Predicting Microfluidic Droplet Diameters in Glass Capillary Devices Using Machine Learning

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Serena Holte

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Dr. Laura Adams

June, 2023

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school. Firstly, I would like to thank Dr. Laura Adams for continuous guidance and support throughout this research project and my time at UMD. I am also grateful to the remainder of my committee, Dr. Sam Kelly and Dr. Eleazar Leal, for their time and effort in reviewing this report and providing constructive feedback. Thank you to Dheeraj Sapkota for collaborating on the experimental data and for fruitful discussions. Thank you also to Xeen Meighan, John Gallagher, and Matt Laine for discussions and experimental help. I am grateful to the University of Minnesota Duluth and for the Mylan Radulovich graduate fellowship that financially supported me throughout my project and time at UMD.

Abstract

I have successfully generated a graphic user interface that predicts microfluidic droplet diameters from a neural network. The neural network inputs are fluid properties and geometries of 3D glass capillary devices. For water-in-oil single emulsions, the mean-squared error at the end of 100 epochs for training and validation converged to 7.2% and 7.4%, respectively. The deep machine learning model provides an alternative method of predicting droplet size without the need for rigorous theory. Moreover, the model can be altered to predict other microfluidic parameters or properties and could be extended to other fluids as well.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A very brief outline of the chapters in this thesis is presented in this introductory Chapter. This project involves using machine learning techniques to better understand some of the physics behind drops made with microfluidic devices. The thesis will introduce the theory of droplet-based microfluidics, as well as the relevant machine learning techniques. The methods with which this project was accomplished will follow, before the results are presented. The thesis will discuss the interpretation of the results and provide a conclusion and future prospects of the project. All the Python code that was used to obtain the machine learning and linear regression results is presented in the appendices. The outline of the thesis follows.

- Chapter 2 briefly presents the theory behind the subjects presented in this thesis, as well as the project goal.
- In Chapter 3 the methods of the experimental and machine learning portion are outlined.
- Chapter 4 contains the results and discussion of the project.
- Chapter 5 presents a conclusion of the analyses presented in the thesis.
- Chapter 6 introduces future prospects of the project.

Chapter 2

Theory of Microfluidics and Machine Learning

In section 2.1, a brief overview of the theory of droplet-based microfluidics and relevant physics is discussed. In section 2.2, neural networks, which are a branch of machine learning, are introduced. In section 2.3, the goal of the project is presented.

2.1 A Brief Theory of Microfluidics

Microfluidics is the study or manipulation of fluids at a micro-scale that involves simple or complex, single phase or multiphasic flows circulating through natural or artificial micro systems. In microfluidics, at least one dimension of the system is typically less than 500 micrometers. Naturally, microfluidic systems can be found throughout the human body in capillaries and arterioles or in trees' capillaries where water and nutrients are transferred.

There are numerous applications of microfluidics. For example, microreactors are small-scale controlled chemical reactions, allowing for minimal safety concerns and a low required yield of reactants [1]. The pharmaceutical industry uses microfluidics for drug discovery and development, and for the controlled release of encapsulated materials like drugs, dyes, and enzymes [2]. Similarly, the medical field can use microfluidics

by generating micro-bubbles (2-5 microns in diameter) to enhance ultrasound imaging and for passing through lung capillaries [3]. Figure 2.1 includes images from Dr. Adams' work. The left image shows a multi-component emulsion, which may resemble a microreactor with several reactants contained to a single droplet [4]. The right image shows encapsulated air bubbles, similar to how an encapsulated micro-bubble may be generated [5]. The applications for microfluidics extends far beyond these examples and new uses are emerging steadily.

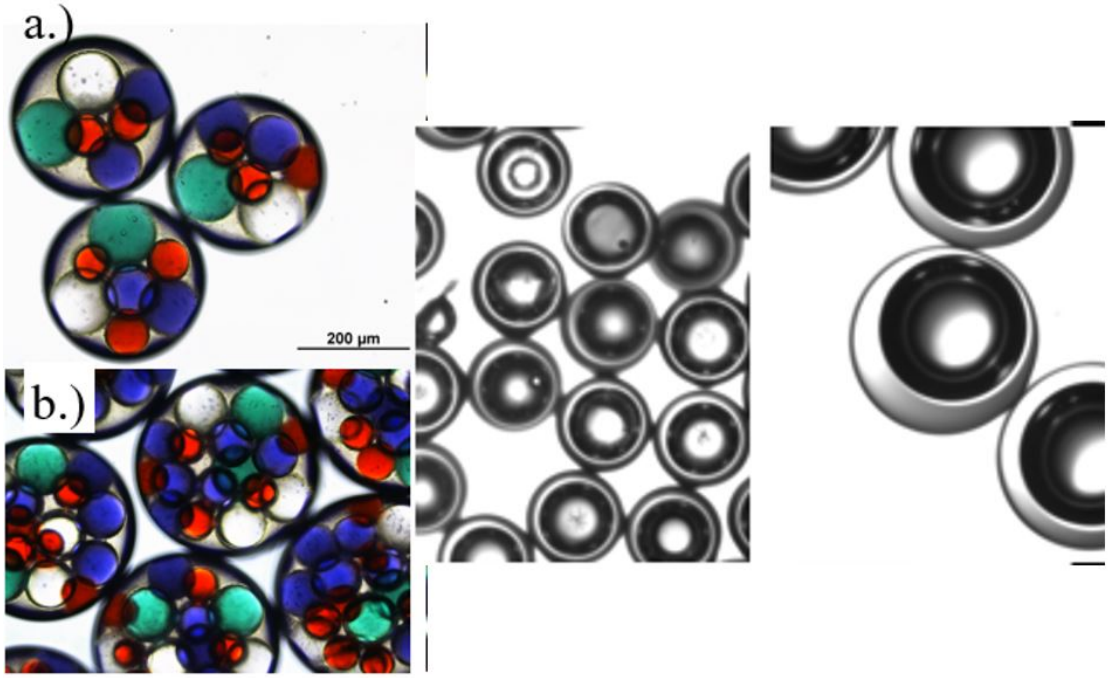


Figure 2.1: Images of multi-component emulsions (left) and encapsulated air bubbles inside an emulsion (right) from Dr. Laura Adams.

Droplet-based microfluidics are discrete volumes of two or more immiscible fluids, like water and oil. The volumes of fluids for emulsions generated with microfluidics are on the order of nano- to pico-liters. For certain applications (typically commercial), it is important that these droplets are uniform in size and shape. To achieve uniform drops, the fluids must remain in the dripping regime. The dripping regime is synonymous with the monodisperse regime, meaning droplets of uniform size. Polydispersed

drops, meaning the drops vary in size, are formed in the jetting regime.

Microfluidics occurs in laminar flow, which describes smooth flow, where streamlines do not cross (no eddies in the fluid) [6]. Laminar flow is constrained by the Reynolds number: the ratio of inertial forces ($\rho v^2 d^2$) to viscous forces ($3\pi\mu v d$) on the fluid. Equation 2.1 is the Reynolds number and demonstrates the relationship between the forces.

$$Re = \frac{\rho l v}{\mu} \tag{2.1}$$

Reynolds number relates the fluid density ρ, the characteristic length l, the fluid velocity v, and the fluid viscosity μ. The Reynolds number is a dimensionless parameter derived from the Navier-Stokes equation. The Navier-Stokes equation governs fluid dynamics and describes macro- and micro-scale fluid dynamics (down to 10-100 nm). Water and oil are the fluids used in this project, so incompressible flow can be assumed. Equation 2.2 describes the motion of the fluid by relating the forces per volume (f=$\frac{F}{V}$) that contribute to the fluid's acceleration with the change in momentum [7].

$$f_a = f_{pressure} + f_{friction} + f_{volume}$$
$$\rho[\frac{\partial v}{\partial t} + (v \cdot \nabla)v] = -\nabla p + \mu\nabla^2 v + \rho g \tag{2.2}$$

where ρ is the fluid density, v is the fluid velocity, p is the pressure, μ is the viscosity, and g is the gravitational acceleration on earth. The left side of equation 2.2 describes the change in momentum due to the change in velocity. The right side describes the forces acting on the fluid: pressure, friction, and volume forces. Volume forces include centrifugal, gravity, and electrostatic forces.

Starting with equation 2.2, and rewriting the Navier-Stokes equation using dimensional analysis techniques and non-dimensional change of variables where:

$$u^* = \frac{u}{v}$$

$$t^* = \frac{t}{T}$$

$$x^* = \frac{x}{L}$$

These are non-dimensional due to the scaling: $v{=}\frac{L}{T}$ is the velocity scale, T is the time scale, and L is the length scale.

