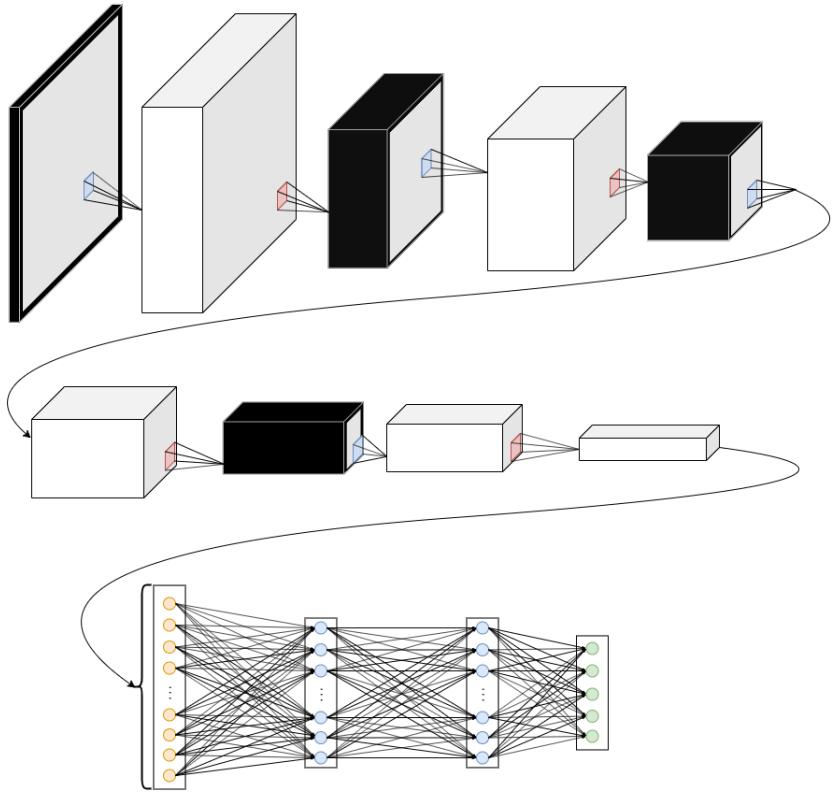
Detection of Breast Cancer in Mammograms

using Convolutional Neural Networks



Project Report
sw514e19

Aalborg University
Electronics and IT

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Detection of Breast Cancer in Mammograms

Theme:
Embedded Systems

Project Period:
Fall Semester 2019

Project Group:
sw514f19

Participant(s):
Abiram Mohanaraj
Christoffer Hansen
Elisabeth Niemeyer Laursen
Fredrik De Frène
Simon Park Kærgaard

Supervisor(s):
Giovanni Bacci

Copies: 1

Page Numbers: 107

Date of Completion:
December 19, 2019

Abstract:

The report explores how machine intelligence can be used in order to detect breast cancer in mammograms. The mortality rate of breast cancer is increased if the cancer is not detected early, and false-negative diagnoses are, therefore, fatal at worst. Different diagnosis techniques were investigated, and classification was chosen as the most suitable, since the different diagnoses can be treated as classes. Furthermore, neural networks and convolutional neural networks were chosen as the preferred method of machine learning since it works well with the images in the CBIS-DDSM dataset. A 'One Vs. All' (Divide & Conquer) strategy and a strategy for classifying all five diagnoses simultaneously were tested. The Divide & Conquer strategy yielded better results but might contradict itself, i.e. two different models diagnose the same image positive. The classifier working on all different diagnoses at once shows some levels of uncertainty, but won't produce contradicting results. In conclusion, both strategies give an educated answer based on the input.

Abiram Mohanaraj
<amohan17@student.aau.dk>

Christoffer Hansen
<ch17@student.aau.dk>

Elisabeth Niemeyer Laursen
<elaurs17@student.aau.dk>

Fredrik De Frène
<fdefre16@student.aau.dk>

Simon Park Kærgaard
<skarga17@student.aau.dk>

Preface

This report is written by group sw514e19, 5th semester Software on Aalborg University, and turned in 19th December 2019. The project theme for the semester is "Embedded Systems". The project will focus on using machine intelligence to detect breast cancer in mammograms.

The project is inspired by two of the group members mothers being diagnosed with breast cancer in Spring 2019.

Frameworks
To construct the models, the framework Tensorflow 2.0 has been used. Tensorflow 2.0 is a end-to-end open source platform for machine learning. It contains the Keras package, which can be used to build and optimise models.[1]

Directional Hypothesis
The type of hypothesis used throughout **Chapter 7** are one-tailed directional hypothesis that predicts the nature of the effect of the independent variable on the dependent variable[2].

Contents

Part I

Introduction & Analysis

1 Introduction

Primary malignant tumour in the mammary glands, more commonly known as breast cancer, is the most frequent type of cancer for women worldwide in 2018, making up 25,5% of all cancer diagnoses[3].

This is also the case in Denmark. The mortality rate of breast cancer is not as high as other types of cancer, but regardless of this it is still the most frequently diagnosed compared to other types of cancer as shown on **Figure 1.1**.[4] Early detection of breast cancer is key for appropriate treatment and greatly increases the probability of survival for the patient, and allows for a wider variety of treatments[5]. The most common examination of breast cancer is an x-ray scanning called a mammography. Examining mammograms makes it possible to detect signs of breast cancer up to three years before symptoms appear.[6]

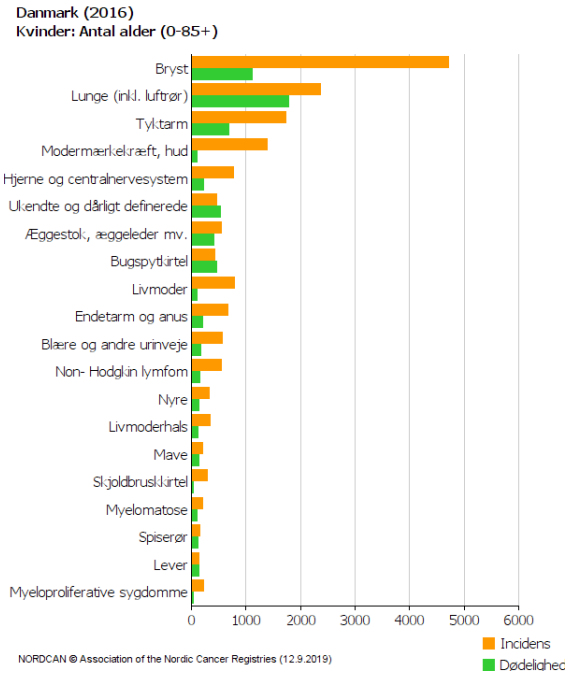


Figure 1.1: Shows the correlation between incidents and mortality among 20 of the most common cancer variations. The orange colour indicating amount of instances diagnosed and the green indicating mortality. Breast cancer can be seen at the top followed by lung cancer[4].

The screening process for breast cancer is costly as it requires radiologists to spend a significant amount of time and resources to examine the diagnosis tests for breast cancer[7]. A specialised system could be able to assist health care professionals in the screening process. It could lessen the burden of examining the mammograms by discarding a significant amount of the mammograms which the program could identify as having no risk of breast cancer. In Denmark, 95-98% of the mammograms are negative meaning that they show no sign of cancer or other types of malignant tumours[8]. This leads to the following initial problem:

Is it possible to ease the workload of radiologist in their task of diagnosing possible breast cancer patients with the use of software?

2 Problem Analysis

The goal of this chapter is to shed light on the screening process of breast cancer. It will also examine the features that define the diagnosis. To this end a variety of currently available detection methods will be reviewed to find a suitable approach for investigating the initial problem. These methods mostly focuses on detection of cancer using artificial intelligence, which includes segmentation, feature extraction, feature selection, and classification.

2.1 Breast Cancer

A tumour occurs when cells in the body develop and divide into new cells too fast. Tumours are commonly thought of as a negative term, but not all tumours are inherently malignant. A tumour can also be benign, which means it is not considered cancerous. It consists of normal looking cells just in an increased number and does not spread to the surrounding tissue. Malignant tumours on the other hand consists of mutated cells that keeps dividing into new mutated cells that spreads beyond the original tumour. Deciding whether a tumour is benign or malignant requires an examination of the tissue.[9] Early detection of malignant tumours is paramount to the survival rate of the patient.[10] Methods hereto will be described later in this section.

The diagnosis of breast cancer is defined as primary malignant tumours in the breast, which can be staged into four states. State 1 is when the tumour has a diameter less than or equal to 20 mm. State 2 occurs when the tumour is between 21 mm and 50 mm. State 3 occurs when the tumour is beyond 55 mm. State 4 is when distant metastasis occurs. Metastasis is a cancer that spreads to another part of the body from the origin.[11]

Besides tumorous mass, breast cancer can also manifest itself as calcifications. Calcifications are small calcium deposits that occur naturally in breasts and has two subcategories: micro and macrocalcification. Macrocalcifications are commonly found in women over age 50, and one in ten women under 50. They are considered non-cancerous. Microcalcifications are the type of calcification that can be cancerous. If they are occurring in specific patterns or clusters, it might be a sign of early breast cancer or precancerous cells.[12]

2.2 Screening Process of Breast Cancer

The usual procedure for detection of breast cancer in Denmark consists of the *triple test* which includes a feeling of the breast by a doctor, a mammography and/or ultrasound scanning of the breast, and finally a biopsy. Other than the aforementioned, methods like Breast MRI are also occasionally used. The purpose of these examinations is firstly to diagnose a patient negative or positive of cancer, where negative means that the patient is cancer free and vice versa. Secondly, it has the purpose of determining how far the disease may have progressed.[13]

Numbers from the Danish breast cancer screening program shows that for every 10.000 women that participate in the process 9750 will get a negative diagnosis. The remaining 250 women with a suspicion of breast cancer, 185 gets the suspicion disproved while the last 65 is diagnosed positive with breast cancer.[14]

Mammography The process of screening for breast cancer begins with a screening mammography. In Denmark, all women age 50-69 are given this offer with an interval of two years on the behalf of the Danish Public Health Authority. A mammography is done using a low dose X-ray machine. Two mammograms are taken of each breast. These mammograms will then be examined by radiologists. A mammography can reveal changes in the breasts as small as one or two millimetres. Changes this size are not detectable by a doctor.[7]

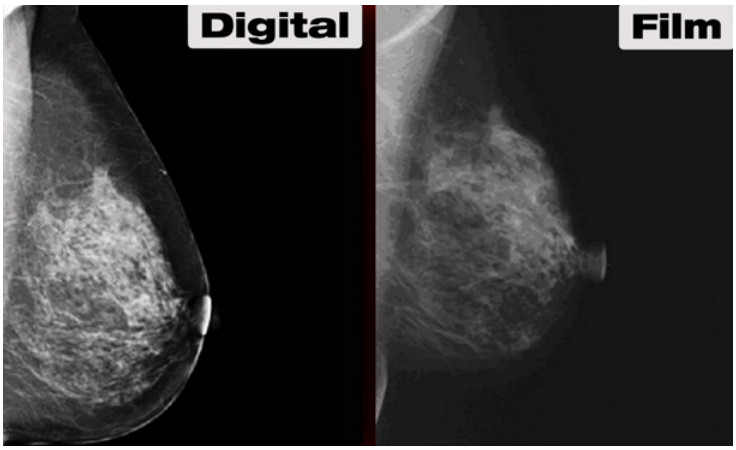


Figure 2.1: The differences of quality in digital and film mammograms.[15]

There are two ways mammograms are taken, one being the traditional, the other being the digital mammography. In the traditional mammography a quick burst of x-ray is fired at a detector photographic film plate on the other side of the breast. This produces a physical film strip which the doctors can examine. The film would have to be digitalised to be viewed on a computer.[16]

A digital mammography follows the same procedure, but it fires at a solid-state detector which can directly send the picture to a computer for previewing. [16]

Clinical Mammography If there is any suspicion of breast cancer from the initial mammography, a more thorough analysis of the breast tissue is required. This is done by a clinical mammography. A clinical mammography consists of x-ray images taken from three different angles and a physical examinations of the breasts and lymph nodes performed by a doctor. This process is occasionally supplied by an ultrasound examination.[7]

Ultrasound Ultrasound is useful when it is necessary to examine breast changes like lumps or changes in women with dense breast tissue. It is especially useful to determine the difference between fluid-filled cysts, which is unlikely to be breast cancer, and solid masses, which would need further testing. The ultrasound examination technique makes use of sound waves to create a picture of the inside of breasts. A transducer is used to send out sound waves and picks up echos which are made into a picture on a computer.[17]

Breast MRI Magnetic Resonance Imaging (MRI) uses a magnetic field, radio waves and a computer to create pictures of structures within the breasts. MRI is usually used in supplement to mammograms or to screen women with a high risk for breast cancer to evaluate the extent of cancer.[18]

Biopsy If the mammography and/or ultrasound examinations show a lump or other signs of cancer, a biopsy is made. A biopsy is a small tissue or cell sample of the area. By examining this sample in a microscope, doctors are able to determine whether there is a case of breast cancer or precursor to breast cancer. Sometimes the entire lump is needed to make this examination.[7]

With a brief look at the detection methods it is known that, even though they share the final purpose of screening patients for breast cancer, they collect different types of data. For this reason, we have to decide which method is most suitable to investigate in order to respond to the initial problem.

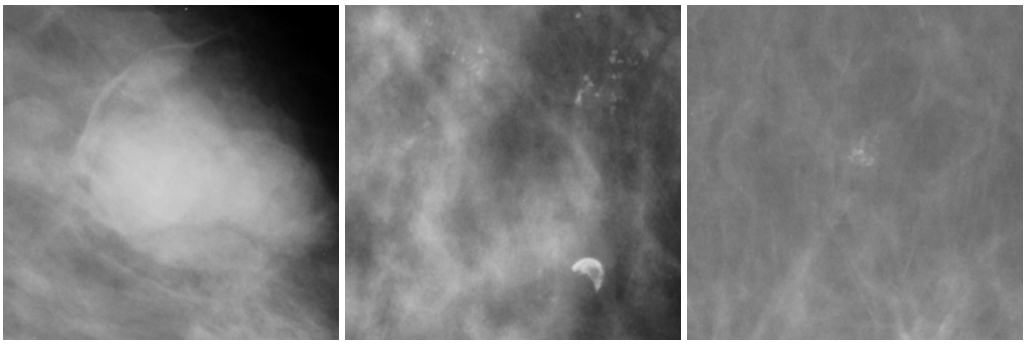
This leaves us with a *learning problem*, which consists of three components: A *task* or *behaviour* that is being improved, the *data* that is utilised to increase performance, and a *measure of improvement* in the form of metrics.[19, p. 268] We are already familiar with the task and we now know that each detection method gives different data. Therefore, different datasets for each detection method should be examined.

2.2.1 Datasets

Below three open-source datasets are presented. The first dataset, the Digital Database for Screening Mammography (DDSM) contains a vast set of digitalised mammograms labelled with different classifications of diagnosis. Furthermore, a Curated Breast Imaging Subset of DDSM (CBIS-DDSM) exists which can be used for analysing. Combined these datasets contain information regarding the specific diagnosis the patient received. The images are of type greyscale, which is what you would get from taking a mammography in a real-world scenario. Examples are shown in **Figure 2.2**. Each

image in the dataset is a tile of size 299x299 from a full mammogram. The dataset includes 55.890 training examples, of which 14% are labelled positive and the remaining 86% negative. The images are labelled with two labels:

- label_normal: 0 for negative and 1 for positive.
- label - full multi-class labels: 0 is negative, 1 is benign calcification, 2 is benign mass, 3 is malignant calcification, 4 is malignant mass.[20]



(a) Malignant mass. **(b)** Malignant calcification. **(c)** Benign calcification.

Figure 2.2: Cutouts from CBIS-DDSM.[20].

The second dataset, namely the Wisconsin Breast Cancer (Diagnostic) Data Set from 1995, contains results from biopsies. It contains the tissue dimensions of cancerous cells. A study of this type of dataset would be driven by patterns of values that already have been measured by medical professionals.[21] The ten features contained in the dataset can be seen in **Table 2.3**, which a diagnosis would be drawn from.

Real-valued features	
1	Radius
2	Texture
3	Perimeter
4	Area
5	Smoothness
6	Compactness
7	Concavity
8	Concave points
9	Symmetry
10	Fractal dimensions

Table 2.3: Ten real-valued features are computed for each cell nucleus.[21]

The third dataset is BACH's, issued by the ICIAR2018 Grand-Challenge. It is a well-documented dataset containing 400 highly detailed imagery of microscopy with 4 classifications, normal, benign, *in situ* carcinoma and invasive carcinoma. Each picture is of size 2048x1536 pixels, labelled, uses the RGB color model, and is previewed on **Figures 2.3 - 2.5**. Additionally the dataset contains 'whole-slide images' depicting the entire tissue that has been extracted from a patient, this image being of sizes up to 42113 x 62525 pixels.[22]

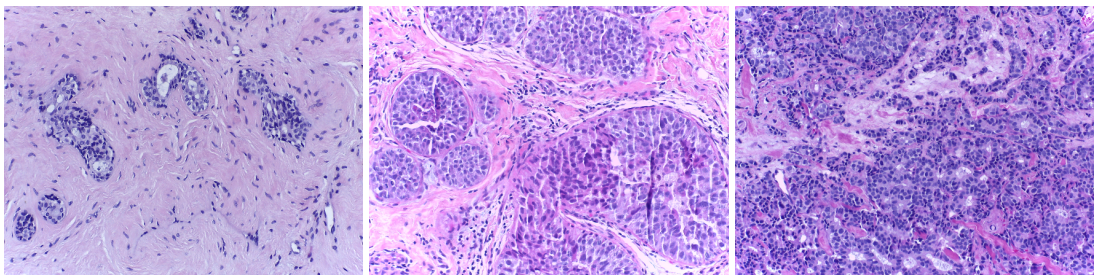


Figure 2.3: Benign.[22] **Figure 2.4:** *in situ* carcinoma.[22] **Figure 2.5:** Invasive carcinoma.[22]

2.2.2 Evaluation of Screening Process

The entire screening process is divided into the three steps as there is a certain waiting period between each of them. This is due to both human verification of each step and a resource cost involved in documenting each choice and diagnose. The mammography is the earliest step in the screening process and is, therefore, the examination type most amount of patients will encounter. Due to this and the large number of mammograms that need to be stored, examined, verified, and documented by health personnel, it would be an ideal part of the screening process to optimise. Another important aspect is early detection is paramount to the survival rate of the patient.

This choice also affects the dataset needed for the solution. The datasets viewed in the previous section addressed different steps of the screening process. The Wisconsin and the BACH dataset is not relevant for the mammography part of the screening process. However, the DDSM database contains suitable information in the dataset for establishing a learning problem for determining if a mammography contains anomalies that warrant further inspection or not.

2.3 Ethics

Computer programs aiding in detection of breast cancer can give rise to ethical and medical-legal issues. A presentation of ethical and other relevant issues will follow.

A relevant problem is how the patients involved in the treatment would react to a machine diagnosing their treatment. Connie Lehmann, chief of breast imaging at Massachusetts General Hospital in Boston states the question in an article from the NRP Health News: "*whether women will be comfortable having this potentially life-or-death task turned over to a computer algorithm.*"[23]. She asked this question to a patient whereto the answer was: "*giving that much control to a computer is 'creepy'*"[23]. The patient does however state: "*as long as a trustworthy human being is still in the loop, she's OK with empowering an algorithm to read her mammogram*" [23]. Thus, including a medical specialist in the screening process could be necessary.

Another obstacle is gaining a positive public opinion in technology especially in the aftermath of Computer-Aided Detection (CAD). CAD is a computer program which is intended to help doctors and radiologists read mammograms. It highlights areas on the mammograms that may be abnormal. Research has shown that CAD might not be as useful as intended, in that it does not improve the accuracy of screening mammograms[24]. Due to the experience with CAD, the public opinion might be inclined to disapprove a similar system.

2.3.1 False-negative and False-positive

False-negative screening results are an ethical concern in the task of computers aiding in making life dependant choices. A false-negative result means that the patient has been diagnosed cancer free, when in reality the patient has cancer.

As described by the American Cancer Society, 1 out of 5 women with cancer receive a false-negative result after a mammography. On the contrary, the amount of false-positives is more prevalent compared to the false-negatives [25]. Thus, resources in terms of time and expenses are wasted, which could be used more efficiently on other patients. There is also an ethical issue regarding these results for the patients, as it is psychological taxing to get a diagnosis of cancer.

2.4 Existing Diagnosis Algorithms

To solve the initial problem, we will briefly review the existing methods and state of the art algorithms existing in the field of mammographic diagnosis. The intention is to define the benefits pertaining to the chosen methods.

2.4.1 Segmentation

The goal of segmentation in detection of breast cancer is to pinpoint the Regions of Interest (ROI) containing abnormalities in the breast tissue. Segmentation can be used in a number of ways in mammography analysis. A few of the relevant studies on segmentation in mammograms will be presented.[26]

One approach is a fractal modelling and segmentation for the enhancement of microcalcification in digital mammograms. The basic idea in this method is to differentiate the properties of disease patterns and background patterns. A separation of the image into different layers can then be accomplished. The algorithm works by first creating a fractal model through segmentation, which is afterwards subtracted from the original image. This will create a new resulting image, where the background structures would be removed.[27]

Another technique that uses segmentation is Radial Gradient Index (RGI) segmentation in mammograms. In RGI, a Gaussian function is used to suppress the influence of background pixels in image data. From a series of contours in the image data, the one whose margin gives the highest RGI value will be chosen as the one that best delineates the lesion. This exploits the fact that lesions are usually brighter than the surrounding breast tissue.[28, p. 511-512]

2.4.2 Feature Extraction

After pinpointing the ROI in the mammograms through segmentation, it is then possible to calculate features for each ROI from the characteristics of them. Commonly, when using computerised analysis to identify microcalcifications, shape-based features are identified. Unfortunately, due to the poor quality of mammograms it can in some cases be difficult to identify microcalcifications only from the extracted shape features. Another possible approach is to also analyse the intensity variation and the texture information of the ROI, which contains important information for the diagnosis. This can also help determine whether the microcalcifications are benign or malignant.[29]

The prior technique described is for a single mammogram. Another computer-aided detection technique is Interval Change Analysis, which is the analysis of multiple mammograms from a single person gathered over time. It analyses the changes to the breast tissue within a given time interval. This is accomplished by extracting the features of the two mammograms in relation to each other.[30]

2.4.3 Feature Selection

On their own, not all features extracted from the ROI are relevant for classification. Therefore the feature selection phase removes the irrelevant or less important features for the model to achieve a higher accuracy.[31] A technique for this could be a genetic algorithm, which differentiates the ROIs as either mass or normal breast tissue and removes the ROIs containing normal tissue. By removing the ROIs with features of normal tissue, this has proven to increase the accuracy of classifying whether the mass is malignant or benign.[32]

2.4.4 Classification

Classification is categorising new input, based on a training set of observations whose category membership is already known. In mammographic image analysis, classification algorithms classifies lesions as either benign or malignant based on selected features from feature selection.

One approach for classifying microcalcification as either benign or malignant is texture analysis. This approach makes use of the fact that texture features in mammograms can be used for differentiation of masses and normal tissues, and that morphological features can be used to distinguish malignant and benign clustered microcalcifications. The central idea is to train a classifier on texture features in mammograms. It can then be used on these features from mammograms to predict whether the microcalcifications are benign or malignant. This approach makes use of the assumption that the texture in regions with malignant microcalcifications could be different from texture in regions with benign microcalcification.[33]

2.4.5 Evaluation of Existing Diagnosis Algorithms

The four existing state of the art diagnosis algorithm types are usually implemented in a chain. This means that errors occurring in any step of the diagnosis process will accumulate and persist throughout the entire process. Due to the complexity of implementing all four algorithm types to the fullest potential, the focus for the rest of the project will be on classification algorithms. Using classification, it will be possible to determine which mammograms contain abnormalities, and whether they are malignant or benign. This could result in a reduction of the workload of diagnosing patients, as radiologists could be assisted in determining in whether a mammogram shows cancer.

2.5 Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence that provides systems with the ability to learn how to accomplish their objective autonomously rather than programming the required functionality explicitly. Machine learning algorithms can solve learning problems like the one proposed in 2.2.

ML methods can broadly be divided in three main categories: unsupervised, supervised, and reinforcement learning. Unsupervised learning is commonly used in scenarios where labelled data is not available or hard to come by. The training data is in most cases a collection of examples where the outcome or answer is not explicitly known, and the model examines the underlying structure or distribution in the data.[34]

Supervised learning on the other hand is performed by relying on a labelled dataset that the model can consult during the training. After it is trained sufficiently, the model will, if presented with unlabelled data, compare it to the training data to make an educated decision. A supervised learning approach is generally endeavoured to achieve a better model for the learning problem proposed in **Section 2.2.2**, with the only drawback being it requires more extensive data.[35]

Reinforcement learning is essentially learning through interaction. A reinforcement learning agent communicates with its environment and can learn to adjust its own behaviour in response to the incentives it receives by observing the effects of its actions.[36]

For this project, a supervised learning approach would procure the best results, as the problem at hand can be categorised as a classification problem[35], and considering a fully labelled dataset is available. Supervised learning covers many different algorithms and to choose an appropriate algorithm for the project requires additional knowledge regarding them. Three different algorithms have been examined below:

Neural Network Neural Networks (NNs) are inspired by the neurons in the brain to simulate learning and decision making. They are designed to recognise patterns by interpreting data through classification. NNs can be used to classify images. This is accomplished by training a neural network with large amount of images of the subject domain. The NN will then learn to extract features from images to match different labels and classify images into the labels. Among a variety of NN architectures, convolutional neural networks are most commonly applied for image analysis, making them a good candidate for classifying mammography scans.[19, p. 308-310]

k-Nearest Neighbours k-Nearest Neighbours (kNN) is an algorithm used for classification and regression. With the help of supervised learning it is possible to perform classifications with a kNN algorithm. The idea is to compare test images with training data. Then to perform the comparison, the algorithm compares each pixel of the test image to the pixels in the training data. The distance between the pixels can be measured with a distance formula. By measuring how close or far between the pixels are from each other, the algorithm can determine if the images are alike.[37, p. 48]

Support-Vector Machine

Support-Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression. All data lies within an n-dimensional space. The amount of dimensions depends on the number of features of the data. The value of each feature decides the data points coordinate.[19, p. 317]

The SVM's task is to use classification to find the *hyperplane* that segregates the two *classes* of data points the best and maximises the distance between them. To find the optimal hyperplane *support vectors* are used, thereby giving the algorithm its name. These support vectors are the frontiers of each data point from each class that lies closest to the hyperplane. Because of this they are the most difficult data points to classify.[38] SVM is well suited for classification of complex but small or medium sized datasets[39].

The three methods presented each have their own advantages and disadvantages. k-NN has limitations when dealing with large amounts of data[37]. NNs on the other hand can make predictions cheaply once the model is finished training. However, NNs have large training periods in regards to learning from datasets. Similar to k-NN, the SVM has limitations when the dataset contains large amounts of data[39]. NN, including convolutional neural network, is deemed the most appropriate machine learning approach to classifying the mammograms in the dataset.

2.6 Embedded System

An embedded system would be embedded in the hospitals' existing examination process. The problem is optimising the processing of mammograms. Therefore, a solution should be embedded into the part of mammographic analysis handling the data of the process.

As mentioned in **Section 2.2**, mammograms are either taken digitally or traditionally through film strips. The film strip can be digitalised. A possible solution could be embedded in the system that is responsible for making the film strips of mammograms digital. Digital mammography is a digital image directly produced by the mammographic machine. However, without access to a mammographic machine it cannot be implemented in practice.

3 Problem Definition

To gain an understanding of the health personnel's workload and to determine where in the breast cancer screening process the biggest potential for reducing the workload can be found, each step of the standard breast screening procedure in Denmark has been examined. The earliest step in the process in most cases, the mammography, was discovered to be the most promising step to optimise as most patients have to go through this process. For this purpose a fitting dataset was found from the DDSM database. It contains an applicable amount of labelled training examples in the form of mammograms. Having a medical specialist in the diagnosis process was also discovered to be necessary due to the concerns of a computer determining life-altering choices.

For analysing mammograms certain algorithm types are used, such as segmentation, feature extraction, feature selection and classification. Due to the level of complexity with implementing all types of the state of the art algorithms, a classification algorithm has been considered sufficient for the purpose of determining abnormalities, and whether they are malignant or benign. Different classifiers has therefore been examined to determine the most suitable for the case of analysing mammograms. The classifiers k-NN and SVM has limitations with large amounts of data, which could potentially cause issues. Among the NN architectures, convolutional neural networks are commonly used for the purpose of image analysis and are on account of this deemed the most suitable architecture.

