VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Diagnosing Cervical Cancer with Artificial Intelligence

Kevin Hoong

Supervisor: Alexander Doronin and Mengjie Zhang

Submitted in partial fulfilment of the requirements for
Bachelor of Software Engineering with Honours.

Abstract

The Ecole Polytechnique in France has made major developments in being able to diagnose cervical cancer through biomedical imaging. A comprehensive data set has been gathered and researched by Ecole Polytechnique using a method called Mueller Polarimetry. Each 5x5mm tissue sample processed by this method is comprised of 600x800x16 matrices known as the Mueller Matrix. Prior statistical analysis of this data set has achieved an average AUC of 0.95 (where an AUC=1 is perfect detection and AUC=0.5 is guessing)[1]. The purpose of this project is to explore Artificial Intelligent methods to classify CIN2-3 (cancerous) and healthy regions of each Mueller Matrix. From the 24 sample measurements of Mueller Matrices provided, 57925 cancerous and 133908 healthy pixels have been extracted and organized into a training data set. By exploring this data using multiple machine learning techniques, 4 different classification models have been trained with the highest recorded AUC score of 0.98 using a multi-layer perceptron. A visualization pipeline has been built to show the cancerous and healthy regions marked by our model to assist in biomedical testing of new samples in the future.

Acknowledgments

I would like to thank my supervisor Dr. Alexander Doronin for the encouragement and mentorship over the course of this project. And also Dr. Mengjie Zhang for guidance in the Machine Learning approaches.

I would also like to thank Dr. Angelo Pierangelo, Dr. Tatiana Novikova, and other health professionals from the Ecole Polytechnique in France for providing the dataset, and guidance throughout the completion of this project.

Contents

Figures

Covid-19

The Covid-19 pandemic caused several delays in the progress towards my project. One of these causes was due to the impact covid-19 had on France, which went into lockdown slightly before we did. This made delays in obtaining the dataset because the researchers at Ecole Polytechnique could not access their facilities. We were able to obtain the dataset through somebody having it on their laptop once the lockdown settled down in France. At the start of Trimester 2 we were finally able to meet with Tatiana and Angelo through Zoom, where we discovered the initial data shared with us was not the Mueller data required to perform our training. This meant the progress towards training the machine learning models couldn't begin until early Trimester 2.

Another problem I faced due to Covid-19 was the lack of study space and resources. This is due to having to share a room at home and having unstable internet connection which cut out multiple times a day I found it very difficult to stay locked into a task as there would always be interruptions. Fortunately I was able to get lab access in the middle of level-3 so I could go into University to complete work but due to built-up assignments I could not complete at home due to not having the correct software, and having to travel 1 hour each way to get to university, not a lot of this time was spent on the project during the lockdown.

Since entering level-2 and level-1 I found it much easier to access the resources to make progress towards the project due to University being open, however the initial impact of Covid-19 put me behind in terms of work done.

Chapter 1

Introduction

Machine Learning has proven to be able to find solutions to complex modern world problems by employing various statistical and probabilistic techniques to 'learn' from given data of past examples[2]. This makes machine learning attractive in the field of biomedical imaging to distinguish and learn features that are not always apparent when health professionals observe a sample of a tissue disease such as cancer.

The Ecole Polytechnique in France has made seminal developments toward cervical cancer diagnostics by creating a biomedical imaging device using Mueller Polarimetry, which contains 24 samples of 600x800x16 dimensional matrices of cervical cancer tissue. The AUC score is the evaluation metric used for determining the accuracy of past solutions, which is calculated using true positive and true negative results. Through achieving a high detection performance of AUC=0.95 using the 3 smallest eigenvalues and the scalar retardance, which are parameters calculated from the Mueller matrix, Ecole has identified that the Mueller Polarimetry data contains valuable information for pre-cancerous cervical cancer detection[1]. Through these findings, we believe that Artificial Intelligence will be able to apply different learning approaches to further discover the correlation of the 16 dimensions of the Mueller Matrices and aid towards the diagnostic for pre cervical cancer detection. As part of a larger French-New Zealand partnership program between The Victoria University of Wellington and Ecole Polytechnique, the purpose of this exploratory project is to bridge the gaps between computer science and biomedical imaging by developing a visualization pipeline and exploring the correlation of the data by analyzing its performance through different machine learning models.

1.1 The Problem

Although a very high score of AUC=0.95 has already been acquired using the data set, the solution Ecole found requires further post-processing of the Mueller matrix to obtain the coherency matrix, and scalar retardance using Lu Chipman decomposition, which are parameters calculated from the depolarizing and non-depolarizing parts of Mueller matrix respectively[1]. The data provided by Ecole is yet to be explored using artificial intelligence, where it may be possible of obtaining accurate classifications using the 16 values of the Mueller matrix alone. If we can achieve a higher AUC score using only the Mueller matrix data, the post-processing step of calculating new parameters may not be needed. This would ultimately lead to a more automated in situ diagnostic method to classify future data which is currently a lengthy process that requires multiple human resources.

1.2 Goals

The goal of this project is to process the required data from the 16 dimensional Mueller matrices of 24 cancerous samples into a format that can be used to train machine learning models using the masked regions provided by Ecole. Currently, these samples are diagnosed by cutting into the tissue and performing histological analysis. If an accurate model can be built using Mueller polarimetry data, there is proof that a step towards less invasive, expensive, and time-consuming cervical cancer diagnostics can be made through the implementation of Artificial Intelligence. The basis of our evaluation can be made by proving that our machine learning model can classify the same cancerous and healthy regions as the health professionals in more efficient time, and using fewer human resources.

Each sample of the data set contains multiple .dat and .mat files that contain different matrices to highlight healthy and diseased regions. The development of a visualization pipeline is necessary to acquire an understanding of the shape and format of these data files before I can begin extracting features from the samples. Upon achieving a good classification accuracy, presenting a machine learning model to Biomedical Scientists isn't likely to be useful, therefore another goal of the visualization pipeline is to plot the new diseased and healthy regions classified by my model. If my machine learning model obtains a higher AUC score than 0.95 and the resulting output can be visualized, Ecole Polytechnique can perform a black-box comparison between the predicted unhealthy and cancerous regions to the ones manually acquired by the pathologist.

1.3 Overview

A visualization pipeline has been implemented which aids in viewing the shape of the data throughout each step of reaching the solution. An initial visualization of the intensity data multiplied by the masked data gives a basis to begin data exploration by proving the plotted images contain the same marked regions as the images provided by Ecole [1]. Further visualization of the Mueller matrix data allowed us to make critical decisions on which features of the full matrix we decided to use for our training. This aids in each step of data pre-processing and training our machine learning models as it provides more information about the sample to the human-eye compared to the matrix data. Upon receiving a good classification accuracy that was obtained after training and evaluating a model, the visualization pipeline plots the classified sample to provide new information to the professionals at Ecole.

Multiple machine learning models have been evaluated on the data set to obtain the model which gives us the highest sensitivity and specificity, which are terms to define the accuracy of the classification results in biomedical imaging. This comes from the basis that in the past, machine learning models were either very intelligible but inaccurate, or very accurate but unintelligible, leading to models either learning incorrect features or learning features that do not apply to the task at hand [3]. Specificity can be defined as the number of true negatives where sensitivity can be defined as the number of true positives [4], this means that even if the model contains a high accuracy, if there is low sensitivity or specificity, the model is not appropriate for real-world use as it can lead to incorrect diagnostics.

The 4 machine learning models evaluated in this report are a Decision Tree, Multilayer Perceptron, 1D CNN (Convolutional Neural Network), and 2D CNN. From evaluating these models, I found that the multi-layer perceptron obtained the highest sensitivity and speci-

ficity when predicting the Mueller polarimetry data at AUC = 0.98. This is a strong indication that the data set correlates between the 16 channels of the Mueller matrix that can be trained with different configurations of Artificial Intelligence. Further detail of the configurations of this multi-layer perceptron and alternative machine learning models will be explained in the design section whereas more analysis of the resulting evaluation metrics will be discussed in the evaluation section.

Chapter 2

Background and Related Work

After reading the Ecole Polytechnique publications on the data set, and other publications regarding biomedical imaging for cancer diagnosis, there is evidence that there's a strong relationship between the biomedical imaging and machine learning. While the current solutions made towards the problem use different parameters derived from the Mueller Matrix, a machine learning model can contribute to the goal by learning additional features through automation. However, to understand how to build our machine learning model we require an understanding of the background and format of the data, as well as the methods that have already been applied.

2.1 Classification Problem

Initial reading of the project outline and publication for 'Polarimetric measurement utility for pre-cancer detection from uterine cervix specimens[1] indicated that the goal was to classify whether an entire sample is cancerous or healthy.

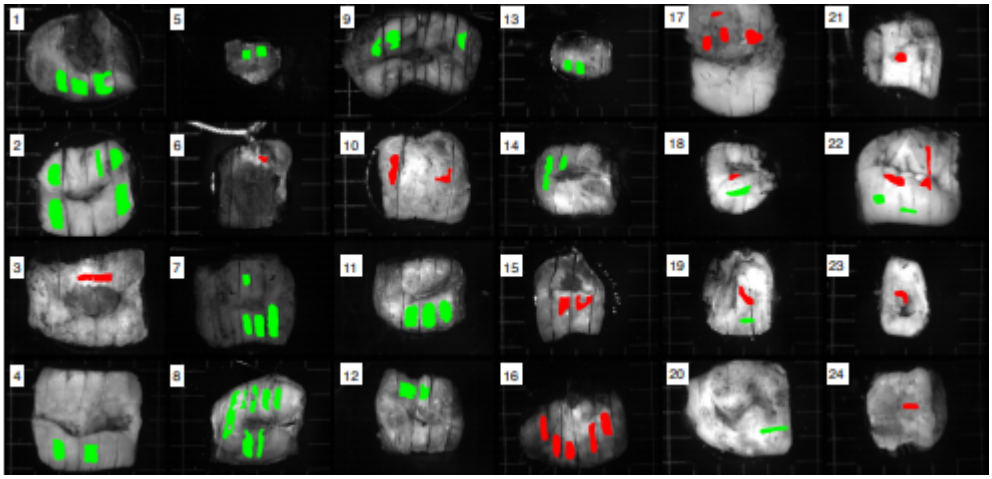


Figure 2.1: Intensity images of 24 excised cervical samples labeled with results from histopathology: CIN2-3(red), healthy(green) tissue.

By observing the images of the pathologist masks, an assumption was made that the samples with green regions were healthy and the samples with red regions were cancerous. Instead of referring to these regions as cancer, the publication uses the terminology CIN2-3. From figure 2.1, 59000 CIN2-3 and 135000 healthy pixels are labeled by Ecole. The definition of CIN2-3 is an abnormal cell that is caused by certain types of human papillomavirus and is

found when a cervical biopsy is done. While CIN2-3 is not cancer, it may become cancer and spread to nearby normal tissue if not treated [5]. Upon meeting with Tatiana and Angelo from Ecole Polytechnique, I found out that all 24 samples were taken from patients who already had cancer. This means that the classification problem for this project is not to classify whether a sample is cancerous or healthy, but to classify CIN2-3 and healthy regions within samples to aid early-cervical cancer diagnostics. This changes the way that we approach the image classification problem, by observing figure 2.1 we can see the data allows for more specific classification of individual areas of the image using the overlaying masks.

2.2 Data Background and Format

The development of biomedical imaging has proven to benefit the diagnostic of cervical cancer, with the most widely used method for reliable cancer diagnosis being excisional tissue biopsy. While effective, this process is also invasive, time-consuming, and expensive, leading to the development of Mueller Polarimetry[6]. The introduction of this biomedical imaging method is capable of label-free visualization of tissue pathology without requiring extensive sample preparation[7]. Because of this, Mueller Polarimetry is a solution to improving the effectiveness and reducing the invasiveness of traditional pre-cancerous diagnostic methods.

The data is captured using "an incoherent white light used to illuminate the sample delivered by an LED source collimated by a convex lens and then passed through a Polarization State Generator(PSG)[8]" the light backscatter by this sample is then passed through an Analyzer which samples the image through a CCD camera. The classes can then be assigned because CIN1 - CIN3 can be identified based on the thickness of the epithelium tissue. The Mueller matrix of the sample at each wavelength is reconstructed from sixteen images acquired from the combinations of four different input and output polarization states measured at a resolution of 600x800. The format to refer to these 16 channels are m1:1 to m4:4 to represent the input and output states. The resulting output data to have a shape of 600x800x16.

