Breast Cancer and Benign Detection using Multi-Stage Deep Convolutional Neural Networks

Submitted by

Yacoub Abu Lubad **201930007**

Supervised by

Assistant Prof. Dr. Talal A. Edwan

Submitted in partial fulfillment of the requirements for the award of the Bachelor of Science in Computer Engineering Degree

Faculty of Engineering
Al-Ahliyya Amman University
Amman - Jordan

May, 2022

Abstract

The implementation of Computer Vision in the prediction of breast cancer is a necessity since it cuts down a lot of costs, with the main cost in this problem is time. This Thesis will discuss the different approaches towards using *Artificial Intelligence (AI)* in detecting of these cancerous and non-cancerous (benign) tumors using *Image Classification* and *Object Detection*. The main goal is to reduce the time taken for a result to be in the reach of a professional, from weeks to matter of **seconds**. Reducing the time will not be the only goal of this thesis, but also the need for a radiologist profession will be optional, given the circumstances of the radiologist being busy or on vacation or having a shortage of staff. This will be possible by cutting the process of waiting for a call after a screening session to only be tested again for diagnostics of the abnormality, but rather have the diagnostics appear during the screening session, cutting down the time to detect and helping out the present radiologist in highlighting the work.

Keywords: *Breast Cancer* *Classification* *Detection*

Contents

List of Figures

Listings

List of Tables

List of Abbreviations

AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
DNN	**D**eep **N**eural **N**etwork
MS-DCNN	**M**ulti-**S**tage **D**eep **C**onvolutional **N**eural **N**etwork
MSOD	**M**ulti-**S**tage **O**bject **D**etector
MAE	**M**ean **A**bsolute **E**rror
MLP	**M**ulti**l**ayer **P**erceptron
GPU	**G**raphical **P**rpcessing **U**nit
HDD	**H**ard **D**isk **D**rive
RAM	**R**andom **A**ccess **M**emory

List of Symbols

Symbol	Name
b	bias
w	weights
E	some loss-function
ϕ	activation function
η	cost function

1 Introduction

The *motivation* of this Thesis will be discussed in Section 1.1, while explaining more what the *actual problem* is and the *purpose of this Thesis* in Section 1.1.1. As for the *objectives* of this Thesis, that will be briefly discussed in Section 1.2, while the *organization* and the structure of this Thesis is to be discussed in Section 1.3.

1.1 Motivation

Breast cancer is the most common cause of new cancer cases, according to the **World Health Organization (WHO)**, standing at 2.26 million recorded cases in 2020, meanwhile, the total count of cancer caused mortality cases are recorded to be 10 million in 2020, breast cancer is documented to have caused 685,000 deaths in 2020 [1]. There are multiple ways of reducing the mortality rate of this disease, one of the main approaches would be early detection. A study has proved that a delayed diagnosis and detection of *more than 6 weeks* gave these cancerous cells enough time to develop into much dangerous stages, but diagnosis that were conducted *under 6 weeks* of delay detected less advanced stages of these cells [2]. So the earlier the detection of these cells are prompted, the safer and less severe it is on the patient.

This sort of detection is done via Screening Mammography where a patient undergoes 4 different kinds x-ray imaging, one image is taken from the side of the breast and the second is from the top, same procedure is repeated on the second breast [3]. This totals in 4 images per patient where the radiologist can evaluate whether there are any *abnormalities* detected, which are usually *lumps* that are classified into two categories, *Benign and Cancer*. The only downfall is that if any abnormalities are detecting after the first evaluation, the patient is contacted, which could typically take up to a week or two, to visit and perform a diagnostics test so that the abnormality could be studied even further in classifying whether it is cancerous or not [3].

1.1.1 The result delay problem

As discussed in Section 1.1, the issues of late diagnosis and detection of these cancerous cells would increase the mortality rate, and it has also been pointed out that the results of any abnormalities detected by the radiologist could only be known after a duration of week or two. This problem can be tackled by reducing the time of detecting the abnormalities from *1-2 weeks* all the way down to matter of **fractions of a second.**

This method is a great aid in early detection, this way the patient would have a time advantage to proceed to the diagnosis of this abnormality right away. However, it is also possible to *skip* the diagnosis procedure as well, saving even more time when all it would require is one mammography screening and the results would be ready the moment the screening is completed.

A key feature in making this quick and accurate detection is using two methods that fall under the **Artificial Intelligence (AI)** domain, *Image Classification* and *Object Detection*.

1.2 Objectives

The goal of this thesis is to compare and contrast the different results that could be obtained using different *Convolutional Neural Network (CNN)* model structures. Applying **4 different concepts** to the same dataset but with different utilization of the data to measure the accuracy that might be obtained when manipulating the data differently. Due to the vast dataset that was obtained, one of the multiple hardware constraints were during loading the images from the *Hard Disk Drive (HDD)* onto the *Random Access Memory (RAM)*, there was no enough memory to load the whole dataset to proceed with the model training.

A different kind of hardware constraint that was faced is the absence of *Graphical Processing Unit* (GPU), a GPU is crucial for *Computer Vision* (CV) due to the enormous data stream that will take place during training [4].

1.3 Organization

The *Literature Review* will be discussed in *Section 2* alongside the essential background on the different kinds of Neural Networks with their diverse functionalities. A deeper dive into the application's functions will be thoroughly described in *Section 2.2* and the two main techniques that will be implemented in this Thesis. In *Section 2.3*, there will be a detailed explanation on the **Multi-Stage approach** that is applied to obtain the results as well as the methods that helped surpassed the constraints that were mentioned in *Secion 1.2*. Detailed dissection of the dataset will be in *Section 3.2*.

2 Background

This thesis will focus on implementing a *Multi-Stage Object Detector*. There has been previous implementations of Brest cancer detection using a *one stage* object detection, which will be explained in *Section 2.2.2*, as seen in [5], but the gap remains with setting up a *two stage* object detector to detect and classify these tumors. In other words, this is an application of **Multi-Stage Deep Convolutional Neural Network (MS-DCNN)** since it relies on *two stage object detection*, which will be explained in *Section 2.3*.

2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a simplified mathematical models that mimic the functionality of a human brain and nervous system [6, 7]. Just like humans, these networks were designed to solve more complex, *non-linear*, highly stochastic and multi-variable problems that a traditional program could not. These problems span out to the fields of medicine, finance, security and many more [8]. *ANNs* are designed to approximating any continuous function thus are used in a wide spectrum of applications such as object detection [9, 10], image classification [11], image enhancement [12] as well as several more uses.

The *ANN* is originally compromised from multiple *neurons*, or *perceptron*, which can be demonstrated in *Figure 2.1*, hence the *perceptron* is the ground foundation of *ANNs*.

