Rheological characterization of non-Newtonian fluids using Deep CNN

Rajorshi Paul
Department of Mechanical Engineering
Stanford University
rajorshi@stanford.edu

Jeevesh Konuru
Stanford Center for Professional Development
Stanford University
jeevesh@stanford.edu

Abstract

In this work, we have developed a Convolutional Neural Network (CNN) model to classify viscoelastic fluids based on droplet deformation at a microfluidic constriction using sequential data. Traditionally, recurrent neural network models are used for classification of sequential data. Here, we show that we can use a simple CNN model to achieve similar accuracy. Our CNN model was able to achieve an accuracy 60% on our dataset which was slightly higher than the accuracy obtained from recurrent neural networks (Simple RNN, GRU and LSTM). The saliency map obtained from the CNN model shows that the model learns to identify the charcteristics of deformation to classify the rheological properties of droplets.

1. Introduction

Biological cells are known to respond to mechanical stresses and can modulate their mechanical properties to perform critical biological functions. Mechanical properties of cells are challenging to quantify because of the inherent inhomogeneities within the cell. Recently, deformability cytometry has emerged as an efficient microfluidic tool to characterize rheological properties of cells [2] [4] [3]. The deformability cytometry allows us to process a large number of cells (of the order of thousands) in a short time. However, because of the absence of a standard mechanical model for cells, the measurements are not very accurate and can vary widely between different models. Therefore, deformability cytometry exists primarily as a tool for qualitative comparison between similar cell types. Recent advances in computer vision based machine learning models allow us to implement machine learning as a tool for implicitly determining the underlying mechanical model for cells. Recently, Combs et al. were able to train a hybrid CNN-GRU model to classify two different drug treated HL60 cell samples with an accuracy of 90% [1]. However, their model is specific to their chosen cell types and cannot be extended to a general frame work consisting of multiple cell types. To our knowledge there does not exist a machine learning model which is capable of interpreting the deformation characteristics of cells and predicting accurately their viscoelastic properties.

The biggest challenge when it comes to working with live biological cells is the limited availability of data. Fluid droplets, however, offer a suitable alternative to droplets. Cells can be modeled as complex droplets and their deformation characteristics are similar to that of droplets. In this work, we used droplets in lieu of live biological cells. The characteristics of deformation and subsequent shape recovery of droplets are correlated to their rheological properties. For example, a non-Newtonian fluid droplet having a high storage modulus would tend to recover their shape faster than a fluid droplet having a high loss modulus. These differences are manifested in the way the droplets deform under hydrodynamic stresses. Therefore, a deep learning model capable of classifying fluids based on their rheological properties can be extended by transfer learning to learn rheological properties of biological cells. This is the motivation behind this work.

In this work, we use deformation characteristics of droplets of varying rheological properties to to classify non-Newtonian fluids. We developed a simple CNN model and trained it on sequential data of droplet deformation. We compared the performance of the CNN model with recurrent neural network models which are designed to handle sequential information. Specifically, we used a simple RNN model, a GRU and an LSTM model.

2. Dataset and features

2.1. Data collection

To characterize the deformation of droplets in a constant shear rate microfluidic channel, we designed a hyperbolic constriction channel with a channel width of 100 μm at the constriction (Fig. 1). The hyperbolic constriction channel was coupled with a flow focusing droplet generator upstream of the constriction. To control the flow rate of the generated droplets, a sheath flow channel was added to the

device design. Mineral oil (M5904-500ML, Sigma Aldrich, density 840 Kg/m^3, viscosity 0.024 Pa-s) with 2% Span 80 was used as the continuous phase. Three dispersed phases were used for droplet generation. These include de-ionized (DI) water, 0.1% methyl cellulose solution in DI water, and 0.25% methyl cellulose solution in DI water. Droplets were generated at a net flow rate of 1 mL/h.Droplet deformation at the hyperbolic constriction was imaged using a Vision Research Phantom v7.3 high speed camera at a frame rate of 200 fps for the DI water droplets and 1000 fps for the methyl cellulose droplets. All experiments were imaged at a resolution of 2000px x 252px under a 10x objective. A sample raw data from the experiment is shown in Fig. 1A. The image contains multiple droplets entering a microfluidic constriction junction where they deform and subsequently relax.