Then Equation 2.2 now becomes:

$$\rho[\frac{\partial u^*}{\partial t^*} + (u^* \cdot \nabla^*)u^*] = -\frac{1}{\rho v^2}\nabla^* p + \frac{\mu}{\rho v L}\nabla^{*2}u^* + \frac{L}{v^2}g \tag{2.3}$$

The friction force term produces a dimensionless coefficient on the $\nabla^{*\ 2}$ u*:

$$\frac{\mu}{\rho v L} = \frac{1}{Re}$$

In microfluidics, gravitational effects are negligible (last term on the right side of equation 2.3) because pressure effects due to gravity are much less than atmospheric pressure due to the mirco-scale characteristic length of the channel. Additionally, viscous effects dominate (second term on right side of equation 2.3). Thus the equation can be simplified by assuming that there is no convection momentum transport (second term on left side of equation 2.3) and flow is stationary or in the laminar regime (first term on left side of equation 2.3).

The Navier-Stokes equation for microfluidics then reduces to equation 2.4.

$$f_{pressure} = -f_{friction}$$

$$\nabla p = \mu \nabla^2 v \tag{2.4}$$

Low Reynolds numbers indicate laminar flow, where the velocity profile of the flow is parabolic (velocity is zero at the walls with the no-slip boundary condition). Turbulent flow is often not reached in microfluidics due to the micro-scale characteristic length. In this microfluidic system, Reynolds number is always less than one, indicating that viscous forces dominate over inertial forces.

Another important dimensionless parameter in microfluidics is the capillary number, *Ca*. The capillary number is the ratio of viscous forces ($3\pi\mu v d$) to interfacial forces ($2\pi\sigma d$). Equation 2.5 shows the relationship between the viscosity μ and interfacial tension σ.

$$Ca = \frac{\mu v}{\sigma} \tag{2.5}$$

Capillary number is important in the dripping regime because the two dominating forces are the viscous and interfacial force. Inertial force is less important due to the inner fluid flow rate being much less than the outer fluid flow rate to generate monodispersed drops.

2.1.1 Theory of Droplet Formation

Microfluidic droplet generation is analogous to a dripping faucet. A drop begins to form, and the surface tension holds the drop to the faucet. As more water is added to the water drop, the volume increases, and gravitational forces ($\rho g V$) overcomes the surface tension forces (σd), and the drop falls. Now, flip the system horizontally. A microfluidics system involves three main forces as shown in Figure 2.2: viscous, inertial, and interfacial. At low fluid velocities, in the dripping regime, the viscous force is larger than the inertial (which depends on the inner fluid velocity). Once the viscous force overcomes the opposing interfacial force, the droplet shears. Equation 2.5 is one of many definitions for calculating a capillary number and is not a direct translation of the force ratios. Thus, the capillary numbers for the dripping regime are always less than one for this microfluidic system [8]. At larger fluid velocities, in the jetting regime, the inertial force is comparable to the viscous force, and the Reynolds number is still less than one. The droplet shears far away from the orifice of the tapered capillary and pinches off from a long neck that develops due to the increase in the inner fluid speed.

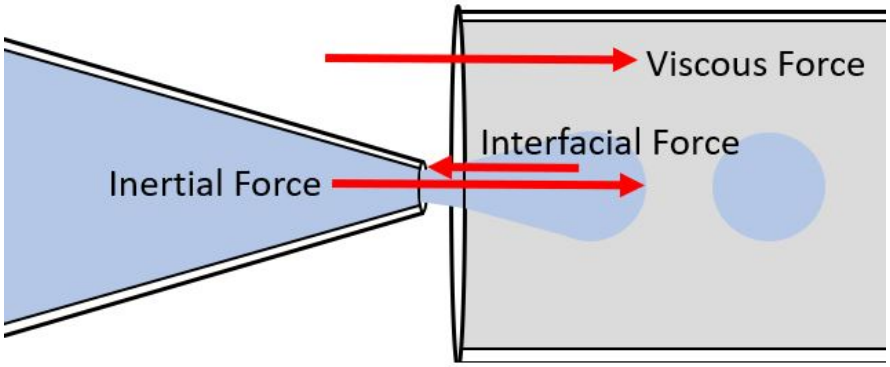


Figure 2.2: Schematic of inertial, interfacial, and viscous forces acting on the fluids in a microfluidic capillary.

The size of the droplet formed depends on a number of parameters. To estimate the size d of the drop given certain constraints and variables, Erb *et al.* [9] fit experimental data to a mathematical model. Equation 2.6 shows the fit for a device generating single emulsions.

$$Ca_{crit}\tilde{\mu}\delta^4 + q_{net}\delta^3 - (q_{net} + Ca_{crit}\tilde{\mu}D^2)\delta^2 - q_1D^2\delta + q_1D^2 = 0 \tag{2.6}$$

$\mathrm{Ca_{crit}}$ refers to the Capillary number at which the drop shears from the orifice. Some dimensionless parameters: the ratio of fluid viscosities is $\tilde{\mu} = \frac{\mu_1}{\mu_2}$, the droplet size $\delta = \frac{d}{D_d}$, and the ratio of capillary diameters is $\mathrm{D} = \frac{D_c}{D_d}$, where D_c is the diameter of the exit capillary and D_d is the diameter of the orifice of the tapered capillary. The flow rates are described by $\mathrm{q_1}=\frac{Q_1}{Q_o}$, $\mathrm{q_2}=\frac{Q_2}{Q_o}$, and $\mathrm{q_{net}}=\mathrm{q_1}+\mathrm{q_2}$. $\mathrm{Q_o}$ is a reference flow rate to non-dimensionalize the flow rates. The takeaway from this complex polynomial (Eq. 2.6) is that this is one example, and possibly the only example, of an analytical method for determining the droplet diameter from experimental data. As shown, this approach for calculating the droplet diameter is tedious. Machine learning can be a useful tool for modeling and predicting microfluidics systems in a much simpler and more time-efficient way.

2.2 A Brief Theory of Machine Learning

Machine Learning (ML) uses large data sets to recognize patterns and make predictions. An artificial neural network (ANN) maps data similar to a brain's neurological branches and connections. This method of modeling has reached a broad scope of fields of study. Beyond computer science and other science fields, machine learning has been widely used for image recognition, recommendation systems, and for making mundane tasks easier. For example, a group used machine learning methods to detect COVID-19 from CRISPR-based surveillance [10]. Another model was created to recommend movies based on user interests [11]. Artificial intelligence has already been replacing human jobs. For example, a literature reviewer at the New York Times experimented with an AI agent to write a book review [12]. More recently, the world of ChatGPT interacts with the user in a conversational way and can be used for writing essays, writing code, summarizing papers, and endless other possibilities [13]. In this age, artificial intelligence is dominating technology and science spaces at rapid rates. For this application, a deep learning model allows predictions of microfluidic droplet sizes, which supplements experiment and theory. An analytical approach for determining the output size of droplets with a given set of initial conditions is not the most reliable. There are various approaches to microfluidics theory, each with their own limitations, which can at best give an approximate estimate for droplet diameter.

Neural networks are comprised of numerous key components. The data must first be sectioned into training and testing data. The training data will set the weights of the network, and the testing data will be used to assess the accuracy of the model. The training data is further separated into two portions: training and validation. The training portion of the training data is responsible for setting initial weights of the network before the validation training data "fine-tunes" the weights each epoch (or iteration) through the network. The input data is the input layer of the network. Following the input layer are the hidden layers, before the output layer of the predicted variable.

Each layer contains nodes, similar to neurons in the brain. The nodes are connected across layers and each connection holds a corresponding weight. Each node

receives data from the previous layer, typically multiple inputs into each neuron. Each input is multiplied by the weight corresponding to the connection and all are added plus a "bias" (or offset value). Note: the weights and biases are random values for the first iteration of the network. The addition of weighted products is then sent to an activation function. If that value exceeds a given threshold, the node will be "activated". If it is activated, the new value will pass to the next layer and repeat a similar process.

Figure 2.3 shows a very simple neural network structure with two input nodes, one hidden layer with two nodes, and the output layer with one node [14].