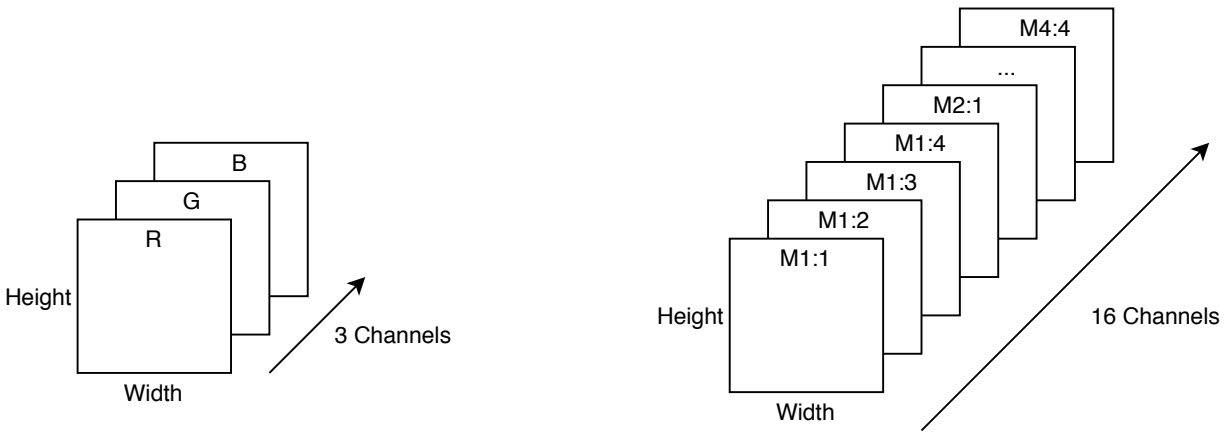


Figure 2.2: Example of traditional RGB image shape compared to Mueller Matrix shape.

To acquire a full-rank linear system in which a Mueller matrix can be constructed, a minimum of sixteen independent measurements are required which is the number of channels we can observe in the data [1]. Comparing the shape of this data to traditional RGB images we obtain a 600x800x16 where the 16 channels contain polarization data rather than color values. The data set contains intensity and Mueller data measured at 450, 550, and 600 nm known as the wavelength, which is the intensity of the incoherent white light used to measure the data. The radiance images shown in Fig. 1 is measured at 550 nm. Ecole has found that the specimens measured at 450 nm achieved the best sensitivity and specificity for optical diagnostics, but performs worst for statistical analysis due to signal-to-noise ratio from patient to patient. A meeting with Tatiana and Angelo from Ecole indicated that the 550 or 600 nm data should be used for our machine learning task.

The way Ecole Polytechnique has used this dataset for precancerous detection is through statistical methods. AUC is the standard metric used to quantify detection performance in medical applications where AUC = 1 represents perfect detection and AUC = 0.5 represents guessing. Using minimal post-processing, an AUC of 0.9 can be achieved, this was further improved by using the scalar retardance and 3 smallest eigenvalues of the matrix for each sample which has achieved an average AUC of 0.95. This was done by using scalar retardance to compute the non-depolarizing part of the Mueller matrix and the 3 smallest eigenvalues to calculate the depolarizing part of the Mueller matrix through the 24 samples[1]. The research and development towards this project aim to contribute to this solution by providing different insights and methods to compare future data to. In the event that a trained model obtains a higher AUC score of 0.95, which would be an indication of higher specificity and sensitivity ratio, the predictions from the model can be used to re-evaluate existing methods and future data.

2.2.1 Types of data

There are 3 different types of data I used that will be referred to frequently throughout this report:

- **Intensity Data:** 600x800x16 matrix which contains the data for plotting the image shape. The wavelength used throughout the project was 550 nm. This is the data used to plot the grayscale images in fig 2.1
- **Mueller Data:** 600x800x16 matrix which contains the data for classification. This has a different purpose to the intensity data as it contains the output of Mueller Polarimetry. The first channel of this data is an intensity image. The wavelength used throughout the project was 550 nm. This is the data shown in fig 2.4
- **Mask Data** 600x800 matrix which marks regions of either healthy pixels, CIN2-3 pixels, or both health and CIN2-3 pixels. The purpose of this data is that not all pixels of a sample is classified, therefore this mask data must be applied to the Mueller data to obtain the classified pixels. This is data represents the green and red regions shown in fig 2.1

2.2.2 Scalar Retardance

Scalar Retardance has been used in previous solutions to distinguish healthy and CIN2-3 zones in the Mueller data. This parameter is measured using the Lu-Chipman algorithm to perform polar decomposition of the Mueller Matrix[9]. Image segmentation is then performed using degree thresholding to produce sample images of red and green regions to represent cancerous and healthy tissue respectively.

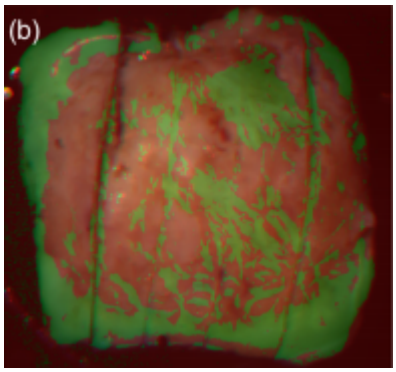


Figure 2.3: Image segmentation using a threshold of 8.9 deg for the scalar retardance R for measurements performed at 550 nm

The scalar retardance will be used in the evaluation section of this report to perform visual analysis. This will be done by comparing the cancerous and healthy regions predicted by our trained machine learning models to the scalar retardance images provided by Ecole.

2.3 Features and Classification

The existing solution for extracting features from the Mueller matrix uses statistical functions to transform the matrix represented as **m** to remove noise. A mathematical observer then computes a scalar-valued test statistic by logging the ratio of likelihoods of each feature using probability density functions. This can be written as:

$$\lambda(v) = ln[pr_1(v)] - ln[pr_2(v)]$$

While this has been proven efficient in providing a high AUC, it is a complex solution which requires a large amount of statistical knowledge to apply functions to high dimensional matrices to extract features. The need for post-processing of the Mueller data to achieve these features is also computationally expensive and the implementation of a machine learning model can hope to automate the feature selection process to improve the current complexity of the problem so that classification can be made sooner.

Looking at the structure of the data samples given, there are two sets of unhealthy and healthy masks, one prepared by Meredith, a postdoctoral fellow at Ecole Polytechnique, and one prepared by Prof. Angelo Pierangelo. Upon meeting with Ecole we have gathered that the masks shown in their most recent publication are the regular masks made by Angelo. Therefore the same mask should be used in our machine learning model so that accurate comparisons can be made between the CIN2-3 and healthy regions of Angelo's mask and the predicted regions.

2.3.1 Feature Extraction

Although previous solutions have used statistical post-processing to obtain additional functions of the Mueller matrix, the goal of this project is to explore the 16 matrix values to find out if these post-processing steps are necessary. Upon visualizing all 16 values of a Mueller sample, we can observe which combination of the 16 values could be considered in feature extraction.

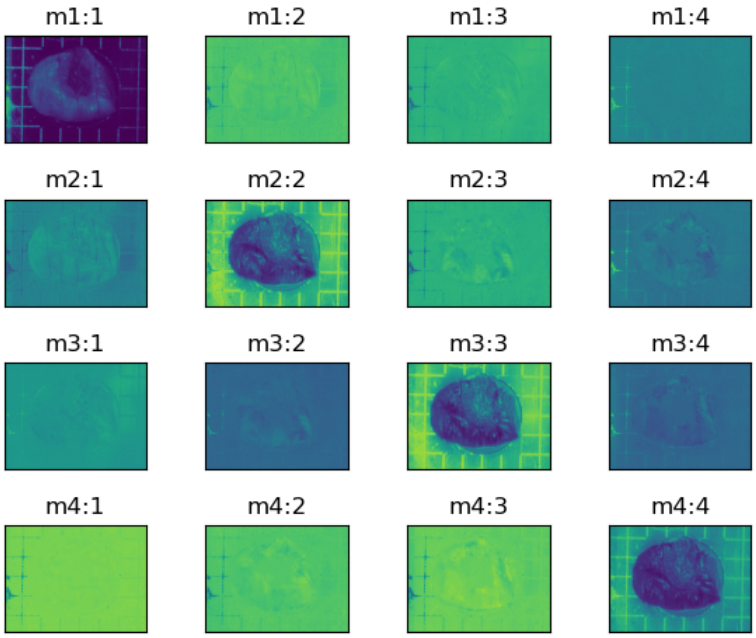


Figure 2.4: Mueller images of the 16 channels for sample 1

By observing the shape of the plotted samples using the visualization pipeline in figure 2.4 we can gather initial information on the data. Looking at these images we can make assumptions that the diagonal matrix of m1:1, m2:2, m3:3, and m4:4 may contain the most information based on how clear the specimen can be seen. The two combinations of features we decided to use from this was the diagonal matrix and all 16 values of the Mueller matrix. The reason for this was to see whether the results of our machine learning exploration relate to the observable information, or if the Mueller matrix contains information that the human eye cannot process. The resulting data sets will have 59,000 CIN2-3 and 135,000 instances of 4 input variables for the diagonal matrix and 16 input variables for the full matrix. The output variable is the class of the pixel, where 0 is healthy and 1 is CIN2-3

2.3.2 Dealing with Class Imbalance

The ratio of 59,000 CIN2-3 and 135,000 instances means that roughly 30 percent of the data we can fit to our models are of class 1 and 70 percent of the data we fit to our model are of class 0. This is known as class imbalance and can lead to biases and inaccurate results when training our models. For example, if our model were to classify every instance as 0, it would obtain an accuracy of 70 percent by default. To ensure our results are accurate regardless of the class imbalance we use the terms specificity and sensitivity, which are the amount of true positives and true negatives predicted by our model. The evaluation metric of the AUC score used by Ecole is achieved using these two values to ensure that the model performs well for both classes. The machine learning models explored in this project will also be

evaluated using the AUC score to ensure the results can be compared to past solutions.

2.3.3 Convolutional Neural Networks

One configuration of machine learning that we can use to predict these cancerous and healthy regions are CNNs (Convolutional Neural Networks), which is one of the most popular classification approaches for 2D data. A CNN contains a layer called the convolutional layer which takes an input of a matrix (h x w x d). Multiple kernels are then applied to the Convolutional layer as a filter by multiplying each position with the corresponding number on the input matrix (in this case a Mueller Matrix sample).

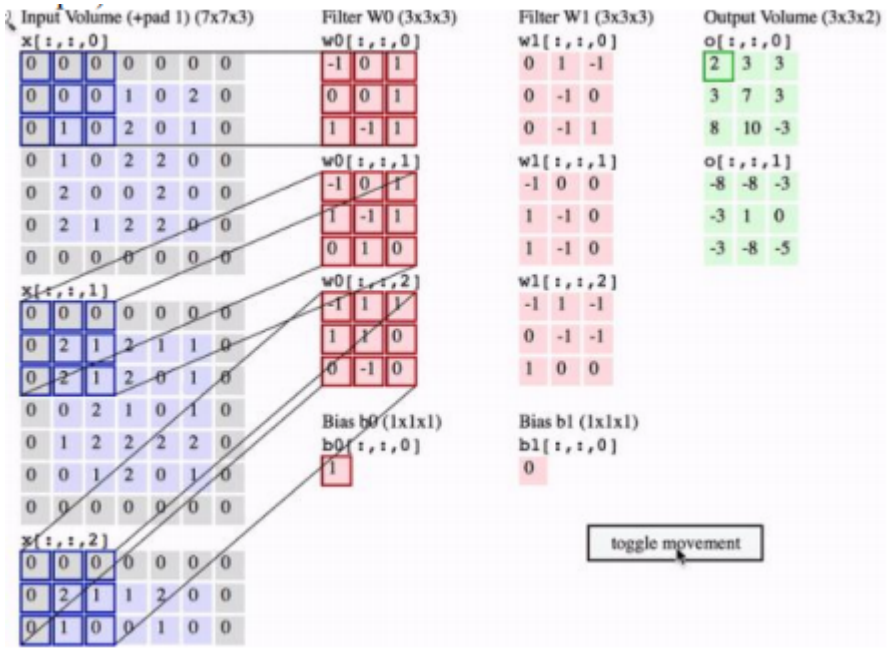


Figure 2.5: Example of a Convolutional Neural Network process

From the image above we can consider the input volume to be our (600x800x16) Mueller Matrix. CNN will then perform backpropagation to update the values of the required filters after each train. The sum of the values from applying the kernel to the convolutional layer is then mapped to the output layer where an activation function is applied to extract features. Finally, the CNN will apply pooling to further reduce the size of our extracted features to help reduce overfitting and computation[10].

