



VCU

Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations

Graduate School

2023

Innovations in Drop Shape Analysis using Deep Learning and Solving the Young-Laplace Equation for an Axisymmetric Pendant Drop

Andres P. Hyer
Virginia Commonwealth University

Follow this and additional works at: <https://scholarscompass.vcu.edu/etd>



Part of the [Artificial Intelligence and Robotics Commons](#), [Fluid Dynamics Commons](#), [Non-linear Dynamics Commons](#), and the [Ordinary Differential Equations and Applied Dynamics Commons](#)

© The Author

Downloaded from

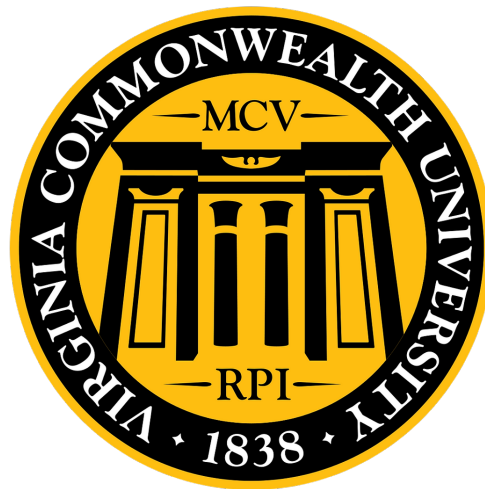
<https://scholarscompass.vcu.edu/etd/7352>

This Thesis is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.

Innovations in Drop Shape Analysis using Deep Learning and Solving the Young-Laplace Equation for an Axisymmetric Pendant Drop

Andres P. Hyer

A thesis presented for the degree of
Master of Science



Advisor: James K. Ferri, PhD
Chemical and Life Science Engineering
Virginia Commonwealth University
Richmond VA, USA
May 2023

Acknowledgements

I wish to express my deepest appreciation to everyone who has helped me reach this point. There are many individuals in my life whose unwavering support has been instrumental in shaping my current position.

Foremost, I thank Adam Luxon, my initial mentor who fostered my passion for coding, particularly in the realms of machine learning and deep learning. I am also indebted to Dr. Tyler McQuade who gave me the opportunity to commence my research at VCU. Without the guidance of both Adam and Dr. McQuade, I believe I wouldn't have nurtured the fervor for coding that I possess today. Their support transcended academics and research, providing a dependable pillar during my early college years.

A special acknowledgement goes to Dr. James Ferri who mentored me during my later college years and throughout my Master's studies. His guidance allowed me to mature professionally, teaching me not only how to learn but also how to strive diligently. The projects we collaborated on remain close to my heart, and I cherish the experiences beyond the research environment and the classroom - especially the memorable summer party steak!

I also extend my gratitude to Quang Le, Jermaine Cort, Jordan Nowaczyk, Kaitlin Kay, and Rebecca Walker. Quang, in particular, is not just a colleague but also one of my best friends. Together with Jordan and Jermaine, we braved the challenging Master's program. Kaitlin and Rebecca deserve a special mention for their collaborative efforts in my research projects.

Lastly, I am immensely thankful to my family. Their unwavering support and

love have been my stronghold during academic and personal trials. Life presents a myriad of challenges, and having a supportive family to weather them with is an invaluable blessing. My deepest gratitude goes to my fiancée, Joanna Like. Her constant presence and support have been my guiding light, and I couldn't imagine journeying through life without her.

Thank you all for being part of my academic journey and for contributing to the successful completion of my Master's thesis.

Contents

1	Introduction	3
2	Perturbation Solution	7
2.1	Perturbation Approach	7
2.2	Scaling Profile	11
3	Deep Learning Approach	15
3.1	Training Convolutional Neural Network (CNN) Models	16
3.1.1	Numerically Generating Drop Profiles	16
3.1.2	Numerically Generating Drop Images	20
3.1.3	Training CNN Models and Accuracy in Generated Data	27
3.2	Testing Final Models on Experimental Pendant Drop Images	30
4	Conclusion	35
A	Code for Deep Learning Approach	36
A.1	Imported Packages	36
A.2	Backend Functions	37
A.3	Numerical Drop Profile Creation	40
A.4	Numerical Drop Image Creation	42
A.5	CNN Model Architecture	46
A.6	Training and Testing Models	46
A.7	Configuration File	49
A.8	Wrapper to Generate Data	52
A.9	Wrapper to Predict Fluid Properties	59

List of Figures

1	Axisymmetric drop shape analysis experimental setup.	5
2	Comparison of the Numerical and Perturbation Solution to the Young-Laplace equation.	12
3	Comparison of the Numerical and Corrected Solution to the Young-Laplace equation.	13
4	Difference between scaled profiles and non-scaled profiles.	14
5	Workflow for using a CNN to Fluid Properties.	18
6	Affect of Bo and Wo on numerically generated drop profiles.	19
7	Representative Samples from Image Group Training Sets.	24
8	Machine Learning Regression Metrics.	26
9	Experimentally calculating Bo using traditional method.	31

List of Tables

1	Predicted Properties.	17
2	Summary of Image Groups for CNN Model Training. Parameters for Generating Drop Images.	22
3	Performance metrics to evaluate how each CNN model in each group performed in predicting fluid properties.	28
4	Predicted versus actual fluid properties.	32

1 Introduction

Surface (or interfacial) tension is a thermodynamic property describing the contractile tendency of an interface due to the attractive interactions between molecules in the bulk phase and the loss of coordination that is associated with the interface [1]. One consequence of this is that in the absence of external forces, drops and bubbles contract into shapes that minimize the surface to volume ratio [2]. Even when other forces, such as those produced by gravity, flow, or electric fields, are present, the shape of a bubble or drop is still determined, at least in part, by the surface tension [3]. In fact, if gravity and surface tension are the only acting forces, the surface tension of the drop can be calculated from its shape [4]. This is the basis of the simple, yet elegant method for measuring surface tension: axisymmetric drop shape analysis (ADSA) [5]. Accurate surface measurements are important to optimize surfactant concentrations in formulations, characterize interfacial properties of biomolecules, and investigate the wetting properties of surfaces [6]. Moreover, ADSA can be employed to study the performance and stability of surface-active materials such as coatings and adhesives [7].

For a drop at rest, the Young Laplace equation relates the mean curvature of the drop interface ($2H$) to the forces acting on the drop, namely the pressure drop (ΔP) across the drop interface and the surface tension (γ) [8].

$$2H\gamma = \Delta P = \Delta P_0 + \Delta\rho g z \quad (1)$$

Using geometric relationships to describe the surface coordinates as shown in Figure 1, the Young-Laplace equation can be transformed to a system of first order ordinary differential equations (ODEs) [9].

$$\frac{d\phi}{ds} = 2 \pm \mathbf{Bo} z - \frac{\sin \phi}{x} \quad (2)$$

$$\frac{dx}{ds} = \cos \phi \quad (3)$$

$$\frac{dz}{ds} = \sin \phi \quad (4)$$

$$\phi(s=0) = x(s=0) = z(s=0) = 0 \quad (5)$$

Where s is the arc length along the drop, ϕ is the angle between dz and dx , and x is the radius of the drop at a height of z . The variables s and ϕ can be written as $(ds)^2 = (dx)^2 + (dz)^2$ and $\tan \phi = dz/dx$, which are manipulations of Equations 2-4. There is a single dimensionless group which describes the shape of a pendant drop.

$$\mathbf{Bo} = \frac{\Delta \rho g R_0^2}{\gamma} \quad (6)$$

Where $\Delta \rho$ is the change of densities across the fluid interface, g is gravity (assumed to be positive), R_0 is the drop radius, and γ is the surface tension. The different variables can be seen in Figure 1. The Bond (\mathbf{Bo}) number represents the balance between gravitational forces which distend the drop from the spherical shape, and surface tension forces, which are a contractile tendency that favors the spherical shape. In order to successfully leverage observations of drop and bubbles to measure surface tension, there are established ranges ($0.09 < \mathbf{Bo} < 0.35$) that constrain precision [1].

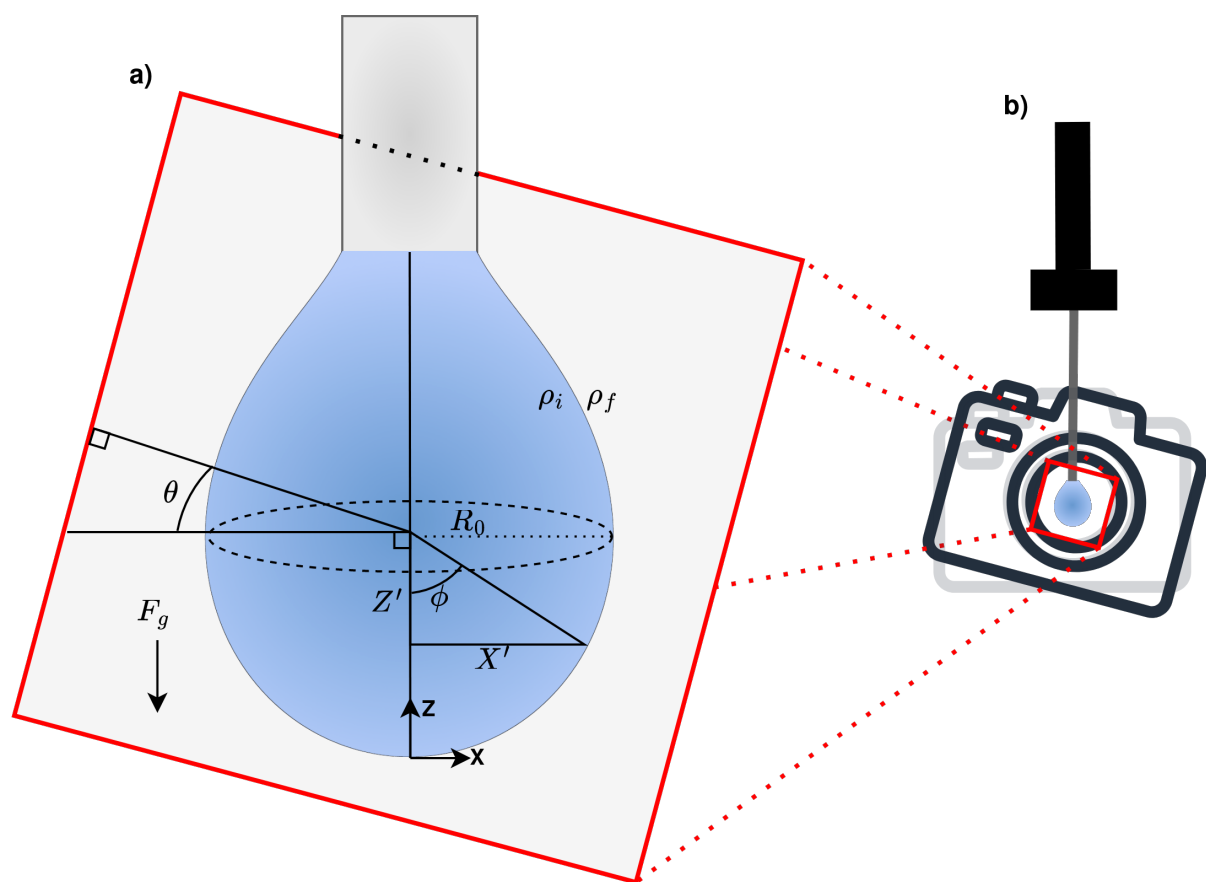


Figure 1: Axisymmetric drop shape analysis. a) The coordinate system based on Young-Laplace equation, with rotation of image accounted for. b) General experimental setup, where the camera can be rotated in the xz -axis.