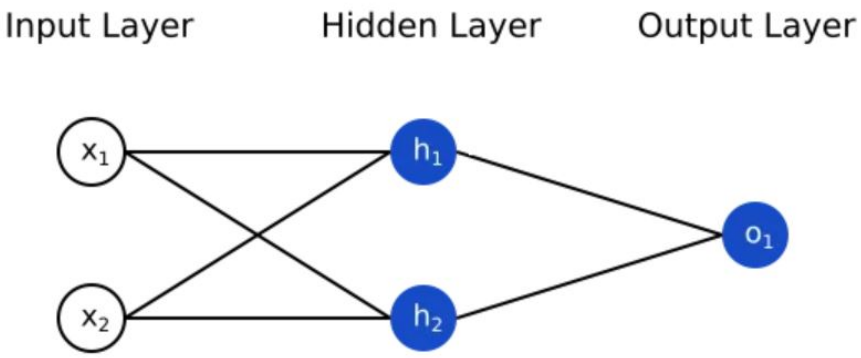


Figure 2.3: Schematic of a simple neural network.

The node h_1 in the hidden layer is the output of the activation function, with the input: x_1*w_1 (weight 1) + x_2*w_2 (weight 2) + b (the bias) [14].

A metric must be used to determine the performance of the training. This is called a loss function. A loss function is essentially a way to determine the error between the predicted value and the true value. Optimization functions aid in minimizing the loss function by determining the weights and biases of the nodes. Optimizers use gradient descent by calculating the gradient of the data to find a local minima, which is used to update the weights from the previous epoch of the network [15].

The training continues through the set number of epochs. Assessing the training of the network is important to ensure the model has not over-fit to the data. Over-fitting is essentially the model memorizing the training data and not being able to predict

well with "unseen" data. Finally, the test data can be sent through the network to get predictions. This process will not affect the final weights of the NN. From there, the predictions can be compared to the true value to determine the accuracy of the model.

2.3 Project Goal

The goal of this project is to predict droplet size using machine learning for three-dimensional glass capillary microfluidic devices from a set of parameters: orifice width, capillary spacing, flow rates, capillary numbers, and the regime (dripping or jetting). Using experimental data, different computational methods are implemented and compared with experimental results to assess the accuracy and success of each method. The best of these can then be used in future research projects. For this application, machine learning can reduce post-experimental analysis time and give more confidence in experimental results. The methods that will be compared are linear regression and a neural network. These are implemented in Python, as well as a graphic user interface that allows ease of predicting droplet properties. Additionally, due to the difficult nature of manipulating 3D glass microfluidic device geometries, this machine learning approach provides knowledge of droplet diameters based on their fluid and device parameters. Previous uses of machine learning in this field have predicted the device sizes, with inputs of droplet diameter. This works well with PDMS devices, where microfluidic device sizes can be fine-tuned.

Chapter 3

Methods

In section 3.1, the experimental methods of creating microfluidic droplets and fluid properties are presented. Section 3.2 discussed the methods of modeling the data: neural networks and linear regression.

3.1 Methods of Microfluidics

Glass capillary microfluidic devices allow for the generation of three-dimensional drops [16] and are relatively simple to make. Since glass capillaries are inert, they are convenient for creating emulsions which require both organic and inorganic materials. A more common microfluidic device is fabricated using PDMS (polydimethylsiloxane), which is a type of polymer than can be easily manipulated to a precise geometry using photolithography [17]. While glass capillaries have advantages over PDMS devices, there are constraints in glass capillary devices for producing droplets of a desired size. For example, when making devices, the one controlled parameter that relates to the process and also relates to the outcome of droplet size, is the diameter of the inner glass capillary's orifice. The standard technique for creating the orifice is to use a pipette puller which heats the capillary and pulls on it until it breaks into two. The point at which it breaks can be somewhat controlled, but the final diameter of the pulled portion can not be fine-tuned without the use of harsh chemicals [18]. Instead, researchers are left with making the most of the pulled capillary that the pipette

puller generates. The obvious drawback is that tuning the drop's diameter is severely limited since the size of the orifice is limited. The lower image in Figure 3.1 shows the pulled pipette on the left of the zoomed-in portion. Drops are formed in the pulled section and flow through the exit capillary.

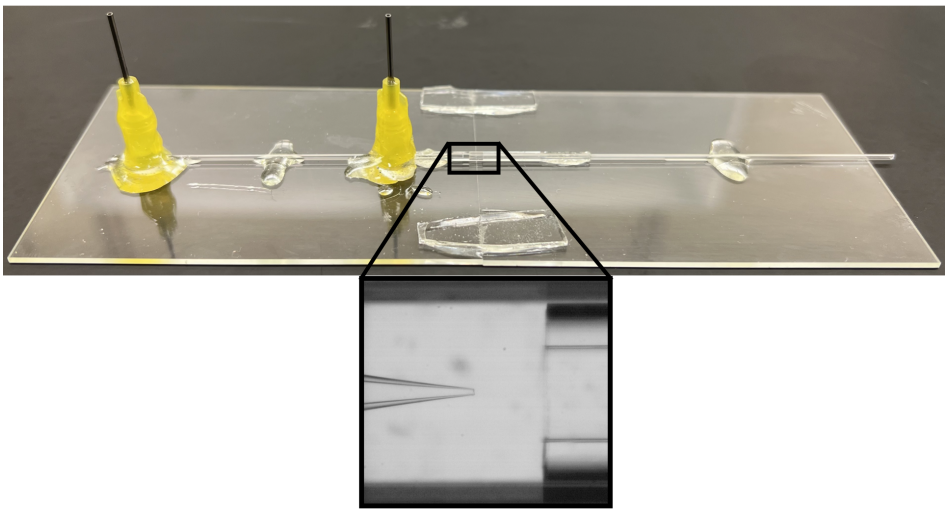


Figure 3.1: Image of the glass capillary device for generating single emulsions and a closer image of the pulled pipette and untapered exit capillary. Photo taken by Dheeraj Sapkota.

For a simple single-emulsion glass capillary device, one round capillary with a tapered orifice and another untapered round capillary are slid into each end of a square capillary. The outer dimension of the round capillaries match the inner dimension of the square capillary creating a snug fit. The two round capillaries are brought close together and after alignment is checked, the capillaries are sealed to a glass slide with five-minute epoxy. In the co-flow geometry, which is used here, the inner fluid flows in the same direction as the outer fluid as shown in the schematic of Figure 3.2. Droplets are generated when the shear force of the outer fluid is higher than the interfacial force which seeks to keep the drop at the orifice.

Fluids are sent into the device using calibrated syringe pumps. The inner fluid is distilled water and typical flow rates for the inner fluid are 50 - 500 μL/hr. The

outer fluid which flows between the space of the inner round capillary and the outer square capillary is soybean oil with 10% Abil EM90 surfactant. Typical flow rates used for the outer fluid are between 2000 - 70000 μL/hr. Table 3.1 contains the key fluid properties for the water and oil: viscosity and interfacial tension. These two parameters comprise the capillary number, which is used as an input for the machine learning model.

Table 3.1: Summary of the fluid properties. Capillary number is defined by μ_w: viscosity of water, μ_o: viscosity of oil, U_w: velocity of water, U_o: velocity of oil, and γ: interfacial tension between water and oil. Measurements were made by Dheeraj Sapkota (2022).

Fluid	Viscosity (mPa s)	Interfacial Tension (mN/m)	Capillary Number
Water	1	6.83	$\frac{\mu_w U_w}{\gamma}$
Oil w 10% EM90	75		$\frac{\mu_o U_o}{\gamma}$

As shown in Figure 3.2, a schematic of a single emulsion device labels the key parameters. The inner fluid (water) flow rate is denoted by Q_1, and the outer fluid (oil) is denoted by Q_2. The orifice width, w, is the diameter of the pulled pipette capillary. The capillary spacing, s, is the distance between the edge of the tapered capillary and the flat exit capillary. The droplet diameter, d, is measured along the horizontal axis.