The input $\boldsymbol{x}$ of size $\boldsymbol{n}$ for this perceptron is denoted as the input vector that is composed of numerical values representing different features of a single

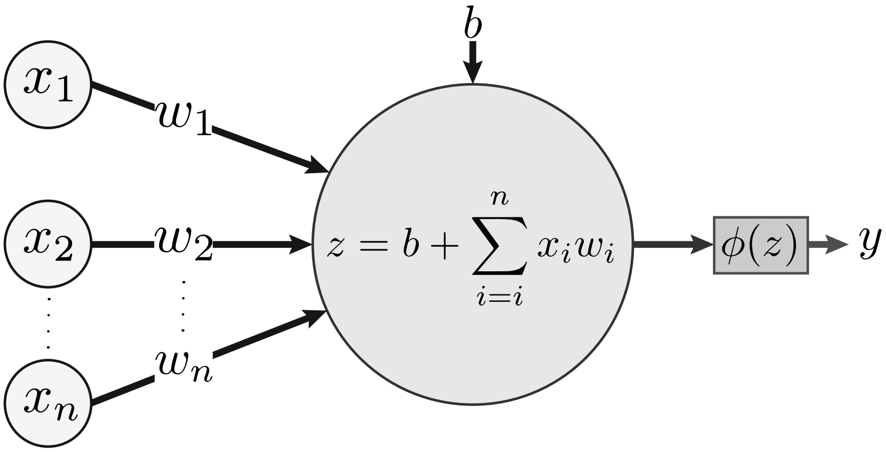


Figure 2.1: Artificial Neuron *(Perceptron)*

entry as seen below.

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{1}$$

Depending on the features of a given model data, different features require different *weights*, denoted as $\boldsymbol{w}$, since one feature would have more effect on the final output more than a different kind of feature, thus an input must be multiplied with a weight to determine its importance to the final output.

$$\boldsymbol{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix} \tag{2}$$

The above vector will differ based on the next layer's size, given that there is $\boldsymbol{m}$ number of neurons in the next layer, the *weight's matrix* will have a size of $\boldsymbol{m} \times \boldsymbol{n}$ as seen in *Equation 6.* The weight vector is a *row vector* due to the relationship it contains with the input as every $\boldsymbol{n^{th}}$ input, there's a $\boldsymbol{n^{th}}$ weight corresponding to it.

After the input is multiplied with its assigned weight, it is now referred to as the **weighted input**, that weighted input is summed with the rest of the weighted inputs which then derives us the **weighted sum**. Then as all these inputs are summed, a *bias* $\boldsymbol{b}$ is added which is an additional parameter that is used to adjust the *output* of the perceptron as well as the weighted sum that is *inputted* into the perceptron.

The output of the perceptron can be denoted as $\boldsymbol{y}$ and is calculated as follows:

$$y = \phi(z), \tag{3}$$

where

$$z = b + \sum_{i}^{n} x_i w_i \tag{4}$$

The ϕ is an arbitrary function known as the **activation function** which is responsible for causing the perceptron to *fire* generating an output. This activation function is deduced by a threshold that is set based on the different types of activation functions alongside their different uses which can limit the output of reaching an undesired or unacceptable value [13].

One of the most common activation functions is the *Rectified Linear Unit (ReLU)*[14] which is defined as:

$$\phi(z) = max(0, z) = \begin{cases} z & z > 0 \\ 0 & z < 0 \end{cases} \tag{5}$$

and it is widely used in *Neural Networks* as the *hidden layer's* perceptrons', which is discussed in *Section 2.1.1*, activation function.

The expansion from a single perceptron to multi perceptrons forms a *single layered neural network* as seen in *Figure 2.2*; using the input vector as seen in *Equation 1* and defining the weights matrix

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \tag{6}$$

where $\boldsymbol{m}$ is the number of perceptrons and $\boldsymbol{n}$ is the number of inputs. The weight responsible for the $\boldsymbol{n^{th}}$ input and $\boldsymbol{m^{th}}$ perceptron is written as $\boldsymbol{w_{m,n}}$. While the bias vector is defined as:

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{7}$$

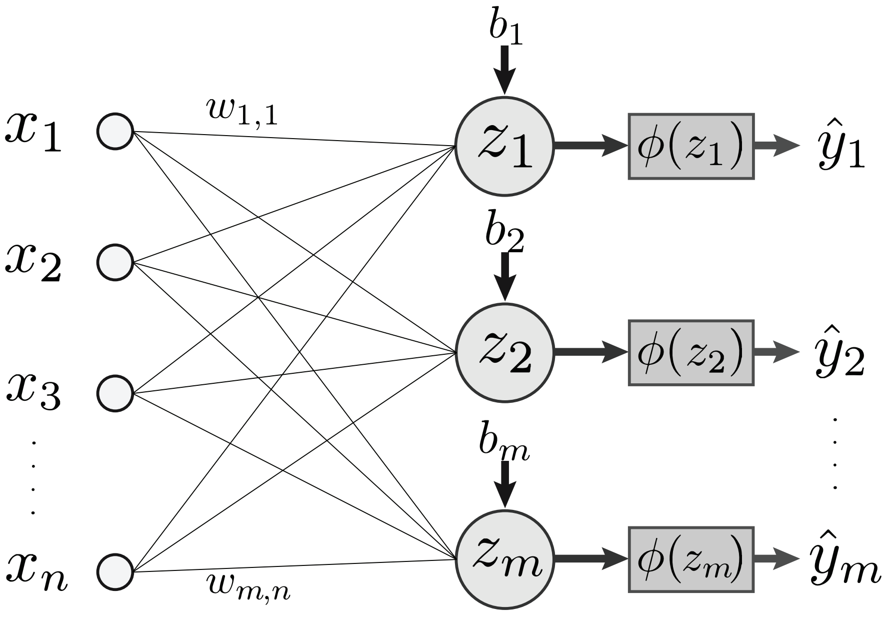


Figure 2.2: Artificial Neural Network Model *(Single Layer)*

The *multi-perceptron, single layer, output* can be computed as:

$$\hat{\boldsymbol{y}} = \phi(\boldsymbol{b} + W\boldsymbol{x}) \tag{8}$$

So far, this has been a **feed-forward** neural network without any kind of *learning*, for the network to start learning there must be a *weight update* algorithm which updates the weights regularly after every entry. This algorithm is known as *Backpropagation* [15], and the formula can be seen in *Equation 9* below:

$$W^{t+1} = W^t - \eta \frac{E}{\Delta W} \tag{9}$$

where W^t denotes the weight at iteration t of the gradient descent and η is a *cost function.*

Cost functions, also technically known as **Loss functions**, vary depending on the faced problem and the desired output of the neural network, for instance, a *Regression* problem will most likely use a **Mean Absolute Error (MAE)**[16] as a cost function which can be computed as:

$$MAE = \frac{\sum_1^n |y_i - x_i|}{n} \tag{10}$$

where y_i is the predicted value, x_i being the true value and n the total number of data points. However, an *Image Classification* problem will most likely use **Categorical Crossentropy**[17] which quantifies the difference between probability distributions in a a multi-class classification problem and it can be computed as:

$$E = \frac{-(\sum_1^N y_i \log(x_i))}{N} \tag{11}$$

where y_i is the predicted value, x_i being the true value and N the number of classes.

2.1.1 Multilayer Perceptrons

Multilayer Perceptrons (MLPs) is one class of the *feed-forward ANNs* where, at least, one **hidden layer** is present between the input layer and the output layer, a demonstrative diagram can be seen in *Figure 2.3*, the hidden layers act like a *black box* where the objective of this so called black box is to extract features from the input, the *deeper* the network, the more features it will be able to extract. But having a *Deep Neural Network (DNN)* could be troubling sometimes as there would be more than 2-3 layers which, in coherence to the more feature extraction, it also acts in a much more complex way which applies a lot of constraints in areas such as software and hardware. The reason for the software constraint with *DNN* is that it will require more data to learn unlike shallower models. As for hardware constraints, the process that is responsible for training *DNNs* is known as *Deep Learning*, Deep Learning requires advanced *GPUs* which could get costly for a user, the essence of a GPU is crucial due to the time constraint that might be present in the absence of a GPU [4].