The experimental setup is relatively simple. Typically, a light source is used to cast a silhouette of a drop onto a camera, where the shape of the drop is captured and used for analysis. As it is traditionally best when the light source is diffuse to minimize optical aberrations and reflections from the drop interface [10]. In traditional ADSA, image rotations are minimized. The rotation of an image can be seen in Figure 1, represented as θ .

ADSA is a powerful technique to accurately determine the surface tension of liquids which is widely employed in various applications, including chemicals, materials, and consumer products manufacturing. Its noninvasive nature and broad applicability makes ADSA a versatile tool in product development. One key application of ADSA is determining the optimal surfactant concentration in formulations, which helps in minimizing costs and ensuring product performance [11, 12, 13]. ADSA also plays a crucial role in studying the fundamental adsorption phenomena of biomolecules, such as proteins, peptides, and lipids, at interfaces [14]. This information is essential for understanding biological processes, optimizing drug delivery systems, and designing advanced biomaterials [15, 16, 17, 18, 19]. Additionally, ADSA is used to measure contact angle, which provide insights into surface wettability. This information has broad implications in various applications, including self-cleaning surfaces, inkjet printing, and biological adhesion [20, 21, 22, 23, 24]. Furthermore, ADSA assists researchers in evaluating the effectiveness and durability of surface-active materials, such as coatings and adhesives. By monitoring the change in interfacial tension and contact angle over time, researchers can assess the stability and performance of these materials, enabling them to optimize formulations and ensure long-term reliability in applications like corrosion protection and adhesion [25, 26, 27, 28].

Traditional ADSA algorithms use numerical ODE solvers to solve the Young-Laplace equation for the drop profile. An initial guess for Bo is taken, and the profile is numerically solved. The error between theory and observation is calculated, and the objective is to identify a best fit Bo for experimental drop image. [9, 29].

Although ADSA is widely used, there are still many limitations in traditional ADSA. The largest limitation is that high resolution images are required. As high pixel densities are required, in order to accurately calculate the error between the true drop profile and the numerically generated drop profile [30]. Traditional ADSA is also computationally very expensive, as the system of ODEs has to be numerically solved many times. High quality images with minimal noise in the images are also required, as even small amounts of noise can lead to very inaccurate measurements [31]. Lastly, images must be perpendicular to the horizontal axis, as Equations 2-4 do not account for possible rotations in the drop profile [32].

These challenges are tackled through the use of a deep learning approach using a convolution neural network (CNN) as well as deriving an analytical solution that can be used to replace numerical approaches.

2 Perturbation Solution

2.1 Perturbation Approach

Deriving an analytical solution that can accurately characterize the profile of an arbitrary axisymmetric drop as $x = f(Bo, z)$ is very challenging. While there is one single set of equations, different axisymmetric drop shapes can arise depending on

the boundary conditions. But the most common boundary conditions are $\phi(s=0) = x(s=0) = z(s=0) = 0$ which lead to sessile and pendant drops.

Sessile drops are drops that are resting on a flat plane and are typically used to measure contact angle. Contact angle measurements are useful to determine the wetting properties of a liquid on a surface interface. Sessile drops are defined by where $\pm Bo = +Bo$ in Equation 2. The positive Bond number leads to an ellipse-like shape. In fact, it has been previously be shown that an ellipse is a valid solution for sessile drops [33]. While sessile drops can be used to measure surface tension, measurements are very challenging when the drop has a low contact angle. Where the contact angle is defined as the angle formed between the tangent line at the liquid-solid-gas interface and the solid surface, measured through the liquid phase.

For surface measurements, pendant drops are often used as the accuracy of the measurements do not depend on any liquid/surface interactions. A pendant drop is a drop that is suspended from a capillary tip. Which represents where $\pm Bo = -Bo$ in Equation 2. While the problem for positive Bond numbers has been reported previously, there has been no solution for negative Bond number drops. Accurate measurements for surface tension are important for many fields, such as pharmaceutical manufacturing, chemical processing, and material science. Manifold studies reference ADSA as an important experimental approach [34].

To solve the set of differential equations, first the following relationships can be determined from Equations 2-4,

$$\frac{d\phi}{ds} = \frac{d}{dx}(\sin \phi) = -\frac{d}{dz}(\cos \phi) \quad (7)$$

which allows for representing the equations as a single, second order ODE.

$$\frac{d}{dx} \left(\frac{d}{dx} \sin \phi + \frac{\sin \phi}{x} \right) + Bo \frac{\sin \phi}{\cos \phi} = 0 \quad (8)$$

A general, exact solution to Equation 8 for arbitrary Bo has not yet been reported. In the limit $Bo \approx 0$, together with the boundary conditions $x(s=0) = 0 : \sin \phi = 0$ and $x(s=0) = 0 : \frac{d}{dx}(\sin \phi) = 1$ a base-state solution, i.e.,

$$\sin \phi = x \quad (9)$$

can be found. This corresponds to the spherical shape, for which there is no limit on the value of the surface tension. Applying Equation 9 into Equation 2, an approximate solution can be found.

$$\cos \phi = \frac{Bo}{2} z^2 - z + 1 \quad (10)$$

Equation 10 together with Equations 3 and 4, the axisymmetric shape is:

$$x = \int \frac{\cos \phi}{\sin \phi} dz = \int \frac{\cos \phi}{\sqrt{2Bo(\cos \phi - 1) + 1}} d\phi \quad (11)$$

Although there is no elementary solution to the integral in Equation 11, it can be solved in terms of elliptic integrals.

$$x = \frac{1}{Bo} \left[E \left(\frac{\phi(z)}{2} \middle| 4Bo \right) - (1 - 2Bo) F \left(\frac{\phi(z)}{2} \middle| 4Bo \right) \right] \quad (12)$$

where $E(\alpha, m)$ is the incomplete elliptic integral of the second kind, and $F(\alpha, m)$ is the incomplete elliptic integral of the first kind. It can be noted that Equation 12

is a perturbation analytical solution that reduces to Equation 9 as Bo approaches zero; accuracy of Equation 12 decreases as Bo and z increase. The accuracy of the analytical solution can be seen in Figure 2.

The perturbation solution presented is analytical as z approaches zero; i.e. as the pressure drop approaches a constant, the drop shape is spherical. As pressure decreases linearly in z according to Equation 4, the error in Equation 12 grows (as expected) and reduces the practical utility in application to experimental data analysis. The solution can be corrected using a power series expansion in z .

$$x = \frac{1}{Bo} \left[E \left(\frac{\phi(z)}{2} \middle| 4Bo \right) - (1 - 2Bo) F \left(\frac{\phi(z)}{2} \middle| 4Bo \right) \right] + \sum_{i=1}^{\infty} a_i z^i \quad (13)$$

Only using the first few terms in the series expansion,

$$a_1 = 0.1349Bo^3 - 0.1208Bo^2 + 0.03847Bo$$

$$a_2 = -0.2681Bo^3 + 0.2502Bo^2 - 0.03327Bo$$

$$a_3 = 0.1462Bo^3 - 0.1244Bo^2 + 0.04438Bo$$

leads to very accurate results. The corrected solution error can be seen in Figure 3. In Figure 3, the error in the corrected solution is very small over the interval above. For the analytical solution, the max error is 8%; in the corrected solution, the max error is 0.05% using the first three terms in the power series.

2.2 Scaling Profile

What is interesting about Equations 12 and 13 is that it is possible to predict the bond number only given a single point. This is not possible with traditional approaches; scaling has to be considered in order to predict the bond number for each point along the profile of a drop. While x and z are already dimensionless in the governing equations, the true observed values are typically scaled after generating a profile to be overlaid on top of the theoretical profile. It is typical that a drop is scaled such that the maximum x value is 1 at the apex. This is a useful mark, because it has been shown that no useful measurements can be made unless a pendant drop is large enough to have a visible apex point. If the drop is scaled as $x = cx'$, $z = cz'$. Where $x'_{max} = 1$ for any arbitrary pendant drop. Then c can be fitted as well for the bond numbers that have been shown experimentally to accurately predict surface tension. The c value can be used to scale the observed x' and z' values to the correct x and z values. The difference between using scaled values and non-scaled values is fairly large, as can be seen in Figure 4. It should be noted that Equation 14 is empirically fitted so that an experimental drop profile can be scaled to accurately measure Bo .

$$c = 1 - (0.09088374064459752 \cdot Bo^2 + 0.1627057021793376 \cdot Bo) \quad (14)$$

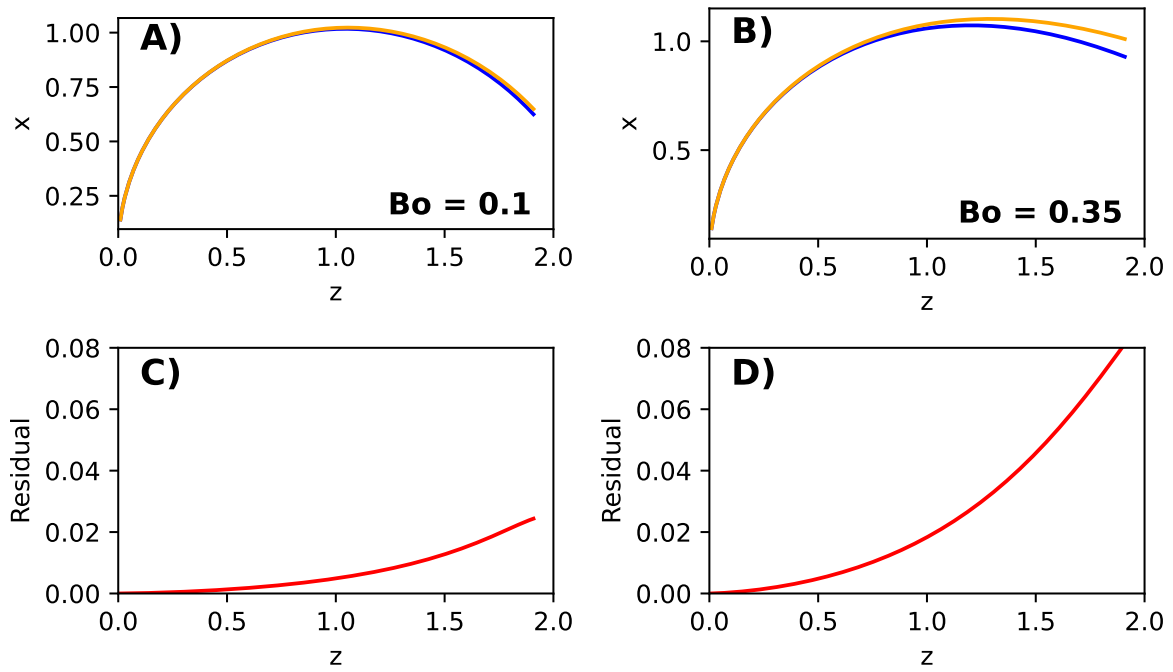


Figure 2: Comparison of the Numerical [34] and Perturbation Solutions to the Young-Laplace equation. A-B. Numerical (orange) and perturbation (blue) solution for different bond numbers. C-D. Residual error as a function of Z and Bo for the same Bo as in A-B.