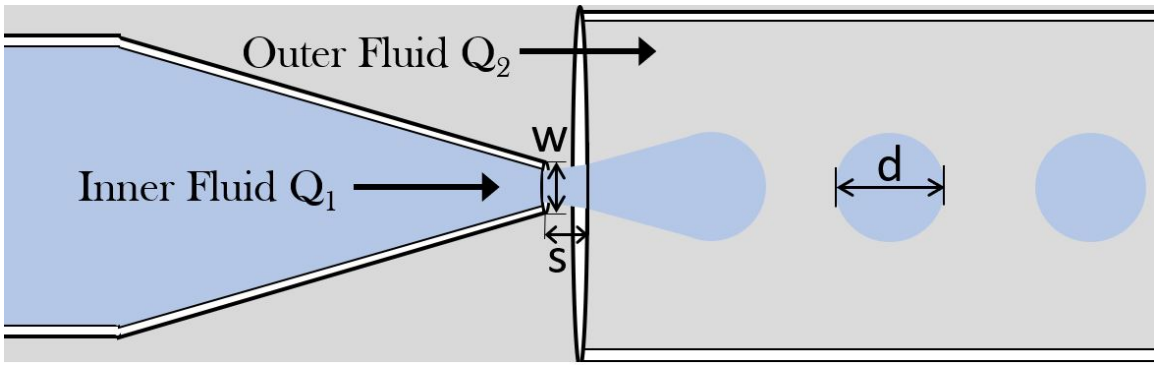


Figure 3.2: Schematic of generation of single emulsions in the dripping regime of a glass capillary device. The inner fluid is water and the outer fluid is oil. Fluid flow rates (Q_1 and Q_2) and viscosities are used for predictions of the droplet diameter, **d**. The orifice width is denoted as **w** and the capillary spacing is **s**.

Droplet sizes are measured from images taken of drops from microfluidic devices under an optical microscope with 4x magnification. The horizontal diameter is measured in ImageJ to convert the pixels to μm. Droplet sizes generated with these single emulsion devices varied from 161 microns to 1085 microns in diameter, and the outer to inner flow rate ratios ($\frac{Q_2}{Q_1}$) varied from 0.75 to 140. The total number of data points is 800, which is comparable to the number of data points used for similar projects (i.e. 888 data points used by Lashkaripour *et al.* [19] or 742 data points used by Mahdi and Daoud [20]).

3.2 Methods of Modeling

3.2.1 Machine Learning

Much of the background for the methods of implementing machine learning to this application comes from Lashkaripour *et al.* [19]. Their method of incorporating machine learning involved inputting data of fluid properties and the desired droplet size of mineral oil in water. Then, a neural network was fabricated to predict the geometry of the drop generator and fluid flow rates. Once this base-model is trained, the input parameters for different fluids can be changed to predict these variables for other systems. This research group used a large data set, which is crucial to effectively training a neural network and avoiding over-fitting. Other research groups' methods, Hadikhani *et al.* [21] and Dressler *et al.* [22], were used for supplemental material for the background of this project. Hadikhani *et al.* used images of droplets to train a deep neural network to predict fluid properties like surface tension and viscosity. Dresseler *et al.* also used image processing to train a network, which outputs flow rate predictions to automate the pressure pumps of the fluids to create a desired droplet size. These three mentioned projects all use PDMS devices, which allow for precise control over the device geometries. However, the nature of PDMS devices constrain the droplets to two-dimensions. In Dr. Adams' research group, three-dimensional devices are used, which creates three-dimensional droplets that can be encapsulated and used for commercial applications. As previously described, manipulating the device geometries for the 3D capillaries is limited.

Applying similar techniques for constructing a neural network, the operation of three-dimensional glass capillary devices can be better understood. Instead of having the network predict the geometries of the device, those will be inputs. The droplet diameter will be the predictor. In the realm of machine learning, a select number of nodes in each layer of the artificial neural network (ANN) determine correlation between the input nodes (input data). The attributes (or input parameters) of this ANN are: orifice width (in μm), capillary spacing (in μm), flow rate ratio, inner fluid capillary number, outer fluid capillary number, and the regime (dripping vs. jetting).

In Figure 3.2, the orifice width, w, is the diameter of the opening with which the inner fluid flows. The capillary spacing, s, is the distance between the orifice opening and the second capillary. The flow rate ratio is the outer fluid flow rate over the inner fluid flow rate $\frac{Q_2}{Q_1}$(flow rates are measured in μL/hr). The capillary numbers are calculated as described in Table 3.1. These parameters were selected because they each affect the droplet diameter.

Reynolds numbers are important parameters to consider for PDMS devices because most experimental groups want to predict the device geometries with a desired droplet size [19], [21], [22]. Capillary numbers were used as a parameter for this project because it describes the forces acting on the fluid, similarly to the Reynolds number. Capillary numbers do not depend on droplet diameter, whereas the Reynolds number does with the characteristic length l. Since the predicted parameter is the droplet diameter, it is crucial that it is independent of the input parameters.

The flow rate ratio captures the inertial forces that a Reynolds number would, which gives confidence that no important information is missing in describing the fluid system.

The data frame consists of 800 points; 80% of the data is for training the model and 20% is for testing. The 80% for training is further split into two subsets, where 80% of the 80% is the actual training data and the remaining 20% of the 80% is the validation data. Then, 20% of the total data is for testing after the network has been trained. This leaves 512 data points for training, 128 for validation, and the remaining 160 for testing. The training data determines the weights of the network, then the validation data fine-tunes these weights each iteration or epoch through the network. The testing data is used to gather a set of predictions to compare with the tabulated data; this data does not alter the network.

The keras ANN model was created in Python (see Appendix A for code). The size of the input layer is the number of attributes, which is six, followed by several hidden layers with rectified linear activation functions (ReLU), as shown in Fig. 3.3. The size of hidden layers are 64, 16, then 8. The output layer is the droplet diameter. Figure 3.3 shows a schematic of a neural network: the input layer contains each parameter, the hidden layers are demonstrated with one layer showing connections between the

nodes in the previous and next layer, and the output layer is a singular node for the droplet diameter.

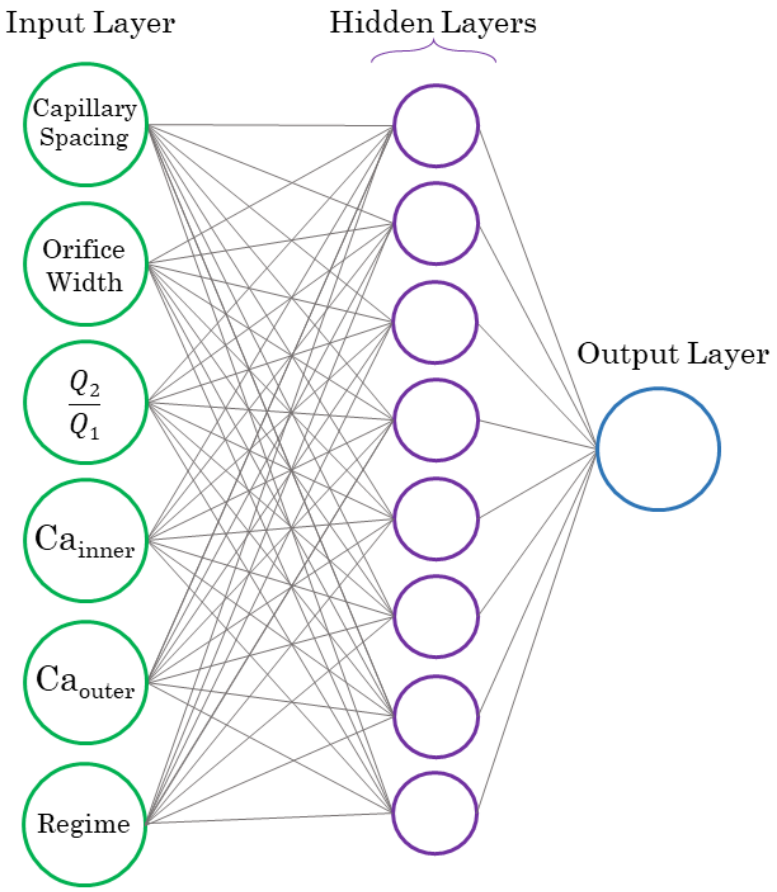


Figure 3.3: Neural network model with inputs, hidden layers (not accurate to actual model), and output layer of prediction.

Educated trial-and-error was the main method for determining the final NN parameters, like the number of hidden layers, nodes per hidden layer, loss function, activation function, optimizer, and learning rate. Understanding why other research groups chose their parameters and researching various methods of building a neural network provided insight and intuition. A rectified linear unit (ReLU) activation function is a simple piece-wise function of linear components that allows for fast computation, while capturing complex behaviors in the data through the non-linearity

of the function [23]. The ReLU proved to be an effective activation function for this data. Mean squared error is a commonly used and simple loss function that is intuitive [24]. Adam optimizer was used to update parameter weights and minimize the mean squared error loss function. The Adam optimizer uses stochastic gradient descent to first and second order estimation with computational efficiency [25]. For the remaining parameters: number of hidden layers, nodes per hidden layer, and learning rate, these values were adjusted until both the training and accuracy of the NN were optimized. The learning rate, which sets the step size at each epoch to minimize the loss function, is 0.0001. The model is saved after training and reloaded for predictions to eliminate the need to retrain the network for each desired prediction.

