MAKERERE UNIVERSITY

COLLEGE OF ENGINEERING, DESIGN, ART AND TECHNOLOGY

FINAL YEAR PROJECT REPORT

FOR THE DEGREE OF BACHELOR OF SCIENCE IN

COMPUTER ENGINEERING

Name: **MUNGUJAKISA NICKSON**

Reg. No: **14/U/700**

Date: **JUNE 2018**

MAKERERE UNIVERSITY

COLLEGE OF ENGINEERING, DESIGN, ART AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CLASSIFICATION OF LESIONS IN BREAST ULTRASOUND IMAGES USING NEURAL NETWORKS

NICKSON MUNGUJAKISA

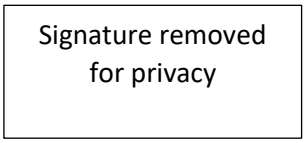
14/U/700

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIRMENTS FOR THE AWARD OF A BACHELOR OF SCIENCE IN COMPUTER ENGINEERING

JUNE 2018

Declaration

I, MUNGUJAKISA NICKSON, have abided by the Makerere University academic integrity policy on this assignment.

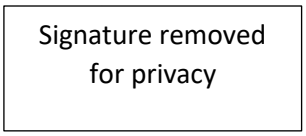
Signature 

on this day of Tuesday, 10 July 2018

Mungujakisa Nickson

14/U/700

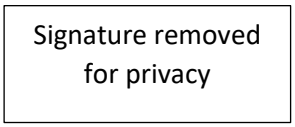
This report has been submitted for examination with the approval of the following supervisors:

Signed 

on this day of Tuesday, 10 July 2018

Ms. Sheila Mugala

Makerere University

Signed 

on this day of Tuesday, 10 July 2018

Mr. Cosmas Mwikirize

Makerere University

Acknowledgement

I would like to thank my project supervisors Ms. Sheila Mugala and Mr. Cosmas Mwikirize for their guidance and assistance over the course of this project.

Special thanks to Ogwal Emmanuel Hian, who I worked with on this project.

Finally, I would like to thank my parents for all the support they provided over the course of this project.

Abstract

In this project we aimed to build and evaluate a neural network framework for classification of lesions in breast ultrasound images. A classification model based on k-Nearest Neighbour (k-NN) algorithm was built to serve as an evaluation baseline. 4 neural network models were then built using the *TensorFlow* and *Keras* deep learning libraries; A fully connected neural network, a custom Convolutional Neural Network (CNN) and two transfer learning networks based on retraining InceptionV3 which is a state of the art general purpose image classification CNN. Neural network approaches outperformed the k-NN. The CNN manages to achieve a low false negative rate and high true positive rate hence a high sensitivity of 0.85. The transfer learning networks underperform due to data limitations. More data is recommended to be used in future studies to fully investigate the potential of transfer learning for breast ultrasound lesion classification.

Table of Contents

List of Tables

List of Figures

List of Acronyms and Nomenclature

Adagrad – Adaptive gradient

ANN – Approximate Nearest Neighbours

AUC – Area Under the Curve

CNN – Convolutional Neural Network

FCN – Fully Connected Neural Network

FN – False Negative

FP – False Positive

k-NN – k Nearest Neighbours

ReLU – Rectified Linear Unit

RGB – Red Green Blue

ROC – Receiver Operating Characteristics

SGD – Stochastic Gradient Descent

TN – True Negative

TP – True Positive

VGG – Visual Graphics Group

Chapter One: Introduction

1.1 Introduction

Breast cancer is the second leading cause of cancer-related deaths in Ugandan women, with an incidence of 45 in 100,000. A steep increase in incidence, currently 6%, is a cause for worry. Early and accurate diagnosis of breast cancer is vital for good prognosis. Traditionally, screening once a year is recommended to check for symptoms of the disease. Screening is performed via X-ray mammograms followed by ultrasound imaging. Suspected malignant tumours are biopsied and subjected to histological studies. However, diagnosis of breast cancer from scans and histology is highly subjective and relies on the experience of the radiologist and/or pathologist. Variability in diagnosis by different radiologists/pathologists for the same case is also common.

1.2 Problem Statement

Diagnosis of breast cancer from scans is highly subjective and relies on the experience and availability of radiologists, who are few in Uganda. According to the Ministry of Health records, as of early 2018 Uganda has only 48 radiologists that are meant to serve the almost 40,000,000 Ugandans [1]. This gives a doctor patient ratio of one radiologist for every 850,000 people. Of the 48 radiologists, 11 are retired leaving only 37 in active service [1]. Radiologists classification of lesions from visual analysis of breast ultrasound images can be subject to variability of diagnosis depending on various factors such as exhaustion or level of experience. Furthermore, certain regions of the country have far less radiologists than others because various economic factors make these places unattractive to medical personnel. All these conditions make automating some of the work of radiologists a worthwhile endeavour.

1.3 Objectives

Main objective:

Develop and validate a neural network framework for classification of breast lesions from ultrasound scans.

Specific objectives:

- Build a non-neural network based image classifier model to serve as a baseline evaluation model
- Build a fully connected neural network (FCN) for lesion classification
- Build a custom convolutional neural network (CNN) for lesion classification
- Re-train a state of the art CNN for lesion classification
- Validate all constructed models using clinical data

1.4 Previous Related work

Human analysis combined with CNNs has been shown to achieve a 99.5 percent success rate in diagnosing metastatic breast cancer from whole slide digital images of breast sentinel lymph nodes [2].

CNNs have also been used to do automated bone age assessment. The system developed was able to give a bone age assessment with a 98.11% accuracy (within 1 year) and a 99.00% accuracy in women and men respectively [3].

In 2016 researchers affiliated with Google investigated how an automated CNN based algorithm compares with manual grading by ophthalmologists in identifying diabetic retinopathy in retinal fundus photographs. The researchers found that deep learning algorithms had high sensitivity and specificity for detecting diabetic retinopathy and macular edema in retinal fundus photographs [4].

Neural networks have also been shown to more accurately detect and localise lesions in breast ultrasound images when compared to other state of the art lesion detection algorithms such as Radial Gradient Index, Multifractal Filtering, Rule-based Region Ranking and Deformable Part Models [5].

Chapter Two: Literature Review

2.1 Dataset

The clinical data that was used to train and validate constructed models was obtained from a team of researchers from Manchester Metropolitan University in the United Kingdom and the University of Girona in Spain.

The dataset consists of 163 scans collected from the UDIAT Diagnostic Center of the Parc Tauli Corporation in Sabadell, Spain. The scans were taken with a Siemens ACUSON Sequoia C512 17L5 HD Linear array transducer (8.5 MHz) ultrasound system. The images are from different women with a mean image size of 760X570 pixels, where each of the images presented one or more lesions. Within the 163 lesion images, 53 were images with cancerous masses and 110 with benign lesions. From the malignant images, 40 were invasive ductal carcinomas, 4 were ductal carcinomas in situ, 2 were invasive lobular carcinomas and 7 were other unspecified malignant lesions. From the benign images, 65 were unspecified cysts, 39 were fibroadenomas and 6 were of other types of benign lesions [5].

The dataset is publicly available. To access the dataset, a licence agreement was signed after which access information to the dataset was given and used to download the files.

The data was then randomly split it into two sets, one with 100 scans the other with 63. The 100 scans (the training set) were used for training and cross validating models while the 63 scans (the test set) were used for final testing and performance evaluation.

2.2 Data Augmentation

The 100 scans in the training set were passed through seven transformations (blurring, two types of reflection, three types of rotations and sharpening) as shown in *Figures 1 to 7*

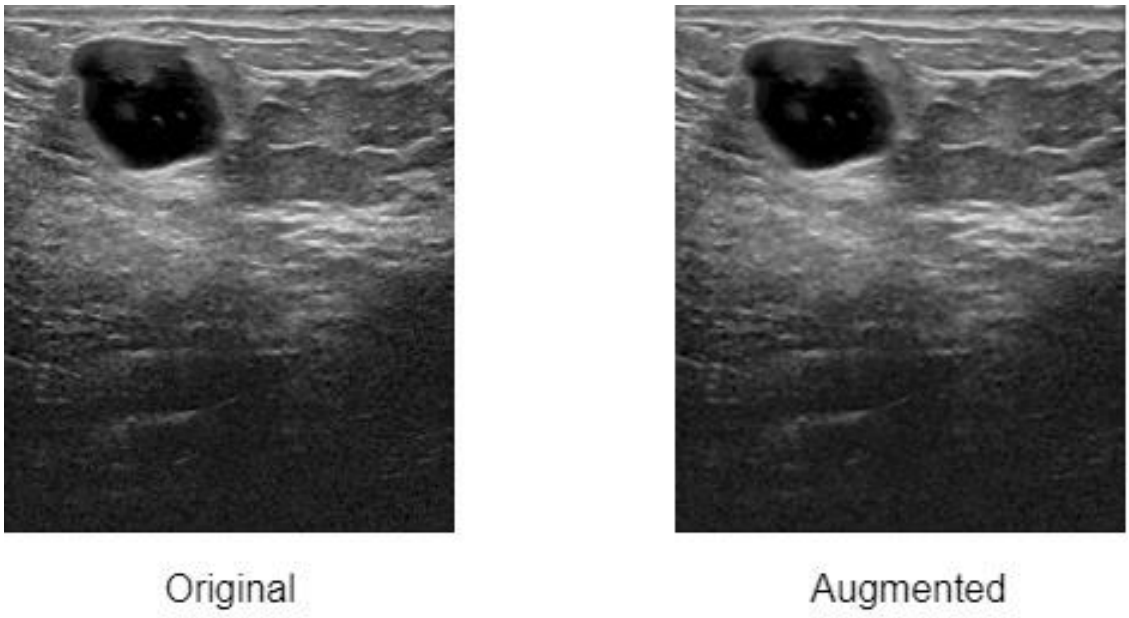


Figure 1: Blurring augmentation

In *Figure 1*, each pixel of the image has its intensity reduced to result in a blurry image. To the human eye the two images in *Figure 1* don't have any clearly visible difference but on a low level to a computer the two images are quite different.