These conclusions have led to the problem statement:

How can recent advances in machine learning be exploited to reduce the workload of specialised personnel to analyse mammograms without increasing the risk of false positives and false negatives in breast cancer diagnosis?

Part II

Design & Training

4 Design Specification

The decisions for which requirements are considered necessary are based on the problem analysis and the problem definition.

The requirements for the system will be presented. To categorise the different requirements, the MoSCoW rules have been exercised[40, p. 140]. The MoSCoW rules classifies the requirements into 4 subsections.

- The *Must have* (M)
- The *Should have* (S)
- The *Could have* (C)
- The *Want to have but Won't be this time around* (W)

Index	MoSCoW	Requirements
R.1	M	The model must be able to classify the different diagnoses in the dataset.
R.2	M	The model must take a greyscale mammogram as input.
R.3	M	The prediction made by the classifier must create a probability distribution of the diagnoses.
R.4	M	The solution must include a neural network specialised for pictures (CNN).
R.5	M	The model must be evaluated based on relevant metrics.
R.6	S	The input image should be processed to match the format of the dataset examples.
R.7	S	The program should include an interface.
R.8	C	Based on the prediction of all mammograms, the program could create a prioritised list of patients.
R.9	W	The classifier will be embedded in an existing system at a hospital practising mammography.

Table 4.1: Table of requirements with index number, MoSCoW classification, and requirements specification.

5 Analysis of Neural Networks

As concluded in ***Section 2.5****, NN architectures such as convolutional neural network were deemed most suitable for the purpose of image analysis. Therefore, the fundamentals and mechanics of NNs will be examined. Convolutional neural networks will thereafter be described to further extend.*

5.1 Neural Networks

A neural network can be considered a function approximation and consists of *neurons* which are structured by layers; the input layer, *n*-hidden layers and an output layer. Depicted on **Figure 5.1** is a fully connected *feed-forward* NN where each neuron in a layer is connected to all neurons in the following layer.

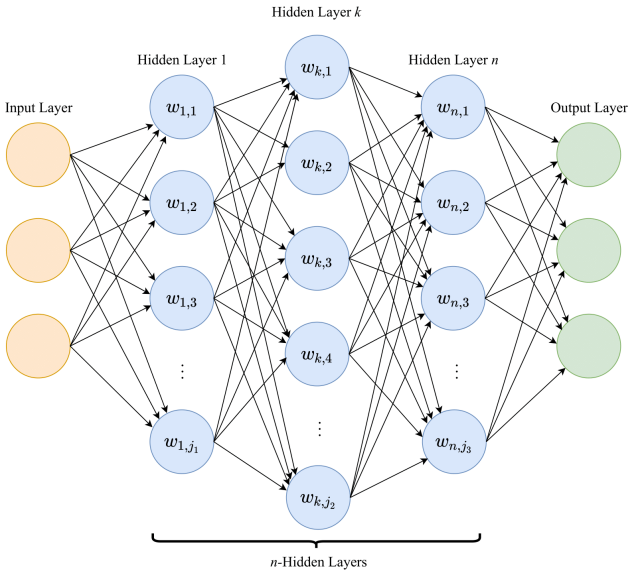


Figure 5.1: A neural network with *n*-Hidden Layers where k < n.

Input layer The input layer is where the data is fed into the model for processing in subsequent layers. This layer, unlike the hidden layers of the model, do not carry any weights and consists of passive neurons that simply pass on the input values without any prior information.

Hidden layer(s) The hidden layers consist of an arbitrary amount of neurons. The purpose of the neurons in these layers is to aid in classification of the input. Different activation functions can be applied to the hidden layers to introduce non-linearity.

Output layer The last part of the NN architecture is the output layer. It is possible to have multiple neurons in the layer. The output holds the prediction made by the model.

The process of predicting and optimising the model consists of performing a forward-propagation for the prediction, measuring the performance of the predictions with a cost function. Lastly, back-propagation is performed with regards to the cost function to adjust and optimise the model.

5.1.1 Forward-Propagation

The forward-propagation process is the act of transferring values forward through the neurons; the output from one layer will be passed to the next layer in the NN's architecture. The output of a neuron is calculated by firstly multiplying the output from the previous layer with their associated weights and adding a bias. Afterwards using an activation function on the calculated value yields the output of a neuron.

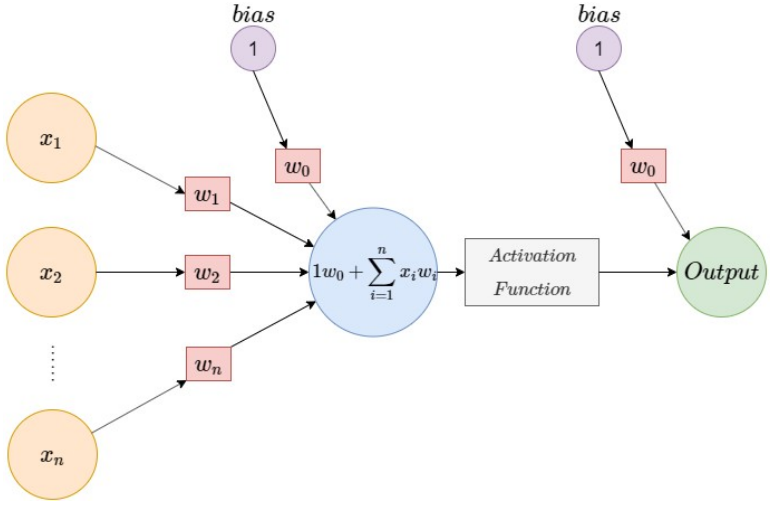


Figure 5.2: A simple NN.

Figure 5.2 shows another view of a simple NN. The weights, $w_1, \ldots w_n$, and biases, w_0, are parameters that have an impact on the predictions. Biases indicates the tendency to prefer one estimation over another. The weights indicates the strength of the connection from input neurons, $x_1, \ldots x_n$, to the neurons in the subsequent layer. The activation function is a mathematical equation that is used to

calculate the output of a neuron, and it determines whether a neuron should be *activated* by considering the relevance of its input compared to the model's prediction. The activation function is also used to introduce non-linearity in the NN, which makes it possible to do more complex predictions (more categories than binary).

The input neurons and their associated weights in the calculation of a layer's output are usually expressed through vectors and matrices. The input in **Figure 5.2** can be considered as $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$, and the associated weights would be $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]$. The bias $\boldsymbol{b}$ is considered a vector with its weights being the components. The activation function is $f()$. The output of the hidden layer can then be expressed as the following mathematical formula:

$$Output_{HiddenLayer} = f(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b}) = f(\begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} w_0 \end{bmatrix})$$

If the hidden layer consisted of an additional neuron, the associated weights would be added as an additional row in $\boldsymbol{w}$, and the associated bias would be added as an additional component in $\boldsymbol{b}$. The equation above describes the general method for forward-propagation from one layer to the subsequent layer. For this example the activation is undefined as multiple activation functions exists.

Activation Functions

The most common activation functions and their distinctive features will be investigated. They include Sigmoid, Tanh, ReLU, Leaky ReLU, and Softmax.

Sigmoid Sigmoid is a logistic function on the form: $\sigma(x) = \frac{1}{1+e^{-x}}$. It has the property of normalising the output of each neuron in the interval between $[0, 1]$, commonly used for probability values. Sigmoid suffers from vanishing gradient. This problem occurs as the function saturates, meaning that large and small inputs would only yield outputs with insignificant difference. Furthermore, sigmoid is not zero centred. This is not optimal in backpropagation, as during gradient descent all weight values will either be adjusted positively or negatively. At last, Sigmoid is also computationally expensive as an exponent is part of the equation.

Tanh The Tanh function, known as the hyperbolic tanget, is a different approach in attempt to solve some of the issues regarding sigmoid. Tanh unlike sigmoid, is 0 centred, which means the range of values it produces lies between $[-1, 1]$. Tanh is formally defined as $f(x) = tanh(x)$ The underlying issues however remain, as values nearing either end of the spectrum produces little to no differences to the subsequent layers. Computationally this operation is as expensive as the sigmoid function.

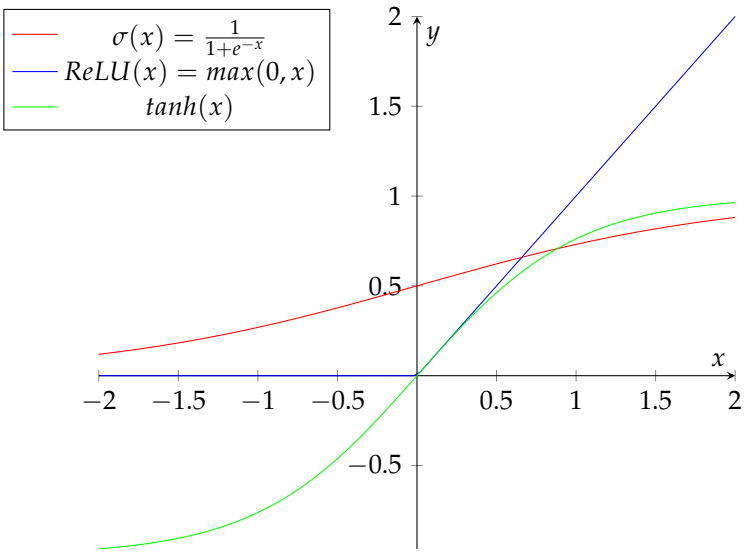


Figure 5.3: Graph showing the three most popular activation functions.

ReLU Rectified Linear Unit, ReLU, is what most modern applications use. Formally it is defined as $f(x) = max(0, x)$. It only allows positive values to pass through to the next layer. This means that if the input value is negative it will produce 0, otherwise the output will be the given input value. Another reason for its popularity is the decreased complexity compared to sigmoid and tanh[41]. The actual implementation of ReLU is to threshold the values in the matrix to 0. ReLU is not a perfect activation function either, as known problems like the *'dying ReLU'*-problem exist, which can cause connections between neurons to 'die'[42]. This means that on certain input, when the gradient is large as a result of the function, future connections can die for the rest of the training. This can also be related to having too large training steps.

Leaky ReLU Leaky ReLU is an attempt to solve the *'dying ReLU'*-problem by adding a slope for values that are negative. Defined as $f(x) = max(cx, x)$, it will allow few negative values to pass onward to the next layer. c in the equation is a constant, which usually is optimal at 0.01. However, research shows that the performance difference in Leaky ReLUs and ReLUs is negligible.[43]

Softmax If the goal is to create a distribution over probabilities of the input given to the layer, then the softmax function is an appropriate choice. Softmax takes a vector as input and calculates the probability of each value in comparison to the others. This means a distribution between 0 and 1 is created of all input values regardless of size. Formally described softmax is as following:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

The sum of the probabilities that softmax outputs will therefore be equal to 1. This is convenient when making predictions and communicating the different probabilities to the user.

Based on the observations made on the examined activation functions, it can be concluded that some have features better suited for for the desired model than others. Computation is an important factor when training and ReLU seems like the optimal choice for the hidden layers. For the output layer, a distribution over the different diagnosis would be helpful to the domain experts when diagnosing the patient. Therefore Softmax is most suitable for the output layer of the model.

5.1.2 Backpropagation

Backpropagation is how the NN updates its weights and biases between its neurons when training. After forward-propagation, a cost is calculated through the *cost function*. This is then propagated backwards in the NN to update the weights and biases, in order to make more precise predictions. This is a key concept in NN's 'learning', and is only applicable in a supervised learning approach.

The main idea with the backpropagation in NN is to calculate the partial derivatives $\frac{\partial C}{\partial w}$ and $\frac{\partial C}{\partial b}$ with respect to any weight w and bias b. The derivatives are used to update the weight and biases to better suit the predictions made by the model. The weights are updated through gradient descent. The derivatives used to update the weights can be calculated with some resulting equations derived with the help of the chain rule. The resulting equations will be presented along with their purposes. In order to calculate the derivative of the cost with respect to each weight in the NN, the cost for each layer must be calculated. First, the cost for the output layer has to be calculated with the following calculation:

$$\delta^L = \nabla_a C \odot f'(z^L)$$

where δ^L is a vector with the components being the cost for each weights in layer L. $\nabla_a C$ is a gradient vector with the component being the partial derivatives $\frac{\partial C}{\partial a_j^L}$, and a_j^L is the activation or output of a neuron. $\odot$ denotes the Hadamard product between two vectors. The function f is the activation function. z^L is a vector with the components consisting of the values gained by $\mathbf{x} * \mathbf{w} + \mathbf{b}$. After calculating the cost of the output layer, it is possible to calculate the remaining layers with the following equation:

$$\delta^L = ((w^{L+1})^T * \delta^{L+1}) \odot f'(z^L)$$

When δ^L is known, it is possible to calculate $\frac{\partial C}{\partial w}$ with:

$$\nabla_w C = a^{L-1} \odot \delta^L$$

where $\nabla_w C$ is a gradient vector with its components being $\frac{\partial C}{\partial w^L_{jk}}$. The relation between the change in cost with respect to the bias and cost is expressed in the equation $\frac{\partial C}{\partial b^L_j} = \delta^L_j$. This means that the gradient of the cost with respect to the bias is exactly equal to the cost of a neuron.[44, chap.2]

Cost Functions

The cost function, used throughout the backpropagation, is a formula used to asses the quality of a prediction compared to the actual output. When deciding on a cost function two assumptions have to hold true. The cost function should be differentiable in order to calculate the backpropagation. Secondly the cost function should take the output of the NN as the input to the function. Presented here are three cost functions proposed for the project.

The Mean Square Error (MSE) takes the mean of the squares of the errors $(Y_i - \hat{Y}_i)^2$. Formally, this corresponds to the following cost function:

$$C(Y, \hat{Y}) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

where n is the number of output neurons, Y_i represents observed values, and $\hat{Y}_i$ represents predicted values.

MSE is not exclusively used as a cost function for NN's. But, when applied as a cost function, it is used to measure how the network performs according to the expected outcome.

Cross entropy, also known as log loss, is a cost function that determines how close the prediction is to the expected value. The product of cross entropy is a value between 0 and 1. The desired value for a perfect prediction would be 0. A categorical version of cross entropy exists for multi class entropy. On **Figure 5.4 & 5.5** are the formula for cross entropy and categorical cross entropy.

$$C(Y, \hat{Y}) = -\sum_{i=1}^{n} \hat{Y}_i * \log(Y_i)$$

Figure 5.4: Cross Entropy

$$C(Y, \hat{Y}) = -\sum_{j=1}^{m} \sum_{i=1}^{n} \hat{Y}_{ij} * \log(Y_{ij})$$

Figure 5.5: Categorical Cross Entropy

The cost functions parameters are Y = the predicted values and $\hat{Y}$ = the expected value and iterates over the values from the output layer's activation function.

Gradient Descent

Gradient Descent is a typical optimisation technique used with most learning algorithms in ML. A gradient is the slope of a function, which is derived from the change of a parameter with the amount of change in another parameter. It is the partial derivative of parameters with respect to its inputs. Gradient Descent is essentially an iterative method to determine the values of the parameters of a

function that will reduce the cost function as much as possible. Commonly, there are three main types of Gradient Descent:

Batch Gradient Descent	Is the act of performing gradient descent on the entire training set. When training sets become large this approach can be problematic. It is computational demanding to train on the entire dataset at once. Another issue with this approach is that the NN is more likely to fall into the local minimum and not search for the global minimum.
Mini-Batch Gradient Descent	One of the solutions to an ever-growing dataset is to perform mini-batch gradient descent. In this approach, the dataset is cut up into smaller *batches* in which training is performed upon. The sizes range from 10-1000 training examples, however, studies show that the most consistent and best-performing size for a batch is 32.[45]
Stochastic Gradient Descent	In short, SGD takes a single random training example from the dataset and uses it for training. This reduces the computations required drastically but requires more iterations, and is prone to randomness when searching for a minimum.

5.1.3 Consideration for Neural Networks

Training an NN is highly affected by the data fed into the model. To get the optimal output different considerations have to be made.

Overfitting occurs when the model makes predictions based on patterns that can be observed in the training set but are absent in the test set. In practice, overfitting is observable when the model has a noticeably lower accuracy on the test data than the training data. There are several methods available to attempt in combating this issue. More training data is generally thought of as the best approach when dealing with overfitting issues, but regularisation is another method that could be used. Regularisers allow penalties to be applied during optimisation on layer parameters or layer activity. Underfitting, on the other hand, indicates that the model doesn't adjust enough on the training and likely more training has to be performed. This is also likely to give an undesired output.[19, p. 298] Both scenarios are depicted on **Figure 5.6**.

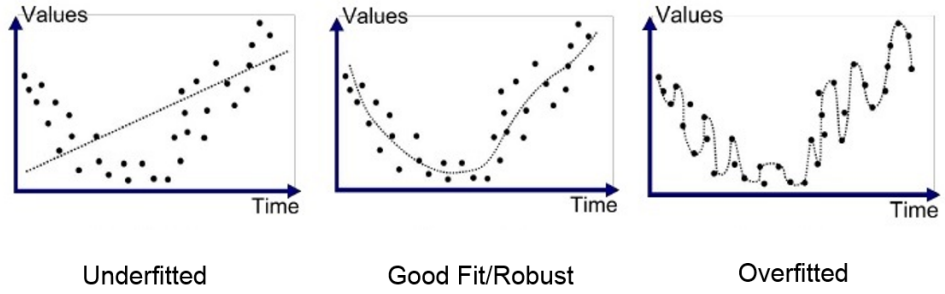


Figure 5.6: Shows the scenarios with underfitting, overfitting and optimal fit.[46]

High variance in the dataset can cause incorrect estimation of the model's performance. If features are under-represented in the dataset the model will have difficulties learning the specific features. Over-representing a feature in the dataset can cause the model to overfit for that specific feature. Therefore, it might be important that the distribution of each feature in the dataset is balanced.

5.2 Convolutional Neural Network

Convolutional Neural Networks (CNN) are designed to process grid structured inputs, which have strong spatial dependencies. An example of such inputs is images, as adjacent spatial locations usually have similar values in terms of individual pixel values. An important reason why CNN has high compatibility with image data is that the data posses a certain amount of translation invariance. This means that an object has the same representation regardless of the location in the image.[47]

A CNN consists mainly of two layer-types, the convolutional layers, and the pooling layers. In addition to the aforementioned layers, it can also possess fully connected layers similar to a traditional NN as the last part for classification. The first fully connected layer is constructed by having each neuron in the last spatial layer connected to each neuron in the fully connected layer.

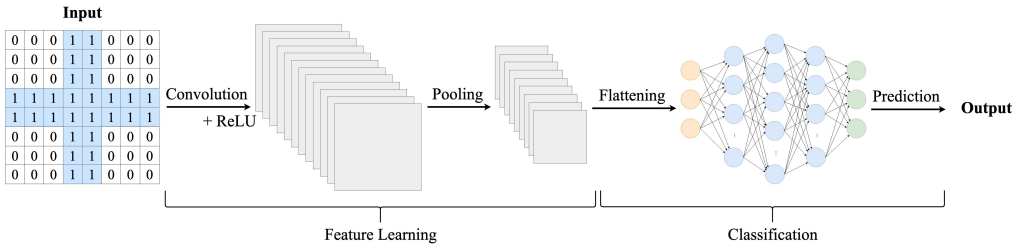


Figure 5.7: A possible architecture of a Convolutional Neural Network.

A possible architecture of a CNN is seen on **Figure 5.7**. The convolutional layers and pooling layers can be interleaved. In the convolutional layers, filters of various sizes can be applied to the input to create feature maps, also known as activation maps. In the pooling layers, filters are used to emphasise important features in the input data. Pooling layers are commonly used directly after convolutional layers. Lastly, fully-connected layers similar to the traditional neural network architecture is used for classification.

Convolutional Layer

In convolutional layers, a convolution operation is applied to the input to activate the next layer. A convolutional layer consists of filters/kernels, where each filter is used in the convolutional operation to calculate a feature map by repeatedly using the filters on the input.

The weights or components of the filters are the parameters of the convolutional layer, and the depth of the filters should be equal to the depth of the input.

The convolutional operation is an operation where the filter is placed at every possible location in the input and calculating the dot product between the overlapped input and the weights in the filter. It is possible to have multiple filters in the same layer. This will result in an increased amount of feature maps.[47, p. 319]

Stride is essentially the notion of the movement of the convolutional filter on the input. A stride is denoted by horizontal movement and vertical movement of the filter. In the case of the input being an image, stride size of 1 means the filter moves one pixel at a time. Larger strides can also be used, and the size of the stride affects the size of the resulting output feature map, in addition to the receptive field in each of the features in the hidden layers.[47, p. 324]

Performing the convolution operation on image data will lower the size of the input. This sort of reduction means that information along the borders of the input will often be lost. This can be avoided by applying padding. Padding adds zeroes around the border of the input. The padded part will not contribute to the dot product calculation.[47, p. 322] Half-padding is a padding technique, that pads relative to the filter size. Using half-padding will result in the output of the convolutional layer to be of the same dimension as the input provided that the stride of the convolutional layer is 1. For a filter of size F equals to $(F-1)/2$ amount of padding.[47, p. 322-323]

Pooling

The pooling layer is mainly used generalise the input data. However, a side effect of this layer is reduction of the input's height and length. Pooling does not affect the depth of a layer, as for each feature map, based on the values and the pooling type, a new feature map is created. This is unlike a convolution operation, which works on all activation maps simultaneously with a filter to create another feature map.

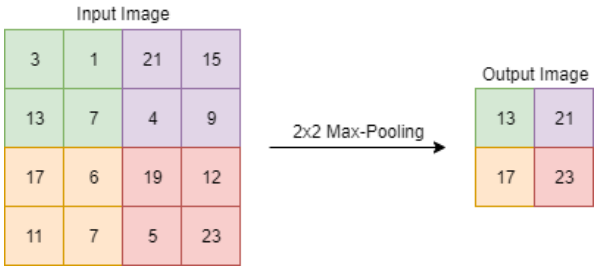


Figure 5.8: Performing 2x2 max-pooling with stride 2 on a 4x4 image.

Three types of pooling exist, namely min-pooling, average-pooling and max-pooling. **Figure 5.8** depicts the application of a 2x2 max-pooling layer with a stride of 2 on a 4x4 image. As the name implies, max-pooling will in this case select the highest value in the 2x2 region, and insert the value in the new feature map to be created. The pooling layers are implemented between selected convolutional layers, but can occur less frequently than these, because only a few drastic pooling layers are required to reduce the spatial map to a desired size. Minimum pooling selects the minimum pixel value, and average-pooling selects the average of all the pixels.[47, pp. 326-327]

Fully-Connected Layers

After the input image has been convolved and pooled, the extracted feature maps can be passed into a series of fully-connected layers. Naturally, since the layers in a fully-connected network is of vector-dimensionality, the output feature-matrices from the convolutional layers must first be flattened into vector format. As described in **Section 5.1.1**, activation functions like ReLU, sigmoid and tanh can be applied to introduce non-linearity to the model.

Backpropagation in Convolutional Neural Network

Backpropagation in CNNs can be viewed as a transposed convolution, where the values of the filter is calculated and updated with respect to the loss of it's output, specifically the partial derivaties of the loss function with respect to that filter. Similar to the traditional backpropagation described in **Section 5.1.2**, backpropagating through convolutions is also calculated with a form of matrix multiplication of the error derivatives for a particular layer.[47]

To compute the loss gradients for a layer q, a vector of the loss gradients for each cell of a layer (q + 1) is multiplied to the weights in the transposed filter between layers q and (q + 1) thereby calculating the loss gradient of the layer q.[47]

Then to update the values for weights, the loss gradients are calculated for each cell in the layer, which can then be compared to the weights in the filter. However the weights in the filter are shared across the entire spatial layer, therefore partial derivatives are calculated for every cell in the layer with respect to a particular weight. Thereafter all partial derivatives for that weight are added up to calculate the new value of the weight.[47]

6 Model Architecture

After the analysis of the structure and techniques associated with CNNs, the focus will now be directed to our implementation of the classifier. Based on the problem statement, two CNN architectures will be proposed. We will afterwards define a model structure that will serve as the initial model.

6.1 Architectures

When classifying breast cancer, two CNN architectures have been considered. The first approach is a single CNN that will produce a probability distribution of the five diagnoses (negative, benign calcification, benign mass, malignant calcification, and malignant mass) as the output. This model is displayed in the **Figure 6.1**. The second approach is inspired by the *"one-vs-all"* method where five specialised CNNs each specialise in classifying one particular class, or in our case diagnosis, in the dataset.[48, p. 576] These architectures are shown on **Figure 6.2**.

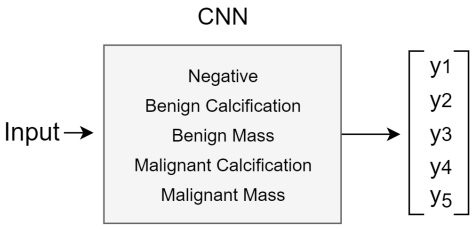


Figure 6.1: Illustration of the FD model's output.

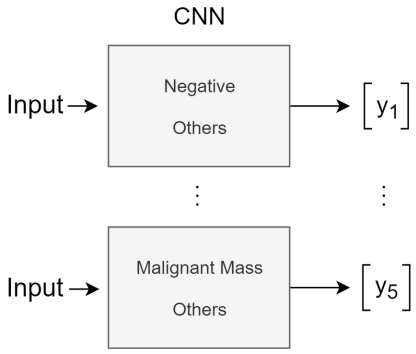


Figure 6.2: Illustration of the DC model's output.

6.1.1 Five Diagnoses Architecture

The Five Diagnoses (FD) Architecture consists of a single CNN. The CNN will take data in the form of a 299x299x1 matrix as input, which is the size of each image in the dataset. It will then calculate a probability distribution of the five diagnosis types as the output. This probability distribution will contain the probability of each of the five diagnoses, and the diagnosis with the highest probability will be the prediction. The prediction would be the output neuron with the highest probability value. This architecture encapsulates all features of the dataset in one CNN, meaning that the CNN should be able to distinguish between every class in the dataset.

6.1.2 Divide and Conquer Architecture

The idea behind the Divide and Conquer (DC) Architecture is to split the problem of classifying five different diagnoses at once, to a possibly more manageable binary classification. This architecture consists of five CNNs; one to distinguish each diagnosis from the others in the dataset. To implement this architecture on all of the features in the dataset, it is necessary to process the dataset into five individual specialised datasets which are explained in **Section 7.2.2**.

The first CNN's output will distinguish between negative and positive labelled images. In this case, the positives are any diagnosis that differs from negative. The four other CNN's will predict whether their respective diagnosis (e.g benign calcification, malignant mass, etc.) is present or not. This could substantially reduce the training time, as fewer features need to be captured, thereby reducing the time required to reach an applicable accuracy. It also has the potential to reach better accuracy levels. A possible problem using this model is the risk of contradictory outputs, meaning an image could be diagnosed with multiple diagnoses, as none of the CNN's have any knowledge of the other's predictions.

6.1.3 Evaluation of Architectures

The two approaches both have pros and cons. The FD Architecture would not predict contradictory outputs due to the fact that every feature is considered during training. This architecture could also capture potential correlations between the features in the dataset that the DC Architecture would miss. While it is possible to get contradictory output using the DC architecture, the training time would be shorter and the prediction could also be more accurate.

As both architectures could solve the problem statement, they will both be evaluated during the training phase. Testing all five CNNs in the DC Architecture would be too time consuming, so only two CNNs will be implemented as a proof of concept. The chosen CNNs are the models for negative classification (NP) and malignant mass classification (MM).

6.2 Initial Model Structures

The model structures presented in this section is the initial models for the two CNN architectures and will be used as the model for training during the first tests. It will be based on intuition and reasoning drawn from the previous theory chapter and refined through **Chapter 7** as hypotheses of the models are tested.

The first convolutional layer of all three models takes input in the form of images with the dimensions 299x299x1. Information along the borders of the input can be important. Therefore applying padding to the input is necessary, as the first convolutional layer will have filters of the size 3x3x1, and the stride will be 1. The padding applied will require two layers of pixels with a value of 0 around the image.