To understand how each parameter affects the predictor (droplet size), their correlations were determined. Table 3.2 contains the values of how closely an input is correlated to the droplet diameter. In other words, if each parameter was plotted with droplet diameter, the correlation is the best fit line of the plot. A value of 1.0 would indicate perfect correlation.

Table 3.2: The correlation magnitude relates how correlated each parameter is with the droplet diameter. In other words, this is the slope if each were linearly compared. A magnitude of 1.0, would indicate perfect correlation.

Parameter	Correlation Magnitude to Droplet Diameter
Capillary Spacing	0.39
Orifice Width	0.25
Flow Rate Ratio	0.44
Inner Capillary Number	0.58
Outer Capillary Number	0.42
Observed Regime	0.58

The inner capillary number and observed regime (dripping or jetting) have the highest correlation values with the droplet diameter (both 0.58). Thus, these parameters are the most influential in determining the size of the drop.

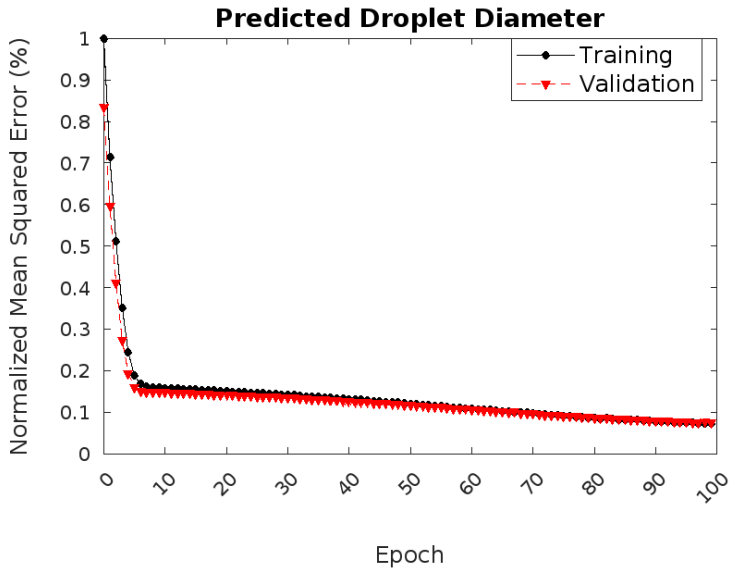
3.2.2 Linear Regression

The experimental dataset indicates a neural network may handle predicting droplet diameter more accurately because trends and patterns in the data break down at certain points. Linear regression was attempted as a comparison with the predictions of the NN (see Appendix B for code). Linear regression maps a linear model to the data by minimizing the residual sum of squares [26]. The same seed was used to split the training and test data as the neural network. This allows for a direct comparison across the same 20% of test data. The 80% of training data was input into a simple linear regression model. The test data was plugged into the linear equation, and each 160 data points produced a corresponding prediction.

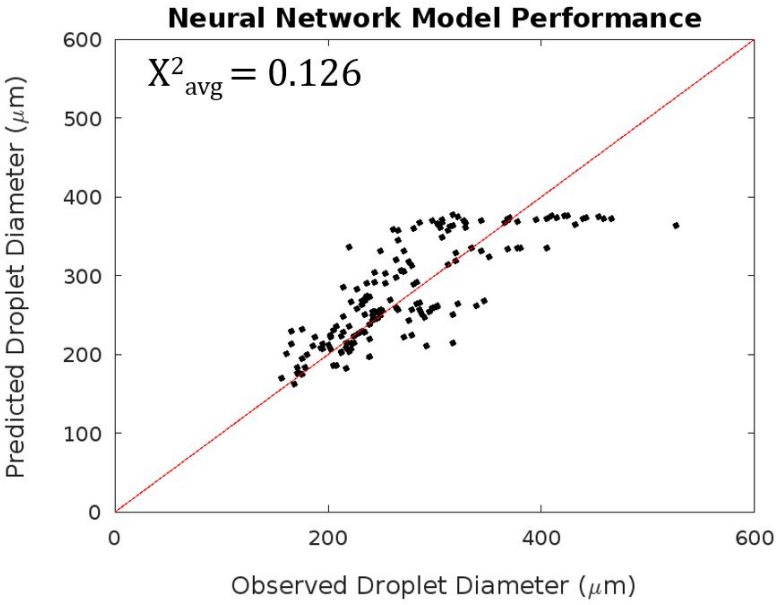
Chapter 4

Results and Discussion

The neural network was trained through 100 epochs to aim for a less than 10% mean squared error for the training and validation. The results from the training and validation data are shown in the top of Figure 4.1. On the bottom, the predictions are plotted with the true experimental data. Values close to the red line (with a slope of 1), are more accurately predicted. A χ^2 value is calculated for the data to be 0.126.



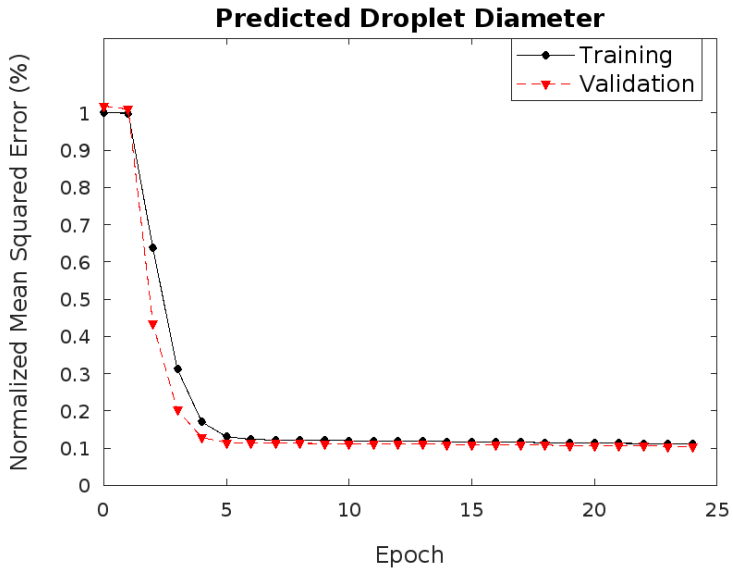
(a) Training and Validation.



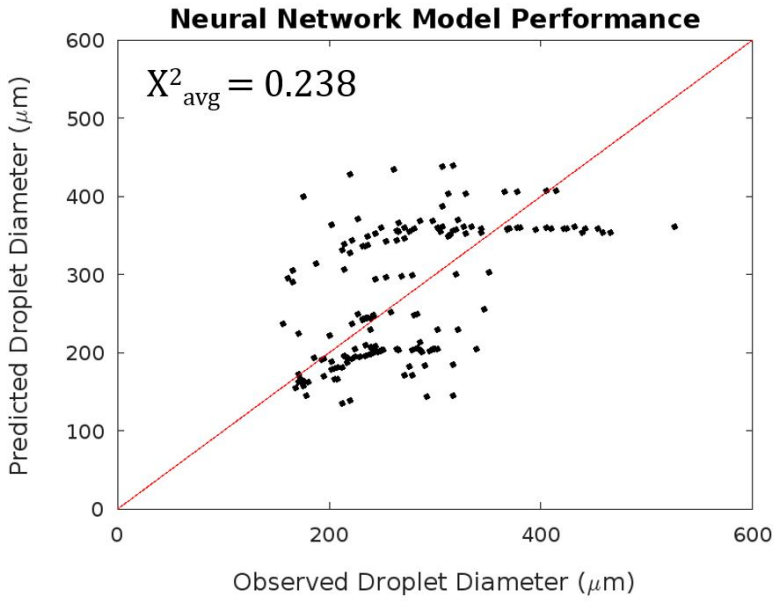
(b) Test Data vs. Predictions.

Figure 4.1: (a) Results of training and validation of droplet diameter prediction model. The mean squared error loss function is plotted for 100 epochs. (b) The predictions from the test data are compared with the actual measured diameter. A slope of 1 refers to a perfect prediction.

From Figure 4.1(a), the training appears to be over-fitting to the data due to the long plateau after about 25 epochs. Using early-stopping methods to avoid the potential over-fitting, 25 epochs were run with all of the same parameters as with 100 epochs. The results from the training and validation data for 25 epochs are shown on the top of Figure 4.2. On the bottom, the predictions are plotted against the experimental data. Due to seeding the split of the data the same for all of these trials, the predictions are from the same input data as with 100 epochs to ensure a direct comparison.