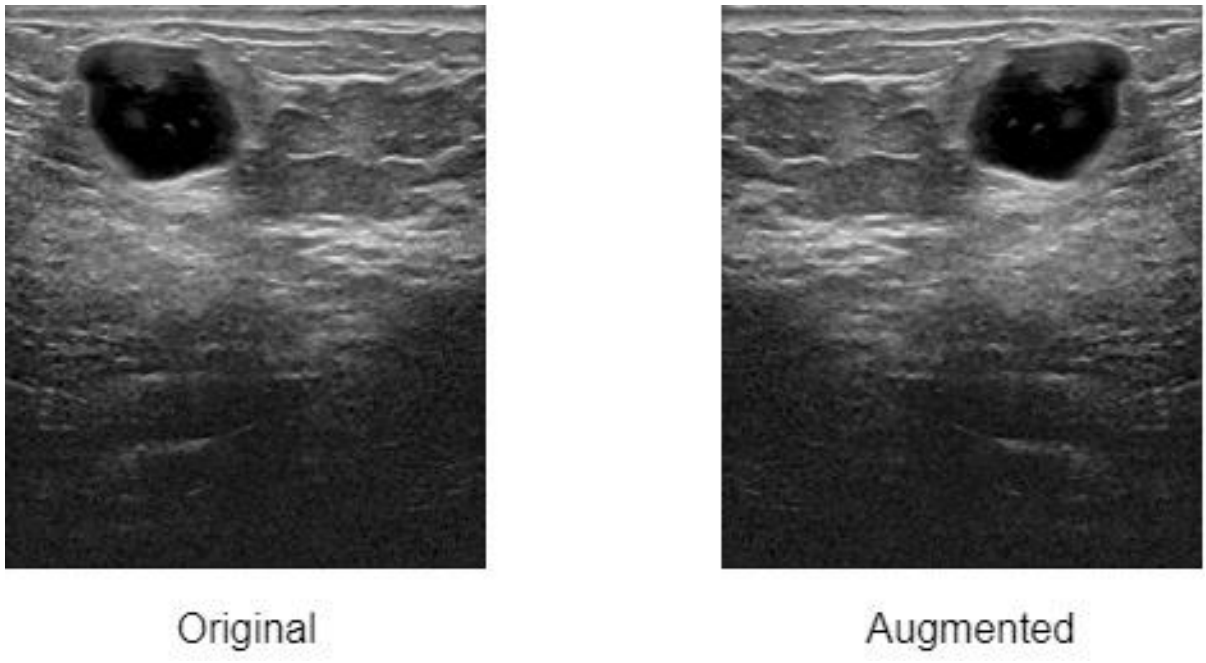


Figure 2: Flip from left to right

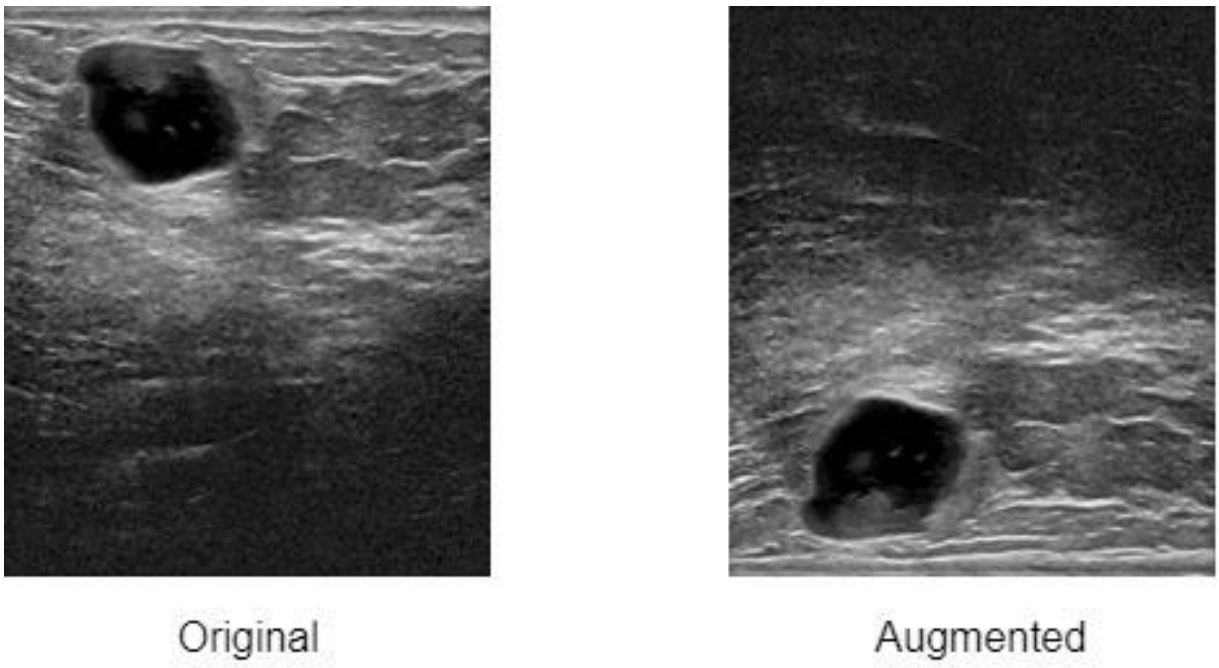


Figure 3: Image flipped from top to bottom

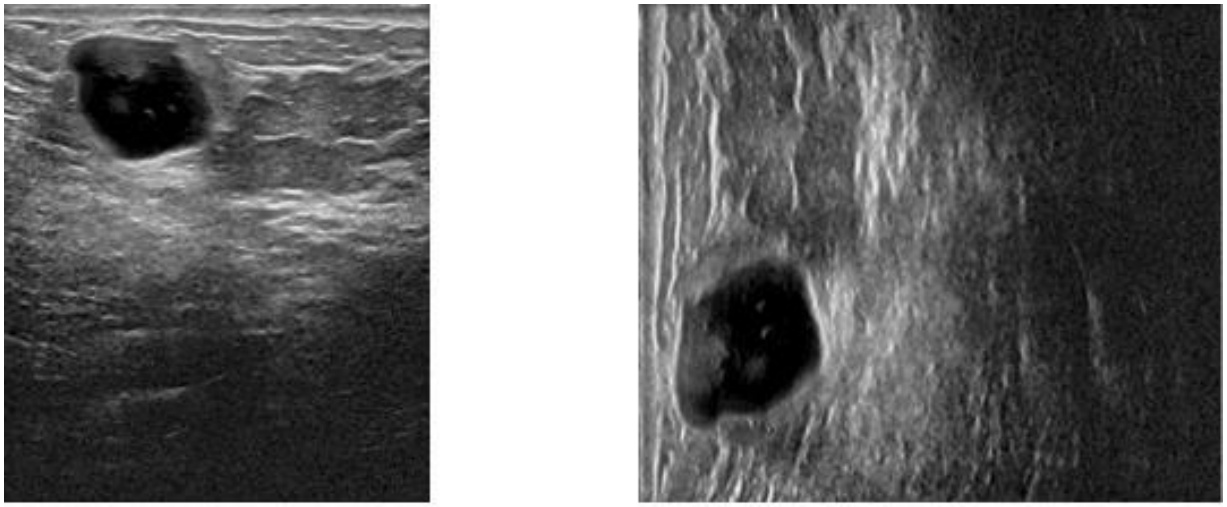


Figure 4: Image rotated through 90 degrees

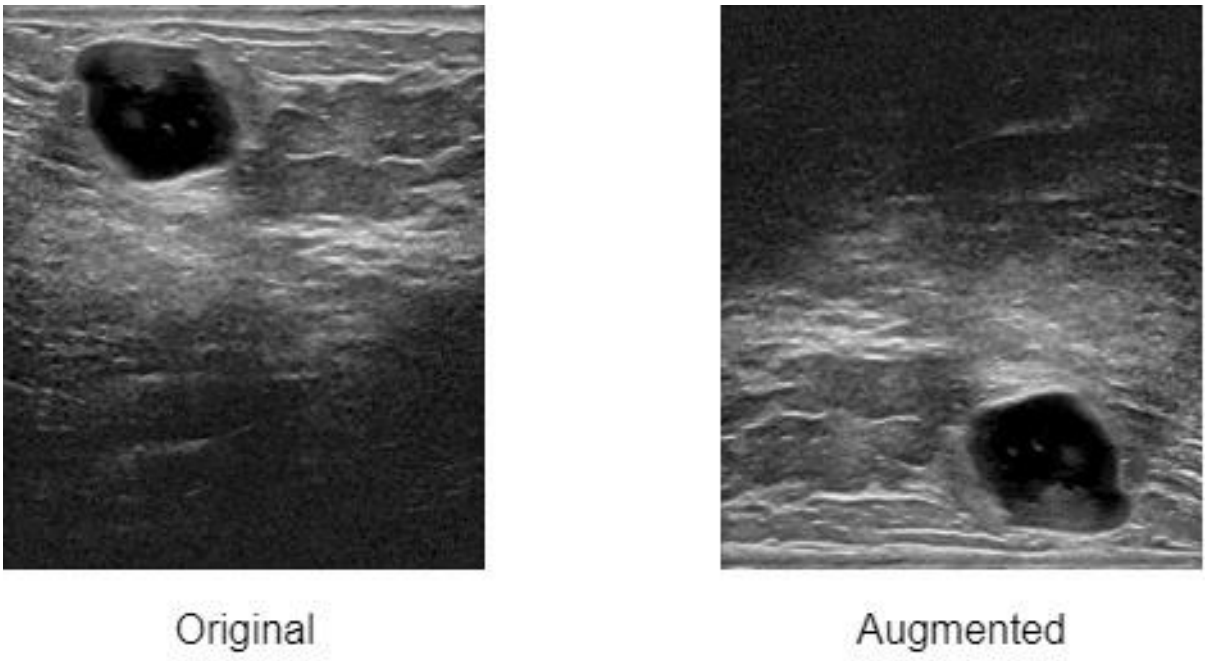


Figure 5: Image rotated through 180 degrees

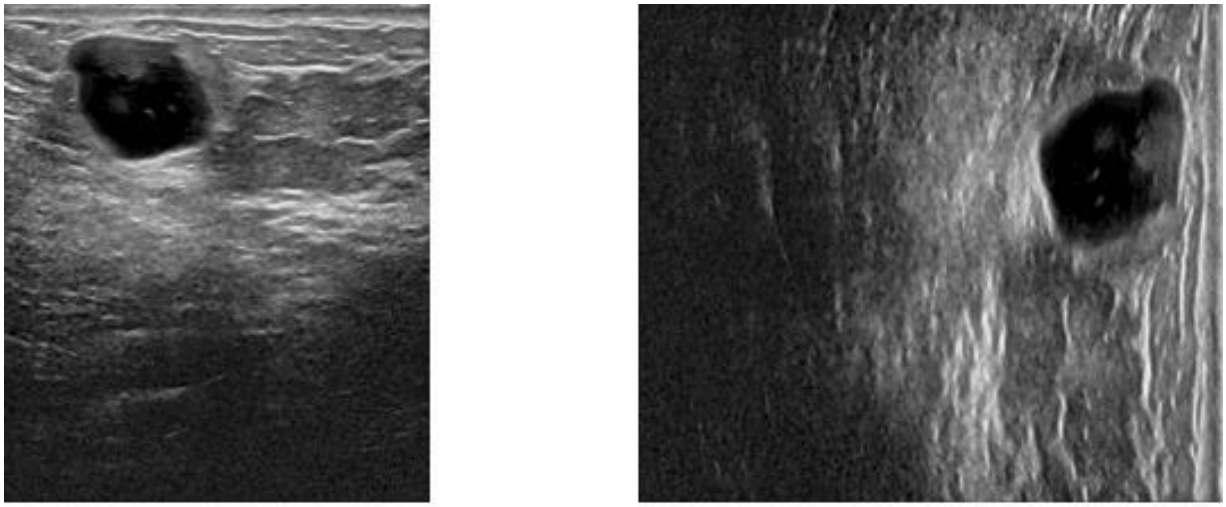


Figure 6: Image rotated through 270 degrees

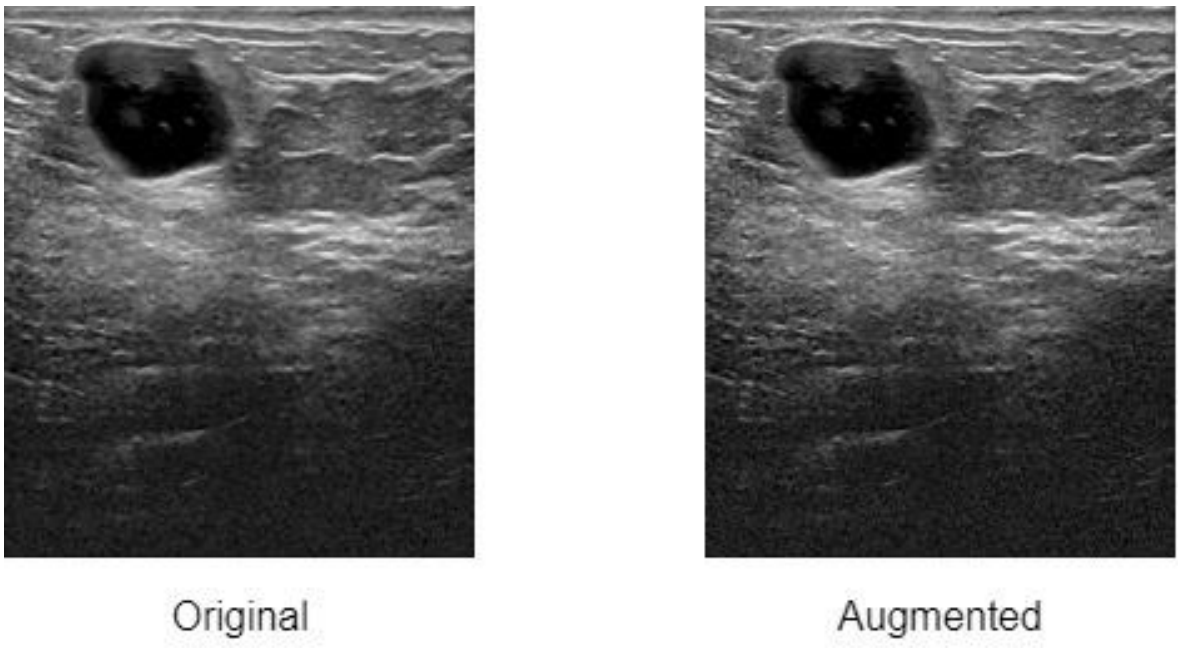


Figure 7: Image sharpened by a factor of 2

After the transformations the training set effectively had 800 scans. Transformations were done using the *Python* image processing library called *Pillow*. Scans in the test set were not transformed.

The rationale behind these transformations is that the features that care used to classify lesions are rotationally agnostic. This makes sense since regardless of an ultrasound scans orientation, a radiologist can still recognise a lesion and classify it. Automated classification models should therefore also be impervious to such distortion.

2.3 Data Pre-Processing

All image analysis algorithms and models pre-process images before they analyse them. Images (even those of the same object) to be analysed usually come from different sources and are in different configurations (e.g. different resolutions). Pre-processing serves to standardise the images before they are passed through algorithms. These algorithms are usually designed to have a relatively fixed structure and therefore expect inputs to be in a specific configuration.

The scans in the dataset used averaged 760X570 pixels and the images are in grayscale (monochrome with one colour channel representing only intensity information). Scans for all constructed models were pre-processed into the configurations shown in *Table 1*.

Table 1: Pre-processing configurations

Configuration name	Horizontal pixels	Vertical pixels	Colour channels
528X360X1	528	360	1
528X360X3	528	360	3
224X224X1	224	224	1
224X224X3	224	224	3
150X150X1	150	150	1
150X150X3	150	150	3

Pre-processing was done using the *Pillow Python* library. The library was used to resize the scans to the different resolutions shown in *Table 1* and to convert from grayscale to Red Green Blue (RGB) format.

2.4 The k-Nearest Neighbour Algorithm

The k-NN algorithm is based on feature similarity: how closely out-of-sample features resemble features in the training samples determines how a given data point is classified. Instead of finding the single closest image in the training set, the top k closest images are found, and they vote on the label of the test image. When k=1, this classification technique is termed the Nearest Neighbour Classifier. Intuitively, higher values of k have a smoothing effect that make the classifier more resistant to outliers [6].

k-NN can be used for classification where the output is a class membership (e.g. malignant or benign). Here an object is classified by a majority vote of its k nearest neighbours [6].

Practically building and applying a k-NN classifier involved two stages;

1. During training, the classifier takes the training data and simply remembers it.
2. During testing, the k-NN classifies every test image by comparing it to all training images and transferring the labels of the k most similar training examples.

The metric that was used to calculate the distance/difference between images was the L2 (Euclidean) distance, i.e. given two images represented by vectors I_1, I_2 , the difference between them is

$$d(I_1, I_2) = \sum_p (I_1^p - I_2^p)^2$$

Some advantages of k-NN models are that very easy to implement and understand and take very little time to train. However, k-NN models incur greater computational cost at test time (or when being practically applied) since classifying an example requires comparison to every single training example. In our use case this meant analysing all 800 training images every time one of the 63 test examples needed classification. Training sets typically range from 2000 to 14,000,000 images without augmentation. This makes applying k-NN to an ideal dataset with a large number of training images in-efficient.

The computational complexity of the Nearest Neighbour classifier is an active area of research, and several Approximate Nearest Neighbour (ANN) algorithms and libraries exist that can accelerate the nearest neighbour lookup in a dataset [7]. These algorithms allow one to trade off the correctness of the nearest neighbour retrieval with its space/time complexity during retrieval, and usually rely on a pre-processing/indexing stage that involves building a k-d tree. A k-d tree is an efficient data structure which can accelerate searches in high-dimensional data (e.g. images).

The value of k is a hyper-parameter. A hyper-parameter is a variable that is an integral part of a classification model and has to be manually selected. Hyper-parameter values can greatly affect the performance of a classification model. Because of this, they need to be carefully selected. One way of selecting hyper-parameters is through cross validation.

Cross-Validation

A training set is usually further split into a validation set and a second test set. This split can then be used to train models while varying hyper-parameters until the best are found. Using these optimal parameters, a final model is then trained and this time it is validated on the first test set.

The idea in cross-validation is that instead of arbitrarily picking a set of datapoints to be the validation set and rest training set, you can get a better and less noisy estimate of how well a certain value of hyper-parameter works by iterating over different validation sets and averaging the performance across these. For example, in 5-fold cross-validation, the training data would be split into 5 equal folds. In the first training iteration, 4 folds would be used for training, and 1 for validation. In subsequent iterations the validation fold is varied until each fold has served as a validation fold one time. In each iteration performance is evaluated and finally the average performance across the different folds is calculated and used to select the best hyper-parameters.

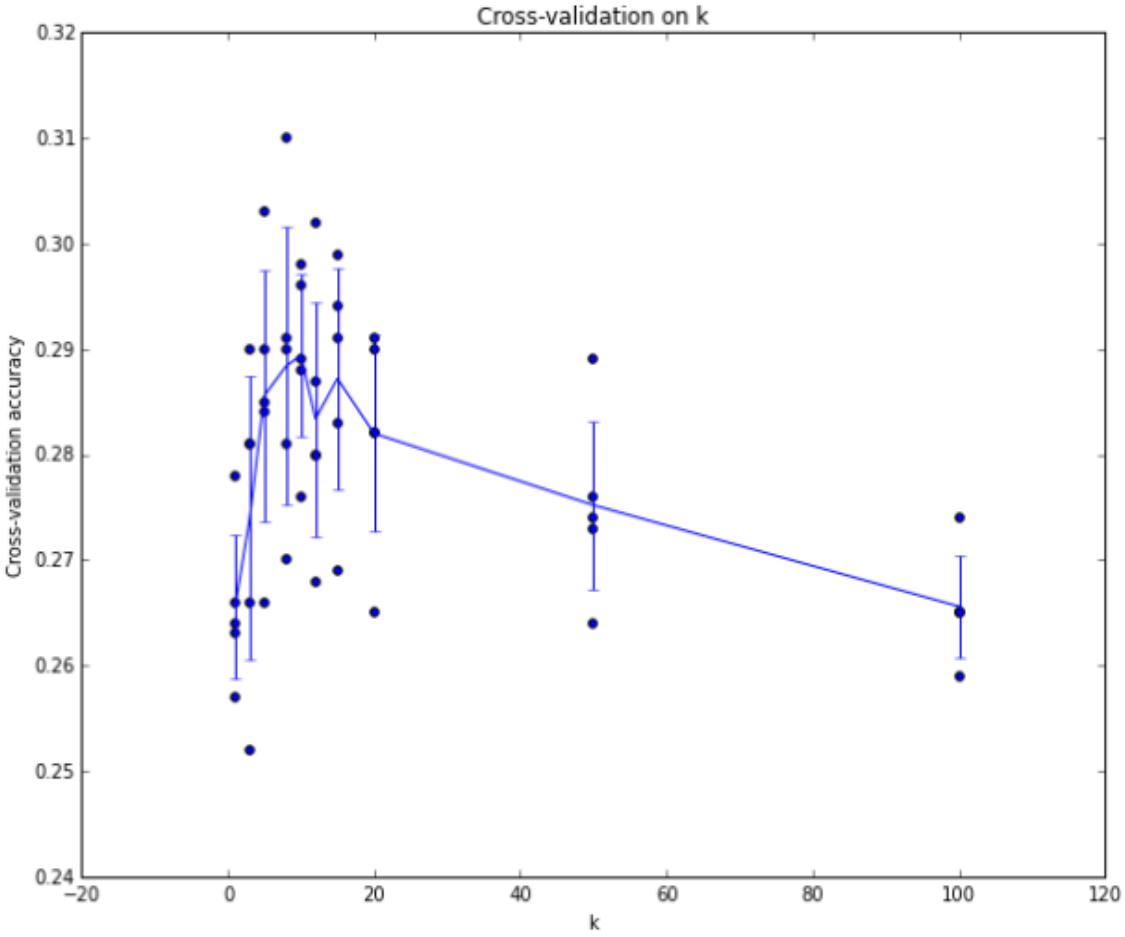


Figure 8: Example of 5-fold cross validation results for determining k

In *Figure 8*, the trend line is drawn through the average performance and the error bars indicate the standard deviation. In this particular case, cross-validation suggests that a value of k = 7 works best on this particular dataset.

In practice, people tend to avoid cross-validation in favour of having a single validation split, since cross-validation can be computationally expensive. The splits tend to use between 50% - 90% of the training data for training and the rest for validation. However, this depends on multiple factors: For example, if the number of hyperparameters is large you may prefer to use bigger validation splits. If the number of examples in the validation set is small (perhaps only a few hundred or so), it is safer to use cross-validation. Typical number of folds you can see in practice would be 3-fold, 5-fold or 10-fold cross-validation [7].

2.5 Artificial Neural Networks

Artificial neural networks are a biologically inspired programming technique which enable computers to learn from observational data [8]. The core computational unit in these networks is an artificial neuron.

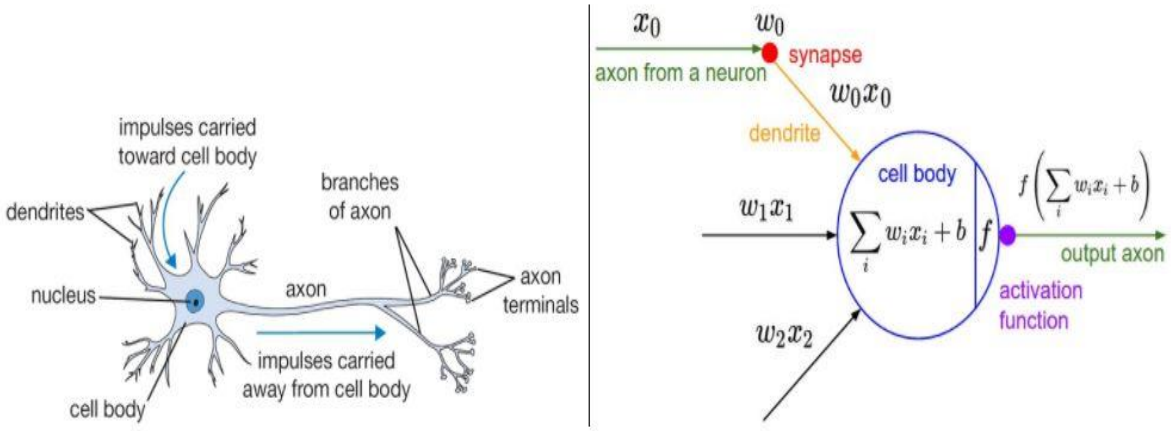


Figure 9: A simplified biological neuron (left) and its mathematical model (right) the artificial neuron. Source: Felipe Perucho licensed under CC-BY 3.0

According to Michael Nielsen's Neural Networks and Deep Learning book [8]:

> The basic computational unit of the brain is a neuron. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14} to 10^{15} synapses. Each neuron receives input signals from its dendrites and produces output signals along its (single) axon. The axon eventually branches out and connects via synapses to dendrites of other neurons. In the computational model of a neuron, the signals that travel along the axons (e.g. x_0) interact multiplicatively (e.g. w_0x_0) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g. w_0). The idea is that the synaptic strengths (the weights w) are learnable and control the strength of influence (and its direction: excitory (positive weight) or inhibitory (negative weight)) of one neuron on another. In the basic model, the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon. In the computational model, we assume that the precise timings of the spikes do not matter, and that only the frequency of the firing communicates information. Based on this rate code interpretation, we model the firing rate of the neuron with an activation function f, which represents the frequency of the spikes along the axon. Historically, a common choice of activation function is the sigmoid function σ, since it takes a real-valued input (the signal strength after the sum) and squashes it to range between 0 and 1

Aside from the sigmoid function other non-linearities include *tanh*, *Rectified Linear Unit (ReLU)*, *maxout* etc.