The models will consist of 4 convolution layers of varying amounts of filters, 4 max-pooling layers, 1 flatten layer, 2 fully connected layers with a fixed amount of neurons, and 1 output layer. The convolution layers and max-pooling layers are interleaved. The fully connected layers will each have 512 neurons in the FD model, and 256 neurons in the DC models. The reasoning for the differing amounts of neurons is due to the fact that the FD architecture needs to capture more connections. The output layer will vary depending on the type of architecture. The FD architecture utilises softmax as the activation function to the output layer, which consists of 5 neurons. The DC architecture's output layer also uses softmax but only has 2 neurons. The convolutional and fully connected layers will all use the ReLU activation function, and the weights of the layers are initialised with a random Gaussian distribution. The cost function for the initial models is categorical cross entropy. The gradient descent strategy for the initial models will be the stochastic gradient strategy in conjunction with a batch size of 32. An illustration for the initial model can be seen on **Figure 6.3**.

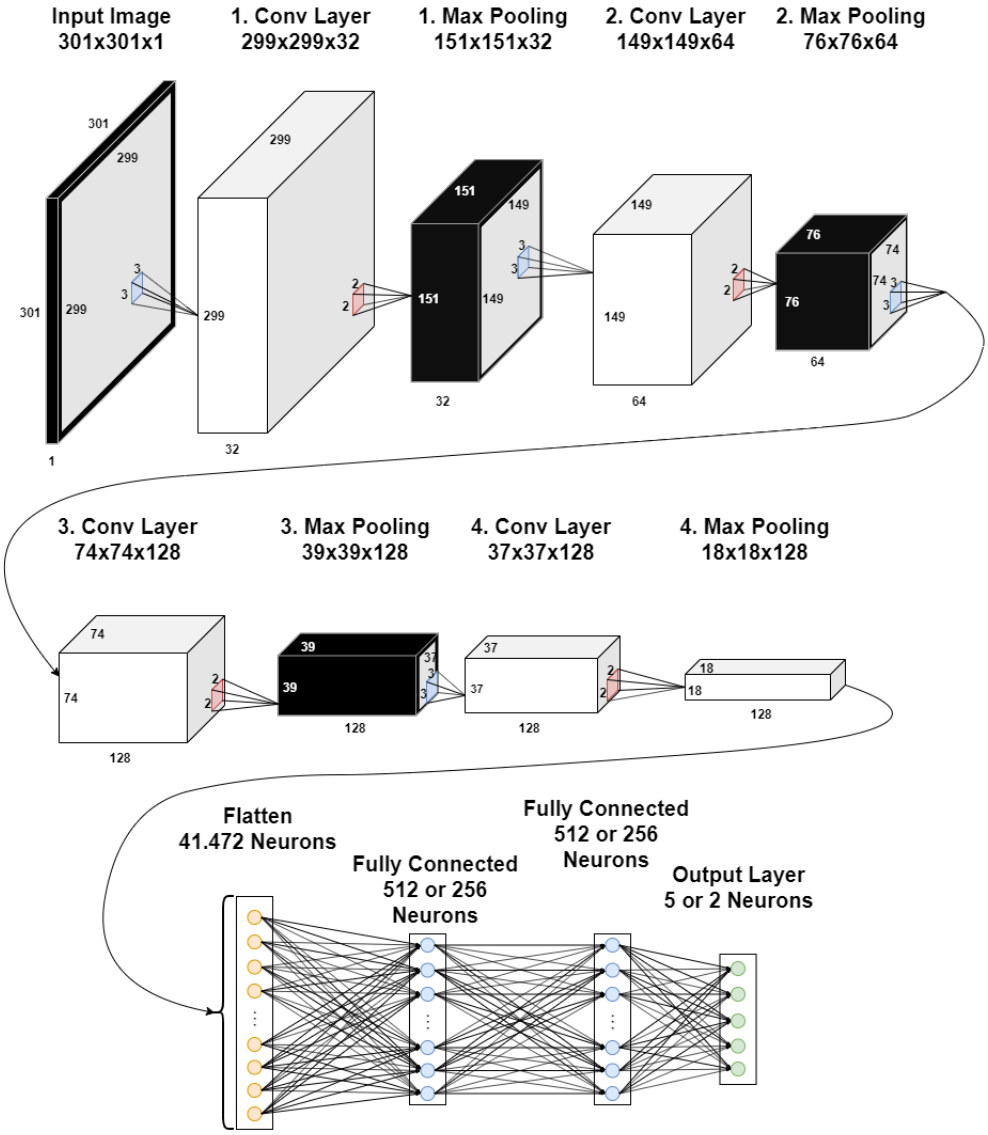


Figure 6.3: The architecture for the initial CNN. Surrounding the convolutional layers is the half-padding. Note that the last amount of neurons in the fully connected layers varies depending on the type of model (FD or DC).

7 | Training

After defining the initial models for the two architectures, we can now proceed to improve the internal structure of them. The improvements will be implemented based on hypotheses, whereto tests will be made to confirm or disprove them. Evaluation metrics will be introduced as a key concept to measure the improvements and to choose the most optimal result moving forward to the next hypothesis test. The hypotheses will, in general, relate to techniques for improving the dataset, gradient descent strategies, and other possible optimisations regarding training the models.

7.1 Evaluation Metrics

Throughout the testing of hypotheses, different evaluation metrics are used to monitor the performance of the models. Due to the models being different in their way of handling predictions, the metrics should suit this.

Five Diagnoses Metrics

The metrics *accuracy* and *confusion matrix* are used to evaluate the FD model. Accuracy describes the overall correctness of the model, and the confusion matrix displays a table that compares the model's predictions to the real diagnosis.

Accuracy Accuracy is the fraction of correctly classified instances. It will be calculated by measuring the correct predictions and dividing them with the total amount of predictions. A correct prediction is a prediction that is equal to the expected outcome. E.g. a prediction and expected outcome could be [0.2, 0.4, 0.1, 0.1, 0.2] and [0, 1, 0, 0, 0]. The prediction would be interpreted as [0, 1, 0, 0, 0] as 0.4 is the highest value in the prediction. As the interpretation of the prediction is equal to the outcome, the prediction would be classified as a correct prediction.

Confusion Matrix

A confusion matrix is a summary of the predictions compared to their true classifications. This creates a distribution of the frequencies of the different diagnoses. Following the previous example, a confusion matrix with this prediction is presented as shown in **Confusion Matrix 7.2**. The interpreted prediction is added to the highlighted row, specifying the expected outcome, which in this case is BC (benign calcification). This representation depicts a correct prediction for BC. The confusion matrix can be used to view the predictions (columns) compared to their true diagnosis (rows). The *Column-Total* shows the total amount of guesses for the specific class, while the *Row-Total* shows the true amount.

	N	BC	BM	MC	MM	*Row Total*
N	**0**	**0**	**0**	**0**	**0**	*0*
BC	**0**	**1**	**0**	**0**	**0**	*1*
BM	**0**	**0**	**0**	**0**	**0**	*0*
MC	**0**	**0**	**0**	**0**	**0**	*0*
MM	**0**	**0**	**0**	**0**	**0**	*0*
Column Total	*0*	*1*	*0*	*0*	*0*	*1*

Confusion Matrix 7.2: Confusion matrix for a single observation of a correctly predicted benign calcification

Accuracy and confusion matrix will be used to monitor and evaluate the model's performance through validation tests. Accuracy will mainly be used to compare the hypothesis results. However, when the confusion matrix provides additional useful details, it will be shown as well.

Goodness of Fit

Chi-squared test for Goodness Of Fit (GOF) looks at two marginals of a table and determines if there is a statistical significant present between the observed values and the expected. Three things must hold true to use the test: The data must be randomly selected from a distribution, the values must be independent, and none of the expected frequencies should be below 5. As we will see later, all of these hold true for us. In order to use the Chi-squared test, one must establish a null hypothesis which will either be accepted or rejected.[49, p. 216-217] The formula is formally defined as:

$$\chi_k^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Where k is the degrees of freedom, O is the observed value and E is the expected. If we consider **Table 7.2**, then the observed value would refer to each $i \in I$ for $I = \{O_N, O_{BC}, O_{BM}, O_{MC}, O_{MM}\}$ in *Column Total*, while the expected value would be each *Row Total*. The *degrees of freedom* is defined as $k = n - 1$ categories, for **Table 7.2** it would be 4. The degrees of freedom and α-level is used for looking up in a Chi Squared Table[1] to evaluate weather to reject the null hypothesis and accept the alternative hypothesis.

[1] **Appendix C** shows the Chi Squared Table

Divide and Conquer Metrics

Below, *accuracy*, *precision* and *recall* are described. The output of each model can be divided into four categories: *true positive*, *true negative*, *false positive* and *false negative*. True positive is a correct prediction for a positive diagnosis. True negative is a correct prediction for a negative diagnosis. False positive is an incorrect prediction for a negative diagnosis. False negative is an incorrect prediction for a positive diagnosis. From the predictions, it is possible to calculate the different metrics.

Accuracy Accuracy describes how often the model is correct. The accuracy is calculated by taking the amount of correct predictions divided with the total amount predictions. It is possible to compute in terms of the four possible outcomes of a binary classifier: $Accuracy = \frac{true\ positives + true\ negatives}{total\ examples}$.[50]

Precision Precision is used to calculate how many times the model correctly predicts a positive diagnosis. It is therefore calculated by taking the amount of true positives and divide it by the sum of true positives and false positives. ($Precision = \frac{true\ positives}{true\ positives + false\ positives}$).[50]

Recall Recall is used when the cost of false negatives is too high. It can be used to determine the rate between true positives compared to false negatives. It is calculated from the amount of true positives divided by the sum of true positives and false negatives. ($Recall = \frac{true\ positives}{true\ positives + false\ negatives}$).[50]

Throughout this chapter, accuracy, precision, and recall will be measured to evaluate the DC models' performance on validation data. In **Section 2.3.1** the importance of false negatives and false positives were showcased, therefore recall and precision could be optimal measurements to monitor. Accuracy, however, displays the correctness of the model in all contexts and a model with high accuracy will likely also have a high recall. Therefore, accuracy will be used as the main evaluation metric.

The test results are plotted in graphs to give the reader a visual representation. To keep this chapter's length to a minimum and reduce redundancy, some graphs might be excluded in the report. A full collection of the graphs can be found for each test in **Appendix B**. We encourage the reader to refer to the appendix while reading this section, in order to better understand our reasoning.

7.2 Data Pre-Processing

The dataset has been partitioned into training, validation and testing sets. The *training set* is the dataset which is used to train the CNN. This data forms experiences that improve the subsequent predictions by the model. *Validation set* is tested during the training and indicates how well the model generalises towards unseen data. The *testing set* is used to evaluate the model after all training is done, this data is different from both the training and validation set. All datasets presented following this section are structured as displayed on **Figure 7.1**.

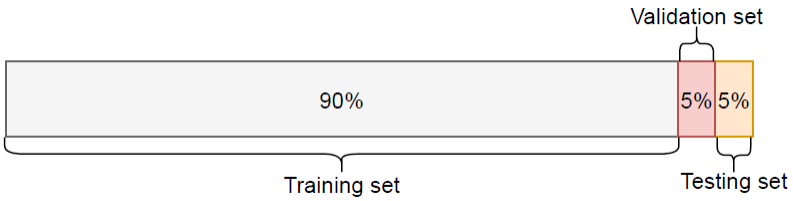


Figure 7.1: Each dataset is divided into three sections, 90% training set, 5% validation and testing set

7.2.1 Original Dataset

The original dataset (introduced in **Section 2.2.1**) consists of 86% negative labelled 0 and 14% positive diagnosis types. Each of the positive diagnosis types are labelled 1 to 4 for the different classifications.

Label Representation

The original dataset is labelled with real values. This representation is shown in **Table 7.5**. This representation might not be usable as it would map a relation between the features in the dataset. The value of benign calcification is 1, and malignant calcification is denoted by 3. This would imply that malignant calcification is equal to three times that of benign calcification, which is an invalid comparison. To avoid encoding this sort of relation, an indicator variable representation has been implemented [19, p.275]. A representation is shown in **Table 7.4**.

Example	Y_0	Y_1	Y_2	Y_3	Y_4
e_1	0	0	0	1	0
e_2	1	0	0	0	0
e_3	0	1	0	0	0
e_4	0	0	0	0	1
e_5	0	0	1	0	0

Table 7.4: Indicator variable for $Y_0, \ldots, Y_4$, if the diagnosis Y_k is present, then it is marked with 1 and rest is 0

Example	Y
e_1	3
e_2	0
e_3	1
e_4	4
e_5	2

Table 7.5: Real-value variable Y, where each diagnosis has a number associated with it.

With the indicator variable representation, the label for an input is represented by an array with the elements being selected features in the dataset. The array will consist of a one-value at the index in the array denoting the target feature of input and zeroes at the other entries in the array.

Proportion of Samples in Dataset

Out of the total 55.890 samples in the dataset, 86% are negatives, thus denoting an imbalance in the frequency of each diagnosis type. On **Figure 7.2** the positive box relates to benign calcification, benign mass, malignant calcification, malignant mass and will be referred to as such through the chapter.

Figure 7.2: The distribution of the entire original dataset

This distribution of data might lead to overfitting as the model does not train as extensively on the positive diagnosis types' data compared to the negative's data. This leads to the first hypothesis:

An imbalanced dataset will cause the model to overfit. (H.1)

We trained the initial FD model on the original dataset with the purpose of confirming **H.1**. Our test shows that for the first 20 epochs the model learns from the training data. The validation accuracy does not increase with additional training and finally ends up with the same as it started with at 86%. It is likely that the 86% validation accuracy is caused by the distribution of negative and positive data in the validation dataset, as the model learns that predicting negative diagnoses yields the best results and thereby the model overfits. This indicates that **H.1** can be confirmed. The fact that mostly negative predictions occur is backed by the confusion matrix produced by the test. The model classifies 90.9% of the validation data as negative. Among the true predictions, the fraction of a negative prediction actually being negative (accuracy) is 97%, while the fractions of the other diagnosis types being true is in the interval of 9% to 45%. This would mean that the model classifies the negative data well but fails to find the traits of the other diagnosis types, resulting in a high level of overfitting.

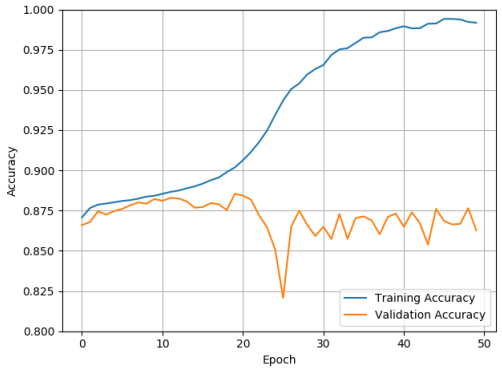


Figure 7.3: Original dataset training accuracy through 50 epochs on the FD model.

	N	BC	BM	MC	MM	*Total*
N	**2353**	**25**	**17**	**18**	**6**	*2419*
BC	**69**	**20**	**4**	**6**	**2**	*101*
BM	**31**	**11**	**30**	**3**	**19**	*94*
MC	**51**	**10**	**5**	**8**	**10**	*84*
MM	**32**	**5**	**10**	**3**	**42**	*92*
Total	*2536*	*71*	*66*	*38*	*79*	*2790*

Confusion Matrix 7.6: Confusion matrix for the original dataset

Changing the distribution of the number of negatives and positive training examples might yield better results. Before we calculate the χ^2 of **Confusion Matrix 7.6**, we need to establish a null hypothesis.

H_0: There is no anomaly between observed and expected predictions.

Usually statistical significant begins at an α-level of 0.05, which would mean that if the value of the table exceeds 9.488, then we should reject H_0 and accept the alternative hypothesis. After calculating χ^2 for **Confusion Matrix 7.6** the outcome is 49.937 giving a p-value <0.00001. The null hypothesis is rejected as we are dealing with α-level of 0.05 in favour of the alternative hypothesis:

H_A: There is an anomaly between observed and expected predictions.

This leads us to the following hypothesis:

A larger[1] balanced dataset prevents the model from dominantly predicting negative. (H.2)

7.2.2 Enhancement of Dataset

From the previous test, it is clear that the distribution of training examples is imbalanced. To counter this, a new dedicated and balanced dataset will be created for each model. To create a binary output of a CNN, the label structure in the dataset needs to be reconfigured. The labels will be structured as described in **Section 7.2.1**. The two new dataset structures will be presented below.

Five Diagnoses Dataset

The new FD dataset is restructured to have an even distribution of training examples in regards to the five different diagnoses. Each diagnosis type will now be represented with a frequency of 20%, as shown on **Figure 7.4**.

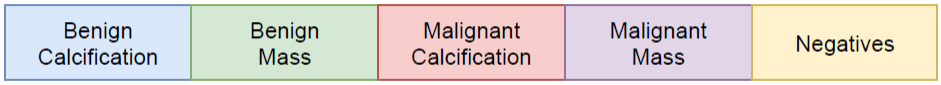


Figure 7.4: The structure of the five category dataset

Divide & Conquer Datasets

The dataset for the DC model(s) will be structured as shown on **Figure 7.5**. As there is a limited amount of data for each of the specific diagnosis types, each dataset will be split into two halves. The first half will be the diagnosis type the model wishes to classify, while the second half will be the remaining diagnosis types. In this example, the dataset is structured to fit the DC model to classify benign calcification.

[1]larger will be defined as an increased amount of positive diagnosis samples

In order to keep an even distribution of training examples in the second half of the datasets, a limit of the number of negatives is needed to avoid running into the preceding problem.

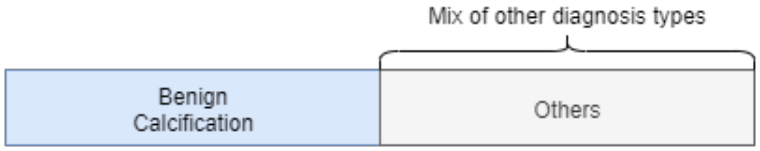


Figure 7.5: The distribution of benign calcification and other diagnosis types for binary classification

The two new dataset structures will be used when training the models in the subsequent tests.

Data Augmentation

Data augmentation is performed to diversify and create *more* data for training, in an attempt to generalise the CNN's predictions for unseen data. A variety of data augmentation techniques and considerations are presented in this section.

Rotation One approach to produce more data is to rotate the images. This technique is also applicable to the images in the dataset. Because each image in the dataset is randomly cropped from a full mammogram, the ROI might be located near the edge of the image. This means important features can be lost if the images are not rotated 90° at a time. **Figure 7.6** depicts this issue.

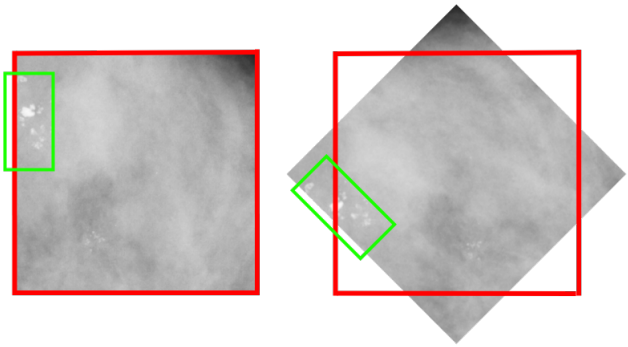


Figure 7.6: Right-hand illustration displays the issue with rotating images 45°, when features exit the original frame.

Zoom Zooming, or cropping and resizing is another technique for creating additional data. However, some training examples suffer from the same issue as with the rotation, where important features can be located near the edge of the image. By zooming in it means that these features will be cropped out and lost. Zooming out would also impact the size of the features represented in the picture. As mentioned in **Section 2.1**, the diagnosis type is dependant on the size of the tumour, meaning zooming in and out could potentially change the diagnosis type.

Noise Additional noise can be added to the original images to generalise for classification tasks. An implementation of noise would be to add random white and black pixels to the image. However, this causes the concern that the added pixels could represent a wrong diagnosis or accidentally cover up the signs of a diagnosis in the mammograms.

Flip All the images can be flipped (across the horizontal axis) or up to down (across the vertical axis), and thereby double the dataset. This method will also not allow for any unwanted distortion as the picture will keep its spacial properties.

The chosen augmentation methods were to first rotate the images 90°, 180° & 270°, and thereafter flip all of them vertically from left to right. In total, this quadruples the amount of data from the original dataset. As there is an abundance of negative images, they have not been augmented. The other augmentation methods, zoom, and noise, are deemed unsuitable to augment the data as they potentially could end up damaging or altering the data.

Applying Data Augmentation to the Balanced Dataset

H.2 has been tested by training the FD model on the enhanced FD dataset. The test results are presented in the **Confusion Matrix 7.9** below. The confusion matrix displays that the model predicts correctly with a higher frequency. The second highest prediction for each row also indicates that the model captures a connection between certain diagnoses. For example, for the calcification predictions the model finds similarities. The accuracy also increases, which confirms **H.2**. To further illustrate the improvements we calculate the χ^2 for the **Confusion Matrix 7.9** and use the same null hypothesis and α-level. The χ^2 is 16.893, which corresponds to a p-value of 0.00203, this means we reject H_0 in favour of H_A again.

	N	BC	BM	MC	MM	*Total*
N	**579**	**69**	**39**	**45**	**10**	*742*
BC	**71**	**566**	**74**	**107**	**36**	*854*
BM	**27**	**76**	**535**	**50**	**97**	*785*
MC	**57**	**143**	**70**	**258**	**43**	*571*
MM	**15**	**37**	**129**	**42**	**465**	*688*
Total	*749*	*891*	*847*	*502*	*651*	*3640*

Confusion Matrix 7.9: Confusion matrix for the test performed on the enhanced dataset.

The results illustrate how the enhancement of the dataset has improved the model's performance. Based on the results it is reasoned that the enhanced dataset should be used going forward.

7.3 Batch Sizes' Impact

We will now test if the batch size will affect the performance of the FD model. The initial chosen batch size of 32 will be compared to 64 and 128, which correlates to the hypothesis below. The dataset is batched by first shuffling the data and then dividing them into batches. As our hardware is limited in terms of memory, we are not able to go above a batch size of 128.

A batch size of 32 will achieve the highest validation accuracy. (H.3)

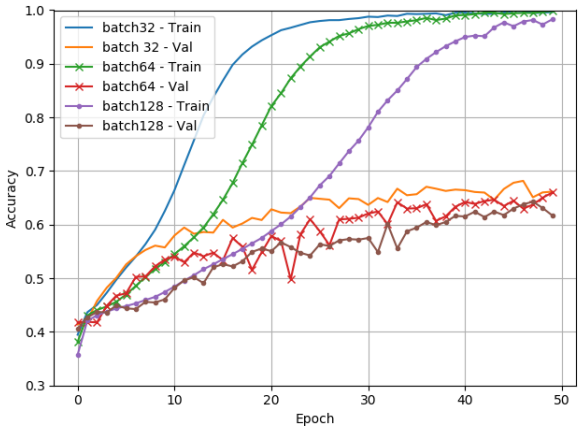


Figure 7.7: Results of training the model with batch size 32, 64, and 128 on the FD model.

The results of the **H.3** test are plotted in **Figure 7.7**. As expected, the 32 batch size reaches the highest validation accuracy. When comparing the three batch sizes it is possible to conclude that increasing the batch size will gradually reduce accuracy.

Consequential, the models will remain with a 32 batch size. The next concern at this stage would be the large amount of overfitting, however, there are techniques to reduce this. We are confident in using these techniques to reduce overfitting and achieve higher validation accuracy.

7.4 Gradient Descent Strategy

Different Gradient Descent (GD) strategies can be used on the two types of CNN architectures to potentially achieve higher accuracy. SGD, which is the GD strategy used in the previous tests, RMSProp, and Adam have been tested and compared to determine which performs better.

RMSProp RMSProp, short for Root Mean Square Propagation, is an optimised version of mini-batch gradient descent. It is a parameter specific learning method utilising "signal-to-noise" normalisation. This refers to keeping track of the magnitude of a gradient. It means that gradients with consistent signs are accentuated rather than inconsistent gradients.[47, p.140-141]

Adam Adam, like RMSProp, is a parameter specific learning method that also utilises "signal-to-noise" normalisation, but with momentum incorporated in the update of weights and biases. An exponentially smoothed value of the gradient is used to update the weights and thereby implement momentum.[47, p.140-141]

The GD strategy chosen has an influence on the training approach used for the models. RMSProp and Adam are similar with the only difference being Adam's incorporation of momentum, so it is expected that Adam would optimise the models better than RMSProp. However, with an exceeding momentum the local minimum could potentially be missed, causing inefficient optimisation. Therefore, the possibility of RMSProp performing better than Adam exist. However, our expectation is that Adam performs better, which leads to the following hypothesis:

Adam is the optimal gradient descent strategy for the models. (H.4)

Gradient Descent Strategies' Application on Models

To test **H.4**, SGD, RMSProp, and Adam has been individually tested on the models.

The results of the FD model shows that Adam initially performs the best. However, around epoch 15 the validation and training accuracy of SGD exceeds the accuracy of Adam and continues to do so throughout the training session. For the NP and MM models, SGD and Adam perform similar, although Adam reaches a slightly higher accuracy.

RMSProp performs poorly among all models. This strategy significantly decreases the accuracy of the models though it is expected not to, because it is optimised to work with mini-batches. This is especially present on the test of the NP model, shown on **Figure 7.8**, where the model loses almost 30% accuracy between the first and last epochs. The issue is less present with the MM and FD model, but the other gradient descent strategies still exceed.

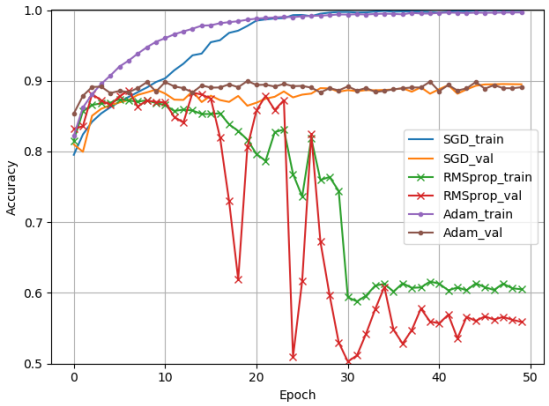


Figure 7.8: Gradient descent strategies applied to the NP model.

Summing up the results of the tests for gradient descent. On the FD model, the SGD strategy performs best. On both DC models, Adam performs best. A common issue with the tests of data augmentation, batch size & gradient descent is the likelihood of

the model overfitting. This is encapsulated in the following hypothesis:

Overfitting can be reduced, while maintaining accuracy. (H.5)

This hypothesis will cover testing of multiple overfitting methods. Thus x amount of sub-hypotheses will be made indexed by H.5.x below.

7.5 Max-Pooling Alternatives

Before testing the different methods of preventing overfitting, it was decided to test max-pooling alternatives. This means the sub-hypotheses of **H.5** will follow in **Section 7.6**.

The initial models use max-pooling layers interleaved between the convolutional layers. As max-pooling would lead to the CNN accentuating the brighter areas of an image, it would be intuitive that max-pooling will perform best due to observations made of the mammograms. This leads to following hypothesis:

Max-pooling will yield the highest accuracy. (H.6)

To evaluate **H.6**, max-pooling have been tested against two alternatives, namely: Average-pooling and min-pooling.

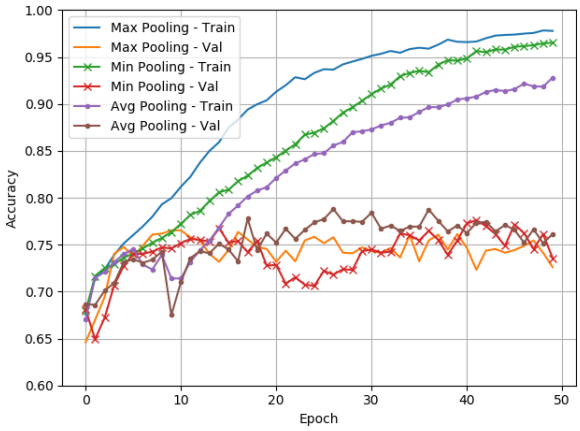


Figure 7.9: Different pooling strategies for the MM model.

After testing the different pooling strategies on the FD and NP models, it is clear that max-pooling is the ideal strategy, because it achieves the highest accuracy throughout the entire training compared to the other strategies. The results of the MM model as shown in **Figure 7.9** indicates that average-pooling is the most suitable pooling strategy. Average-pooling being most ideal for the MM model

could mean that the malignant masses are defined through features that are not necessarily in either end of the grey-scale spectrum.

7.6 Reducing Overfitting

A reoccurring issue in the previous tests has been overfitting. The concept of regularisation was briefly touched upon as a possible solution in **Section 5.1.3**. On that account different countermeasures for overfitting will now be presented. These will then be tested on the models in attempt to reduce overfitting.