The *black box* could be seen also in *Figure 2.3* as a light gray box captioned *Hidden Layers*.

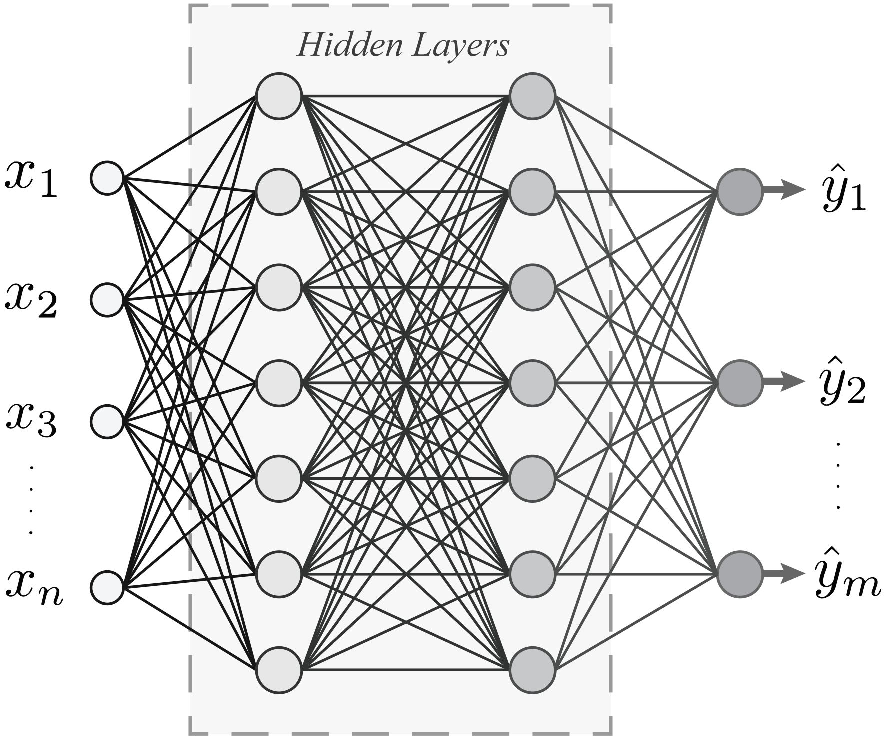


Figure 2.3: Three layer feed-forward network

However, each layer, of these hidden layers, is still computed as *Equation 8*. Bare in mind that the input of the next layer would be the previous layer's computed $\hat{\boldsymbol{y}}$, using *Equation 8* for the first layer and computing the output layer number $\boldsymbol{i}$ as:

$$^{(i)}\hat{\boldsymbol{y}} = \, ^{(i)}\phi(^{(i)}\boldsymbol{b} +^{(i)} W^{(i-1)}\hat{\boldsymbol{y}}) \tag{12}$$

where $\boldsymbol{i \geq 2}$
Moreover, layers where each perceptron is connected to every perceptron of the following layer, as seen in *Figure 2.3*, are called *dense* or *fully connected* (FC) layers.

2.1.2 Convolutional Neural Networks

Section 2.1.1 included an explanation on *ANNs* and how it helps in learning more complex functions using deeper networks, however image inputs are not suitable for *MLPs*, since *MLP* networks handle vector inputs, hence requiring the image to be flattened leading to an extreme increase in number of trainable parameters. In other words, using an image of size (512x512) as an input to extract the features would lead to a single perceptron containing **262.144** trainable parameters (excluding bias) and increasing number of perceptrons in the first hidden layer to around 100, which is not convenient for the size of the image but just setting it as an example, would set the number of trainable parameters to an astonishing **26.214.400** just for the first layer. In addition, increasing the resolution of the image and using a three-channel *RGB* image will also exceedingly increase the trainable parameters. Therefore, a different kind of architecture is needed to handle images, and that new architecture is called *Convolutional Neural Network (CNN).*

As stated in [18], "Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers". For an input image I and kernel (filter) K , the discrete convolution operation is defined as [18]:

$$c_{i,j} = (I * K)_{i,j} = \sum_m \sum_n I_{m,n} K_{i-m,j-n} \tag{13}$$

In general the convolution operation on these images would produce an output feature image which is produced by convolving the input image with the kernel, the kernel would be a set of weights (trainable parameters) which the size could be defined by a user.

Although many implementations talk about convolution and applying *Equation 13* to the "*Convolutional* Neural Network", an alternative method is used which is called *cross-correlation*. As we can see in *Equation 13*, where m and n iterate over valid subscripts of both $I_{m,n}$ and $K_{i-m,j-n}$, the *filter* K is flipped thus producing a **flipped** output, since the filter would be sliding in the negative direction. Consequently, an alternative variant is introduced called *cross-correlation* and it is computed as follows:

$$c_{i,j} = (I * K)_{i,j} = \sum_m \sum_n I_{i+m,j+n} K_{i,j} \tag{14}$$

Hence producing an output which is not flipped as seen in *Figure 2.4*.

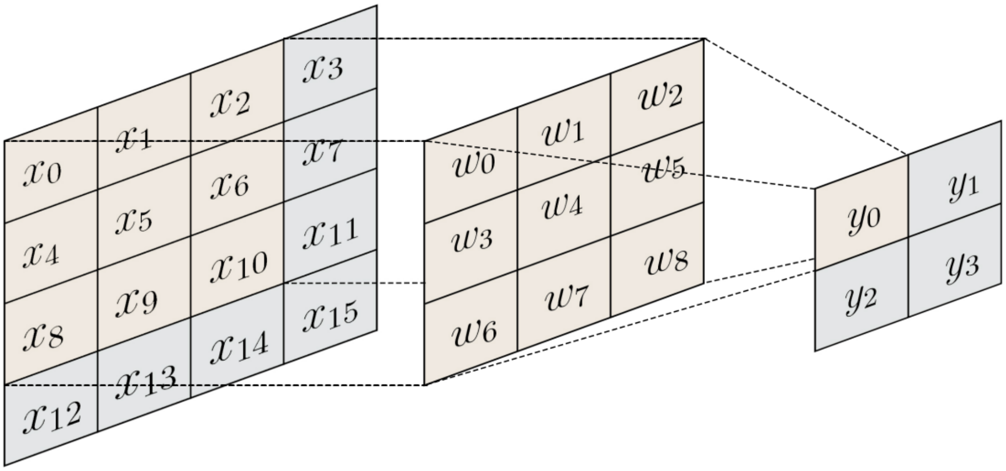


Figure 2.4: An illustration of *Cross-Correlation*

The process can be seen as the summation of the element-wise product between the kernel and any spatial location of center (i, j), then moving the kernel by the amount of *stride* to another location until the whole image is covered. Additionally, for multi-channel input image the *Equation 14* can be expanded as follows:

$$c_{i,j} = (I * K)_{i,j} = \sum_m \sum_n \sum dI_{i+m,j+n,d} K_{i,j,d} \tag{15}$$

where d iterates over valid subscripts of both $I_{i+m,j+n,d}$ and $K_{i,j,d}$. Note it is frequent to add bias to the equation. Multiple kernels can be applied to the same input image to obtain several feature maps. Thus, the output depth is equal to number of kernels / filters applied. One of the characteristics of a convolutional layer is **parameter sharing**; it is possible to assume that if a specific filter is useful in some region, then it is useful in other regions as well. Under this assumption, the parameters are shared along the depth [19] reducing the number of learnable parameters. Moreover, it is frequent to apply an activation function to feature maps in order to obtain activation maps.