Artificial neurons are arranged to make a neural network to achieve a specific computational goal. The most common arrangement involves organizing the neurons (units) into distinct layers. For regular neural networks the most common layer type is the fully-connected layer

in which neurons between two adjacent layers are fully pairwise connected but neurons within a single layer share no connections [8].

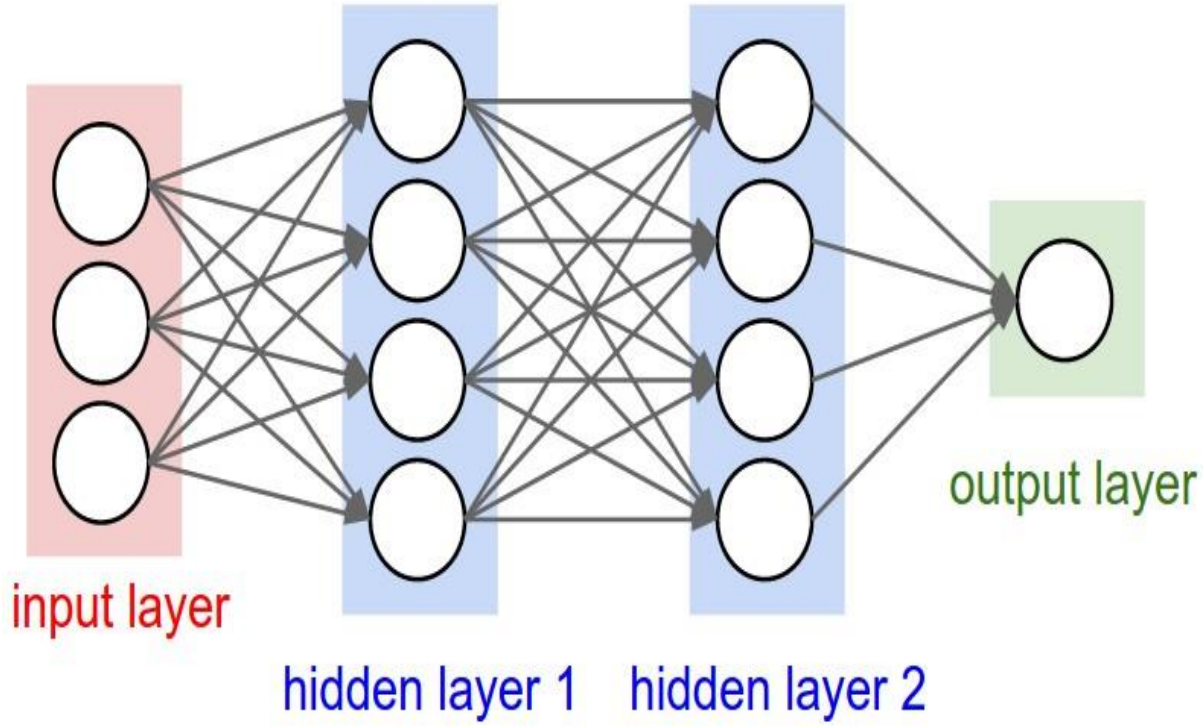


Figure 10: A 3-layer fully connected neural network with three inputs, two hidden layers of 4 neurons each and one output layer.

With a network architecture in place, the weights of the connections need to determined. Initial values of weights can be set by selecting from a specific set, such as a small set of random numbers. For networks using the *ReLU* non-linearity, the current recommendation is to initialize the neurons from a selection of small numbers with a variance of $\frac{2.0}{n}$ [9] where n is the number of inputs.

Optimal values of the weights are determined through training the neural network. Training involves giving the network a large number of inputs of known classes, for example images with lesions that are either malignant or benign. The network then classifies the examples and gives them a score for each possible class e.g. [malignant: 3.5, benign: 6.5]. A loss function tells how good the classification score is. An example of a loss function is the cross-entropy loss. Given a training example (x_i, y_i) where x_i is the image to be classified and y_i is the (integer) label. The scores vector is, $s = f(x_i, W)$ where W is the set of weights and f represents the function the model is trying to represent. The cross-entropy loss L_i has the form;

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

The network can then improve itself by using an optimizer function (such as *Stochastic Gradient Descent (SGD), Adaptive Gradient (Adagrad)* or *Rmsprop*) that adjusts the values of its weights so as to minimize the loss, effectively improving its classification capability. *SGD* adjusts the weights in the negative direction of the loss gradient, i.e. it adjusts the

weights to values that will reduce the loss. Assuming a vector of parameters x and their gradient dx, a basic *SGD* update has the form;

$$x = x - learning\ rate * dx$$

Where $learning\_rate$ is a hyper-parameter, much like the value of k in the k-NN algorithm. *Adagrad* and *Rmsprop* are more effective optimizer functions than plain *SGD* [10], [11]. *SGD* tunes the leaning rate globally for all weights while both *Adagrad* and *Rmsprop* adaptively tune the learning rate. This adaptive tuning is less computationally expensive.

The cycle of calculating the loss and then updating the weights is repeated until optimal weights are found. This is how a neural network learns. Neural network frameworks like *TensorFlow and Keras* help simplify this process.

Neural networks can be thought of as models that try to describe a certain dataset. In *Figure 11* below, an example training dataset is represented by the red and green points. This is a binary classification problem and the three images represent three models aiming to classify the data into two regions.

Figure 11: Model fitting: from left; Underfit, good fit and overfit

The network on the left in *Figure 11* incorrectly classifies the most examples (it gets 5 wrong). The right most network fits all the training data but at the cost of excessive segmentation of the space into many disjoint decision regions. The middle model classifies the data in broad strokes. It treats the few red points in the green space as outliers. In practice this can lead to better generalization on the test set. The right most model is said to have overfit. Overfitting is when a model describes random error or noise instead of the underlying relationships in the data it is being trained on.

Overfitting can be controlled using several techniques such as *L1* and *L2* regularization. In *L1* regularization, for each weight, *w*, the term $\lambda|w|$ is added to it and in L2 regularization, $\frac{1}{2}\lambda w^2$ is added. λ is a hyper-parameter called the regularization strength. *L1* and *L2* regularization techniques have the effect of penalizing certain weights causing the network to prefer certain weights over others helping with generalization.

Overfitting can also be controlled using *Dropout*. *Dropout* is a recently developed regularization technique that has been shown to be extremely effective [12]. During training, *Dropout* is implemented by only keeping a neuron active with some probability p (which is also a hyper-parameter) otherwise setting it to zero. *Figure 12* illustrates this concept.

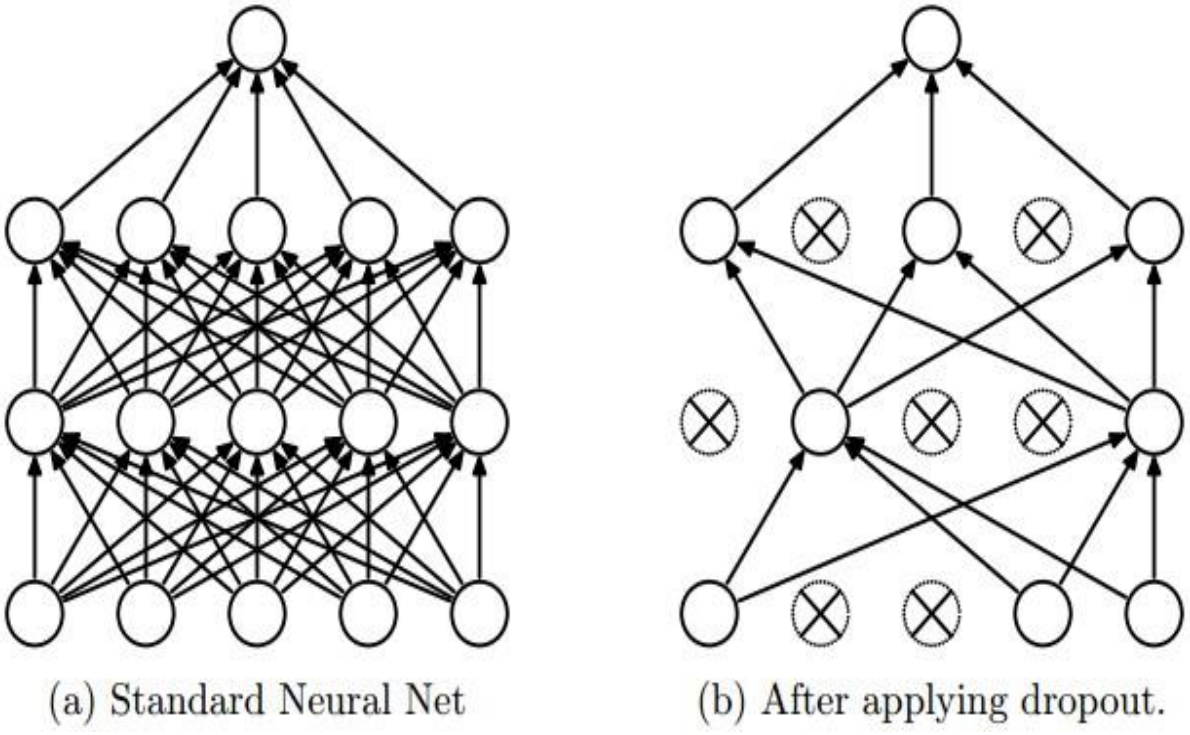


Figure 12: The idea behind dropout. Source; The dropout paper

One interpretation of dropout is that during training, it samples a sub network within the whole network and only updates weights of the sub network based on the input data [12]. During testing there is no dropout applied. The interpretation of this is that the network evaluates an averaged prediction across the exponentially sized ensemble of all sub-networks [12].

2.6 Convolutional Neural Networks (CNNs)

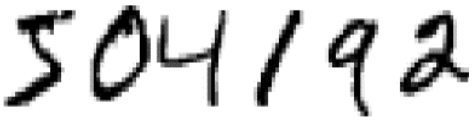


Figure 13: A set of numbers

Most people effortlessly recognize the digits in *Figure 13* as 504192. That ease is deceptive. Our brains utilize a network of at least 140 million neurons, with tens of billions of connections between them to recognize, interpret and understand that image [8]. But nearly all that work is done unconsciously. And so, we don't usually appreciate how tough a problem our visual systems solve.

As soon as it was possible to scan and load medical images into a computer, researchers have built systems for automated analysis [13]. These systems have been traditionally built using conventional computer programming techniques.

In the conventional approach to computer programming, we tell the computer what to do, breaking big problems up into many small, precisely defined tasks that the computer can easily perform. On the other hand, in a neural network we don't explicitly tell the computer how to solve our problem. Instead, it learns from observational data, effectively figuring out its own solution to the problem at hand [8].

CNNs are a type of artificial neural network in which the connectivity pattern between the artificial neurons is inspired by the organization of an animal's visual cortex. CNNs are particularly suited to image recognition because of the sheer complexity of visual data that makes it difficult to build accurate analysis systems using conventional programming techniques.

A CNN takes an image and passes it through a series of convolutional, non-linear, pooling (down sampling) and fully connected layers to get an output. This output can be a single class (e.g. malignant (1) or benign (0)) or a probability of classes.

Figure 14 shows a visualisation of a convolutional layer.

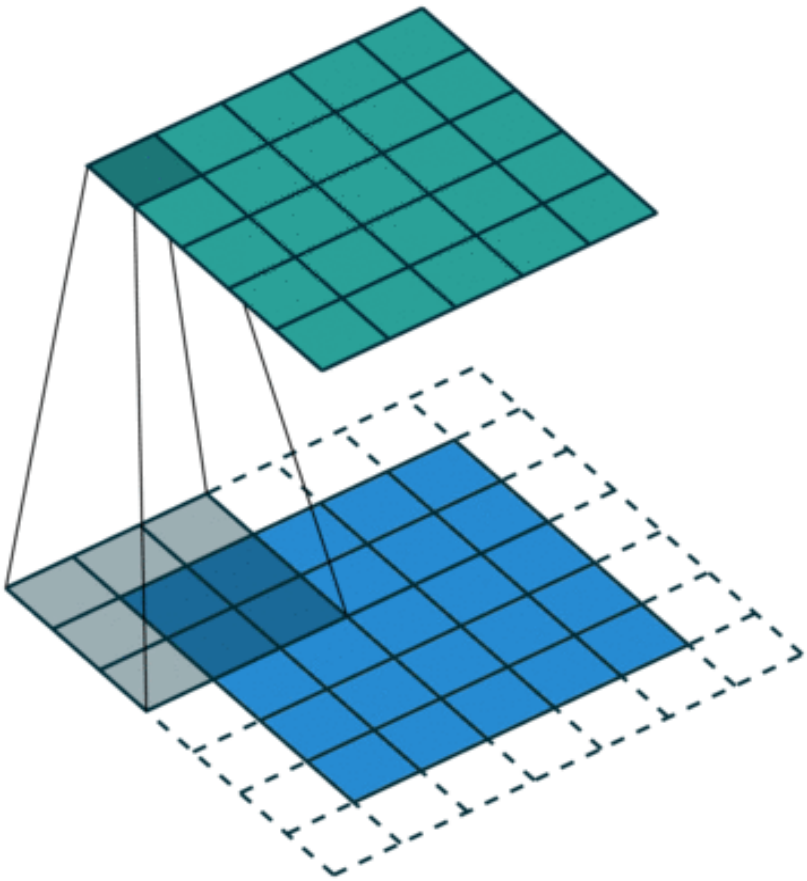


Figure 14: Convolution operation visualized

In *Figure 14*, The 3x3 filter (convolution kernel) is slid over the width and height of the input image (in this case a 5x5 image) and each time a dot product is computed (convolved) between the entries of the filter and the input at that position. The image is padded with zeros (zero padding) to transform it from 5x5 to 7x7 so that the filter covers every pixel evenly. The CNN is trained much like a conventional neural network, and after a while the filter weights have values such that the whole 3x3 filter is essentially a data structure capable of recognising a certain feature or aspect of the dataset on which the CNN has been trained. A CNN has very many of these filters so it can learn a lot of details about the problem its being trained on.

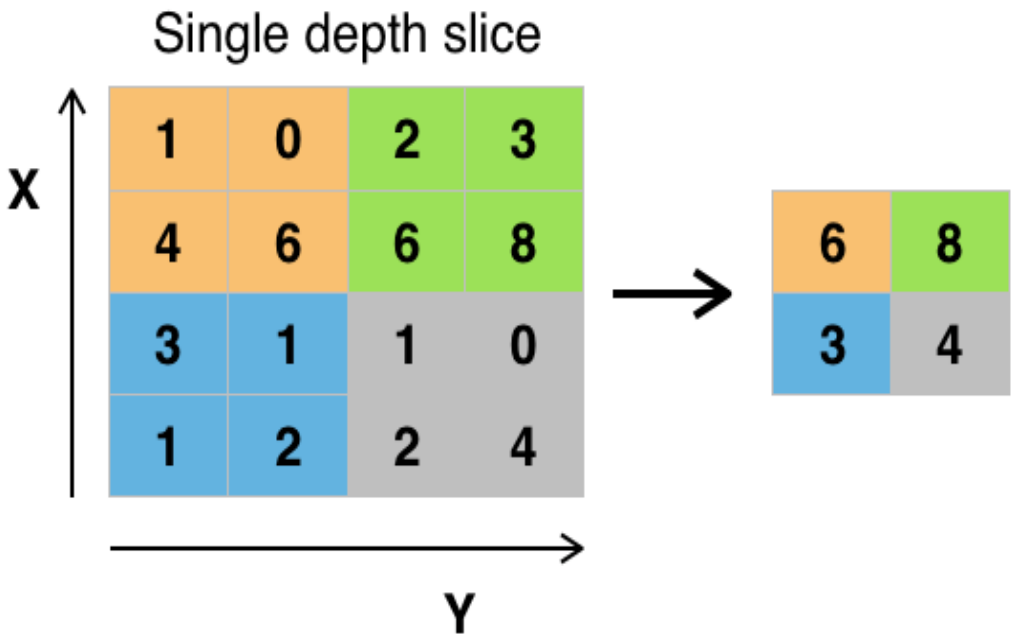


Figure 15: Max Pooling operation of size 2 visualized

Pooling reduces the total amount of parameters (helping with computational efficiency) and helps controls overfitting.