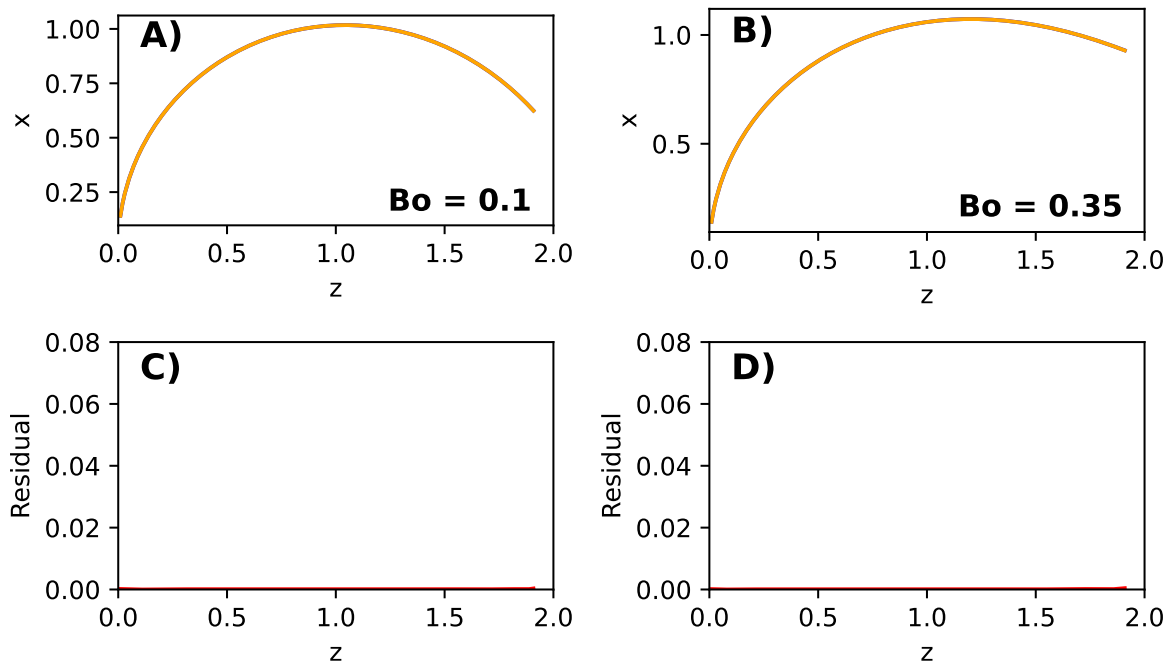


Figure 3: Comparison of the Numerical and Corrected Perturbation Solutions to the Young-Laplace Equation, where the first 3 terms to the power series are shown. A-B. Numerical (orange) and corrected perturbation (blue) solution for different bond numbers. C-D. Residual error as a function of z and Bo for the same Bo as in A-B.

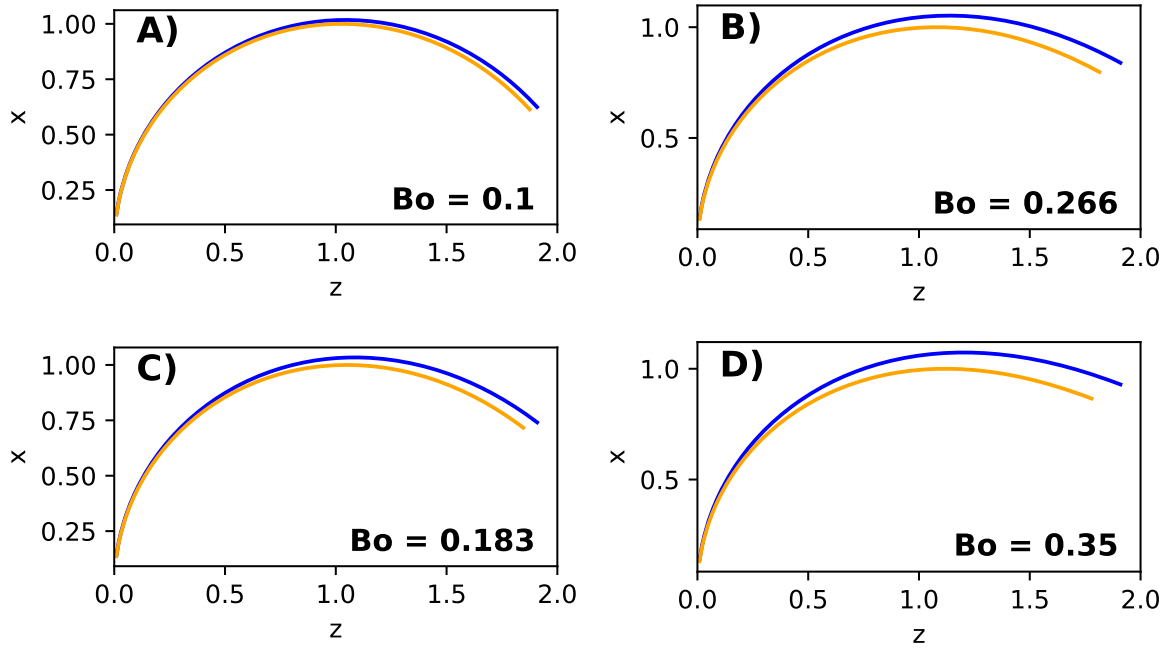


Figure 4: Difference between scaled profiles and non-scaled profiles. The blue line represents non-scaled values that are generated from the numerical solution. The orange line is scaled such that the maximum x value is 1.

3 Deep Learning Approach

There have been attempts to improve ADSA using a machine learning approach. Soori and Tejaswi used a convolutional neural network (CNN) to predict Bo of pendant drops [35]. The model presented was a classification model that was trained on real pendant drop images, where pendant drop images were classified as ranges of surface tension values. This method imposes low accuracy based on the size of the training data set and the precision of measurements. In Kratz and Felix, an artificial neural network (dense neural network) was used to predict Bo and the apex pressure of pendant drops [36]. While this method did improve computational speed greatly with a high accuracy in predicting Bo , it did not account for rotations or noise in the images, nor was it validated on real pendant drop images.

We use a regression CNN trained on computationally generated pendant drop images to determine surface tension from an image directly. Training on computationally generated images has many advantages. Images can have noise added to them, be rotated, the drop within the images can be re-sized and re-positioned, and images can be blurred. This allows for a regression CNN that can accurately predict Bo of drops in a much wider range of images [37]. The benefits to the method presented is faster computational speed, allowing for image rotations, tolerance for low resolution images, low sensitivity to the presence of noise in images, and permitting for images that are blurred. While Bo is the main focus of most ASDA methods, the following parameters can be also be predicted: drop volume, drop surface area, capillary diameter, apex pressure, and the Worthington (Wo) number. As each of these parameters can be calculated given the drop profile, as de-

defined in Table 1. Finally, the method presented is validated on real drop images and has been shown to be effective in predicting fluid properties of real drops.

3.1 Training Convolutional Neural Network (CNN) Models

3.1.1 Numerically Generating Drop Profiles

Pendant drop shapes were simulated using a numerical solver. The only parameters that are needed to generate a drop profile are Bo and Wo . From the drop profile, other geometric and hydrostatic drop properties can be calculated. Table 1 summarizes all calculated and predicted pendant drop geometric and hydrostatic properties.

To develop a training set of pendant drop images, the Young-Laplace equation is used to generate drop profiles. From these profiles, different group of images are generated and for each group, CNN models are trained to predict Bo , V , A , D_n , and R_0 . The most accurate models were further tested on real pendant drop images.

Only one parameter is dependent on an absolute length scale, R_0 . Traditionally, a reference length c is measured and converted to pixel equivalent.

$$R_0 = \frac{R_0}{\text{image width}} = \frac{R_0 \text{ (pix)}}{c \text{ (pix)}} = \frac{R_0 \text{ (m)}}{c \text{ (m)}} \quad (15)$$

Given c , $\Delta\rho$, and g are known, all drop properties can be calculated from the drop shape, via. Bo , V , A , D_n , and R_0 . Therefore, we use a CNN model for each parameter, and the remaining properties can found from arithmetic calculations using the predicted values.

Parameter	Notation	Value
Bond Number	Bo	$\frac{\Delta\rho g R_0^2}{\gamma}$
Dimensionless Drop Volume	V	$V_d \frac{1}{\pi D_n R_0^2}$
Dimensionless Drop Area	A	$A_d \frac{1}{\pi D_n R_0}$
Dimensionless Capillary Diameter	D	$D_n \frac{1}{R_0}$
Dimensionless Drop Radius	R_0	$R_0 \frac{1}{c}$
Worthington Number	Wo	BoV
Apex Pressure	P_a	$2 \frac{\gamma}{R_0}$
Surface Area to Volume Ratio	SA/V	$\frac{A}{V} \frac{1}{R_0}$

Table 1: Prediction of Fluid and Geometric Properties from a Pendant Drop, given g , $\Delta\rho$, and c . Where D_n is the diameter of the capillary tip which the drop is suspended from, A_d is the surface area of the drop, and V_d is the volume of the drop.

