S. A. *APL Bioeng.* **2018**, *2*, 032002.

(62) Pires, R. H.; Shree, N.; Manu, E.; Guzniczak, E.; Otto, O. *Philos. Trans. R. Soc. B* **2019**, *374*, 20190081.

(63) Kobayashi, H.; Lei, C.; Wu, Y.; Mao, A.; Jiang, Y.; Guo, B.; Ozeki, Y.; Goda, K. *Sci. Rep.* **2017**, *7*, 1–9.

(64) Masaeli, M.; Gupta, D.; O'Byrne, S.; Tse, H. T.; Gossett, D. R.; Tseng, P.; Utada, A. S.; Jung, H.-J.; Young, S.; Clark, A. T., et al. *Sci. Rep.* **2016**, *6*, 1–11.

(65) Guzniczak, E.; Zadeh, M. M.; Dempsey, F.; Jimenez, M.; Bock, H.; Whyte, G.; Willoughby, N.; Bridle, H. *Sci. Rep.* **2017**, *7*, 1–11.

(66) Guck, J.; Schinkinger, S.; Lincoln, B.; Wottawah, F.; Ebert, S.; Romeyke, M.; Lenz, D.; Erickson, H. M.; Ananthakrishnan, R.; Mitchell, D., et al. *Biophys. J.* **2005**, *88*, 3689–3698.

(67) Mosier, A. P.; Kaloyeros, A. E.; Cady, N. C. *J. Microbiol. Methods* **2012**, *91*, 198–204.

(68) Kubánková, M.; Hohberger, B.; Hoffmanns, J.; Fürst, J.; Hermann, M.; Guck, J.; Krater, M. *bioRxiv* **2021**.

(69) Hodgson, A. C.; Verstreken, C. M.; Fisher, C. L.; Keyser, U. F.; Pagliara, S.; Chalut, K. J. *Lab Chip* **2017**, *17*, 805–813.

(70) Chen, J.; Zheng, Y.; Tan, Q.; Shojaei-Baghini, E.; Zhang, Y. L.; Li, J.; Prasad, P.; You, L.; Wu, X. Y.; Sun, Y. *Lab Chip* **2011**, *11*, 3174–3181.

(71) Hou, H. W.; Li, Q.; Lee, G.; Kumar, A.; Ong, C.; Lim, C. T. *Biomed. Microdevices* **2009**, *11*, 557–564.

(72) Yang, X.; Chen, Z.; Miao, J.; Cui, L.; Guan, W. *Biosens.* **2017**, *98*, 408–414.

(73) Adamo, A.; Sharei, A.; Adamo, L.; Lee, B.; Mao, S.; Jensen, K. F. *Anal. Chem.* **2012**, *84*, 6438–6443.

(74) Ren, X.; Ghassemi, P.; Babahosseini, H.; Strobl, J. S.; Agah, M. *ACS Sens.* **2017**, *2*, 290–299.

(75) Zheng, Y.; Nguyen, J.; Wang, C.; Sun, Y. *Lab Chip* **2013**, *13*, 3275–3283.

(76) Tsai, C.-H. D.; Sakuma, S.; Arai, F.; Kaneko, M. *IEEE. Trans. Biomed. Eng.* **2014**, *61*, 1187–1195.

(77) Nyberg, K. D.; Bruce, S. L.; Nguyen, A. V.; Chan, C. K.; Gill, N. K.; Kim, T.-H.; Sloan, E. K.; Rowat, A. C. *Integr. Biol.* **2018**, *10*, 218–231.

(78) Nyberg, K. D.; Hu, K. H.; Kleinman, S. H.; Khismatullin, D. B.; Butte, M. J.; Rowat, A. C. *Biophys. J.* **2017**, *113*, 1574–1584.

(79) Mietke, A.; Otto, O.; Girardo, S.; Rosendahl, P.; Taubenberger, A.; Golfier, S.; Ulbricht, E.; Aland, S.; Guck, J.; Fischer-Friedrich, E. *Biophys. J.* **2015**, *109*, 2023–2036.

(80) Ge, Y.; Rosendahl, P.; Durán, C.; Töpfner, N.; Ciucci, S.; Guck, J.; Cannistraci, C. V. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2019**.

(81) Riordon, J.; Sovilj, D.; Sanner, S.; Sinton, D.; Young, E. W. *Trends in Biotechnol.* **2019**, *37*, 310–324.

(82) Luo, S.; Shi, Y.; Chin, L. K.; Hutchinson, P. E.; Zhang, Y.; Chierchia, G.; Talbot, H.; Jiang, X.; Bourouina, T.; Liu, A.-Q. *Adv. Intell. Syst.* **2021**, *3*, 2100073.

(83) Hinkle, P.; Westerhof, T. M.; Qiu, Y.; Mallin, D. J.; Wallace, M. L.; Nelson, E. L.; Taborek, P.; Siwy, Z. S. *Sci. Rep.* **2017**, *7*, 1–14.