2.2. Data processing

The raw data collected from the droplet deformation experiments was processed using the Image Processing Toolbox in MATLAB R2019a. A droplet detector code was developed to detect droplets passing through a predefined checkpoint in each video frame. We tracked a droplet at different points on its trajectory and overlayed them on a frame not containing any droplets. Using this process, we generated frames containing a single droplet at four time instants. The data was augmented by offsetting each image by a frame. Finally, the resolution of each image was reduced to generate a final image of resolution 125px x 25px. We also performed background subtraction to eliminate effects of non-uniform lighting. For the LSTM model, the training examples consisted of images of droplets at four distinct time instances. Therefore, for the LSTM model, no overlaying was performed.

3. Methods

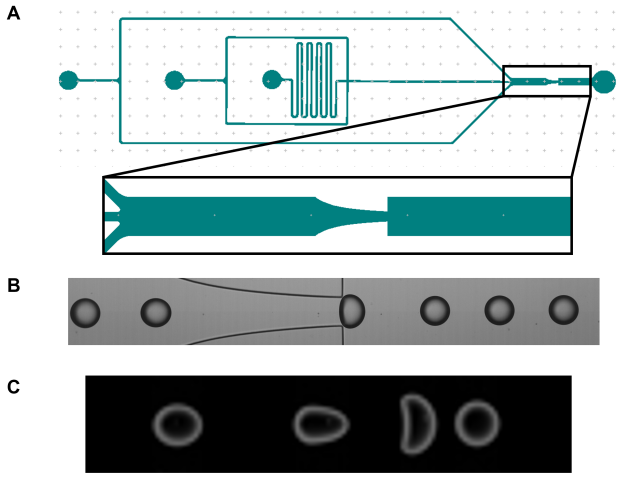
Two different approaches were considered for tackling this problem. The first approach was to use a ConvNet and the second and more traditional approach to the problem was to use LSTM. Previous paper have has success with LSTM and GRU since the data is sequential. As a control for our data-set we constructed an LSTM model using the previous approach as a guide. The second approach was to construct a data set that layered snapshots of the droplet at certain key points along its trajectory through the constriction.

3.1. ConvNet Approach

3.1.1 Pre-processing

The raw data was processed to show the trajectory of a single droplet at specific points of the trajectory (Fig. 1C). This

Figure 1. Data collection and processing. A. Schematic of microfluidic device with inset showing the hyperbolic constriction channel. B. Raw image showing multiple droplets inside the hyperbolic channel. C. Processed data showing the trajectory of a single droplet inside the hyperbolic channel

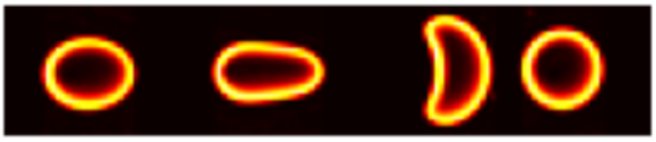


was done to be able to train a deep CNN model on sequential image data. The input image data has a resolution of 125px x 25px.

3.1.2 Experiments

Several experiments were conducted in order to tune the model. Saliency maps were very useful in verifying that the model was correctly identifying the features that we expected it to use for classification. Whenever a significant change to the model was made the saliency map could be used to ensure that our heuristics on the important features of the model were being captured for classification. Across several models we found that the saliency maps consistently picked up on the circumferential features of the droplets

Figure 2. Saliency map of one droplet example



Early on a learning curve was plotted as an exercise to validate the model and training data at a high level. The training error was plotted with increasing amounts of data starting with a single data-point. The training error eventually increased indicating that the model was capable of learning.

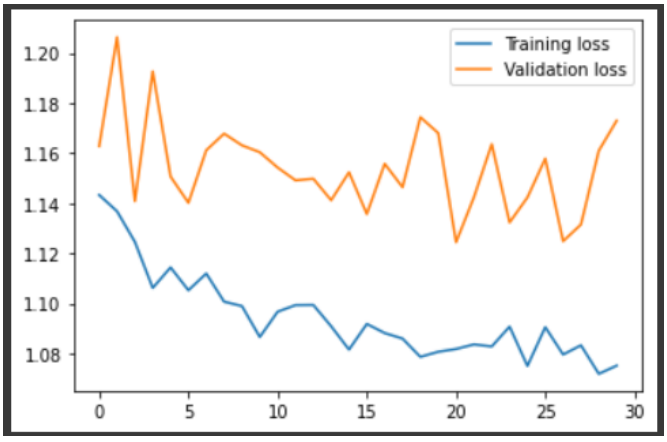
Train and validation accuracy was also important metric

through the experimental process. For example, early on validation losses were far higher than training losses suggesting that we were over-fitting the data. This was resolved by introducing a dropout between each of the convolutional layers until the overfititng was resolved. This occurred around a dropout of 0.3.

