Breast Cancer and Benign Detection using Hierarchical 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

January, 2022

Acknowledgments

Acknowledgment Goes Here...

Contents

List of Figures

List of Tables

Descriptive Abstract

Abstract goes here...

Keywords: *Keyword1, Keyword2, ... , Keywordn*

List of Abbreviations

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

List of Symbols

Symbol	Name
b	bias
w	weights
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] [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 3*, there will be a detailed explanation on the **Hierarchical 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 3.1*.

2 Background

This thesis will focus on implementing a *Hierarchical Object Detector with Classifier*. 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 **Hierarchical Deep Convolutional Neural Network** since it relies on *Hierarchy*, which will be explained in *Section 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

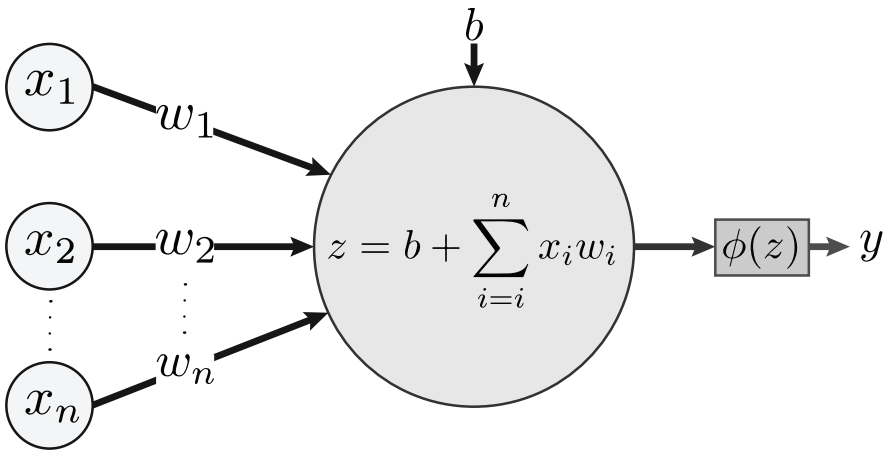

Figure 2.1: Artificial Neuron *(Perceptron)*

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 5*. 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** [13] 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 [14].

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

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \tag{5}$$

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:

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{6}$$

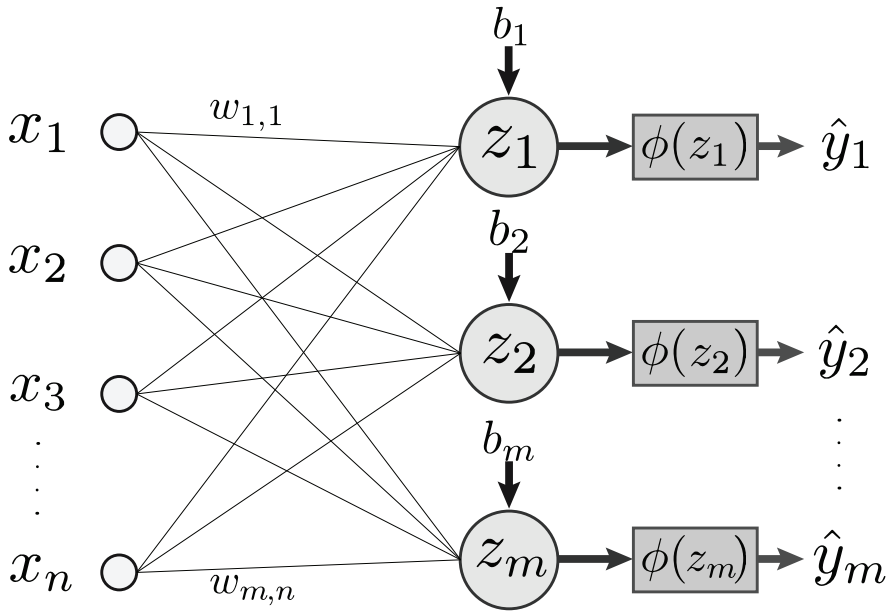

Figure 2.2: Artificial Neural Network Model *(Single Layer)*

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

$$\hat{\boldsymbol{y}} = \phi(\boldsymbol{b} + W\boldsymbol{x}) \tag{7}$$

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 8* below:

$$W^{t+1} = W^t - \eta \frac{E}{\Delta W} \tag{8}$$

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:

$$MAE = \frac{\sum_1^n |y_i - x_i|}{n} \tag{9}$$

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:

$$E = \frac{-(\sum_1^N y_i \log(x_i))}{N} \tag{10}$$

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 *feedfowrard 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*.