Convolution reduces the size of the input image. For this reason, when several convolution layers are used, the image size decreases drastically. As a counter-measurement, an outer frame is added to the image, limiting the size reduction. This process is called **padding**. It is common to use zeros as values for the frame and this is called *zero padding*, while reflecting values of rows and columns into the frame is named *reflection padding.*

For an image of height (H_i) and width (W_i), the size of the image post convolution is computed as:

$$\hat{W}_i = \frac{W_i - k + 2P_i}{s} + 1 \tag{16}$$

$$\hat{H}_i = \frac{H_i - k + 2P_i}{s} + 1 \tag{17}$$

where $\hat{W}_i$ and $\hat{H}_i$ are the new width and height, respectively, of post convolution process. k is the kernel size, s being the stride and the padding size denoted as P_i.

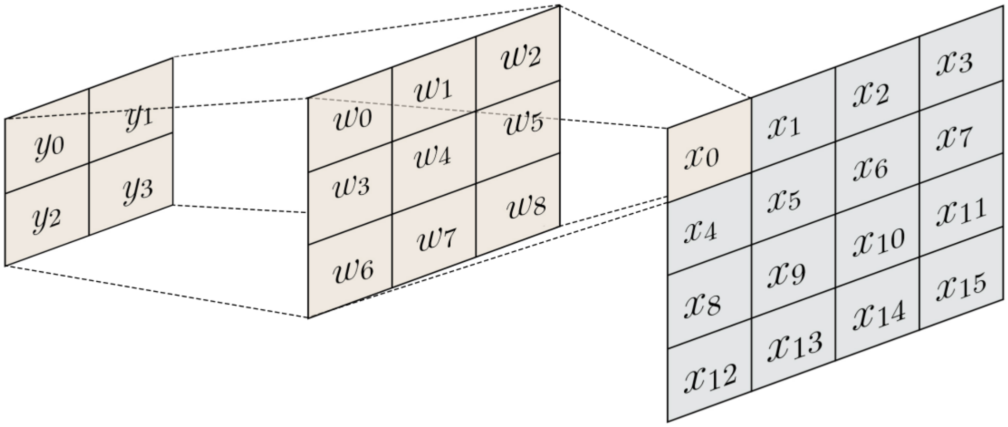
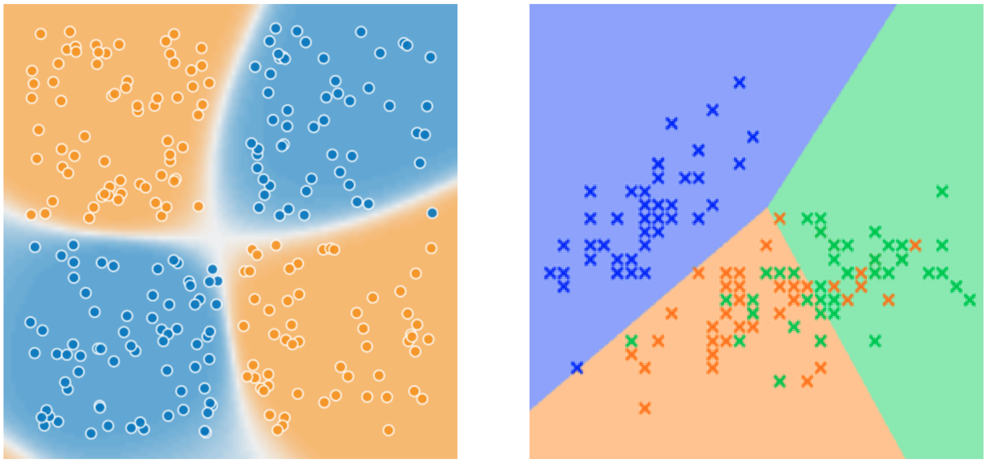


Figure 2.5: An illustration of *Transposed Convolution*

In a convolutional layer, the kernel defines a convolution that has a forward pass and backward pass. Flipping the passes would result in *Transposed Convolution* or known as *fractionally strided convolution* [20], which is used for upsampling. Transposed Convolution can as well be visualized in *Figure 2.5.*

2.2 Computer Vision

Human vision is similar to *Computer Vision (CV)*, with the exception that people have a head start. Human vision benefits from lifetimes of context to teach it how to distinguish objects apart, how far away they are, whether they are moving, and whether something is incorrect with an image. CV teaches computers to execute similar tasks, but using cameras, data, and algorithms rather than retinas, optic nerves, and a visual cortex, it must do it in a **fraction of the time**. Because a system trained to check items or monitor a production asset can assess hundreds of products or processes



(a) Binary Classification [21]

(b) Multi-Class Classification [22]

Figure 2.6: Displaying the difference between binary and multi-class classification

per minute, detecting faults or issues that are invisible to humans, it can swiftly **outperform humans**. The speed and the methods will be discussed in *Section 2.2.1* and *Section 2.2.2*.

2.2.1 Image Classification

Image Classification tasks are generally easy to grasp, it is simply the discrimination between classes, whether *Binary* or *Multi-Class* classification, a classifier's task will be to pick up on the features of each class and *learn* them to be able to label input data. Binary Classification and Multi-Class Classification is displayed clearly in *Figure 2.6*. Knowing that a classifier can classify multiple classes and not necessarily just a binary classification, this also means that it can classify up to $\boldsymbol{n}$ numbers of classes. A good example of multi-class classification is the MNIST hand-written digit classification [23], this problem hosts **60.000** hand-written **labeled** digits, as seen in *Figure 2.7*, where $\boldsymbol{n} = \mathbf{10}$ as we have 10 numerical digits. This sort of dataset is fed into an Image Classifier which then develops an understanding of the differences each hand-written digit contains, this classifier then outputs the estimated label of the digit. A classifiers flow is seen in in *Figure 2.8*.

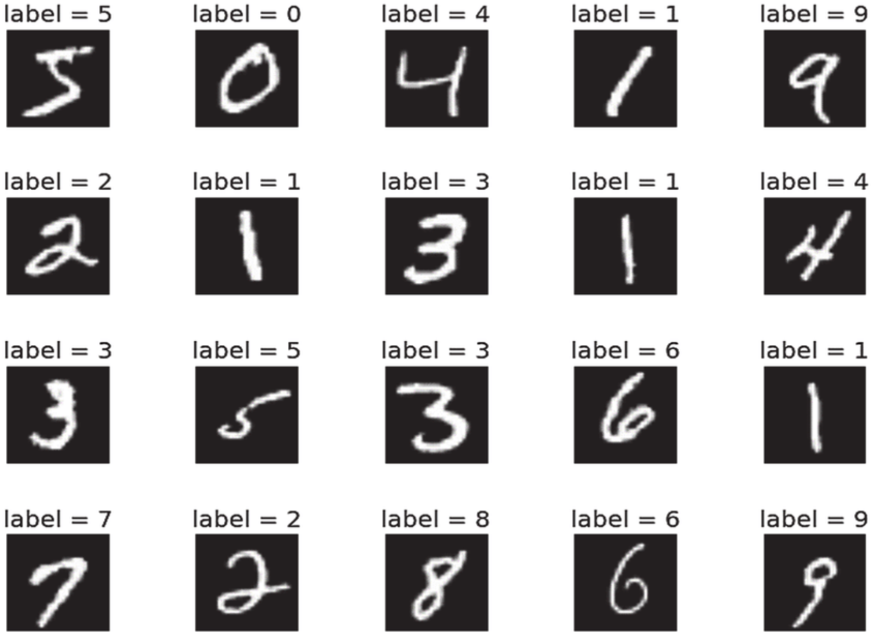


Figure 2.7: MNIST dataset with each class represented above the image respectively