(a) Training and Validation.



(b) Test Data vs. Predictions.

Figure 4.2: (a) Results of training and validation of droplet diameter prediction model. The mean squared error loss function is plotted for 25 epochs. (b) The predictions from the test data are compared with the actual measured diameter. A slope of 1 refers to a perfect prediction.

Figure 4.2(b) displays the χ^2 value for these predictions: 0.238, and the model performed worse across 25 epochs than across 100. Though the training data does not seem to over-fit the data as much as with 100 epochs, the accuracy of the predictions with 100 epochs exceed those with 25 epochs.

Linear regression was used as a simple comparison with a neural network, and the trained model is Equation 4.1.

$$d = 17.18 + 0.268s + 5.208w - 0.893\frac{Q_2}{Q_1} + 1.865 * 10^6 Ca_{inner} + 426.0 Ca_{outer} + 51.36 regime \tag{4.1}$$

The coefficients on linear regression model in Eq. 4.1 show the relationship with the droplet diameter d with each of the dependent terms: capillary spacing s, orifice width w, flow rate ratio $\frac{Q_2}{Q_1}$, inner fluid capillary number Ca_{inner}, outer fluid capillary number Ca_{outer}, and the $regime$ (dripping or jetting). From these values, the inner fluid capillary number proves to have the largest impact on the droplet size by a large margin. The capillary spacing s contributes the least out of the six input variables.

Predictions were made, again, with the same exact set of test data as with the neural networks. Figure 4.3 shows the results of these predictions compared to the experimental data.

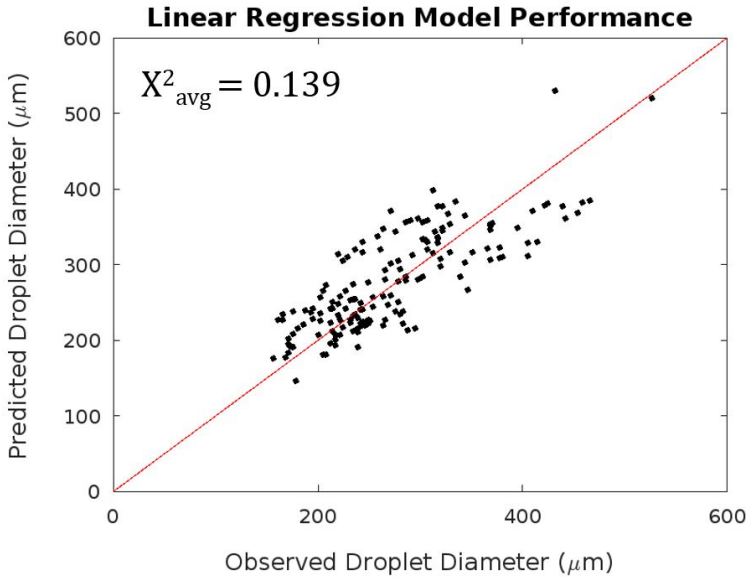


Figure 4.3: The LR model performance is assessed with the same test data as the NN. Again, the predictions are compared with the actual measured diameter. A slope of 1 refers to a perfect prediction.

The χ^2 is 0.139, which is better than the NN with 25 epochs. The best model performance was with the neural network training over 100 epochs. However, the linear regression model seems to handle the larger droplet predictions better than the neural networks. The neural network predictions of droplet diameters greater than about 400μm have larger deviations from the observed droplet diameters, as seen in Fig. 4.1(b) and Fig. 4.2(b). The linear regression model predicts droplet sizes better at larger diameters. Experimentally, there are limits to the droplet production size. The data contains the first droplet sizes that can be produced given the geometric constraints of the device. There is also an upper limit at which the microfluidic droplets reach the jetting regime, producing polydispersed drops that vary greatly in size. Some data from this regime was used to train the neural network to predict the regime (mono vs. polydispersed). However, due to the varied nature of the diameters in the polydispersed region, the droplet diameter predictions are inherently not as accurate beyond the dripping regime.

A tkinter graphic user interface (GUI) was created in Python to allow for a streamlined means of predicting droplet properties. As shown in the top of Figure 4.4, the user will type inputs of capillary spacing, orifice width, fluid flow rates, and dripping or jetting regime. Then, a selection of "first" or "not the first" will determine if the network will be trained or if it will use a previously-trained iteration. Once the model is trained efficiently, the "not the first" option should be selected to make faster predictions. The network does not need to be retrained in between inputs. After the inputs are gathered, the capillary numbers are calculated before calling the model and extracting a prediction. After pressing the "predict" button, a new window pops up with the droplet diameter prediction in μm, as shown in the bottom portion of Figure 4.4.

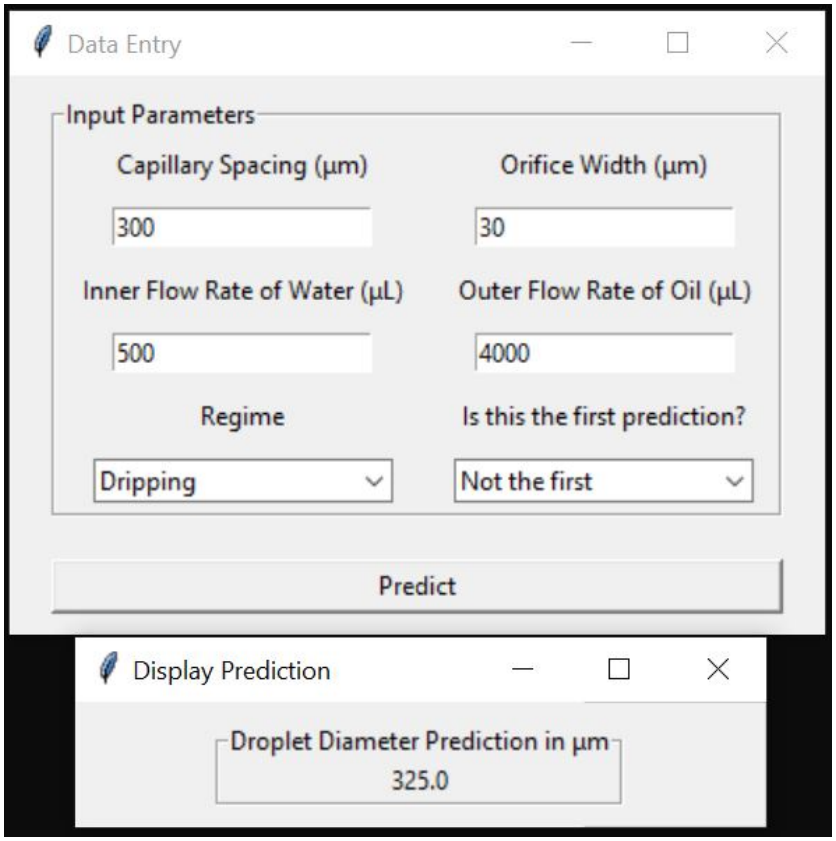


Figure 4.4: Graphic User Interface with inputs of fluid and device parameters, outputs the predicted droplet diameter.

Chapter 5

Conclusion and Discussion

A neural network (NN) model successfully generates predictions of microfluidic droplet diameters for co-flow glass capillary microfluidic devices. A graphic user interface takes a few input parameters and outputs predictions with the click of a button. The training of the model converged to a 7.2% loss and the validation of the model converged to a 7.4% loss, which gives confidence in the accuracy of predictions. A simpler model, linear regression, was compared with the neural network. The neural network with 100 epochs ultimately performed slightly better in the dripping regime, which is more important for applications requiring monodispersed drops. The predictions provide a quick way to understand the outputs of a specific experimental set-up and provide the researcher with timely feedback without needing to use trial and error in selecting fluid flow rates to achieve monodisperse drops of a given size. Due to the difficult nature of generating 3D glass microfluidic devices, where geometries cannot be fine-tuned, this machine learning approach provides knowledge of droplet diameters based on their fluid and device properties. The NN model also effectively eliminates the need for rigorous theory for determining droplet sizes. Modifications of this model can be easily incorporated for other fluids and to predict other properties such as droplet spacing, flow rate ratios to produce a desired droplet size, and regime (mono- vs. polydispersed).