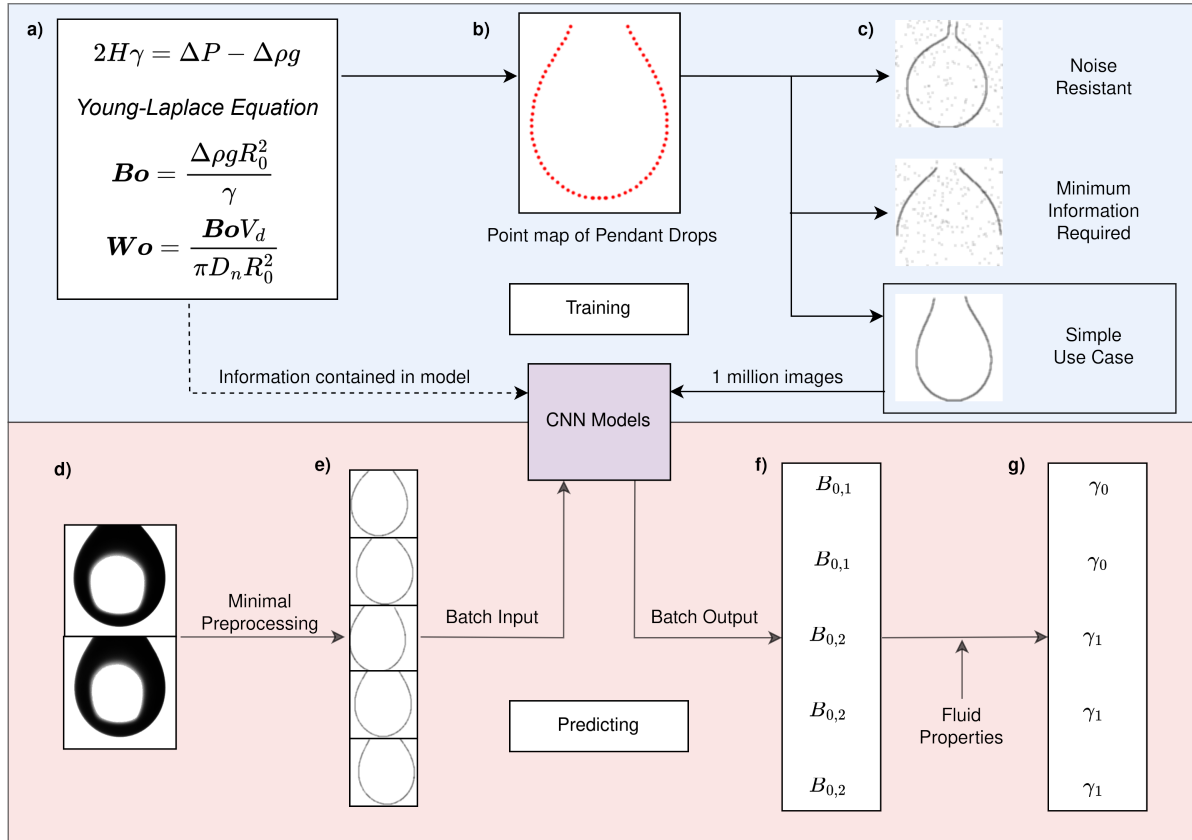


Figure 5: Workflow for using a CNN to Predict Pendant Drop Surface Tension. a) Young-Laplace equations are used to generate drop profile. The Bo and Wo are varied for each drop to generated parent families drop shapes. b) Representative drop shape c) Representative image groups used to train the CNN models d) Real drop images as inputs to the model e) Image preprocessing, where each drop is rotated and shifted randomly multiple times. f) CNN predictions. Where the same prediction is made for each image that came from the same starting drop. Here, Bo is predicted, but can be Bo, V, A, D_n , or R_0 . g) Final predicted fluid, geometric or hydrostatic properties

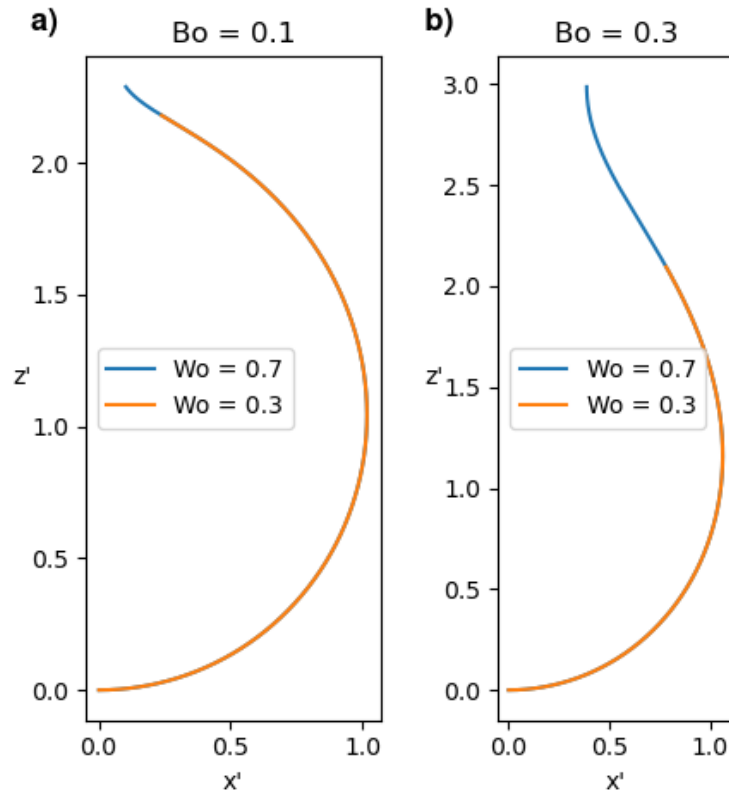


Figure 6: Affect of Bo and Wo on numerically generated drop profiles. a) For small Bo drops large changes in Wo do not lead to large changes in drop volume or drop area which is contrasted to large Bo drops in b).

Figure 5 highlights the overall workflow. Figures 5a through 5c represent the parts of the workflow that were used for training the CNN models. Figures 5d through 5g represent the elements of the workflow used to make fluid property predictions, as well as test the models. In Figure 5a, the Young-Laplace equation is used to generate drop profiles. The main parameter Bo , which determines the shape of the drop and Wo constrains the drop volume. Wo is defined as the drop volume divided by the maximum drop volume, which can be rewritten in terms of Bo and the dimensionless drop volume as seen in Table 1. [9]

In Figure 5b, given Bo and Wo , a drop profile is generated. To generate a given drop profile, we chose $0.1 < Bo < 0.35$ and $0.35 < Wo < 1$. In this particular workflow, `scipy.integrate.odeint` (SIO) was used to generate drop profiles. SIO uses `lsoda` (Livermore Solver for Ordinary Differential equations) from the FORTRAN library `odepack` [38]. Profiles were generated for a given Bo . If the target Wo number is less than the calculated Wo , the drop volume is reduced until the target is reached. If the target Wo is greater than the calculated, the target is updated to be equal to the calculated Wo . The rest of the dimensionless parameters are then calculated from the drop profile, except for the dimensionless drop radius. The dimensionless drop radius is calculated after all modifications to the drop profile have been made. The affect of Bo and Wo on the drop profile can be seen in Figure 6.

3.1.2 Numerically Generating Drop Images

In Figure 5c, some example generated images are shown. While there are only two parameters to generate a drop profile, there are many parameters to generate image groups given the basis family of drop profiles. Table 2 shows the different

image groups that we used for representing challenges to optical alignment and camera focus. For a given image group, a CNN model was trained and tested for each property: Bo , V , A , D_n , and R_0 . This section of the workflow is to test which groups images are valid to predict the fluid properties of real drops.

Table 2 represents the different image groups that were used. All image groups used the same basis family of 100,000 drop profiles. Each image in an image group is drawn using a randomly selected drop profile. To test model accuracy for low resolution images, all images have a pixel resolution of 128x128 pixels. The drops in the images were rotated between -10 and 10 degrees, to show that the model is robust in predicted drop properties on rotated drops. The drops were also resized within the image, with a relative size range. A relative size of 1 represents a drop that has a maximum length (maximum of either maximum x or maximum z of the drop) equal to the image width, and 0.95 represents a maximum length scale of 95% of the image width.

Drops were randomly positioned within each frame on the image canvas. The relative size and random position are important to ensure the model does not overfit to the location and position of the drops. Images were defocused using a Gaussian blurring algorithm after generation, which is standard practice for all CNN image analysis models [39]. The global parameters were chosen because regardless of image pre-processing tuning, each model should be robust with low quality images, rotated drops, and random placement of drops in the image. Each group further modifies the images. Example images for each group are shown in Figure 7. The details of each image parameter are further discussed.

Group	Capillary	Capillary Zero	Above Apex	Noise	Test B_o MAE
Group A (Control)	[0, 0]	1	False	0	1.69E-03
Group B	[0, 0]	1	False	0.01	1.83E-03
Group C	[0, 1.5]	0.15	False	0	3.01E-03
Group D	[0, 0]	1	True	0	2.43E-03
Group E	[0, 1.5]	0.15	False	0.01	3.39E-03
Group F	[0, 0]	1	True	0	2.19E-03
Group G	[0, 1.5]	0.15	True	0	2.35E-03
Group H	[0, 1.5]	0.15	True	0.01	3.35E-03

Table 2: Summary of Image Groups for CNN Model Training. Parameters for Generating Drop Images. The test B_o MAE is the mean absolute error of the predicted B_o of the test set.

In most ADSA methods, the capillary tip from the image is removed from the image either manually or through image pre-processing. However, a CNN model is able to infer the relevant drop profile, even if a capillary tip is present. The category Capillary represents the range that the dimensionless capillary length can be, where the capillary length is made dimensionless by the drop radius. For each drop image, a dimensionless capillary length is randomly chosen that is within the range. The capillary length is then added onto the drop profile prior to drawing the profile onto an image. The category Capillary Zero represents the finite experimental probability that the capillary is not presented in the measured image. Capillary Zero is used to ensure that the models do not overfit to the capillary tip being present.

Although this workflow focuses on analysis of pendant drop images, there are many different axis-symmetric drop shapes that can be drawn from Equations 2-4. The only difference between the various axis-symmetric drop shapes are the boundary conditions, and the sign of $B\sigma$. Our CNN model can predict fluid properties on a variety of different drop shapes. The category Above Apex means all drop profiles have the part of the drop below the apex removed prior to drawing the drop onto an image, where the cutoff point is defined where $x' = x'_{\max}$. It should be noted that none of the fluid properties were re-calculated for groups that have Above Apex as True, which demonstrates that the entire profile is not necessary to accurately predict fluid properties.

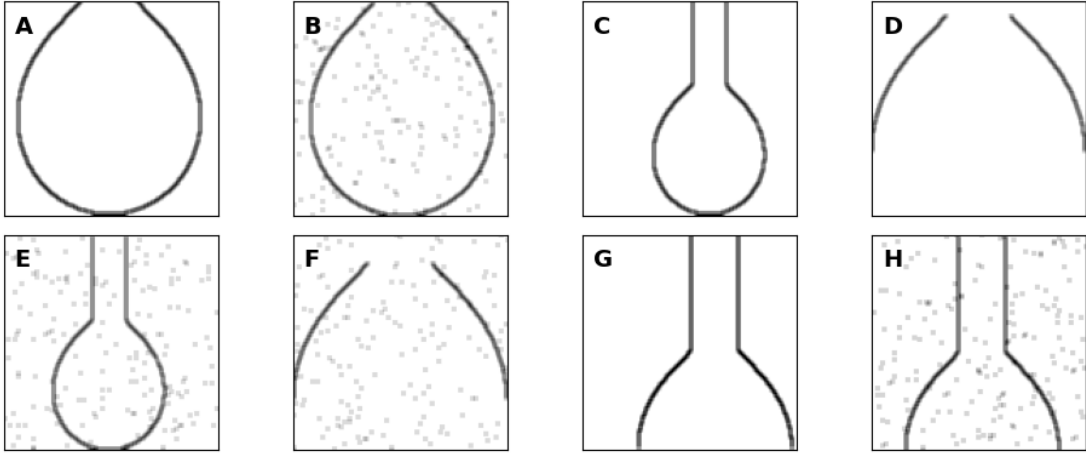


Figure 7: Representative Samples from Image Group Training Sets. B_o and W_o are fixed for each image. Each label corresponds to Groups in Table 2.

Image Generation. After a drop profile is modified according to its respective image group in Table 2, it is then placed onto a blank palette. All modifications that are made to the randomly chosen drop profile prior to drawing the drop profile to an image are: 1) adding a capillary tip, 2) cutting of the apex, 3) rotation of drop shape about z-axis, and 4) in-plane rotation with respect to the z-axis. All images in the final model have a white background, and the modified drop profiles are drawn into the image using black pixels. All drop images are blurred as a final step.