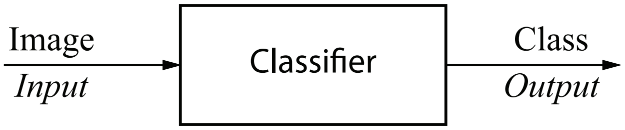


Figure 2.8: Classifier Diagram

2.2.2 Object Detection

In an object detection task, a model has to both classify objects / instances appearing in an image and localize them within the image. The localization is represented by a 2D bounding box, in which the structure of the object detector is visible in *Figure 2.9.* State of the art networks tackling this task are divided into two main groups. The first group are the single stage object detection network, which prioritize the **inference speed** over than the accuracy. Single stage methods include *YOLO* [10] and *RetinaNet* [24].The second group of methods is the two stage methods, which are tuned for **accuracy** over inference speed. An example of two stage methods is the Faster R-CNN [9]. Unlike Image Classification, Object Detection is only responsible for detecting and localizing learned features no matter how many *Classes'*

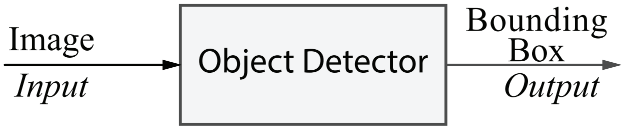
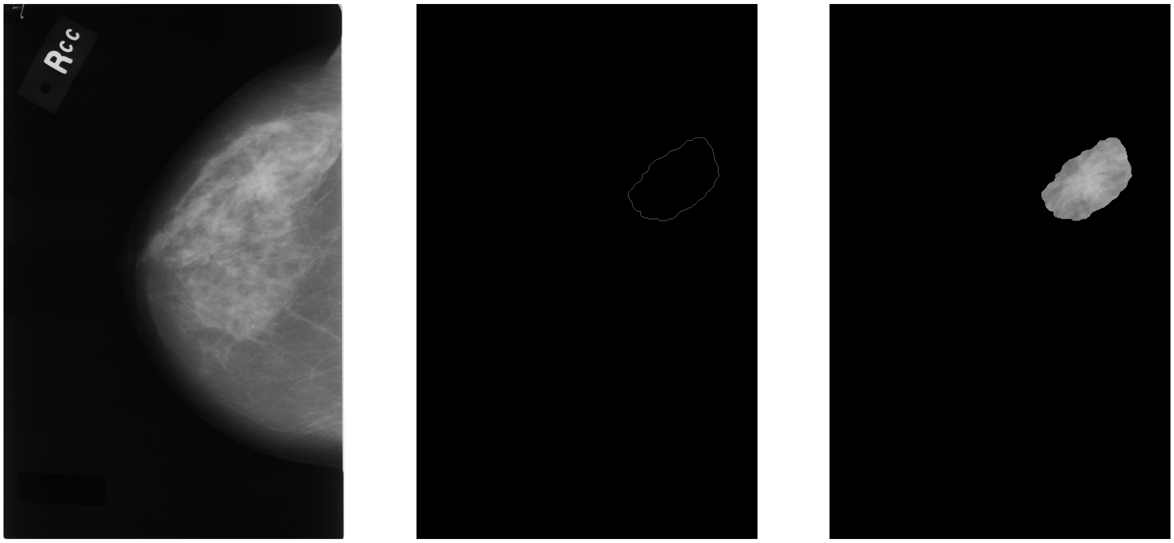


Figure 2.9: Object Detection Diagram



(a) Image containing cancer cell

(b) Contour Mask

(c) Result Image

Figure 2.10: An image and the mask of the cancerous cell to isolate the cell from the rest of the image so the object detector would be able to learn the features of given cell with the isolation procedure beginning with the mask filled with ones, then overlapping the original image which contains the cancerous cell, the two images are then multiplied by each other making everything black except for whatever is inside the mask as seen in the *Result Image*

features a detector learns. In instance when it comes to the two different kinds of object detectors, single and two stage detectors, this thesis will conduct the two stage object detector method as the results do not really rely on inference speed. And as discussed earlier, single stage object detectors would prioritize inference speed over accuracy as they would be widely used in real-time object detection. However, the dataset, which will be discussed in *Section 3.2*, is not a live image but rather a static image which is processed after performing x-ray imaging [25]. An object detector learns by inputting a single object that is restrained by a bounding box or a mask. An image with a contour mask could be seen in *Figure 2.10.*

2.2.3 Hierarchy Classification

In *Section 2.2.1*, it has been explained that a classifier can classify between multiple classes, say there is three different classes $\boldsymbol{x}$ $\boldsymbol{y}$ and $\boldsymbol{z}$. These three classes have two classes which are quite similar, say that these similar classes are $\boldsymbol{x}$ and $\boldsymbol{y}$ sharing multiple features and are completely different from class $\boldsymbol{z}$ where they share no similar features. The classifier will be most likely to have a **larger error** when it comes to classifying between class $\boldsymbol{x}$ and $\boldsymbol{y}$, while it will be easier to classify class $\boldsymbol{z}$.

This is where **Hierarchical Classification** comes into play, which could be graphically demonstrated in *Figure 2.11*, as it will feature two separate classifiers, say one classifier to distinguish between two majorly different classes such as $\boldsymbol{\{x, y\}}$ and $\boldsymbol{z}$, this classifier will be denoted as **Classifier 1** which outputs **Class 1** that is a binary category on whether the input is classified as $\boldsymbol{z}$ or as $\boldsymbol{\{x, y\}}$. The reason that **Classifier 1** is set up is because it will be able to better distinguish between these two vastly different classes since it is easier to learn and distinguish their features, hence **Classifier 1** would be trained on the binary categorization, and if the output is classified as $\boldsymbol{\{x, y\}}$, then the same input will be taken to **Classifier 2**. **Classifier 2** is trained on discriminating between $\boldsymbol{x}$ and $\boldsymbol{y}$, and that is **Class 2**, which will give the classifier much better judgment over the two similar classes and have no third class interfere with the learning of the distinction between the two similar

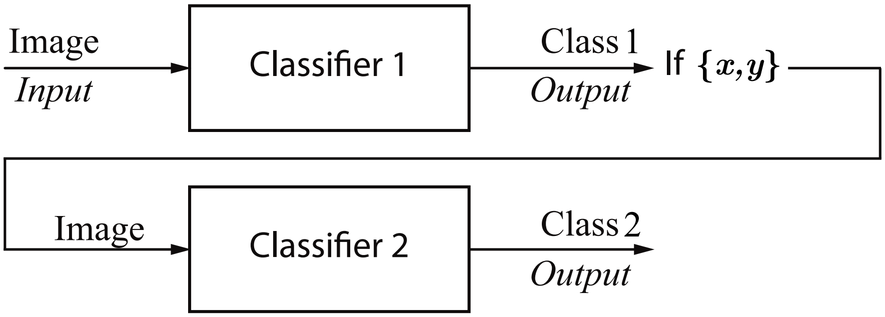


Figure 2.11: Hierarchical Classifier Diagram

classes. This approach has been used in [26] for x-ray image classification of patients with Pneumonia, Covid-19 *(Unhealthy)* or Neither *(Healthy)*. This Hierarchy can be seen in *Figure 2.12*.