Hyper-parameter tuning was performed by looping over a range of training epochs, learning rates, hidden layers, and types of optimizer. Major model changes were manually adjusted based on heuristics from the model. A random parameter search was also attempted but was determined to not be required given the low training time for this model when using smaller data-sets.

Throughout the parameter tuning process train and validation losses were plotted over training epochs. In several cases these plots helped identify issues with the training process. For example at one point validation loss was much higher than training losses. This was another indication of over-fitting. The model was able to memorize features of the droplets but failed to generalize them to perform on the validation sets. This was again resolved by tuning dropout and by increasing the number of images that were used. Ultimately we also had to decrease the number of layers to resolve the issue.

Figure 3. Training loss decreasing faster than validation loss indicating over-fitting



These plots were also useful for epoch and learning rate tuning. For example some plots showed losses that swung at very large magnitudes and manually dropping the learning rate helped decrease these swings. In other cases it it was possible to project that the losses were still decreasing when the training ended. In these cases the number of epochs were increased until the losses reached an asymptote.

3.1.3 Architecture

Once all of the hyper parameters were tuned and experiments completed the resulting convolutional network had

Figure 4

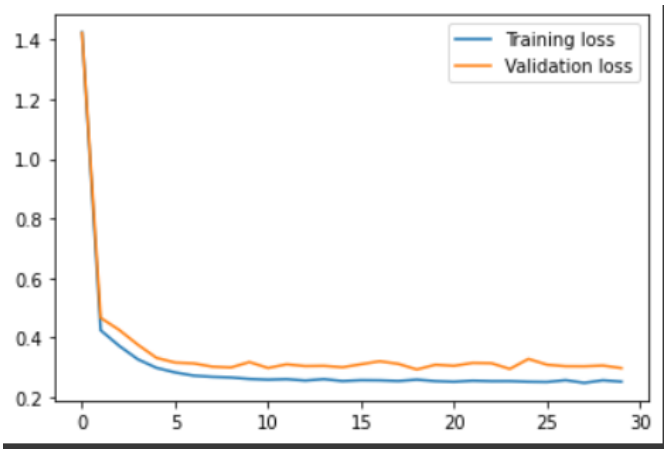


Figure 5. Losses asymptoted when epochs were increased

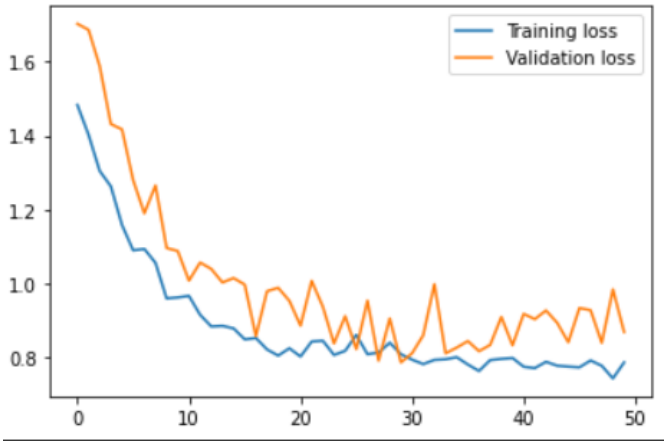
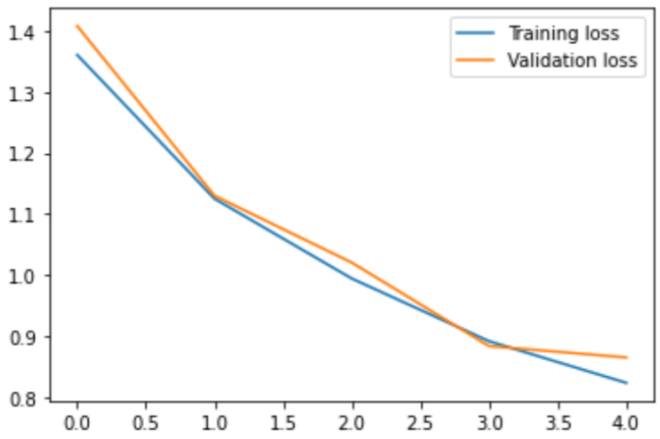