7.6.1 Early Stopping

A technique for regularisation is *Early Stopping*, which stops the training of a neural network when it starts to overfit. Early stopping does not interfere with back propagation, but it can analyse through callbacks whether signs of overfitting are present and halts the training if it is. There are different signs for determining overfitting, e.g. when the training accuracy is higher than the validation accuracy.This leads us to the following hypothesis:

Applying early stopping when the val. accuracy stops increasing reduces overfitting. (H.5.1)

Early stopping is applied to the models by implementing callbacks. The callbacks will halt the training in the event of the validation accuracy not increasing within 5 epochs from the highest recorded value, which is an indication of overfitting. This number is reasoned by analysing prior tests, which show that the level of overfitting will likely become unacceptable after 5 epochs without an increase in validation accuracy. As seen in **Table 7.11** all models stopped before reaching 50 epochs, which was previously set to be the maximum, meaning that they all stopped earlier compared to previous tests. This allows for more tests to be conducted and reduces overfitting per our observed indicator of overfitting. This confirms **H.5.1**. Early stopping will be applied to all three models in the consecutive tests. We can also remove the epoch boundary of 50 due to the fact that some tests might require the models to train for longer, and that the models will halt training when their learning regress.

	FD Model	**NP Model**	**MM Model**
Epoch	36	10	19

Table 7.11: Number of epoch the model stopped training caused by early stopping.

7.6.2 L2 & L1 Regularisation

L2 Regularisation, also known as weight decay, is one of the most commonly used regularisation techniques. The intuition is to add a term to the cost function for the weights called the regularisation term. This term is the sum of squares of all weights in the network but does not include the bias. By

adding this regularisation term to the cost function, the model will learn to correctly classify samples with the smallest possible weights. When constricting the size of weights during training, the model is forced to generalise more than if without regularisation. Large changes in weights can only occur if the original cost function can be considerably improved.[44, chap. 3]

L1 regularisation, known as Lasso regression, is a similar regularisation approach to L2. At the end of the weight's cost function, the sum of absolute weights are added as the regularisation term. This is a constant value that is added and can therefore cause the values to be reduced to 0.[19, p. 305][44, chap. 3]

It is expected that among the two regularisation techniques, L2 will perform superior, as it allows for weights to change relative to their value. Because L1 adjusts weights by a constant, this technique carries the risk of no values to be passed through the layer, leaving the model unable to comprehensively reason its prediction.

L2 regularisation reduces overfitting during training. (H.5.2)

Regularisation methods have been applied to the initial models in the fully-connected layers as they have the most direct impact during backpropagation. The applied regularisation is denoted LX/LY, where LY refers to the regularisation method applied closest to the output layer, and LX refers to regularisation method on the second closest layer to the output layer. These regularisation methods should force the model to generalise better and thereby reduce overfitting.

Observations throughout the different tests show the same methods applied to different models and datasets can yield vastly different results. On multiple accounts applying L1 regularisation resulted in the model being unable to learn anything, simply terminating after 5 epochs, as per early stopping. This occurred in the training of both the FD model and MM model. Unlike this, the L1 regularisation seems suitable for the NP model, as it and managed to train for 34 epochs instead of 10 and increased the validation accuracy. Therefore, L1/L1 will be applied to the NP model in the following tests.

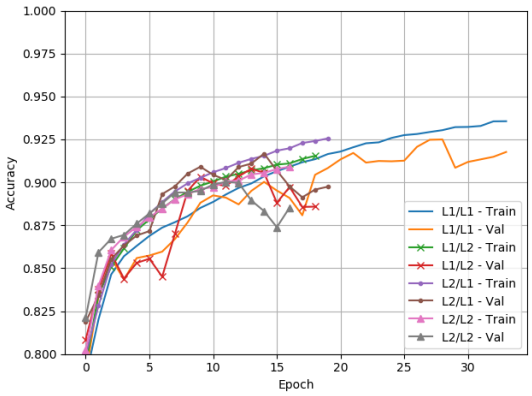


Figure 7.10: Regularisation methods applied to the NP model.

After examining the results of the FD model and MM model, applying L1 & L2 regularisation to the models reduce overfitting but also decrease the accuracy. Due the purpose of this section being to reduce overfitting, regularisation will be applied to models. For the following tests, the FD model will be utilising L2/L1 and the MM model L2/L2. This is illustrated on **Table 7.12**.

	FD Model	NP Model	MM Model
Early Stopping	Applied	Applied	Applied
L1 & L2	L2/L1	L1/L1	L2/L2

Table 7.12: Overfitting technique each model will use.

7.6.3 Dropout

The technique concerns temporarily removing random neurons during training, to prevent the model from making its decisions based on specific connections or spurious features that may be present in the dataset. This has proven to improve the performance of NNs and decrease the generalisation error.[51]

Each neuron is retained with a probability value. This value can be set to 0.5 which is the optimal value for the hidden layers in most networks. For the input neurons it is often more optimal to retain closer to 0 than 0.5. Alternatively, different probability values of dropout can be tested for our particular case.[51]

In our case, dropout in the first convolutional layer might remove important features from the data, as some features can be as minuscule as a single pixel. This leads us to the following hypothesis:

Strategically applied dropout reduces overfitting and increases generalisation. (H.5.3)

All models are tested with a dropout of 10%, 30%, and 50% with regularisation, thereafter without. If 50% proves to be the best dropout value, then additional tests with higher dropout rate will be executed. The reason behind testing dropout without regularisation is to inspect whether dropout alone is more desirable in regards to reducing overfitting. Test results for all three models with and without regularisation have similar outcomes. The results for NP model is shown below in **Figures 7.11** and **7.12**.

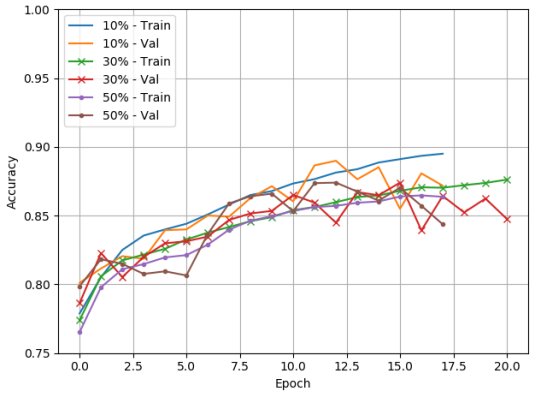


Figure 7.11: Dropout on the NP Model with L1L1 regularisation.

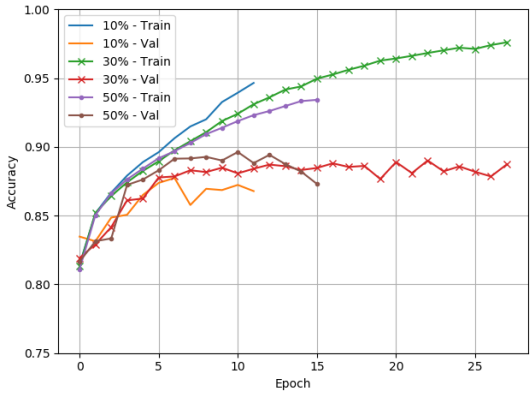


Figure 7.12: Dropout percentage on the NP Model without regularisation.

Adding dropout on top of the regularisation did not entail any positive results for any of the models. Instead it decreases the accuracy by 2% compared to the results of using regularisation alone. Using dropout alone, does not yield any desired results either, as the FD and NP model return to overfitting more than prior to **H.5.3**. The only favourable result from this test comes from adding 30% dropout to the MM model while removing the L2/L2 regularisation. The overfitting techniques moving forward are shown on **Table 7.13**.

	FD Model	**NP Model**	**MM Model**
Early Stopping	Applied	Applied	Applied
L1 & L2	L2/L1	L1/L1	Not Applied
Dropout	Not Applied	Not Applied	30 %

Table 7.13: Overfitting technique each model will use.

7.6.4 Hypothesis 5 Conclusion

A variety of techniques has been examined to reduce overfitting, namely: Early Stopping, L1 & L2 regularisation, and Dropout. Early stopping is found to have a positive effect on the models' performance, as it halts the training once the models' validation accuracy no longer increases.

The outcome of the **H.5.2** test for the NP model yields the best results. The training accuracy decreases and the validation accuracy increases, thus closing the gap between them, using L1/L1. This means the model is able to generalise better. When dropout is added on-top of the previous regularisation methods, the model does not reduce overfitting but decreases the accuracy instead. No desirable results are found when removing the initial regularisation either.

Similar, the FD model shows positive changes when added the L2/L1 regularisation, but negatively when adding dropout. Therefore the FD model will only be using L2/L1 regularisation in the following tests.

The MM model displayes acceptable results from the regularisation tests, decreasing slightly in accuracy but keeping overfitting to a minimum. When applying dropout to the model without utilising regularisation the model shows better results, with an increase in validation accuracy while keeping overfitting at acceptable levels.

7.7 Tuning the Model

For the last part of the training, the models will be tuned by different parameters. As it is impossible to test all possible combinations of hidden layers, neurons, filter etc. the approach will be to loosely define tuning points. By doing this it will hopefully be conclusive when certain characteristics are identified. The models will go through a systematic change of firstly convolutional layers, then increasing the filter size, amount of hidden layers, and at the end neurons. It could be entirely possible that a reversed order could lead to a different outcome, however due to time constraint this is the chosen approach.

7.7.1 Amount of Convolutional Layers

The first part of the tuning process will be to alter the amount of convolutional layers in the CNN. By increasing the convolutional layers the model could potentially capture more complex structures within the image that the current model does not allow for. However, a lower amount of convolutional layers might prove more optimal when capturing simpler structures in the mammograms. When selecting the number of layers we wanted to test both a substantially smaller and larger number of layers compared to the initial four layers. We decided on two, six, eight and ten convolutional layers.

The test results for the FD model clearly shows that four convolutional layers is the optimal decision. There also seems to be a tendency for higher amount of convolutional layer to be prone to overfit to a high degree, while a lower amount of convolutional layers do not perform well either, as the test with two convolutional layers is stopped early without significant effect. For the NP model, the tests of the different amount of layers perform quite similarly, but if investigated closely four convolutional layers seem to perform better than the others. However, as they perform quite similar in terms of accuracy, precision has also been explored, and with precision eight convolutional layer performs best. For MM tests, eight convolutional layer seems optimal as the validation accuracy is highest among the alternatives and the overfitting is to a lesser degree. Thus, for the FD model four convolutional layers and for the NP and MM models eight convolution layers will be used in future testing.

Filter Size

We have chosen to test three different filter sizes, namely 9x9, 7x7, and 5x5. For reference the initial model uses 3x3 in all four convolutional layers. The way the change of the filter sizes will be applied in the tests is by having the biggest filter in the first layer, followed by smaller consecutive filters until the last layer is of size 3x3. The specific filter sizes for each test are shown in **Table 7.14**.

	1st Conv. layer	**2nd Conv. layer**	**3rd Conv. layer**	**4th Conv. layer**		**8th Conv. layer**
9x9	9x9	7x7	5x5	3x3	...	3x3
7x7	7x7	5x5	3x3	3x3	...	3x3
5x5	5x5	3x3	3x3	3x3	...	3x3
3x3	3x3	3x3	3x3	3x3	...	3x3

Table 7.14: Overview of the filter sizes for the filter size tests.

Our intuition is to have the large filters in the first convolutional layers to capture larger structures in the image. The following layers will then, with smaller filter sizes, refine the larger structures into more distinctive features.

When comparing the results of the increased filter sizes, 3x3 seems to perform the best on all three models judged by accuracy. These accuracies are, nonetheless, just within a small margin of 4-5%. An observation that the FD and NP model have in common shows that increasing the filter size leads to a higher degree of overfitting. Looking at the NP model, the validation accuracy begins to decline around epoch 10. The most unexpected results from this comes from the MM model's test shown **Figure 7.13**. The model was unable to learn, when increasing the filter size to anything above 3x3.

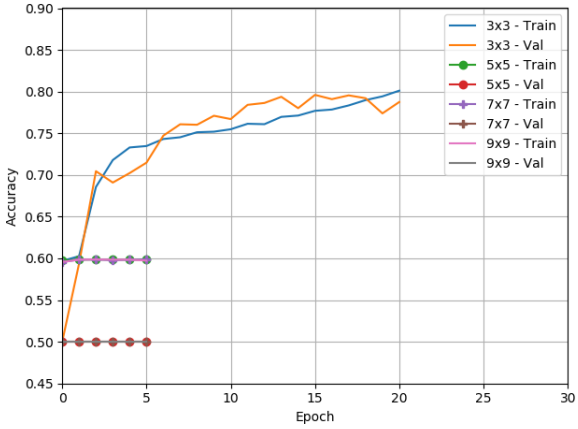


Figure 7.13: Results of changing the filter size on the MM model.

What these results might indicate is that the models are dependant on detailed features to be extracted from the start. Larger filters than 3x3 extracts features that are too generic, thus making the models unable to capture the characteristics of the features.

7.7.2 Amount of Pooling

Tests have been conducted with the purpose of testing different amounts of pooling. The models have been tested with 4, 5, and 6 layers of pooling. If the amount of convolutional layers are less than the pooling layers, the pooling layers will be placed after the last convolutional layer. If the opposite scenario occurs, there will be no pooling layers for the remaining convolutional layers. Two and three pooling layers will not be tested due to hardware limitations.

For the FD and NP models 5 max-pooling layers performs the best, compared to the other amounts of pooling. With the NP model the results of the tests are quite similar, although 5 max-pooling layers has a slightly higher validation accuracy without much more overfitting. For the FD model the validation accuracy is at times higher than the training accuracy, as seen on **Figure 7.14**. The training and validation accuracy graphs also accompanies each other which also suggests less overfitting compared to the earlier test (filter size 3x3 test for FD model).

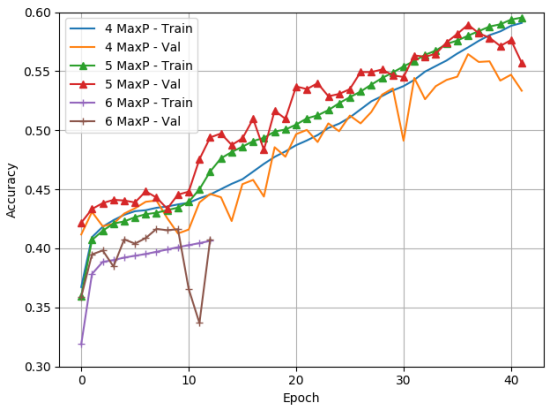


Figure 7.14: Amount of pooling for the FD Model

The MM model performs the best using 4 average-pooling layers. When adding additional pooling layers, the model overfit more and did not reach the same validation accuracy.

7.7.3 Amount of fully-connected layers

Another part of tuning the model is to change the amount of fully-connected layers in the CNN. By increasing or decreasing the amount of layers used to classify the diagnosis could affect the model positively. The prior result of having 2 fully-connected layers will be compared to having 1, 4 and 8 fully-connected layers. When increasing the amount of fully-connected layers it would provide a larger encapsulation for the classification. Thus decreasing the amount of layers would simplify the encapsulation. The regularisation tested and applied earlier will still be applied to the added fully-connected layers.

The FD model reaches the highest accuracy with only having one fully-connected layer with L2 regularisation but with a higher level of overfitting. Therefore after examining the results, both the FD model and MM model prove to perform best with utilising two fully-connected layers, which is chosen for the initial model. Only the NP model shows a difference from the initial choice of layers. As seen on **Figure 7.15**, the highest validation accuracy is reached when only having a single fully-connected layer with L1 regularisation. This will therefore be used for the final test of the NP model.

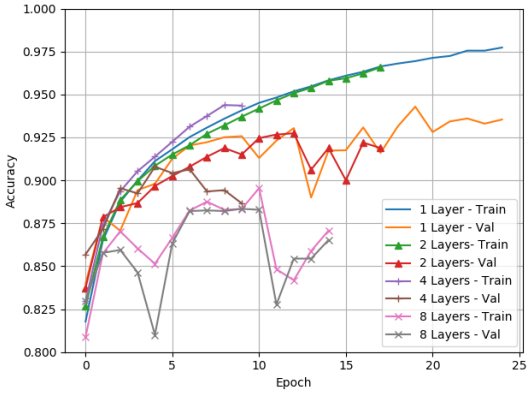


Figure 7.15: Different amounts of fully-connected layers for the NP model.

The overall result of these tests show that having a minimal amount of fully-connected performed most optimal, which can also be viewed on the graph for the NP model. Excessively many fully-connected layers proved overly complex to learn anything from in a reasonable amount of time.

Amount of Neurons

When testing the amount of neurons in the fully connected layers, it was decided to stick to the convention of having neurons equal to 2^k. Each test was performed using 2^{k-1} and 2^{k+1}, in order to observe if any patterns emerge. Therefore, the FD model is tested with 256, 512 and 1024 neurons and the DC models with 128, 256 and 512 neurons. If either test resulted in an improved performance the k would have been further adjusted.

After examining the results of the FD model and NP model, both models performs best with their initial amount of neurons. A reason for this could be that the model is optimised to the chosen amount of neurons. The NP model will therefore continue to utilise the 256 neurons for it's fully-connected layer and the FD model will utilise 512 neurons. Having the final accuracy for the FD model reaching 58% and for the NP model reaching 94.29%.

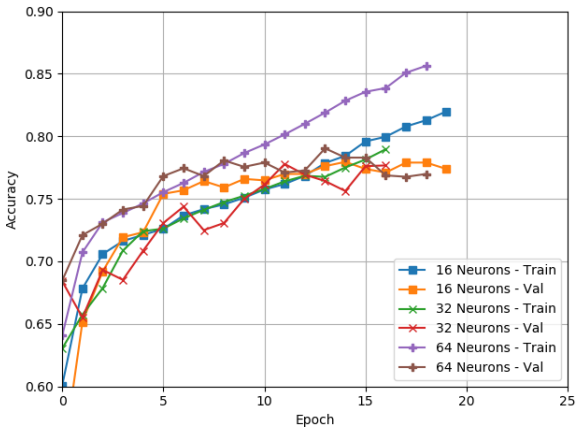


Figure 7.16: Different amount of neurons in the MM models fully-connected layers.

Only the MM model shows in positive changes to the accuracy when decreasing the amount of layers. The results show that with a higher amount of neurons, the model will learn quicker but in turn also overfit more, whereas a lower amount of neurons will keep the training and validation accuracy more aligned. Therefore, as seen on **Figure 7.16**, the model was also tested with 16, 32 and 64 in addition to initial test of 128, 256 and 512. 32 neurons were chosen as the best strategy for the MM model because of how well it generalises. This concludes the training of the models.

Part III

Implementation and Evaluation

8 | Model Evaluation

During previous chapters multiple hypotheses were created and tested on the three models. To get an overview, we will take a step back and look at the difference between the initial model structure and the final model structure. A table will be presented with a comparison of the model's accuracy.

8.1 Five Diagnosis Model

For the FD model a probability distribution over all diagnoses is provided. To evaluate this model, accuracy and a confusion matrix is used. The initial FD model trained on the augmented dataset, peaks at a training accuracy of 99.8% and validation accuracy of 68.1%. The final model peaks at training accuracy of 59.5% and validation accuracy of 58.9%. Even though the validation accuracy decreased in the final model, it generalises better because overfitting has been significantly reduced. This behaviour is shown on **Figure 8.1**.

A confusion matrix is also used as an evaluation metric for this model. The confusion matrix for the initial model is shown to the left on **Confusion Matrix 8.1**, where the predictions are evenly distributed diagonally which implies correct predictions. It can also be seen that wrong predictions often occur when having to determine whether an abnormality is benign or malignant.

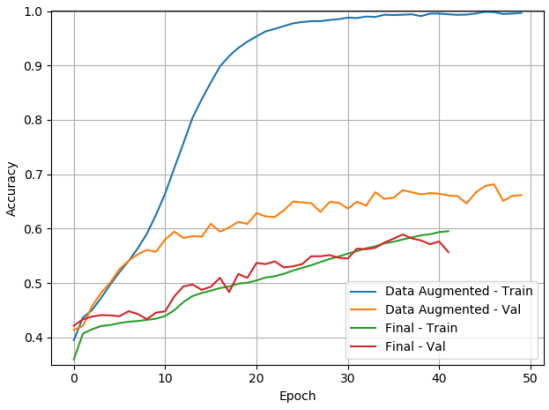


Figure 8.1: Comparing the accuracy for the FD model of its first test on the augmented dataset and its final version.

The rightmost confusion matrix is for the final FD model. As seen the amount of correct predictions for each diagnosis have decreased. Malignant calcification has most significantly decreased compared to the other diagnoses. The issue of having to determine an abnormality to be either malignant or benign also proves difficult. The model often correctly predicts if the input contains either calcification or mass, but struggles to determine whether or not it is benign or malignant. An example can be seen in the confusion matrix that the model often guesses malignant calcification to be benign calcification.

	N	BC	BM	MC	MM	*Total*
N	**579**	**69**	**39**	**45**	**10**	*742*
BC	**71**	**566**	**74**	**107**	**36**	*854*
BM	**27**	**76**	**535**	**50**	**97**	*785*
MC	**57**	**143**	**70**	**258**	**43**	*571*
MM	**15**	**37**	**129**	**42**	**465**	*688*
Total	*749*	*891*	*847*	*502*	*651*	*3640*

	N	BC	BM	MC	MM	*Total*
N	**578**	**126**	**27**	**6**	**5**	*742*
BC	**211**	**361**	**189**	**27**	**66**	*854*
BM	**136**	**184**	**245**	**11**	**209**	*785*
MC	**104**	**233**	**132**	**44**	**58**	*571*
MM	**57**	**100**	**218**	**27**	**286**	*688*
Total	*1086*	*1004*	*811*	*115*	*624*	*3640*

Confusion Matrix 8.1: Confusion matrices of the initial (left) and final (right) FD model.

8.2 Divide & Conquer Model

The most notable improvement for the NP model was in regards to its validation accuracy, where it increased from 89.98% to 94.24% at its peak, as seen on **Figure 8.2**. The recall and precision of the model remains largely unchanged. This can be contributed to the shift in training accuracy from the initial to the final model. When the model is overfitted, the output distribution will contain extreme values, resulting in an increasing recall and precision.

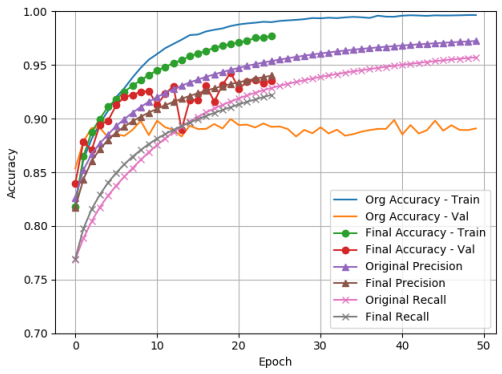


Figure 8.2: Comparing the designated metrics for the NP model of its first test on the augmented dataset and its final version.

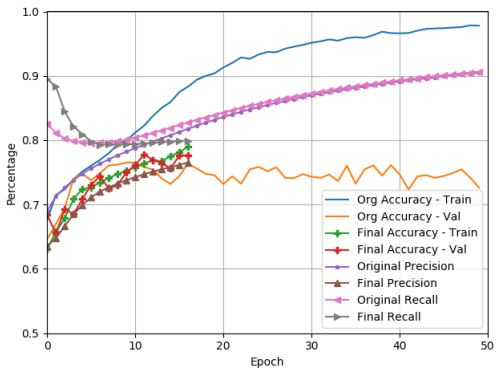


Figure 8.3: Metrics of the first iteration and final version of the MM model trained on the augmented dataset.

The graph depicted on **Figure 8.3** shows the initial iteration of the MM model plotted together with the final version of the model. The initial model reaches 97.8% training accuracy and 76.5% validation accuracy while the final model achieves 78.9% training accuracy and 77.8% validation accuracy. In addition to the increase in validation accuracy, the overfitting is also significantly reduced. Overall, these results are very desirable as a generalised model is important for the problem domain.

8.3 Architecture Comparison

This section aims to compare the two architectures against each other. The interpretations of the output of the two architectures is shown in **Figure 8.4**. The figure shows how an input image is classified using the two architectures and how the outputs are presented.

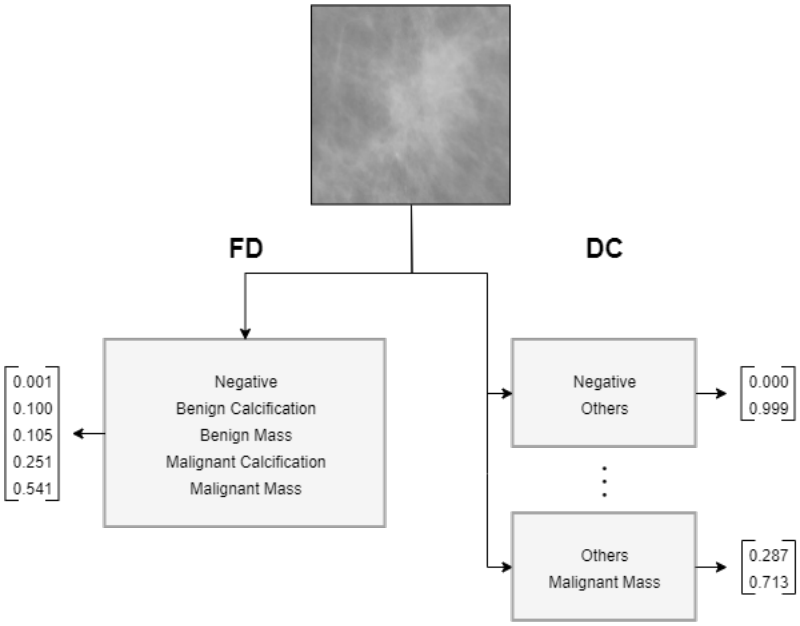


Figure 8.4: Comparison of the architectures' prediction of a MM labelled image.

The final accuracy of the models is not a favourable indicator for the comparison of each architecture, as a reasonable comparison with the accuracy can only be accomplished with the FD's accuracy to an average of all five DC models. However, class-wise accuracies for the FD model can be used against the accuracies of the two DC models. It is possible to calculate the class-wise accuracy for the diagnoses using **Confusion Matrix 8.1**.

The negative class-wise accuracy for the final FD model is 77.9%, while the malignant mass accuracy for the FD model is 41.6%. When comparing the class-wise accuracies from the FD model with the NP model's accuracy at 94.3 % and the MM model's accuracy at 77.8%, it is clear that the DC architecture performs better than the FD architecture in terms of accuracy. The fact that the FD model needs to capture more complex features in the dataset at once while the DC architecture only needs to focus at one set of features at a time could be possible reason for this outcome.

9 Implementation

In this chapter relevant parts of the implementation of the CNN model is highlighted. This includes how the data pipeline and data pre-processing is handled and how the training is performed.

9.1 Dataset sorting

The implementation of the sorting and processing of the dataset will be described. The flow of the processing of the dataset for positive diagnosis types is shown in **Figure 9.1**. It is not necessary to process the dataset in the same manner for the negatives as they are dominant in the original dataset.

Figure 9.1 shows that the original dataset is categorised into a dataset of a particular positive diagnosis and a dataset of other diagnosis types. The dataset with other diagnosis types will be created after performing data augmentation on the positive diagnosis dataset. The unique positive diagnosis is then rotated by 90° three times, effectively quadrupling the dataset. Afterwards, the particular positive diagnoses dataset is flipped from left to right, doubling the dataset. Afterwards, a similar amount of mammograms with other diagnoses is found in the original dataset. Then 5% of both datasets is saved as a test tfrecord, and another 5 % as validation tfrecord. The remaining part of the datasets is saved as the training dataset. The datasets are saved as tfrecords to make use of tensorflow's data pipeline, which is optimised to effectively use the gpu. To use the tfrecord format, the data must be converted to a specific format. This conversion is shown in **Code Snippet 9.2**.