Convolutional networks typically have sparse connectivity (sparse weights). This is accomplished by making the convolution kernel smaller than the input. For example, when processing an image, the input image might have thousands or millions of pixels, but small meaningful features such as edges can be detected with kernels that occupy only tens or hundreds of pixels. This means that fewer parameters need to be stored, which both reduces the memory requirements of a model and improves its statistical efficiency [14]. *Figure 16* shows sparse connectivity

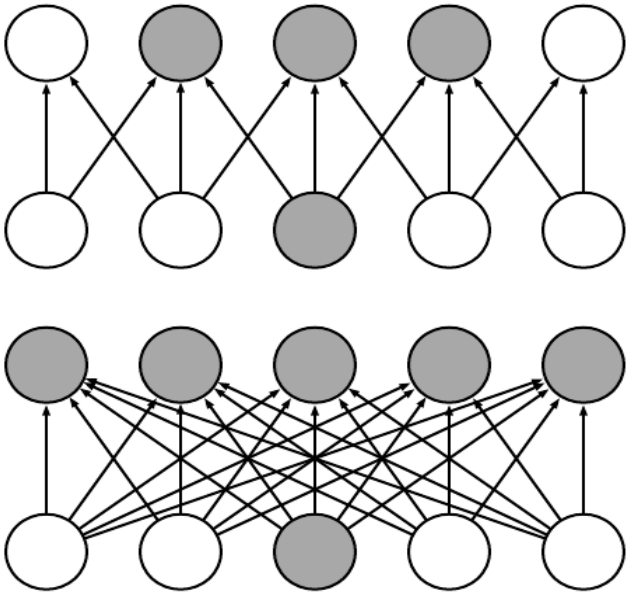


Figure 16: Sparse connectivity visualized

Another tenet of CNNs that makes them efficient is parameter sharing. Parameter sharing refers to using the same parameter for more than one function in a model. In a traditional neural net, each element of the weight matrix is used exactly once when computing the output of a layer. It is multiplied by one element of the input and then never revisited [14]. In a convolutional neural net, each member of the kernel is used at every position of the input. The parameter sharing used by the convolution operation means that rather than learning a separate set of parameters for every location, we learn only one set. This does not affect training time but it further reduces the storage requirements of the model. Convolution is thus dramatically more efficient than dense matrix multiplication (of traditional neural networks) in terms of the memory requirements and statistical efficiency [14]. *Figure 17* visualizes parameter sharing

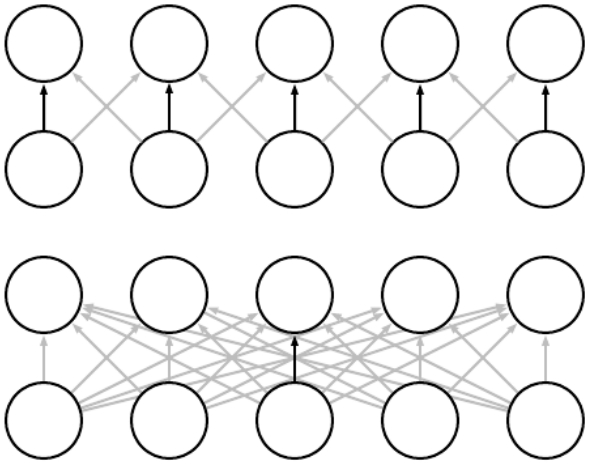


Figure 17: Parameter sharing in CNNs

Successive combinations of Convolution, Activation, Convolution, Activation and Pooling layers followed by a fully connected layer constitute a Convolutional Neural Network. One such network that can be used to classify vehicles is shown in *Figure 18*.

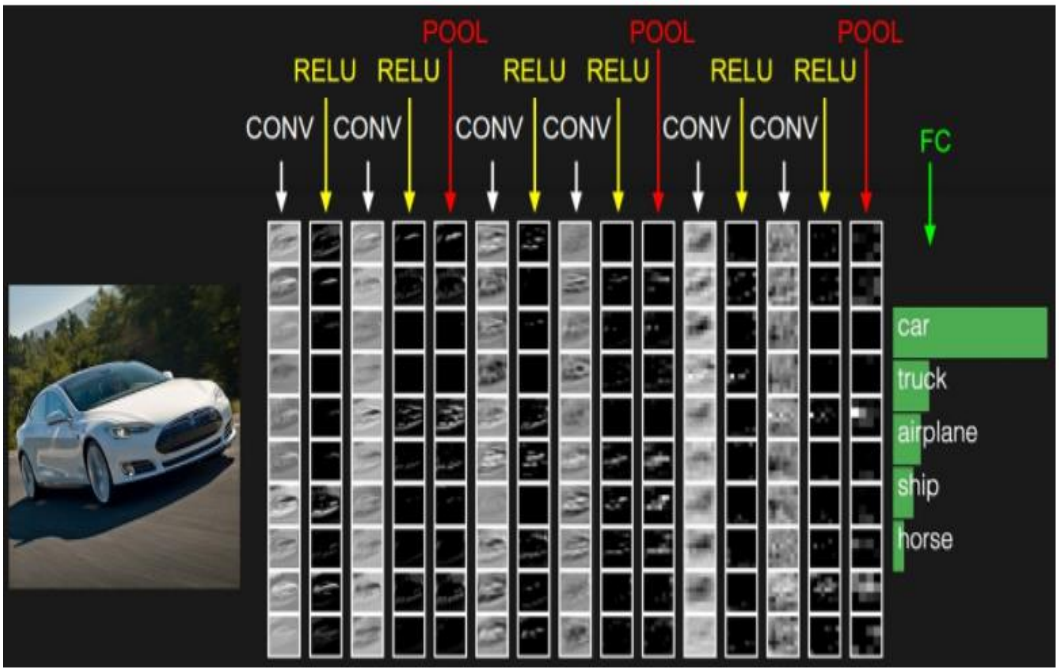


Figure 18: Visualisation of a CNN doing vehicle classification

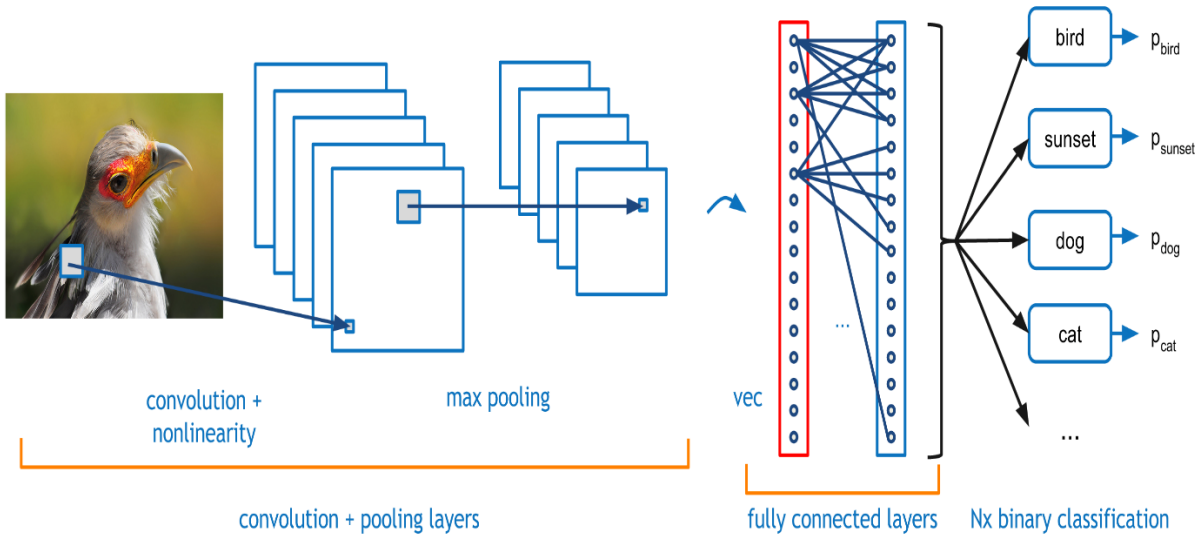


Figure 19: Visualisation of a CNN doing animal classification

2.7 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. It is an optimization that allows rapid progress or improved performance when modelling the second task. Transfer learning is related to problems such as multi-task learning and concept drift and is not exclusively an area of study for machine learning [15].

In transfer learning, we first train a base network on a base dataset and task, and then repurpose the learned features, or transfer them to, a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task [16].

The form of transfer learning used in deep learning is called inductive transfer. This is where the scope of possible models (model bias) is narrowed in a beneficial way by using a model fit on a different but related task. *Figure 20* illustrates this concept.

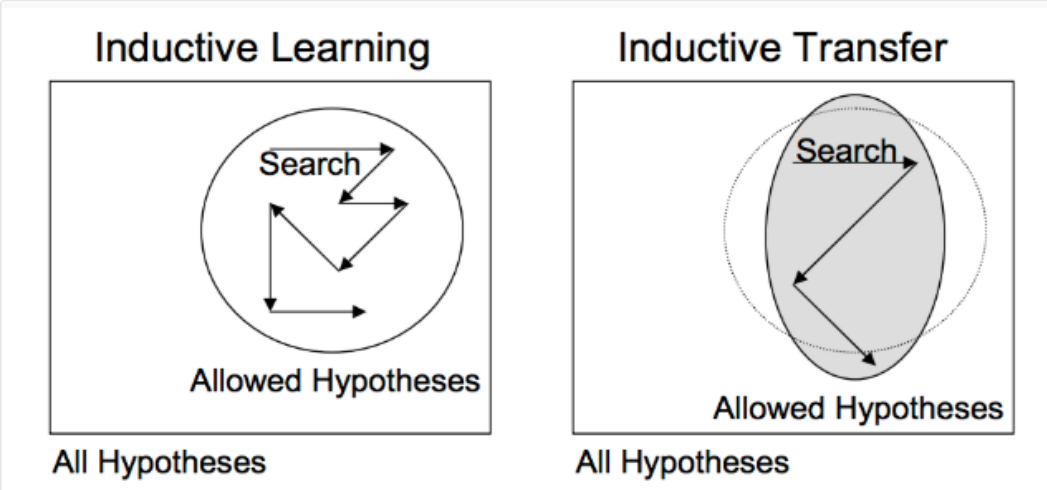


Figure 20: Inductive learning

Applying transfer learning involves selecting a high performing CNN model trained on natural images and then fine tuning it on a different dataset [17]. Transfer learning has been widely used recently in biomedical research [18], [17], [13].

Common image classification models that are used in transfer learning include AlexNet [13], [19], Oxford Visual Graphics Group Net (VGG Net) [20] and GoogLeNet [21], [22].

Figure 21 shows the AlexNet architecture as visualized by its creators in their paper, “Imagenet classification with deep convolutional neural networks” [19]

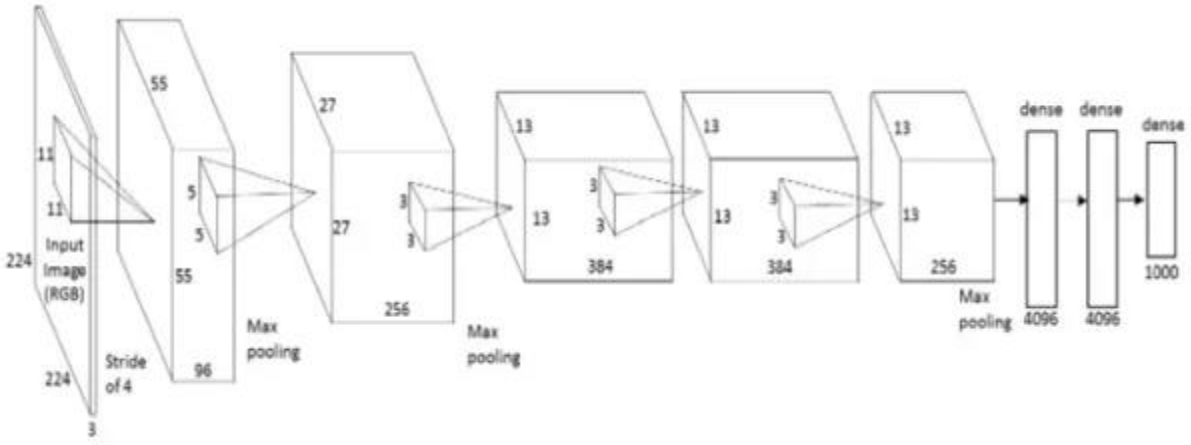


Figure 21: AlexNet architecture

Figure 22 shows the VGG architecture.

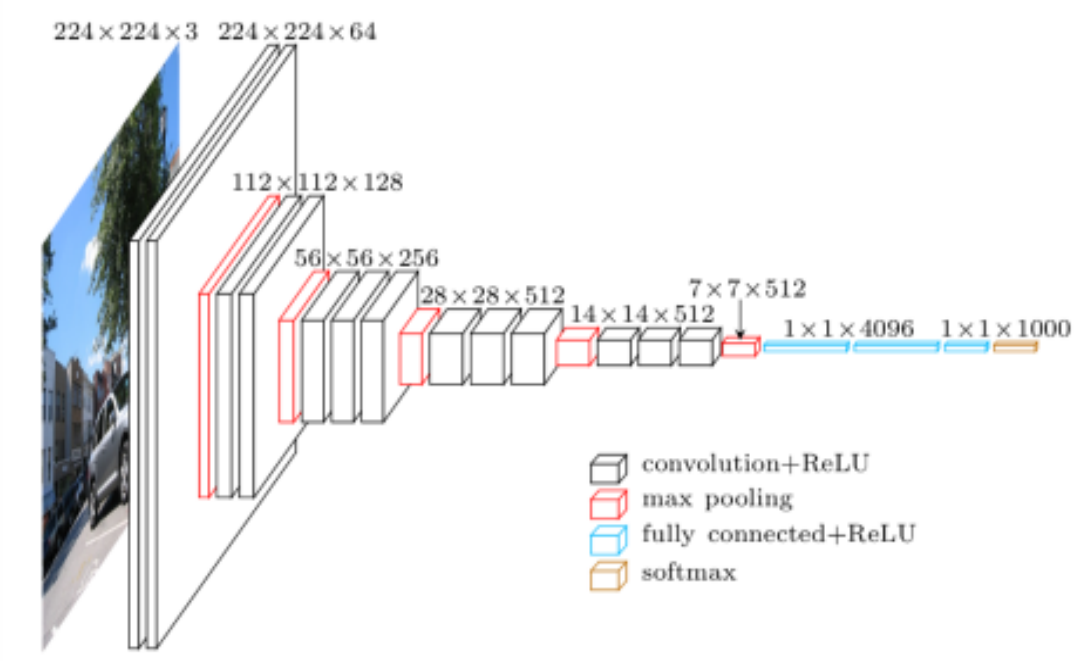


Figure 22: VGG net general architecture

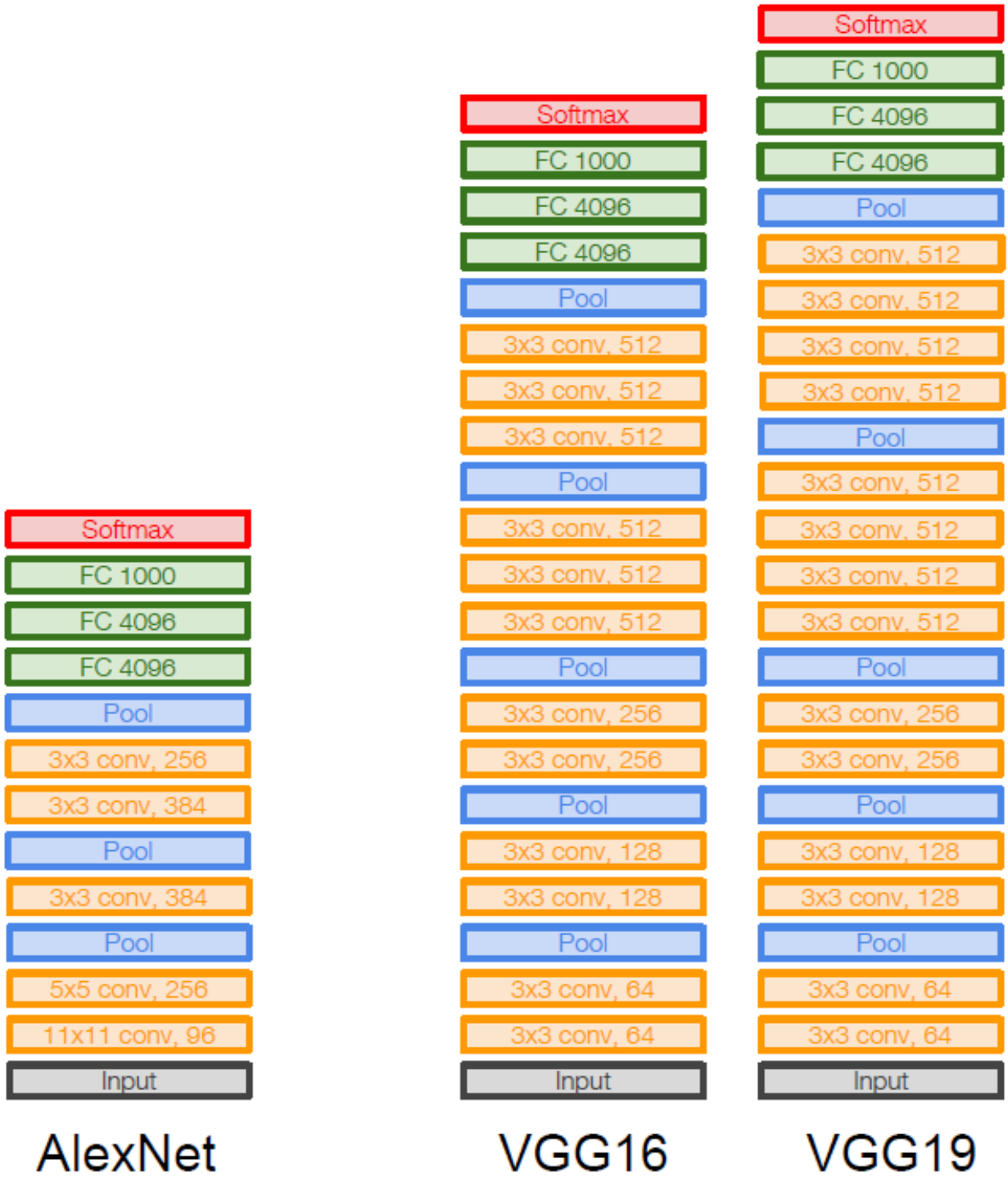


Figure 23: AlexNet and VGG architecture simplified to show layers

Figure 23 shows the layer structure of AlexNet, VGG16 (the original VGG) and VGG19 (a more recently updated variant of VGG). GoogLeNet takes a different approach to structuring its layers. GoogLeNet was designed by researchers at Google and won the 2014 ImageNet competition [22], [21].