Over the course of the project, the proposed solution changed from classification using pattern detection to pixel-wise classification. Due to realizing every sample was taken from a cancerous specimen, classifying entire samples images with 3D CNN as originally planned would become irrelevant as every sample would be labeled as cancer. looking at figure 2.1 we can see that only certain portions of each sample are masked, by approaching the problem using pixel-wise classification instead, we can extract an instance of the 16 channel values for every marked pixel. This allows us to extract significantly more data for training and testing so that our model can be applied to a specific sample. The output of each sample prediction will contain 600x800 = 48000 classified pixels for each sample which can be reshaped to plot an image of the newly marked regions and the unmarked regions for each sample.

Further research of CNN for lower dimensional data such as 1D and 2D have indicated CNN can still be used to perform pixel-wise classification. The newly proposed data format contains 16 channel values for each instance which can be passed into a 1D CNN or reshaped into a 4x4 array to be passed into a 2D CNN. Recent studies on 1D CNN have been made for signal recognition such as the Time Series Classification problem where each feature in an instance changes over time[11]. This approach can relate to our data because each 16 channel values for a pixel represents the same object but changes based on input and outputs of the depolarizing state rather than time.

A 1D CNN is likely to be the best solution for applying convolution to our data, due to the extracted 16 channel data format being 1D. However, 2D CNN has been known to learn more complex features and may also perform well on a 4x4 input shape. From this research, a 1D and 2D CNN will be both explored to compare the results.

Chapter 3

Design

This chapter discusses the selected design, along with the alternative designs which were implemented or considered to perform analysis using machine learning on the Mueller polarimetric data set. The design decisions made throughout this project are separated into the visualization pipeline design, data set design, machine learning model design, and how I present my results using evaluation metrics.

3.1 Visualization Pipeline Design

The visualization pipeline serves as the bridge between code and user which processes the Mueller data to plot information about the data at different steps throughout the machine learning task. Because this is an ongoing project, it is likely that a student in the following years will try to improve my findings or observe them as a starting point towards new solutions. One thing I personally found difficult and time-consuming while starting this project was understanding the data provided. Each sample folder in the data contains 8-9 different mask files that have different purposes. The only examples of the use of each mask were spread throughout over 100 mat-lab files created by Ecole, making it very difficult to identify how each mask was used. The purpose of the visualization pipeline was designed to make the initial understanding of pre-processing of this data easier by using python libraries so that anybody with similar experience as myself can pick up an understanding of the data faster. Because the goal of this project is to assist further research rather than implement a usable system or software, the majority of the design aspects are catered towards students at a similar level as myself to understand the findings of this project.

The visualization pipeline is split into two parts, the pre-processing pipeline, and the post-processing pipeline:

- The purpose of the **pre-processing pipeline** is to aid in understanding the matrix data and apply transformations such as reshaping, and matrix multiplications to obtain the selected regions of healthy and CIN2-3 tissue from the data. By plotting the data shape we can observe whether the correct data is being used by comparing it to the shape of Ecole Polytechniques data.
- The purpose of the **post-processing pipeline** is to take the 480000 classified pixels from the machine learning predictions and reshape the data into a 600x800 image. By applying thresholding to these pixels, class 0 becomes green and class 1 becomes red to indicate healthy and CIN2-3. The entire predicted sample can then be plotted for evaluation by comparisons with expected regions.

3.1.1 Examples

Pre-Processing Visualization Pipeline

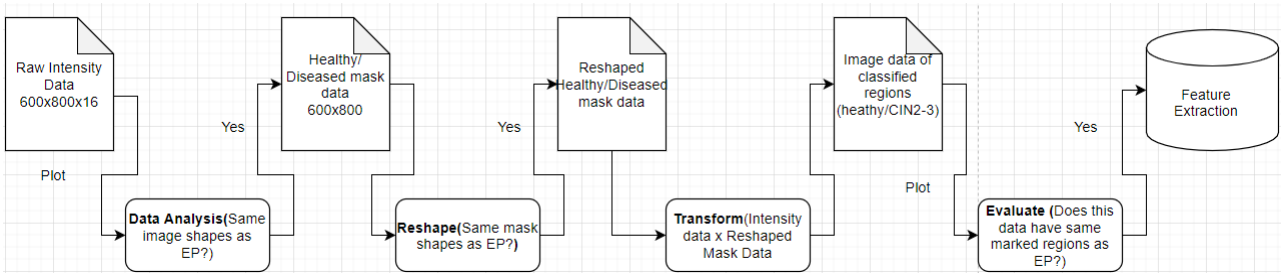
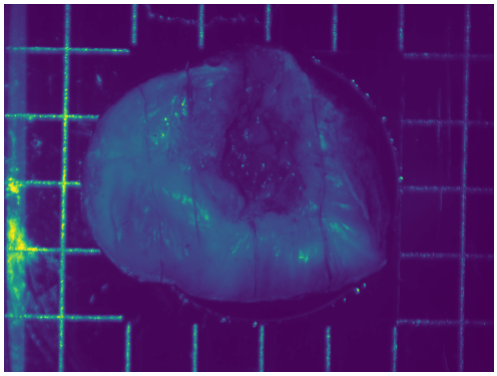


Figure 3.1: Design of the **pre-processing pipeline**

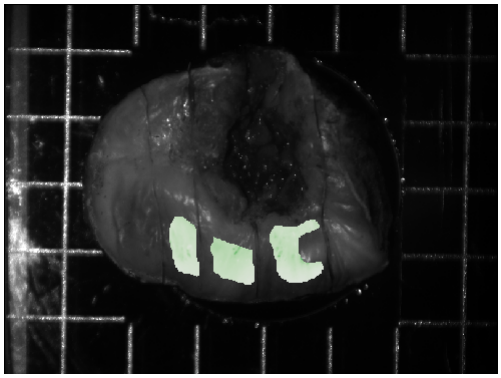
Figure 3.1 shows how data is processed and evaluated through the pipeline. Figure 3.2 shows an example of the images the pipeline plots to obtain a masked image of sample 1 which can be used to compare to sample 1 in Figure 2.1. Each transition shown in the design image is further explained in the following steps:

- Step 1: Input Intensity or Mueller data and plot image. Compare the visualized image with EP image without masks, If they match it means that the correct data is used.
- Step 2: Load data for healthy and diseased masks, check if the mask shape matches the correct sample mask seen in fig 2.1. I found that loading this data with the Python library NumPy causes a flipped array. Therefore reshaping is required to obtain the same regions as EP.
- Step 3: Multiply the raw input data with the reshaped mask data to obtain the marked healthy and diseased regions. Some samples contain both types of mask, healthy masked data are plotted as green pixels whereas CIN2-3 masked data are plotted as red pixels to differentiate the classes.
- Step 4: The image data should now be the regions of classified healthy or diseased tissue. Plot image and compare the shapes with EP images. If they match, Feature Extraction may be done with this sample using the Mueller data.

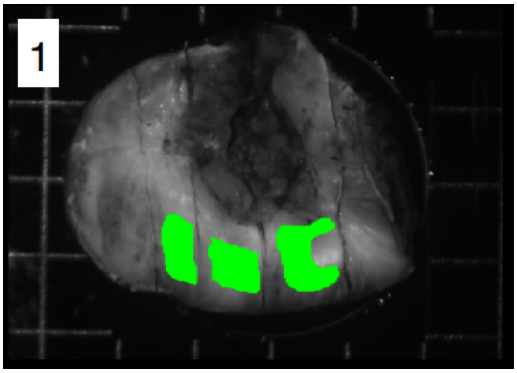


(a) Intensity image of Sample 1

(b) Sample 1 extracted Healthy pixels



(c) Grayscale intensity image with a green color map for the healthy mask.



(d) Sample 1 of fig 2.1 classified by EP

Figure 3.2: Example of Images observed in the pre-processing pipeline

The pre-processing visualization pipeline should be used on the **intensity data** because it is used solely for visualization of data shape to ensure the correct regions are marked, the reason why the intensity data is used rather than the Mueller data is that the samples shown in fig 2.1 are plotted using intensity data. The Mueller data may be used instead considering they have the same shape. However, as shown in figure 2.4 only m1:1 of the Mueller data is useful for comparison to the EP images.

Post-Processing Visualization Pipeline

The **post-processing visualization pipeline** was designed after evaluation on the data set achieved good results, but there wasn't a way to determine whether the predictions match the EP diagnostics. By extracting each sample into individual data sets with the same shape as our training and test data, I was able to classify every pixel as CIN2-3 or healthy using the trained model on unseen data so that comparisons can be made to the regions marked by histopathology.

Figure 3.3 shows how data is processed through the post-processing pipeline to transform predicted pixels into a threshold image which contains cancerous and healthy regions. Figure 3.4 shows an example of how the output of this pipeline looks. One design decision I made was to include the Pathologist's cuts in the threshold image, these cuts are made on the sample to diagnose the regions represented by the masks. The reason I included these cuts is that there is a 200 to 500 micrometers of positional uncertainty of the real cut[9]. Including these cuts may allow the biomedical professionals at Ecole to obtain additional information about the classification. Each process shown in the design image for the post-processing visualization pipeline is further explained in the following steps:

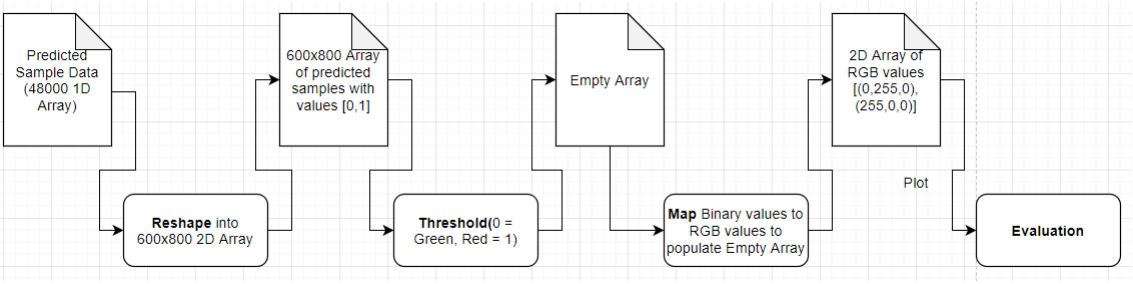


Figure 3.3: Design of the **post-processing pipeline**

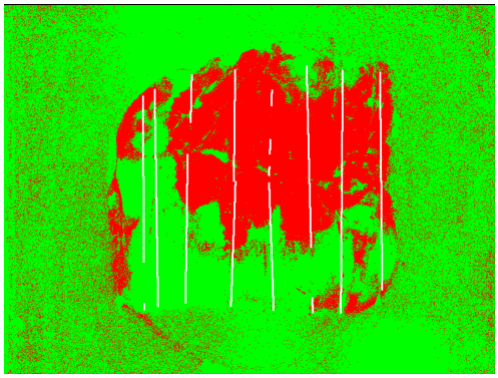


Figure 3.4: Predicted Image of sample 22 visualized by the **post-processing pipeline**

- Step 1: Input Intensity or Mueller data and plot image. Compare the visualized image with the EP image without masks, If they match it means that the correct data is used.
- Step 2: Load data for healthy and diseased masks, check if the mask shape matches the correct sample mask seen in fig 2.1. I found that loading this data with the Python library NumPy causes a flipped array. Therefore reshaping is required to obtain the same regions as EP.
- Step 3: Multiply the raw input data with the reshaped mask data to obtain the marked healthy and diseased regions. Some samples contain both types of mask, healthy masked data is plotted as green pixels whereas CIN2-3 masked data is plotted as red pixels to differentiate the classes.
- Step 4: The image data should now be the regions of classified healthy or diseased tissue. plot image and compare the shapes with EP images. If they match, Feature Extraction may be done with this sample using the Mueller data.

3.2 Data Set Design

In a machine learning problem, the data set is the most important factor in achieving good results. No matter how optimized a model is, if the features selected to train the model have no correlation, high accuracy cannot be obtained. The format of the Mueller data provided for this project could not be used right away to train machine learning models for binary pixel-wise classification. Instead, comprehensive feature selection had to be done to acquire a good data set to train our models.