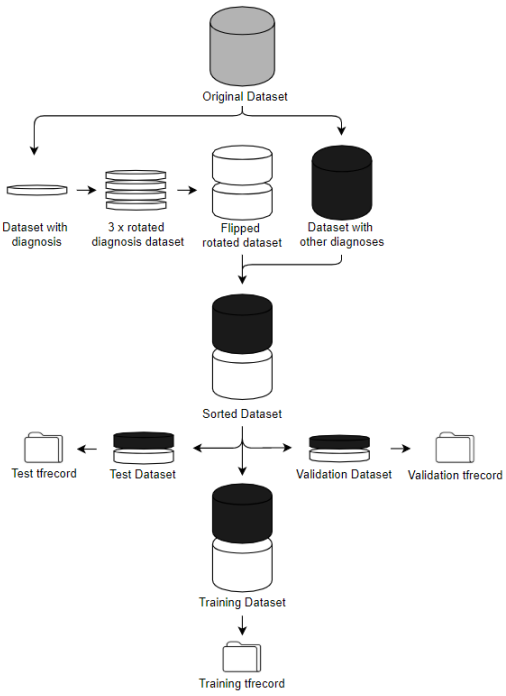


Figure 9.1: Illustrates the processing of the dataset, where the white cylinders represent a positive diagnosis in the original dataset, while the black ones are an even distribution of the other diagnoses.

```
def serialize_example(image, label):
    feature = {
        'image': _bytes_feature(image),
        'label': _bytes_feature(label),
    }
    example_proto = tf.train.Example(features=tf.train.Features(feature=feature))
    return example_proto.SerializeToString()
```

Code Snippet 9.2: Encodes each feature of the dataset to a byte representation in preparation for file writing.

Code Snippet 9.2 depicts the encoding of each sample in the dataset when saving it to a tfrecord. To save the dataset on tfrecord form it is necessary to define the features in the dataset. Features in this case means a description of the elements' encoding of the dataset. Each image and label is converted into a string through a Numpy[1] *tostring* function and then to bytes. After the conversion to bytes, the data is written to tfrecord files. When reading the tfrecord files before training, a different feature description is necessary as the file reader reads the input of the type string. Therefore, each pair of image and label is read as strings. After reading each sample in the dataset, it is processed with the code in **Code Snippet 9.3**.

```
def decode_indicator_variables(serialized_example):
    feature = tf.io.parse_single_example(serialized_example, features_description)
    image = tf.io.decode_raw(feature['image'], tf.uint8)
    label = tf.io.decode_raw(feature['label'], tf.int64)

    image = tf.reshape(image, [299, 299, 1])
    image = tf.cast(image, tf.float32)
    image /= 255.0
    return image, label
```

Code Snippet 9.3: Each sample is decoded from a tfrecord format to an image of correct shape and label.

Code Snippet 9.3 shows that each sample is processed by first decoding each sample into image and label. Then they are reshaped to their appropriate spatial structures. As the last part of the processing the pixel values in the image is normalised from the grey scale values to the interval of 0 and 1. The reasoning for normalising the grey scale values in the image is the cost calculated during training often exceeds the maximal value for float.

9.1.1 Indicator Variables

In **Section 7.2.1**, it was determined that an indicator variable representation of the dataset's labels is necessary. **Figure 9.4** illustrates how the indicator variable representation is applied. Each label is now an array converted to a tensor and appended to a tensor array made up of all positive diagnoses to perform data augmentation on.

[1]Numpy is a Python library for performing matrix operations.

```
    for image, old_label in dataset:
        if old_label == 1: # Benign Calcification
            indicator_variable = tf.convert_to_tensor([0, 1, 0, 0, 0], dtype=tf.int64)
            positive_images.append(image)
            positive_labels.append(indicator_variable)
        # . . .
```

Code Snippet 9.4: Converting a sparse representation of a label into an indicator variable for the FD dataset. The DC dataset will only have a tensor of length 2 as indicator variable. [0, 1] or [1, 0].

9.2 Model Implementation

The models are implemented using the Keras library (API). Keras allows for fast experimentation with CNN structures and offers a wide variety of appropriate variables to interact with the different layers. The type of layers includes convolutional layers and fully-connected layers as well as operations like pooling and dropout as layers.[52]

```
Conv2D(amount_of_feature_maps,
      (n, n),                    # Filter size of n * n * 1 (depth is init. to 1)
      strides=(s1, s2),          # Stride of s1, s2
      padding='same',            # Half-padding
      activation='relu',         # Activation function
      use_bias=True,             # Enable bias
      kernel_initializer=keras.RandomNormal(mean=0.0,
                                            stdev=0.05,
                                            seed=None), # Normal distribution
       bias_initializer='zeros', # Initialise the bias weights to zero
       kernel_regularizer=None,  # Penalty regularisation (E.g. l1 or l2)
       )
```

Code Snippet 9.5: The implementation of a convolution layer.

On **Figure 9.5**, a possible configuration of a convolutional layer is shown. Padding is specified with a string denoting the type of padding. In our case, we have used the '*same*' padding which is the Keras implementation of half-padding. The *kernel_initializer* signifies the initialisation of the weights in the layer. In this case, a random normal distribution of the weights is used. The *kernel_regularizer* defines regularisation methods, which are not applied to the initial model.

9.3 Training

The *compile* method is another part of the Keras library. It allows the specified *MODEL* to train by giving it certain parameters. The model in this example on **Code Snippet 9.6** is a DC model. This means the *optimizer* parameter (GD) is set to be Adam. This choice was concluded from the results of **H.4**. The *loss*/cost method is categorical cross entropy. The final parameter specified is *metrics*. This

parameter receives a tuple of three different metrics to be calculated: *Categorical accuracy*, *precision*, and *recall*.

```
1 MODEL.compile(optimizer = tf.keras.optimizers.Adam(),
2               loss = tf.keras.losses.CategoricalCrossentropy(),
3               metrics = [tf.metrics.CategoricalAccuracy(),
4                          keras_metrics.precision(),
5                          keras_metrics.recall(),
6                          ])
```

Code Snippet 9.6: Code snippet of the compile method for the DC models. The FD will use SGD as GD and other metrics.

The *fit* method specified on **Code Snippet 9.7** configures the number of epochs the models are trained on. The amount of steps per epoch is defined by the size of the training data divided by the batch size. The batch size is set to be 32 as a result of **H.3**. It also specifies the dataset to train and validate on line 1 and 2. *shuffle* is a boolean parameter to decide whether to shuffle the training data or not before each epoch. *callbacks* on line 7 is a list of callbacks to be applied during training and validation after each epoch of the training procedure. Callbacks gives a view of the internal states and statistics of the model during training. It is from these callbacks we are able to implement early stopping and create the graphs of the results in previous chapter.

```
1 MODEL.fit(batched_training_data,                          # Training data
2           validation_data = batched_val_data,            # Validation data
3           steps_per_epoch = FILE_SIZE // BATCH_SIZE,
4           validation_steps = TEST_SIZE // BATCH_SIZE,
5           epochs = EPOCHS,                               # Amount of epochs
6           shuffle = True,                                # Shuffles the data
7           callbacks = [callback_1, ... callback_n])      # Save training data
```

Code Snippet 9.7: Code snippet of the fit method.

9.4 Resizing of New Mammograms

The models are trained on images of size 299x299 it is necessary to process any new mammograms for prediction. This can be seen on **Code Snippet 9.8**. The process is to crop the full sized mammogram into 299x299 segments so it fits the data the model is familiar with. One issue occurs if the full image's width and height is not a divisor of 299. To solve this, we add black padding to the image so the size can be divided with 299. After this it is possible to crop it into uniform sized segments.

```
def resize_image_padding(image_path):
    image_in = cv2.imread(image_path)
    size = image_in.shape[:2]                                  # Gets the size of the
                                                               # input mammogram picture
    divisor = math.ceil(299/size[0])

    max_dim = 299 * divisor
    image_w = max_dim - size[1]
    image_h = max_dim - size[0]
    top,bottom = image_h // 2, image_h - (image_h // 2) # Padding for top, bottom
    left,right = image_w // 2, image_w - (image_w // 2) # Padding for left, right
    color = [0, 0, 0]                                          # Sets padding to black
    image_out = cv2.copyMakeBorder(image_in, top,              # Creates the image with
        bottom, left, right, cv2.BORDER_CONSTANT,              # needed amount of padding
        value = color)
    image_name = os.path.split(image_path)[1].split('.')[0]
    crop_image(image_out, image_name)                          # Crops image into
    return image_name, image_out                               # 299x299 sizes
```

Code Snippet 9.8: Code showing the process of cropping a full image of arbitrary size to segments of size 299x299.

10 | Discussion

*The discussion chapter will first deal with an evaluation of the requirements for the project stated in **Chapter 4**. The discussion will cover some of the reflections made during the project. It will also include topics of what we believe is worth investigating further as future work.*

10.1 Evaluation of Requirements

Table 10.1 shows the original requirements with an associated evaluation column (fulfilled (F), partially fulfilled (PF), or not fulfilled (NF)). Below the table the reason for the fulfilment of each requirement will follow.

Index	MoSCoW	Requirements	F	PF	NF
R.1	M	The model must be able to classify the different diagnoses in the dataset.	✓		
R.2	M	The model must take a grey scale mammogram as input.	✓		
R.3	M	The prediction made by the classifier must create a probability distribution of the diagnoses.	✓		
R.4	M	The solution must include a neural network specialised for pictures (CNN).	✓		
R.5	M	The model must be evaluated based on relevant metrics.	✓		
R.6	S	The input image should be processed to match the format of the dataset examples.	✓		
R.7	S	The program should include an interface.		✓	
R.8	C	Based on the prediction of all mammograms, the program would create a prioritised list of patients.			X
R.9	W	The classifier will be embedded in an existing system at a hospital practising mammography.			X

Table 10.1: Table of requirements with index number, MoSCoW classification, requirements specification, and degree of fulfilment.

R.1 This has been achieved by designing two architecture types. For the FD architecture, a single model has been trained. The DC architecture contains five different models, but only two of them are trained as proof of concept. These three models have thereafter been modified through all the hypotheses in **Chapter 7**.

R.2 As described in **Section 6.2** the first convolutional layer of the models take images of 1 depth as input. This means that grey-scale images are supported by the models.

R.3 The three different models use the activation function softmax to the output layer. As described in Section 5.1.1, softmax provides a probability distribution, which fulfils the R.3.

R.4 For the purpose of classifying breast cancer in mammograms, a convolutional neural network was used, because it is specialised for images. Using a convolutional neural network fulfils the R.4.

R.5 Relevant metrics are found and used in **Chapter 7**. For the FD model, accuracy and a confusion matrix are used to evaluate the model. For the DC models, accuracy, precision and recall are used. These metrics are used to analyse and evaluate the models through the testing of different hypothesis.

R.6 The model is optimised to take the format of images in the dataset examples as input. The full mammograms are taken as input and are then resized to fit the segment size of 299x299 as described in **Section 9.4**.

R.7 We do manage to create a simple Graphical User Interface (GUI) that allows the user to select both the kind of model (FD or DC) and what picture to predict on. Despite of this, the requirement is still deemed as *partially fulfilled* as the GUI still is a proof of concept. As we are unaware of the desired features to implement we are uncertain of how to evaluate the GUI.

R.8 A proof of concept of the priority list is presented in future works to illustrate a possible implementation. However, as it would require domain specific information, it is not possible to realise this in the project.

R.9 Our solution has not been embedded into the hospital system. Therefore, R.9 is evaluated to not fulfilled. This however is within our expectations and has not been considered within the scope of this project as we did not have access to hospital grade hardware.

10.2 Considerations

During this project, certain considerations surfaced which will be elaborated in the following sections. Systematic testing and data augmentation are contemplated and described here.

10.2.1 Test Variance

When evaluating the hypothesis throughout **Chapter 7**, the different tests were only completed once. Many hyperparameters in the CNN are influenced by randomness such as weight initialisation, shuffling of the dataset, and GD. This means the test results could vary if the model is trained multiple times, thus giving a different outcome for the hypotheses.

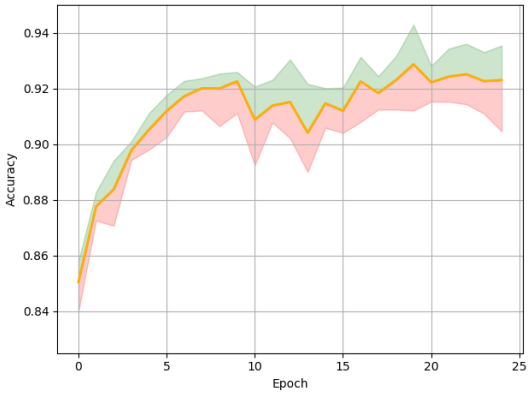


Figure 10.1: Shows the maximum (green), average (orange), and minimum (red) margins of test results of the the final NP model. The average is calculated from the same test being conducted 5 times.

To make the test results more reliable, multiple iterations of each test could have been completed. The average of these results would then be used in the comparison to determine the correct strategy. An example of this is shown on **Figure 10.1**. Here, the NP model's final test results can be seen, which is conducted 5 times. If the minimum graph (red) are the initial results in the tests regarding the amount of fully-connected layers, 2 fully-connected layers would have been chosen over 1.

An important observation regarding this test is that we did not apply early stopping, as we wanted each test to run for the same amount of epochs in the example. But there is also a possibility that some tests could stop earlier than others and thereby give different results.

Due to time constraints we did not do this and can not deduce with certainty how and if it would have affected all test results. If the above described test procedure were performed in every test, the final model might vary from the current one.

10.2.2 Systematic Testing

Through all the tests of the different hypothesis in **Chapter 7**, we strove to achieve the best possible model. In the latter part of the chapter, the model's structure was tuned through the amount of different layers, filter size, and neurons. Most of these tests showed little to no improvement from what we initially had. A probable reason for this could be that the models were already optimised with the constant parameters found in the initial model, hence changing these now would not yield any significant improvement. Therefore, there is the possibility that tuning the model before doing the optimisations could lead to different results.

10.2.3 Data Augmentation Problematics

As we investigated in **Section 7.2.2** some data augmentation methods were not applicable due to the nature of the data. E.g. we were not able to rotate the pictures in other ways than 90, 180, and 270 degrees, as we would be running a risk to loose important information in the image near the edges. A solution to this could have been to admit a bigger uniform size for all the images. While still keeping the spacial properties of the original 299x299 sized images, we could increase the size of each image to 423x423 by adding black padding around it.

This technique would make it possible to rotate the image 360° without losing important information near the edges, and would enable entirely new data augmentation methods. The black padding could be utilised to move the existing image around in numerous ways. In theory, this technique would also help the model to learn how to classify independently from the location of the ROIs in the image.

It could also be an assistance in making the models more susceptible to new images to predict. As each picture the model has trained on is 299x299, it means that we have to cut new mammograms into these sizes. Our implemented solution to this is to make sure the mammogram's width and height scales with 299. If it is not the case we pad the entire image with black bars to make it scale correctly. Afterwards we cut it up into 299x299 sized images. The problem with this method is that it adds a new type of information to the pictures that the model might not be susceptible to, and thereby making it affect the prediction, which is not the intended outcome. By having the model train on pictures with the black padding we could prevent this issue.

Unfortunately, we got acquainted with this technique late in the project, after multiple hypothesis tests already had been done, which lead us to the conclusion that the payoff vs. the workload of redoing the data augmentation with the new methods and the tests, was too small.

10.3 Future Works

This section is dedicated to features that were not implemented in the project due to time constraints or lack of accessibility.

10.3.1 Embedded System

The proposed solution is supposed to fit the medical equipment related to mammograms as defined in the requirement **R.9**. This was not fulfilled due to the medical equipment being costly to use, therefore, the project has not been tested in on the real system at the hospital. Testing the program on their current system could provide greater insight to detection of breast cancer and possible restrictions that the program should fit. Having a medical professional involved in the project could also yield insight to the diagnosis process, which could result in further optimisations to the models and/or improvements to the project as a whole.

This topic also attaches to the ethical question stated back in **Section 2.3** about the public opinion having a worrying attitude against CAD-like systems. In order to change the public opinion to be more susceptible to the idea of having AI, more specifically our CNN models, assisting medical specialists in making *life-or-death* decisions the medical specialists them self must firstly be able to trust the output. The way things are at the moment, you only get a numerical output from the models. It is unlikely that medical specialists will let this simple output affect the diagnosis of their patients. That requires the models to produce the reasoning behind the output. A possible approach to create reasoning is object detection.

10.3.2 Object Detection

Object detection is the technique for recognising and labelling different segments in an image. This could be used to highlight the ROIs in mammograms, making it easier for the doctors to locate abnormalities. As seen on **Figure 10.2**, an illustration of how object detection could be used to highlight abnormalities in the output.

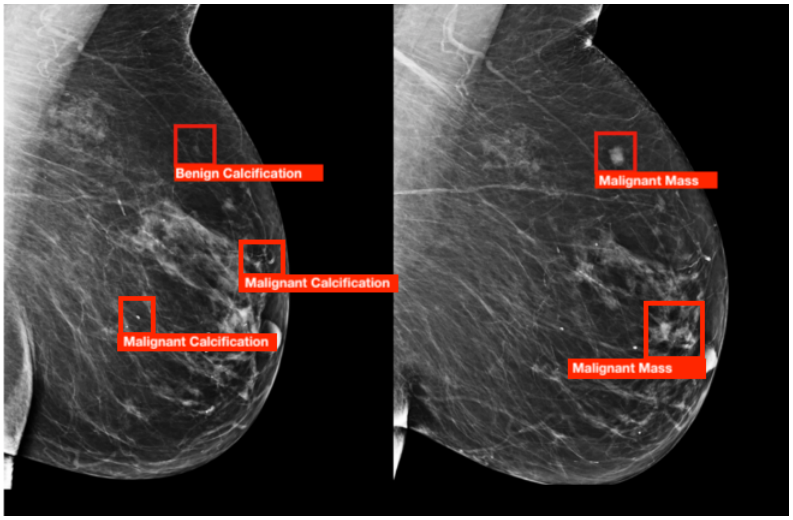


Figure 10.2: Object detection to highlight both malignant and benign calcifications and masses[53]

The current resizing functionality along with the CNN classifier is a form of classification that highlights different segments to show which of them contain abnormalities. But this technique does not support the case of having multiple different abnormalities within a segment, because the highest diagnosis probability is chosen. After having a positive diagnosed segment, the segment will be highlighted. An example of how this works is given in **Appendix D.1**. Here, an entire mammogram is given as input and each segment is given a classification. As output, the image is reconstructed. Each segment of this image is highlighted, based on the classifier's predictions. On the other hand, object detection would be able to find the specific location of the abnormalities and is able to locate multiple abnormalities within a segment.

10.3.3 Full Mammograms

Another additional extension to the program would be to analyse a full mammogram rather than segments of one. The dataset divides the images into segments to create more data to test on. Unfortunately, this creates a possibility of having an abnormality at the boarders of the segments, which would then divide the abnormality over segments. This could potentially be classified as not having a positive diagnosis, because a partial abnormality might not be generalised as a positive diagnosis. Analysing full mammograms would also make other data augmentation methods such as zoom possible.

10.3.4 Prioritised List of Mammograms

Once an outcome is produced it would be possible to prioritise which patients are the most urgent to treat. We propose a proof of concept solution inspired by the lottery concept from the expected utility theory. While we do not have any insight for the exact precedence of each diagnosis, we can conceptually imagine an implementation of the prioritised list. Predictions from the classifier can be interpreted as a probability distribution over the diagnoses. It is then possible to calculate a score based on the probabilities from the aforementioned probability distribution, weights found based on medical experts, and a bias to encode other considerations. This can be derived as the following equation:

$$score = b + \sum_{i \in I} \sum_{j}^{n} p_j w_j$$

In the equation above, the bias could encapsulate the period of time in which a patient has been on the list. Each day a patient occurs on the list the bias should be increased ensuring that a false negative is handled properly. The equation sums over the value of each possible segment's prediction of an input image ($i \in I$) multiplied by the precedence weights of the diagnoses (w_j) and adds the bias. Using the equation, scores for each mammogram could be calculated and be used as a measure for prioritising the mammograms. It is important to emphasise that all images should be reviewed by a medical specialist.

11 | Conclusion

On the topic of detecting breast cancer with the use of machine intelligence, we have examined the screening process. Each step in the process is evaluated and the mammography step is chosen. Regarding this the rate of false-positive and false-negative diagnoses are determined to be significant for the domain. To detect breast cancer in these mammograms, different existing state of the art algorithms are analysed and the classification algorithms are deemed most suitable. More specifically NNs including CNNs, are best suited for the CBIS-DDSM dataset.
This led us to the following problem statement:

How can recent advances in machine learning be exploited to reduce the workload of specialised personnel to analyse mammograms without increasing the risk of false positives and false negatives in breast cancer diagnosis?

In the process of investigating NNs in further detail, a set of requirements is established to ensure a sufficient solution for the problem statement. For the internal components of NN and CNNs we have looked at different strategies and evaluated them against each other to create the structure of the initial models to be used for training on the dataset. Two architectures are introduced, the FD architecture to encapsulate predictions of all five classes in the dataset and the DC architecture following the *one-vs-all* method. Of the two architectures, three models were created.

Regarding the dataset we have encountered unsatisfactory results, as the distribution of the labelled training examples were very unbalanced. To act against this, we exploit multiple data augmentation methods to create a more even distribution and more training examples in the datasets, which proved to increase the performance.

During testing of the three models, we establish a number of hypotheses to test. The different strategies were investigated theoretically to determine which would give the best results. We define different metrics for each architecture to evaluate the results. A full list of the hypotheses can be seen in **Appendix A**. When evaluating the model architectures, it is concluded that the DC architecture performs favourably compared to the FD architecture. However the concern of receiving contradictory results in the DC architecture exists.

In the discussion the majority of the requirements is deemed fulfilled. The models have provided an educated prediction of diagnoses on mammograms, thereby assisting radiologist in identifying breast cancer.

Bibliography

[1] Tensorflow. *Tensorflow*. 2019. URL: https://www.tensorflow.org.

[2] S. A. McLeod. *What is a hypothesis?* Aug. 2018. URL: https://www.simplypsychology.org/what-is-a-hypotheses.html.

[3] American Institude for Cancer Research. *Worldwide cancer data*. URL: https://www.wcrf.org/dietandcancer/cancer-trends/worldwide-cancer-data.

[4] Kræftens Bekæmpelse. *De hyppigste kræftformer*. URL: https://www.cancer.dk/hjaelp-viden/fakta-om-kraeft/kraeft-i-tal/de-hyppigste-kraeftformer/.

[5] My Virtual Medical Centre. *Early Detection of Breast Cancer*. URL: https://www.myvmc.com/investigations/early-detection-of-breast-cancer/.

[6] National Breast Cancer Foundation Inc. *Early Detection*. URL: https://www.nationalbreastcancer.org/early-detection-of-breast-cancer/.

[7] Kræftens Bekæmpelse. *Sådan foregår screening for brystkræft*. URL: https://www.cancer.dk/forebyg/screening/brystkraeft/saadan-foregaar-undersogelsen/.

[8] Kræftens Bekæmpelse. *Mammografiscreening*. URL: https://www.cancer.dk/hjaelp-viden/undersoegelser-for-kraeft/scanninger-billedundersoegelser/mammografi/.

[9] Kræftens Bekæmpelse. *Hvad er kræft?* URL: https://www.cancer.dk/hjaelp-viden/fakta-om-kraeft/hvad-er-kraeft/.

[10] NationalBreastCancer. *Breast Tumors*. 2019. URL: https://www.nationalbreastcancer.org/breast-tumors.

[11] sundhed.dk. *Brystkræft*. URL: https://www.sundhed.dk/sundhedsfaglig/laegehaandbogen/brystsygdomme/tilstande-og-sygdomme/kvinder/brystkraeft/.

[12] WebMD. *Breast Calcifications*. 2018. URL: https://www.webmd.com/women/guide/breast-calcification-symptoms-causes-treatments.

[13] Kræftens Bekæmpelse. *Undersøgelser for brystkræft*. URL: https://www.cancer.dk/brystkraeft-mammacancer/undersogelser-brystkraeft/.

[14] Sundhedsstyrelsen. *Tilbud om screening for brystkræft*. URL: https://www.sst.dk/da/sygdom-og-behandling/screening/~/media/9F07C6DAC2D0434393D1A573DE6A7E3C.ashx.

[15] DoctorOZ. *Digital Mammograms Improve How Doctors Find Breast Cancer*. URL: https://www.doctoroz.com/article/digital-mammograms-improve-how-doctors-find-breast-cancer.

[16] National Institute of Biomedical Imaging and Bioengineering. *Mammography*. URL: https://www.nibib.nih.gov/science-education/science-topics/mammography.

[17] American Cancer Society. *Breast Ultrasound*. URL: https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/breast-ultrasound.html.

[18] RadiologyInfo. *Magnetic Resonance Imaging (MRI) - Breast*. URL: https://www.radiologyinfo.org/en/info.cfm?pg=breastmr.

[19] David L. Poole and Alan K. Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. 2017.

[20] Eric A. Scuccimarra. *tfrecords files of scans from the DDSM dataset*. URL: https://www.kaggle.com/skooch/ddsm-mammography.

[21] Olvi L. Mangasarian Dr. William H. Wolberg W. Nick Street. *Breast Cancer Wisconsin (Diagnostic) Data Set*. URL: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%5C%28Diagnostic%5C%29.

[22] ICIAR. *ICIAR 2018 Grand Challenge on Breast Cancer Histology images*. URL: https://iciar2018-challenge.grand-challenge.org/Dataset/.

[23] Richard Harris. *Training A Computer To Read Mammograms As Well As A Doctor*. URL: https://www.npr.org/sections/health-shots/2019/04/01/707675965/training-a-computer-to-read-mammograms-as-well-as-a-doctor?t=1568148221374&t=1568291762793.

[24] BreastCancer.org. *Computer-Aided Detection May Not Improve Breast Cancer Screening*. URL: https://www.breastcancer.org/research-news/computer-aid-may-not-improve-screening.

[25] American Cancer Society. *Limitations of Mammograms*. URL: https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/mammograms/limitations-of-mammograms.html.

[26] Various authors. *Mammographic Image Analysis Homepage - Algorithms*. URL: http://www.mammoimage.org/algorithms/.

[27] Huai Li, K. J. Ray Liu, and Shih-Chung B. Lo. "Fractal Modeling and Segmentation for the Enhancement of Microcalcifications in Digital Mammograms". In: *IEEE* 16 (1997). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=650875.

[28] Peter Hogg, Judith Kelly, and Claire Mercer. *Digital Mammography - A Holistic Approach*. 2016.

[29] Atam P. Dhawan & Yateen Chitre & Christine Kaiser-Bonasso & Myron Moskowitz. "Analysis of Mammographic Microcalcifications Using Gray-Level Image Structure Features". In: *IEEE Transactions on Medical Imaging* 15 (1996). URL: https://ieeexplore.ieee.org/document/1028488.

[30] Sheila Timp and Nico Karssemeijer. "Interval change analysis to improve computer aided detection in mammography". In: *Medical Image Analysis* 10 (2006). URL: https://www-sciencedirect-com.zorac.aub.aau.dk/science/article/pii/S1361841505000484.

[31] Raheel Shaikh. *Feature Selection Techniques in Machine Learning with Python*. URL: https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e.

[32] Berkman Sahiner et al. "Image feature selection by a genetic algorithm: Application to classification of mass and normal breast tissue". In: (1996). URL: https://aapm-onlinelibrary-wiley-com.zorac.aub.aau.dk/doi/abs/10.1118/1.597829.

[33] Heang-Ping Chan et al. *Computerized classification of malignant and benign microcalcifications on mammograms: texture analysis using an artificial neural network*. URL: https://iopscience.iop.org/article/10.1088/0031-9155/42/3/008/pdf.

[34] Nagdev Amruthnath and Tarun Gupta. *A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance*. URL: https://ieeexplore.ieee.org/abstract/document/8387124.

[35] Agnieszka Wosiak, Agata Zamecznik, and Katarzyna Niewiadomska-Jarosik. *Supervised and unsupervised machine learning for improved identification of intrauterine growth restriction types*. 2016. URL: https://ieeexplore.ieee.org/document/7733260#deqn1.

[36] R.S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MA:MIT Press, 1998.

[37] Francesco Ricci and Lior Rokach and Bracha Shapira and Paul B. Kantor. *Recommender Systems Handbok*. Sprginer, 2010.

[38] Dr. Saed Sayad. *Support Vector Machine - Classification (SVM)*. URL: https://www.saedsayad.com/support_vector_machine.htm.

[39] Adebola Lamidi. *Breast Cancer Classification Using Support Vector Machine (SVM)*. 2018. URL: https://towardsdatascience.com/breast-cancer-classification-using-support-vector-machine-svm-a510907d4878.

[40] David Benyon. *Designing Interactive Systems*. 3. edition. Pearson, 2018, p. 140.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. 2012. URL: http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf.

[42] Stanford CS class. *CS231n: Convolutional Neural Networks for Visual Recognition*. 2019. URL: http://cs231n.github.io/neural-networks-1/#actfun.