Chapter 6

Future Project Prospects

There are endless ways to use machine learning in research settings, and machine learning can be particularly useful for commercializing or creating automation in microfluidics [27]. For this project, an easy extension would be to change the predictor. Predicting the regime: mono- vs. polydispersed would be a useful tool. Additionally, using the neural network for other fluids beyond oil and water. This would require the user to change the viscosities and interfacial tension values within the code or add these to the GUI. The NN may need to be retrained with the new data of other fluids. Dr. Adams's research focuses on double emulsions. Predicting the diameter of each of the inner drops as well as the outer drop could be added to this project. Double emulsions have commercial applications (e.g. encapsulating drugs), so generating predictions of double emulsions would be a practical use of this NN. Finally, the GUI could be translated into a phone application for ease of use for the experimentalists and theorists that would want to interface with microfluidic predictions.

Chapter 7

Appendices

7.1 Appendix A

7.1.1 Code of Neural Network

Neural Network with Test Data:

```
from tensorflow import keras
import numpy as np
import pandas as pd
from keras.callbacks import CSVLogger

df=pd.read_csv('ourdata6.csv')

train = df.sample(frac=0.8, random_state=0)
test = df.drop(train.index)

x_train=train.iloc[:,0:5]
x_test=test.iloc[:,0:5]

y_train=train.iloc[:,6]
y_test=test.iloc[:,6]
```

```
16 y_test=np.asmatrix(y_test)
17
18
19 def build_nn():
20         model = keras.Sequential()
21         model.add(keras.layers.Dense(64, input_shape=(len(
               x_train.columns),)))
22         model.add(keras.layers.Dense(16, activation="relu"))
23         model.add(keras.layers.Dense(8, activation="relu"))
24         #model2.add(keras.layers.Dense(2, activation="relu"))
25         model.add(keras.layers.Dense(1, activation="relu"))
26         print(model.summary())
27
28         optimizer = keras.optimizers.Adam(learning_rate=0.0001)
29         model.compile(optimizer=optimizer, loss='mse')
30
31         csv_logger = CSVLogger('log2.csv', append=True,
               separator=';')
32
33         model.fit(x_train, y_train, batch_size=5, epochs=100,
               validation_split=0.2, callbacks=[csv_logger])
34
35         model.save('predict_diam_model')
36
37         return model
38
39 def predict():
40     data=np.asmatrix(x_test)
41
42     model = keras.models.load_model('predict_diam_model')
```

```
43
44      diam=model.predict_on_batch(data)
45
46      return data, diam
47
48 model=build_nn()
49 [data, diam]=predict()
50 np.savetxt("ytestNN.csv", y_test, delimiter=";")
51 np.savetxt("ypredNN.csv", diam, delimiter=";")
```

Neural Network and GUI:

```
# Serena Holte
# 06-2022 to 06-2023
# Use a GUI to input parameters --> output predictions
# This code outputs droplet diameter with the following inputs:
    # Orifice width
    # Capillary spacing
    # Flow rate ratio
    # Capillary number (inner and outer)
    # Regime (dripping vs. jetting)

from tkinter import ttk
import tkinter as tk
import math
from tensorflow import keras
import pandas as pd
import numpy as np
from keras.callbacks import CSVLogger

def predict():
    data=np.genfromtxt('gui_input.csv')
    data=np.asmatrix(data)

    model1 = keras.models.load_model('predict_regime_model')
    model2 = keras.models.load_model('predict_diam_model')

    regime1=model1.predict_on_batch(data)
    regime=np.argmax(regime1,axis=1)
    diam=model2.predict_on_batch(data)
```

```
29
30      np.savetxt('regimeprediction.txt',regime)
31      np.savetxt('diamprediction.txt',diam)
32      return regime, diam
33
34
35  def build_nn():
36
37          df=pd.read_csv('ourdata6.csv')
38
39          train = df.sample(frac=0.8, random_state=0)
40          test = df.drop(train.index)
41
42          x_train=train.iloc[:,0:5]
43          x_test=test.iloc[:,0:5]
44          y_train=train.iloc[:,6]
45          y_test=test.iloc[:,6]
46          y_test=np.asmatrix(y_test)
47
48      # model predicts droplet diamter
49          model = keras.Sequential()
50          model.add(keras.layers.Dense(64, input_shape=(len(
                x_train.columns),)))
51          model.add(keras.layers.Dense(16, activation="relu"))
52          model.add(keras.layers.Dense(8, activation="relu"))
53          #model2.add(keras.layers.Dense(2, activation="relu"))
54          model.add(keras.layers.Dense(1, activation="relu"))
55          print(model.summary())
56
57          optimizer = keras.optimizers.Adam(learning_rate=0.0001)
```

```
58          model.compile(optimizer=optimizer, loss='mse')
59
60          csv_logger = CSVLogger('log2.csv', append=True,
                separator=';')
61
62          model.fit(x_train, y_train, batch_size=5, epochs=100,
                validation_split=0.2, callbacks=[csv_logger])
63
64          model.save('predict_diam_model')
65
66          return model
67
68
69  def input_data():
70
71      def enter_data():
72
73          capspace=capsp_entry.get()
74          orf=orf_entry.get()
75          regime=regime_combobox.get()
76          flowin=flowin_entry.get()
77          flowout=flowout_entry.get()
78          num=prd_combobox.get()
79
80          viscoil=0.075
81          viscwater=0.001
82          inttension=0.00683
83          channelarea=math.pi*(0.58*10**-3)**2
84          conv=2.78E-13 # ul/hr to m^3/s conversion
85
```

```
86          capnumo=viscoil*float(flowout)/channelarea*conv/(
                inttension)
87          capnumi=viscwater*float(flowout)/channelarea*conv/(
                inttension)
88          flowratio=float(flowout)/float(flowin)
89
90          if regime == "Dripping":
91              regime=1
92          else:
93              regime=2
94
95          data=np.array([capspace,orf,flowratio,capnumo,capnumi
                ])
96          np.savetxt('gui_input.csv',data, fmt='%s')
97
98          if num =="First":
99              build_nn()
100             predict()
101             diampred=open("diamprediction.txt").read()
102             diampred=round(float(diampred),1)
103
104             regimepred=open("regimeprediction.txt").read()
105             regimepred=round(float(regimepred),0)
106
107             if regimepred==0.0:
108                 regimepredtext='Monodispersed'
109             else:
110                 regimepredtext='Polydispersed'
111
112             window1=tk.Tk()
```

```
113             window1.title("Display Prediction")
114
115             frame1=tk.Frame(window1)
116             frame1.pack()
117             results_frame=tk.LabelFrame(frame1, text="Droplet
                    Diameter Prediction in \u03BCm")
118             results_frame.grid(row=0, column=0, padx=20, pady
                    =10)
119
120             string_variable = tk.StringVar(results_frame,
                    diampred)
121
122             results_frame2=tk.LabelFrame(frame1, text="Regime
                    Prediction")
123             results_frame2.grid(row=1, column=0, padx=20, pady
                    =10)
124             string_variable2 = tk.StringVar(results_frame,
                    regimepredtext)
125
126
127             label = tk.Label(results_frame, textvariable=
                    string_variable)
128             label2 = tk.Label(results_frame, textvariable=
                    string_variable2)
129             label.pack()
130             label2.pack()
131
132             window1.mainloop()
133
134         else:
```

```
135             predict()
136
137             diampred=open("diamprediction.txt").read()
138             diampred=round(float(diampred),1)
139
140             regimepred=open("regimeprediction.txt").read()
141             regimepred=round(float(regimepred),0)
142
143             if regimepred==0.0:
144                 regimepredtext='Monodispersed'
145             else:
146                 regimepredtext='Polydispersed'
147
148             window1=tk.Tk()
149             window1.title("Display Prediction")
150
151             frame1=tk.Frame(window1)
152             frame1.pack()
153             results_frame=tk.LabelFrame(frame1, text="Droplet
                    Diameter Prediction in \u03BCm")
154             results_frame.grid(row=0, column=0, padx=20, pady
                =10)
155
156             string_variable = tk.StringVar(results_frame,
                diampred)
157
158             label = tk.Label(results_frame, textvariable=
                string_variable)
159             label.pack()
160
```

```
161             window1.mainloop()
162
163
164         return data
165
166     window=tk.Tk()
167     window.title("Data Entry")
168
169     frame=tk.Frame(window)
170     frame.pack()
171
172     fluid_props_frame=tk.LabelFrame(frame, text="Input
            Parameters")
173     fluid_props_frame.grid(row=0, column=0, padx=20, pady=10)
174
175     capsp_label=tk.Label(fluid_props_frame, text="Capillary
            Spacing (\u03BCm)")
176     capsp_label.grid(row=0, column=0)
177     capsp_entry=tk.Entry(fluid_props_frame)
178     capsp_entry.grid(row=1, column=0)
179
180     orf_label=tk.Label(fluid_props_frame, text="Orifice Width
            (\u03BCm)")
181     orf_label.grid(row=0, column=1)
182     orf_entry=tk.Entry(fluid_props_frame)
183     orf_entry.grid(row=1, column=1)
184
185
186     flowin_label=tk.Label(fluid_props_frame, text="Inner Flow
            Rate of Water (\u03BCL)")
```

```
187     flowin_label.grid(row=2, column=0)
188     flowin_entry=tk.Entry(fluid_props_frame)
189     flowin_entry.grid(row=3, column=0)
190
191
192     flowout_label=tk.Label(fluid_props_frame, text="Outer
            Flow Rate of Oil (\u03BCL)")
193     flowout_label.grid(row=2, column=1)
194     flowout_entry=tk.Entry(fluid_props_frame)
195     flowout_entry.grid(row=3, column=1)
196
197     regime_label=tk.Label(fluid_props_frame, text = "Regime")
198     regime_combobox=ttk.Combobox(fluid_props_frame, values=["
            Dripping","Jetting"])
199     regime_label.grid(row=4, column=0)
200     regime_combobox.grid(row=5, column=0)
201
202     prd=tk.Label(fluid_props_frame, text = "Is this the first
            prediction?")
203     prd_combobox=ttk.Combobox(fluid_props_frame, values=["
            First","Not the first"])
204     prd.grid(row=4, column=1)
205     prd_combobox.grid(row=5, column=1)
206
207
208     for widget in fluid_props_frame.winfo_children():
209         widget.grid_configure(padx=10, pady=5)
210
211     button=tk.Button(frame, text="Predict", command=
            enter_data)
```

```
212         button.grid(row=1, column=0, sticky="news", padx=20, pady
                =10)
213
214         window.mainloop()
215
216
217 input_data()
```