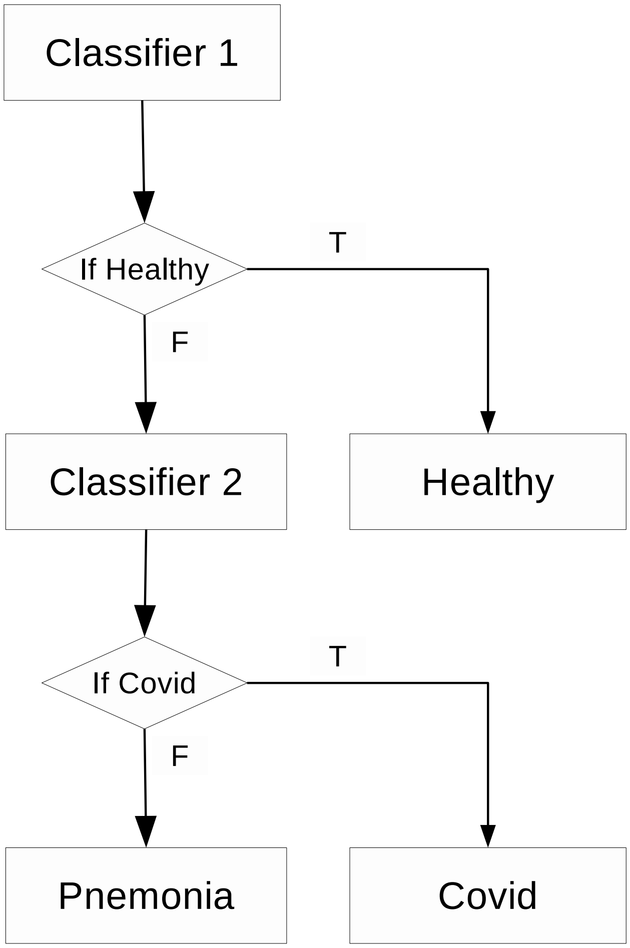


Figure 2.12: Hierarchical Classifier Flowchart

2.3 Multi-Stage Object Detector

Single-Stage object detectors work with having the classification process happen at the end of the detector since that is the quicker method, hence the name given to it is the *"Single-Stage"*. However, in *Two-Stage* object detectors, the object detector works on a single model and then if a detector detects an object, it is passed to another model and that is the second stage which is a classifier, hence the name *"Two-Stage"* object detector is given and is also called the **Multi-Stage Object Detector (MSOD)**, this can be clearly seen on how insignificant it is in regards to inference speed in comparison to the Single-Stage object detector.

Figure 2.13 shows the work-flow of a *MSOD*.

The importance of using a *MSOD* rather than a Single-Stage object detector is mainly due to the higher accuracy that will be acquired by the model since the object detector is separated from the classifier and each carries out a different task, the detector focusing on the features of the tumor and detecting the tumor, no matter which *Class* said tumor is given to, while the classifier would receive an image solely of the tumor rather than the whole raw image, as would the second classifier from the *Hierarchical Classifiers* in *Section 2.2.3* receive and that being the whole image. *Figure 2.14* shows the cropped image of the tumor entering the classifier in the *MSOD* model as a product from *Figure 2.10.* Having a cropped image of the tumor as an input for a classifier would help focus all the attention on the features of the two different kinds of tumors, thus improving the learned features, while correspondingly reducing the amount of learning parameters as the input size would be smaller hence reducing computational effort and time.

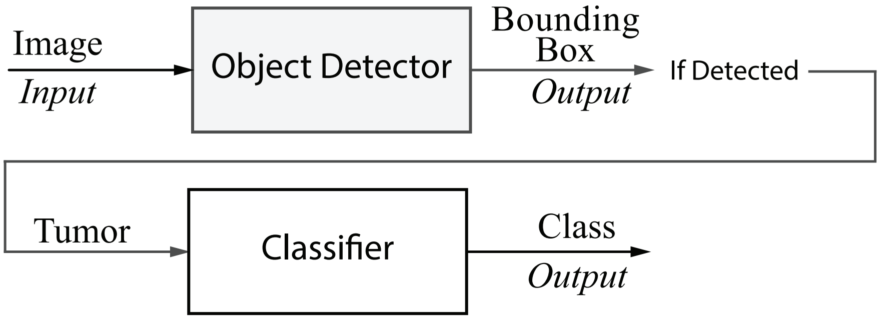


Figure 2.13: Multi-Stage Object Detection Diagram

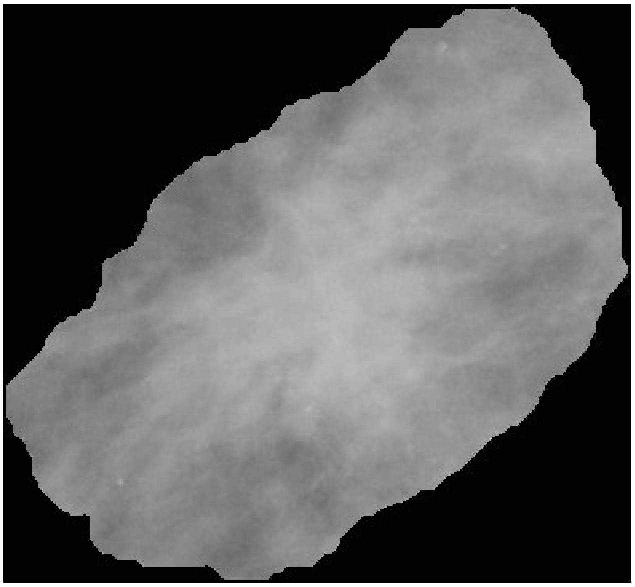


Figure 2.14: Cropped input for *MSOD* Classifier

3 Design

3.1 Methodology

3.2 Dataset

Choosing a dataset for this problem is crucial to many stages, first and foremost is the *quality of the data*. Does the data clearly portray the goal that should be acquired? Secondly is the *quantity of the dataset*, just like the quality, quantity is as important. **Neural Networks** require a lot of data, but the quantity mainly depends on the following:

- features that must be extracted from the data
- type of Neural Network
- data quality
- the desired goal

With that being said, the data is a vital part of this project, and to be able to achieve high accuracy and precision of detecting the cancerous cells, it is a must to obtain high grade dataset. The most common age for the diagnosis of breast cancer is *over 50* years of age. This has been conducted by the National Cancer Institute that the median age of breast cancer patients is between the age of *55* to *64* [27].

3.2.1 Data collection and statistics

In this Thesis, the procedure for *data collection* was straight-forward as there is an already published dataset [28] with the purpose of research that was used in this Thesis. There is quite astonishing remarks regarding the dataset as it holds data for **1.952** patients with **4 images per patient**, two views per breast, which sets the total number of data to **7.808 images** in possession for processing, analyzing and configuring.