My approach for selecting features from my data set was to use the Pre-Processing Visualization Pipeline to ensure the diseased and healthy masks were aligned correctly for each sample. I was able to extract 57925 cancerous pixels and 133,910 healthy pixels from all 24 samples of the 550nm masked Mueller data. The reason why I used 550nm is that not all samples had 600nm data, and having different wavelengths would affect the training of each model. Two data sets have been produced using this data, one using only 4 values from the diagonal Mueller Matrix, and another using all 16 values.

Dataset 1: the format of this data is 4 numeric input features (M1:1, M2:2, M3:3, M4:4) and 1 binary output feature (0,1). The reason for exploring the diagonal matrix is because when visualizing images of all 16 Mueller values seen in figure 2.3, the images of the specimen are most prominent within these channels. This was an **alternate** design that was not used due to poor performance.

Dataset 2: the format of this data is 16 numeric input features (M1:1 to M4:4) and 1 binary output feature. The goal of this project is to obtain the best AUC score using Mueller data, hence why all 16 features are used. This was the final design for the dataset because it achieved the best results after evaluation.

The file format I chose to store these two data sets was CSV (comma separated values). The reason why I chose this format is that csv is plaintext and widely used, therefore takes less storage space while being easier to import using python libraries such as NumPy.

3.2.1 Train Test Split

The training to test ratio I chose to use was 70 percent for training and 30 percent for testing. Due to having imbalanced class data, uniform stratified shuffling is required to obtain a good representation of the diseased to healthy data ratios. The random seed I used to shuffle and split the data for every model was 8. These configurations for the train test split have to be consistent throughout each model to ensure the same data is being evaluated.

3.3 Baseline Model - Decision Tree

When dealing with complex data, a baseline model can be used as a valuation tool by producing a very basic model/solution to obtain an initial accuracy. The design for my baseline model uses decision trees because it requires less data-processing and generates understandable rules to observe which features are being used to train. A decision tree also has a very low time complexity of O(mn) where m is the size of the training data and n is the number of features[12]. This allowed me to determine whether an extracted data set was likely to be useful in training more complex models by observing how well it initially performs on the baseline model.

In figure 3.5 we can see that the tree chose the root node to be 'X2' which corresponds to M1:3. Based on the value of this feature, the tree splits into two different paths to narrow down the data until a leaf node containing the classification is found. The 'value' indicates how many classes match the path to the node, for example when the traverses left, 34815 healthy and 34595 cancerous pixels contain an 'X6' or M2:3 value that is less or equal to 0.018. By observing this tree we can gather information such as 34595/40497 = 0.854 percent of cancerous pixels have an M2:3 value of less or equal 0.018.

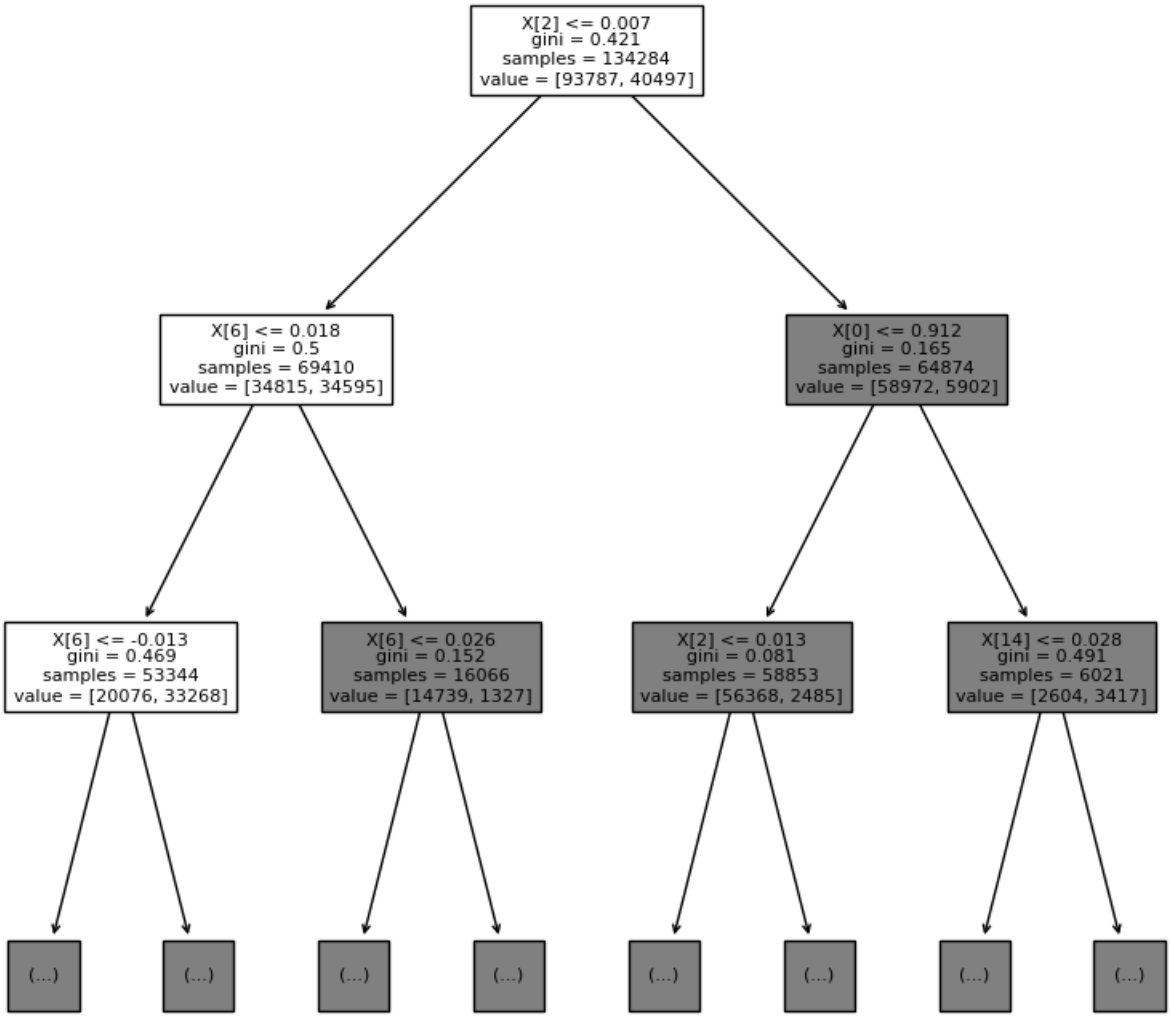


Figure 3.5: decision tree model with max depth = 3

3.4 Deep Learning Model Designs

The data sets acquired have a familiar data format that can be easily passed into different deep learning models. The goal of this project is to acquire the best possible results from each explored algorithm, therefore a good model design is indicated from the optimization of hyperparameters to create neural networks that best fit our data. By doing this, we look to improve the training efficiency and accuracy of our models.

3.4.1 Multi-Layer Perceptron

To obtain higher accuracy than the baseline model, more complex patterns can be trained using deep learning, which is a subfield of machine learning inspired by the function of the human brain[13]. One configuration of deep learning I chose to explore is a multi-layer perceptron, which feeds input data through a network of fully connected layers to approximate a function f() to map to an output[13]. After each feed forward, back-propagation is done by recalculating the weights in the network from the last hidden layer to the input layer to minimize error through multiple iterations.

The MLP I've trained is designed to acquire the best possible results based on the data set. While there is no sure way to configure an MLP, educated decisions can be made to optimize the hyperparameters more efficiently through trial and error. The final design of my multi-layer perceptron looks to achieve the highest AUC score by minimizing the loss function. Each decision made to tune these hyperparameters will be discussed.

Number of Hidden Layers and Nodes

The complexity of the MLP architecture is determined based on the number of hidden layers, and the number of nodes in each layer. Due to the complex nature of the Mueller data, I have chosen to use 3 hidden layers. The reason I chose this design was that the initial design of using 1 or 2 hidden layers achieved higher loss after 100 epochs(amount of times all training vectors are used to update weights.) compared to 3 hidden layers. By increasing the hidden layers, the model also becomes more complex as the function are summed together:

$$f(x) = f(3)(f(2)(f(1)x)) \tag{3.1}$$

Where x is in the input data, more complex features are learned as we increase hidden layers. The reason I did not choose to add more layers than 3 is because the number of attributes in our data is relatively small and can overfit if the neural network has more information processing capacity than required[14].

Each hidden layer contains a given amount of neurons that have a large influence on the final output. The number of neurons for each hidden layer should be carefully considered, as using too few neurons can result in underfitting and using too many neurons can result in overfitting[14]. The number of hidden neurons should be relative to the size of the input and output layers so that the model does not become overly complex. Data scientist Dr. J. Heaton has come up with 3 rules of thumb that can be used to find the right number of neurons required in a neural network[15]. Each of these 3 rules was applied to my model to find the configuration with the lowest loss.

- **Rule 1: The Number of neurons should be between the size of the input and output layer.** For this rule I configured my model to have 3 hidden layers with 8 neurons per layer. The loss function after 100 epochs was 0.113
- **Rule 2: The Number of neurons should be between the size of the input and output layer.** For this rule I used 3 hidden layers with 11 neurons per layer. the loss function after 100 epochs was 0.0856
- **Rule 3: The Number of neurons should be less than twice the size of the input layer** For this rule, I used 3 hidden layers with 32 neurons per layer. the loss function after 100 epochs was 0.0347

From this analysis we can see that the loss function decreases as the number of neurons increases. I also tried a configuration of 3 hidden layers with 64 neurons per layer which managed to achieve a slightly smaller loss. However, this model took significantly longer to train with only about 0.008 difference in loss compared to 32 neurons. This indicates that the configuration using rule 3 with 32 neurons per layer was the best for my model.

Activation Function

The activation function is used to determine the output of a neural network. This function is attached to each neuron in the network and determines whether it should be activated based on whether the input is relevant for the model's prediction. My final MLP design uses the **ReLU(rectified linear unit)** function. The reason why I chose this function is because it is linear for values greater than zero, but is non-linear with negative values. This allows the model to train faster as it has the properties of a linear function with positive values while still being non-linear so that it can back-propagate[16]. By not activating on all negative inputs, our model becomes sparse, which is faster and causes less overfitting/noise to the other activation functions considered such as sigmoid[16].

Adam

My final MLP model uses the Adam optimization algorithm instead of classical stochastic gradient descent(SGD) to upgrade network weights. The problem with classical SGD is that it's more computationally expensive when using large data sets and the learning rate and momentum must be optimized to obtain the best possible results. Adam solves this problem by adaptively changing the learning rate, allowing our MLP model to converge faster. Adam is also computationally efficient and has little memory requirements[17], which is optimal for our exploratory machine learning task.

3.4.2 Convolutional Neural Networks

Convolutional Neural Networks is another deep learning algorithm often used in image classification. The term convolution involves applying kernel filters to input data to extract feature maps that are used for classification. Similar to hidden layers in multi-layer perceptrons, adding more filters to a CNN model increases the complexity of features it can learn. My final design explores both **1D** and **2D** CNN's to identify whether pattern detection can be used in classifying CIN2-3 and Healthy pixels in our data set. The final design of these two models required hyperparameter optimization to determine the amount and types of filters needed to obtain the best results possible.

In order to explore the 16 Mueller values in both 1D and 2D space, the same network architecture had to be used so that they could be compared. The only difference between the two models is the input shape and the shape of the kernel filters.

Design Architecture for 1D and 2D CNN
For both 1D and 2D models, the input convolution has 16 filters. The Majority of 2D and 3D CNN use more filters generally ranging from 32 to 512 for image data, which usually has high dimensionality. The reason why my chosen CNN models only use 16 filters is due to our input data only contain 16 values, and having an excessive amount of filters will make the model overly complex. The padding for this layer is 'same' which means padding is applied so that the output feature map maintains the same shape. The activation function for this layer is ReLU. the reason for choosing this function is similar to its use in MLP, allowing the model to be linear for positive values and non-linear for negative values to improve training efficiency.

- For the **2D CNN**, the shape of the input layer is (4,4,1)
- for the **1D CNN**, the shape of the input layer is (16,1)

Another Convolution layer of 16 filters is then applied using the new feature map so that more complex features can be learned further. This layer also uses padding and the ReLU function to obtain our main feature map. The reason why only 1 Convolution layer is applied after the input layer is due to the need to continually add padding around the image, with such small input data, applying too much padding can lead to inaccurate features being learned for classification. This is because padding pixels add 0 pixels, which are then applied to the corners of the output feature map.

- For the **2D CNN**, A flatten layer is then applied to each CNN to convert our a 3D feature map of shape (4,4,16) into a vector of shape (256) that can be fed into a fully connected neural network classifier.
- for the **1D CNN**, this flatten layer is applied to a 2D feature map of shape (16,16) into a vector of shape (256) instead.

The new vector of shape (256) is the result of the convolution layers, which are then passed into a dense layer to generate predictions. This output layer uses the 'softmax' activation function instead of 'ReLU'. The reason for this is because softmax calculates a probability distribution using the flattened feature vector values to obtain a fully connected layer. The probability of the pixel being CIN2-3 or healthy is then summed to 1 and the probability function is used to make predictions.

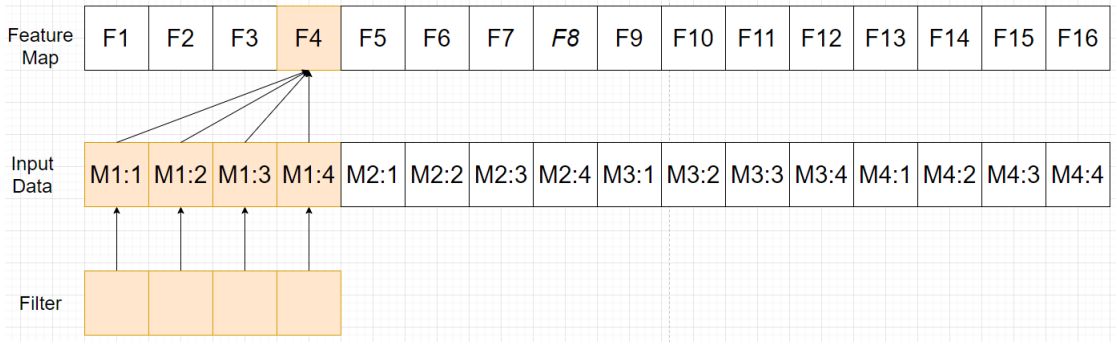


Figure 3.6: Convolution example on 1D space

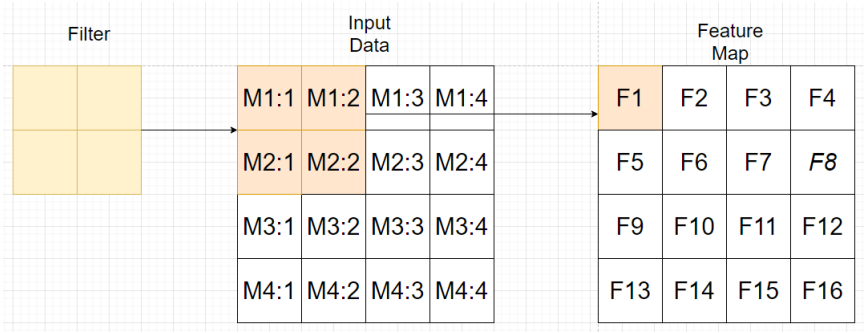


Figure 3.7: Convolution example on 2D space

Looking at the examples of 1D and 2D convolution in Figures 3.6 and 3.7 respectively, we can see that each model operates similarly but learns different patterns. A **1D CNN** may only move in one direction so that a linear signal of M1:1 to M4:4 is learned. A **2D CNN** may move in both the x and y directions, allowing the Mueller data to be convoluted in a 2D space to learn different features. The performance of these two models will be further explained in the evaluation section.

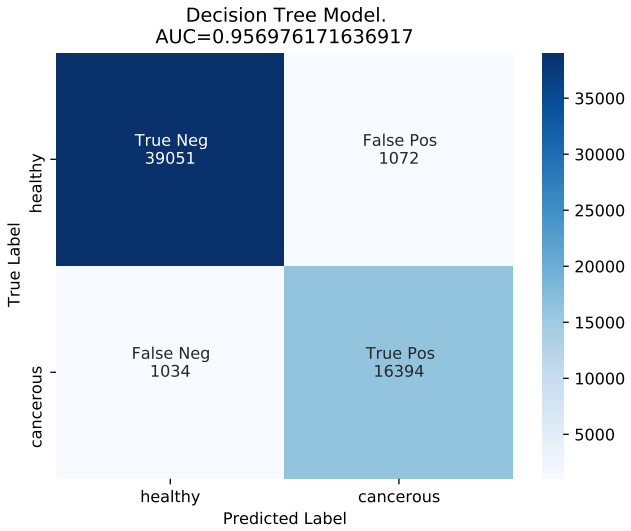
3.5 Confusion Matrix Design

A confusion matrix is a table that contains the performance of a classification model on a set of test data. It shows the number of true positives, true negatives, false positives, and false negatives predicted as an indication of how accurate the model predicts on each specific class rather than the overall accuracy. A research on reporting machine learning predictive models in biomedical research was conducted and found that due to the complexity of machine learning models, the results are often insufficiently reported in research articles[18]. One of the key points in this article is to report the results in a way that shows the limitation of the model with consideration to potential pitfalls[18]. In relation to classifying cancerous or non-cancerous regions of the Mueller data set, the evaluation metric used is the AUC score. The AUC is calculated using the number of true positives and true negatives to deal with the class imbalance in our problem. However, the score alone does not give an indication of the exact performance for each class.

The Python library sklearn allows easy calculation of the confusion matrix to be printed in a terminal or python console in text format. While this allows fast visualization to analyze the results when training models, the intended audience which are the professionals at Ecole Polytechnique are likely to be confused with a matrix without labels. My final design for the confusion matrix uses Python library SNS to plot the matrix as a heat map with appropriate labels so that the intended audience can process the information more easily.

```
[[39051  1072]
 [ 1034 16394]]
AUC: 0.956976171636917
```

(a) Initial Design: Text Matrix



(b) Final Design: Heatmap Matrix

Figure 3.8: Confusion Matrix Designs

By observing the images in Figure 3.8, we can see that the text representation of the confusion matrix is not informative. looking at subfigure a. we don't know what the numbers represent, the ratios of correct or incorrect predictions, and which model was used to obtain these results. Comparing this to figure b, we can see a heat map that scales on a blue gradient which gets darker depending on the distribution of the 4 results(TN, FN, FP, TP). The matrix is now clearly labeled with the classes and the predictions so that the audience does not require initial thinking of what value represents true positive or true negative. The name of the model used to calculate this result with the total AUC score is also shown above the matrix so that users know where the results came from. The intention of having this as my final design is that presenting an AUC score is not informative enough, and presenting the number of true positives and true negatives compared to false positives and false negatives allows the audience to see the specificity and sensitivity of the model using the values that the AUC score was calculated from.

Chapter 4

Implementation

The implementation of this project was split into three different steps, the visualization pipeline, feature extraction, and training the machine learning models. Each of these steps will be discussed in the following sections to explain why they were needed to reach the goal of this project.

4.1 Visualization Pipeline

The most important step before data pre-processing is ensuring that we have the right data, and can understand the data format. The data provided by Ecole are comprised of 2D matrices, so that mathematical operation can be used to explore and process each sample. The intensity data, Mueller data, and mask data provided have a .dat format which must be plotted before we can view the image of each sample. The shape of each .dat file is explained below.

- **Intensity data:** 600x800x16 intensity images with 550 wavelengths for each sample
- **Healthy mask:** 600x800 matrix which is >0 for each healthy pixel for each sample
- **Cancer mask:** 600x800 matrix which is >0 for each cancerous pixel for each sample

The initial step in beginning our data exploration is to plot these data files to ensure the correct images and masks align with the 24 masked images provided by Ecole Polytechnique. This allows us to obtain the correct regions to extract features from the labeled pixels in each sample by comparing the output of our visualization pipeline to Figure 2.1. Two main Python libraries that were used for this implementation are the MatPlotLib and NumPy libraries. The purpose of using the MatPlotLib library was due to the many tools it provides to assist in formatting and plotting matrices to obtain clear image visualizations. The purpose of using the NumPy library was to easily read data files as NumPy arrays, which allowed the use of multiple tools to reshape the masks and update the output data.

The pseudocode below shows how each sample was processed and visualized with the corresponding masks:

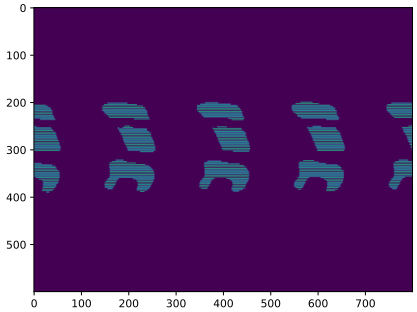
Algorithm 1: Visualization algorithm

```
Result: Intensity Images with cancerous and healthy regions.
Initialize figure and axes;
for Samples in intensity data: do
    load sample;
    load healthy mask;
    load cancer mask;
    reshape masks to 600x800;
    flip masks by axis 1;
    rotate mask 90 degrees;
    for row in range 600 do
        for col in range 800 do
            if healthy mask[row][col] greater than 0 then
                sample[row][col] = green
            end
            if cancer mask[row][col] greater than 0 then
                sample[row][col] = red
            end
        end
    end
    add Sample to output figure
end
plot figure
```

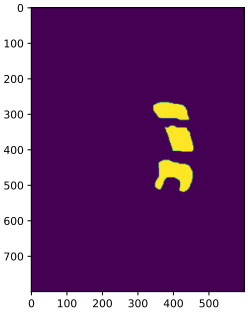
This algorithm iterates through the intensity data for all 24 samples, reshapes the respective masks, and then applies thresholding to change the respective array values to green or red to visualize the healthy or cancerous regions. The reshaping step is required as I found that the position and shape when loading the .dat masks were in the wrong position. This is due to the raw data being shaped as a 1D array of 480,000 instead of a 2D array of 600x800 when loaded as a NumPy array.

The following figure shows how a mask is shaped after the initial loading of the raw mask data, and how each step reshapes the image into a mask which matches the position and shape of the same mask shown in the Ecole Polytechnique images:

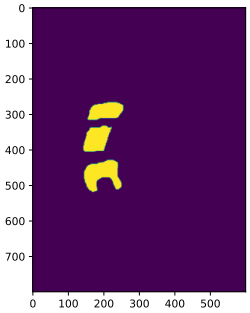
- **Sub figure a.** - The mask.dat was loaded in as a 480,000 value 1D array. I tried to reshape this data into 600x800 but noticed a strange repeat pattern of the mask.
- **Sub figure b.** - The mask.dat image was reshaped into 800x600 instead, from this I could observe that the mask had the correct shape but wrong orientation along the y-axis.
- **Sub figure c.** - The mask is flipped along the y-axis so that we can obtain the right shape of the mask by looking at it along the y-axis.
- **Sub figure d.** - The mask is rotated by 90 degrees so that it has the correct orientation along the x-axis, and is reshaped to 600x800 so that it can overlay our intensity data.
- **Sub figure e.** - The mask is applied to the intensity data using thresholding to obtain an image that can be compared to the Ecole images.



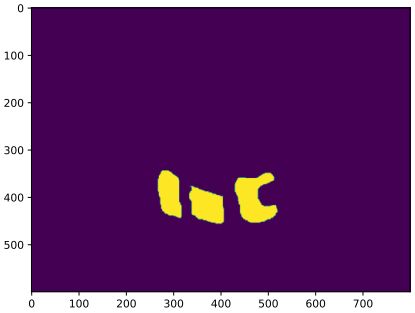
(a) Sample 1 healthy mask data reshaped into 600x800



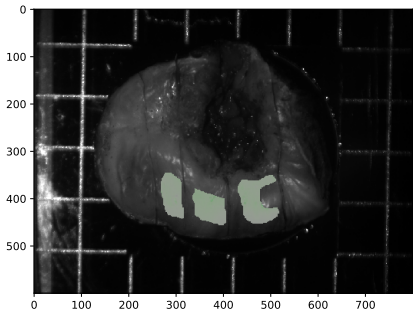
(b) Sample 1 healthy mask data reshaped into 800x600



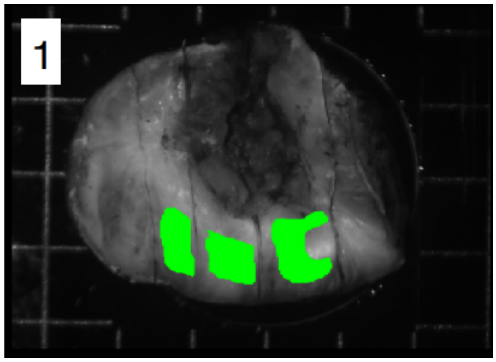
(c) flipped mask image along the y axis



(d) mask image rotated by 90 degrees



(e) reshaped mask applied to intensity data



(f) Ecole Polytechniques image of sample 1

Figure 4.1: Mask reshaping steps

By comparing Sub figure e to Sub figure f, we can now see that the mask is reshaped into the correct position. This reshaping algorithm was applied to all 24 samples to ensure the correct regions are masked before I began feature extraction.

4.2 Feature Extraction

Once the visualization pipeline was implemented, and I had a good understanding of the data, I could begin extracting features using the Mueller data. Fortunately, the Mueller data had the same shape as the Intensity data used in the visualization pipeline, which meant the Mueller data itself did not require any further pre-processing as the same reshaped masks could be used. An algorithm was written to iterate through each Sample's Mueller data to append labeled cancerous and healthy instances to a CSV file.

Algorithm 2: Feature extraction algorithm

```
Result: CSV file with mueller data formatted for training
Initialize array containing sample numbers of healthy samples;
Initialize array containing sample numbers of cancerous;
Create csv file in write mode;
for Samples in healthy data: do
    load sample Mueller data;
    load reshaped healthy mask;
    for row in range 600 do
        for col in range 800 do
            if health mask[row][col] greater than 0 then
                create new instance array;
                for dim in range 16 do
                    append Mueller[row][col][dim] to instance;
                end
                append class=0 to instance;
                Convert instance array into String seperated by ,;
                write instance to text file;
            end
        end
    end
end
for Samples in Cancerous data: do
    load sample Mueller data;
    load reshaped cancer mask;
    for row in range 600 do
        for col in range 800 do
            if cancer mask[row][col] greater than 0 then
                create new instance array;
                for dim in range 16 do
                    append Mueller[row][col][dim] to instance
                end
                append class=1 to instance;
                Convert instance array into String seperated by ,;
                write instance to text file;
            end
        end
    end
end
```

This algorithm works by first creating two arrays which separates sample numbers based on whether they contain healthy or cancerous pixels. The reason for doing this is because samples with only one type of labeled class will still have the corresponding mask file of the other class. For example, Sample 1 only has healthy masked data but also contains a cancer mask file in which all values are 0. By separating the two classes, the time complexity of running this algorithm is decreased as it prevents unnecessary iterations through non-useful files.

The algorithm then iterates through each set of healthy and cancerous samples and checks whether an individual pixel has a mask value (greater than 0). When a mask value is found, all 16 values of the Mueller data for that pixel is appended to an instance, and the corresponding class of 0 = healthy or 1 = cancerous. This was applied to the full 16 value Mueller matrix and 4 value diagonal matrix to create the data set format that we used for training and testing our machine learning models.

1	m1:1	m2:2	m3:3	m4:4	CLASS
2	1.027199	0.336008	0.325248	0.218945	1
3	0.991956	0.294502	0.287581	0.222682	1
4	1.027607	0.331759	0.325613	0.218199	1
5	1.053237	0.325147	0.320402	0.251420	1

Figure 4.2: Example data format from the diagonal matrix data set

By observing Figure 4.2 we can see that the data is now in a format that is very easy to input into our machine learning models.

4.3 Training and Testing Machine Learning Models

Once the data was formatted into a CSV file, we could begin training different machine learning models to classify cancerous and healthy regions. The four models I trained were a Decision Tree, Multi-layer Perceptron, 1D Convolutional Neural Network, and 2D Convolutional Neural Network. Each of the models was trained using CPU, which resulted in longer training times compared to GPU. The libraries and frameworks I used to implement these models, and the resulting architecture of each model will be discussed in the following subsections.

4.3.1 Decision Tree

To implement the decision tree model, I used the python sklearn.tree.DecisionTreeClassifier library. This allows us to easily build a decision tree that takes in our training data and make splits based on evaluation criteria. My implementation used a stratified split of 70 percent train and 30 percent test using the python `train_test_split` library. The use of a stratified split was required so that the training and test sets maintain the same ratio due to the data having a class imbalance. The evaluation criteria I used to measure the splits for my decision tree was the Gini impurity score. The Gini impurity picks a random datapoint and classifies it using the class distribution in the dataset, the chance of getting this classification wrong is the resulting Gini impurity score, which means a lower gini impurity score is better, and a score of 0 is perfect classification.

If we revisit Figure 3.5 we can see that the root nodes gini impurity score was 0.421, which shows that feature 2 has a 0.421 chance of incorrectly classifying. As more splits are made

from the root node we can observe that the gini impurity score decreases for splits that go inward. For example, after depth 2 in our Tree, we can see the inwards splits of X6 <= 0.026 and X2<=0.013 have the gini impurity scores of 0.152 for 16066 instances and 0.081 for 58853 instances respectively. This means after 3 splits our decision tree model has already selected 3 features in which there is a high chance of correct classification for 16066 + 58853 = 74919 train instances.

After this model is trained, unseen test instances can be predicted using the sklearn model.predict function and an AUC score can be calculated by comparing these predictions to their true labels using sklearn function `metrics.roc_auc_score` which calculates the area under a roc curve based on the number of true positives and false positives the model predicts.

4.3.2 Multi-Layer Perceptron

For my multi-layer perceptron model, I used the python sklearn.MLPClassifier library which takes in training data to build a neural network. This model was configured using the optimized hyperparameters which were explained in the design section to best fit our data. The same train test splits that were made for training the decision tree and CNN models were also made with the MLP model. This is so the results are consistent with the number of instances used to train, so that each model can be evaluated and compared to other models fairly.

How I used the MLP model

Algorithm 3: Using the MLP library

```
hidden layers = 3;
hidden layer size = 32;
activation = 'relu;
max iterations = 100;
solver = 'adam';
tolerance = 0.001;
verbose = True;
build MLP Model(hidden layers, hidden layer size, activation, max iterations,
  solver, tolerance, verbose);
fit training data to model;
predict test data;
```

looking at the above pseudocode for using the MLP library, we can see that the implementation is very simple due to the building of the MLP being automated. This allowed me to focus more on optimizing hyperparameters to achieve the best results possible from the data set. Two of the parameters used were not explained in the design section as they are more useful for implementation rather than the design.

- The parameter **verbose** is set to True so that progress messages are printed in the python console so we can gather information about each epoch. This aids in implementation so that we can stop training if there is an indication of the model getting stuck or overfitting.
- The parameter **tol** is set to 0.001 as a stopping criterion where the model will stop training if loss hasn't improved by more than the tolerance amount in over 10 epochs.

Example of a MLP model

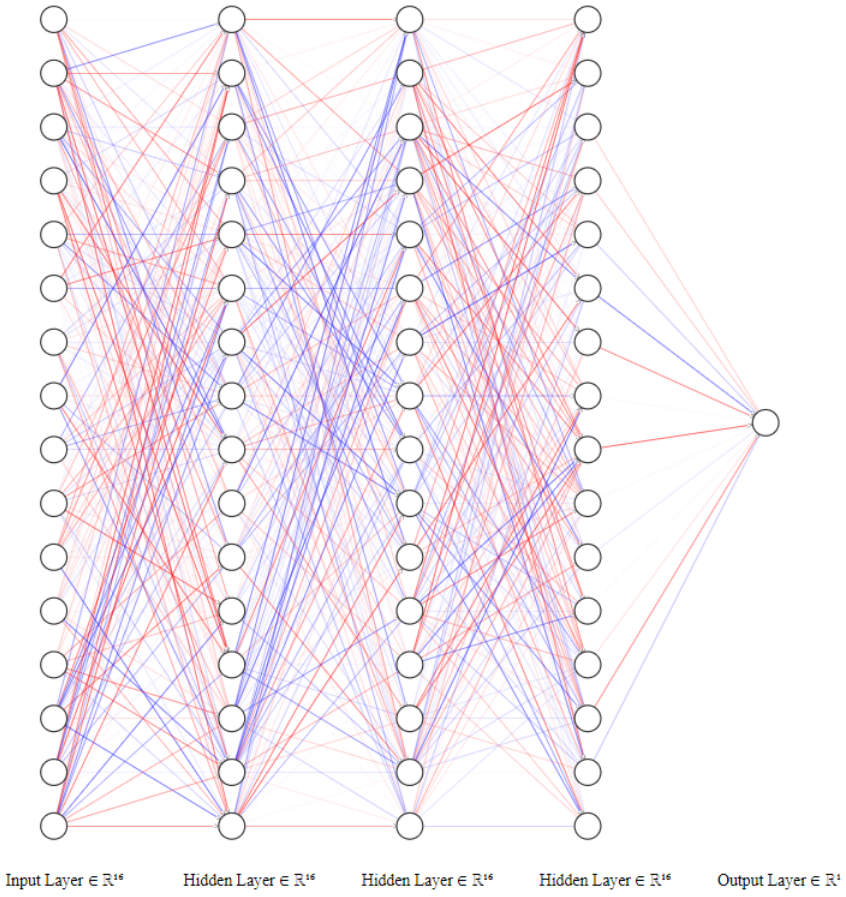


Figure 4.3: Example MLP visualized using the NN-SVG tool

Figure 4.3 is an example of an MLP with 3 hidden layers of 16 neurons instead of 32 so that we can clearly see the connections between each layer. This visualization was done using a neural network visualization tool called NN SVG[19]. The important part of this figure is the blue and red edges where blue is negative and red is positive. Because we're using the ReLU function, a neuron is only activated if the input value is positive, which allows the activation function to act linear, but also be nonlinear so back-propagation of errors can be calculated to update weights for training the neural network. Each epoch represents a forward propagation and backpropagation cycle where the 16 Mueller values for each instance are passed through each hidden layer to produce an output value for prediction. The neural network then calculates the error between the expected outputs, and outputs forward propagated. This error is used to go back through the hidden layers to update the weights of the neural network[20].

My final MLP model had a set max iterations of 100 but met the stopping criteria set by the tolerance parameter at approximately 70 iterations. This is an indication that the model has finished training as any improvement in the loss is minimal at this point.

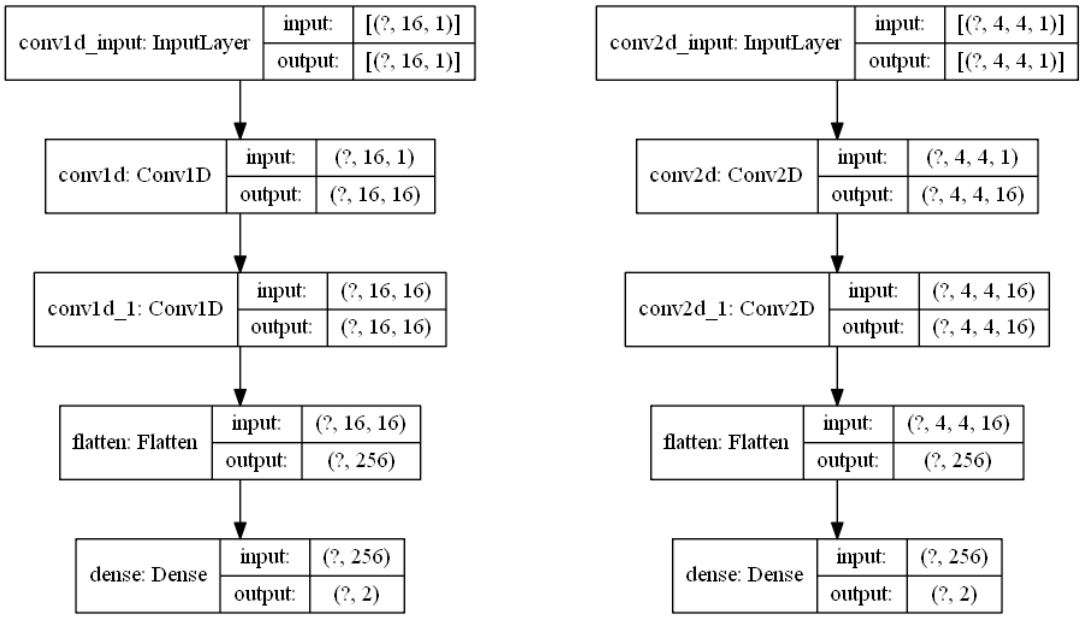
4.3.3 Convolutional Neural Networks

For the implementation of the 1D and 2D CNN models, I used the Python Keras library which provides tools that aid in building a neural network by adding different layers. The Keras.Sequential model allowed me to explore a CNN model with different convolution, dropout, pooling, and connected layers to improve the model's overall performance. The optimization of the hyperparameters and why I used certain layers for my final model were discussed in the design section. The same train test split was used for both 1D and 2D CNN so that every trained model can be compared fairly.