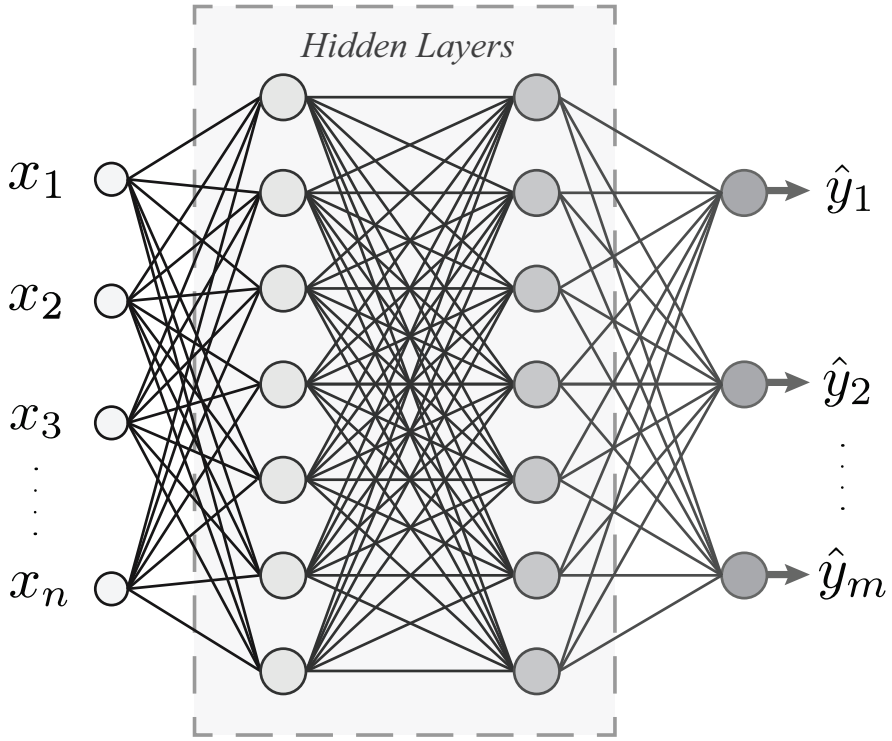

Figure 2.3: Three layer feedforward network

However, each layer, of these hidden layers, is still computed as *Equation 7*. Bare in mind that the input of the next layer would be the previous layer's computed $\hat{\boldsymbol{y}}$, using *Equation 7* for the first layer and computing the output layer number $\boldsymbol{i}$ as:

$$^{(i)}\hat{\boldsymbol{y}} = {}^{(i)}\phi(^{(i)}\boldsymbol{b} +^{(i)} W^{(i-1)}\hat{\boldsymbol{y}}) \tag{11}$$

where $\boldsymbol{i} \geq \mathbf{2}$

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]:

$$c_{i,j} = (I * K)_{i,j} = \sum_{m} \sum_{n} I_{m,n} K_{i-m,j-n} \tag{12}$$

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 12* to the "*Convolutional* Neural Network", an alternative method is

used which is called *cross-correlation*. As we can see in *Equation 12*, 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:

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

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

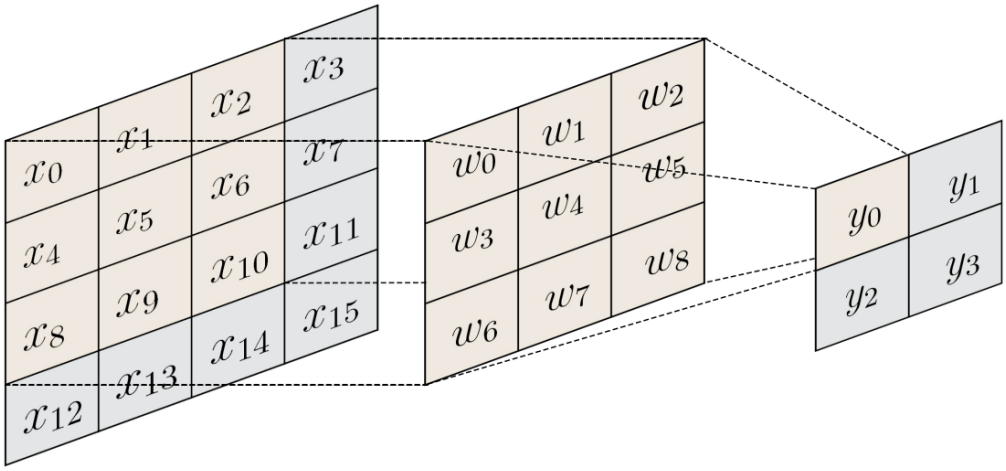

Figure 2.4: An illustration of *Cross-Correlation*

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 13* can be expanded as follows:

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

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:

$$\hat{W}_i = \frac{W_i - k + 2P_i}{s} + 1 \tag{15}$$

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

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

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*.

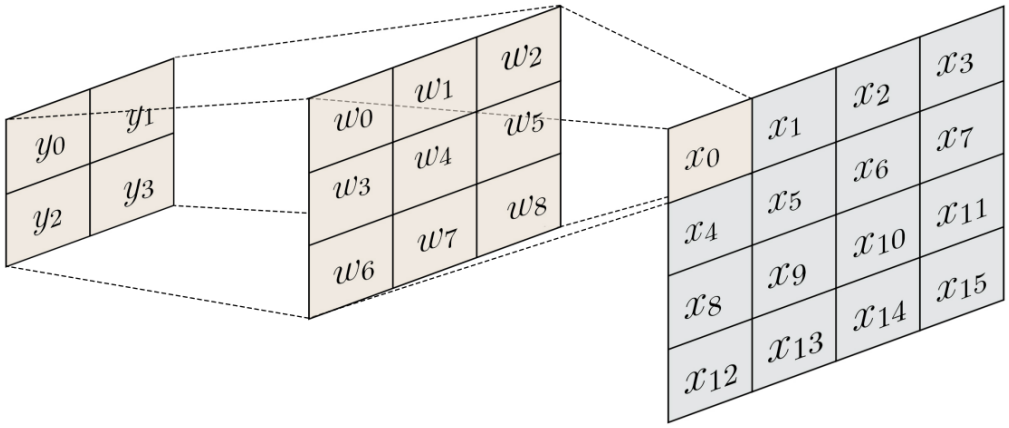

Figure 2.5: An illustration of *Transposed Convolution*

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 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

Some text about **Image Classification**

Figure 2.6: 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 2.7*. 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* [21].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].

Figure 2.7: Object Detection Diagram

2.3 Hierarchy Classifier

An explanation on the Hierarchical approach. some basic explanation on

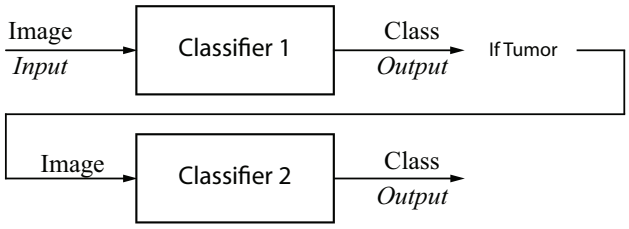

Figure 2.8: Hierarchical Classifier Diagram

Figure 2.8 which is a normal approach found in /refGJU paper.

