Breast Cancer Detection using Neural Networks

Systems & Biomedical Engineering Department
Special Functions & Partial Differential Equations (MTH2245)
BioGenesis

Dr. Samah El-Tantawy
Faculty of Engineering - Cairo University

Table of contents

Table of Figure

1. Abstract

During the last decades, artificial intelligence algorithms have altered the world with their applications in all different fields, including medicine. Deep learning is a sub-field of artificial intelligence that has shown impressive results in a variety of fields, especially in the biomedical industry. Deep learning methods such as convolutional neural networks (CNNs) can be used to analyze a large amount of imagery data. CNNs are developed from scratch using partial differential equations, they then are tested and used in the research to detect breast cancer based on patients' breast images (Both ultrasound and x-ray images).

Partial differential equations are used in the backpropagation to calculate the gradient of the loss function with respect to parameters to help optimize and update the neural network weight and achieve the best weights that will affect the model's accuracy.

2. Problem definition

What is cancer?

Cancerous tumor results from the abnormal growth of cells that invade the surrounding tissues of the human body. There are two types of tumors, benign and malignant, and when no tumor is present in the breast, it is considered normal. Benign tumor cells are benign cells that only grow locally and cannot spread by invasion. While malignant tumors are cancer cells and have the ability to multiply out of control, spread to different parts of the body, and invade surrounding tissues.

Breast cancer is a disease in which cells in the breast grow out of control. There are different types of breast cancer. The type of breast cancer depends on which cells in the breast develop into cancer. Breast cancer can start in different parts of the breast. A breast consists of three main parts: lobules, milk ducts, and connective tissue. The lobules are the glands that produce milk. Ducts are tubes that carry milk to the nipple. Connective tissue (consisting of fibrous and fatty tissue) surrounds and holds everything together. Most breast cancers start in the ducts or lobules.
Breast cancer can spread through blood vessels and lymphatic vessels outside the breast. When breast cancer spreads to other parts of the body, it is said to have metastasized

Why the choice of breast cancer?

The most commonly occurring cancer in women is breast cancer (BrC). As claimed by the World Health Organization (WHO), BrC was diagnosed in 2.3 million women, and 685,000 deaths were recorded globally in 2020 (fig.1) [1]. In addition, the WHO predicts that the number of new BrC patients will increase by seventy percent (70%) in the next twenty years. Besides, BrC is the 5th-most deadly disease out of distinct cancer types, such as lung, colorectal, liver, and stomach cancers [2]. which further states that in 2030 the death ratio from cancer is expected to increase up to 27 million. So, on-time and accurate detection, early diagnosis, and active prevention are critical requirements for reducing mortality rate among women.

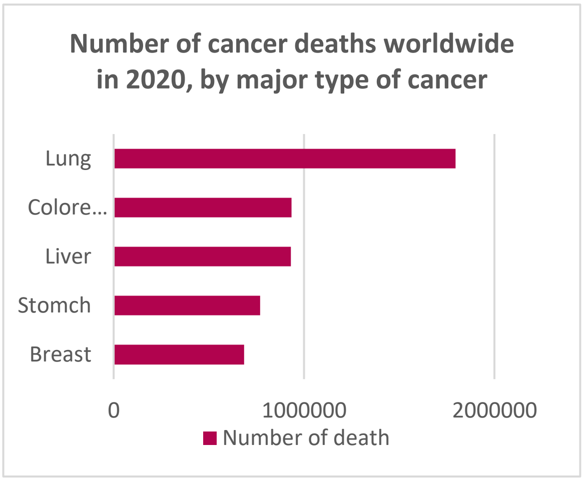


Figure 2- Number of cancer deaths worldwide in 2020, by major type of cancer

The breast cancer in Egypt.

In Egypt, breast cancer is the most common malignancy in women, accounting for 38.8% of cancers in this population, with the estimated number of breast cancer cases nearly 22,700 in 2020 and forecasted to be approximately 46,000 in 2050. It is estimated that the breast cancer mortality rate is around 11%, being the second cause of cancer-related mortality after liver cancer [3].

Age and stage at diagnosis of breast cancer in Egypt were compared from the Gharbia Cancer Registry (GCR) with the U.S. Surveillance, Epidemiology, and End Results (SEER) Program database (for 2004–2008). It was found that GCR cases were a full decade younger than that SEER cases (mean age = 51.0 vs 61.4 years), with nearly 19% of GCR cases aged ≤ 40, compared with only 6% of SEER cases. Moreover, there was a clear difference in stage at diagnosis between the two populations. GCR cases were diagnosed at more advanced stages, with nearly 5%, 39%, 44%, and 12% diagnosed at stage I, II, III, or IV, respectively, compared with 48%, 34%, 12%, and 5% of SEER cases diagnosed at stage I, II, III, or IV, respectively (Fig.2) [4].