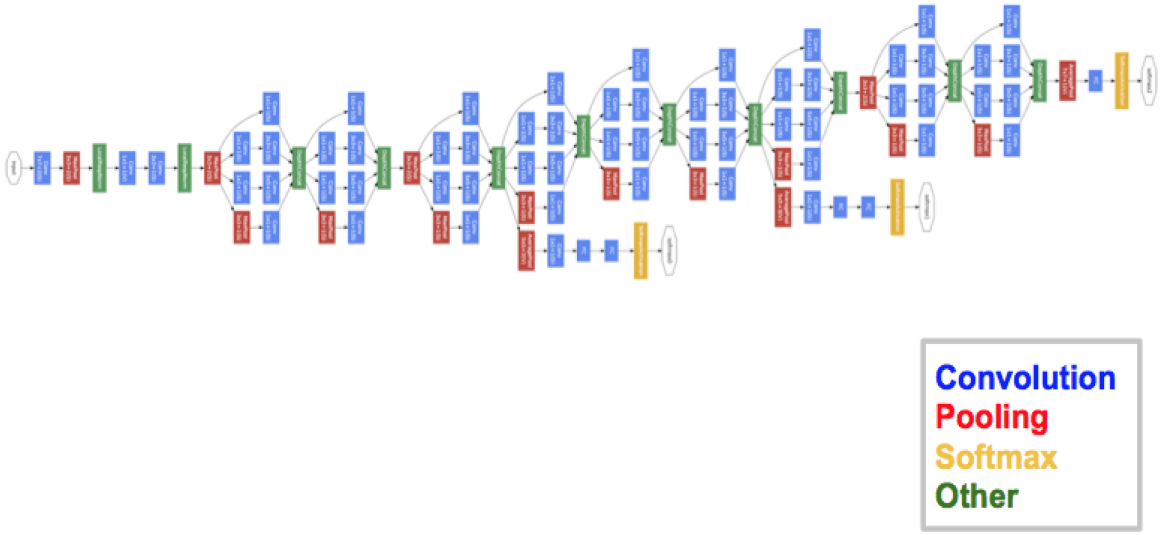


Figure 24: GoogLeNet a.k.a Inception. Source: Inception paper

Figure 24 shows GoogLeNet which has 22 layers. Until GoogLeNet, network layers were sequentially arranged as in *Figure 23* with one segment per layer. GoogLeNet introduced the concept multiple segments per layer. They call these compound layers "Inception" layers. The effect of this is that the network can choose during training whether to convolve an input or to pool it directly [21]. These architectural choices make Inception net train faster than AlexNet and VGG. The size of a pre-trained Inception model is also smaller than a VGG net model. Inception also uses less computational resources than both AlexNet and VGG (2.5 Gigaflops compared to 3 and 32 Gigaflops for AlexNet and VGG respectively) [23]. Overall, Inception net is the most efficient neural network [23].

Inceptions main disadvantage is that in terms of raw recognition accuracy, architectures like VGG Net and ResNet from Microsoft [24] achieve slightly higher accuracy in recognition. Newer versions of Inception such as the InceptionV3 (Version 3) model that we used have alleviated these concerns.

2.8 General Terminology for Results

- TN is the True Negative. These are images correctly classified as not having malignant lesions.
- FP is the False Positive. These are images incorrectly classified as having malignant lesions.
- FN is the False Negatives. These are images incorrectly classified as not having malignant lesions.
- TP is the True Positive. These are images correctly classified as having malignant lesions.

Sensitivity is a function of False Negatives and True Positives. Because we are dealing with cancer, False Negatives should be very low and True Positives high. Low number of False Negatives and a high number of True Positives gives a high sensitivity. A high sensitivity is desired in a good model

$$Sensitivity\ (True\ Positive\ Rate) = \frac{TP}{TP + FN}$$

$$False\ Positive\ Rate = \frac{FP}{FP + TN}$$

TN, FP, FN and TP values can be tabulated in a confusion matrix. A confusion matrix is a table that is often used to describe the performance of a classification model [25].

ROC Curve

A ROC curve (Receiver Operating Characteristic Curve) is a graph showing the performance of a classification model at all classification thresholds. It plots the True Positive Rate (TPR) and the False Positive Rate (FPR) [26].

A ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. *Figure 25* shows a typical ROC curve.

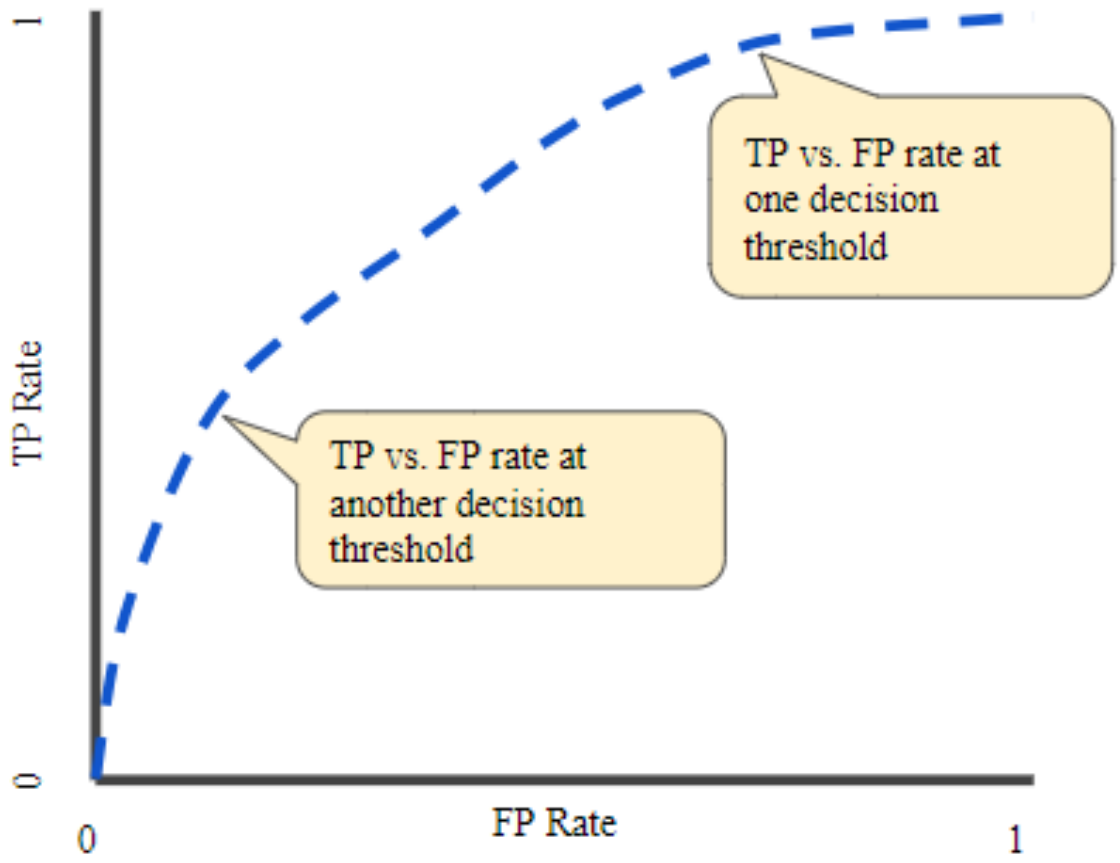


Figure 25: Typical ROC curve

To compute the points in a ROC curve a model could be evaluated many times with different classification thresholds. This would be inefficient. A more efficient way is to find the AUC: Area Under the ROC curve.

AUC: Area Under the ROC Curve

The AUC measures the entire two-dimensional area underneath the ROC curve from (0, 0) to (1, 1) [26]. *Figure 26* shows AUC visualized.

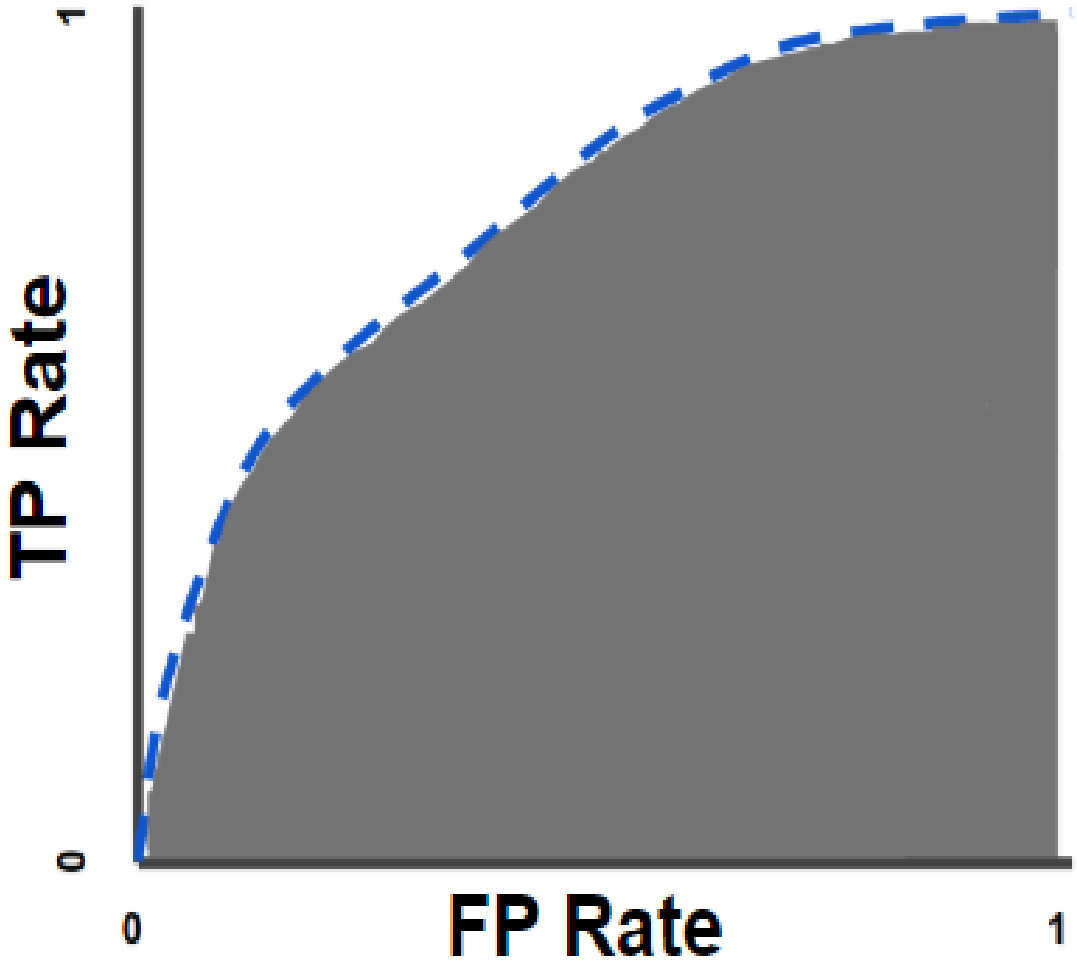


Figure 26: Example AUC

AUC provides an aggregate measure of the performance across all possible classification thresholds [26]. AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0. AUC is classification-threshold-invariant, meaning it measures the quality of a model's predictions irrespective of what classification threshold is chosen.

Classification-threshold invariance is not always desirable. In cases where there are wide disparities in the cost of false negatives vs. false positives, it may be critical to minimize one

type of classification error [26]. For example, when doing email spam detection, you likely want to prioritize minimizing false positives (even if that results in a significant increase of false negatives). When dealing with health-related classification, such as in this project, you want to minimize false negatives and maximize true positives. Because of this deficiency if the AUC metric, we report both sensitivity and AUC.

Chapter Three: Methodology

3.1 Building A Baseline Classifier Using The k-Nearest Neighbour Algorithm

To get the best value of k, multi fold cross validation was used. The 800 training images were split into 5 folds of 160 images each. Choices for k were 1, 3, 5, 8, 10, 12, 15, 20, 50, 100. For each possible value of k out of the 10, the k-NN algorithm is run 5 times, where in each iteration (fold) all but one of the folds is used as training data and the last fold used as validation data. Each time the accuracy of the classifier (number of correct classifications) is recorded.

In total 50 variants of the k-NN were evaluated, 5 for each of the 10 values k.

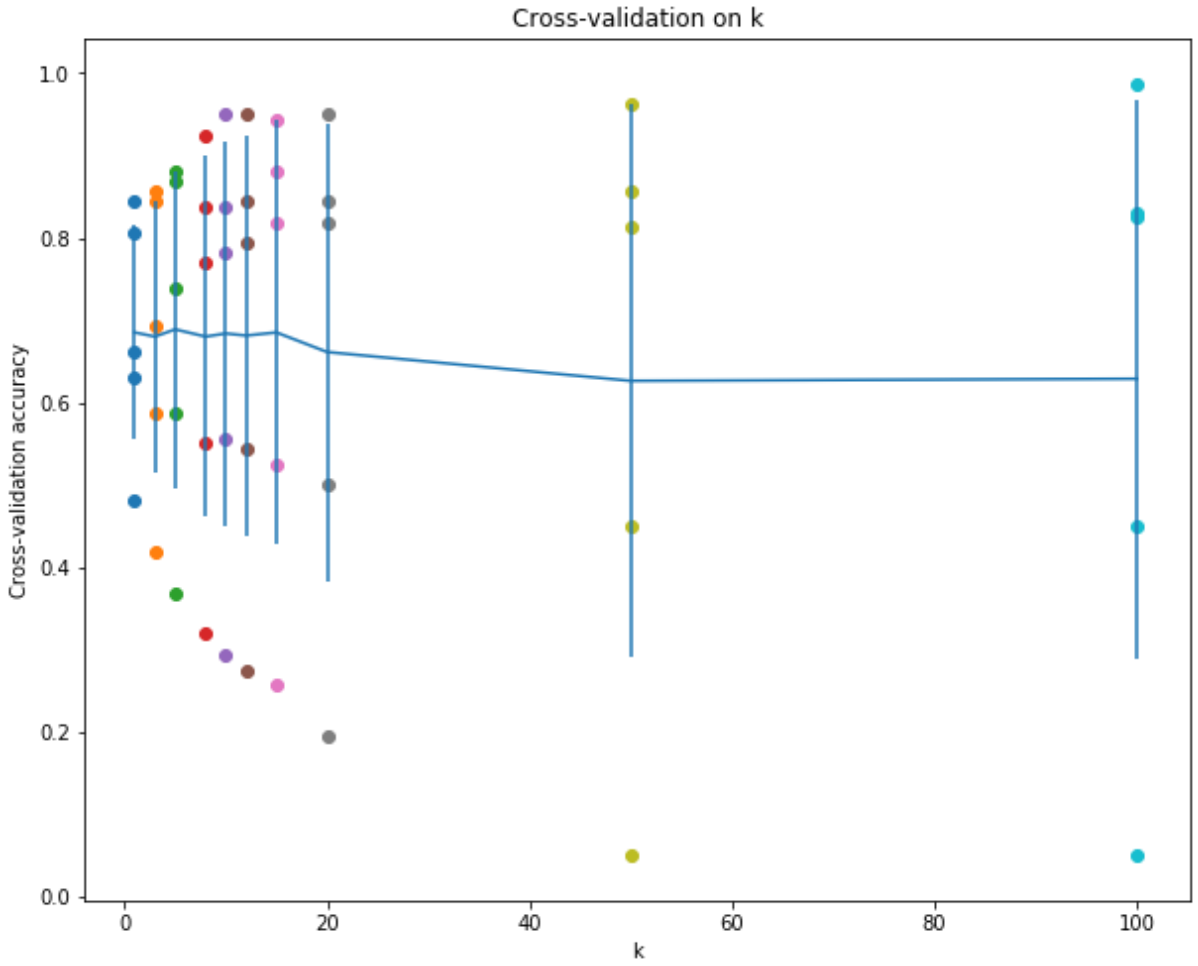


Figure 27: k-NN cross validation results

In *Figure 27*, accuracy is in the y-axis and the x-axis shows the values of k that were tested. Each coloured point is an accuracy result. The vertical bars (error bars) indicate the standard deviation in the five folds and the horizontal trend line is drawn through the average of the results for each k.

Based on the cross-validation results, the best value for k out of the 10 was 3. With k now known, the classifier was retrained using all the 800 images and tested on the 63 testing images using k = 3.

The k-NN classifier was implemented using *Python* and the algorithm used the 528X360X1 image configuration.