Some constraints and observations were met regarding the labeling of the data where a patient is diagnosed as *Cancer* or *Benign* based on the presence of the tumor in at least one of their breast. While that could be true for the case of the patient, however, it is false labeling when it comes to the methodology this Thesis follows as a breast which does not contain any tumor cannot be labeled as not *Normal*, this case could be seen in *Table 1.*

Hence, all the patients that were diagnosed with a tumor in a single breast would be automatically diagnosed the same status in the other breast even though it is clear that the second breast does not contain any tumor as seen in *Table 1*, patient *No. 0236* where only their right breast contains *Benign* cells but their left breast did not contain any tumors. Taking pictures individually with the patient's label is a wrong approach to training an *Image Classifier* as it will *"confuse"* the model and stray it off and the model will start learning

Table 1: A preview for the labeling of the dataset

fileName	View	Side	Status	Age	Tumour_Contour
C_0236_1.LEFT_CC.png	CC	LEFT	Benign	67	-
C_0236_1.LEFT_MLO.png	MLO	LEFT	Benign	67	-
C_0236_1.RIGHT_CC.png	CC	RIGHT	Benign	67	RIGHT_CC_Mask.png
C_0236_1.RIGHT_MLO.png	MLO	RIGHT	Benign	67	RIGHT_MLO_Mask.png
B_3008_1.LEFT_CC.png	CC	LEFT	Cancer	53	-
B_3008_1.LEFT_MLO.png	MLO	LEFT	Cancer	53	-
B_3008_1.RIGHT_CC.png	CC	RIGHT	Cancer	53	RIGHT_CC_Mask.png
B_3008_1.RIGHT_MLO.png	MLO	RIGHT	Cancer	53	RIGHT_MLO_Mask.png
A_0218_1.LEFT_CC.png	CC	LEFT	Normal	59	-
A_0218_1.LEFT_MLO.png	MLO	LEFT	Normal	59	-
A_0218_1.RIGHT_CC.png	CC	RIGHT	Normal	59	-
A_0218_1.RIGHT_MLO.png	MLO	RIGHT	Normal	59	-

the wrong features.

Some other special cases where found such as a patient containing multiple tumors in one breast, containing tumors in both their breasts and in very rare occasions only having the tumor detected in one view rather as in both views.

These ordinary and extra ordinary cases will be taken care off so that there will not be any case of **miss-labeling** of any data.

Some statistics regarding patients is shown in *Figure 3.1* and *Figure 3.2*.

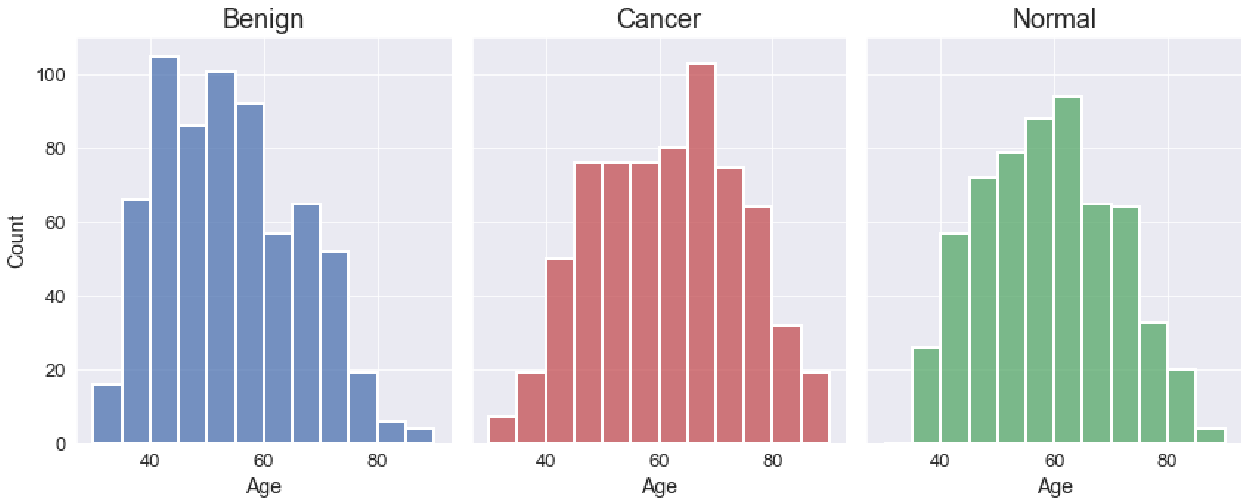


Figure 3.1: Age Distribution

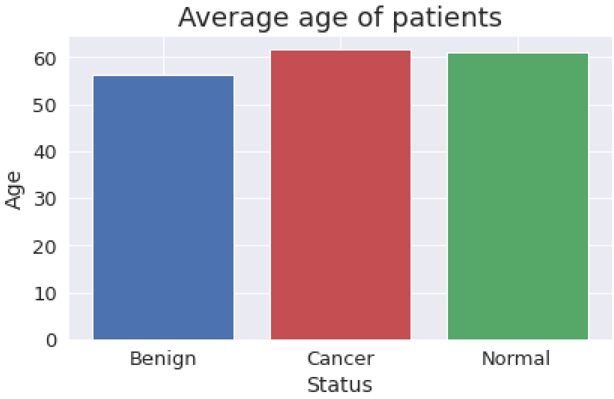


Figure 3.2: Average Age

3.2.2 Data imbalance

After performing the method described in *Section 3.2.1*, the dataset will be thrown into an imbalanced state where the amount of Normal (Tumor-less) images are severely outnumbering the two other cases, this can be seen in *Figure 3.3*.

As seen in the above figure, the dataset is 1:1:4 where the normal cases are **x4** the amount of the Benign and Cancer cases. to counter this we simply store 1 Normal case for every 4 cases that are encountered. The Listing below explains the function that was used to counter the imbalance set.

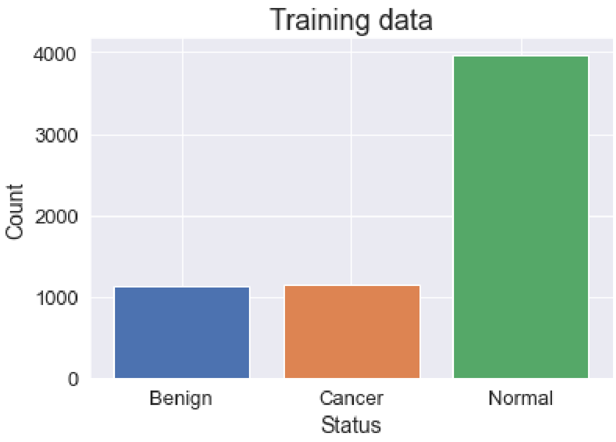


Figure 3.3: Visualization of imbalanced dataset

After implementing this snippet into the **DataLoader**, the dataset imbalance was resolved, as could be seen in the figure below. On another note, for the **Validation Data**, this imbalance will not affect the results in any way. That is why *Listing* **??** will not be implemented for the loading of the validation data.

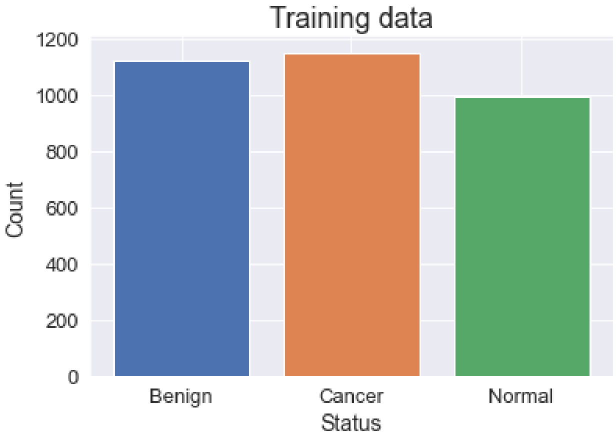


Figure 3.4: Visualization of resolved imbalance

3.2.3 Preprocessing of Dataset

3.3 Image Classifier

The first approach to creating the MSOD model is to create the initial classifier which will be a broader approach to detecting whether the patient is diagnosed with *Cancer*, *Benign* or has no Tumor *(Normal)*. Unfortunately, during training, the whole image is passed which could be seen in *Figure 2.10 (a)*, and that contains many unwanted features that will reduce the performance of the trained model. With transfer learning using the **Efficient Net b-0**, the accuracy of 46.8% was achieved.