How I used the Keras libraries to build the 1D and 2D CNN Output models

Algorithm 4: Using the Keras Sequential library

```
reshape input data to (4,4,1) for 2D CNN or (16,1) for 1D CNN;
initialize sequential model;
add Conv1D/Conv2D layer as input layer;
add Conv2D layer;
add Flatten layer;
add Dense layer to map outputs;
compile model using categorical cross-entropy;
fit training data to model;
predict test data;
```



(a) 1D CNN Model (b) 2D CNN Model

Figure 4.4: Output CNN Models

Figure a. and Figure b. show the resulting model architectures created from the above pseudocode. From these visualizations, we can see that the only key difference in each model is the data shapes passed into the input layer, which changes the input and output shapes of each following hidden layer. This allows the CNN to learn models in the 1D and 2D space of the Mueller data so that the 256 flattened output values used for classification

are comprised of different patterns. Neural Networks are known as black-box algorithms which means that although we can observe the architecture of a model, there is no easy way of looking inside to identify the complex features that are learned[21]. Because of this, we rely on evaluation metrics such as AUC and loss to determine whether the model has learned features that contribute to accurate classifications.

Chapter 5

Evaluation

The goal of this project was to explore a variety of machine learning algorithms to determine whether the cervical cancer data from Mueller polarimetry can be classified correctly using artificial intelligent methods. This chapter will discuss whether the final models met this goal by comparing the AUC scores to existing solutions and evaluating the shapes of the predicted samples with existing threshold images.

5.1 Model Evaluation

Previous solutions required post-processing to compute parameters from the non-depolarizing and polarizing part of the Mueller matrix. The parameters calculated were scalar retardance, and the 3 smallest eigenvalues of the matrix were used to perform statistical methods to obtain an AUC of 0.95[1]. A comparison has been made between Ecole Polytechnique's AUC score and the AUC score of the four machine learning models I have trained using only the 16 values of the Mueller Matrix. These AUC scores were calculated using the output confusion matrix and the sklearn.metrics.RocAucScore function.

Model Results			
Machine learning algorithm	AUC Score	Loss	Training time
Decision Tree	0.956	-	2.9s
MLP	0.987	0.0333	171s
1D CNN	0.984	0.0426	424s
2D CNN	0.985	0.0368	433s

Table 5.1: Trained Machine Learning Model Comparison

by observing the AUC scores of each of the trained models, our baseline Gini decision tree model already acquires a very high AUC score similar to the result in the publication[1]. This is an initial indication that the Mueller data fits very well to machine learning algorithms and that the post-processing step of calculating scalar retardance and eigenvalues may not be required to obtain accurate classifications. Further analysis using the deep learning models MLP, 1D CNN, and 2D CNN resulted in very similar AUC scores of approximately 0.98 for each model through 100 iterations of training. This is an indication that the 16 features of the Mueller matrix have a strong correlation which allows each deep learning model to learn consistent patterns throughout the data.
So how do we determine which model is best for training and classifying the Mueller

polarimetry data?

if we observe the loss and time that each deep learning model took to train, we can clearly see that the MLP model takes almost half the amount of time compared to the CNN models, while obtaining the highest AUC score with the minimum loss. This is due to the time each convolution layer takes to perform convolution and map features into a feature map. By removing the 2nd convolution layer from each CNN, the total training time was reduced to approximately 250 seconds but resulted in an approximately 0.02 decrease in accuracy and 0.05 increase in loss. Even when this layer is removed, the MLP performs the best in terms of AUC, loss, and training time. From this analysis, we can conclude that the best model we have trained is the MLP model, which should be used for training new Mueller polarimetry data in the future to make the most accurate classifications.
The purpose of using AUC instead of classification accuracy for this problem is due to class imbalance, as approximately 70% of the data are healthy instances, and 30% are cancerous instances. The AUC score is calculated using the confusion matrix which represents the true positives, true negatives, false positives and false negatives from comparing the model predictions with the true test labels.

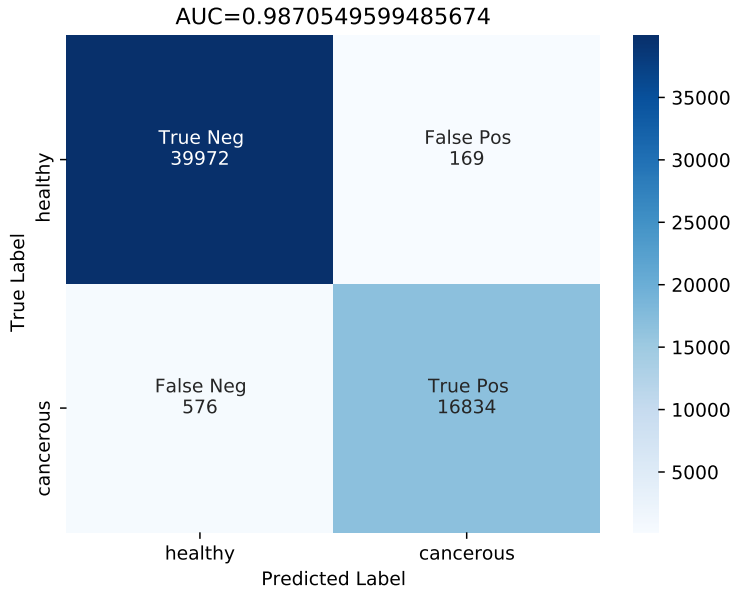


Figure 5.1: Visualized confusion matrix from applying the MLP model to test instances

This figure presents the confusion matrix as a more informative visualization with the intention of giving users an overview of the model's performance on each individual class so that the specificity and sensitivity of the model can be evaluated. A diagnostic performance test can be done by dividing the proportions of instances that are classified as healthy or cancerous by the overall amount of instances for that label[22].

$$Sensitivity = TP/(TP + FP) \tag{5.1}$$

$$Specificity = TN/(TN + FN) \tag{5.2}$$

Using equation 5.1 and 5.2 respectively, the sensitivity and specificity for the MLP confusion matrix can be calculated.
Sensitivity = 16834/(16834 + 169) = 0.99006

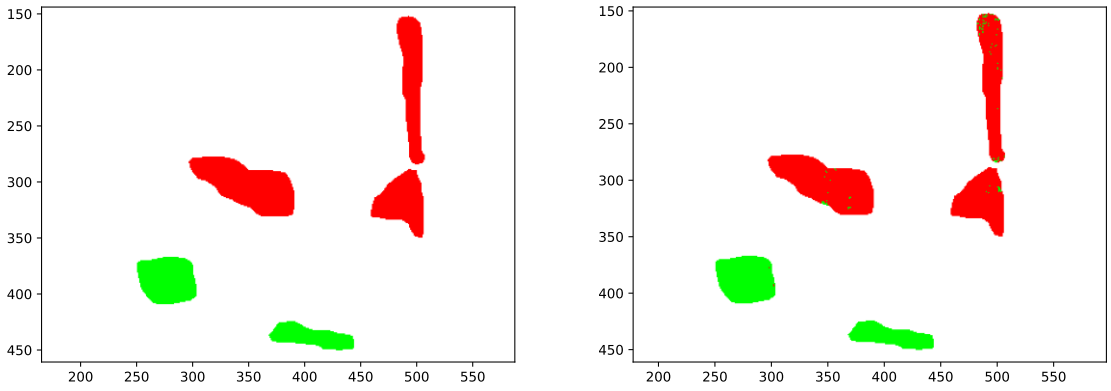
Specificity = 39972/(39972+576) = 0.98580
From these calculations, we are able to conclude that the model classifying a pixel as cancerous has a 99% chance of being correct, and the model classifying a pixel as healthy has a 98.5% chance of being correct.

5.2 Visual Analysis

The masked images in Figure 2.1 are the gold standard that we can compare the predictions of the final MLP model to. The masks used for these images were marked using histology, which requires cutting into the tissue of each sample to perform diagnoses on different regions. If our model can classify the exact same masked regions as histology using the Mueller polarimetry data, it will contribute towards the overall goal of this project and Ecole Polytechniques research to find a less invasive and time-consuming method to diagnose CIN2-3 tissue through Mueller polarimetry. Evaluation using visual analysis will be done in two parts. The first part is comparing predicted mask regions to the expected mask regions to try to achieve the same results as histology. The second part is predicting entire samples to make comparisons with the scalar retardance images.

5.2.1 Mask Predictions

For each of the 24 samples, an unlabelled CSV file has been extracted so that the entire sample can be predicted. Image thresholding was performed on the predicted labels of all 600x800 = 480,000 instances for each sample. Pixels are set to green if the predicted label is 0, or red if the predicted label is 1. Using the visualization pipeline, a full image that shows predicted cancerous and healthy regions has been constructed, and thresholding using the mask values is used to visualize only the masked regions to make comparisons with Ecole Polytechnique's masks.



(a) expected values from EP mask

(b) predicted values from MLP

Figure 5.2: Side by side comparison of EP mask and MLP predictions on sample 22

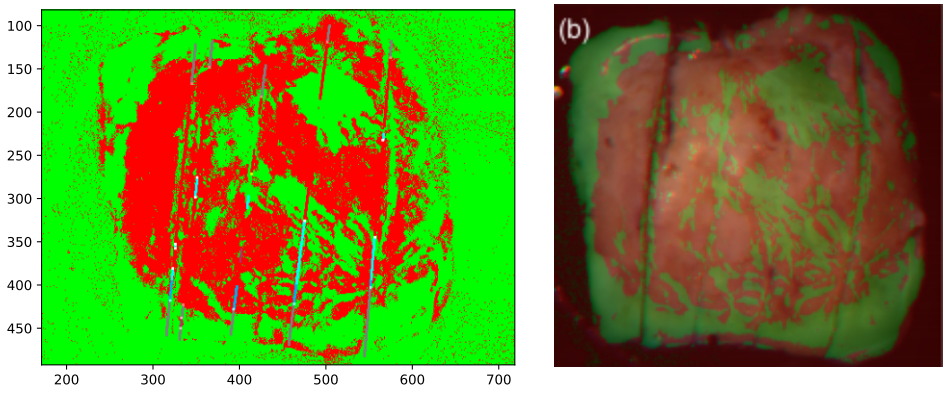
Looking at the predictions of the masked areas for sample 22, we can see that our MLP obtains the same cancerous and healthy regions as the masks from histology. If we look very closely at the predicted image, there are a few green pixels within the expected cancerous regions. The cause of this can be due to a few cancerous instances having very similar attribute values as healthy pixels resulting in being slightly on the wrong side of the model's

decision boundary. Another cause of this could be if there is any noise in the data, which in this case would be minimal. The important part of this evaluation is that the predicted masks and almost identical to the EP mask. This achieves the goal of the project by providing strong evidence that applying the MLP model on Mueller polarimetry data can obtain the same classifications as diagnoses by histology.

5.2.2 Scalar Retardance Comparison

In the previous section, predictions were made on only the labeled data which was randomly split before training our model. This means that there may be some bias when predicting the entire mask as some of the pixels could have been used in the training. This section will evaluate predictions on a full sample so that completely unseen data can be predicted and compared to past research images.

In previous studies performed on fresh conization specimens, where conization is the process of surgical removal of the diseased part of the cervix[23], the scalar retardance was found to be the most relevant parameter for healthy and CIN2-3 classification[9]. In a previous paper published by Ecole Polytechnique, image segmentation was performed on the scalar retardance measurements to obtain an image of only CIN2-3 or healthy tissue[9].
This is relevant to our evaluation because our MLP model is able to predict all 480,000 pixels of each sample and perform thresholding to visualize the CIN2-3 or healthy tissue regions. This creates a predicted image that represents the same information that the scalar retardance images show in the paper[9] so that comparisons can be made.



(a) Visualized image of sample 10

(b) Scalar Retardance image processed by EP

Figure 5.3: Side by side comparison of predicted image and scalar retardance image

Figure b was achieved by image segmentation using a threshold of 8.9 degrees for the scalar retardance measurements performed using the 550nm Mueller data. To ensure that our evaluation is valid, the data used for training the MLP model, and predicting figure a was also done using the 550nm Mueller data. By observing the patterns across figure a and figure b, we can see that the cancerous and healthy regions predicted are very similar. **This gives a strong indication that the MLP model can achieve similar prediction results without the need to calculate the scalar retardance from the non-depolarizing part of the matrix**. By removing this post-processing step, predictions can be made sooner for future diagnostics. In figure a, if we look very closely, we can observe vertical lines across the image with different colors. This is a separate mask used where each color represents a different

result from histology, this will be further explained in the future work section.