3.2 Building A Fully Connected Neural Network for Lesion Classification

Using *Python*, *Keras* and *TensorFlow*, a fully connected neural network was constructed. The network had 3 layers with 256 units per hidden layer and one output layer. Each hidden unit used the *ReLU* activation function and the output layer unit used the sigmoid activation function. The network input images were in the 224X224X3 configuration. This means the input layer had 150528 units. *Dropout* was used for regularization. *Figure 28* shows a summary of our model generated using the *Keras* model method *model.summary()*

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 256)               38535424
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
=================================================================
Total params: 38,601,473
Trainable params: 38,601,473
Non-trainable params: 0
_________________________________________________________________
```

Figure 28: Fully-connected network model summary

The network was trained using the *Adagrad* optimizer. *Figure 29* and *30* show performance during training.

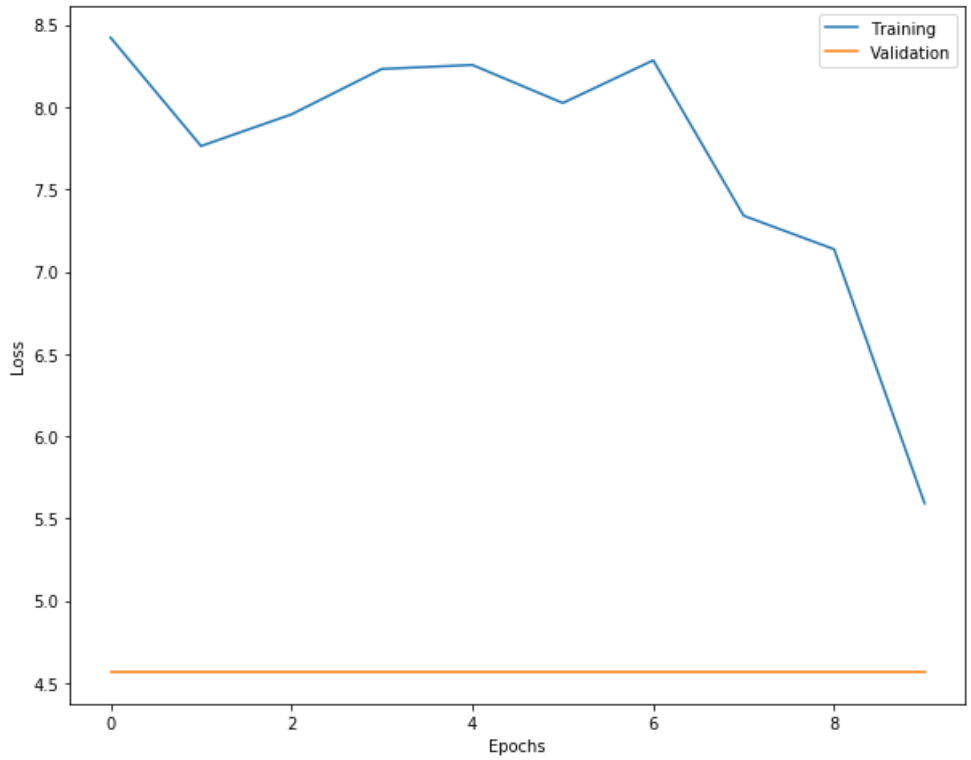


Figure 29: Training loss over 10 training epochs

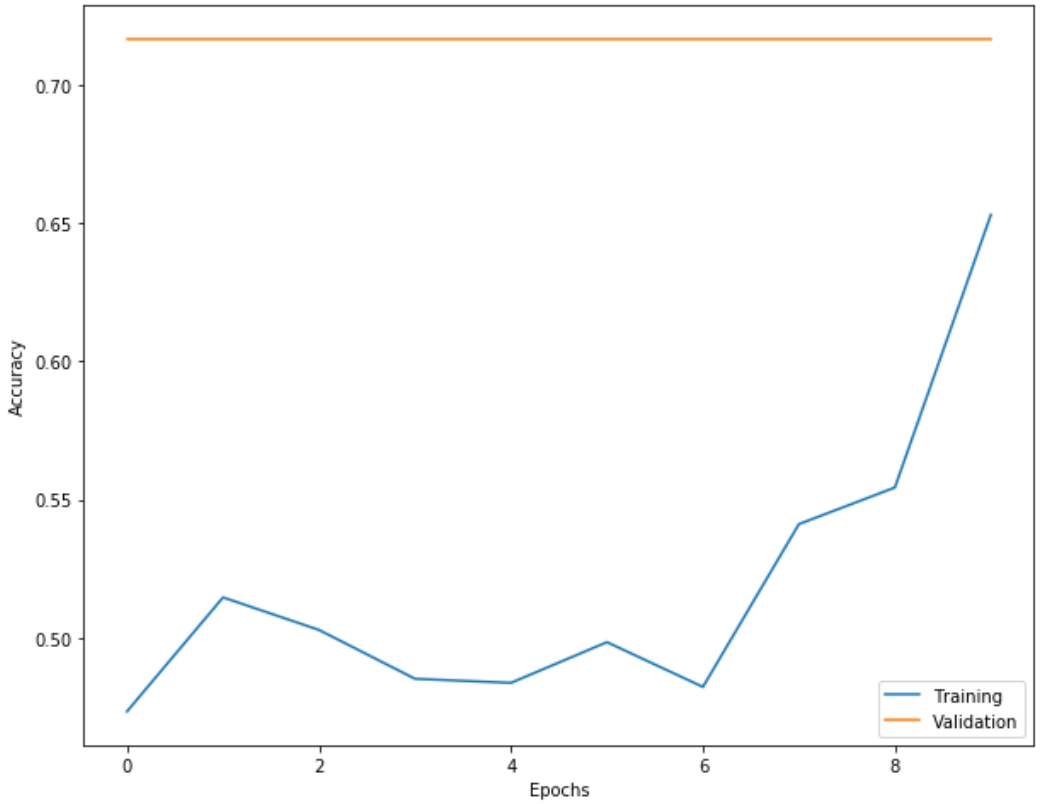


Figure 30: Accuracy over 10 epochs

Different fully connected network configurations were also evaluated before settling on the one described in this report. For network depth we evaluated 2 to 6 hidden layers with 3 having the best performance. For the number of hidden units per layer the range from 64 to 512 was evaluated with 256 found to be optimal. For input configurations all the configurations in *Table 1* were evaluated with the 224X224X3 (150528) input configuration having the best performance.

3.3 Building A Custom Convolutional Neural Network for Lesion Classification

A CNN was constructed using *Python*, *TensorFlow* and *Keras*. Of the 800 training images, 528 images were used for training. This was because the malignant cases were less than the benign cases so a number of them were not utilized in order to balance out the dataset and improve the CNNs generalization.

All input configurations in *Table 1* were tested with the 224x224x3 configuration being the most optimal in terms of accuracy and computational resources. Input is normalized before being passed through three convolutional layers, each of which is followed by the *ReLU* non-linearity (activation function) and a max pooling layer of size 2.

The first two convolutional layers each have 32 filters and the last convolution layer has 64 filters. 32 filters at the start of the network force it to narrow in on lesions and neglect non-distinguishing features of the ultrasound images such as the common background that they all have. After the two layers of 32 filters the 64 filters can now learn finer distinguishing features of lesions. Filters kernels were of size 3x3 with zero padding.

The three convolution layers are then followed by a fully connected layer with 64 units (using the *ReLU* activation function), a dropout layer and finally an output layer with a single unit using the *sigmoid* activation function.

Training was done using the *Adagrad* optimizer for 50 epochs. Different filter sizes and network configurations were evaluated before choosing the one described. *Figure 31* shows a summary of the constructed CNN.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 222, 222, 32)	896
activation_1 (Activation)	(None, 222, 222, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 111, 111, 32)	0
conv2d_2 (Conv2D)	(None, 109, 109, 32)	9248
activation_2 (Activation)	(None, 109, 109, 32)	0
max_pooling2d_2 (MaxPooling2	(None, 54, 54, 32)	0
conv2d_3 (Conv2D)	(None, 52, 52, 64)	18496
activation_3 (Activation)	(None, 52, 52, 64)	0
max_pooling2d_3 (MaxPooling2	(None, 26, 26, 64)	0
flatten_1 (Flatten)	(None, 43264)	0
dense_1 (Dense)	(None, 64)	2768960
activation_4 (Activation)	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65
activation_5 (Activation)	(None, 1)	0

Total params: 2,797,665
Trainable params: 2,797,665
Non-trainable params: 0

Figure 31: The best performing CNN

3.4 Building Transfer Learning Based Models Using InceptionV3 For Lesion Classification

Once again *Python*, *Keras* and *TensorFlow* were used for constructing the modified Inception model. *Keras* comes with base implementations of most state of the art image classification models [27]. It also provides pre-trained weights for these models. The weights come from training the models on over 10 million natural images [22].

The first variant of transfer learning that was tried involved freezing all convolutional layers of the base model. Once the base InceptionV3 implementation and weights were loaded (excluding the top layer), a custom fully connected layer was added on top of the base model. This new top section consists of a fully connected layer with 256 units using the *ReLU* activation function followed by an output unit.

Next, all the convolutional InceptionV3 layers are frozen. This is to prevent relearning of filters. Freezing these layers enables the transfer learning. By freezing the layers, the classification capability from the frozen weights and filters learned from millions of natural images can now be transferred for use in lesion classification.

batch_normalization_180 (BatchN	(None, 5, 5, 320)	960	conv2d_180[0][0]
activation_182 (Activation)	(None, 5, 5, 384)	0	batch_normalization_182[0][0]
activation_183 (Activation)	(None, 5, 5, 384)	0	batch_normalization_183[0][0]
activation_186 (Activation)	(None, 5, 5, 384)	0	batch_normalization_186[0][0]
activation_187 (Activation)	(None, 5, 5, 384)	0	batch_normalization_187[0][0]
batch_normalization_188 (BatchN	(None, 5, 5, 192)	576	conv2d_188[0][0]
activation_180 (Activation)	(None, 5, 5, 320)	0	batch_normalization_180[0][0]
mixed9_1 (Concatenate)	(None, 5, 5, 768)	0	activation_182[0][0] activation_183[0][0]
concatenate_4 (Concatenate)	(None, 5, 5, 768)	0	activation_186[0][0] activation_187[0][0]
activation_188 (Activation)	(None, 5, 5, 192)	0	batch_normalization_188[0][0]
mixed10 (Concatenate)	(None, 5, 5, 2048)	0	activation_180[0][0] mixed9_1[0][0] concatenate_4[0][0] activation_188[0][0]
global_average_pooling2d_2 (Glo	(None, 2048)	0	mixed10[0][0]
dense_3 (Dense)	(None, 256)	524544	global_average_pooling2d_2[0][0]
dense_4 (Dense)	(None, 1)	257	dense_3[0][0]

Total params: 22,327,585
Trainable params: 524,801
Non-trainable params: 21,802,784

Figure 32: Transfer learning using Keras with frozen convolutional layers

Figure 32 shows a truncated view of this first implementation. A noticeable difference between the architecture shown here and our fully custom implementation shown in *Figure 31* is the total number of parameters. This InceptionV3 based network has 22,327,585 parameters compared to the 2,797,665 parameters shown in *Figure 31*. 21,802,784 of the 22,327,585 total parameters are non-trainable meaning they are frozen as earlier described.

The second variation on transfer learning is to freeze all lower convolutional layers except a few of the top layers. These top convolutional layers, along with a newly added fully connected layers can then be retrained. Such a network was also constructed and trained.

```
conv2d_188 (Conv2D)              (None, 5, 5, 192)    393216      average_pooling2d_18[0][0]
__________________________________________________________________________________________________
batch_normalization_180 (BatchN (None, 5, 5, 320)    960         conv2d_180[0][0]
__________________________________________________________________________________________________
activation_182 (Activation)      (None, 5, 5, 384)    0           batch_normalization_182[0][0]
__________________________________________________________________________________________________
activation_183 (Activation)      (None, 5, 5, 384)    0           batch_normalization_183[0][0]
__________________________________________________________________________________________________
activation_186 (Activation)      (None, 5, 5, 384)    0           batch_normalization_186[0][0]
__________________________________________________________________________________________________
activation_187 (Activation)      (None, 5, 5, 384)    0           batch_normalization_187[0][0]
__________________________________________________________________________________________________
batch_normalization_188 (BatchN (None, 5, 5, 192)    576         conv2d_188[0][0]
__________________________________________________________________________________________________
activation_180 (Activation)      (None, 5, 5, 320)    0           batch_normalization_180[0][0]
__________________________________________________________________________________________________
mixed9_1 (Concatenate)           (None, 5, 5, 768)    0           activation_182[0][0]
                                                                  activation_183[0][0]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)      (None, 5, 5, 768)    0           activation_186[0][0]
                                                                  activation_187[0][0]
__________________________________________________________________________________________________
activation_188 (Activation)      (None, 5, 5, 192)    0           batch_normalization_188[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)            (None, 5, 5, 2048)   0           activation_180[0][0]
                                                                  mixed9_1[0][0]
                                                                  concatenate_4[0][0]
                                                                  activation_188[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_2 (Glo (None, 2048)         0           mixed10[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                  (None, 256)          524544      global_average_pooling2d_2[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                  (None, 1)            257         dense_3[0][0]
==================================================================================================
Total params: 22,327,585
Trainable params: 11,639,681
Non-trainable params: 10,687,904
```

Figure 33: Transfer learning with some trainable convolution blocks

Figure 33 shows a truncated summary of this implementation. By retraining the top convolution blocks, only 10,687,904 parameters are frozen instead of the 21,802,784 frozen parameters shown in *Figure 32*.

The rationale behind this variant of transfer learning is the relatively large disparity between the natural images that the base InceptionV3 model was trained on and the breast ultrasound images that our custom models are trying to classify. Retraining the top two InceptionV3 convolutional layers serves to evaluate performance when filters more specific to ultrasound images are learnt much like in the custom convolutional network implementation in *Chapter 3.3*. This network takes significantly more time to train than the variant in *Figure 32* with all convolutional layers frozen. This is because 11,114,880 more parameters need to be trained.

A third variant of transfer learning is to take a model, like InceptionV3, preserve its architecture but retrain the whole network i.e. have it learn new weights for all its filters. This technique is worth investigating if a study has a large enough dataset. This is because all state of the art networks (including InceptionV3) are trained on over 1 million images [18], [22] therefore relearning their weights using a dataset with a few images will result in sub optimal weights. Due to dataset limitations this third variant of transfer learning was not investigated.

Chapter Four: Results

4.1 k-NN Model

Table 2: Confusion matrix showing k-NN results

n = 63	**Classified: Benign**	**Classified: Malignant**	
Actual: Benign	TN = 42	FP = 0	42
Actual: Malignant	FN = 4	TP = 17	21
	46	17	

Sensitivity = 0.81

4.2 Fully Connected Neural Network Model

Table 3: Confusion matrix showing FCN results

n = 63	**Classified: Benign**	**Classified: Malignant**	
Actual: Benign	TN = 42	FP = 0	42
Actual: Malignant	FN = 21	TP = 0	21
	63	0	

Sensitivity = 0.0

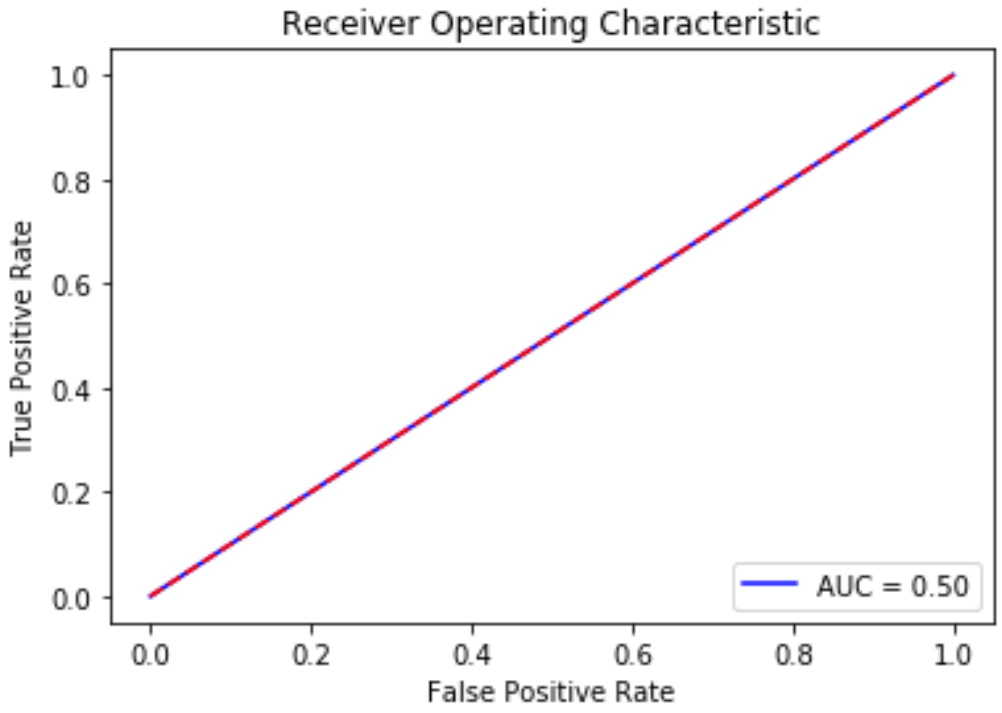


Figure 34: ROC curve for the FCN

4.3 CNN Model

n = 63	**Classified: Benign**	**Classified: Malignant**	
Actual: Benign	TN = 17	FP = 25	42
Actual: Malignant	FN = 3	TP = 18	21
	20	43	

Sensitivity = 0.85

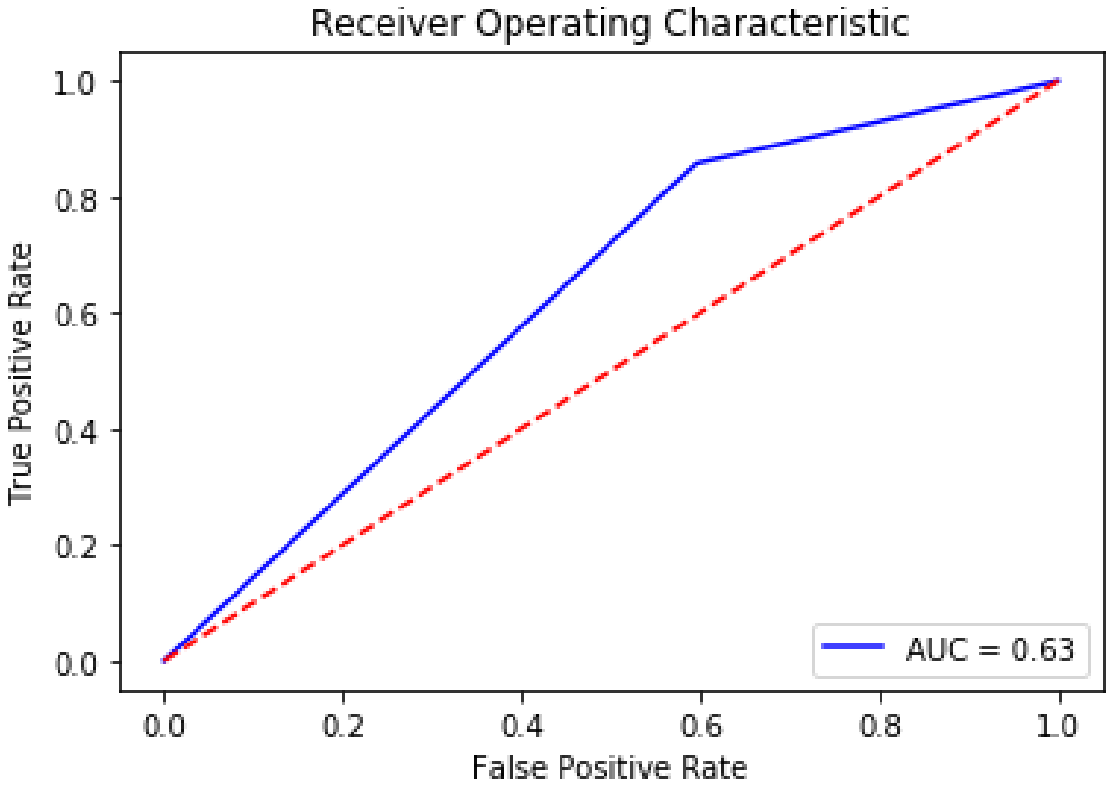


Figure 35: ROC curve for our custom CNN

Different learning algorithms were also evaluated. Overall, we tested *SGD*, *Adam*, *Rmsprop* and *Adagrad*, with *Adagrad* showing the best performance.

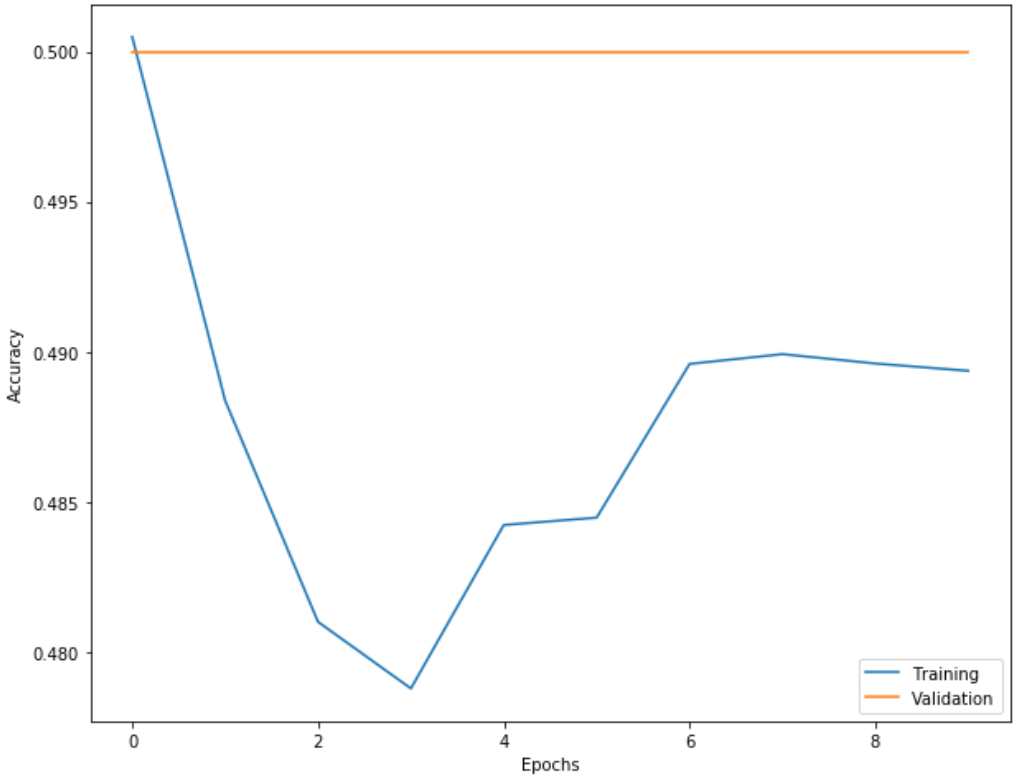


Figure 36: Training with SGD

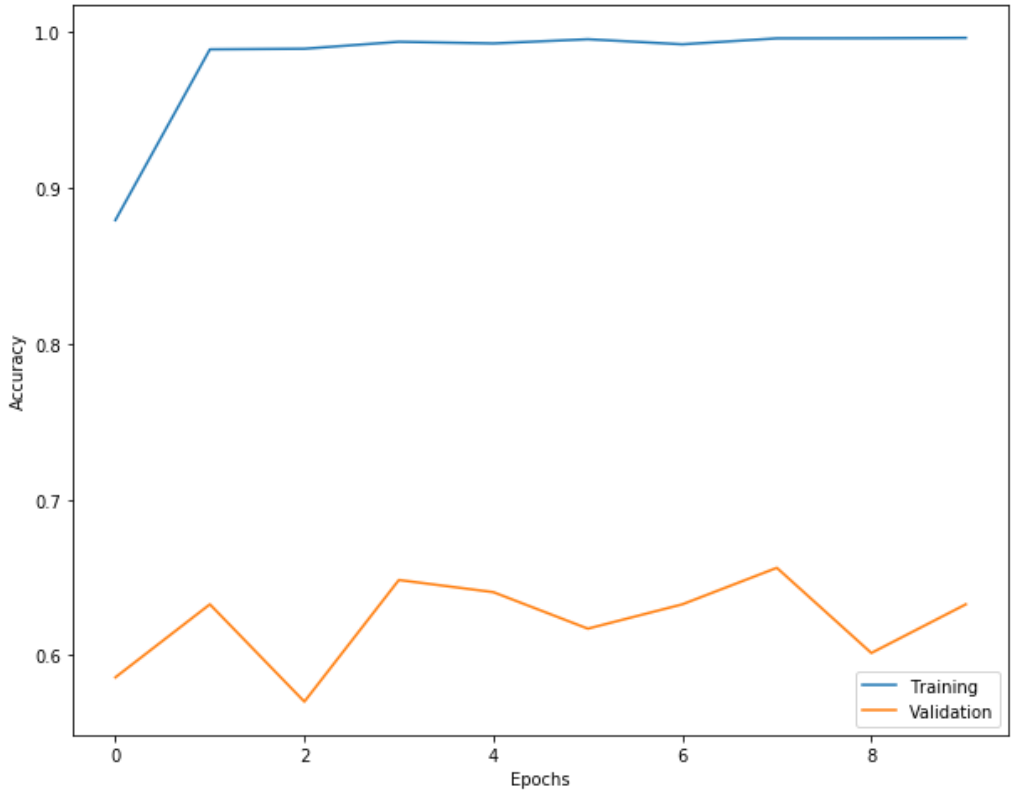


Figure 37: Training with Adam

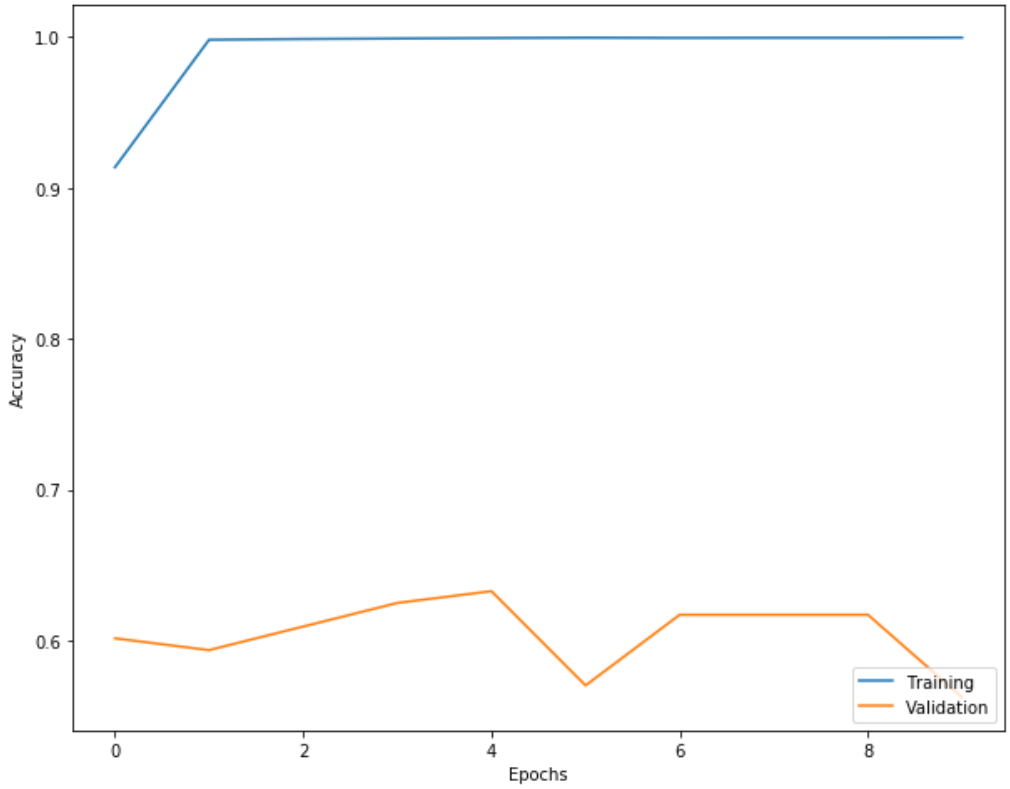


Figure 38: Training with Rmsprop

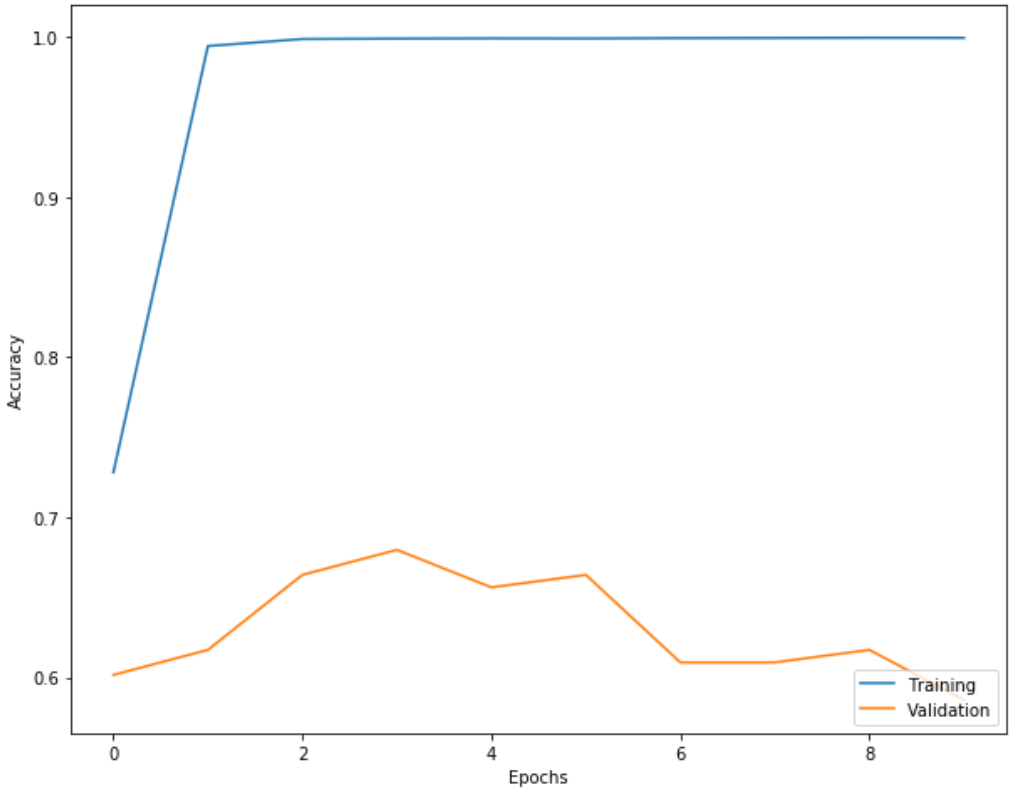


Figure 39: Training with Adagrad

4.4 Transfer Learning Models

Both transfer-learning based models had the same results that are show in *Table 4 and 5.*

Table 4: Confusion matrix showing results for the first variant of transfer learning

n = 63	**Classified: Benign**	**Classified: Malignant**	
Actual: Benign	TN = 0	FP = 42	42
Actual: Malignant	FN = 0	TP = 21	21
	0	63	

Sensitivity = 1.0

Table 5: Variant 2 of transfer learning

n = 63	**Classified: Benign**	**Classified: Malignant**	
Actual: Benign	TN = 0	FP = 42	42
Actual: Malignant	FN = 0	TP = 21	21
	0	63	

Sensitivity = 1.0

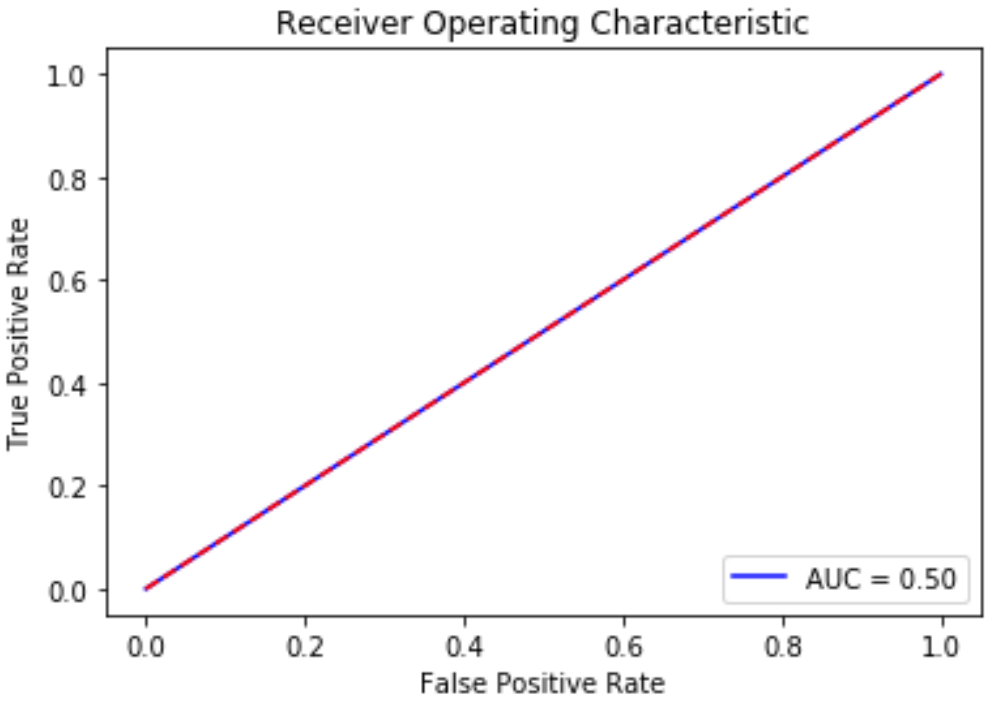


Figure 40: ROC curve for variant 1of transfer learning

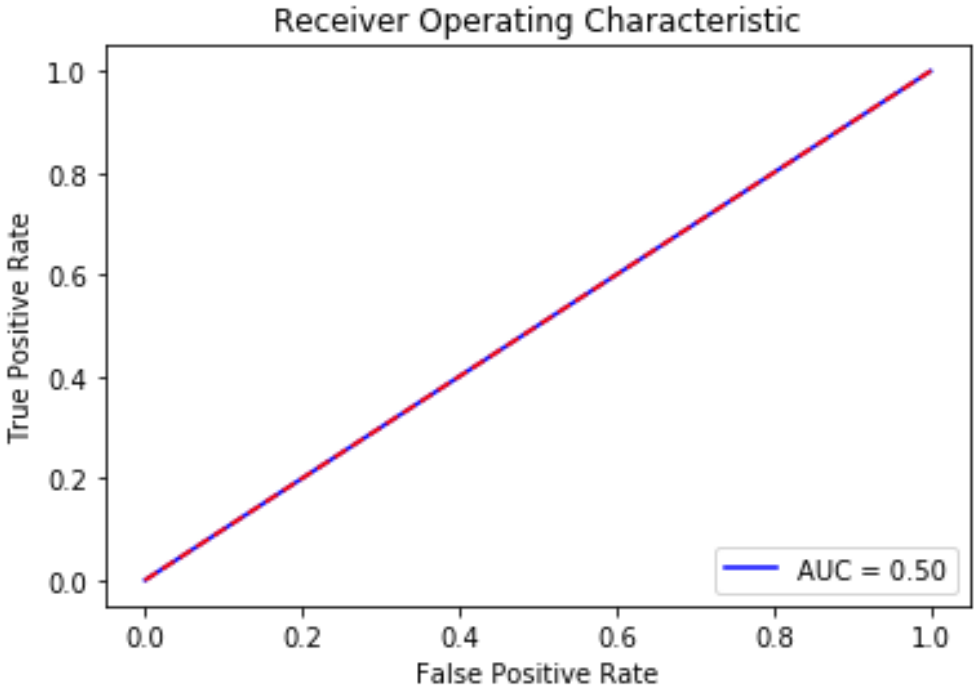


Figure 41: ROC curve for Variant 2 of transfer learning

Chapter Five: Interpretation/Analysis of Results

5.1 k-NN Model

The k-NN classifiers performance was erratic during cross validation as shown in *Figure 27*. This means the algorithms performance was more dependent on the nature of the image configuration used during each training iteration than on a generalisation of features in a malignant or benign lesion. This unstable performance makes the k-NN model unsuitable for real world usage even though it shows good sensitivity. This is because should the model be given images with a higher resolution than those used during training, or images compressed differently then the performance will be different from what we recorded with our test set.

5.2 Fully Connected Network Model

The fully connected network overfit. The network learned to only classify benign lesions. The network overfit even though we balanced out the training set by using an equal number of malignant and benign lesions for training. Altering input image configuration and normalising input also did not improve performance. Generally, it is often the case that 3-layer neural networks will outperform 2-layer nets, but going even deeper (4,5,6 layers) rarely helps. We observed this phenomenon too. This is in contrast to CNNs where depth positively affects classification performance.