In the figure below, the confusion matrix is shown for the first stage image classifier which was discussed in *Section 2.2.1.* Some quick analysis from this Confusion Matrix is that the model has learned to classify the Normal vs Tumor in an almost efficient way. But given that the labels are not *'Normal'*, *'Tumor'* but rather *'Normal'*, *'Benign'*, *'Cancer'*, the model had a hard time differentiating between the two tumor classes. Which is why the **Hierarchy Classifier**, which was discussed in *Section 2.2.3*, will achieve better results

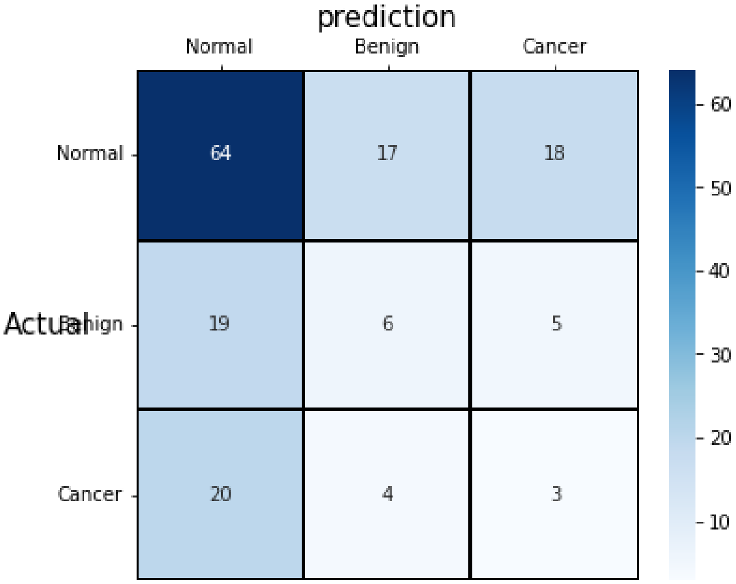


Figure 3.5: Confusion matrix of the first stage Classifier

given that the first stage classifier will predict the two classes *‘Normal‘*, *‘Tumor‘* and the second classifier will have the task of differentiating between *‘Benign‘*, *‘Cancer‘*.

3.4 Hierarchy Classifier

code for running Hierarchy Classifier

```
    def unfix_layers_Backbone_weights(self, Nlayer=3):
    """To unfix the backbone weights

    Args:
    Nlayer (int, optional): the amount of layers to unfix from the end.
  Defaults to 3.
    """
    for i, _ in enumerate(self.model.layers[-5 - Nlayer:]):
    self.model.layers[i].trainable = True

```

Listing 1: Backbone layer unlocking function

3.4.1 Transfer Learning

In transfer learning, it is crucial to keep the *pre-trained weights* intact, at least most of them from the upmost layers, because they are the most important weights when it comes to feature extraction. This is why in *Listing* **??** *line 9*, we have set that the *Backbone's*, which corresponds to the pre-trained model, weights to be untrainable. This helps preserve the weights and not tamper with them during following training.

After successfully training the *Fully Connected* (FC) layers using the function from *Listing* **??**, right after training the fully connected layers only, we will then begin to unlock just a few of the tailing layers from the backbone as seen in the function constructed in *Listing 1.* This function is automatically called from the function that is responsible for training the classifier with some **N** layers of the backbone which can be seen in *Listing* **??**

This allows the last 3 layers of the backbone to unlock and be able to *fine-tune* to our own custom dataset.

3.4.2 Training

Some text about training

Some text about training classifier with backbone

4 Conclusion

In conclusion, in this thesis, the main objective will be to undergo several cases which were listed in *Section 2.2* and end the Thesis with the chosen method which is a *Multi-Stage Object Detector (Two-Stage object detector)*, this focuses on bridging the gap in paper [5] where the chosen approach was a *Single-Stage object detector*. In theory, this should improve the results since it is known that Two-Stage object detection prioritizes accuracy over inference speed while Single-Stage object detectors rather sacrifice accuracy for inference speed (as used in [5]).

References

[1] L. F. C. M. M. L. P. M. Ferlay J, Ervik M, "Global cancer observatory: Cancer today. lyon: International agency for research on cancer," vol. 72, 2 2021. [Online]. Available: https://gco.iarc.fr/today

[2] L. Caplan, "Delay in breast cancer: implications for stage at diagnosis and survival," *Frontiers in public health*, vol. 2, p. 87, 2014.

[3] T. H. E. Team, "How long does it take to get a mammogram and receive the results?" 4 2020. [Online]. Available: https://www.healthline.com/health/how-long-does-a-mammogram-take

[4] Z. Chen, J. Wang, H. He, and X. Huang, "A fast deep learning system using gpu," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS).* IEEE, 2014, pp. 1552–1555.

[5] H. Jung, B. Kim, I. Lee, M. Yoo, J. Lee, S. Ham, O. Woo, and J. Kang, "Detection of masses in mammograms using a one-stage object detector based on a deep convolutional neural network," *PloS one*, vol. 13, no. 9, p. e0203355, 2018.

[6] B. Yegnanarayana, *Artificial neural networks.* PHI Learning Pvt. Ltd., 2009.

[7] A. Abraham, "Artificial neural networks," *Handbook of measuring system design*, 2005.

[8] D. Graupe, *Principles of artificial neural networks.* World Scientific, 2013, vol. 7.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[10] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[11] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition.* IEEE, 2012, pp. 3642–3649.

[12] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[13] N. Siddique and H. Adeli, *Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing.* John Wiley & Sons, 2013.

[14] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[15] C. W. A. H. J. K. A. M. John McGonagle, George Shaikouski, "Backpropagation," *Brilliant.org.*

[16] J. Qi, J. Du, S. M. Siniscalchi, X. Ma, and C.-H. Lee, "On mean absolute error for deep neural network based vector-to-vector regression," *IEEE Signal Processing Letters*, vol. 27, pp. 1485–1489, 2020.

[17] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2019.

[18] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[19] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[20] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[21] TensorFlow, "Tensorflow playground." [Online]. Available: https://playground.tensorflow.org

[22] V. Malhotra, "Intro to multi-class classification," 10 2020. [Online]. Available: https://medium.com/analytics-vidhya/ml06-intro-to-multi-class-classification-e61eb7492ffd

[23] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[24] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[25] N. C. Institute, "Mammograms," 9 2021. [Online]. Available: https://www.cancer.gov/types/breast/mammograms-fact-sheet

[26] M. I. Daoud, Y. Alrahahleh, S. Abdel-Rahman, B. A. Alsaify, and R. Alazrai, "Covid-19 diagnosis in chest x-ray images by combining pre-trained cnn models with flat and hierarchical classification approaches," in *2021 12th International Conference on Information and Communication Systems (ICICS)*. IEEE, 2021, pp. 330–335.

[27] B. Lana, "What is the link between age and breast cancer?" 7 2019. [Online]. Available: https://gco.iarc.fr/today

[28] C. D. Lekamlage, F. Afzal, E. Westerberg, and A. Cheddad, “Mini-ddsm: Mammography-based automatic age estimation,” in *2020 3rd International Conference on Digital Medicine and Image Processing*, 2020, pp. 1–6.