[43] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. *Rectifier Nonlinearities Improve Neural Network Acoustic Models*. 2013. URL: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

[44] Michael Nielsen. *Neural Networks and Deep Learning*. 2019. URL: neuralnetworksanddeeplearning.com.

[45] Dominic Masters and Carlo Luschi. *REVISITING SMALL BATCH TRAINING FOR DEEP NEURAL NETWORKS*. 2018. URL: https://arxiv.org/pdf/1804.07612.pdf.

[46] Anup Bhande. *What is underfitting and overfitting in machine learning and how to deal with it.* 2018. URL: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76.

[47] Charu C. Aggarwal. *Neural Networks and Deep Learning - A Textbook*. 2018.

[48] K. V. Anisimov et al. *Proceedings of the Scientific-Practical Conference 'Research and Development'*. 2016. URL: https://books.google.dk/books?id=kNpCDwAAQBAJ&printsec=frontcover&hl=da&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false.

[49] M. KRASKA-MILLER. *NONPARAMETRIC STATISTICS FOR SOCIAL AND BEHAVIORAL SCIENCES*. 2013.

[50] Tensorflow. *Module: tf.keras.metrics*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/metrics.

[51] Nitish Srivastava et al. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. URL: http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf.

[52] Keras. *Keras: The Python Deep Learning library*. URL: https://keras.io/.

[53] Adam Conner-Simons and Rachel Gordon. *Using AI to predict breast cancer and personalize care*. 2019. URL: http://news.mit.edu/2019/using-ai-predict-breast-cancer-and-personalize-care-0507.

Part IV

Appendices

A List of Hypotheses

	Hypotheses
H.1	*An imbalanced dataset will cause the model to overfit.*
H.2	*A larger balanced dataset prevents the model from dominantly predicting negative.*
H.3	*A batch size of 32 will achieve the highest validation accuracy.*
H.4	*Adam is the optimal gradient descent strategy for the two architectures.*
H.5	*Overfitting can be reduced, while maintaining accuracy.*
H.6	*Max-pooling will yield the highest accuracy.*
H.5.1	*Applying early stopping when the val. accuracy stops increasing, reduces overfitting.*
H.5.2	*L2 regularisation reduces overfitting during training.*
H.5.3	*Strategically applied dropout reduces overfitting and increases generalisation.*

B Hypotheses' Test Results

H.1 Imbalanced Dataset

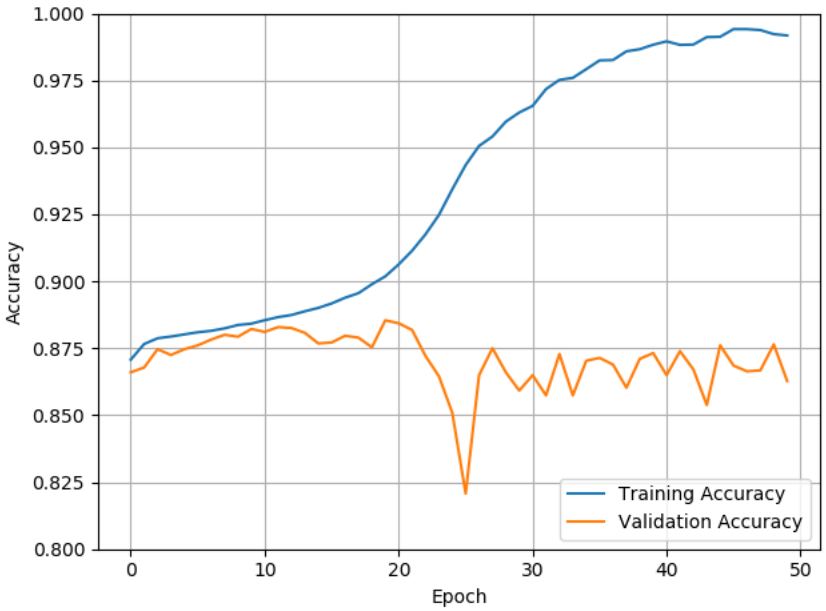


Figure B.1: Original dataset training accuracy through 50 epochs on the FD model.

H.2 Balanced Dataset

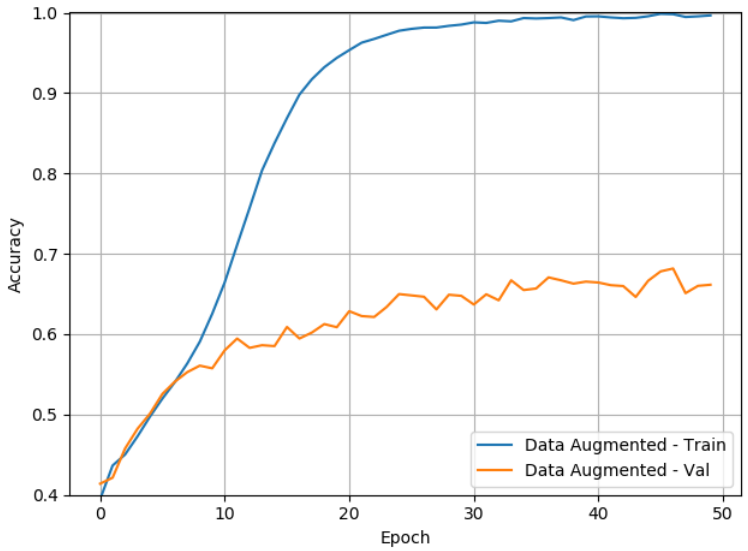


Figure B.2: The data augmented dataset training through 50 epochs on the FD model.

H.3 Batch size

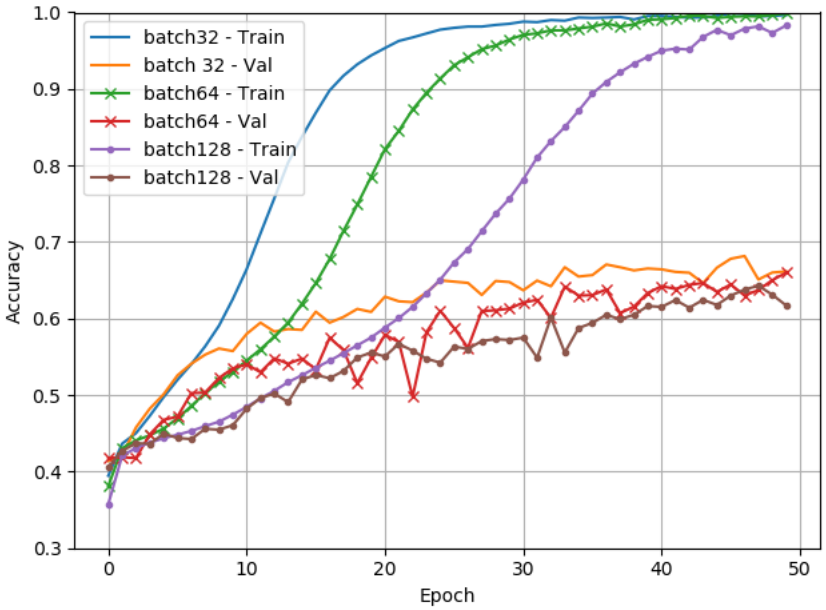


Figure B.3: Results of training the model with batch size 32, 64, and 128 on the FD model.

H.4 Gradient Decent Graphs

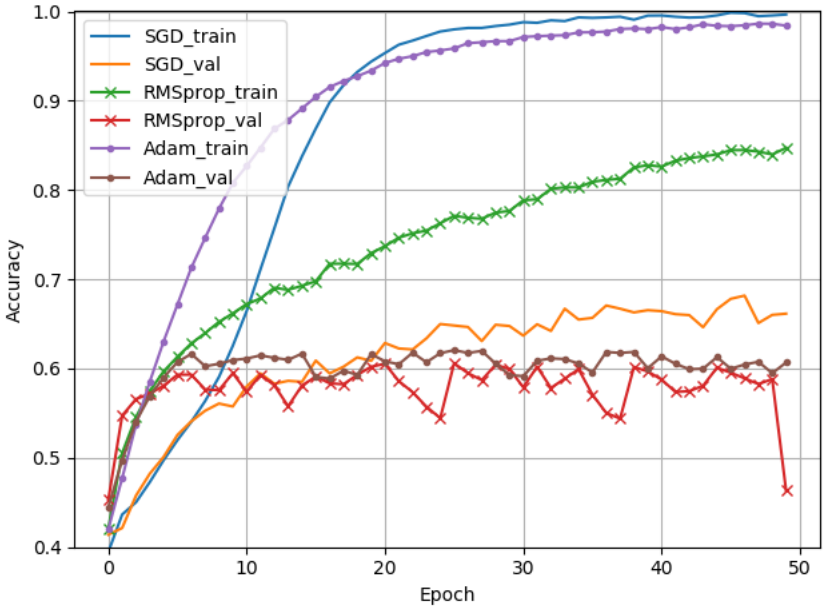


Figure B.4: Gradient descent strategies applied to the FD model.

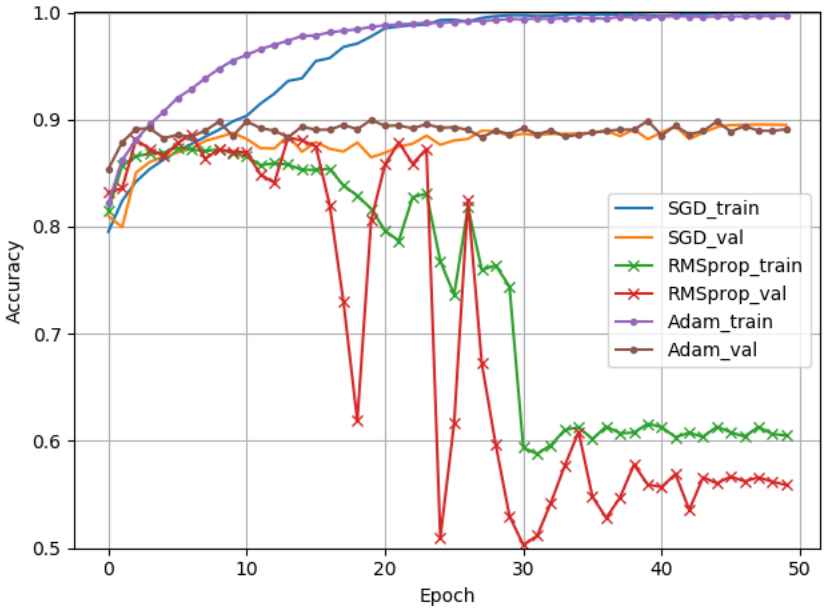


Figure B.5: Gradient descent strategies applied to the NP model

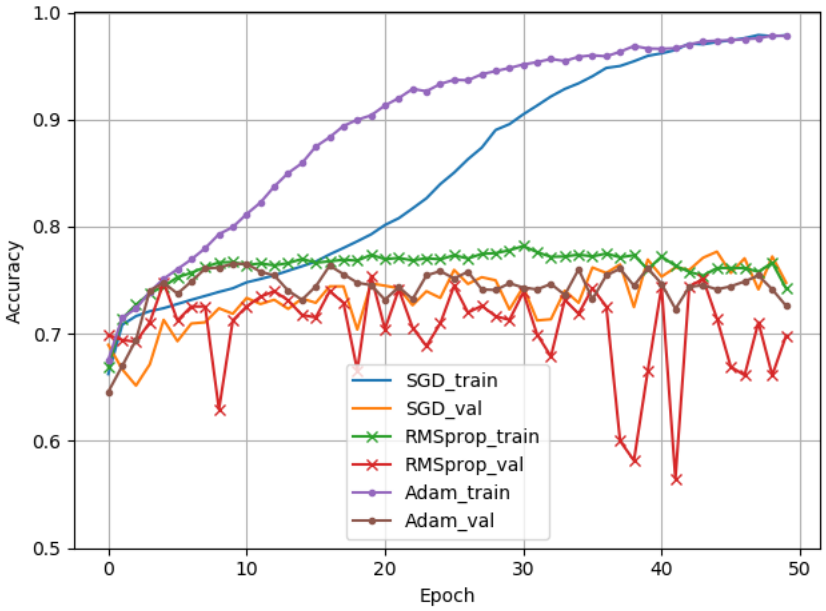


Figure B.6: Gradient descent strategies applied to the MM model

H.6 Pooling Strategy Graphs

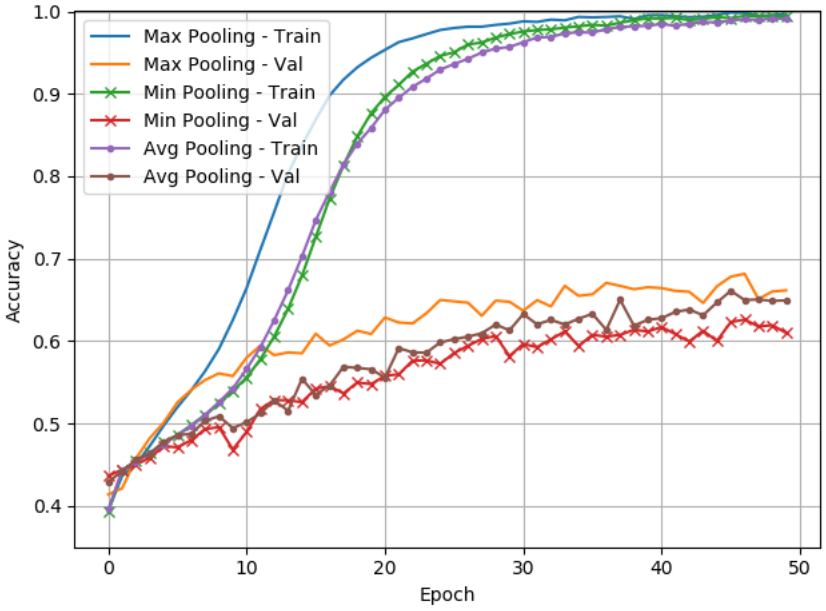


Figure B.7: Pooling strategies applied to the FD model

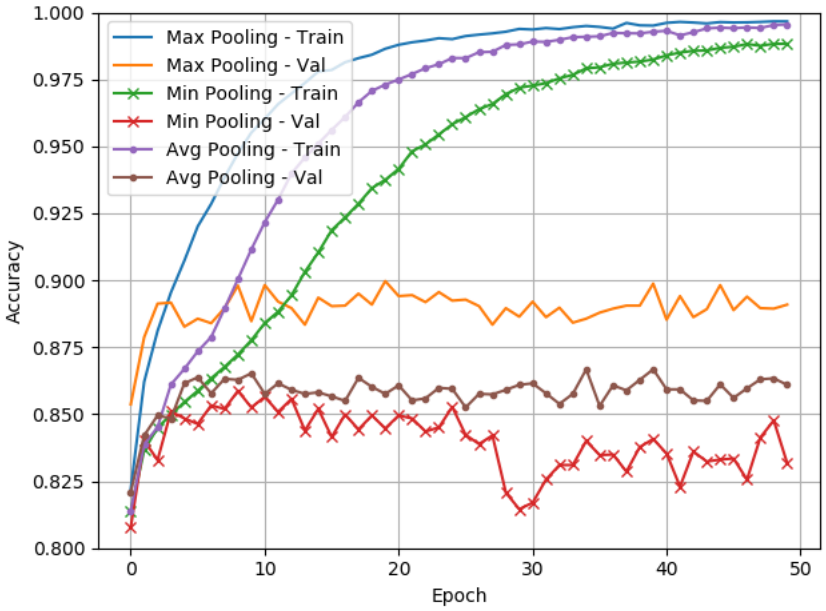


Figure B.8: Pooling strategies applied to the NP model

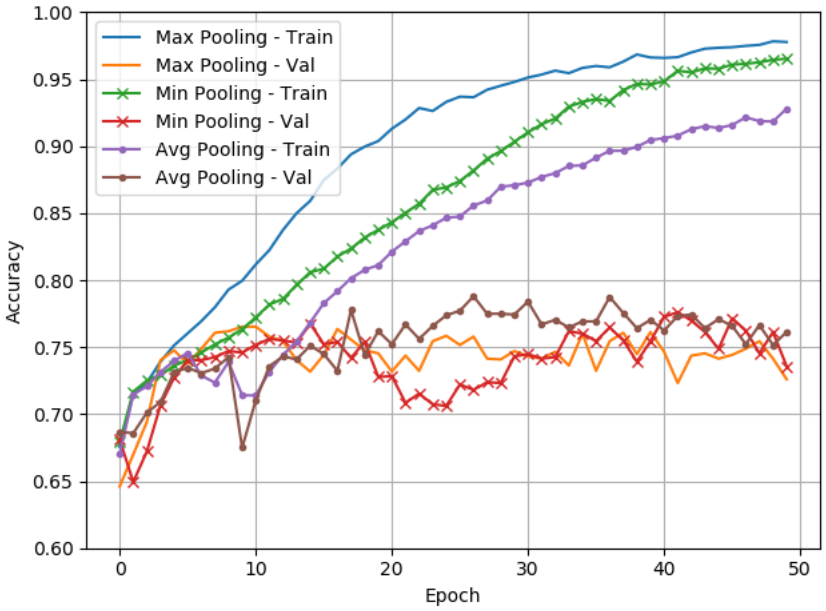


Figure B.9: Pooling strategies applied to the MM model

H.5.2 L2 and L1 Regularisation

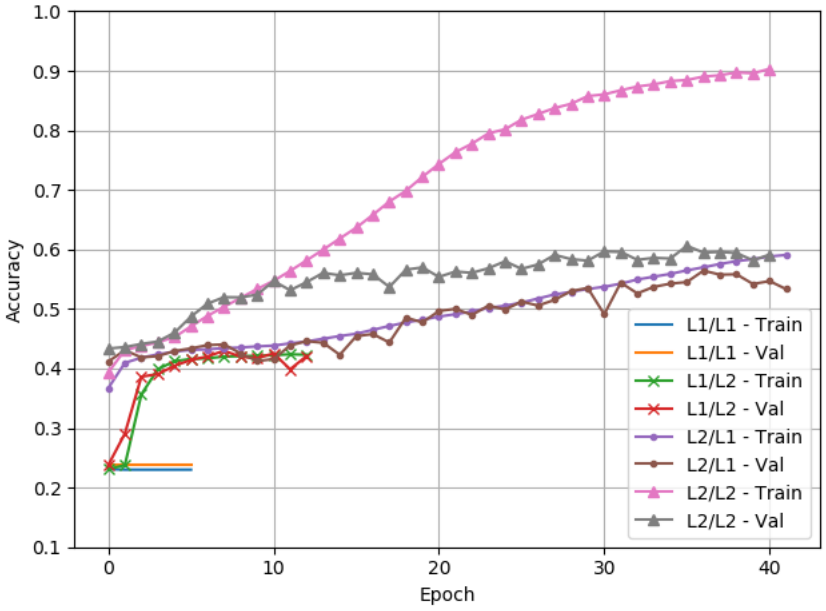


Figure B.10: L1 and L2 regularisation applied to the FD model

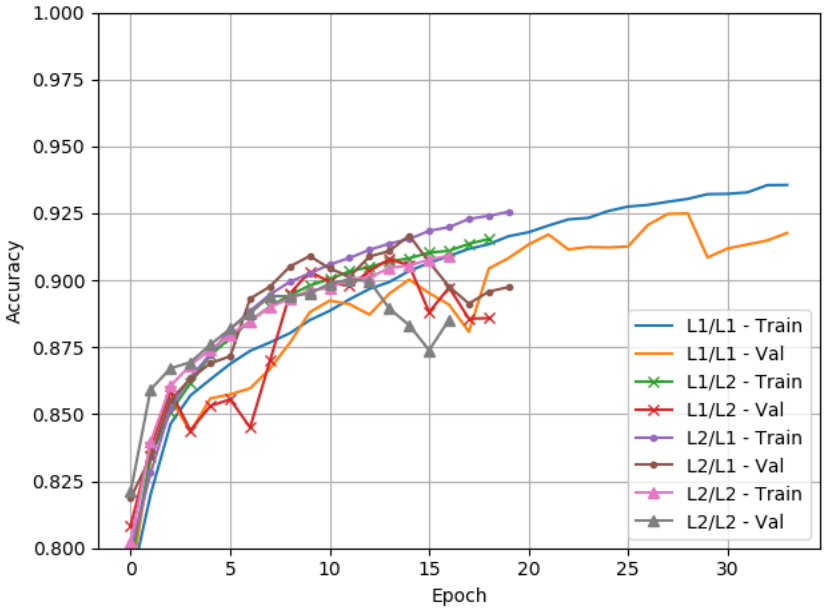


Figure B.11: L1 and L2 regularisation applied to the NP model

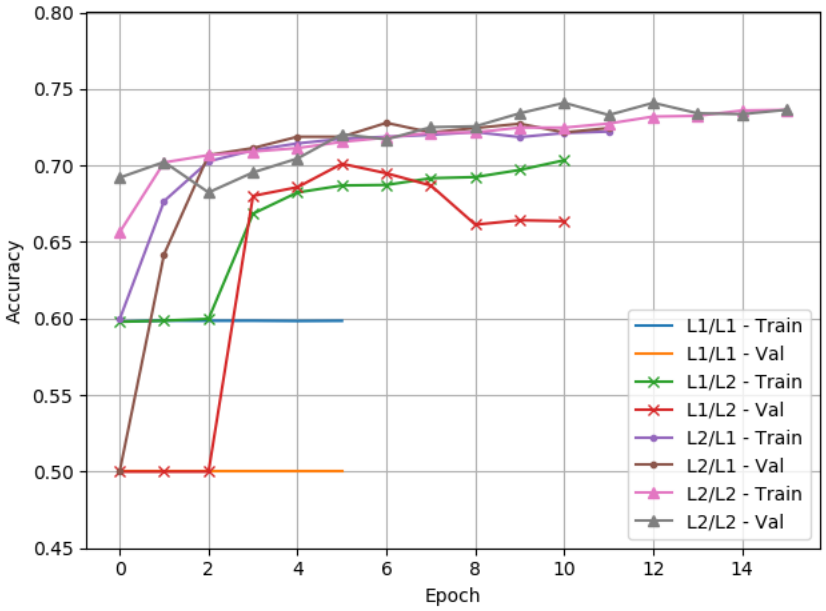


Figure B.12: L1 and L2 regularisation applied to the MM model

H.5.3 Dropout

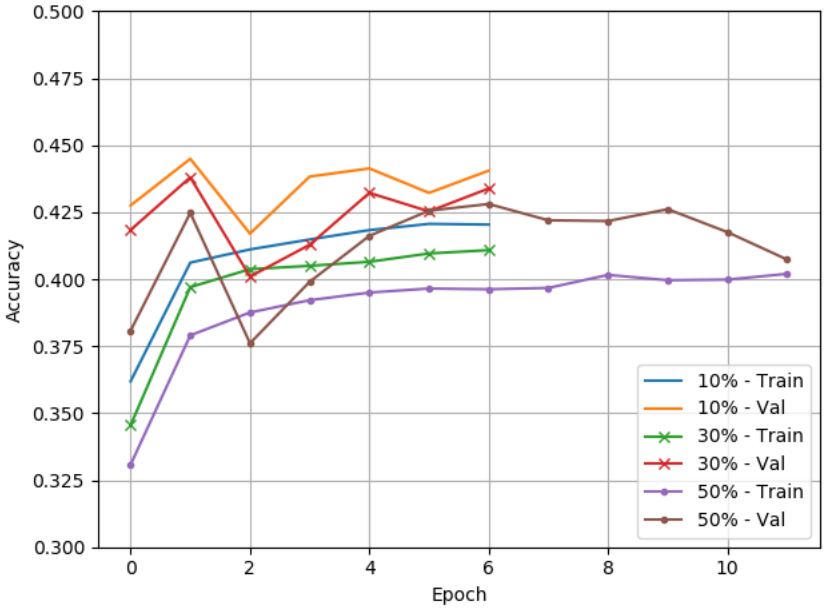


Figure B.13: Dropout percentage on the FD Model with L2/L1.

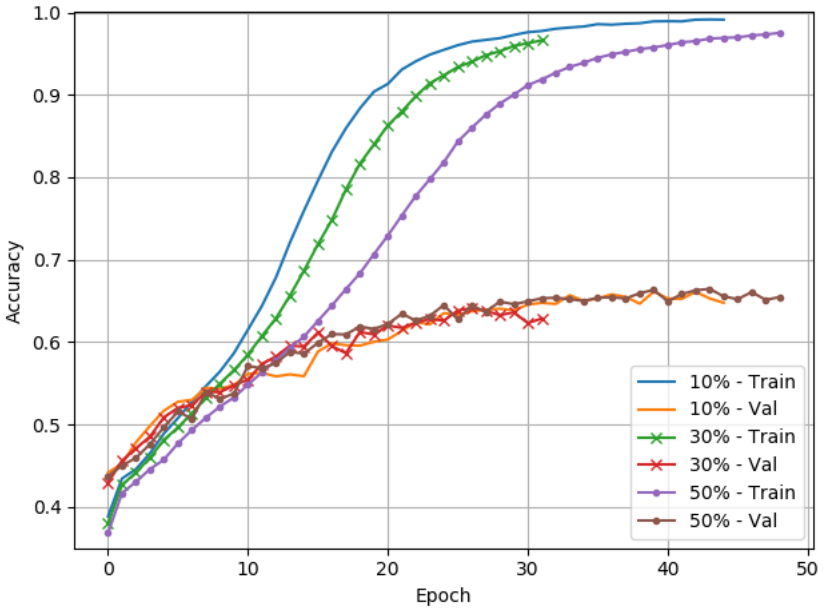


Figure B.14: Dropout percentage on the FD Model without regularisation.

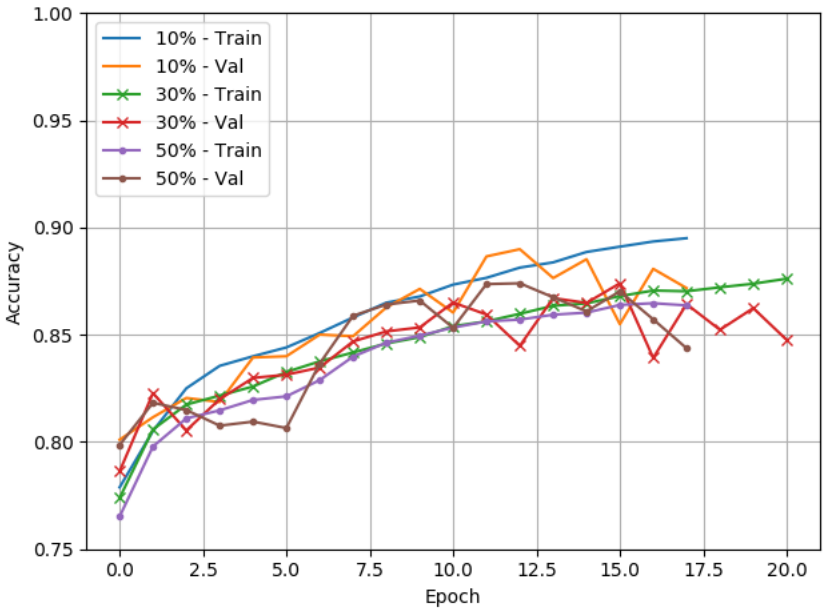


Figure B.15: Dropout on the NP Model with L1L1 regularisation.

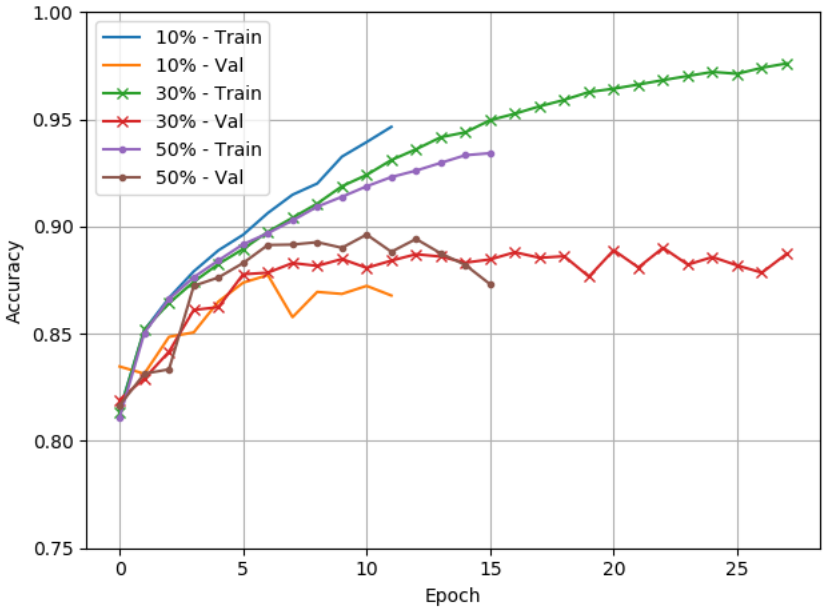


Figure B.16: Dropout percentage on the NP Model without regularisation.