5.3 Convolutional Neural Network Model

The custom CNNs performance was better than the FCN and k-NN model. The model performs better than the FCN even though both are data constrained. Sensitivity of 0.85 is comparable to results from similar projects [28], [29] in spite of having access to significantly less data (90% less data). This can be attributed to the use of techniques such as dropout and modern network optimizers. An Area Under Curve (AUC) of 0.63 is low but can be improved with more data added to the training pipeline.

5.4 Transfer Learning Models

Transfer leaning approaches show promise (they had no false negatives) but are hindered by a lack of data. The large visual disparity between breast ultrasounds and the natural images on which InceptionV3 was trained also negatively impact the transfer learning models. The feature extraction benefits that can come from transfer learning were not realised in this case as the InceptionV3 model has a large number of layers and filters, so the relatively few images that were used to train the modified model were insufficient for all these filters to extract many meaningful features from the breast ultrasounds. The second approach of retraining the top convolutional layers also had no positive impact on classification performance. This approach would have had more effect if the training set was larger because more meaningful features would could be learnt by the filters made weightless by unfreezing the top most convolution blocks.

The third variant to transfer learning that involves taking a model, preserving its architecture then retraining the whole network, was not investigated due to dataset limitations. This technique is worth investigating if a study has a large enough dataset.

Chapter Six: Observations, Recommendations and Conclusions

6.1 Observations

Neural network approaches are more efficient than the k-NN algorithm for lesion classification. The FCN is not as sensitive as the k-NN but gives more stable (albeit incorrect) results than the k-NN. CNN approaches, both custom and transfer learning based, show good performance in classification.

6.2 Conclusion

k-NN algorithms are not well suited for lesion classification due to computational inefficiency and classification performance instability. Fully connected networks and transfer learning based networks both require significantly more training examples than the 100 (800 plus augmentation) used in this project if they are to be used to construct accurate models. Overall Convolutional Neural Networks are a viable technique for building models for automated classification of lesions in breast ultrasounds, given enough data.

6.3 Recommendations

More data should be used in future studies to fully investigate the potential of transfer learning for ultrasound lesion classification. A larger dataset will also aid in constructing more accurate CNN models

References

[1] "Uganda has only 37 radiologists – records," *Daily Monitor*. [Online]. Available: http://www.monitor.co.ug/News/National/Uganda-has-only-37-radiologists-records/688334-4309954-v9e8i8/index.html. [Accessed: 13-Jun-2018].

[2] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep Learning for Identifying Metastatic Breast Cancer," *ArXiv160605718 Cs Q-Bio*, Jun. 2016.

[3] C. Spampinato, S. Palazzo, D. Giordano, M. Aldinucci, and R. Leonardi, "Deep learning for automated skeletal bone age assessment in X-ray images," *Med. Image Anal.*, vol. 36, pp. 41–51, 2017.

[4] V. Gulshan *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *Jama*, vol. 316, no. 22, pp. 2402–2410, 2016.

[5] M. H. Yap *et al.*, "Automated breast ultrasound lesions detection using convolutional neural networks," *IEEE J. Biomed. Health Inform.*, 2017.

[6] A. Bronshtein, "A Quick Introduction to K-Nearest Neighbors Algorithm," *Medium*, 11-Apr-2017. .

[7] "CS231n Convolutional Neural Networks for Visual Recognition." [Online]. Available: https://cs231n.github.io/classification/#knn. [Accessed: 13-Jun-2018].

[8] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *ArXiv150201852 Cs*, Feb. 2015.

[10] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *J. Mach. Learn. Res.*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[11] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[13] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, 2017.

[14] I. Goodfellow, Y. Bengio, A. Courville, and F. Bach, *Deep Learning*. The MIT Press, 2016.

[15] J. Brownlee, "A Gentle Introduction to Transfer Learning for Deep Learning," *Machine Learning Mastery*, 20-Dec-2017. .

[16] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *ArXiv14111792 Cs*, Nov. 2014.

[17] N. Tajbakhsh *et al.*, "Convolutional neural networks for medical image analysis: Full training or fine tuning?," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[18] C.-K. Shie, C.-H. Chuang, C.-N. Chou, M.-H. Wu, and E. Y. Chang, "Transfer representation learning for medical image analysis," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, 2015, pp. 711–714.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ArXiv Prepr. ArXiv14091556*, 2014.

[21] C. Szegedy *et al.*, "Going deeper with convolutions," 2015.

[22] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[23] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *ArXiv Prepr. ArXiv160507678*, 2016.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] “Simple guide to confusion matrix terminology,” *Data School*, 26-Mar-2014. [Online]. Available: http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/. [Accessed: 02-Jun-2018].
[26] “Classification: ROC and AUC | Machine Learning Crash Course,” *Google Developers*. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc. [Accessed: 14-Jun-2018].
[27] F. Chollet, *Keras*. 2015.
[28] S. Weng, “Automating Breast Cancer Detection with Deep Learning,” *Insight Data*, 13-Jun-2017. [Online]. Available: https://blog.insightdatascience.com/automating-breast-cancer-detection-with-deep-learning-d8b49da17950. [Accessed: 03-Jun-2018].
[29] B. Huynh, K. Drukker, and M. Giger, “MO-DE-207B-06: Computer-Aided Diagnosis of Breast Ultrasound Images Using Transfer Learning From Deep Convolutional Neural Networks,” *Med. Phys.*, vol. 43, no. 6Part30, pp. 3705–3705.

Appendix 1: Deploying Developed Classification Models for Real World Usage

A web application was built to demonstrate a potential application and deployment scenario for the models that were built during this project.

Figure 42: Mindray DC-7 ultrasound machine

The ultrasound machine show in *Figure 42* is commonly used in health centers in Uganda. The machine has a Universal Serial Bus (USB) port that can be used to transfer ultrasound scans from it. Once transferred this scan can be uploaded to an application, such as a web application, for automated analysis.

Using HTML, CSS, JavaScript and React.js, a web application was built. *Figures 42 to 45* show a potential use case scenario for the application.

1. A radiologist or sonographer would log into the application.

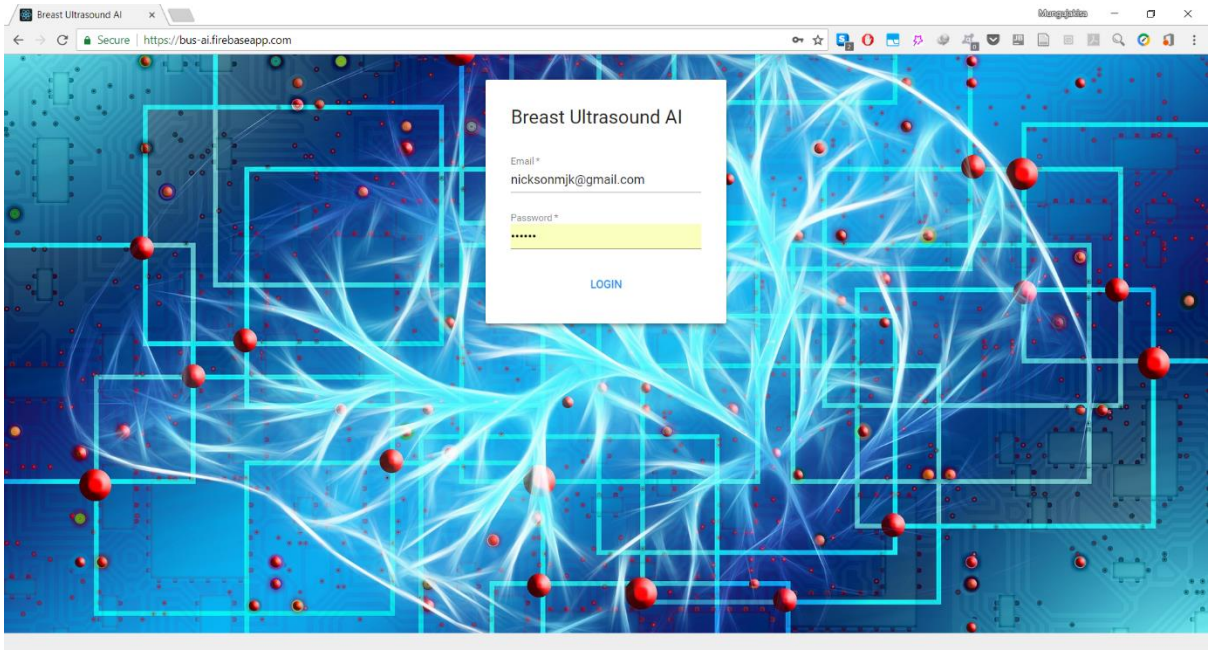


Figure 43: Sign in screen for the app

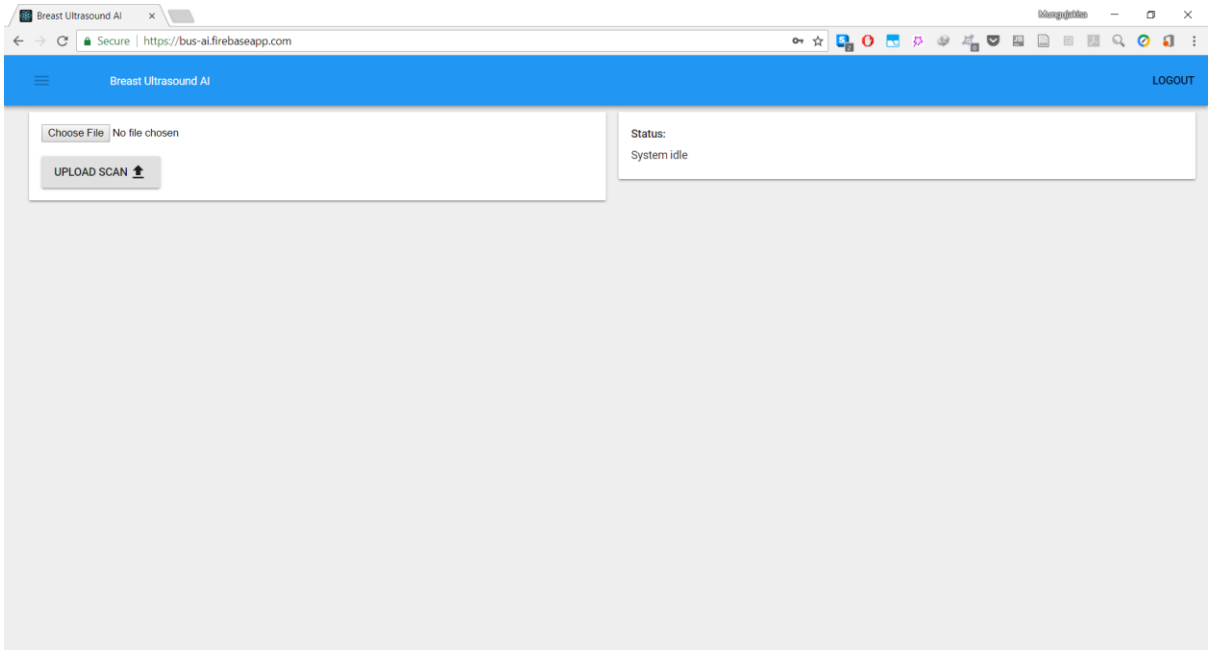


Figure 44: After signing in and before uploading a scan

2. After signing into the application the user uploads a breast ultrasound scan for automated analysis

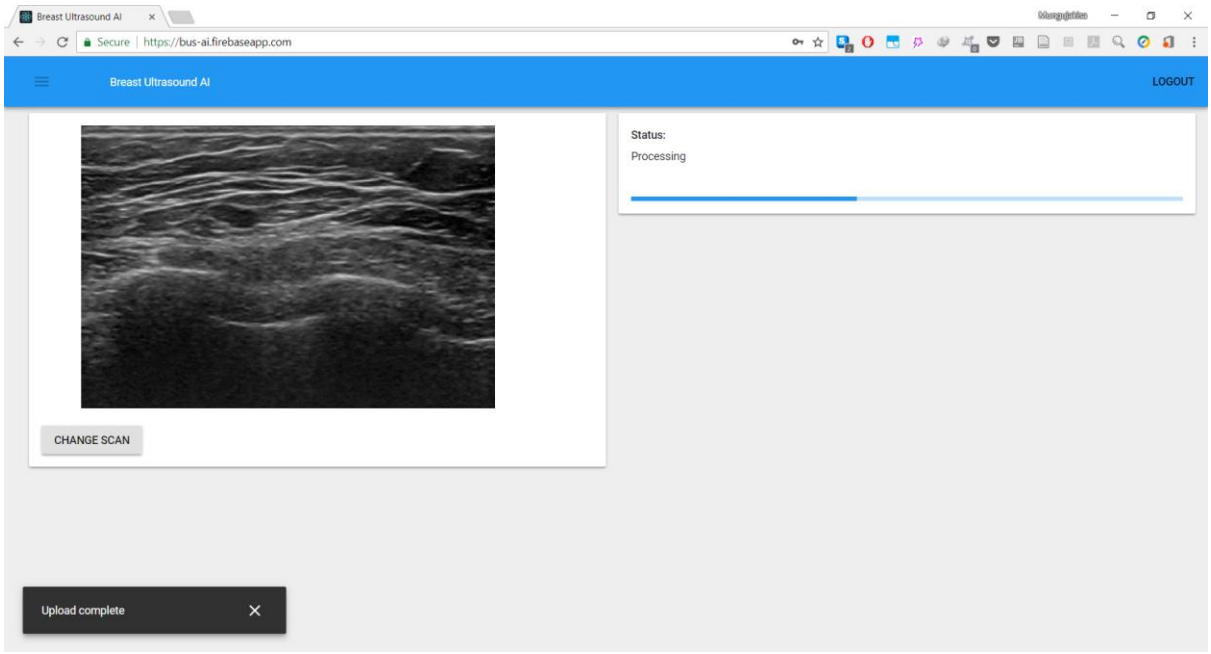


Figure 45: After uploading an Ultrasound scan the system performs analysis

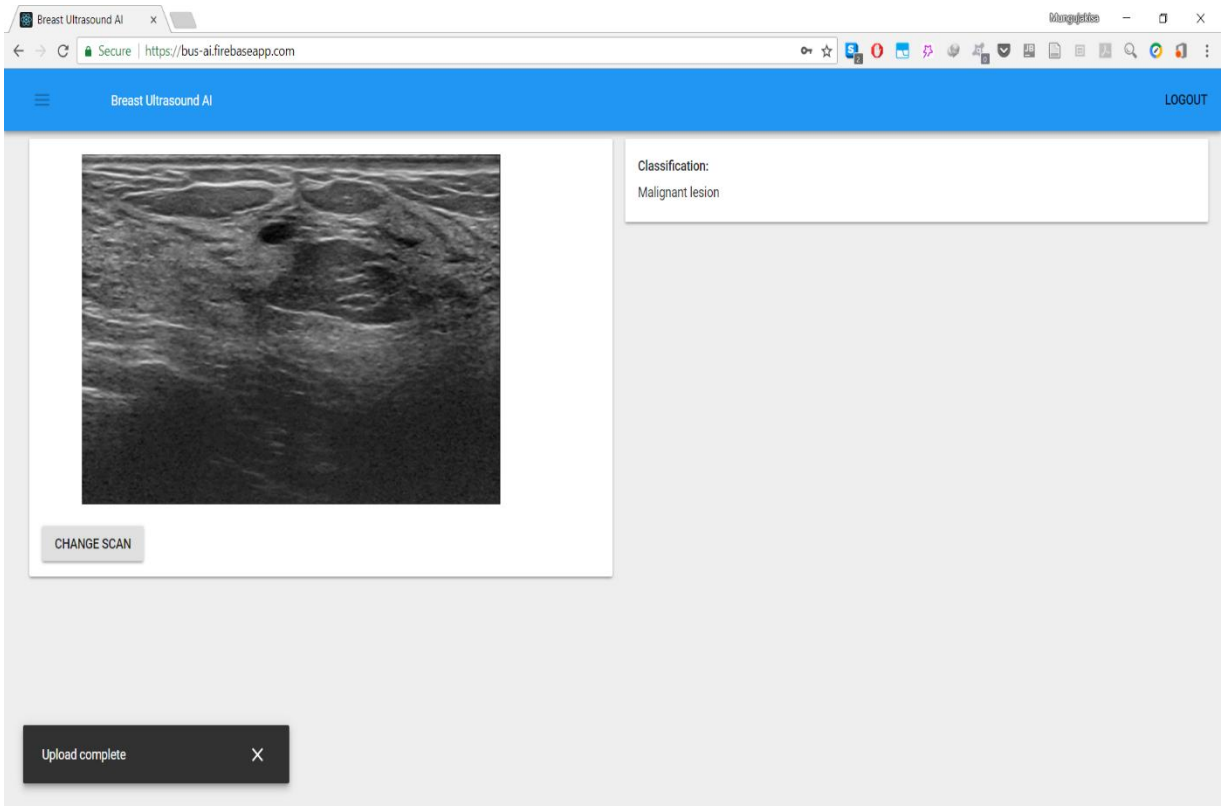


Figure 46: After analysis using the CNN model a lesion classification is displayed

On the server side the application runs in a Node.js container. The CNN model was deployed on the Google Cloud Platform.