Noise. Although blurring is important for all CNN models, blurring does not account for all potential noise in the image. The Noise category represents the average percentage of pixels that are forced to be black prior to blurring. For an image that has 128x128 pixels, 0.01% noise represents on average 163 random pixels are forced to black. If a pixel is already black from the drop profile, it is left black. While 163 pixels is apparently small, 0.01% Noise adds a fairly significantly amount of image disruption as seen in Figure 7.

Calculating R_0 in Generated Images. All the modifications to the drop profile are completed prior to locating the drop profile onto the image canvas. The equation to calculate the dimensionless radius for each drop from the modified drop profile is:

$$R_0 = \frac{R_x}{L_M} S_D \cos \theta \quad (16)$$

where R_x is the maximum x value in the unmodified drop profile, θ is the rotation of the image, L_M is the maximum length scale value ($\max[\max x, \max z]$) after rotation and adding capillary tip, S_D is the relative size of the entire modified drop profile that is between [0.95, 1].

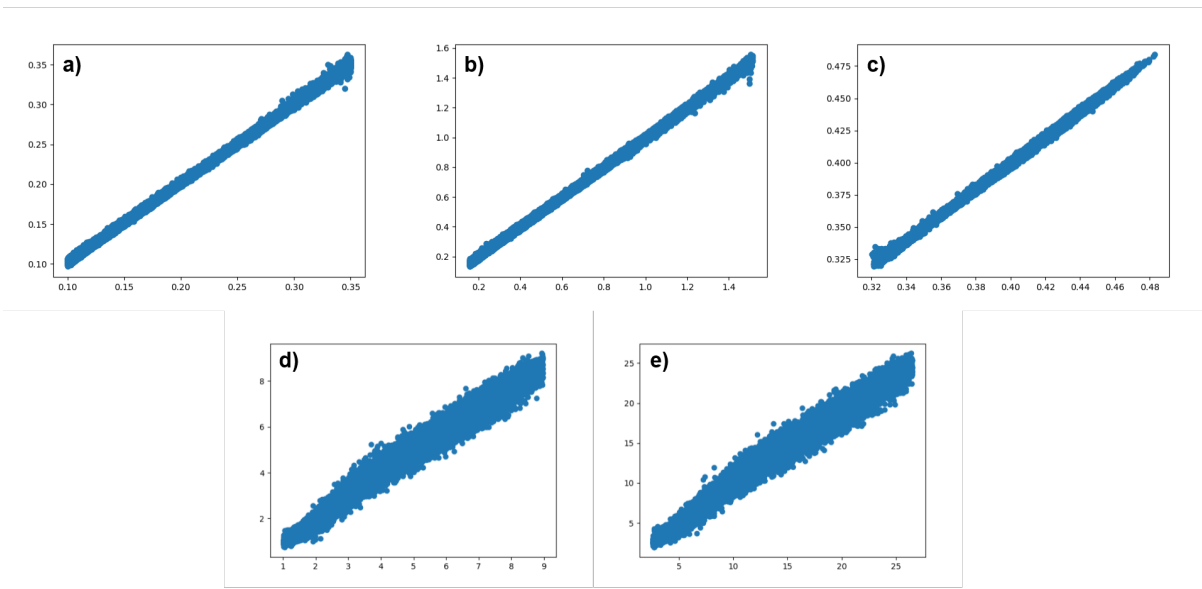


Figure 8: Machine Learning Regression Metrics. Predicted versus actual graphs for the different drop properties that were predicted from the control group, which gives a visual representation of the prediction error. The values shown are dimensionless, but are not normalized. The x-axis is the actual value, and the y-axis is the predicted value. The variables in each predicted vs actual graph are a) Bo , b) D_n , c) R_0 , d) V , and e) A .

The relative size can be rewritten as:

$$S_D = \frac{L_M}{c} \quad (17)$$

which shows that Equation 16 is equivalent to Equation 15, after accounting for the fact that the drop can be rotated.

3.1.3 Training CNN Models and Accuracy in Generated Data

Training and Accuracy. Each model was trained on the same model architecture, with one output neuron predicting either Bo , V , A , D_n , or R_0 . For each group in Table 2, we trained and tested the CNN using computationally generated data with 100,000 drop profiles and 1,000,000 drop images generated in each group. The images were shuffled and used a data split of training 80%, validation 10%, and testing 10%. Keras was used to train CNN models, with a batch size of 256 and 10 epochs. Each CNN model in each group was trained on the same split [40]. The results for MAE of the predicted surface tension can be found in Table 2; Figure 8 shows the accuracy in predicting all the normalized fluid properties.

Accuracy of Surface Tension. The most important predicted parameter Bo represents the dimensionless surface tension. The test Bo MAE in Table 2 is the mean average error of the predicted Bo of the test set for each group. It can be seen that the control group has the lowest prediction error. The reported MAE can also be used to calculate the theoretical accuracy in predicting the surface tension of fluids. Assuming there is no error in any other fluid property measurement, the error of the predicted surface tension can be estimated to be $\gamma_{\text{error}} = \gamma_{\text{true}} \cdot Bo_{\text{MAE}}$.

Group	Bo_m	V_m	A_m	D_m	R_m
Control	6.76E-03	1.50E-02	1.55E-02	5.09E-03	4.46E-03
Noise	7.33E-03	1.67E-02	1.79E-02	6.46E-03	6.05E-03
Capillary	1.20E-02	2.15E-02	2.43E-02	5.69E-03	6.96E-03
Top	9.73E-03	1.71E-02	1.65E-02	5.69E-03	2.38E-02
Noise Capillary	1.36E-02	2.34E-02	2.54E-02	9.14E-03	4.43E-03
Noise Top	8.76E-03	1.63E-02	1.57E-02	6.83E-03	2.41E-02
Capillary Top	9.41E-03	1.82E-02	1.72E-02	6.10E-03	5.13E-03
Noise Capillary Top	1.34E-02	2.01E-02	2.08E-02	9.07E-03	6.20E-03

Table 3: Table of performance metrics to evaluate how each CNN model in each group performed in predicting fluid properties. Subscript m represents the mean absolute error, MAE, for the normalized predicted properties of the test set for the numerically generated drops in the group. Where the data has been normalized using min-max scaling. R represents the drop radius, typically denoted as R_0 .

For pure water (72.4 mN/m), the theoretical average error in the predicted surface tension is ± 0.122 mN/m using the control *Bo* CNN model. This shows that the CNN model matches state of the art measurements on the best model. However, even using the model with the lowest accuracy, Group E, leads to an average error in the predicted surface tension of ± 0.245 mN/m. This underscores that all CNN models closely match state of the art measurements even on highly distorted images. More importantly, the relationship for error prediction is biased at larger values of surface tension. Therefore, image analysis on pendant drops of lower surface tension will display lower error than those reported here.

Other Metrics. While *Bo* is the most important prediction, model performance in the prediction of geometric and hydrostatic properties is also important. The control model performs the best, or close to the best, on all the parameters. The accuracy of predicting *Bo*, *D_n*, or *R₀* for pendant drops is very accurate for all the models, with an min-max normalized MAE error range of $4.46 \cdot 10^{-3} < E(\mathbf{Bo}, \mathbf{D}_n, \mathbf{R}_0) < 6.96 \cdot 10^{-3}$. The predictions for *V* and *A* are also accurate, but less accurate than the other predicted properties with an min-max normalized MAE error range of $1.50 \cdot 10^{-2} < E(\mathbf{V}, \mathbf{A}) < 2.48 \cdot 10^{-2}$. A complete breakdown of all the prediction metrics can be found in Table 3. In Figure 6 it can be seen that for low *Bo* drops, large changes in *Wo* do not lead to large changes in the drop profile. For high *Bo* drops, changes in *Wo* lead to much larger changes in the drop profile. This implies that the model can more accurately predict *V* and *A* as *Bo* increases. Figure 8 shows that the theoretical error for *V* and *A* decrease as *V* and *A* decrease as *Wo* increases.

3.2 Testing Final Models on Experimental Pendant Drop Images

Data Analysis on Experimental Observations of Pendant Drops. The experimental section shows that the CNN models are very accurate at predicting the fluid properties of computationally generated drops. Here, we validate these results on experimental observations of pendant drops. Pendant drop images were measured using a liquids of pure water and a solution of 75 m% Ethanol and 25 m% DI Water. These liquids have known surface tensions at 20 C of 72.7 mN/m and 35.3 mN/m respectively. The cameras used were Flea3 CMOS imaging cameras from Point Grey (Richmond, BC, Canada), equipped with either a Sony IMX035 CMOS, 1/3" sensor or On Semi VITA1300 CMOS, 1/2" sensor. The lens used was a C-mount CCTV lens, F=1.4, utilising a macro tube to provide the necessary zoom level. Backlighting was from a white LED (CREE C503C-WAS) diffused with a ground-glass disc. The optical setup was tested extensively for image aberrations by imaging a number of objects of known geometry at different orientations.

Predicting True Bo . The final CNN models are compared against traditional ASDA. To show that the predicted Bo from traditional ASDA is accurate, the numerical profile from the predicted Bo is overlaid on top of the true extracted profile. The comparison between the true drop profile and numerical profile from the predicted Bo using traditional ASDA can be seen in Figure 9b. Figure 9b, shows that the calculated Bo from traditional ASDA is accurate.

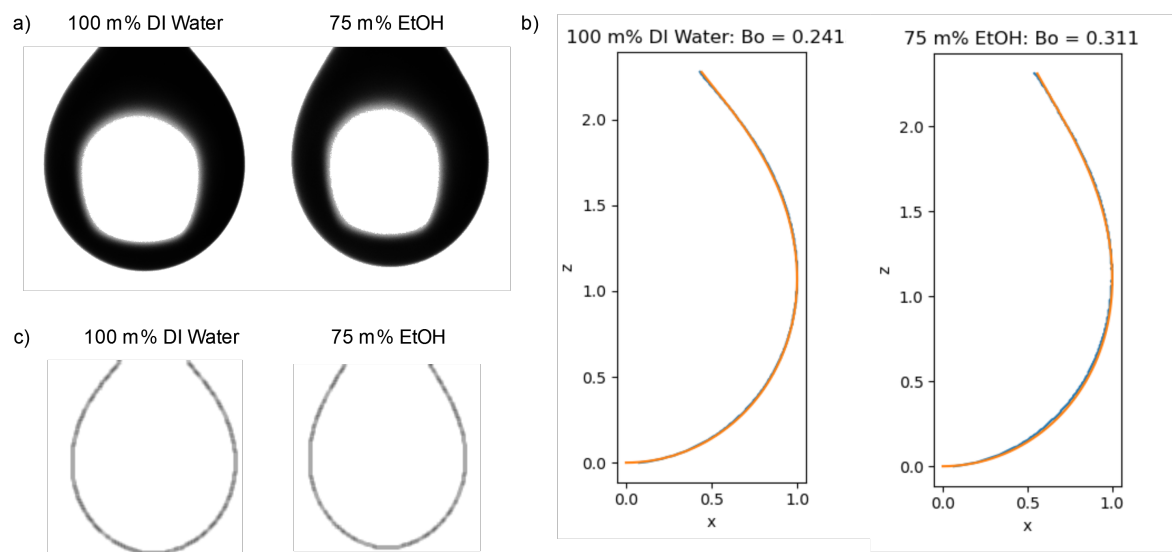


Figure 9: Experimentally calculating Bo using traditional method. A) Pendant drop images of pure DI water and a solution of 75 m% Ethanol and 25 m% DI Water. B) Comparison of true drop profile versus numerically generated drop profile from traditional ASDA. The blue line is the true drop profile, and the orange line is numerically generated from the predicted Bo using traditional ASDA. C) Images that have been preprocessed by first extracting the profiles, then generating the final images using the same pipeline as the numerically generated profiles.

Property	DI Water		75 m% Ethanol	
	True	Predicted	True	Predicted
<i>Bo</i>	0.241	$0.243 \pm 2.10\text{E-}3$	0.311	$0.306 \pm 3.60\text{E-}3$
γ (mN/m)	72.7	72.0	35.3	35.9
<i>V</i>	1.85	1.76 ± 0.135	1.51	$1.59 \pm 9.60\text{E-}2$
<i>A</i>	5.59	4.89 ± 0.294	4.42	4.30 ± 0.412
<i>D</i>	0.820	$0.825 \pm 7.50\text{E-}3$	1.05	$1.06 \pm 5.90\text{E-}3$

Table 4: Table of predicted versus actual fluid properties. Predicted properties are reported as mean \pm standard deviation of predictions. The predicted surface tension is calculated as $\gamma_{\text{predicted}} = \gamma_{\text{true}} \cdot Bo_t/Bo_p$. ***R*₀** is not reported, as it is dependant on rotation and drop scale, which is not constant for all images.

Image Pre-processing. In an experimental setup, image pre-processing can be one of the most challenging steps. In the case of using a CNN, it is very important that the pre-processed images are as similar to the training data as possible. The experimental pendant drop images were tested on the models in the control group. Images were pre-processed by edge detecting the drop profile, then generating images using the same pipeline as the control group images. For each image, 5 extra images were generated by rotating the pre-processed images randomly between $[-5, 5]$ degrees. Pre-processed images can be seen in Figure 9c.

Fluid Property Predictions. From the preprocessed images, the predicted versus actual fluid properties can be found in Table 4. It can be seen that the model is very accurate at predicting the fluid properties of both pure DI water and the high concentration ethanol solution. Most notably, it can be seen that the error in the measured surface tension between traditional ASDA and using CNN model is only ± 0.690 mN/m. This accuracy is achieved with image resolutions of only 128×128 pixels and rotating the images. The other fluid properties were also accurately predicted. The errors for the other predicted fluid properties for the concentrated ethanol are $E(V) = \pm 8.00 \cdot 10^{-2}$, $E(A) = \pm 0.120$, and $E(D_n) = \pm 5.00 \cdot 10^{-3}$. R_0 is not reported, as it is dependant on rotation, which is not constant for all the images.

Computation Time The method presented in this article has been shown to be as accurate as traditional ADSA methods on distorted and low quality images. The computation time needed to make fluid predictions using a CNN model is significantly faster as well. To compare computational speed, the CNN is timed against a traditional ADSA approach to predict Bo of 10^6 drops. For the CNN model, all 10^6 of the generated training images were used. For the traditional method, 10^6 drop profiles were chosen from the generated point profiles. This was done to avoid

comparing image pre-processing times, and strictly compare the time needed to predict \mathbf{Bo} of each method. The computer specifications that were used for both test are: Ubuntu 20.04.5 LTS, Intel i9-9960X CPU, and Quadro RTX 6000 GPU.

For the traditional method, all calculations were parallelized on 32 threads. From a predicted \mathbf{Bo} , a point map is generated by solving the differential equations as outlined earlier ($\Delta s = 10^{-3}$, $W = 1$). A spline function is then fitted using `scipy.interpolate.UnivariateSpline` with zero smoothing. The spline defines a function $x_p = f(z_p)$ for the predicted \mathbf{Bo} . Then the error is defined as $E(\mathbf{Bo}) = \sum_{i=0}^N |x - f(z)|$ where the true z points are feed into the fitted function and compared against the true x points. The true value for \mathbf{Bo} is where $E(\mathbf{Bo})$ is minimized. The error function was minimized using `scipy.optimize.minimize` using the L-BFGS-B method, maximum iterations of 100, and bounds of (0.1, 0.35).

It takes 35 hours to predict 10^6 drop profiles using traditional ADSA, which is 0.13 seconds per images. Using a CNN model, it only takes 11 minutes to predict 10^6 images, which is 0.66 milliseconds per image. Even trying to minimize the time need for traditional ADSA, using a CNN model is order of magnitudes faster than using a traditional ADSA approach.

4 Conclusion

In summary, we have provided a novel analytical solution to the Young-Laplace equation for an axisymmetric pendant drop, which is the common test frame for surface tension measurement. The perturbation approach employed and the correction improve the solution accuracy, making it a valuable tool for various fields. The analytical solution presented not only enhances computational efficiency but also contributes to a better comprehension of the underlying physics.

In addition we describe a new method for Axisymmetric Drop Shape Analysis (ADSA) using a trained convolutional neural network (CNN) model. The new method offers faster computational speed, tolerates lower image resolution, and has low sensitivity to noise in images, making it a more cost-effective alternative to traditional ADSA techniques. The CNN model achieved a theoretical error of 0.122 mN/m in predicting the surface tension of computationally generated drops and demonstrated an error of only 0.690 mN/m in predicting the surface tension of real experimental images. Additionally, the final CNN models can predict fluid properties at a rate of 1,500 images per second. This allows for a single central system to monitor the fluid properties of many drop simultaneously in real time, reducing cost significantly.

The use of computationally generated images for training the CNN model provides several advantages, allowing the model to account for rotations or noise in the images. However, there are still limitations in the method presented, including that the drop image must be undistorted by lensing. Overall, the proposed ADSA method using a trained CNN model offers significant improvements over traditional ADSA techniques and existing improvements. It has the potential to impact

various fields, including a wide range of chemical and materials manufacturing and consumer products process, where accurate surface and interfacial tension measurements are critical quality attributes.

A Code for Deep Learning Approach

A.1 Imported Packages

```
from pathlib import Path
from os import mkdir, listdir
from os.path import exists
from shutil import rmtree
from json import load, dump
from time import sleep
from random import shuffle, uniform, choice, randrange
from math import pi, sin, cos, radians

from tqdm import tqdm
from cv2 import imread, cvtColor, COLOR_BGR2GRAY, blur
from cv2 import threshold, THRESH_BINARY, findContours, RETR_EXTERNAL
from cv2 import CHAIN_APPROX_NONE, imwrite
from numpy import array, expand_dims, arange, concatenate, dot, where
from numpy import unique, zeros, sin, cos, argmin, argmax, savetxt, loadtxt
from numpy.random import rand
from sklearn.preprocessing import MinMaxScaler
```



```

from keras.utils import Sequence
from keras.models import Sequential, load_model
from keras.callbacks import Callback
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout
from joblib import load as joblib_load, dump as joblib_dump
from scipy.integrate import odeint
from matplotlib import pyplot as plt, use

```

A.2 Backend Functions

Cast a list of parameters in a dict to a writable file name

```

def parameters_to_name(parameters, suffix):
    name = ""
    for key, value in parameters.items():
        name += f"${key}={round(value, 6)}"
    if name:
        name = name[1:] + suffix
    return name

```

Inverse of parameters_to_name

```

def name_to_parameters(name):
    parameters = dict()
    for key_value in name.stem.split("$"):
        key, value = key_value.split("=")
        parameters[key] = float(value)
    return parameters

```

```

def name_to_y(file, y_key):
    file = Path(file)
    parameters = name_to_parameters(file)
    return parameters[y_key]

def generate_y_scaler(train_files, y_key):

    all_y = []
    for file in train_files:
        all_y.append(name_to_y(file=file, y_key=y_key))

    y_scaler = MinMaxScaler()
    y_scaler.fit_transform(array(all_y).reshape(-1, 1))

    return y_scaler

def img_file_to_img(file):
    file = Path(file)
    x_i = imread(str(file))
    x_i = cvtColor(x_i, COLOR_BGR2GRAY)
    x_i = 255 - x_i
    x_i = x_i / 255
    x_i = expand_dims(x_i, axis=2)
    return x_i

class DataGenerator(Sequence):

```

```

def __init__(self, files, y_scaler, y_key, batch_size):
    self.files = files
    self.y_scaler = y_scaler
    self.y_key = y_key
    self.batch_size = batch_size

    self.len_files = len(self.files)

def on_epoch_end(self):
    shuffle(self.files)

def __len__(self):
    return self.len_files // self.batch_size

def __getitem__(self, index):

    max_index = self.batch_size * (index + 1)
    min_index = self.batch_size * index
    batch_files = self.files[min_index:max_index]

    x = []
    y = []
    for file in batch_files:
        x.append(img_file_to_img(file))
        y.append(name_to_y(file=file, y_key=self.y_key))

```

```

x = array(x)

y = self.y_scaler.transform(array(y).reshape(-1, 1))

return x, y

```

A.3 Numerical Drop Profile Creation

```

def new_drop_profile(bond_number, max_worthington_number, delta_s):

    # Calculate baseline drop profile

    def pendant_drop(y, s):

        x, z, phi = y

        d_x = cos(phi)

        d_z = sin(phi)

        d_phi = 2 - bond_number * z - sin(phi) / x

        return d_x, d_z, d_phi

    x0, z0, phi0 = delta_s, 0, delta_s

    s_range = arange(delta_s, 4, delta_s)

    sol = odeint(pendant_drop, (x0, z0, phi0), s_range)

    # Ensure there are only two inflection points

    apex_index = argmax(sol[:, 0], axis=0)

    max_index = argmin(sol[apex_index:, 0], axis=0)

    sol = sol[:max_index + apex_index, :]

```

```

# Calculate drop parameter

volume = 0
area = 0
max_drop_radius = 0
worthington_number = 0
upper_index = 0

for i in range(len(sol)):
    upper_index = i
    row = sol[i]
    x, z, phi = row

    volume += pi * (x ** 2) * sin(phi) * delta_s
    area += 2 * pi * x * delta_s
    cap_diameter = 2 * x

    if x > max_drop_radius:
        max_drop_radius = x

    worthington_number = bond_number * (volume / (pi * cap_diameter *
                                                    (max_drop_radius ** 2)))

    if worthington_number > max_worthington_number:
        break

```

```

# Cutoff drop where max_worthington_number = worthington_number,
#   or at the max height
sol = sol[:upper_index, :]

# Only keep x, z data
sol = sol[:, :2]

# Normalize parameters
drop_parameters = dict(
    volume = volume / (pi * cap_diameter * (max_drop_radius ** 2)),
    area = area / (pi * cap_diameter * max_drop_radius),
    cap_diameter = cap_diameter / max_drop_radius,
    worthington_number = worthington_number,
    bond_number = bond_number,
)

return drop_parameters, sol

```

A.4 Numerical Drop Image Creation

```

def new_drop_image(drop_profile, img_size_pix, rotation, drop_scale, noise,
                    rel_capillary_height, above_apex, delta_s):

# Grab drop radius

```

```

relative_drop_radius = drop_profile[:, 0].max()

# Add capillary tip
if rel_capillary_height != 0:
    max_z_index = drop_profile[:, 1].argmax()
    xf, zf = drop_profile[max_z_index, :]
    z_new = arange(zf, zf + relative_drop_radius * rel_capillary_height,
                    delta_s)
    x_new = array([xf] * len(z_new))
    cap = array(list(zip(x_new, z_new)))
    drop_profile = concatenate((drop_profile, cap))

# Cutoff drop below apex
if above_apex:
    max_x_index = drop_profile[:, 0].argmax()
    drop_profile = drop_profile[max_x_index:, :]

# Mirror drop
drop_z = drop_profile[:, 1]
mirror_x = - drop_profile[:, 0]
mirrored_drop_profile = array(list(zip(mirror_x, drop_z)))
drop_profile = concatenate((drop_profile, mirrored_drop_profile))

# Rotate drop
rotation = radians(rotation)

```

```

rotation_matrix = array([[cos(rotation), sin(rotation)],
                          [-sin(rotation), cos(rotation)]])

drop_profile = dot(drop_profile, rotation_matrix.T)
relative_drop_radius = relative_drop_radius * cos(rotation)

# Scale drop to image size
drop_profile[:, 0] = drop_profile[:, 0] - drop_profile[:, 0].min()
drop_profile[:, 1] = drop_profile[:, 1] - drop_profile[:, 1].min()
max_length_scale = max(drop_profile.max(axis=0))
scaler_factor = img_size_pix * drop_scale / max_length_scale
drop_profile = drop_profile * scaler_factor
relative_drop_radius = relative_drop_radius * scaler_factor / img_size_pix

# Drop duplicate rows
drop_profile = drop_profile.astype(int)
drop_profile = unique(drop_profile, axis=0)

# Flip image axis
drop_profile[:, 1] = - drop_profile[:, 1]
drop_profile[:, 1] = drop_profile[:, 1] - drop_profile[:, 1].min()

# Shift drop in image
x_shift = randrange(0, img_size_pix - drop_profile[:, 0].max())
z_shift = randrange(0, img_size_pix - drop_profile[:, 1].max())
drop_profile[:, 0] = drop_profile[:, 0] + x_shift

```



```

drop_profile[:, 1] = drop_profile[:, 1] + z_shift

# Draw drop points onto image
drop_image = zeros((img_size_pix, img_size_pix))
for i in range(len(drop_profile)):
    x, z = drop_profile[i, :]
    drop_image[z, x] = 255

# Add salt and pepper noise
if noise != 0:
    rand_matrix = rand(img_size_pix, img_size_pix)
    rand_matrix = where(rand_matrix < noise, 1, 0)
    rand_matrix = rand_matrix * 255
    drop_image = drop_image + rand_matrix
    drop_image = where(drop_image > 255, 255, drop_image)

# Blur image
drop_image = blur(drop_image, (3, 3))

# Invert image
drop_image = 255 - drop_image

return relative_drop_radius, drop_image

```

A.5 CNN Model Architecture

```
def simple_model():  
    # Same model structure that all models use  
    model = Sequential()  
    model.add(Conv2D(16, kernel_size=3, activation='relu'))  
    model.add(MaxPool2D(pool_size=2))  
    model.add(Conv2D(32, kernel_size=3, activation='relu'))  
    model.add(MaxPool2D(pool_size=2))  
    model.add(Flatten())  
    model.add(Dense(64))  
    model.add(Dropout(0.2))  
    model.add(Dense(1))  
    model.compile(optimizer='Adam', loss='mae', metrics=['mse', 'mae'])  
    return model
```

A.6 Training and Testing Models

```
def train(split, y_key, epochs, batch_size):  
  
    y_scaler = generate_y_scaler(train_files=split["train"], y_key=y_key)  
  
    train_generator = DataGenerator(files=split["train"], y_scaler=y_scaler,  
                                    y_key=y_key, batch_size=batch_size)  
    val_generator = DataGenerator(files=split["val"], y_scaler=y_scaler,  
                                   y_key=y_key, batch_size=batch_size)
```

```

model = simple_model()

history = model.fit(train_generator, epochs=epochs,
                    validation_data=val_generator,
                    callbacks=Callback())

# Only keep loss and val_loss from history
history_data = dict(loss=history.history['loss'],
                    val_loss=history.history['val_loss'])

return model, history_data, y_scaler

def test(model, y_key, y_scaler, y_files, batch_size):

    def _predict(b):

        # Gather x and y data in files
        x = []
        y = []
        for file in b:
            x.append(img_file_to_img(file))
            y.append(name_to_y(file=file, y_key=y_key))
        x = array(x)

        # Normalize y data
        y_norm = y_scaler.transform(array(y).reshape(-1, 1))

```

```

# Predict property

y_pred_norm = model.predict(x, verbose=0)
y_pred = y_scaler.inverse_transform(y_pred_norm)

# Flatten y data

y_norm = y_norm.flatten()
y_pred_norm = y_pred_norm.flatten()
y_pred = y_pred.flatten()

# Update PVA data

for i, y_norm_i in enumerate(y_norm):
    y_pred_norm_i = y_pred_norm[i]
    pva_norm.append([y_norm_i, y_pred_norm_i])
for i, y_i in enumerate(y):
    y_pred_i = y_pred[i]
    pva.append([y_i, y_pred_i])

pva = []
pva_norm = []

# Predict on files in batches

batch = []
for file in tqdm(y_files, desc=f"Predicting {y_key}"):
    batch.append(file)

```

```

    if len(batch) == batch_size:
        _predict(batch)
        batch = []
    if batch:
        _predict(batch)

    return array(pva_norm), array(pva)

```

A.7 Configuration File

```

c = {
    "directory": "data",
    "drop_profiles": {
        "bond_number_range": [0.1, 0.35],
        "min_worthington_number": 0.35,
        "delta_s": 1e-3,
        "N": 1e5
    },
    "image_parameters": {
        "img_size_pix": 128,
        "rotation_range_degrees": [-10, 10],
        "drop_scale_range": [0.95, 1],
        "N": 1e6
    },
    "model_parameters": {

```

```

    "epochs": 10,
    "batch_size": 256,
    "train_percent": 0.8,
    "val_percent": 0.1
},
"models" : [
    {
        "name": "control",
        "noise": 0,
        "rel_capillary_height_range": [0, 0],
        "prob_capillary_zero": 1,
        "only_keep_drop_above_apex": False
    },
    {
        "name": "noise",
        "noise": 0.01,
        "rel_capillary_height_range": [0, 0],
        "prob_capillary_zero": 1,
        "only_keep_drop_above_apex": False
    },
    {
        "name": "capillary",
        "noise": 0,
        "rel_capillary_height_range": [0, 1.5],
        "prob_capillary_zero": 0.15,

```

```

    "only_keep_drop_above_apex": False
},
{
    "name": "top",
    "noise": 0,
    "rel_capillary_height_range": [0, 0],
    "prob_capillary_zero": 1,
    "only_keep_drop_above_apex": True
},
{
    "name": "noise_capillary",
    "noise": 0.01,
    "rel_capillary_height_range": [0, 1.5],
    "prob_capillary_zero": 0.15,
    "only_keep_drop_above_apex": False
},
{
    "name": "noise_top",
    "noise": 0.01,
    "rel_capillary_height_range": [0, 0],
    "prob_capillary_zero": 1,
    "only_keep_drop_above_apex": True
},
{
    "name": "capillary_top",

```

```

        "noise": 0,
        "rel_capillary_height_range": [0, 1.5],
        "prob_capillary_zero": 0.15,
        "only_keep_drop_above_apex": True
    },
    {
        "name": "noise_capillary_top",
        "noise": 0.01,
        "rel_capillary_height_range": [0, 1.5],
        "prob_capillary_zero": 0.15,
        "only_keep_drop_above_apex": True
    }
]
}

```

A.8 Wrapper to Generate Data

```

def generate_data(c):

    # Make new directory to store all data
    root_directory = Path(c["directory"])
    mkdir(root_directory)

    # Generate new drop profiles
    drop_profile_directory = root_directory / "drop_profiles"

```



```

mkdir(drop_profile_directory)

for _ in tqdm(range(int(c["drop_profiles"]["N"]))):
    desc="Generating Drop Profiles"):
        bo = uniform(*c["drop_profiles"]["bond_number_range"])
        wo = uniform(c["drop_profiles"]["min_worthington_number"], 1)
        delta_s = c["drop_profiles"]["delta_s"]
        drop_parameters, drop_profile = new_drop_profile(bond_number=bo,
                                                         max_worthington_number=wo,
                                                         delta_s=delta_s)
        file = parameters_to_name(drop_parameters, suffix=".csv")
        savetxt(drop_profile_directory / file, drop_profile, delimiter=",")

# Create images for models

drop_profiles_paths = listdir(drop_profile_directory)
drop_img_directory = root_directory / "drop_images"
mkdir(drop_img_directory)

for model in c["models"]:

    model_img_directory = drop_img_directory / model["name"]
    mkdir(model_img_directory)

    for _ in tqdm(range(int(c["image_parameters"]["N"]))):
        desc=f"Generating Drop Images for {model['name']}"):

            # Grab random drop profile

```

```

drop_profile = choice(drop_profiles_paths)
drop_profile = drop_profile_directory / drop_profile
drop_parameters = name_to_parameters(drop_profile)
drop_profile = loadtxt(drop_profile, delimiter=",")

# Randomly select further parameters for image manipulations
rotation = uniform(*c["image_parameters"]["rotation_range_degrees"])
drop_scale = uniform(*c["image_parameters"]["drop_scale_range"])
cap_height = uniform(*model["rel_capillary_height_range"])
if uniform(0, 1):
    cap_height = 0

# Update Parameters
drop_parameters["rotation"] = rotation
drop_parameters["drop_scale"] = drop_scale
drop_parameters["rel_capillary_height"] = cap_height

# Get other variables
img_size_pix=c["image_parameters"]["img_size_pix"]
noise=model["noise"]
above_apex=model["only_keep_drop_above_apex"]
delta_s=c["drop_profiles"]["delta_s"]

# Generate drop images, calculating the relative drop radius
drop_radius, drop_image = new_drop_image(drop_profile=drop_profile,

```

```

img_size_pix=img_size_pix,
rotation=rotation,
drop_scale=drop_scale,
noise=noise,
rel_capillary_height=cap_height,
above_apex=above_apex,
delta_s=delta_s
)

drop_parameters["drop_radius"] = drop_radius
file = parameters_to_name(drop_parameters, suffix=".png")
imwrite(str(model_img_directory / file), drop_image)

# This part just creates datasplits for the different groups

# Define split percents
train_percent = c["model_parameters"]["train_percent"]
val_percent = c["model_parameters"]["val_percent"]
test_percent = 1 - train_percent - val_percent
N = c["image_parameters"]["N"]
train_percent = int(train_percent * N)
val_percent = int(val_percent * N)
test_percent = int(test_percent * N)

# Split datasets

```

```

root_model_directory = root_directory / "models"
mkdir(root_model_directory)

for model in tqdm(c["models"], desc="Splitting datasets"):
    model_directory = root_model_directory / model["name"]
    mkdir(model_directory)

    # Gather img paths for a particular model
    model_img_directory = drop_img_directory / model["name"]
    img_paths = []
    for img_path in model_img_directory.iterdir():
        img_paths.append(img_path)

    # Split
    shuffle(img_paths)
    train_imgs = img_paths[:train_percent]
    val_imgs = img_paths[train_percent:train_percent+val_percent]
    test_imgs = img_paths[train_percent+val_percent:]

    # Save Split
    model_split = dict(
        train=train_imgs,
        val=val_imgs,
        test=test_imgs,
    )

    with open(model_directory / "split.json", "w") as f:

```

```

dump(model_split, f, default=str, indent=4)

# Train models, and test on numerically generated data test set
all_y_keys = ["bond_number", "volume", "area", "cap_diameter" , "drop_radius"]
for model in c["models"]:

    # Define directories
    main_model_path=root_model_directory / model["name"]

    for y_key in all_y_keys:

        # Load data split
        with open(main_model_path / "split.json", "r") as f:
            data_split = load(f)

        # Define directory to save model
        key_model_path=root_model_directory / model["name"] / y_key
        mkdir(key_model_path)
        print(f"training model: {key_model_path}")

        # Train model, collecting y scaler and loss vs epochs
        epochs = c["model_parameters"]["epochs"]
        batch_size = c["model_parameters"]["batch_size"]
        keras_model, history_data, y_scaler = train(split=data_split,
                                                    y_key=y_key,

```

```

epochs=epochs,
batch_size=batch_size
)

# Generate Predicted vs Actual data

normalized_pva, pva = test(model=keras_model, y_key=y_key,
                           y_scaler=y_scaler,
                           y_files=data_split["test"],
                           batch_size=batch_size)

# Save data

with open(key_model_path / "y_scaler.save", "wb") as f:
    joblib_dump(y_scaler, f)

with open(key_model_path / "history.json", "w") as f:
    dump(history_data, f)

keras_model.save(key_model_path / "model")

savetxt(key_model_path / "normalized_pva.csv", normalized_pva,
        delimiter=",")

savetxt(key_model_path / "pva.csv", pva, delimiter=",")

plt.scatter(pva[:, 0], pva[:, 1])
plt.savefig(key_model_path / f"pva.png")
plt.close()

```

```
plt.scatter(normalized_pva[:, 0], normalized_pva[:, 1])
plt.savefig(key_model_path / f"pva_normalized.png")
plt.close()
```

A.9 Wrapper to Predict Fluid Properties

```
def default_predict(img_files: list, parameter: str):

    model_path = Path(c["directory"]) / "models/control" / parameter

    # Load scaler from control model
    scaler_path = model_path / "y_scaler.save"
    with open(scaler_path, "rb") as f:
        scaler = joblib_load(f)

    # Load control model
    model_path = model_path / f"model"
    model = load_model(model_path)

    # Load images
    imgs = []
    for img in img_files:
        imgs.append(img_file_to_img(img))
    imgs = array(imgs)
```

```

# Make predictions

pred = model.predict(imgs, verbose=0)

pred = scaler.inverse_transform(pred).flatten()

return pred

```

References

- [1] AW Adamsom and AP Gast. "Physical chemistry of surfaces". In: *A Wiley-Interscience Publication*, (1997).
- [2] Pierre-Gilles Gennes, Françoise Brochard-Wyart, David Quéré, et al. *Capillarity and wetting phenomena: drops, bubbles, pearls, waves*. Springer, 2004.
- [3] Jens Eggers and Emmanuel Villerraux. "Physics of liquid jets". In: *Reports on progress in physics* 71.3 (2008), p. 036601.
- [4] Francis Bashforth and John Couch Adams. *An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluid*. University Press, 1883.
- [5] AW Neumann and RJ Good. "Techniques of measuring contact angles". In: *Surface and Colloid Science: Volume 11: Experimental Methods* (1979), pp. 31–91.
- [6] Dietmar Möbius and Reinhard Miller. *Proteins at liquid interfaces*. Elsevier, 1998.

- [7] Bea Briscoe, P Luckham, and S Zhu. "The effects of hydrogen bonding upon the viscosity of aqueous poly (vinyl alcohol) solutions". In: *Polymer* 41.10 (2000), pp. 3851–3860.
- [8] Pierre Simon marquis de Laplace and Nathaniel Bowditch. *Celestial mechanics*. Vol. 194. American Mathematical Soc., 1966.
- [9] Joseph D Berry et al. "Measurement of surface and interfacial tension using pendant drop tensiometry". In: *Journal of colloid and interface science* 454 (2015), pp. 226–237.
- [10] N Ashgriz and JY Poo. "Coalescence and separation in binary collisions of liquid drops". In: *Journal of Fluid Mechanics* 221 (1990), pp. 183–204.
- [11] Milton J Rosen and Joy T Kunjappu. *Surfactants and interfacial phenomena*. John Wiley & Sons, 2012.
- [12] Denise Neibloom, Michael A Bevan, and Joelle Frechette. "Surfactant-Stabilized Spontaneous 3-(Trimethoxysilyl) Propyl Methacrylate Nanoemulsions". In: *Langmuir* 36.1 (2019), pp. 284–292.
- [13] Xiaoqing Hua, Michael A Bevan, and Joelle Frechette. "Competitive adsorption between nanoparticles and surface active ions for the oil–water interface". In: *Langmuir* 34.16 (2018), pp. 4830–4842.
- [14] YY Zuo and AW Neumann. "Application of axisymmetric drop shape analysis (ADSA) to the study of biomolecules". In: *Molecular Interfacial Phenomena of Polymers and Biopolymers*. Elsevier, 2005, pp. 249–285.
- [15] Elisa Migliorini, Marianne Weidenhaupt, and Catherine Picart. "Practical guide to characterize biomolecule adsorption on solid surfaces". In: *Biointerphases* 13.6 (2018), p. 06D303.

- [16] Xiaoqing Hua, Joelle Frechette, and Michael A Bevan. "Nanoparticle adsorption dynamics at fluid interfaces". In: *Soft Matter* 14.19 (2018), pp. 3818–3828.
- [17] Anahita Fathi Azarbayjani, Abolghasem Jouyban, and Sui Yung Chan. "Impact of surface tension in pharmaceutical sciences". In: *Journal of pharmacy & pharmaceutical sciences* 12.2 (2009), pp. 218–228.
- [18] Graham Buckton. *Interfacial phenomena in drug delivery and targeting*. CRC press, 2000.
- [19] Carel Jan van Oss. "Development and applications of the interfacial tension between water and organic or biological surfaces". In: *Colloids and surfaces B: Biointerfaces* 54.1 (2007), pp. 2–9.
- [20] Mira T Guo et al. "Droplet microfluidics for high-throughput biological assays". In: *Lab on a Chip* 12.12 (2012), pp. 2146–2155.
- [21] Mingxiang Luo, Rohini Gupta, and Joelle Frechette. "Modulating contact angle hysteresis to direct fluid droplets along a homogenous surface". In: *ACS applied materials & interfaces* 4.2 (2012), pp. 890–896.
- [22] Jaroslaw Drelich and Emil Chibowski. "Superhydrophilic and superwetting surfaces: definition and mechanisms of control". In: *Langmuir* 26.24 (2010), pp. 18621–18623.
- [23] Florian Geyer et al. "When and how self-cleaning of superhydrophobic surfaces works". In: *Science advances* 6.3 (2020), eaaw9727.
- [24] Srijita Nundy, Aritra Ghosh, and Tapas K Mallick. "Hydrophilic and superhydrophilic self-cleaning coatings by morphologically varying ZnO microstructures for photovoltaic and glazing applications". In: *ACS omega* 5.2 (2020), pp. 1033–1039.

- [25] DE Sullivan. "Surface tension and contact angle of a liquid–solid interface". In: *The Journal of chemical physics* 74.4 (1981), pp. 2604–2615.
- [26] SG Croll. "Surface roughness profile and its effect on coating adhesion and corrosion protection: A review". In: *Progress in Organic Coatings* 148 (2020), p. 105847.
- [27] Lishen Zhang et al. "Functional and versatile superhydrophobic coatings via stoichiometric silanization". In: *Nature communications* 12.1 (2021), p. 982.
- [28] Sina Ebnesajjad. "3 - Surface Tension and Its Measurement". In: *Handbook of Adhesives and Surface Preparation*. Ed. by Sina Ebnesajjad. Plastics Design Library. Oxford: William Andrew Publishing, 2011, pp. 21–30. ISBN: 978-1-4377-4461-3. DOI: <https://doi.org/10.1016/B978-1-4377-4461-3.10003-3>. URL: <https://www.sciencedirect.com/science/article/pii/B9781437744613100033>.
- [29] Jinlong Yang, Kyle Yu, and Yi Y Zuo. "Accuracy of axisymmetric drop shape analysis in determining surface and interfacial tensions". In: *Langmuir* 33.36 (2017), pp. 8914–8923.
- [30] OI Del Rio and AW Neumann. "Axisymmetric drop shape analysis: computational methods for the measurement of interfacial properties from the shape and dimensions of pendant and sessile drops". In: *Journal of colloid and interface science* 196.2 (1997), pp. 136–147.
- [31] Sameh MI Saad, Zdenka Policova, and A Wilhelm Neumann. "Design and accuracy of pendant drop methods for surface tension measurement". In: *Colloids and Surfaces A: Physicochemical and Engineering Aspects* 384.1-3 (2011), pp. 442–452.

- [32] Tammar S Meiron, Abraham Marmur, and I Sam Saguy. "Contact angle measurement on rough surfaces". In: *Journal of colloid and interface science* 274.2 (2004), pp. 637–644.
- [33] E Hernández-Baltazar and J Gracia-Fadrique. "Elliptic solution to the Young–Laplace differential equation". In: *Journal of Colloid and Interface Science* 287.1 (2005), pp. 213–216.
- [34] Y Rotenberg, Lr Boruvka, and A Wilhelm Neumann. "Determination of surface tension and contact angle from the shapes of axisymmetric fluid interfaces". In: *Journal of colloid and interface science* 93.1 (1983), pp. 169–183.
- [35] Tejaswi Soori et al. "A machine learning approach for estimating surface tension based on pendant drop images". In: *Fluid Phase Equilibria* 538 (2021), p. 113012.
- [36] Felix S Kratz and Jan Kierfeld. "Pendant drop tensiometry: A machine learning approach". In: *The Journal of Chemical Physics* 153.9 (2020), p. 094102.
- [37] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7482–7491.
- [38] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open Source Scientific Tools for Python*. <https://www.scipy.org/>. Accessed on March 29, 2023. 2001.
- [39] Athanasios Voulodimos et al. "Deep learning for computer vision: A brief review". In: *Computational intelligence and neuroscience* 2018 (2018).
- [40] François Chollet et al. *Keras*. <https://keras.io>. 2015.