Figure 6. Losses were still decreasing indicating that more cycles are required



4 convolutional layers followed by a fully connected layer. ReLU, Batchnorm, and pooling were used between the lay-

ers. An Adam optimizer with a learning rate 1e-3 was selected. Early we ran into both the vanishing gradients issue. Adding ReLU activations helped with this since ReLU has a zero gradient. Later in the tuning process however we observed large changes were between batch/iterations that indicated exploding gradients. Two changes were made in order to combat this issue. We also introduced a Xavier weight initializer. This helped since it decreased the sizes of the initial weights that we used. The second thing we did was to add L2 Regularization. This succeeded in establishing smaller weight updates.

The model produced the best results when trained for 30 epochs. Increasing hidden layer sizes were used to maintain increasing model complexity. Generally speaking, larger layers were found to produce better results as the model continues to pick up on smaller features in the form factor of the droplets.

3.1.4 Results

We achieved a validation and train accuracy of 0.82 and 0.83 respectively when training on class 1 and class 2 only. As expected, accuracy decreased to around 0.60 validation when a third class was introduced. It did however seem like we performed less consistently well when the third class was introduced. This is likely due to more outliers in that data set.

3.2. Recurrent neural networks

3.2.1 Pre-processing

Recurrent neural networks require a sequence of data as inputs to make predictions. For training recurrent neural networks, we isolated 26px x 25px images of droplets at four distinct time points from the raw data. The image data was flattened to generate a training set of shape (N, 4, 650) where N is the number of training examples. The dataset consisted of 12,751 image sequences which was consequently split into training, validation and test sets in the ratio of 8:1:1.

3.2.2 Architecture

We tried three different recurrent neural network architectures to compare the performance of our aforementioned CNN network. These architectures include a simple recurrent neural network (Simple RNN), a gated recurrent unit network (GRU) and a long short term memory network (LSTM). The architecture used here is as follows:

Input $\rightarrow$ BatchNorm $\rightarrow$ Rec Block (256) $\rightarrow$ Rec Block (256)

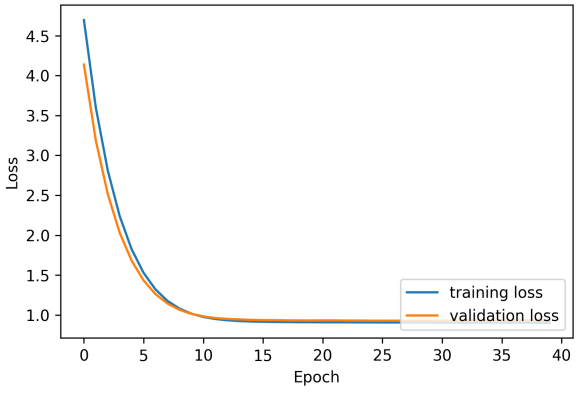
$\rightarrow$ Fully Connected (24) $\rightarrow$ Fully Connected (3) $\rightarrow$ Output

The Rec Block was a unit of the Simple RNN/GRU/LSTM layer. ReLU activation was used in between the fully connected layers. The architecture used here was similar to that used by Combs et al.

3.2.3 Experiments and Results

Hyperparameter tuning was performed to reduce overfitting and increase test accuracy. Figs. 6 and 7 show the variation of loss and accuracy during training and validation. The loss curves nearly coincide indicating that the model is not overfitting the training examples. Hence, the model seems to be learning the features of training examples instead of memorizing the images themselves. However, the accuracy of prediction is quite low. Although the model performs better than random guessing which would allow and accuracy of 33%, the test accuracy in the GRU model is only 47.4%. Therefore, there seems to be a high bias in the recurrent neural network models, similar to what we observed in the CNN model. Table 1 summarizes the accuracy observed in the three recurrent neural networks.

Figure 7. Training and validation losses in the LSTM model



Model	Train	Validation	Test
Simple RNN	52.53%	46.82%	45.53%
GRU	49.01%	46.20%	47.41%
LSTM	51.24%	45.73%	46.16%

Table 1. Comparison between different recurrent neural network architectures

4. Conclusion and future work

The CNN model used in this work was found to have a better test accuracy than the recurrent neural networks tested. This indicates that it is possible to use a simple CNN architecture to process sequential data. However, a simple