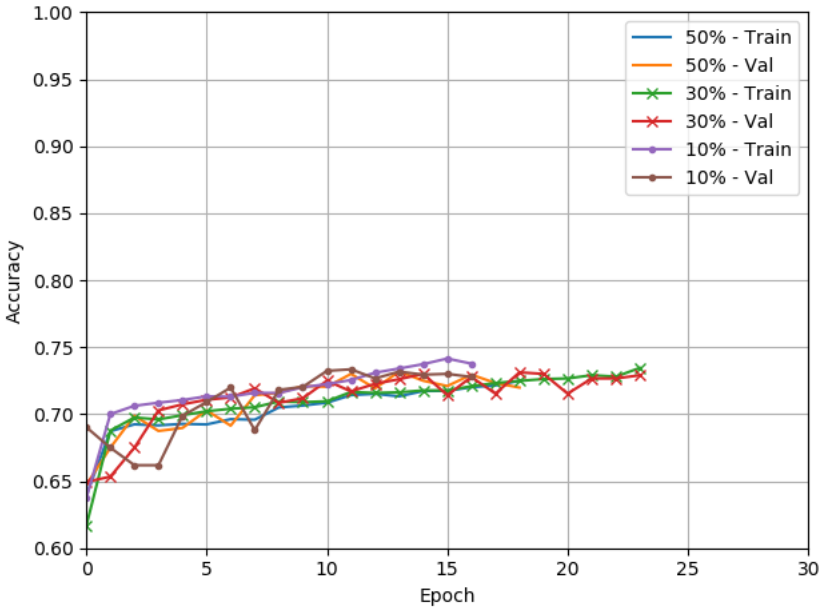


Figure B.17: Dropout percentage on the MM Model with L2L2 regularisation.

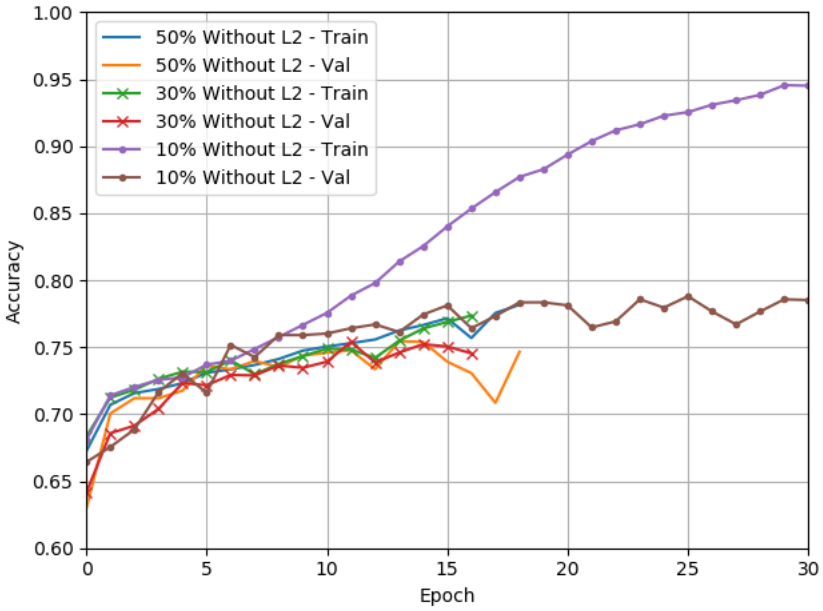


Figure B.18: Dropout percentage on the MM Model without regularisation.

Amount of Convolutional Layers

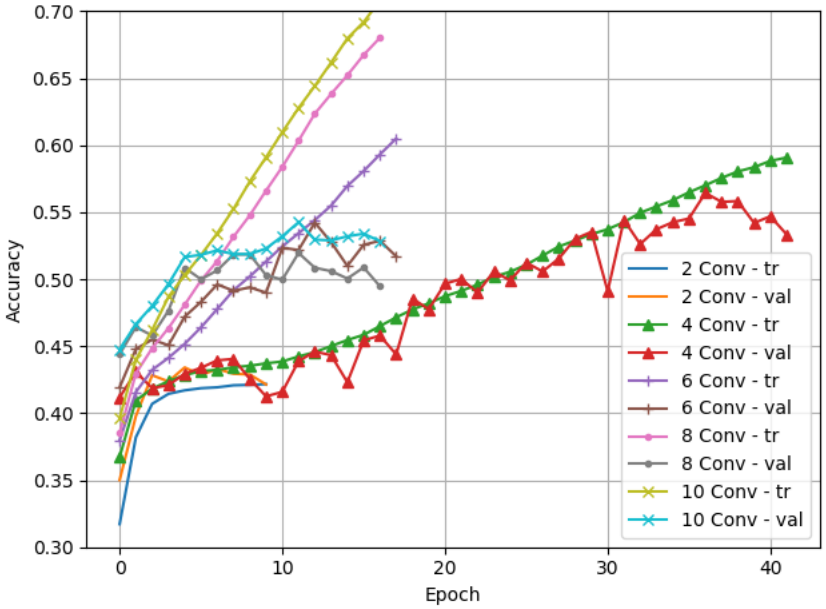


Figure B.19: Test of amount of convolutional layers on the FD model.

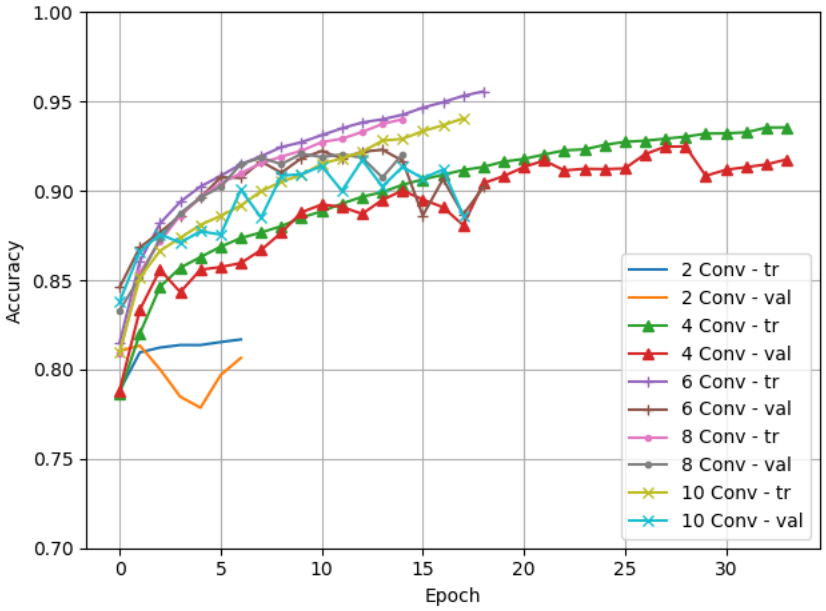


Figure B.20: Accuracy for convolutional layers in NP model.

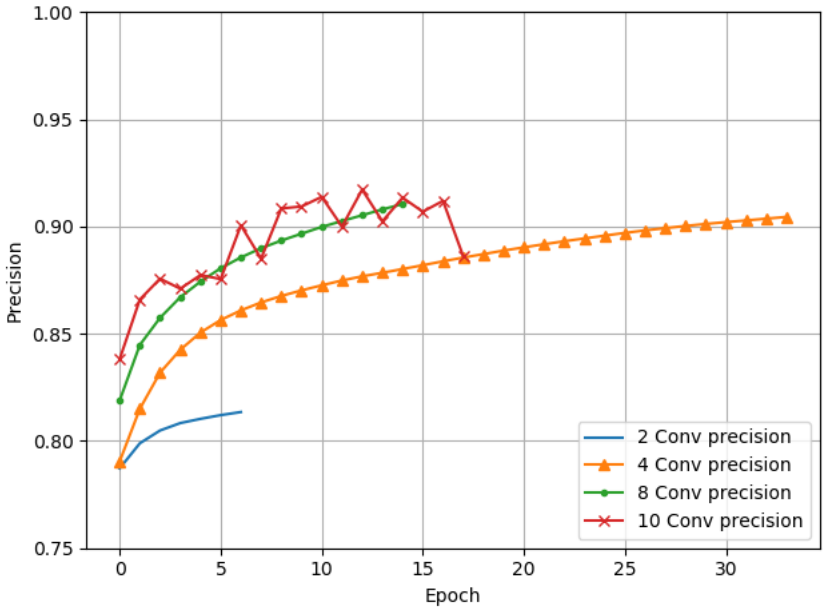


Figure B.21: Precision for convolutional layers in NP model.

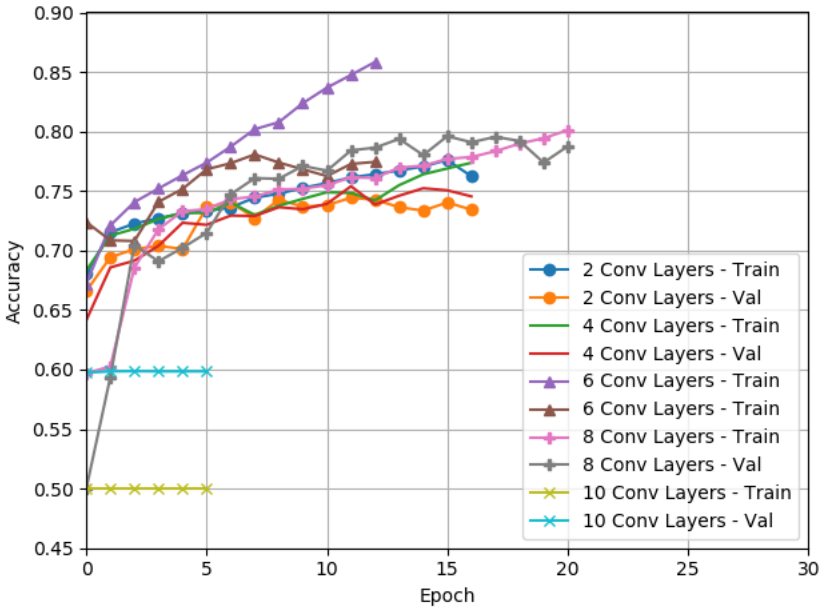


Figure B.22: Test of amount of convolutional layers on the MM model.

Filter Size

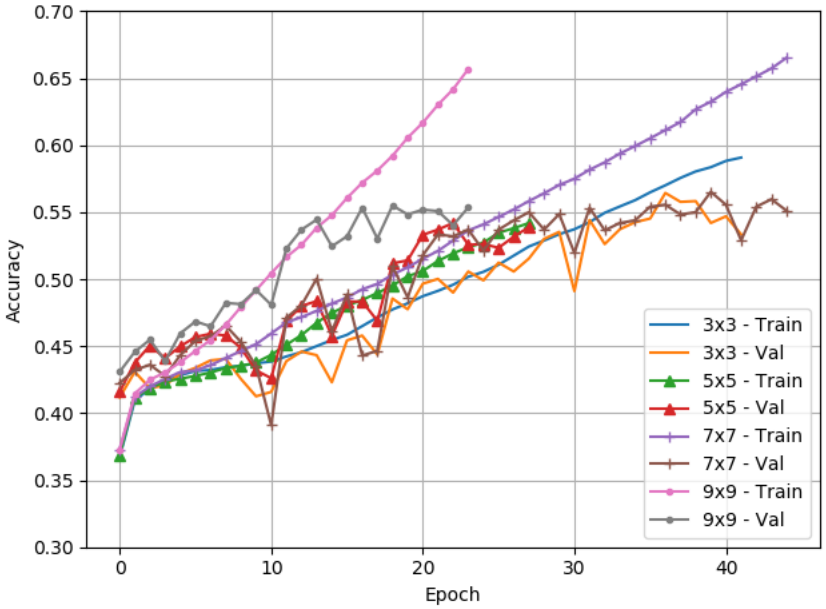


Figure B.23: FD Model

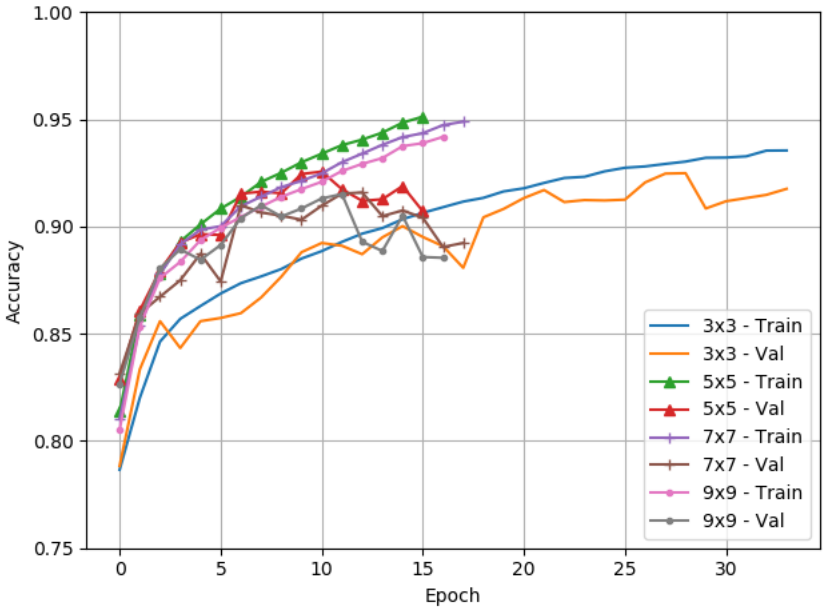


Figure B.24: NP Model

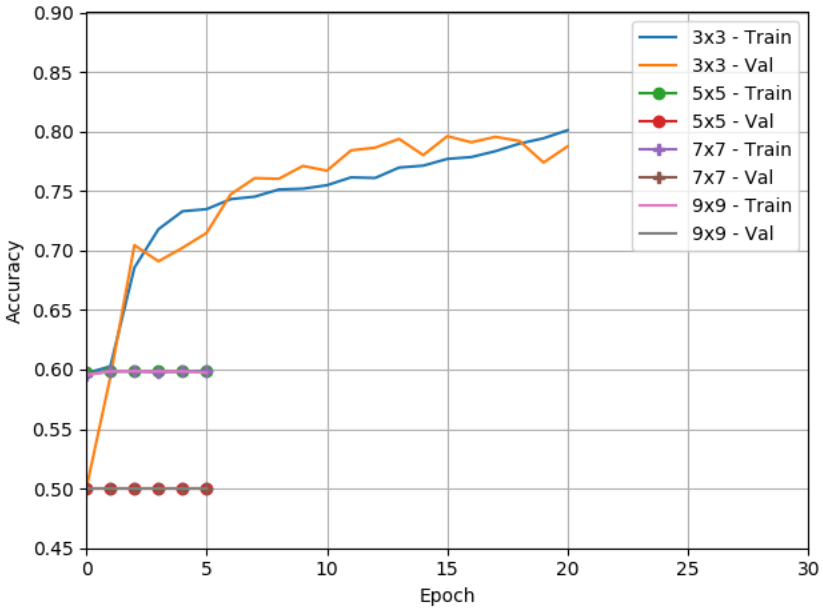


Figure B.25: MM Model

Amount of Pooling Layers

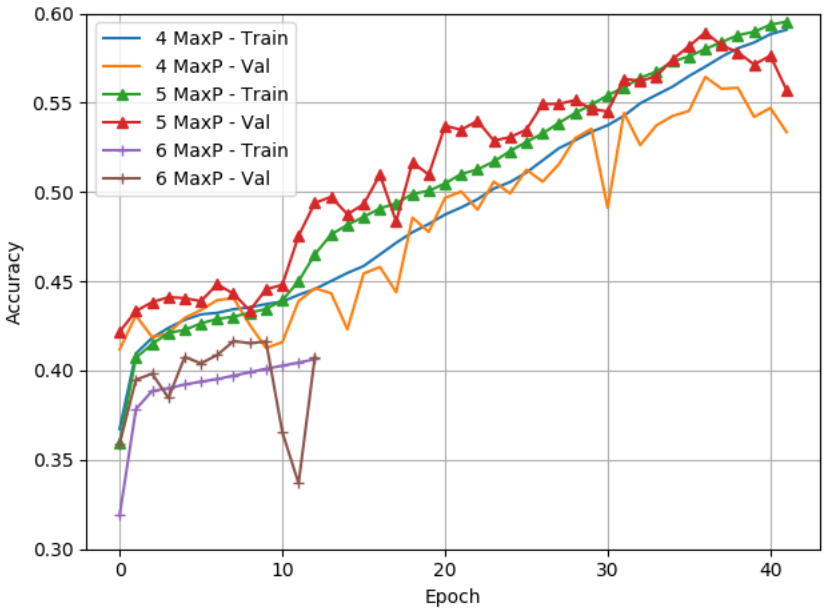


Figure B.26: Test of amount of pooling layers on the FD model.

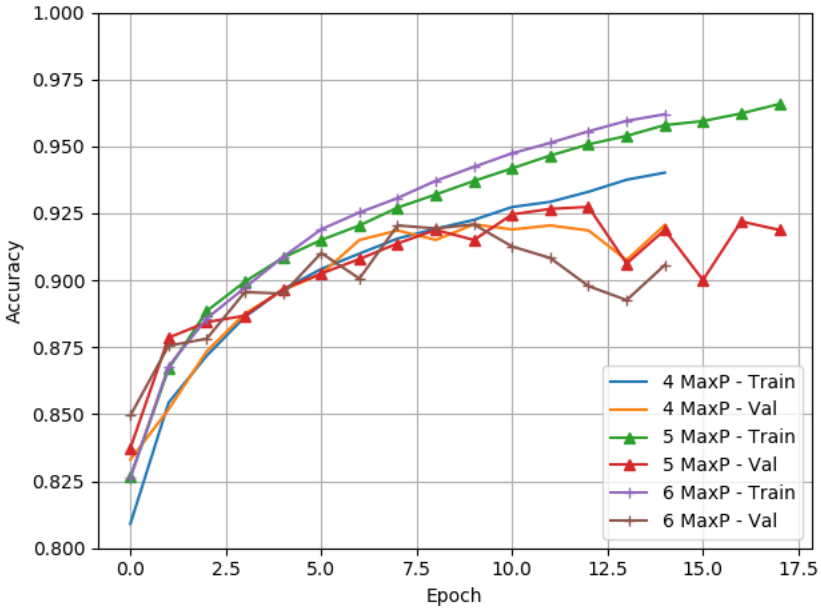


Figure B.27: Test of amount of pooling layers on the NP model.

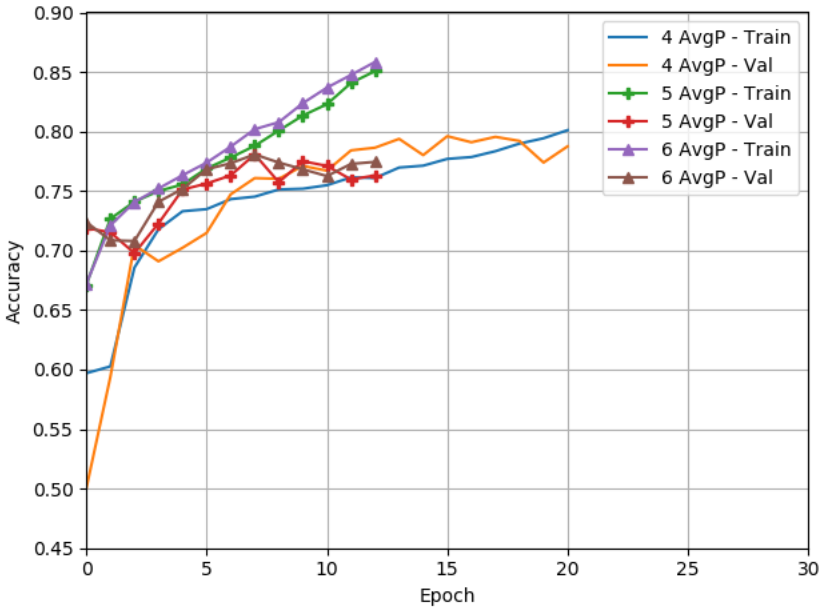


Figure B.28: Test of amount of pooling layers on the MM model.

Amount of Fully-Connected Layers

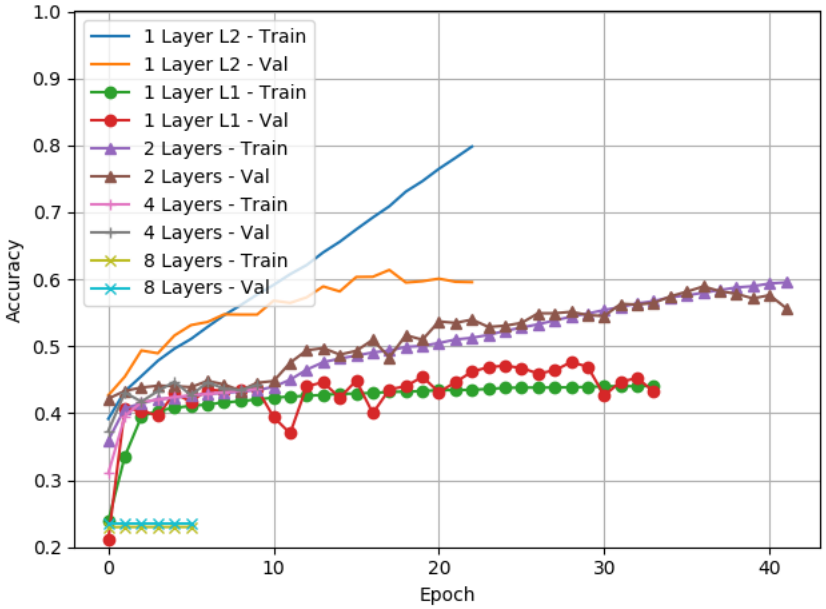


Figure B.29: Test of amount of fully-connected layers on the FD model.

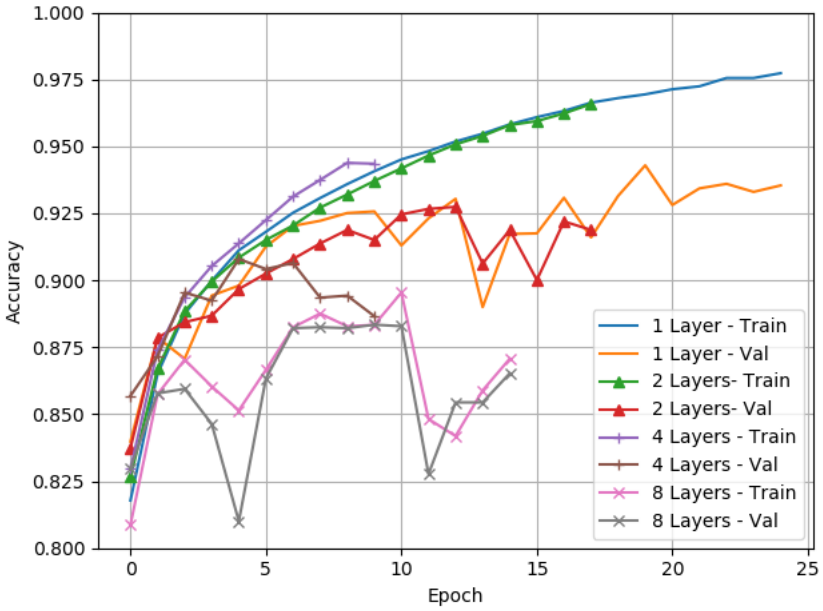


Figure B.30: Test of amount of fully-connected layers on the NP model.

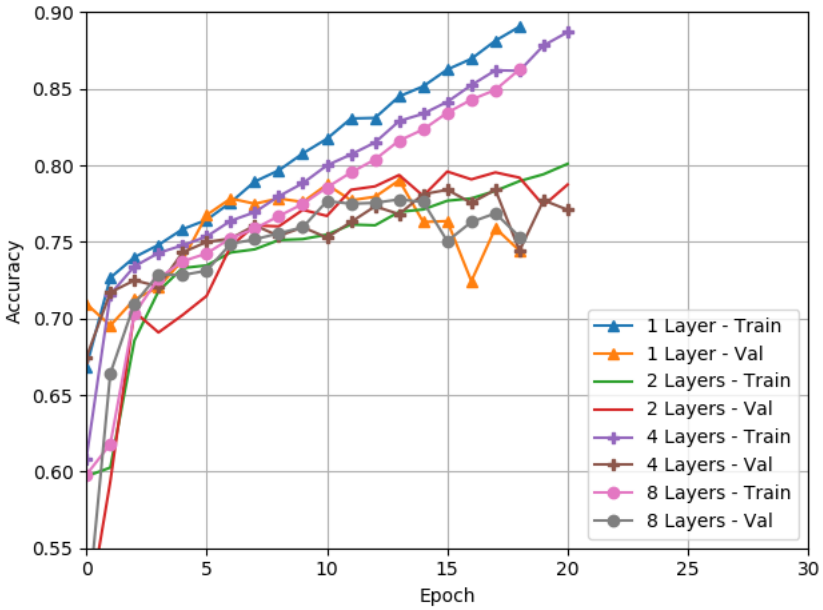


Figure B.31: Test of amount of fully-connected layers on the MM model.

Amount of Neurons

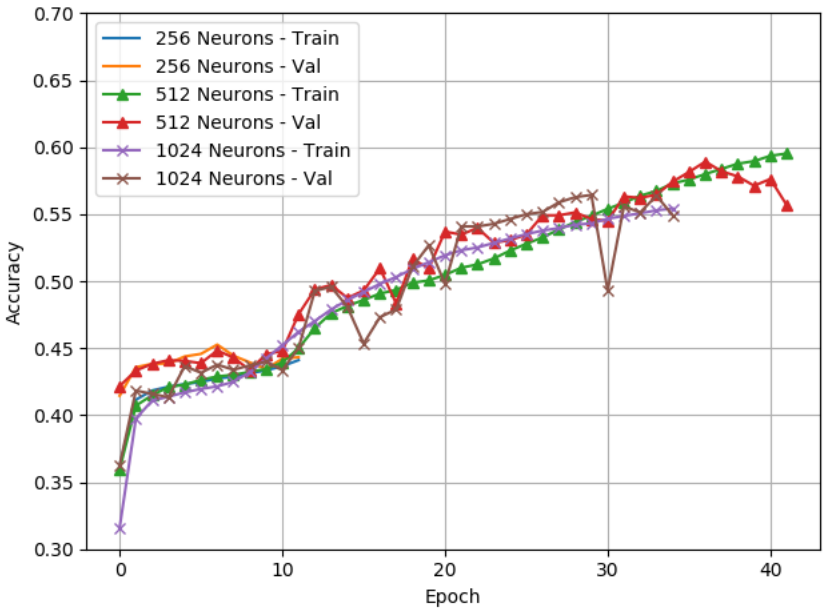


Figure B.32: Test of amount of neurons on the FD model.

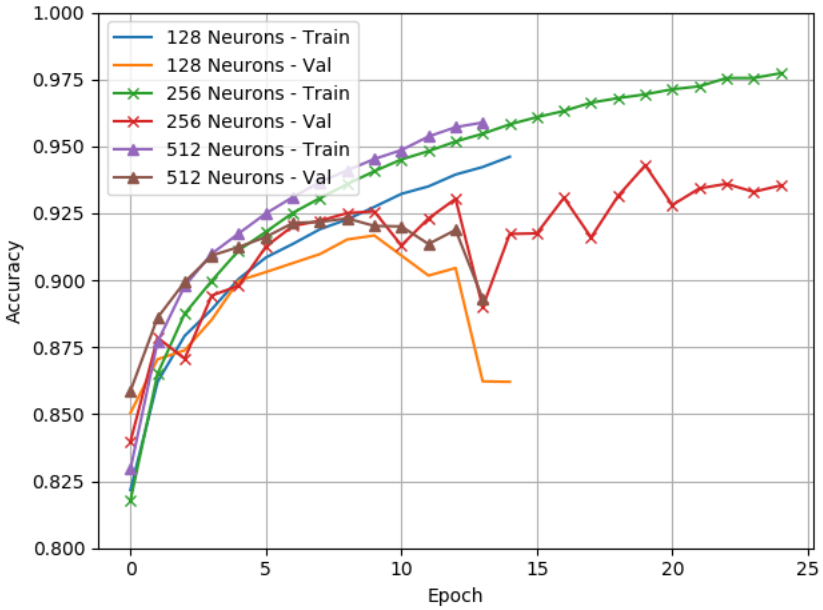


Figure B.33: Test of amount of neurons on the NP model.