Final meeting with Ecole Polytechnique
A final meeting with Ecole Polytechnique took place on Friday the 9th of October 2020 to share the results of applying different machine learning models to the Mueller polarimetry data with Angelo and Tatiana, who are the domain experts of the research at Ecole. During this meeting, figure a and figure b were compared, and a conclusion was drawn that the MLP model used to classify figure a could be used for further analysis by Ecole. Angelo and Tatiana also requested predicted images for all 24 samples to be sent to them so they could compare them to existing and future results to determine how accurate the model is in real-world application. An ideal result from this is that the model is used to predict a completely new unseen sample at 550nm, and the histology results return the same cancerous or healthy regions. However, due to the time it takes to acquire new sample data and the Covid-19 situation in France, this was not possible before the project deadline.
Overall, the results of visualization, feature extraction, and training machine learning models have satisfied the project goals of using machine learning to perform binary classification of CIN2-3 or Healthy tissue using the Mueller polarimetry data. This is evident in obtaining a high AUC score, and proven by predicting very similar regions as existing solutions through the visual evaluation of images. The use of our MLP model allows Mueller polarimetry data to predict the same regions as results from histology, which satisfies a project goal of contributing towards less invasive and time-consuming methods. The use of the visualization pipeline allows predicted images to be compared with scalar retardance images which were found to have very similar patterns, this further satisfies the project goal of finding a less time-consuming method by obtaining the same results as post-processed parameters of the Mueller data.

Chapter 6

Conclusions and Future Work

This project is part of a larger French-New Zealand partnership program between The Victoria University of Wellington and Ecole Polytechnique with the purpose of bridging the gap between computer science and biomedical imaging. My individual contributions have been towards the first year of this program, which is likely to continue in the following years where a new student may continue this research. This chapter will discuss potential future work that could be carried out on this program, as well as summarize the report and findings.

6.1 Future Work

The machine learning models explored in this project had an emphasis on deep learning approaches such as MLP and CNN models. From this, there is strong evidence that different artificial intelligent methods could be explored such as evolutionary computation to try decrease the time complexity or increase the accuracy of the existing model. However, the Mueller polarimetry data has clearly shown a strong correlation between each variable as every machine algorithm I trained obtained very high specificity and sensitivity. To prove if these models are sufficient, Ecole Polytechnique must first perform an evaluation on their end to observe the performance compared to existing and future solutions.

The classification task for this project was binary, where a sample cervix tissue is labeled as either CIN2-3 or Healthy. CIN2-3 was used for pre-cervical cancer diagnosis as it is a type of tissue that slowly evolves into cancer over 5 to 10 years, which meets the criteria set by the World Health Organization as a disease that can be effectively managed by screening[24]. In the previous solutions explored by Ecole Polytechnique, there are more histological types than just CIN2-3[9].

Legend	Histological type
EMS	Healthy squamos epithelium
MEM	Squamos metasplasia
EGL	Glandular epithelium
HPV	Infection(Papilloma virus)
CIN 1	Cervical Intraepithelial Neoplasia - Grade 1
CIN 2	Cervical Intraepithelial Neoplasia - Grade 2
CIN 3	Cervical Intraepithelial Neoplasia - Grade 3
OE	Endocervical canal
NOI	Non identified

Table 6.1: **Histological types of epithelium**

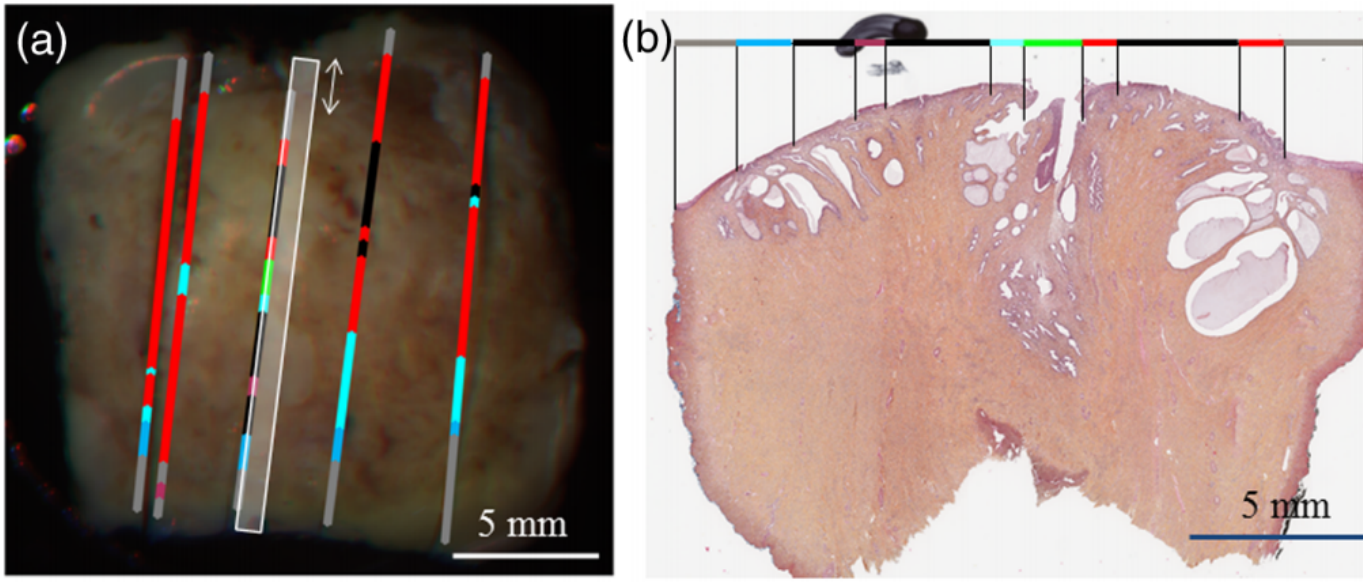


Figure 6.1: marked cuts with the diagnosed types from Histopathology

If we revisit Figure 5.3.a which is a visualized image of sample 10, if we look very closely, vertical lines can be observed across the sample. This is known as the 'colored mask' which represents where the cuts were made during Histopathology. From the table and Figure above, we can see that CIN2-3 is only a portion of the Histological types that can be masked. While there is no current emphasis on these other types, this is an indication that in the future there is a possibility that the research focuses on other diseases and may end up being a multi-class problem.

6.2 Conclusions

For this project, a visualization pipeline, feature extraction pipeline, and trained machine learning models have been designed, implemented, and evaluated. The main goal of this project was to explore different machine learning algorithms to analyze whether they are

suitable for classifying the Mueller polarimetry data. This is required due to the fact that artificial intelligence has yet to be explored using the data, and current methods of pre-cervical cancer diagnosis are invasive and time-consuming. If a specific machine learning model is able to predict the same regions as histological diagnosis, the model can be used to contribute towards less invasive and time-consuming methods in the future.

The high AUC performance of each model has proven that there is a strong correlation between the 16 values of the data and the label CIN2-3 or Healthy. The best model, a feed-forward multi layer perceptron, achieved an AUC of 0.98 which is calculated from the specificity and sensitivity(true positive and true negative rate) of the predictions. This is an indication that the model is not biased towards the class imbalance as it obtains accuracy in classifying both CIN2-3 and healthy pixels.

By visualizing our predictions for each sample and comparing them to the mask data, it is evident that the MLP model's predictions obtain the same regions as those marked by histology. The current method for histopathologic evaluation of cervical tissue is excisional tissue biopsy, which is defined by the process of complete removal of infected tissue, and some normal tissue around it[25]. Further comparisons between a predicted image of sample 10, and the scalar retardance image from previous solutions, have indicated that the model predicts similar patterns to scalar retardance which is the most important parameter for the current solution. This is an indication that the post-processing step of calculating the retardance from the non-polarizing part of the Mueller matrix may not be required in the future if artificial intelligence is able to achieve better results. The purpose of Mueller polarimetry is to find a new solution that does not require this extensive sample preparation. The trained machine learning model can be used by Ecole Polytechnique on existing and future samples to obtain faster diagnostics of the cancerous regions of a specimen, where comparisons with the existing solutions will be made to determine whether the model is fit for real-world application.

To summarize this report, the contributes I have made through the duration of this project are:

- The design and implementation of a visualization pipeline, which was used for visualizing Mueller polarimetry samples to obtain an initial understanding of the data, and for visualizing the cancerous and healthy regions predicted by the trained models.
- The design and implementation of a data processing pipeline that reshapes the cancerous and healthy masks and iterates through each Mueller sample to extract instances of labeled pixels.
- The design, implementation, and evaluation of a trained decision tree, MLP, 1D CNN, and 2D CNN model which are evaluated by AUC score which can be potentially used as a new solution for cancer diagnostics with Mueller polarimetry in the future.
- A confusion matrix visualization that allows users to see the performance of each model in terms of sensitivity and specificity.

Bibliography

[1] F. G. R. O. A. P. J. R. J. V. T. N. Meredith Kupinski, Matthieu Boffety, "Polarimetric measurement utility for pre-cancer detection from uterine cervix specimens," pp. 1–6, 2018.

[2] D. S. W. Joseph A. Cruz, "Applications of machine learning in cancer prediction and prognosis," p. 1, 2007.

[3] J. G. Y. Lou, R. Caruana, "Accurate intelligeble models with pairwise interactions," pp. 1–3, 2013.

[4] A. M. Abdul Ghaaliq Lalkhen, "Clincal tests: sensitivity and specificity," pp. 1–2, 2008.

[5] *NCI Dictionary - CIN 2/3*. National Cancer Institute.

[6] R. R.-K. N Thekkek, "Optical imagine for cervical cancer detection: solutions for a continuing global problem," 2008.

[7] J. e. a. A. Pinkert, M.A Westreich, "A multiscale mueller polarimetry module for a stereo zoom microscope," 2019.

[8] J. G. L. E. Garcia-Caurel, A. De Martino, "Application of spectroscopic ellipsometry and mueller ellipsometry to optical characterization," 2013.

[9] A. N. T. N. A. P. F. M. S. Deby, B. Teig, "Ex vivo mueller polarimetric imaging of the uterine cervix: a first statistical evaluation," pp. 1–5, 2016.

[10] Arunava, "An introduction to convolutional neural networks," 2016.

[11] L. L. T. Z. J. J. M. B. Wensi Tang, Guodong Long, "Rethinking 1d-cnn for time series classification: A stronger baseline," p. 1, 2020.

[12] H. Z. Jiang Su, "A fast decision tree learning algorithm," pp. 1–2, 2006.

[13] N. K. Kain, "Understanding of multilayer perceptron(mlp)," p. 1, 2018.

[14] Y.-W. T. Geoffrey E. Hinton, Simon Osindero, "A fast learning algorithm for deep belief nets," pp. 1527 – 1554, 2006.

[15] J. Heaton, *Introduction to Neural Networks with Java*. Heaton Research, 2005.

[16] A. Gupta, "Understanding activation functions in neural networks," pp. 1–8, 2018.

[17] J. L. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," pp. 1–7, 2015.

[18] T. T.-S. G. S. R. C. K. A. S. J. Y. N. D. T. B. H. S. V. M. B. Wei Luo, Dinh Phung, "Guidelines for developing and reporting machine learning predictive models in biomedical research: A multidisciplinary view," 2016.

[19] A. Lenail, "Nn-svg publication ready nn-architecture schematics," 2019. Visualization tool.

[20] J. Brownlee, "A gentle introduction to the rectified linear unit (relu)," 2019.

[21] J. M. D. Judith E. Dayhoff, "Artificial neural networks, opening the black box," 2001.

[22] J. M. B. Douglas G Altman, "Diagnostic tests 1: sensitivity and specificity," p. 1, 1994.

[23] G. W. M. Danielle B. Cooper, Jose Carugno, *Conization of Cervix*. StatPearls, 2020.

[24] A. B. P. V. H. C. T. N. B. H. I. S. M. C. F. M. A. A.-D. M. Angelo Pierangelo, André Nazac, "Polarimetric imaging of uterine cervix: a case study," pp. 1–3, 2013.

[25] G. J. S. Cassandra j. Beard, Subitchan Ponnarasu, *Excisional Biopsy*. StatPearls, 2020.