(84) Yang, D.; Zhou, Y.; Zhou, Y.; Han, J.; Ai, Y. *Biosens.* **2019**, *133*, 16–23.

(85) Choi, G.; Tang, Z.; Guan, W. *Nanotechnol. Precis. Eng.* **2021**, *4*, 045002.

(86) Nawaz, A. A.; Urbanska, M.; Herbig, M.; Nötzel, M.; Kräter, M.; Rosendahl, P.; Herold, C.; Toepfner, N.; Kubánková, M.; Goswami, R., et al. *Nat. Methods* **2020**, *17*, 595–599.

(87) Lundberg, S. M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J. M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; Lee, S.-I. *Nat. Mach. Intell.* **2020**, *2*, 2522–5839.

(88) Golfier, S.; Rosendahl, P.; Mietke, A.; Herbig, M.; Guck, J.; Otto, O. *Cytoskeleton* **2017**, *74*, 283–296.

(89) Piergiovanni, M.; Galli, V.; Holzner, G.; Stavrakis, S.; DeMello, A.; Dubini, G. *Lab Chip* **2020**, *20*, 2539–2548.

(90) Liu, Y.; Wang, K.; Sun, X.; Chen, D.; Wang, J.; Chen, J. *Cytometry A* **2022**, *101*, 434–447.

(91) Abdulla, W. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow, `https://github.com/matterport/Mask_RCNN`, 2017.

(92) Cooper, J. A. *J. Cell Biol.* **1987**, *105*, 1473–1478.

(93) Bashashati, A.; Lo, K.; Gottardo, R.; Gascoyne, R. D.; Weng, A.; Brinkman, R. In *Annu. Int. Conf. IEEE Eng.* 2009, pp 4945–4948.

(94) Scholtens, T. M.; Schreuder, F.; Ligthart, S. T.; Swennenhuis, J. F.; Greve, J.; Terstappen, L. W. *Cytom. A.* **2012**, *81*, 138–148.

(95) Valkonen, M.; Kartasalo, K.; Liimatainen, K.; Nykter, M.; Latonen, L.; Ruusuvuori, P. *Cytom. A.* **2017**, *91*, 555–565.

(96) Duetz, C.; Van Gassen, S.; Westers, T. M.; van Spronsen, M. F.; Bachas, C.; Saeys, Y.; van de Loosdrecht, A. A. *Cytom. A* **2021**.

(97) Lundberg, S. M.; Lee, S.-I. In *Adv. Neural Inf. Process. Syst.* Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: 2017, pp 4765–4774.

(98) Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. *Preprint at https://arxiv.org/abs/1412.3555* **2014**.

(99) Hu, M.-K. *IEEE Trans. Inf. Theory* **1962**, *8*, 179–187.

(100) Keller, L.; Pantel, K. *Nat. Rev. Cancer* **2019**, *19*, 553–567.

(101) Batlle, E.; Clevers, H. *Nat. Med.* **2017**, *23*, 1124.

(102) Combs, C.; Seith, D. D.; Bovyn, M. J.; Gross, S. P.; Xie, X.; Siwy, Z. S. *Biomicrofluidics* **2022**, *16*, 014104.

(103) Kopf, A.; Fortuin, V.; Somnath, V. R.; Claassen, M. *PLoS Comput. Biol.* **2021**, *17*, e1009086.

(104) Ternes, L.; Dane, M.; Gross, S.; Labrie, M.; Mills, G.; Gray, J.; Heiser, L.; Chang, Y. H. *Commun. Biol.* **2022**, *5*, 1–10.

(105) Caron, M.; Bojanowski, P.; Joulin, A.; Douze, M. In *ECCV*, 2018, pp 132–149.

(106) Liu, Y.; Wang, K.; Sun, X.; Chen, D.; Wang, J.; Chen, J. *Microfluid. Nanofluid.* **2020**, *24*, 1–11.

(107) Biewald, L. Experiment Tracking with Weights and Biases, Software available from wandb.com, 2020.

(108) Van Gansbeke, W.; Vandenhende, S.; Georgoulis, S.; Proesmans, M.; Van Gool, L. In *ECCV*, 2020, pp 268–285.

(109) Liu, Z.; Song, Y.-q.; Xie, C.-h.; Tang, Z. *Signal Image Video Process.* **2016**, *10*, 359–368.

(110) Chang, J.; Wang, L.; Meng, G.; Xiang, S.; Pan, C. In *ICCV*, 2017.

(111) Xie, J.; Girshick, R.; Farhadi, A. In *ICML*, 2016, pp 478–487.

(112) Tian, Y.; Sun, C.; Poole, B.; Krishnan, D.; Schmid, C.; Isola, P. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6827–6839.