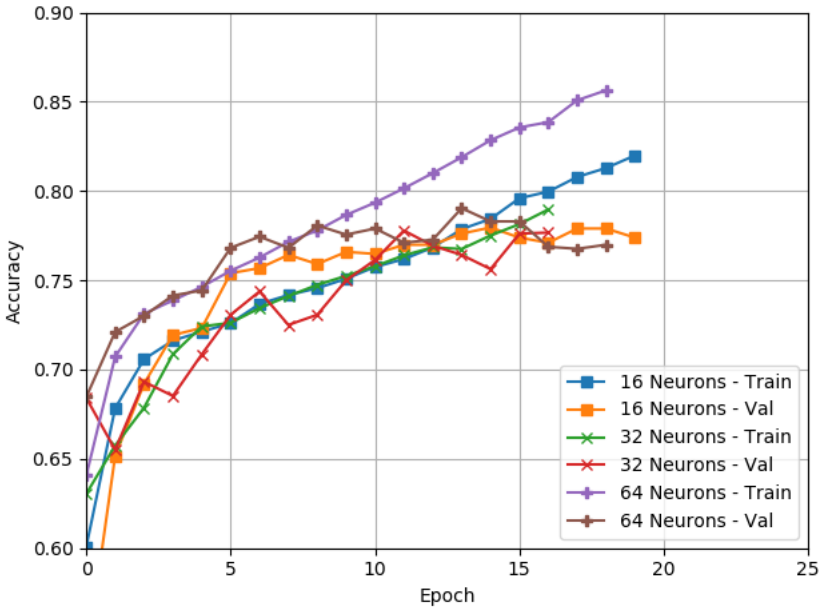


Figure B.34: Test of amount of neurons on the MM model.

Final Five Diagnosis Model

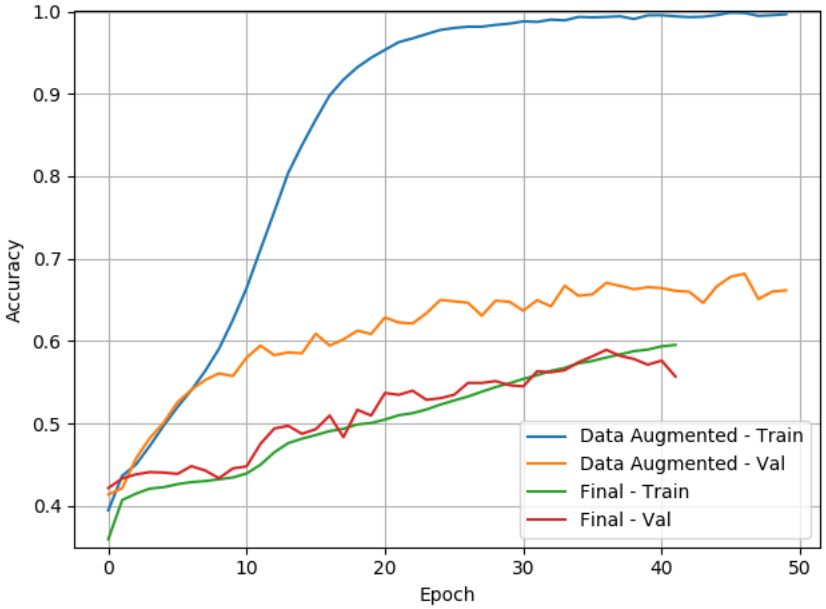


Figure B.35: Comparing the designated metrics for the FD model of its first test on the augmented dataset and its final version.

Final Divide & Conquer Model

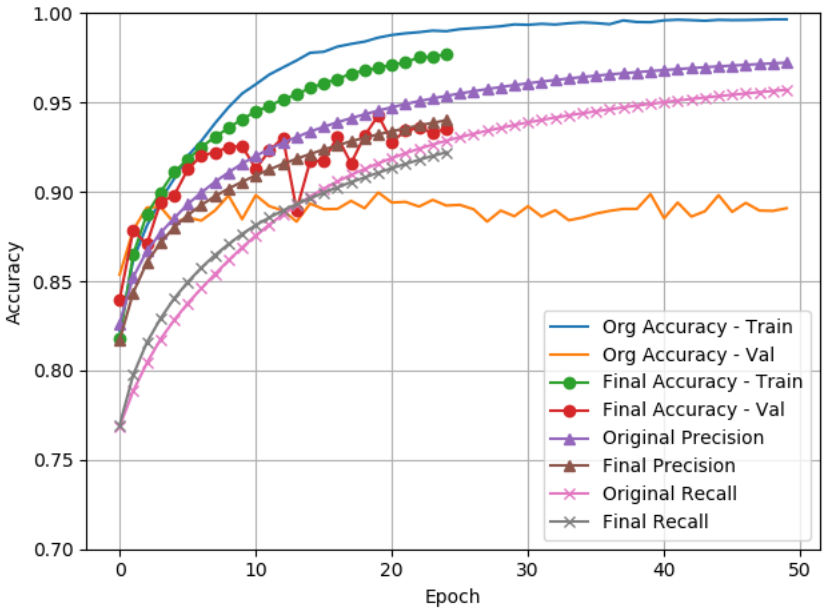


Figure B.36: Comparing the designated metrics for the NP model of its first test on the augmented dataset and its final version.

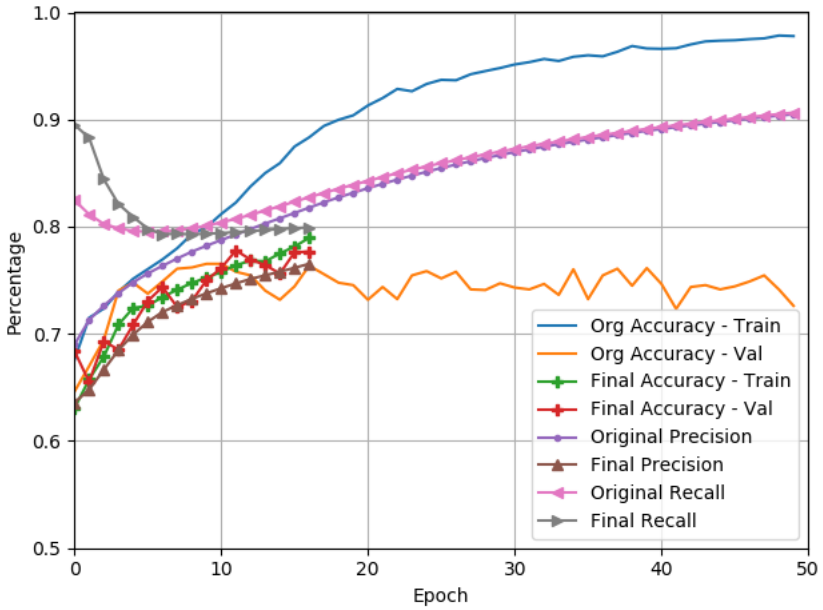


Figure B.37: Metrics of the first iteration and final version of the MM model trained on the augmented dataset.

C | Chi-Squared Table

C.1 Probability Values

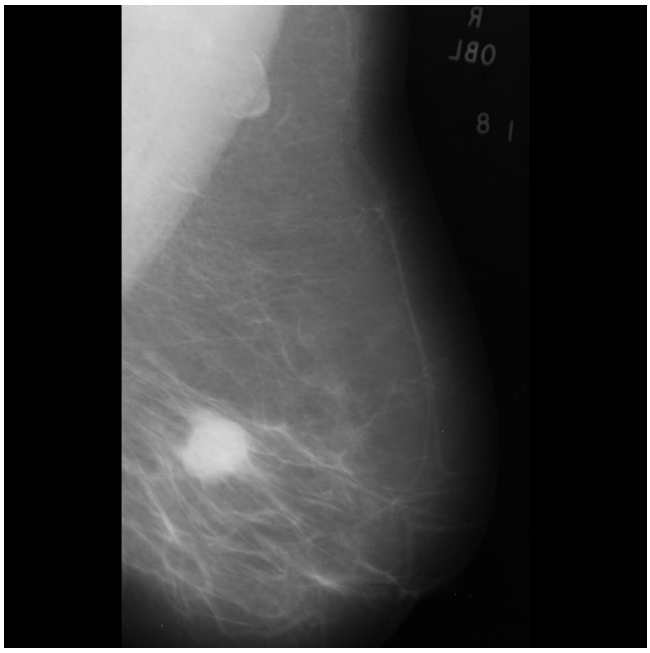
Df	*0.5*	*0.10*	*0.05*	*0.02*	*0.01*	*0.001*
1	0.455	2.706	3.841	5.412	6.635	10.827
2	1.386	4.605	5.991	7.824	9.210	13.815
3	2.366	6.251	7.815	9.837	11.345	16.268
4	3.357	7.779	9.488	11.668	13.277	18.465
5	4.351	9.236	11.070	13.388	15.086	20.517

D Predictions

D.1 Predictions by the Five Diagnosis Model

Analysis of mdb028.jpg

Original Image:



Date: 2019-12-07

Model Version:	Model_Version_Final_FD
Classification:	Five Category

Image: 1 of 16

0.07859 Negative
0.34545 Benign calcification
0.22096 Benign mass
0.22997 Malignant calcification
0.12503 Malignant mass

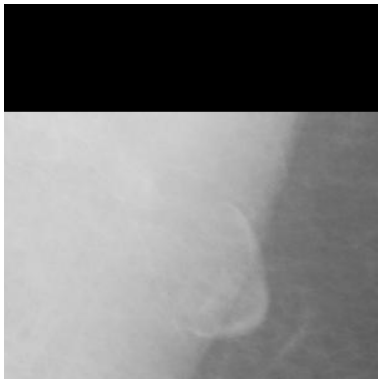


Image: 2 of 16

0.01244 Negative
0.00677 Benign calcification
0.78012 Benign mass
0.01216 Malignant calcification
0.18851 Malignant mass

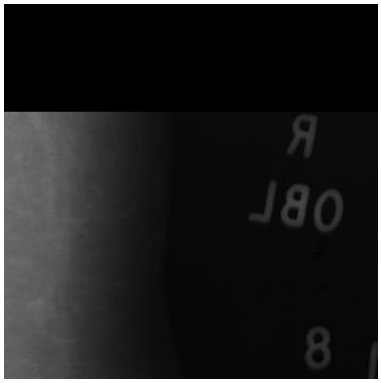


Image: 3 of 16

0.95289 Negative
0.02358 Benign calcification
0.00374 Benign mass
0.01469 Malignant calcification
0.00511 Malignant mass

Image: 4 of 16

0.82885 Negative
0.05590 Benign calcification
0.05306 Benign mass
0.03492 Malignant calcification
0.02727 Malignant mass

Image: 5 of 16

0.38593 Negative
0.12113 Benign calcification
0.26590 Benign mass
0.08544 Malignant calcification
0.14160 Malignant mass

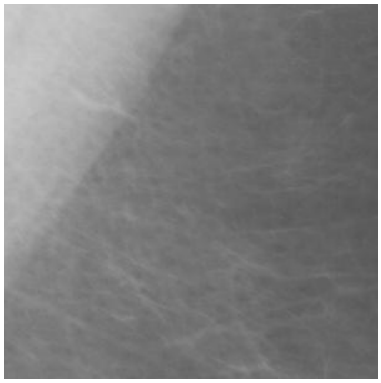


Image: 6 of 16

0.63381 Negative
0.20569 Benign calcification
0.03649 Benign mass
0.09926 Malignant calcification
0.02475 Malignant mass

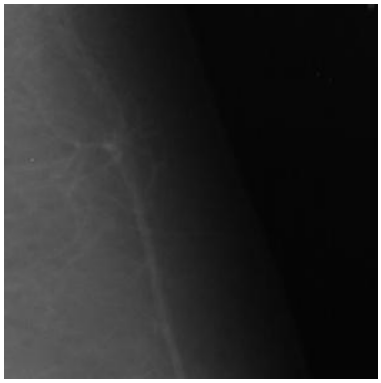


Image: 7 of 16

0.98909 Negative
0.00422 Benign calcification
0.00107 Benign mass
0.00405 Malignant calcification
0.00158 Malignant mass

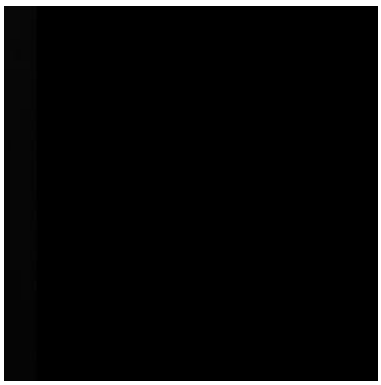


Image: 8 of 16

0.88407 Negative
0.04711 Benign calcification
0.02665 Benign mass
0.02679 Malignant calcification
0.01538 Malignant mass

Image: 9 of 16

0.48665	Negative
0.11278	Benign calcification
0.21131	Benign mass
0.07777	Malignant calcification
0.11150	Malignant mass

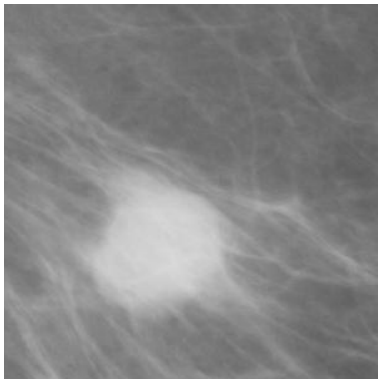


Image: 10 of 16

0.00010	Negative
0.00031	Benign calcification
0.07149	Benign mass
0.00469	Malignant calcification
0.92341	Malignant mass

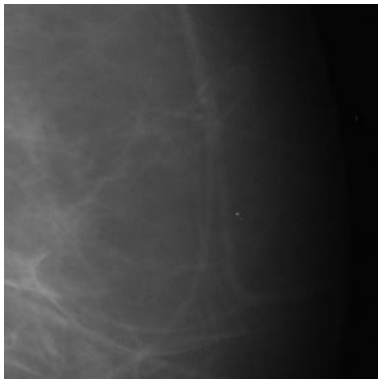


Image: 11 of 16

0.87837	Negative
0.05959	Benign calcification
0.01391	Benign mass
0.03734	Malignant calcification
0.01079	Malignant mass

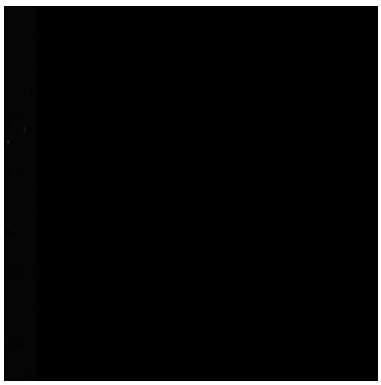


Image: 12 of 16

0.82572	Negative
0.06085	Benign calcification
0.04947	Benign mass
0.03680	Malignant calcification
0.02715	Malignant mass

Image: 13 of 16

0.64597	Negative
0.09187	Benign calcification
0.13223	Benign mass
0.06099	Malignant calcification
0.06894	Malignant mass

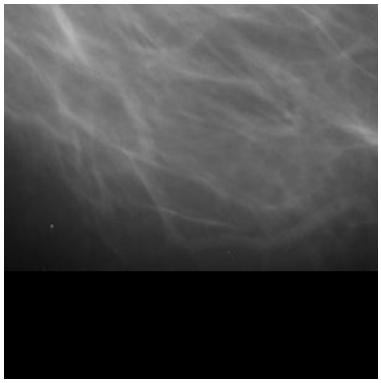


Image: 14 of 16

0.00909	Negative
0.17632	Benign calcification
0.37641	Benign mass
0.26043	Malignant calcification
0.17775	Malignant mass

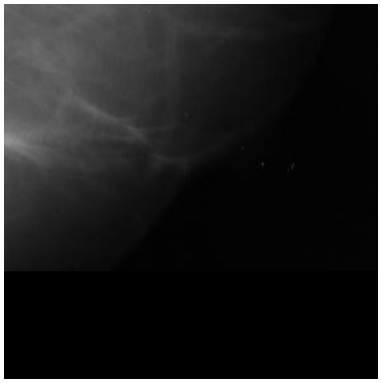


Image: 15 of 16

0.97110	Negative
0.01234	Benign calcification
0.00360	Benign mass
0.00913	Malignant calcification
0.00383	Malignant mass

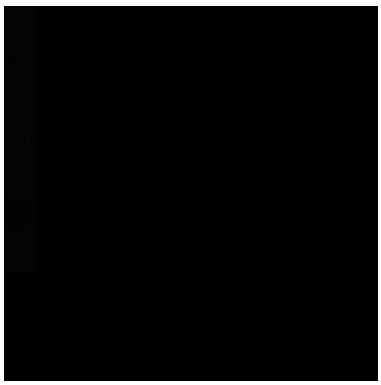
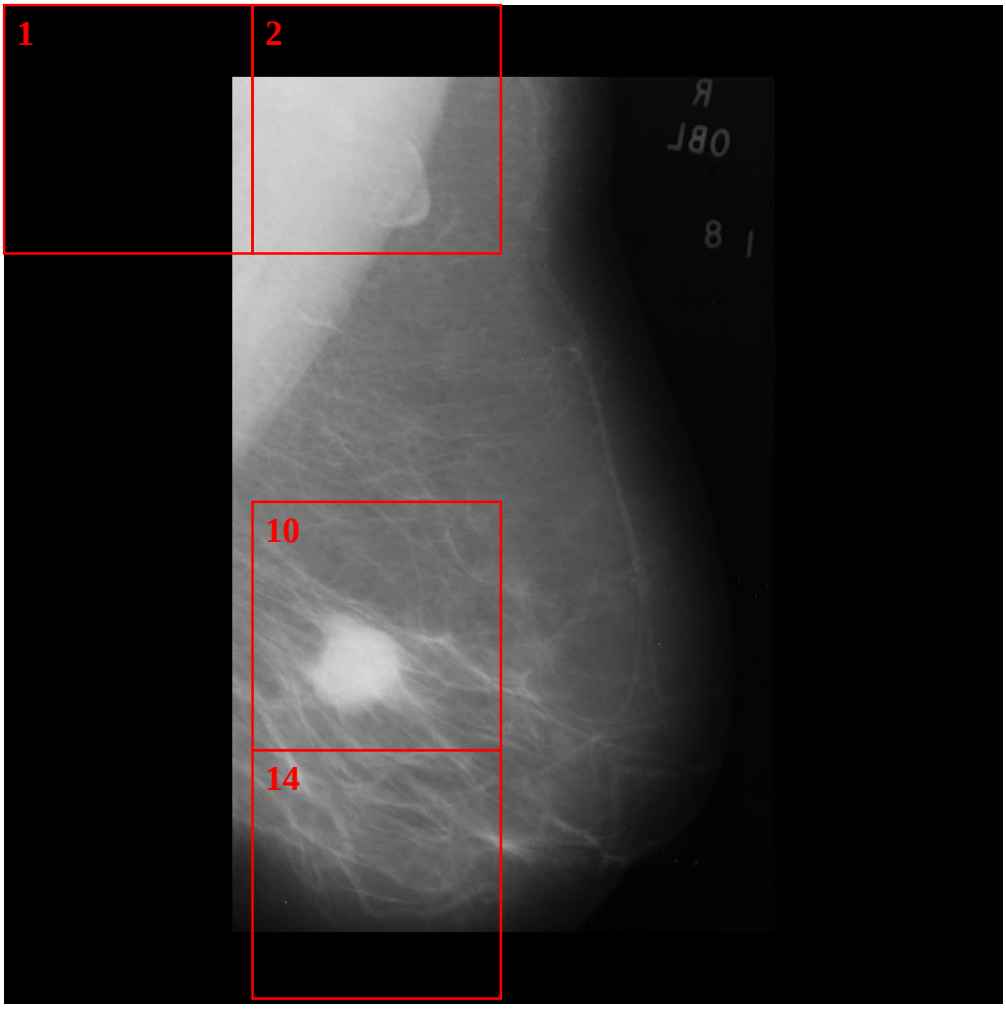


Image: 16 of 16

0.66911	Negative
0.08412	Benign calcification
0.12667	Benign mass
0.05652	Malignant calcification
0.06358	Malignant mass

Product Image:



The Regions of Interest

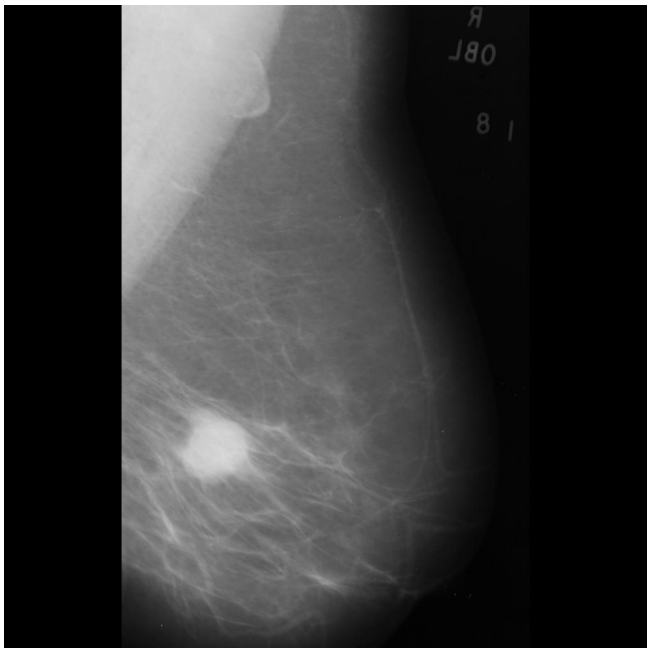
Diagnosis of Interest

- Benign Calcification
- Benign Mass
- Malignant Mass

D.2 Predicitons by the Negative/Positive Model

Analysis of mdb028.jpg

Original Image:



Date: 2019-12-06

Model Version:	Model_Version_2_06c
Classification:	Binary

Image: 1 of 16

0.03313 Negative
0.96687 Positive

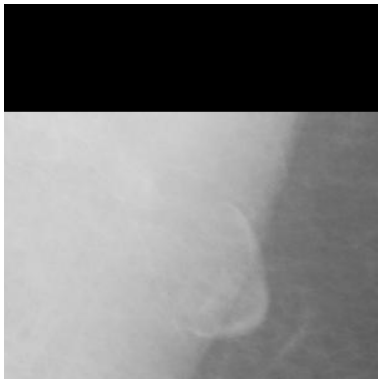


Image: 2 of 16

0.00000 Negative
1.00000 Positive

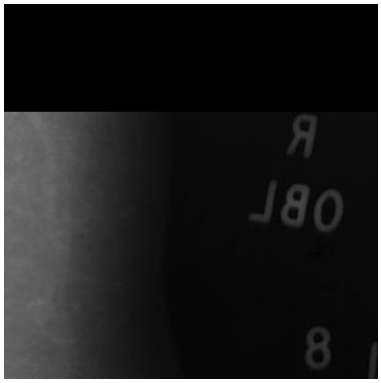


Image: 3 of 16

1.00000 Negative
0.00000 Positive

Image: 4 of 16

0.99685 Negative
0.00315 Positive

Image: 5 of 16

1.00000 Negative

0.00000 Positive

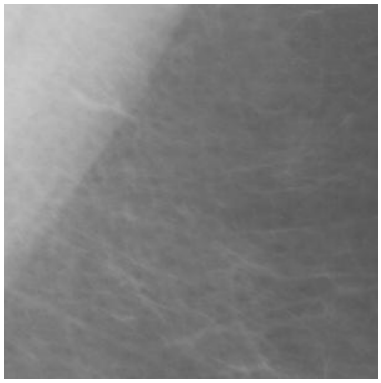


Image: 6 of 16

0.97589 Negative

0.02411 Positive

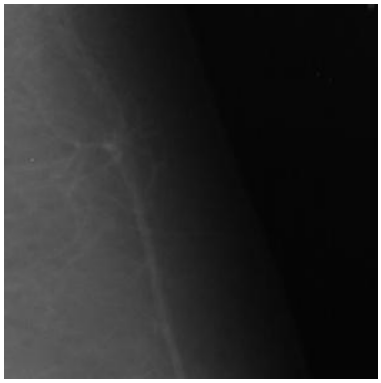


Image: 7 of 16

1.00000 Negative

0.00000 Positive

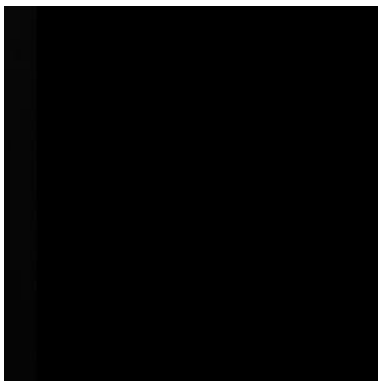


Image: 8 of 16

0.99648 Negative

0.00352 Positive

Image: 9 of 16

0.99486 Negative

0.00514 Positive

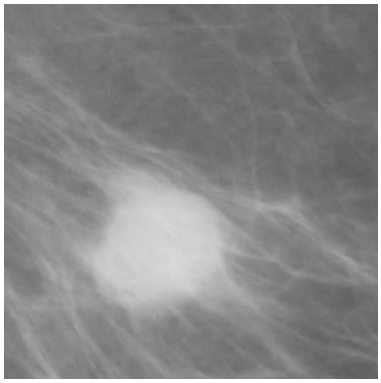


Image: 10 of 16

0.00000 Negative

1.00000 Positive

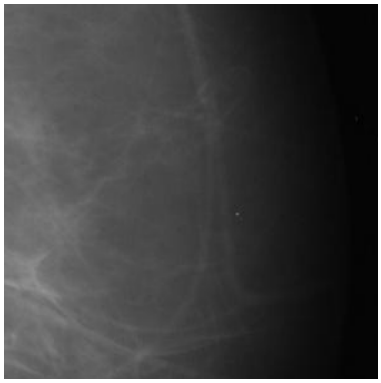


Image: 11 of 16

1.00000 Negative

0.00000 Positive

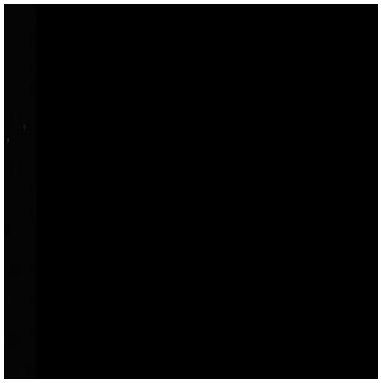


Image: 12 of 16

0.99640 Negative

0.00360 Positive

Image: 13 of 16

0.99491 Negative

0.00509 Positive

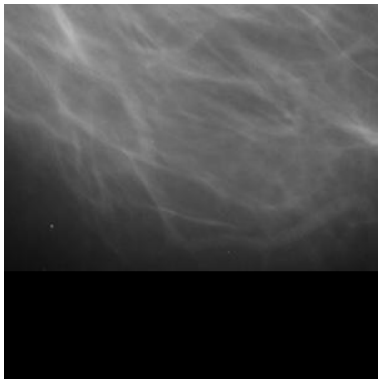


Image: 14 of 16

0.00000 Negative

1.00000 Positive

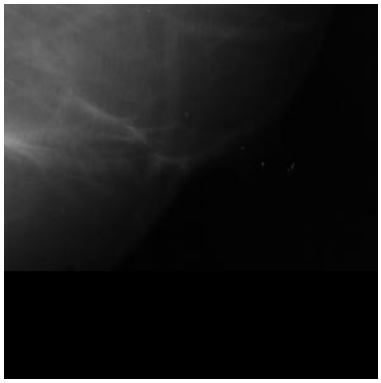


Image: 15 of 16

0.99980 Negative

0.00020 Positive

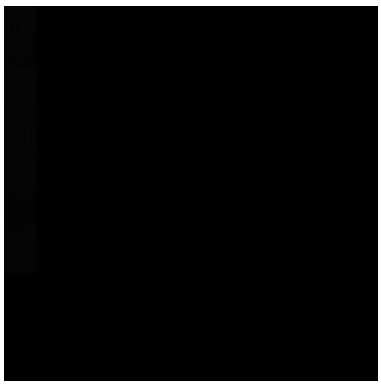


Image: 16 of 16

0.99588 Negative

0.00412 Positive

Product Image: