PROCESS-PROPERTY LINKAGES CONSTRUCTION FOR INKJET PRINTING WITH MACHINE LEARNING

by

Fataneh Jenabi

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Electrical and Computer Engineering

Boise State University

August 2022

© 2022

Fataneh Jenabi

ALL RIGHTS RESERVED

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Fataneh Jenabi

Thesis Title: Process-Property Linkages Construction for Inkjet Printing with Machine Learning

Date of Final Oral Examination: 27 June 2022

The following individuals read and discussed the thesis submitted by student Fataneh Jenabi, and they evaluated the student's presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Harish Subbaraman, Ph.D.	Chair, Supervisory Committee
David Estrada, Ph.D.	Member, Supervisory Committee
Kurtis Cantley, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Harish Subbaraman, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

DEDICATION

To my lovely family.

ACKNOWLEDGMENTS

I would like to thank Dr. Subbaraman for giving me this opportunity to work on this project. His support and trust were critical to the success of this project. I also would like to thank Dr. Estrada and Dr. Cantley for agreeing to be on my thesis committee and for their constructive feedback on the proposal. Special thanks to the ANML members for their constant support during my three years of presence in the group. Also, I would like to thank the undergraduate researchers, Andrew Timmons, Cosmin Fologea, and Joseph Kaempen for their help in experimental database generation. Finally, I would like to thank the National Science Foundation and NSF I/UCRC ATOMIC for the project support.

ABSTRACT

Printed electronics are emerging technologies that can potentially revolutionize the manufacturing of electronic devices. One promising technology for printed electronics is inkjet printing. Inkjet printing offers both low-cost processing and high resolution. Being a subset of additive manufacturing, inkjet printing minimizes waste and is compatible with a wide range of inks. However, inkjet printing of electronic devices is still in its infancy. One major challenge for inkjet printing is the complexity of the process optimization and uncertain high throughput production. To achieve a high-quality print, there is a complex parameter space of materials and processing parameters that needs to be optimized. To address this challenge, in this thesis work, we develop a machine learning algorithm to connect the processing parameters to print morphology for inkjet processes. To achieve this goal, we developed more than 200 experimental samples and processed the print images automatically with OpenCV-based codes. Finally, we correlated the morphology specifications, i.e., print line width, overspray, and roughness to the processing parameters, i.e., cartridge height, nozzle voltage, and drop spacing, via a neural network model. The order of machine learning model accuracy from high to low is for line width, roughness, and overspray, respectively. The model's low predictability of overspray can be attributed to our limited dataset, Dimatix unreliable performance, or the low dependency of overspray on the processing parameters of this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IJP	Inkjet Printing
ML	Machine Learning
DOD	Drop on Demand
CIJ	Continuous Inkjet
EDA	Exploratory Data Analysis
NN	Neural Network
RSME	Root Mean Square Error
OpenCV	Open Source Computer Vision

CHAPTER ONE: INTRODUCTION

Printed electronics have recently experienced tremendous growth and attracted considerable attention both in industry and academia (Khan et al., 2020). The main cause of this interest compared to other traditional electronic device manufacturing techniques such as photolithography is that printed electronic technologies are not expensive, do not produce excessive waste, and are suitable for rapid prototyping.

As the name suggests, printed electronics means printing electronic patterns or structures on substrates that are very thin and flexible, which finally results in inexpensive electronic device fabrication. Being printed on flexible substrates, the technology enables us to shape these devices in different forms. For example, printed electronic devices can be curved, rolled, stretched, folded, etc., and this capability opened a wide range of applications for printed electronics, from aerospace to medicine, see Figure 1 (*Nanotechnology in Flexible Electronics*, 2012). We note that printed electronics are not limited to electronic devices. With this digital fabrication, we can print a layer of conductive or semiconductor ink directly on substrates. The additive manufacturing nature of printed electronics makes the manufacturing simple with minimal waste and more environmentally friendly.

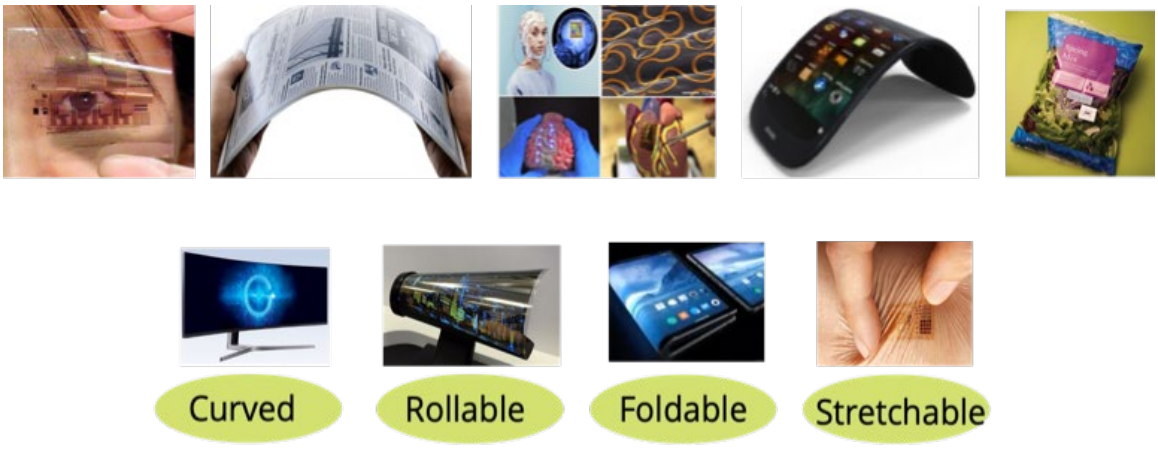

Figure 1. Flexible electronics device applications (*Nanotechnology in Flexible Electronics*, 2012).

Printing Technologies

Printing technologies enable us to fabricate flexible electronic devices. There is a wide range of printing technologies, and they are primarily divided into two main categories: Contact and Non-contact printing. Contact printing technologies include Offset printing, Flexographic, Gravure printing, etc. Some examples of non-contact printing technologies are Aerosol-jet printing, Organic Vapor-jet printing, and Inkjet printing. Figure 2 has listed the commonly available printing technologies (Saengchairat et al., 2017). Recently non-contact printing technologies have gained more attention because they have lower waste, create high-resolution patterns, and have the potential for mass production. Among non-contact printing technologies, Inkjet Printing (IJP) has received more attention primarily because it can produce the highest resolution patterns and the process is fairly simple. Figure 3 shows the schematic of the traditional, contact, and non-contact electronic device fabrication methodologies (Zhang & Moon, 2021).

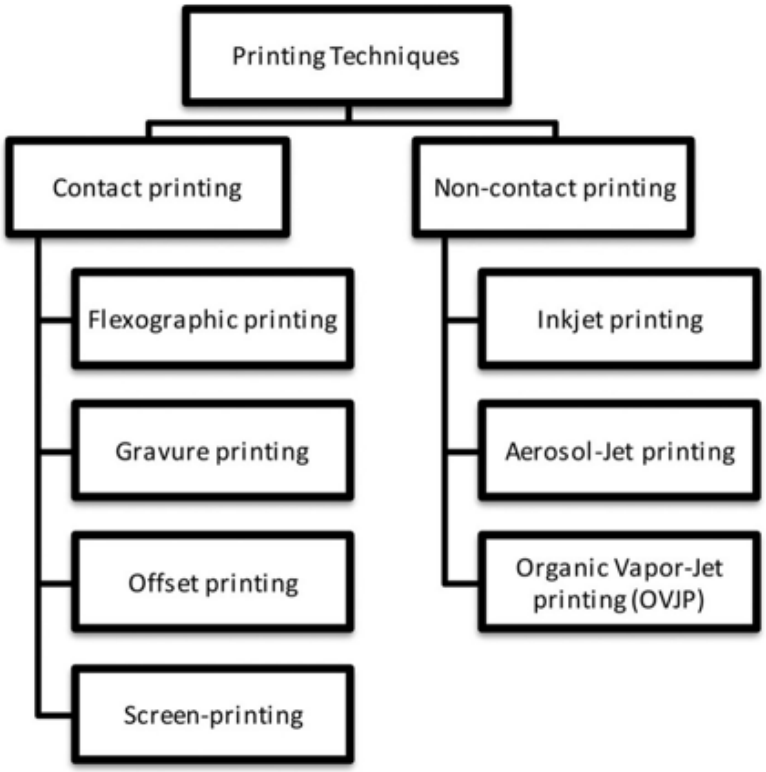

Figure 2. **Commonly available printing technologies (Saengchairat et al., 2017).**

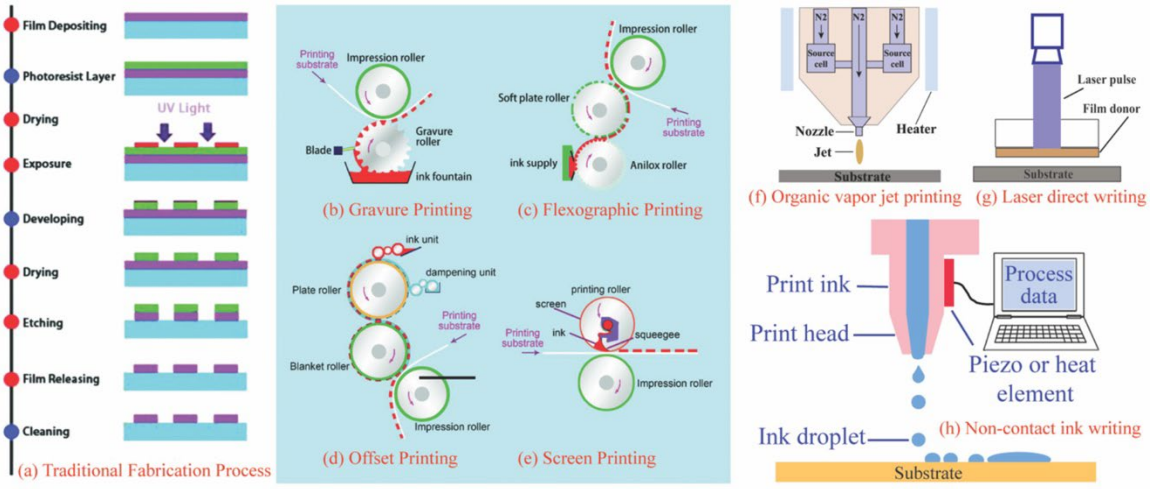

Figure 3. **Schematic of (a) Photolithography, (b-e) Contact printing, (f-h) Non-contact printing technologies (Zhang & Moon, 2021).**

Inkjet Printing

There are several reasons why Inkjet Printing (IJP) has become very popular these days, but the primary reason is that this process is cheap. IJP can produce complex patterns with high resolution. The patterns in IJP are totally digital, which means we create the patterns in CAD software and there is no need for a stencil or a template. That makes IJP ideal for rapid prototyping, and as mentioned above, IJP has minimal waste. In IJP, we can use a wide range of inks from metals to dielectrics to semiconductors, e.g., silver, carbon nanotubes, etc. IJP is compatible with flexible substrates that can be bent, stretched, and rolled, without having the common complexity that we typically experience with silicon.

For inkjet printing, there are two main jetting mechanisms including continuous inkjet (CIJ) and drop-on-demand inkjet (DoD) (Hoath, 2016). In the CIJ method, the ink droplets are continuously produced and jetted. In this technique, the droplets are electrically charged and get deflected selectively with an electrostatic field. In the DoD method, the droplet jetting is controlled by piezoelectric or thermal actuators. Between CIJ and DoD, the DoD methods have received more attention due to the complexity of ink control in CIJ.

In DoD techniques, for the piezoelectric case, voltage pulses are applied to the piezoelectric material, and in reaction, it produces pressure on the ink and jet it out of the nozzle. A broad range of water-based or solvent-based inks, which are either conductive or non-conductive, can be deposited using this technique. In the thermal process, the ink is heated rapidly to produce a bubble and then the bubble pushes out the ink as a droplet. The creation of vapor bubbles can sometimes lead to the formation of a layer of ink over

the resistive heater. This could decrease the performance of the printer over time. Between the DoD methods, the piezoelectric method is more controllable. The machine that we are using in this project used a piezoelectric printhead. Figure 4 shows the different inkjet printing technologies (Hu et al., 2018).

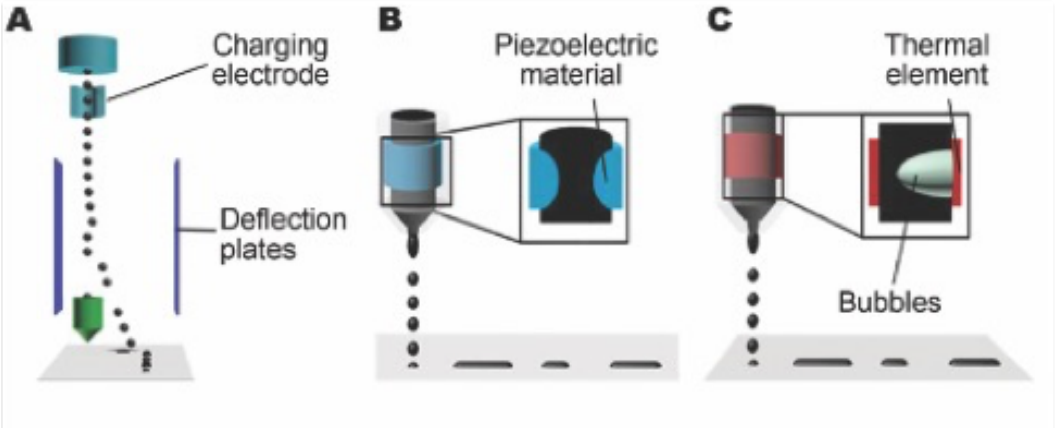

Figure 4. Different inkjet printing technologies, (a) Continuous inkjet, (b) Piezoelectric DoD, and (c) Thermal DoD (Hu et al., 2018).

Despite all the benefits of the IJP technology, printing a high-quality structure or pattern is a very challenging task. Several properties of inks such as viscosity, surface tension, density, and particle sizes determine whether ink is printable or not. Extensive studies in classical printing technologies have been conducted and a guide chart to determine the printability of the ink has been developed (McKinley & Renardy, 2011). Figure 5 shows a good benchmark to determine the ink printing behavior. The y-axis of the chart is the Ohnesorge number (Oh) and the x-axis is the Reynolds number (Re).

$$Oh = \frac{\mu}{\sqrt{\rho\sigma D}} \tag{1.1}$$

$$Re = \frac{\rho V D}{\mu} \tag{1.2}$$

where μ is the dynamic viscosity of the liquid, ρ is the density of the liquid, σ is the surface tension, D is nozzle diameter, and V is fluid speed.

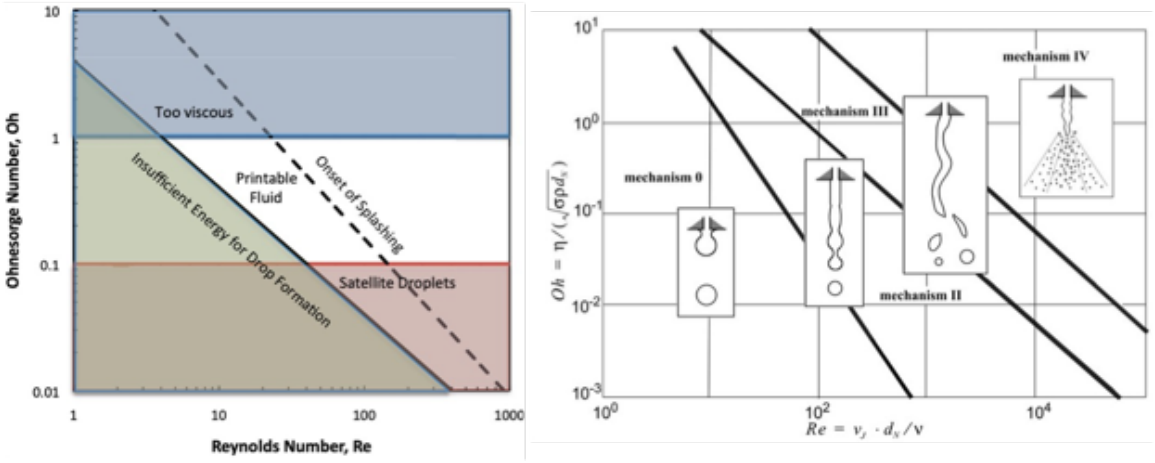

Figure 5. Guide chart for ink printability domain determination (Heinzen et al., 2004; McKinley & Renardy, 2011).

Beyond printable ink development, the printed patterns must be defect-free to make them functional in applications. Several forms of defects have been identified for IJP technology including, disconnected prints, rough edges, coffee rings (when the ink particles move to the perimeter and leave the middle part non-conductive), and shrinkages (when the ink particles shrink into the pattern's middle part), etc. Figure 6 shows some common forms of IJP defects (Anyfantakis & Baigl, 2015, and this work).

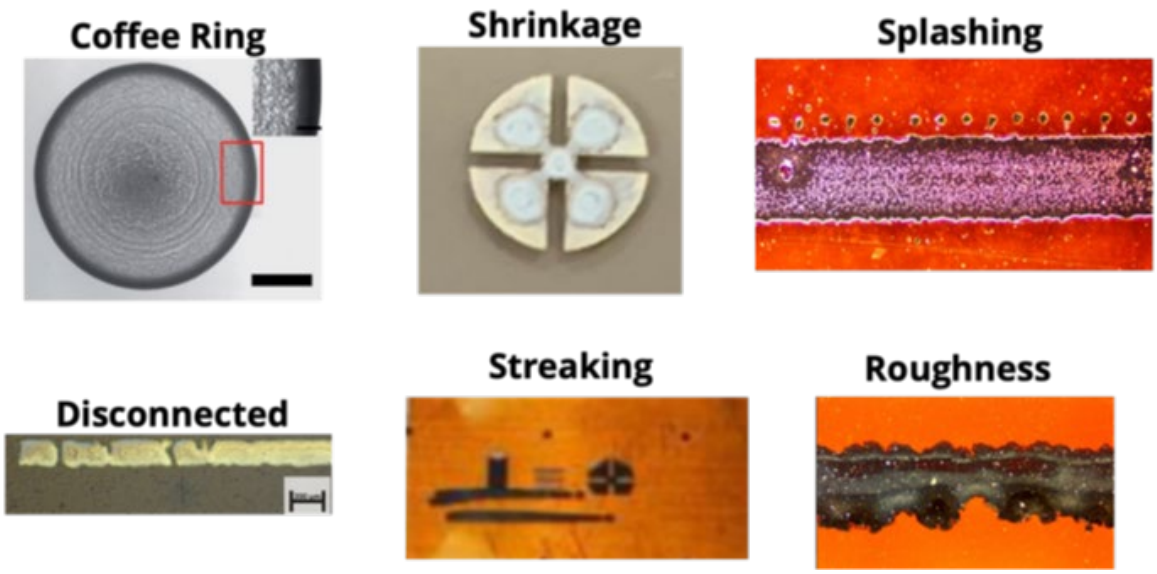

Figure 6. Typical defects that are observed in IJP prints (Anyfantakis & Baigl, 2015, and this work).

In addition to the ink, process parameter control is also critical in minimizing defect formation, specifically in morphological defects. The main processing parameters for IJP are nozzle voltage, drop spacing, cartridge height, platen temperature, etc. Proper process parameter selection is crucial for getting smooth patterns. In this project, we constructed process-property linkages for IJP with machine learning. Specifically, we focused on three processing parameters namely, cartridge height, nozzle voltage, and drop spacing, and we studied their effects on print line width, roughness, and overspray. Chapter Two describes the experimental database generation. Chapter Three discusses the print lines image processing. Chapter Four covers the machine learning model development. Finally, in Chapter Five, we provide our conclusions.

CHAPTER TWO: EXPERIMENTAL DATABASE GENERATION

Data gathering/generation is the most critical and time-consuming part of most machine learning projects. For this project, because we did not have a prior database, we developed a unique database including hundreds of prints. The printer that we used in this study was a Dimatix DMP- 2831, shown in Figure 7 (Dimatix, 2022). The printer is suited to print fluids (inks) containing proteins, nucleic acids, nanoparticles, conductive polymers, etc. Dimatix can print inks with viscosity in the range of 8–12 Cp and surface tension in the range of 28–42 mN/m. The printer substrate holder temperature can vary between ambient to 60°C and the recommended drop velocity range for this printer is 7–9 m/s.

Figure 7. Dimatix (DMP- 2831) IJP printer (Dimatix, 2022).

To narrow down the scope of the project, we kept the ink consistent. While within our Advanced Nanomaterials and Manufacturing Laboratory (ANML) at Boise State, we have the expertise to develop in-house inks, ink development was out of scope for this project. Therefore, we used commercial inks in this project. Specifically, we used JS-A191 Silver inks developed by NovaCentrix. Table 1 shows the characterization data of the ink. By transforming these data into Ohnesorge and Reynolds numbers, we can see that their combination fit the ink within the printable window.

Table 1. Physical properties and specifications of the JS-A191 Silver ink (NovaCentrix, 2022).

Rheology properties	**Values provided by manufacturer**
Viscosity (cP)	7.5
Nozzle diameter (um)	21
Surface tension (dyne/cm)	34.2
Density (g/cm3)	1720
Fluid velocity (m/s)	11
Oh	0.211
Z	4.7
Re	54.23

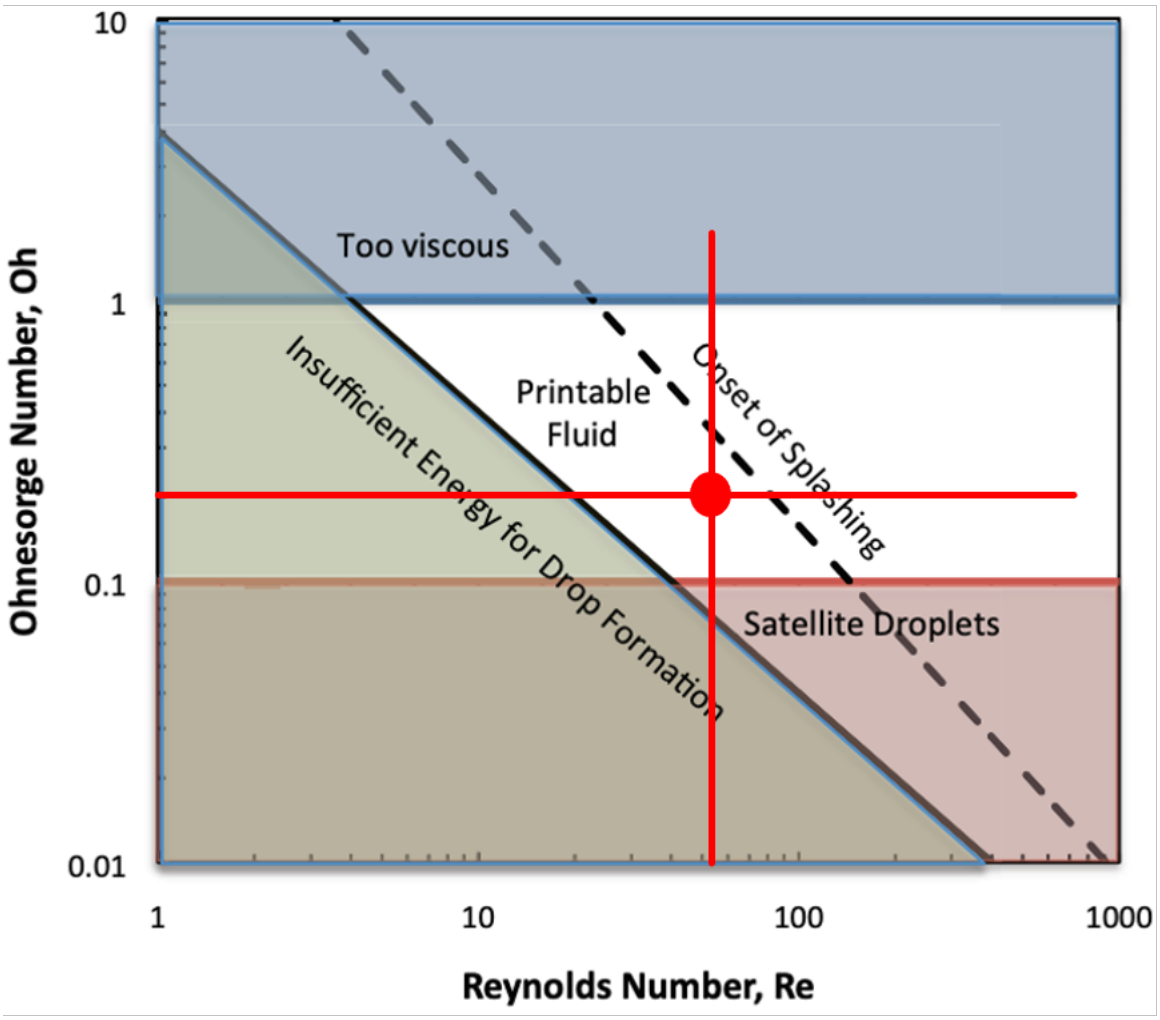

Figure 8. **The physical properties of the silver ink combined in the form of Oh and Re numbers show that the ink is printable.**

Database Generation

There are several processing parameters that we could focus on, but in this project, we focused on three crucial processing parameters including cartridge height, nozzle voltage, and drop spacing. The cartridge height is defined as the distance from the nozzle to the top of the substrate. The nozzle voltage determines what voltage that we apply to the piezoelectric head. Piezoelectric materials will deform proportionally with voltage. Therefore, higher voltage leads to higher deformation and ultimately higher ink drop velocity. Drop spacing is another important parameter that we studied in this

project. A large drop spacing could make the print beaded. On the other hand, if the drop spacing is too small, the drops would be on top of each other and we might get too much ink overlap, also known as bulging.

For cartridge print height, we studied the range from 650 um to 800 um with a grid size of 50 um. For nozzle voltage, we studied the range from 25 Volts to 40 Volts with a grid size of 3 Volts. And finally, for the drop spacing, we studied the range from 8 um to 17 um with a grid size of 1 um.

In total, with these combinations, we produced more than 200 samples. It is worth noting that printing 200 samples is not a trivial task, and it was done over several months. For this part of the project, we worked closely with several undergraduate researchers. Therefore, consistency was an important factor for us. We tracked every single print using a list that showed when and who printed a specific sample, labeling each print with a unique ID. We also tracked the print head that had been used in printing. Our experience with inkjet printing showed that the printhead is a critical component. We occasionally had some print heads fail. Thus, it was very important for us to be able to track the printheads and identify them in case of failure.

In this project, again based on our experience, we did not use multiple nozzles. Although it can make the process faster, it would make the problem much more complex than needed. So, we decided to just print with a single nozzle, leading to our long time in data collection. Printing with multiple nozzles increases the chance of clogging and will introduce much more uncertainty to the problem. We also kept the jetting frequency constant for all prints.

For each set of processing parameters, we produced two sets of lines and Cloverleaf patterns for the safety factor in case one fails due to clogging, etc. Figure 9 shows a set of patterns that we produced for each set of processing parameters. The printed lines were used for print morphology analysis and Cloverleaf patterns were used for resistivity measurement, which is out of the scope of this work but will be used by future students to further improve the model.

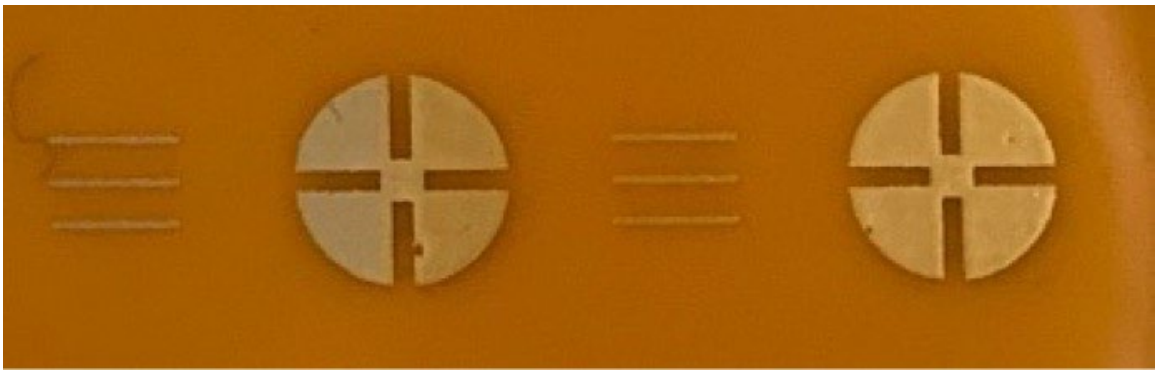

Figure 9. The pattern that we generate for each set of processing parameters.

CHAPTER THREE: IMAGE PROCESSING OF PRINTED LINES

In this section, we will show a Python-based image processing code that enables automatic morphology analysis of the prints. The code is based on the OpenCV image processing package (OpenCV, 2022a). The goal is to read the print lines and automatically measure their line width, roughness, and overspray. This code will create the tabular database that we need for the machine learning part.

Image Preparation

As mentioned in Chapter Two, for each set of processing parameters, we produced six lines and two cloverleaf patterns. We studied the printed lines for morphology analysis and the cloverleaf patterns will be used for the Van der Pauw resistivity measurement in the future. We used a ZEISS Axio Imager microscope to image the printed lines. The microscope zoom was not enough to cover the entire length of lines, therefore, we had to take four images for each set of lines. Figure 10 shows the set of images that we took for each set of prints.

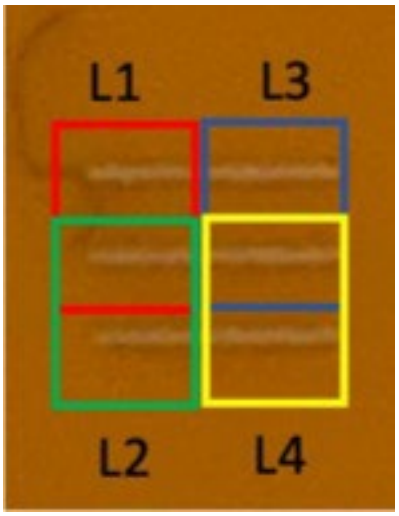

Figure 10. Four images were taken for each set of prints (L stands for the left. We have similar images for R1, R2, R3, and R4 for the right lines in Figure 9.

Considering both left and right lines in Figure 9, we had eight images for each print, i.e., four images for the left and four images for the right. Figure 11 shows a typical microscope image that we produced for the printed lines.

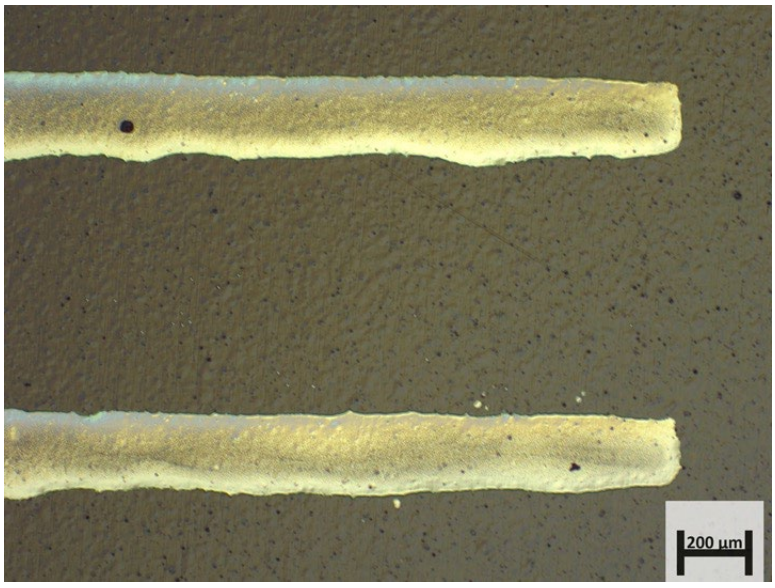

Figure 11. Typical microscope image of two print lines.

Since in image processing we want to analyze one image at a time, we need to crop the images in a way so that just one line remains in each image. In cropping, since the middle line is shared between two images, it would be cropped two times. Therefore, we need to be careful not to count them twice. For that purpose, we meticulously named the images based on their position in the image, i.e., the top left is L1, the next is L2, the bottom left is L3, and the next to it is L4, the same naming nomenclature is true for the right images. Since we are dealing with hundreds, and in the future thousands, of images, it is important to make the cropping process automatic. We wrote a Python code to perform the cropping and naming automatically. With proper cropping, we have 12 images for each processing parameter. Figure 12 shows one sample of the cropped images that we fed into the image processing code.

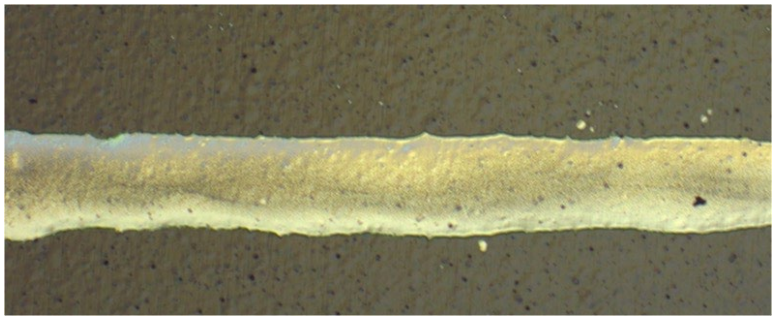

Figure 12. Cropped printed line ready for image processing.

Image Processing

After preparing the printed lines, we performed image processing to extract the line width, roughness, and overspray for each print. To perform this task, we used the OpenCV (Open Source Computer Vision) library (*Opencv/Opencv*, 2012/2022). OpenCV is a library of programming functions originally developed by Intel that is mainly aimed

at real-time computer vision. We describe the image processing steps by using the image in Figure 12 as an example. Initially, after reading the image, we need to perform a two-step image preparation - 1) make the image gray, 2) make the image binary, i.e., black (pixel number 0) and white (pixel number 255), see Figure 13.

Figure 13. The binary version of a printed line.

To make an image binary in OpenCV, we need to define a threshold pixel number as the border in which the OpenCV makes the pixels smaller than that zero, i.e., black, and makes the pixel numbers larger than that 255, i.e., white. Since the experimental data generation has been a long process and the microscope that we are using is shared equipment, it happened quite often that the microscope produced images with different color tones over time. Figure 14 shows some of the pictures that we have produced in our work.

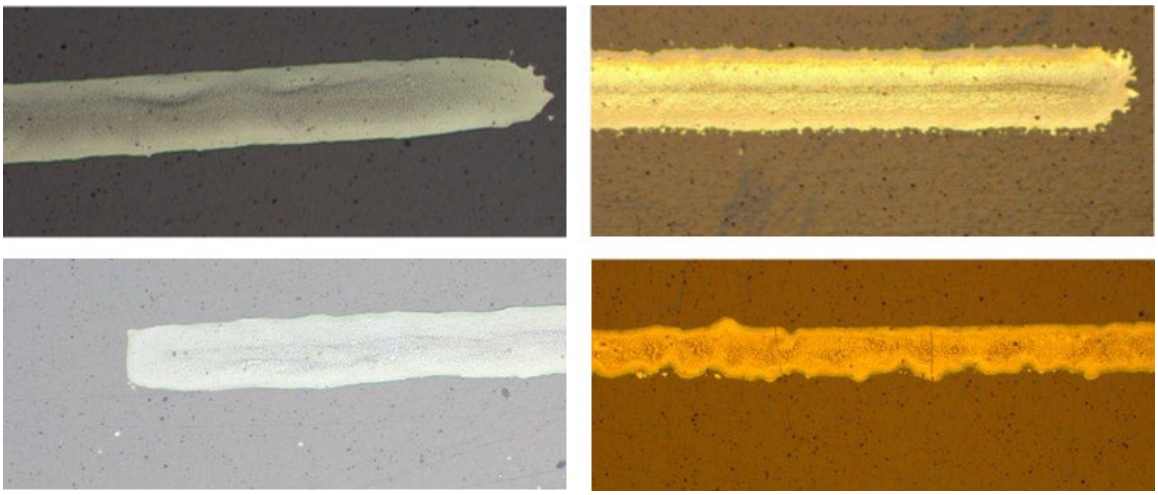

Figure 14. Various print colors were produced by the microscope.

The variation of color tone makes the binary image production very challenging as we need to manually set, by trial and error, the threshold pixel number for every single image. To overcome this challenge, we made the threshold determination automatic by producing the pixel number distribution for each individual image and then finding the mean of the existing pixels weighted by their count in the histogram, see Figure 15. Since the majority of the pixels in the images are for the background, we increased the calculated mean by 10% and consider the new number as the threshold for making the black and white pictures. With this technique, we were able to automatically process the images without worrying about the pictures' color tone.

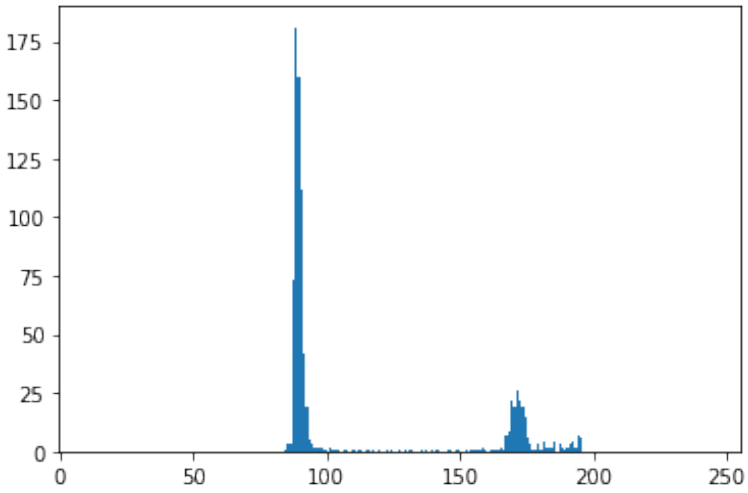

Figure 15. Pixel distribution in a sample picture.

After generating a binary version of the print image, we first focused on the line width and roughness. Since overspray introduces some errors in line width and roughness, we first need to remove them. To perform this task in the code, we first find all contours in the image and extract the small contours, here we defined contours with the area between 1 to 5000 pixels as small, and then remove them, by making their color black, similar to the background. Figure 16 shows the print image with removed overspray.

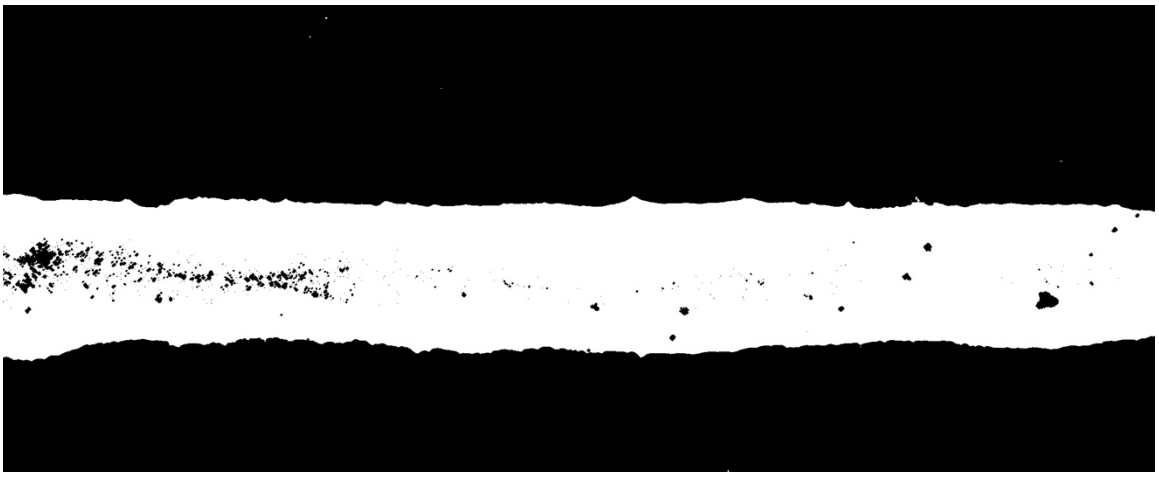

Figure 16. Print line with removed overspray.

Since the print surface is not smooth, some parts of the printed surface are dark in the image so it might seem the print has hollows, but indeed those parts are prints and not substrate. Therefore, we need to remove the black dots on the print image. One effective way to remove the salt-and-pepper noise effects in the images with OpenCV is using the cv2.medianBlur() function (OpenCV, 2022b). The cv2.medianBlur() computes the median of all the pixels under the kernel window and the central pixel is replaced with this median value. Figure 17 shows the printed line with a smooth surface (compare to Figure 16).

Figure 17. The printed line with removed Salt&Pepper effects.

Line Width Calculation

After preparing the images, we find the print's top and bottom lines. To do so, we first find the columns where the print starts and finishes. With these, we can find the print length. Please note that in all OpenCV calculations, we are working with pixels, so when talking about the print length, we are looking at the print length in pixels. In the end, depending on the scale bar, we can transform the pixels to their physical sizes. After finding the print start and end, for each column in between them, we find the vertical

location, the y-axis, where we have the pixel number 255 for the first time, this shows the top, and for the last time, this shows the bottom, and store all the top and bottom vertical locations into specific arrays. Finally, we find the print top and bottom lines by linear regression between the top points and bottom points. After having the top and bottom lines, we can calculate the print width by averaging the print width, i.e., by averaging the difference between the top and bottom lines for all columns in the print area.

Roughness Calculation

Roughness is an important morphology parameter for printed electronics. Generally, we are interested in printing patterns with minimum roughness. For two-dimensional objects, we have profile roughness, in contrast to surface roughness for three-dimensional objects. The profile roughness parameters are typically calculated based on BS EN ISO 4287:2000 British standard (BS EN ISO 4287, 2000) that is identical to the ISO 4287:1997 standard (ISO 4287, 1997). There are several parameters to quantify the roughness. The most common parameter is *Ra*, which is the arithmetic average of profile height deviations from the mean line. In this project, we used the same parameter for the roughness quantification (Malkin & Isayev, 2022).

$$Ra = \frac{1}{L}\int_0^L |z(x)|dx \tag{2.1}$$

Overspray Calculation

For overspray calculation, we start with the binary image, i.e., Figure 13. In this case, we first find all contours in the image and calculate their centroid coordination by using the M_{00}, M_{10}, and M_{01} image moments. For an image, the moment M_{ij} is defined as follows (Hart et al., 2000):

$$M_{ij} = \sum_x \sum_y x^i y^i I(x, y) \tag{2.2}$$

where M_{ij} is the moment, *x* and *y* are the coordinates, and *I(x,y)* is the intensity of a pixel at the location *(x,y)*. Using this definition, the centroid of an image can be calculated by (Hart et al., 2000):

$$x_{center} = \frac{M_{10}}{M_{00}} \quad (2.3)$$

$$y_{center} = \frac{M_{01}}{M_{00}} \quad (2.4)$$

After finding the centroids of all contours, we find the overspray contours by excluding all the contours where the *y*-coordinate of their centroid is either above the top line or below the bottom line.

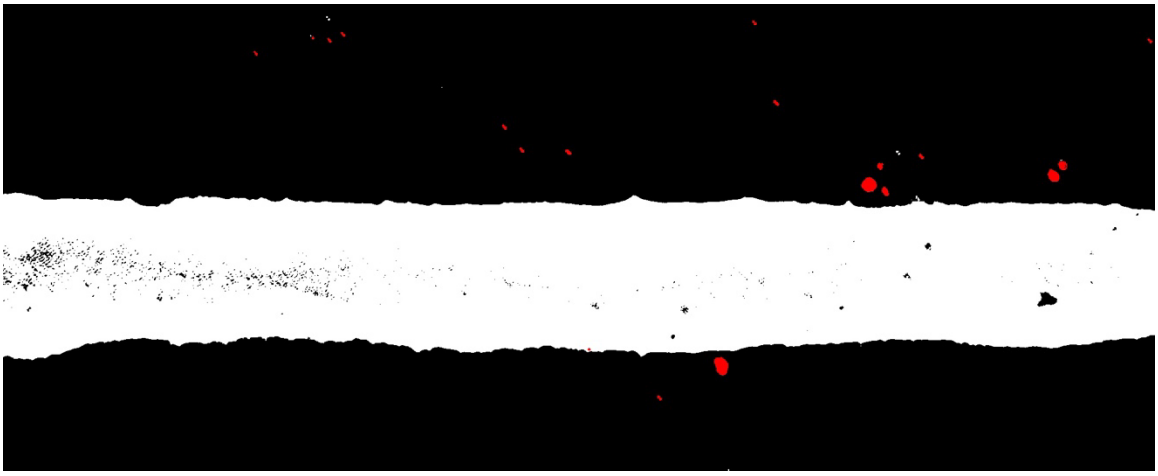

Figure 18. Red contours are identified as overspray.

After identifying all overspray contours, we calculate overspray. We need to consider two aspects in overspray calculation - 1) since the print length varies in different images, we must normalize the overspray with print length, and 2) overspray is not just the area of the splash around the print line, but the distance from the print line also matters. Specifically, for two identical contours, overspray is higher for the ones that are farther from the print line. Therefore, we defined the overspray as follows:

$$Overspray = \frac{\sum_C A \times y}{L} \quad (2.5)$$

where C is the list of all identified overspray contours, A is the area of the contour, y is the y-distance of the contour centroid from the print line, and L is the print length.

Finally, after performing all the analyses we can extract the processing print line. Figure 19 shows the processed print line with identified print line and overspray.

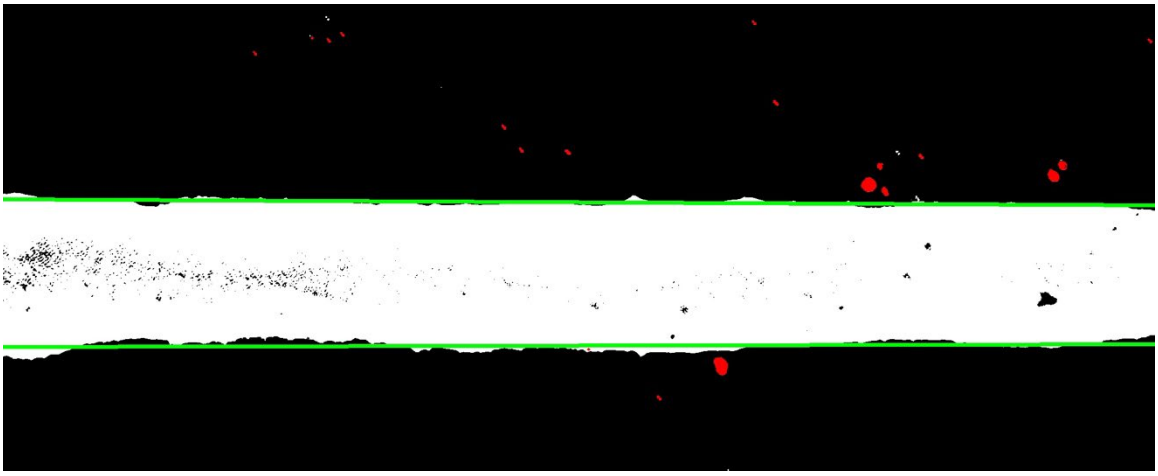

Figure 19. The processed print line (Print is white, the print line is green, and the overspray is red).

All the calculated parameters in the image processing stage are in pixel size. Therefore, we need to transform them into physical length scales, i.e., micrometers. To do so, we read the scale bar in the image and find out how many pixels exist in the scale bar, then we can calculate the physical length of each pixel. Image analysis of Figure 20 shows that are 202 pixels in the scale bar. Therefore, the physical size of each pixel is approximately 1 micrometer. Since we have imaged all pictures with the same microscope and magnification, the physical length of each pixel is consistent for all images.

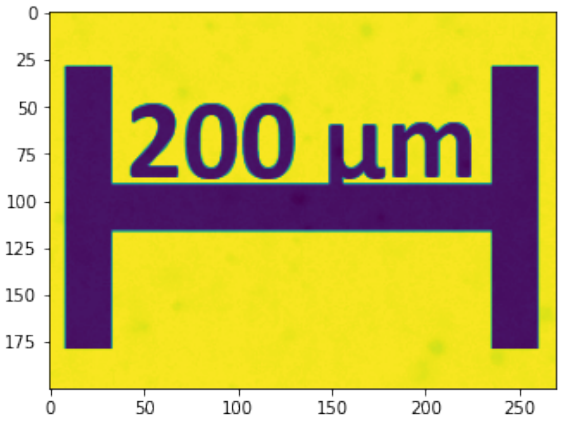

Figure 20. Image analysis shows the physical length of each pixel is 1 micrometer.

We defined the image processing part, that we described above, in a function called *ImageProcessing.* This function takes a print line image as an input and calculates the line width, overspray, and roughness. With that function, we can loop over all the cropped images and automatically calculate their morphology parameters. This will generate a tabular database that we need for the machine learning section. Figure 21 shows the *ImageProcessing* generated database.

	File Name	Print ID	Print Height	Nozzle Voltage	Drop Spacing	Line Width	Overspray	Roughness
0	1-800-25-8-L1_top.png	1	800	25	8	291.874868	0.0	42.771747
1	1-800-25-8-L2_bot.png	1	800	25	8	363.114495	31.00406	296.820248
2	1-800-25-8-L2_top.png	1	800	25	8	259.106667	0.0	123.661153
3	1-800-25-8-L3_top.png	1	800	25	8	260.506809	0.0	72.22255
4	1-800-25-8-R1_top.png	1	800	25	8	276.59084	6.325386	162.214829
...	...	...	...	...	...	...	...	...
2228	99-750-34-16-R2_bot.png	99	750	34	16	175.921831	0.0	158.359276
2229	99-750-34-16-R2_top.png	99	750	34	16	176.54047	0.0	16.05233
2230	99-750-34-16-R3_top.png	99	750	34	16	179.138182	0.0	18.593236
2231	99-750-34-16-R4_bot.png	99	750	34	16	176.129524	0.0	195.39744
2232	99-750-34-16-R4_top.png	99	750	34	16	176.195338	0.0	52.049856

Figure 21. The generated database after image processing.

CHAPTER FOUR: MACHINE LEARNING

In this section, we present two machine learning (ML) models to connect the processing parameters, i.e., nozzle voltage, cartridge height, and drop spacing, to morphological specifications, i.e., line width, roughness, and overspray. A typical procedure for all standard ML studies includes data cleaning, data pre-processing, algorithm development, model hyperparameters optimization, and model predictability assessment.

Data Cleaning

We performed data cleaning in two steps - 1) data cleaning on printed images, and 2) data cleaning after initial data pre-processing. In the first round of data cleaning, we closely studied all 2465 images and removed all improper images. Improper images that we removed at this stage were those pictures that were greatly different from their set. As we mentioned in database generation and image processing for each set of process parameters, we have 12 printed lines. If one or a couple of these images were damaged or totally different from the rest of the set, we removed them from the dataset. Figure 22 shows some samples of print images that we removed from the database. We have performed another set of data cleaning only for machine learning of overspray based on the exploratory data analysis and we will describe it in the exploratory data analysis section.

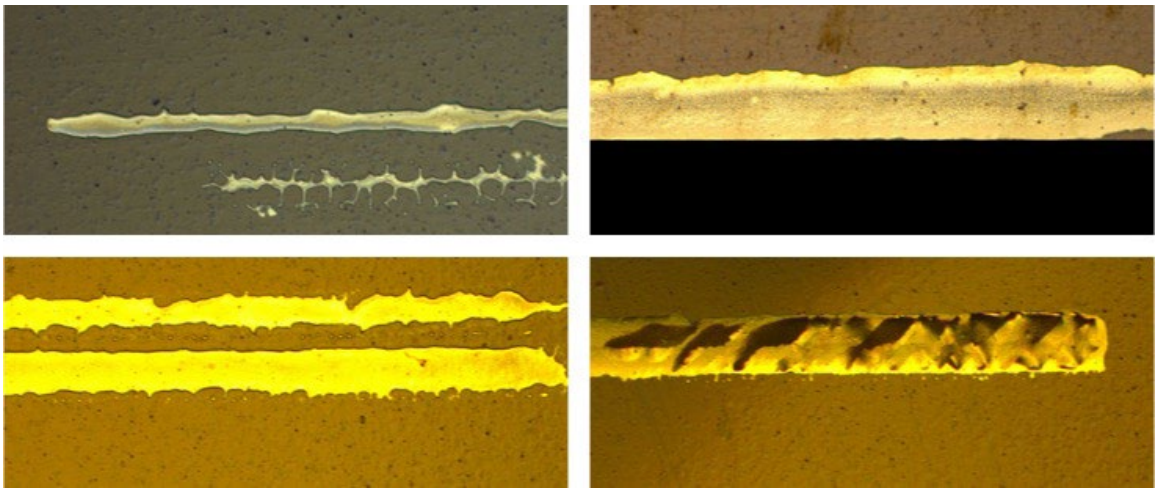

Figure 22. Some print images that we removed in data cleaning.

After removing the outlier images, we performed image analysis and predicted line width, roughness, and overspray for all images. As we originally discussed, we have 12 printed lines for each set of processing parameters, unless some images have been removed from the database in the data cleaning process. Therefore, we have several outputs for each set of processing parameters which is not recommended in ML. Thus, we averaged the target variables for the images that we produced for a single set of processing parameters. After this step, we have 205 unique data that are ready for ML. Figure 23 shows the machine learning-ready dataset generated in this project.

	Print ID	Print Height	Nozzle Voltage	Drop Spacing	Line Width	Overspray	Roughness
0	1	800	25	8	294	12	164
1	2	800	25	9	261	136	141
2	3	800	25	10	218	11	103
3	4	800	25	11	190	15	68
4	5	800	25	12	190	91	90
...	...	...	...	...	...	...	...
200	205	650	31	12	262	17	94
201	206	650	31	13	241	15	86
202	207	650	31	14	191	77	87
203	209	650	31	16	188	1	73
204	210	650	31	17	203	5	45

Figure 23. Machine learning-ready dataset.

Exploratory Data Analysis

After dataset construction, one important step before performing ML is the exploratory data analysis (EDA) (Tukey, 1977). EDA helps us to gain some insights into our dataset and understand the key characteristics of each entity in the dataset. EDA is helpful in - 1) understanding the dataset and fine-tuned cleaning, 2) giving a clear picture of features and their relationships, and 3) identifying the outliers. The first step in EDA is the statistical description of the dataset which is shown in Figure 24. Some important information from this table, in particular for target variables, are their minimum, maximum, mean, and standard deviation. For example, a quick glance shows that the line width and roughness are more well-behaved data than overspray. For a more detailed analysis, we have plotted the box plot and distribution of the target variables in Figure 25. Both plot box and histogram show that overspray has some outlier data points. Detailed analysis shows that about 185 data points have overspray less than 500 and 20 data points

have overspray more than 500. Since these prints are outliers just for overspray, we excluded them in the overspray ML part.

	Print ID	Print Height	Nozzle Voltage	Drop Spacing	Line Width	Overspray	Roughness
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	105.404878	736.097561	31.980488	12.453659	230.639024	245.614634	99.034146
std	60.627414	51.586499	5.080683	2.895915	43.512298	568.407660	30.853604
min	1.000000	650.000000	25.000000	8.000000	112.000000	0.000000	43.000000
25%	53.000000	700.000000	28.000000	10.000000	195.000000	23.000000	75.000000
50%	105.000000	750.000000	31.000000	12.000000	224.000000	72.000000	92.000000
75%	158.000000	800.000000	37.000000	15.000000	260.000000	227.000000	119.000000
max	210.000000	800.000000	40.000000	17.000000	391.000000	4762.000000	192.000000

Figure 24. Statistical analysis of the dataset.

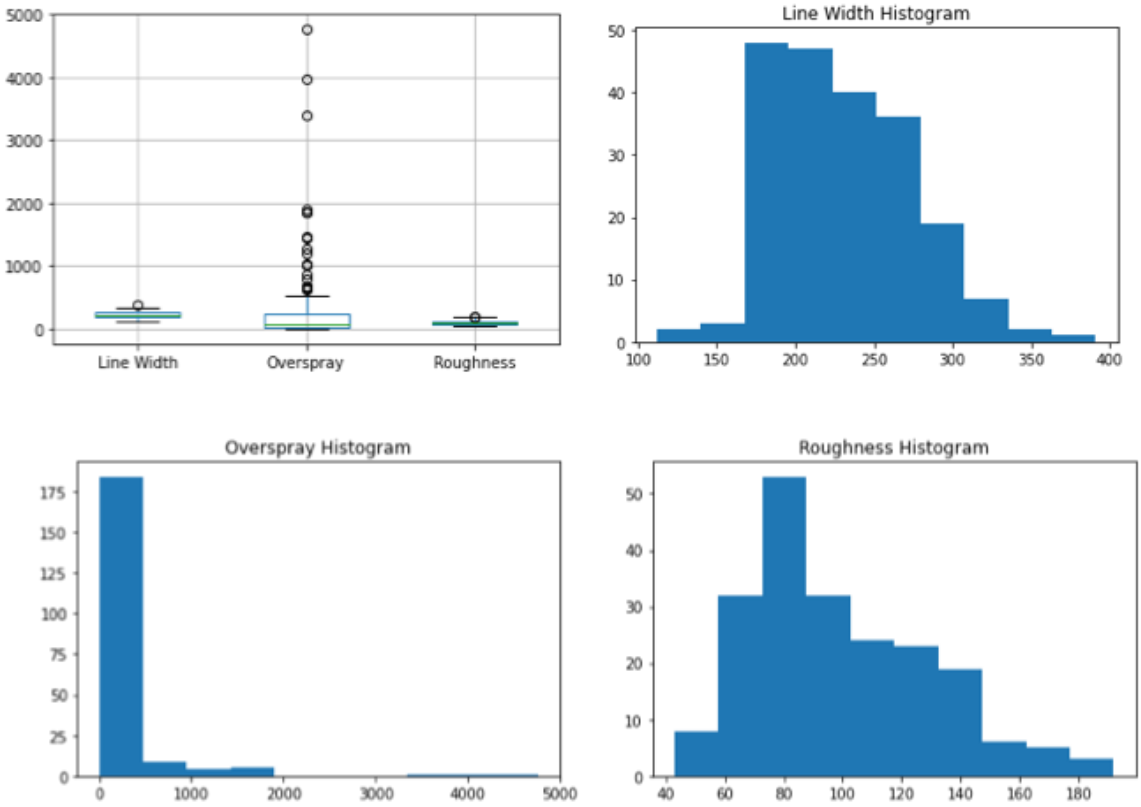

Figure 25. Box and histogram plots for line width, roughness, and overspray.

Machine Learning

After initial data cleaning and EDA, we are ready to perform ML modeling. IBM has this definition for machine learning "Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy." (IBM, 2021) The general workflow in all ML projects has three steps - 1) data gathering, 2) feeding the data into the model and model training, and 3) using the trained model for prediction. If we assume that the ML models are like functions that have some inputs and outputs, the inputs in ML are called features or descriptors, and outputs are called target variables or dependent variables. Data in ML can be either categorical or continuous (Ramezani, 2020). The data type determines the ML tasks. If the data are categorical the ML task is classification, and if the data are continuous the ML task would be regression. In this project we are dealing with numerical data, i.e., continuous, therefore, our ML task would be regression. We have 205 data points. This is a decent dataset size if the problem is not very complex or nonlinear. Generally, complex models, such as deep learning, have shown to be efficient in learning complex tasks but training those models requires a huge amount of data. For a couple of hundred data points, a Multilayer Perceptron Neural Network model could be a good choice (Delashmit & Manry, 2005).

Neural Network

The neural network (NN) is a machine learning algorithm that originally was inspired by mimicking the functionality of the human brain (McCulloch & Pitts, 1943). The architecture of NN has evolved over decades, but the key basic components are input nodes, hidden layer(s), and output nodes. In this project, the input features are three

processing parameters, i.e., cartridge height, nozzle voltage, and drop spacing, and the outputs are print morphology specifications, i.e., print line width, roughness, and overspray. Each node in the input layer connects to the nodes in the hidden layer. The values inside each node in NN, except the input layer that are known, are determined by $\sum(x_i w_{ij} + b_j)$ where the x_i is a vector containing values of the nodes of the previous layer, w_{ij} are weights, and b_j are biases (Bishop, 1995). To add nonlinearity to the NN, the $\sum$ term in each node goes through a nonlinear function called an activation function, otherwise, the NN would be a form of linear regression. There are several forms of activation functions, and generally, the activation function is considered a hyperparameter and users can choose the one that shows higher accuracy. Figure 26 shows the schematic of a NN with our input and output with one hidden layer, the same concept applies when we increase the number of hidden layers (Aggarwal, 2018).

One main task in ML is model training. Model training is analogous to teaching, which means before models can be used for prediction, we need to teach them. The way we teach the ML models is by showing the model some input and output from the ground truth dataset, and then, the model tries to fit its parameters in a way to connect the input to output with the highest accuracy. For the NN models, training is nothing except finding proper weights and biases. The standard NN algorithms do the weights and biases optimization by a technique called backpropagation. The mathematics behind backpropagation can be found in all standard ML books (Aggarwal, 2018), but in nutshell, in NN optimization with backpropagation we start with random weights and biases, and with them, we connect the input layer to the output layer. By comparing the predicted outputs with the ground truth, we measure the error and from that, we move

backward and modify all weights and biases in a way to reduce the error. We loop this cycle until we get the accuracy that we desire or a stop condition hits, such as the limit we put on the number of iterations.

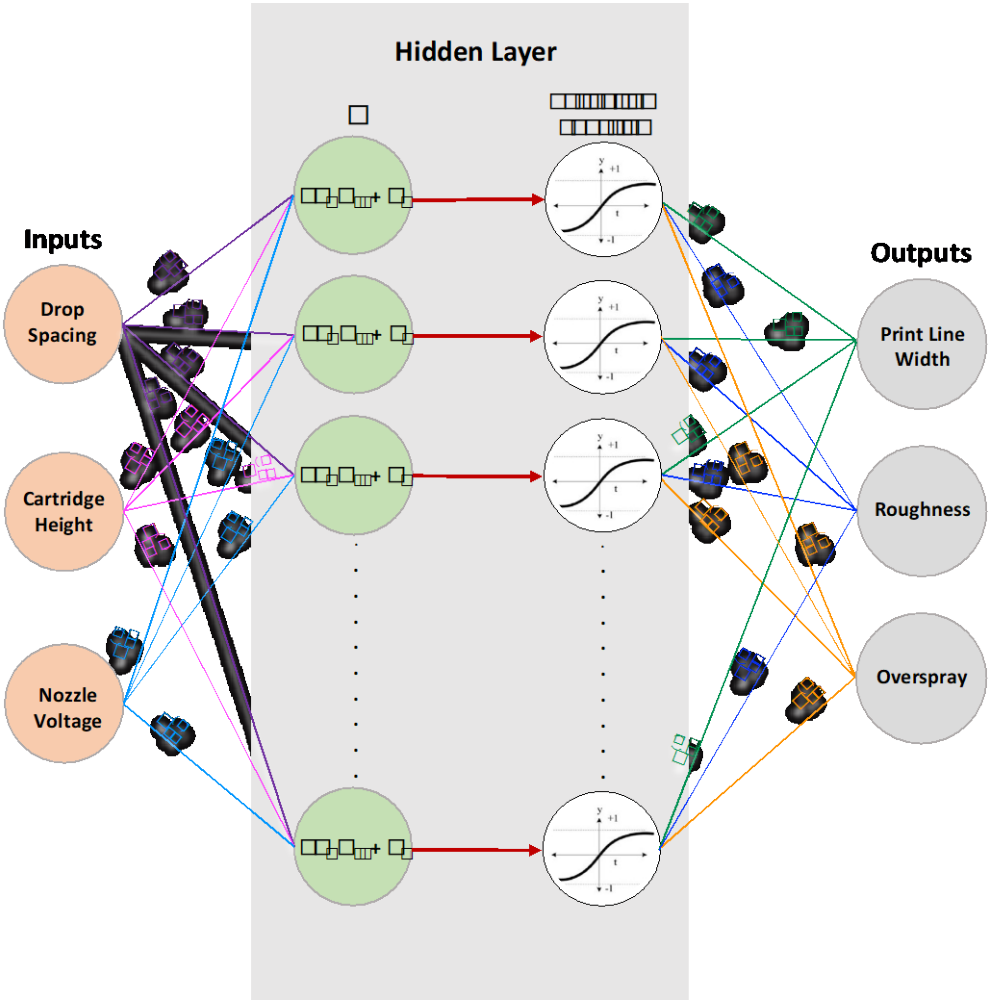

Figure 26. The architecture of a basic NN with one hidden layer.

After model training, we test the model predictability by testing its performance on testing data. There is no solid ratio for training and testing data selection, but a general guideline is to put 70-80% of the data in the training set and 30-20% in the testing set. The model performance against training and testing sets is a good indicator of whether the model is overfitted or not. Overfitting occurs when a model has "memorized" the

training data, i.e., it has not learned the pattern of the data. If the model accuracy for the training set is very high but its accuracy for the testing data is poor, it indicates that the model is overfitted. Underfit happens when the model accuracy is poor for both training and testing datasets and a good fit is when the model has similar (hopefully good) accuracy for both training and testing datasets. Figure 27 shows some examples of overfit, good fit, and underfit models (*Underfitting vs. Overfitting*, 2022).

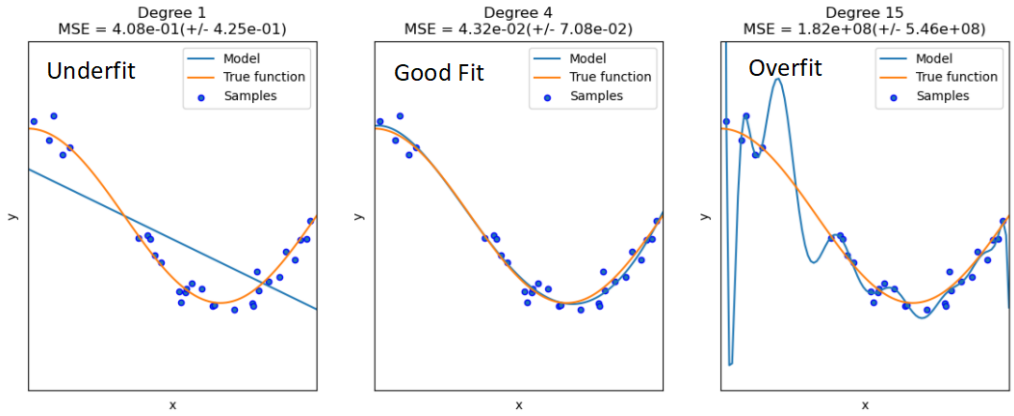

Figure 27. Underfit vs. Good fit vs Overfit in ML (*Underfitting vs. Overfitting*, 2022).

Hyperparameters Optimization

Hyperparameters are the ML model parameters that can make the model either more efficient or accurate. Hyperparameters vary depending on the model of choice. For NN models the key hyperparameters are the number of the hidden layers, the number of nodes in each layer, the learning rate, the activation function, and the solver. The number of hidden layers and their nodes are clear concepts, here we explain the remaining hyperparameters briefly.

Learning rates determines the amount that we change the weight and biases during the backpropagation (Werbos, 1994). The learning rate has a small positive value, often in the range between 0.0 and 1.0. The learning rate controls how quickly the model

is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs. A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

Activation functions, as we mentioned earlier, are added to the NNs to introduce non-linearity to the model. There are several forms of activation functions, but the most common ones are identity, logistic (also known as sigmoid), hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU) (Aggarwal, 2018). Figure 28 shows the plots and functions of the common NN activation functions.

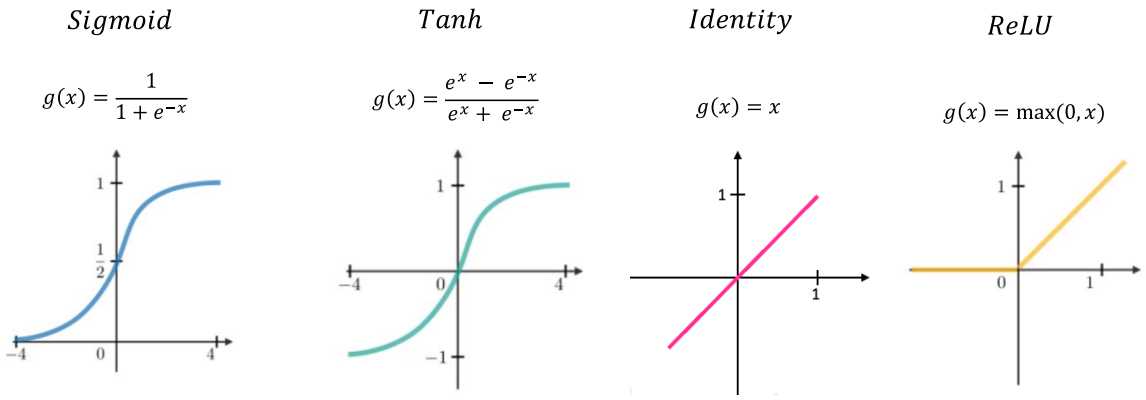

Figure 28. Common activation functions that are used in NN.

Solvers or optimizers are the mathematical algorithms that are used to find the optimized weights and biases. Scikit-learn has three solvers for multilayer NNs including 'lbfgs', 'sgd', and 'adam'. 'lbfgs' is an optimizer in the family of quasi-Newton methods, 'sgd' refers to stochastic gradient descent, and 'adam' refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba (Kingma & Ba, 2017; *Sklearn.Neural_network.MLPClassifier*, 2022). The solver 'adam' works pretty well on

relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score. For small datasets, however, ‘lbfgs’ can converge faster and perform better.

Tunning the hyperparameters for best performance is challenging and the initial selection is based on the code developer experience. Scikit-learn has a specific function, GridSearchCV (GridSearchCV, 2022), that enables us to grid the hyperparameters space to find the best set of hyperparameters. GridSearchCV finds the best combination of hyperparameters based on the scoring criteria that we are interested in, such as root mean square error or R2-score (Ott & Longnecker, 2015).

K-fold Cross-validation

The final concept that is important to cover before model training is the K-fold cross-validation (Ozdemir, 2016). K-fold cross-validation is a critical part of model training and the way it works is that we divide the training dataset into k-buckets, then train the model for (k-1) buckets and test it against the left-out one. We repeat this process k times and then average the computed values in the loop. Typical values of k are 5 or 10. Figure 29 shows a schematic for the process of 5-fold cross-validation (scikit-learn, 2022).

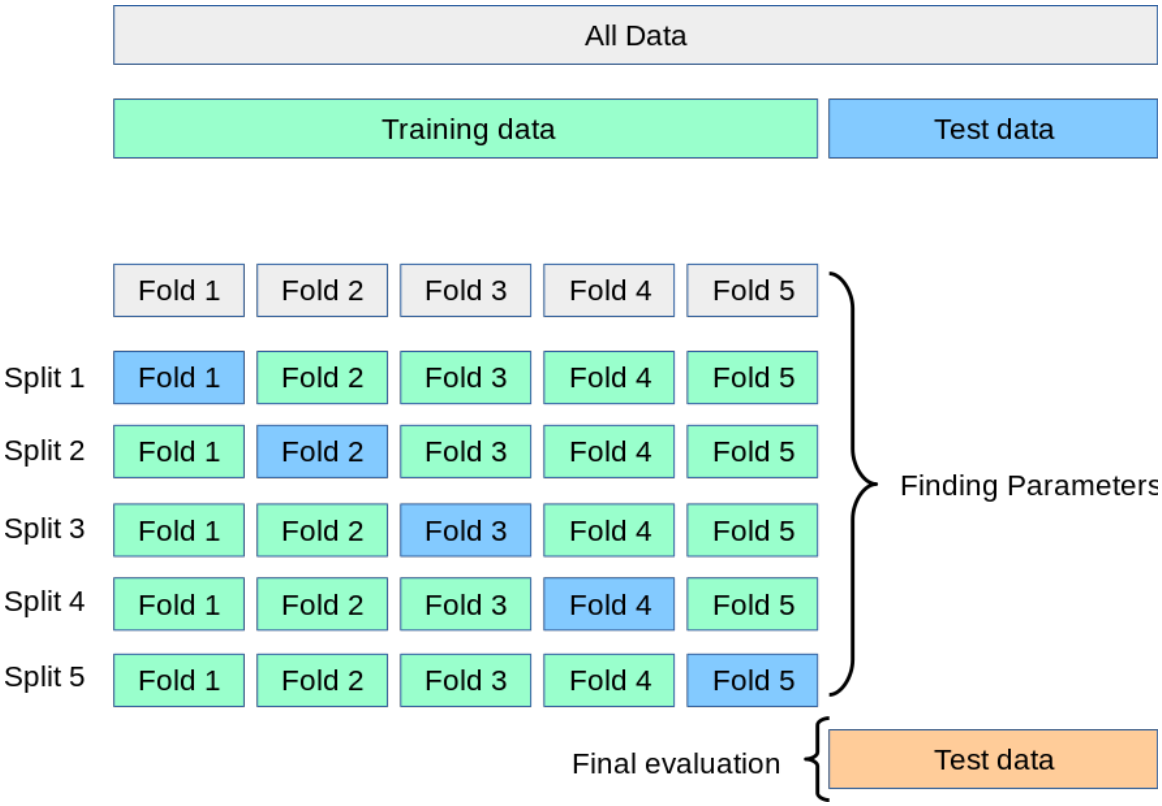

Figure 29. Schematic for 5-fold cross-validation (scikit-learn, 2022).

Results

When we have a dataset with multiple features, it is important to perform data normalization before ML. In particular, when we are dealing with features that are different in values, for example in cases when some features are orders of magnitude higher than the others. If we do not normalize the data, there is a chance that the higher value features show higher impacts on the results. There are several normalization techniques. In this project, we normalized the features in a way to have zero mean and unit variance through the following formula (Dutka & Hansen, 1991)

$$x_i^{(n)} = \frac{x_i - \mu_i}{\sigma_i} \qquad (4.1)$$

where x_i is the feature, $x_i^{(n)}$ is the normalized feature, μ_i is the mean, and σ_i is the standard deviation. Normalization needs to be performed for both training and testing datasets. We note that the normalization must be consistent between training and testing datasets. Therefore, the testing data set is normalized by the mean and standard deviation of the training data.

Our investigations showed that ReLU performs the best among the activation functions. Also, learning rates of 0.001 and 0.0001 works best for our data. My studies showed that the NN architecture plays an important role in model accuracy. We tested different forms of NN architecture from one to five hidden layers with different numbers of nodes. Very simple architectures did not converge well, and complex architectures were overfitted, i.e., they had very high accuracy for training but low accuracy for testing. The best NN architecture that provided the lowest root mean square error (RMSE) for all three target parameters, i.e., line width, roughness, and overspray, was a NN with one hidden layer with 30 nodes.

Figure 30 shows the parity plot for the line width for both training and testing datasets. RMSE for training and testing were 21.9 and 23.7 um, respectively. R2-score for training and testing were 0.73 and 0.72, respectively.

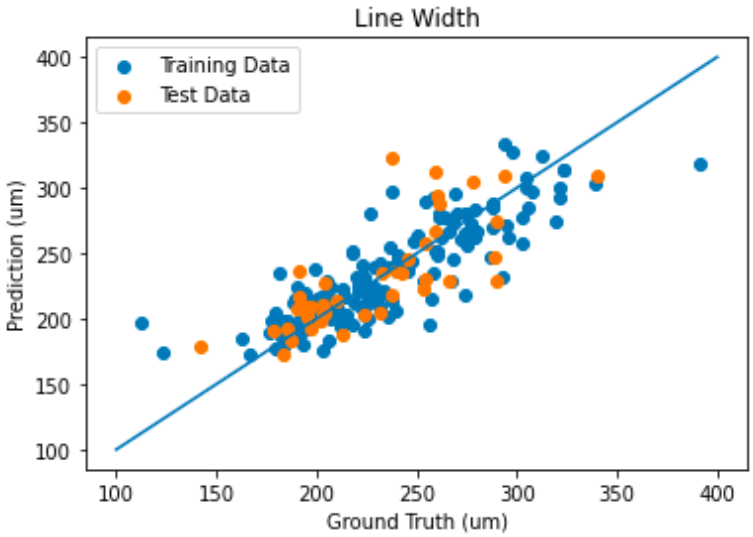

Figure 30. Parity plot for line width for both training and testing datasets.

Figure 31 shows the parity plot for roughness for both training and testing datasets. RMSE for training and testing datasets were 18.9 and 19.4 um. R2-score for training and testing data sets were 0.63 and 0.55 respectively.

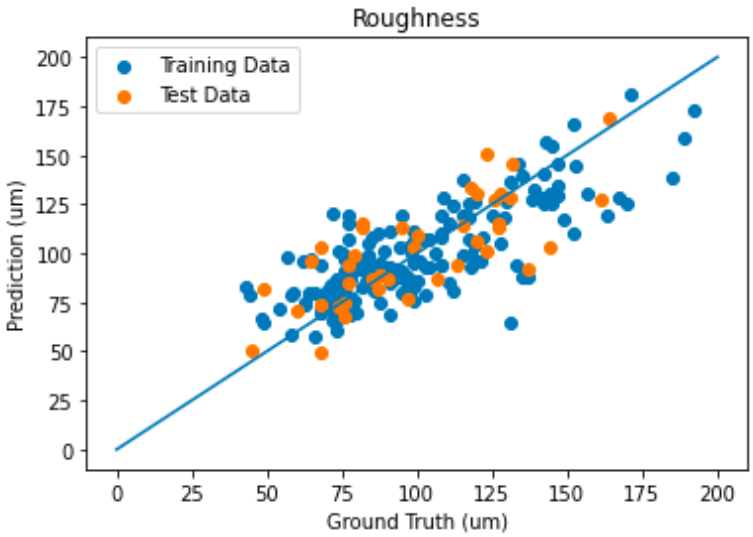

Figure 31. Parity plot for roughness for both training and testing datasets.

Finally, Figure 32 shows the parity plot for overspray for both training and testing datasets. RSME for training and testing datasets were 67 and 129 um^2, respectively. R2-scores for training and testing were 0.67 and -0.32, respectively.

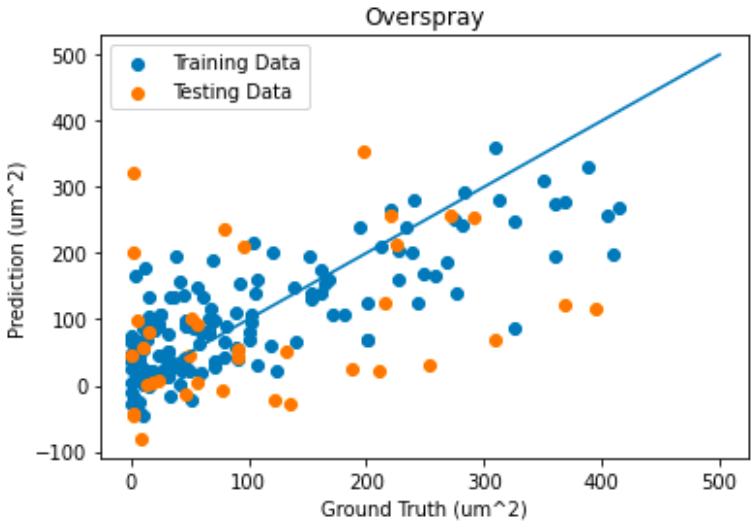

Figure 32. Parity plot for overspray for training and testing datasets.

Analysis of the results shows that the model performance is the best for the line width, then roughness, and then overspray. In general, IJP with a Dimatix machine shows robust performance in printing lines with consistent line width. However, when it comes to overspray it seems that the machine performs randomly and none of the processing parameters that we considered in this study can explain the overspray very well. Figure 33 depicts one example of the strange behavior of overspray in one set of prints. The printed lines in Figure 33 have a similar cartridge height, 800 um, and nozzle voltage, 31 Volt, but drop spacings are 9, 10, and 11 um. Interestingly, we observe that prints with 9 and 11 um drop spacing have zero oversprays, while the print with drop spacing of 10 um shows considerable overspray. These kinds of behaviors are the main reasons why our ML model did not perform well in overspray prediction.

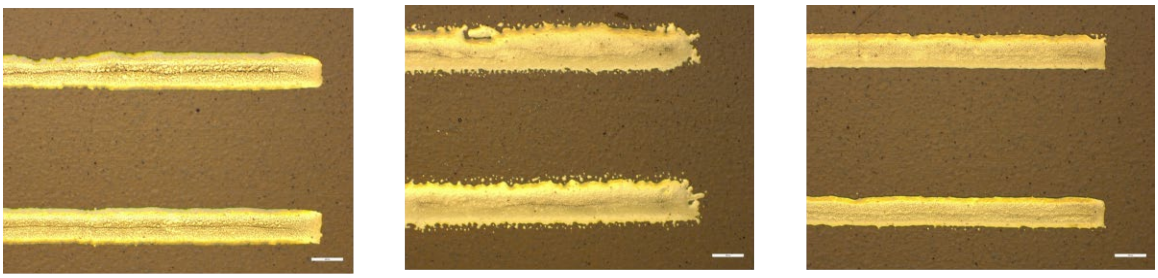

Figure 33. The irregular behavior of the print overspray by varying the drop spacing.

Feature Importance

Some interesting results that ML models can give us are feature importance. Feature importance shows the level of contribution of each input parameter in a specific target data. There are several ways to get the feature importance such as principal component analysis or random forest ML. In this project, we used the random forest to get the feature importance. Figure 34 shows the feature importance of line width. The results clearly show that drop spacing has the most important role in line with determination and the role of cartridge height and nozzle voltage is minimal.

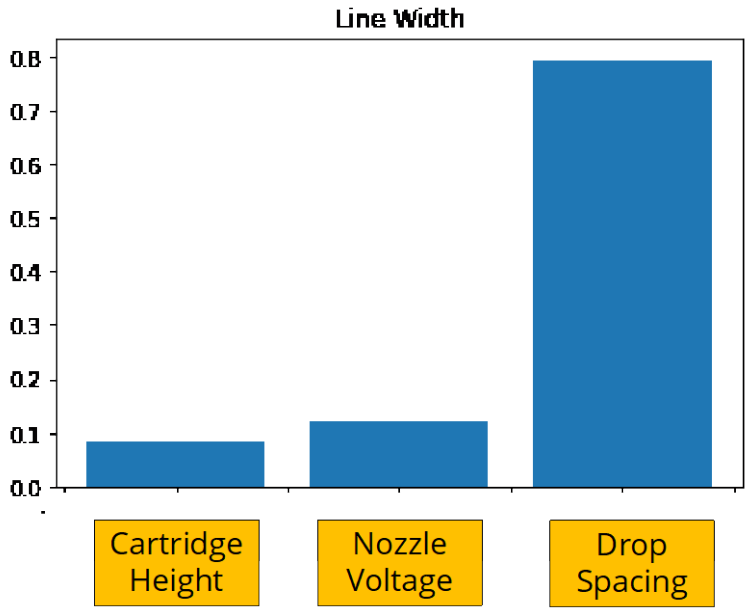

Figure 34. Feature importance scoring for line width.

Figure 35 shows the feature importance for roughness. The drop spacing is also the dominant processing parameter for roughness, but the roles of nozzle voltage and cartridge height are also significant.

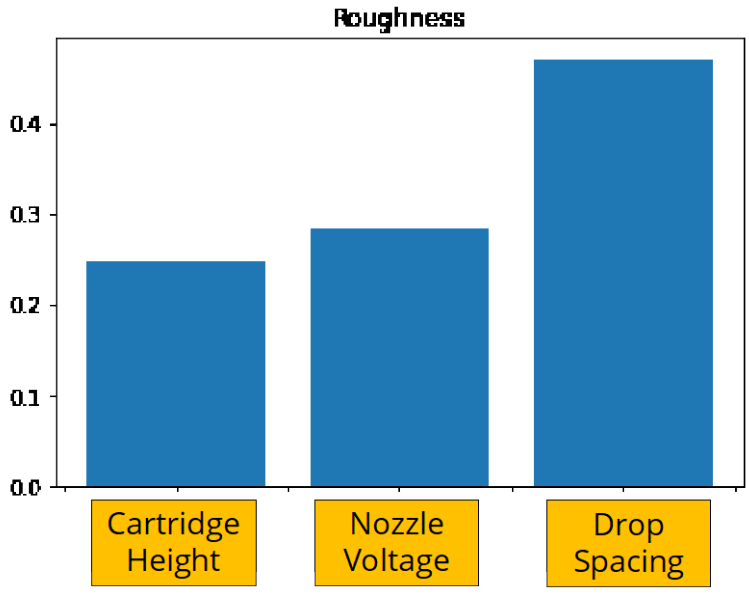

Figure 35. Feature importance scoring for roughness.

Finally, for overspray, Figure 36 shows that the cartridge height is the dominant processing parameter, but nozzle voltage and drop spacing have also significant roles.

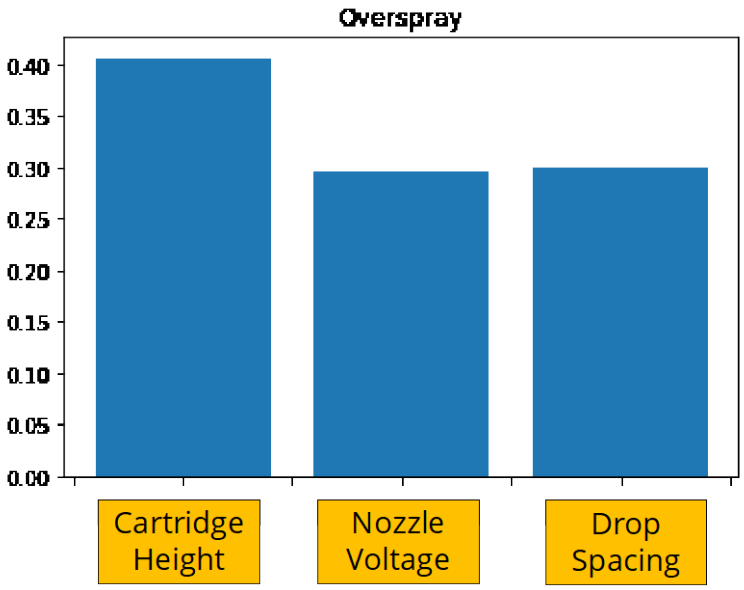

Figure 36. Feature importance scoring for overspray.

CHAPTER FIVE: CONCLUSION

Printed electronics have recently experienced tremendous growth and attracted considerable attention both in industry and academia. Among different forms of printed electronics technologies, Inkjet Printing (IJP) is popular because it can produce high-resolution patterns with minimal waste in a digital process. Despite all advantages of IJP, the technology is still in its infancy. The parameter space of ink and processing prevents the IJP to become a reliable mass production manufacturing technology for printed electronics. To address this challenge, we developed a machine learning model to construct the process-property linkages for IJP. Specifically, in this project, we focused on cartridge height, nozzle voltage, and drop spacing as processing parameters and print line width, roughness, and overspray as print morphology properties.

For this model, we developed an extensive experimental database for a wide range of processing parameters. Then we developed an image processing code that reads the experimental images and calculates their linewidth, roughness, and overspray automatically. We have developed some Python scripts that automate the workflow and after reading all experimental images produce a machine-learning-ready CSV file database. Finally, we performed data cleaning by removing bad images. Exploratory data analysis showed that line width and roughness are more well-behaved than the overspray. For overspray, we noticed some outlier data that we removed during ML modeling of overspray.

After tabular database generation, we developed a multilayer perceptron neural network model to enable print morphology prediction based on processing parameters. Our studies showed that a NN with one hidden layer with 30 nodes predicts the target parameters the best. The NN model accuracy is reasonable for line width and roughness but not very promising for overspray. The main reason for the model's poor performance on overspray could stem from our limited dataset, the Dimatix unreliable printing, or external factors that we did not consider in the model. This is an ongoing project, and our team plans to print 400 more samples. Therefore, we can produce much more accurate models by increasing the training dataset.

REFERENCES

Aggarwal, C. (2018). *Neural Networks and Deep Learning*. https://link.springer.com/book/10.1007/978-3-319-94463-0

Anyfantakis, M., & Baigl, D. (2015). Manipulating the Coffee-Ring Effect: Interactions at Work. *Chemphyschem : A European Journal of Chemical Physics and Physical Chemistry*. https://doi.org/10.1002/cphc.201500410

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

BS EN ISO 4287. (2000). *BS EN ISO 4287: 2000 GEOMETRICAL PRODUCT SPECIFICATIONS (GPS)* -.

Delashmit, W. H., & Manry, M. T. (2005). Recent developments in multilayer perceptron neural networks. *Proceedings of the Seventh Annual Memphis Area Engineering and Science Conference, MAESC*.

Dimatix. (2022). *Dimatix Materials Printer DMP-2850 | Fujifilm [Serbia]*. https://www.fujifilm.com/rs/en/business/inkjet-solutions/inkjet-technology-integration/dmp-2850

Dutka, A. F., & Hansen, H. H. (1991). *Fundamentals of data normalization*. Addison-Wesley Longman Publishing Co., Inc.

GridSearchCV. (2022). *Sklearn.model_selection.GridSearchCV*. Scikit-Learn. https://scikit-learn/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Hart, P. E., Stork, D. G., & Duda, R. O. (2000). *Pattern classification*. Wiley Hoboken.

Heinzen, C., Berger, A., & Marison, I. (2004). Use of Vibration Technology for Jet Break-Up for Encapsulation of Cells and Liquids in Monodisperse Microcapsules. In V. Nedović & R. Willaert (Eds.), *Fundamentals of Cell Immobilisation Biotechnology* (pp. 257–275). Springer Netherlands. https://doi.org/10.1007/978-94-017-1638-3_14

Hoath, S. D. (2016). *Fundamentals of Inkjet Printing: The Science of Inkjet and Droplets*. John Wiley & Sons.

Hu, G., Kang, J., Ng, L. W., Zhu, X., Howe, R. C., Jones, C. G., Hersam, M. C., & Hasan, T. (2018). Functional inks and printing of two-dimensional materials. *Chemical Society Reviews*, *47*(9), 3265–3300.

IBM. (2021, November 5). *What is Machine Learning?* https://www.ibm.com/cloud/learn/machine-learning

ISO 4287. (1997). *ISO 4287:1997*. ISO. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/01/01/10132.html

Khan, Y., Thielens, A., Muin, S., Ting, J., Baumbauer, C., & Arias, A. C. (2020). A new frontier of printed electronics: Flexible hybrid electronics. *Advanced Materials*, *32*(15), 1905279.

Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization* (arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980

Malkin, A. Y., & Isayev, A. I. (2022). *Rheology: Concepts, methods, and applications*. Elsevier.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133.

McKinley, G. H., & Renardy, M. (2011). *Wolfgang von Ohnesorge*. https://doi.org/10.1063/1.3663616

Nanotechnology in Flexible Electronics. (2012, December 31). AZoNano.Com. https://www.azonano.com/article.aspx?ArticleID=3164

NovaCentrix. (2022). JS-A191 Silver Nanoparticle Ink. *NovaCentrix*. https://www.novacentrix.com/product/js-a191-silver-nanoparticles/

OpenCV. (2022a). *OpenCV*. OpenCV. https://opencv.org/

OpenCV. (2022b). *OpenCV: Image Filtering*. https://docs.opencv.org/4.x/d4/d86/group__imgproc__filter.html

Opencv/opencv. (2022). [C++]. OpenCV. https://github.com/opencv/opencv (Original work published 2012)

Ott, R. L., & Longnecker, M. T. (2015). *An introduction to statistical methods and data analysis*. Cengage Learning.

Ozdemir, S. (2016). *Principles of data science*. Packt Publishing Ltd.

Ramezani, N. (2020). Modern Statistical Modeling in Machine Learning and Big Data Analytics: Statistical Models for Continuous and Categorical Variables. In *Handbook of Research on Big Data Clustering and Machine Learning* (pp. 135–151). IGI Global.

Saengchairat, N., Tran, T., & Chua, C.-K. (2017). A review: Additive manufacturing for active electronic components. *Virtual and Physical Prototyping*, *12*(1), 31–46. https://doi.org/10.1080/17452759.2016.1253181

scikit-learn. (2022). *3.1. Cross-validation: Evaluating estimator performance*. Scikit-Learn. https://scikit-learn/stable/modules/cross_validation.html

Sklearn.neural_network.MLPClassifier. (2022). Scikit-Learn. https://scikit-learn/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Tukey, J. W. (1977). *Exploratory data analysis* (Vol. 2). Reading, MA.

Underfitting vs. Overfitting. (2022). Scikit-Learn. https://scikit-learn/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Werbos, P. J. (1994). *The roots of backpropagation: From ordered derivatives to neural networks and political forecasting* (Vol. 1). John Wiley & Sons.

Zhang, H., & Moon, S. K. (2021). Reviews on Machine Learning Approaches for Process Optimization in Noncontact Direct Ink Writing. *ACS Applied Materials & Interfaces*, *13*(45), 53323–53345. https://doi.org/10.1021/acsami.1c04544