3 Hierarchical Method

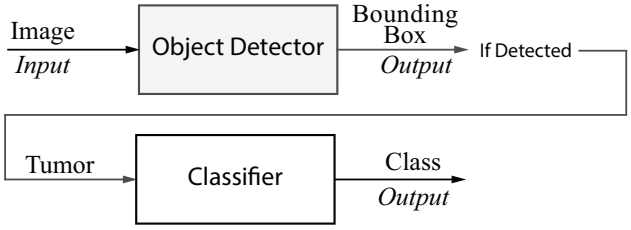

Figure 3.1: Object Detection Diagram

some in depth explanation on *Figure 3.1* alongside many more figures to explain the thought process

3.1 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* [22].

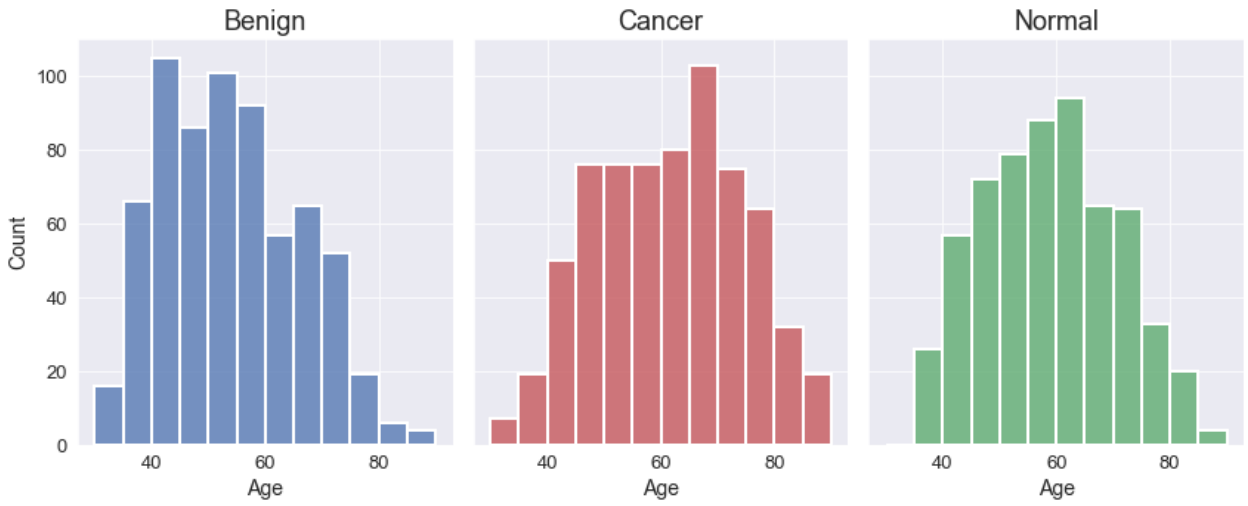

Figure 3.2: Age Distribution

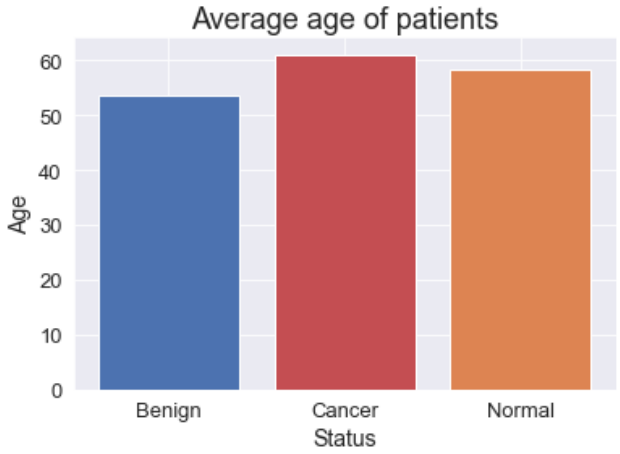

Figure 3.3: Average Age

4 Conclusion

Conclusion goes here

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

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

[15] C. W. A. H. J. K. A. M. John McGonagle, George Shaikouski, "Back-propagation," *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] 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.

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