7.2 Appendix B

7.2.1 Code of Linear Regression

```
import numpy as np
from sklearn.linear_model import LinearRegression
import pandas as pd

df=pd.read_csv('ourdata6.csv')

train = df.sample(frac=0.8, random_state=0)
test = df.drop(train.index)

x_train=train.iloc[:, lambda df: [0,1,2,3,4,5]]
x_test=test.iloc[:, lambda df: [0,1,2,3,4,5]]
x_test=np.asmatrix(x_test)

y_train=train.iloc[:,6]
y_test=test.iloc[:,6]
y_test=np.transpose(np.asmatrix(y_test))

model=LinearRegression().fit(x_train,y_train)

y_pred = model.predict(x_test)


np.savetxt("ytestLR.csv", y_test, delimiter=";")
np.savetxt("ypredLR.csv", y_pred, delimiter=";")
```

References

[1] Petra S. Dittrich, Kaoru Tachikawa, and Andreas Manz. Micro total analysis systems: Latest advancements and trends. *Analytical Chemistry*, **78**:3887–3908, 2006, and references therein.

[2] S. J. Datta, A. Abbaspourrad, E. Amstad, J. Fang, S. H. Kim, M. Romanowsky, H. C. Shum, B. Sun, A. S. Utada, M. Windbergs, S. Zhou, and D. A. Weitz. 25th anniversary article: Double emulsion templated solid microcapsules: Mechanics and controlled release. *Advanced Materials*, **26**(14):2205–2218, 2014, and references therein.

[3] L.-H. Hung S.-Y. Teh, R. Lin and A. P. Lee. Droplet microfluidics. *Lab Chips*, **8**:198, 2008.

[4] L. L. A. Adams, T. E. Kodger, S.H. Kim, H. C. Shum, T. Franke, and D. A. Weitz. Single step emulsification for the generation of multi-component double emulsions. *Soft Matter*, **8**:10719, 2012.

[5] L. L. A. Adams, D. Lee, Y. Mei, D. A. Weitz, and A. A. Solovev. Nanoparticle-shelled catalytic bubble micromotor. *Adv. Mater. Interfaces*, **7**:1901583, 2020.

[6] P. K. Kundu, I. M. Cohen, and D. R. Dowling. *Fluid Mechanics*. Elsevier, 2016.

[7] Eugenia Kumacheva. Lecture notes from boulder summer school, polymers in soft and biological matter, 2012.

[8] Andrew S. Utada, Alberto Fernandez-Nieves, Howard A. Stone, and David A. Weitz. Dripping to jetting transitions in coflowing liquid streams. *Phys. Rev. Lett.*, **99**:094502, 2007.

[9] Randall M. Erb, Dominik Obrist, Philipp W. Chen, Julia Studera, and Andre R. Studart. Predicting sizes of droplets made by microfluidic flow-induced dripping. *Royal Society of Chemistry*, **7**:8757–8761, 2020.

[10] H. Metsky, C. Freije, T. Kosoko-Thoroddsen, P. Sabeti, and C. Myhrvold. Crispr-based surveillance for covid-19 using genomically-comprehensive machine learning design. *bioRxiv*, 2020.

[11] F. Furtado and A. Singh. Movie recommendation system using machine learning. *International Journal of Research in Industrial Engineering*, **9**(1):84–98, 2020.

[12] K. Roose. A robot wrote this book review. *The New York Times*, Nov 2021. Available at: `https://www.nytimes.com/2021/11/21/books/review/the-age-of-ai-henry-kissinger-eric-schmidt-daniel-huttenlocher.html`.

[13] Kevin Pocock. What is chatgpt and what is it used for? *PC guide*, May 2023. Available at: `https://www.pcguide.com/apps/what-is-chat-gpt/`.

[14] Victor Zhou. Machine learning for beginners: An introduction to neural networks. *Towards Data Science*, Mar 2019. Available at: `https://towardsdatascience.com/machine-learning-for-beginners-an-introduction-to-neural-networks-d49f22d238f9`.

[15] Apoorva Agrawal. Loss functions and optimization algorithms. demystified. *Medium*, Sept. 2017. Available at: `https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c`.

[16] A. S. Utada, E. Lorenceau, D. R. Link, P. D. Kaplan, H. A. Stone, and D. A. Weitz. Monodisperse double emulsions generated from a microcapillary device. *Science*, **308**(5721):537–541, 2005.

[17] David C. Duffy, J. Cooper McDonald, Olivier J. A. Schueller, and George M. Whitesides. Rapid prototyping of microfluidic systems in poly(dimethylsiloxane). *Analytical Chemistry*, **70**:4974–84., 1998.

[18] Chunyang Wei, Chengzhuang Yu, Shanshan Li, Jiyu Meng, Tiejun Li, Feng Pan, Zichao Wang, and Junwei Li. Microfabrication of on-demand tapered glass capillaries by etching combined with polishing. *Journal of Microelectromechanical Systems*, **31**(3):473–480, 2022.

[19] A. Lashkaripour, C. Rodriguez, N. Mehdipour, R. Mardian, D. McIntyre, L. Ortiz, J. Campbell, and D. Densmore. Machine learning enables design automation of microfluidic flow-focusing droplet generation. *Nature Communications*, **12**:1–14, 2021.

[20] Y. Mahdi and K. Daoud. Microdroplet size prediction in microfluidic systems via artificial neural network modeling for water-in-oil emulsion formulation. *Journal of Dispersion Science and Technology*, **38**(10):1501–1508, 2017.

[21] P. Hadikhani, N. Borhani, M. Hashemi, and D. Psaltis. Learning from droplet flows in microfluidic channels using deep neural networks. *Sci. Rep.*, **9**:8814, 2019.

[22] O. Dressler, P. Howes, J. Choo, and A. deMello. Reinforcement learning for dynamic microfluidic control. *ACS Omega*, **3**:10084–10091, 2018.

[23] Jason Brownlee. A gentle introduction to the rectified linear unit. *Machine Learning Mastery*, Aug. 2020. Available at: `https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/`.

[24] George Seif. Understanding the 3 most common loss functions for machine learning regression. *Towards Data Science*, May 2019. Available at: `https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3`.

[25] Jason Brownlee. Gentle introduction to the adam optimization algorithm for deep learning. *Machine Learning Mastery*, Jan. 2021. Available at: `https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/`.

[26] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson, 2005.

[27] S. Srikanth, S. K. Dubey, A. Javed, and S. Goel. Droplet based microfluidics integrated with machine learning. *Sensors and Actuators*, **332**:113096, 2021.