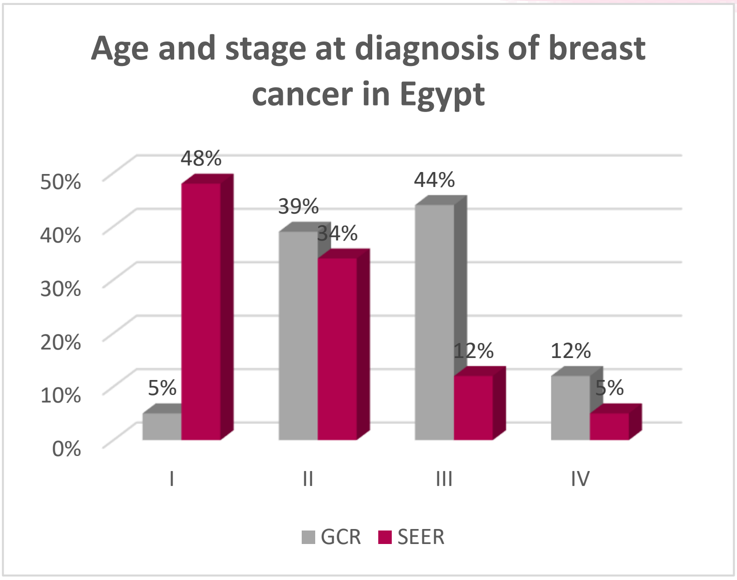


Figure 2- Age and stage at diagnosis of breast cancer in Egypt

Otherwise, in Egypt, most women don't practice a total checkup every 6 months, due to poverty and financial status. This might be a reason for many women having breast cancer without their awareness, which will lead to late detection of breast cancer until there is no treatment available for it. This highlights the importance of taking the specific disease criteria into consideration when planning a breast cancer early detection program [5], so because of the great seriousness of this disease, we thought this project would help solve the crisis in Egypt.

3. Literature Review

Researchers have difficulties in making a diagnosis of BC disease. To identify breast cancer conditions, researchers employed datasets based on mammography (X-rays), magnetic resonance imaging (MRI), ultrasound (sonography), and thermography. Nowadays, many methods are addressed and applied to these kinds of imaging datasets for data classifications and breast cancer is no different. Methods like Machine Learning, transfer learning, and Deep Learning are addressed to datasets for accurate data classification. In our project, we will use a Deep learning algorithm called Convolution Neural Networks (CNNs) which is a derived deep learning algorithm from Artificial Neural Networks.

A study was conducted in February 2022 [6] to identify breast cancer in its early stages. This research provides a heterogeneous ensemble Machine Learning technique. The technique is based on the CRISP-DM process (DT). The suggested method leverages Stacking to create the ensemble model utilizing three separate algorithms: K-Nearest Neighbors (KNN) , Support Vector Machine (SVM) , and Decision Tree. The suggested collaborative model's accuracy rate is 78% at its peak.

Another study was conducted in February 2022 [7]. The team's objective was to develop a hierarchical breast cancer system model that would enhance detection precision and minimize breast cancer misdiagnosis. The dataset was treated to ANN and SVM in classifying breast cancer tumors and comparing their performances. The best classification accuracy was achieved by the SVM using radial features (91.6%), while the ANN achieved 76.6%. Therefore, SVM was employed to conclude the significance of breast screening. In the second phase, AlexNet, InceptionV3, and ResNet101 were trained by employing transfer learning. The results showed that InceptionV3 scored 91.3%, ResNet101 scored 85.51%, and AlexNet scored 81.16%.

In 2022 a research was published [8] In the book “Approaches and Applications of Deep Learning in Virtual Medical Care”, where a team of researchers used two techniques, namely FA-SVM and Box Count Method (BCM) in distinct processes that led to excellent results in certain sectors.
BCM is used to extract the features and the FD feature assesses the difficulty of the 42-image input dataset and then processes the FD generated using the SVM classifier which is used to determine whether the tumor is malignant or benign. The highest accuracy of this technique is 98.13%, But this result is not true since the model will be underfitting due to the insufficient amount of data and the absence of image augmentation techniques.

In 2016 a research paper was published [9] in which the authors presented an available set of 7,909 breast cancer histopathology images. The dataset consisted of a mixture of images of benign tumors and malignant tumors. The aim of this dataset is to automatically differentiate whether this tumor is malignant or benign with the use of multi-classifiers KNN, SVM, quadratic linear analysis, and RF. The accuracy of the results ranged between 80 to 85%, This accuracy is not bad but when it comes to human lives and medical diagnoses this range is not acceptable so, it still needs to be improved.

In February 2022 a research paper was published [10] in the book “Quantum Machine Intelligence”, in this paper researchers investigated a quantitative solution to a machine learning problem. They used transfer learning to train a set of traditional hybrid neural networks, their mission was to tackle BCDR's difficulty in identifying full-image mammograms as malignant or benign.
Data have been collected and utilized in this research to clarify the areas of mammography that were targeted by the networks while measuring different performance indicators. They also point out that some designs perform much better than others depending on the specific task. The highest accuracy they got according to the results was 84%.

In January 2022 a study was published [11] talking about Breast Cancer Detection Models Based on Transfer Learning. They introduced a framework based on the concept of transfer learning in their research. Furthermore, a variety of augmentation procedures, including multiple rotation combinations, scale, and shifting, were implemented to prevent a fitting problem and then generate consistent results by increasing the number of screened mammography pictures. The proposed solution was tested on the Screening mammography Image Analysis Society (MIAS) database, with an accuracy of 70% utilizing the NASNet-Mobile network.

In chapter 3 of the book "Machine Learning Methods and Applications to Brain Disorders" [12] the author talks about a comparison between an SVM that was trained on both human-made features and CNN-extracted features, and the results showed that the SVM that was trained on the CNN extracted features had an accuracy of 88% compared to the 85% of the human-made features, which opens the possibility to the usage of the same technology of CNN in the application in image classification for medical purposes.

in December 2020 a study was published [13] the author talks about several methods that are being used at the current time and their limitations such as the Naïve Bayes classifier, SVM classifier, and RCNN classifier and their limitations receptively were: produce bad results when the dataset is not represented, not effective with big datasets, takes a lot of time to train the network. These limitations led the author to suggest a new method using Deep Neural Networks with Support Value (DNNS) which intends to better the image quality and solve the parameters problems. The results show an accuracy of approximately 96%, which is a significant improvement with reference to the previously mentioned methods, whereas the new method suggests a solution to the limitations met by the other methods.

Table 1 shows a comparison between the different mentioned studies.

TABLE 1. Comparison and limitations of previous studies.

Publication	Model	Accuracy	Limitation
Saeed. S et al. [3]	SVM, BCM	98.13%	-Limited dataset
Nanglia S, et al. [1]	KNN, SVM, DT	78%	-Low accuracy -Recured hand crafted features
Scarpazza C, et al. [7]	SVM, CNN	88%	yet to be implemented for breast cancer.
Spanhol FA, et al. [4]	SVM	80%	-Low accuracy -Recured hand crafted features
Azevedo V, et al. [5]	TL	84%	-low accuracy
Lin RH, et al. [2]	SVM, ANN, TL / (AN, IV3, RESNET101)	91.6%, 76.6%, 81.16%, 91.3%, 85.51. Respectively	Unkown
Alruwaili M, et al. [6]	TL (NASNet-(Mobile	70%	-Low accuracy

Based on all of the previous information we came to the conclusion that:

The usage of convolution neural networks (CNN), a convenient data set that consists of approximately (16000) ultrasound images (both images of the full breast and the mask images of the tumor), and the methods of image augmentation should result in the best possible results.

4. Mathematical Modelling

3.1. CNN:

A convolutional neural network (CNN) is an artificial intelligence neural network that is used for deep learning and specialized in processing data that has a grid-like topology, such as images. CNNs are particularly useful for finding patterns in images to recognize objects.[14]
The CNN usually has three layers: The convolutional layer, the pooling layer, and the fully connected layer.

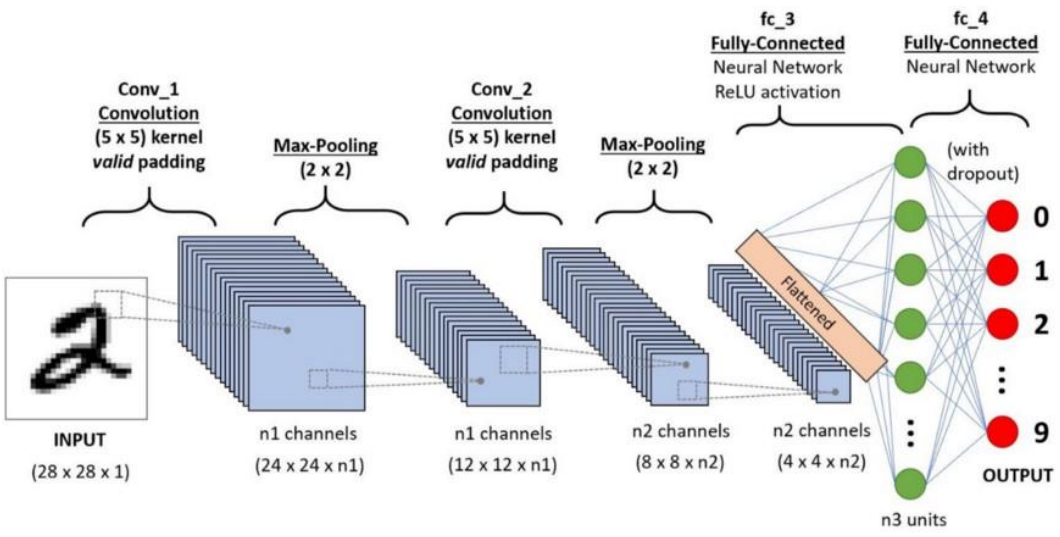


Figure 3- A CNN sequence to classify handwritten digits

i. Kernel:

In CNNs, the kernel is a filter that is used to extract the features from the images. The kernel is a matrix that moves over the input data, performs dot product with the sub-region of the input data, and gets the output as the matrix of dot products.[15]
Output size = [i – k] + 1
Where i is the size of the input, k is the size of the kernel.

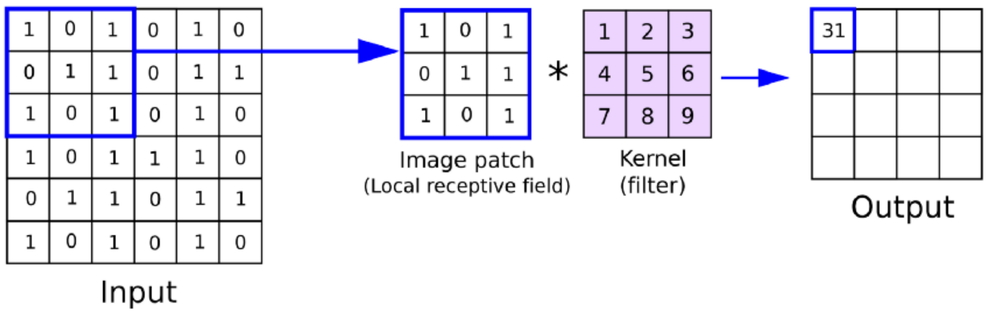


Figure 4 - Output of kernel with a size of 3x3

ii. Stride:

Stride is a parameter of the neural network's filter that modifies the amount of movement over the image. If we have a stride 1, it will take one by one. If we give the stride 2, it will take the value by skipping the next 2 pixels. [16]

Output size = $\left[\mathrm{i} - \frac{\mathrm{k}}{\mathrm{s}}\right] + 1$

Where is the size of the input, k is the size of the kernel, and s is the stride value.

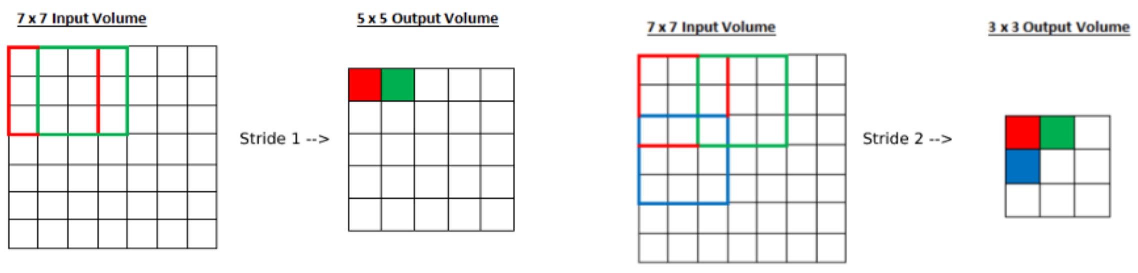


Figure 5- Output of stride value of 1 (Left figure) and output of stride value of 2 (Right figure)

iii. Padding:

Padding is a term relevant to CNNs, it refers to the number of pixels added to an image when it is being processed by the kernel of the CNN. If the padding value is set to zero, then every pixel value that is added will be zero. When we use the filter to scan the image, the size of the image will be smaller, we need to avoid that because we have to preserve the original size of the image to extract low-level features, so we add some extra pixels outside the image which is called the zero-padding in CNNs.

Output size = $\left[\mathrm{i} - \mathrm{k} + \frac{2\mathrm{p}}{\mathrm{s}}\right] + 1$

Where i is the size of the input, k is the size of the kernel, s is the stride value, and p is the amount of padding.

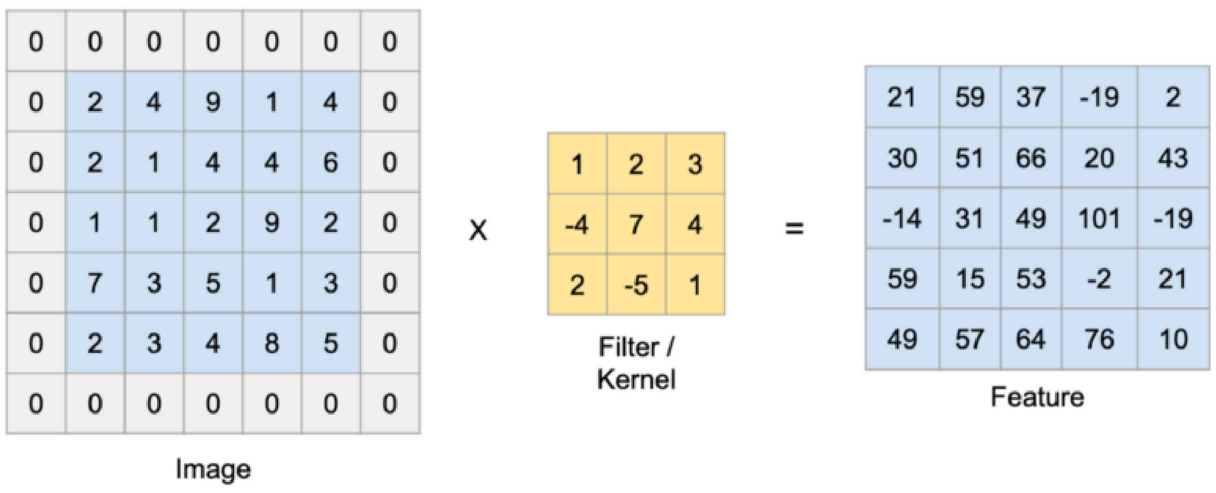


Figure 6 - Output of kernel with a size of 3x3 after using padding technique

iv. Pooling:

Pooling is a technique for generalizing features extracted by convolutional filters and helping the network recognize features independent of their location in the image, it is used to reduce the image size by mapping a patch of pixels to a single value. There are different types of pooling, in our model we are using max-pooling.

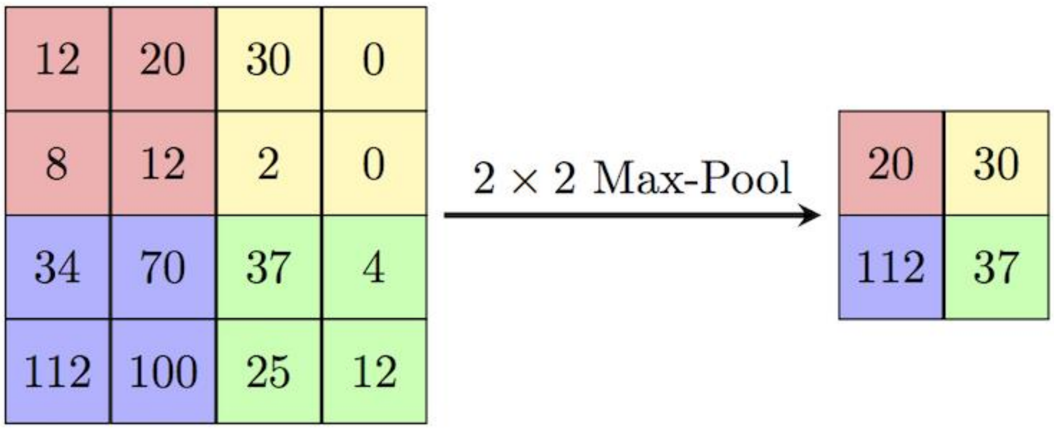


Figure 7 - Output of using 2x2 max-pooling method

v. Layers used to build CNN:

v.1.Convolutional layer

The convolutional layer is the core building block of the CNNs. It carries the main portion of the network's computational load. This layer is the one responsible for performing the dot product between the kernel (The learnable parameters), and the input (The restricted portion of the respective field).
It is used to extract the various features from the input images. In this layer, we use the kernel method to extract features from the inputs. [17]
If we have an input of size W x W x D, and D number of Kernels with a spatial size of F with stride S and amount of padding P, then the size of the output volume can be determined by the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

v.2. Pooling layer

The pooling layer is responsible for replacing the output of the network at certain locations deriving summary statistics of the nearby outputs. This helps in reducing the spatial size of the representation and decreasing the required amount of computation and weights. This is performed by decreasing the connections between layers independently operating on each feature map. [18]

If we have an input of size W x W x D, and D number of pooling Kernels with a spatial size of F, then the size of the output volume can be determined by the following formula:

$$W_{out} = \frac{W - F}{S} + 1$$

3.2. Forward propagation & Backpropagation:

To be trained, a neural network relies on both forward and backward propagation. Backpropagation is used in machine learning and data mining to improve prediction accuracy through backward propagation calculated derivatives. Backward Propagation is the process of moving from the right (output layer) to the left (input layer). Forward propagation is the way data moves from left (input layer) to right (output layer) in the neural network. [19]

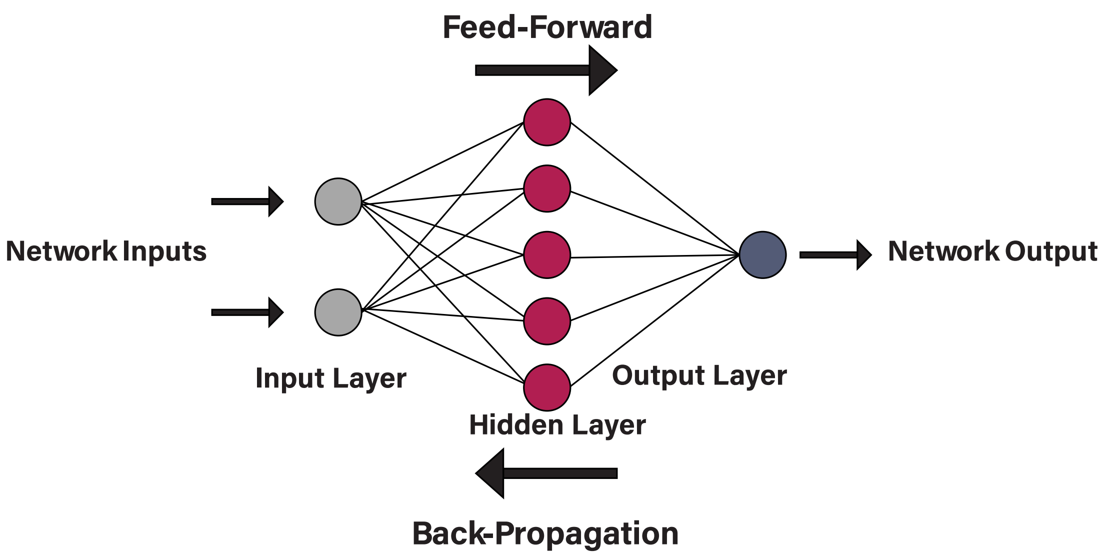


Figure 8 - A simple neural network with a one hidden layer

3.2.1. Forward propagation:

To generate some output, the input data should be fed in the forward direction only. The data should not flow in a reverse direction during output generation otherwise it would form a cycle and the output could never be generated. So, we use forward propagation.

Forward propagation is where input data is fed through a network in a forward direction to generate an output. The data is accepted by hidden layers, processed as per the activation function, and passed to the successive layer. The forward flow of data is designed to avoid data moving in a circular motion, which does not generate an output.[20]

At each neuron in a hidden or output layer, there are 2 steps:

- Preactivation: it is a weighted sum of inputs i.e., the linear transformation of weights w.r.t to inputs available. Based on this aggregated sum and activation function the neuron decides whether to pass this information further or not.
- Activation: the calculated weighted sum of inputs is passed to the activation function. An activation function is a mathematical function that adds non-linearity to the network.
- The common types of activation functions are:
    - ReLU: f(x)=max (0, x)
    - Sigmoid: $y = \frac{1}{1+e^{-x}}$
    - Tanh

- **Math for Forward Propagation:**

Assuming the input of a neural network is a vector x^i, the output is ŷ.
Let's assume our network to be like the following figure:

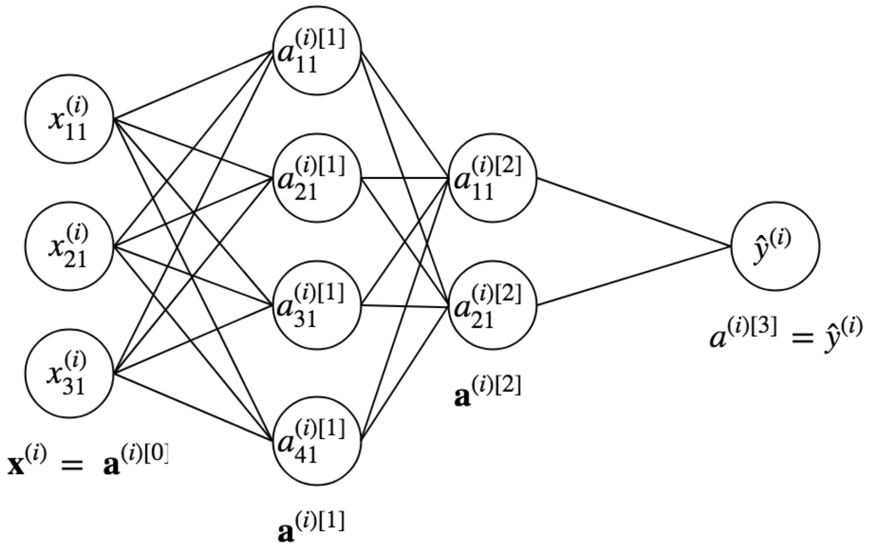


Figure 9 - A neural network with 2 hidden layers

We have hidden layers $a^{(i)[1]}$, and $a^{(i)[2]}$, and output layer $a^{(i)[3]} = \hat{y}^{(i)}$ represented. And our 3-layer hidden network is defined in our diagram.

What happens is the following:

$$x^{(i)} \rightarrow a^{(i)[1]} \rightarrow a^{(i)[2]} \rightarrow \hat{y}^{(i)}$$

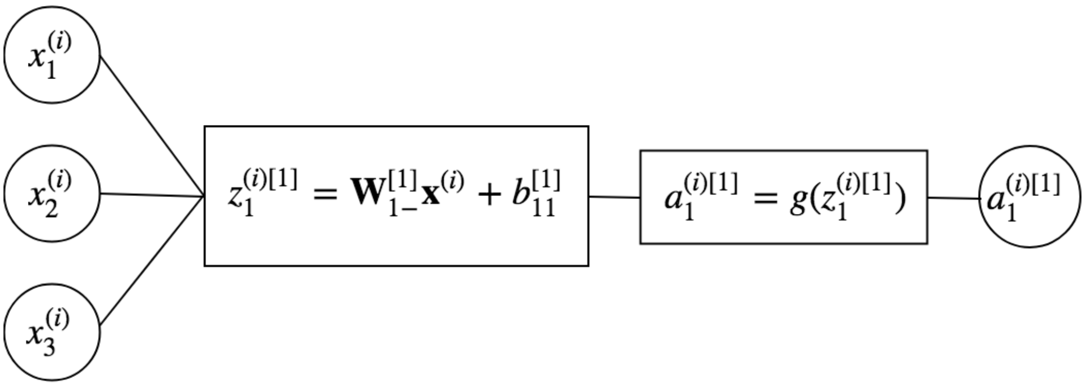
To calculate $a_1^{(i)[1]}$, we take each entry from $x^{(i)}$ and multiply it by weight. Let's say we have $W_{13}^{[1]}$. We multiply this term by the third entry in x, which is $x_3^{(i)}$ to get the first entry in the 1st layer.

The weights corresponding to $a_1^{(i)[1]}$ in our diagram is the matrix $W^{[1]}$:

$$W^{[1]} = \begin{bmatrix} w_{11}{}^{[1]} & w_{12}{}^{[1]} & w_{13}{}^{[1]} \\ w_{21}{}^{[1]} & w_{22}{}^{[1]} & w_{23}{}^{[1]} \\ w_{31}{}^{[1]} & w_{32}{}^{[1]} & w_{33}{}^{[1]} \\ w_{41}{}^{[1]} & w_{42}{}^{[1]} & w_{43}{}^{[1]} \end{bmatrix}$$

We will only use the first row to calculate $a_1^{(i)[1]}$

$$W_{1-}{}^{[1]} = [w_{11}{}^{[1]} \quad w_{12}{}^{[1]} \quad w_{13}{}^{[1]}]$$



$x_1^{(i)}$
$x_2^{(i)}$
$x_3^{(i)}$
$z_1^{(i)[1]} = \mathbf{W}_{1-}^{[1]}\mathbf{x}^{(i)} + b_{11}^{[1]}$
$a_1^{(i)[1]} = g(z_1^{(i)[1]})$
$a_1^{(i)[1]}$

Figure 10 - The equation to get the output $a^{(i)[1]}$

We will add a bias $b_1^{[1]}$ to $W_{1-}^{[1]}\ x^{(i)}$. The bias are other parameters besides the weights that our model learns.

Before mentioning the reason why we need to use a bias, we first should explain our activation function g(). Different neural networks use different activation functions. Let's see one of them which is the ReLU function.

It's defined as:

$$g(z) = \begin{cases} z & if\ z > 0 \\ 0 & otherwise \end{cases}$$

The role of the activation function is to introduce non-linearity into the neural network. Without activation functions, neural networks would simplify to linear functions.

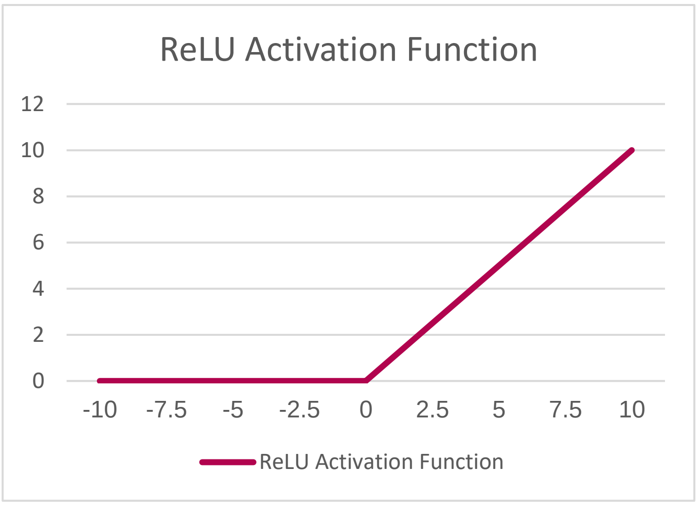


Figure 11 - Plot of ReLU Activation Function

Input → 1st hidden layer:

Returning to our network, the first step is to get the output of the 1st hidden layer as follows:
We start by multiplying $x^{(i)}$ by the weights and bias of the first hidden layer, $W^{[1]}$ and $b^{[1]}$ to get $z^{(i)[1]}$.

$$z^{(i)[1]} = W^{[1]} x^{(i)} + b^{[1]}$$

We start by multiplying $x^{(i)}$ by $W^{[1]}$ then we add the bias $b^{[1]}$. The dimensions of the bias $b^{[1]}$ match the dimensions of $z^{(i)[1]}$. when we get the activity matrix $z^{(i)[1]}$ we apply the activation function to each element in $z^{(i)[1]}$.
So, we get: $a^{(i)[1]} = g(z^{(i)[1]})$
This just indicates that the ReLU function g() is applied elementwise to $z^{(i)[1]}$ to get $a^{(i)[1]}$.

1st hidden layer → 2nd hidden layer:

In this step, we do almost the same steps we did in the previous step. We start by multiplying $a^{(i)[1]}$ by $W^{[2]}$ then we add the bias $b^{[2]}$. The dimensions of the bias $b^{[2]}$ match the dimensions of $z^{(i)[2]}$. Once we get the activity matrix $z^{(i)[2]}$, we apply the ReLU activation function to each element in $z^{(i)[2]}$.
So, we get: $a^{(i)[2]} = g(z^{(i)[2]})$

2nd hidden layer → Output:

Now we have $a^{(i)[2]}$, we again start by multiplying $a^{(i)[2]}$ by $W^{[3]}$, then we add the bias $b^{[3]}$. when we get the activity matrix $z^{(i)[3]}$, we apply the activation function to each element in $z^{(i)[3]}$. Since this is the final layer of our neural network, we will use a sigmoid activation function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The sigmoid activation function is rarely used in modern neural networks because it suffers from the vanishing gradient problem, but it is often used as the final activation function before the output. The reason is that it can squash values to be between 0 and 1.

So, we get: $\hat{y}^{[i]} = a^{(i)[3]} = \sigma(z^{(i)[3]})$

Now that we have the forward propagation figured out, we can generate a prediction $\hat{y}^{[i]}$ given a feature vector for the ith training example $x^{(i)}$.

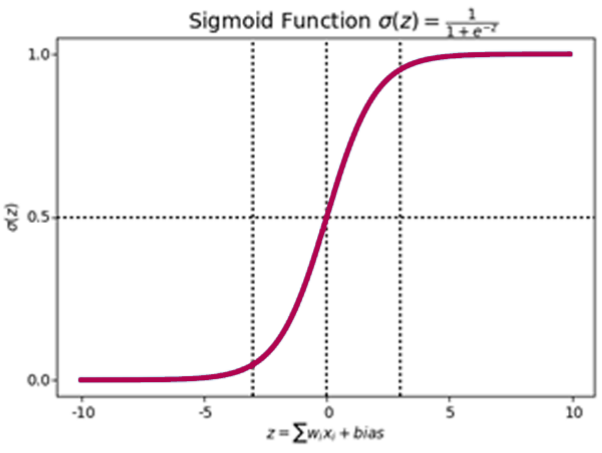


Figure 12 - Plot of Sigmoid Function

3.2.2. Backpropagation:

It is not possible to classify non-linear data using a normal linear model, but we can use feature crossing and create linear features from these non-linear ones. These are used in kernel methods of machine learning. In these methods, we need to tell the machine how to devise a feature that can be easily used to convert the non-linear problem to a linear one.

We usually face non-linear problems so when we find it hard to devise the feature crossing for the classification of the following classes- we use neural networks as they can come up with a non-linear equation that is fit to serve as a boundary between non-linear classes.

The neural network can achieve this equation using activation functions which are the units of non-linearity. They are used at every layer in a neural network. In Neural Network we stack the layers one over the other then we obtain complex functions using cascaded functions F(f(x)) as mentioned in the forward propagation section. [21]

- **Math for Backpropagation**

The error generated is backpropagated from the same nodes and same edges through which forward propagation takes place and reaches from input edges from the output node.

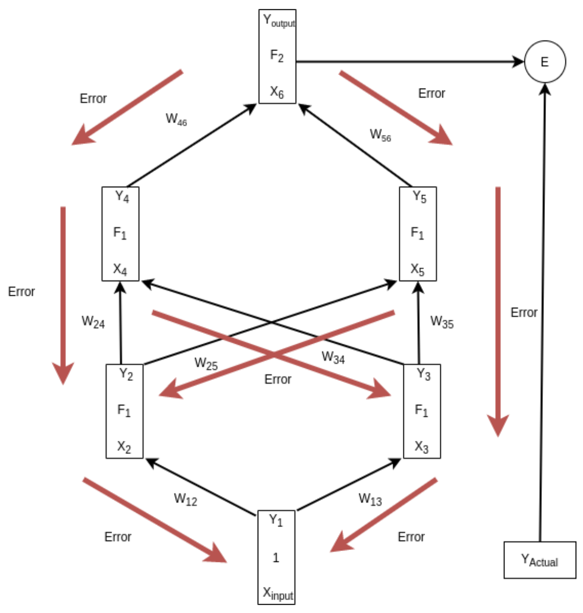


Figure 13 - Neural Network diagram.

The first step, the output node,

$$\frac{\partial E}{\partial Y_{output}} = (Y_{Actual} - Y_{output})$$

This is the derivative of the error with respect to the Y output at the final node.

$$\frac{\partial E}{\partial X_6} = \frac{\partial E}{\partial Y_{output}} \cdot \frac{\partial Y_{output}}{\partial X_6}$$

$$\frac{\partial Y_{output}}{\partial X_6} = \dot{F}_2(X_6)$$

From chain rule.

Now consider F_2 as sigmoid function,

$$\dot{F}_2(X) = F_2(X).\left(1 - F_2(X)\right).\left(Y_{Actual} - Y_{output}\right) = P$$

For sigmoid $\dot{F}(x)$ has that property.

Now we need to find $\frac{\partial E}{\partial W_{56}}$

$$\frac{\partial E}{\partial W_{56}} = \frac{\partial E}{\partial X_6} \cdot \frac{\partial X_6}{\partial W_{56}} \quad , \quad X_6 = W_{56}.Y_5 + W_{46}.Y_4 + B_6$$

$$\frac{\partial X_6}{\partial W_{56}} = Y_5$$

Using the chain rule, we obtain the values:

$$\frac{\partial E}{\partial W_{56}} = Y_5 . P \ ,$$

Similarly,

$$\frac{\partial E}{\partial W_{46}} = Y_4 . P$$

We try to calculate $\frac{\partial E}{\partial Y_5}$ so that we could move to the next level.

$$\frac{\partial E}{\partial Y_5} = \frac{\partial E}{\partial X_6} . \frac{\partial X_6}{\partial Y_5}$$

$$\frac{\partial E}{\partial Y_5} = W_{56} . P$$

Similarly,

$$\frac{\partial E}{\partial Y_4} = W_{46} . P$$

Now we go for the change in error for the change in input for nodes 5 and 4.

$$\frac{\partial E}{\partial X_5} = \frac{\partial E}{\partial Y_5} . \frac{\partial Y_5}{\partial X_5}$$

$$Y_5 = F_1(X_5)$$

$$\frac{\partial Y_5}{\partial X_5} = \dot{F}_1(X_5)$$

Then,

$$\frac{\partial E}{\partial X_5} = W_{56} . P . \dot{F}_1(X_5) = q$$

Similarly,

$$\frac{\partial E}{\partial X_4} = W_{46} . P . \dot{F}_1(X_4) = r$$

Now we can calculate the change in error with the change in weights using the same method we used for W_{56}

$$\frac{\partial E}{\partial W_{24}} = Y_2 . r$$

$$\frac{\partial E}{\partial W_{34}} = Y_3 . r$$

$$\frac{\partial E}{\partial W_{25}} = Y_2 . q$$

$$\frac{\partial E}{\partial W_{35}} = Y_3 . q$$

These are the changes of error with a change in the weights of edges. Now, calculate for Y_2 and Y_3. But one thing to notice is when we are going to calculate the change in error with a change in Y_2 and Y_3 from backpropagation, they will be affected by both the edges from Y_5 and Y_4.

So, the change will be a sum of the effect of change in node 4 and node 5.

We can calculate the effects in a similar way we calculated $\frac{\partial E}{\partial Y_5}$,

$$\frac{\partial E}{\partial Y_2} = W_{24}.r + W_{25}.q$$

$$\frac{\partial E}{\partial Y_3} = W_{34}.r + W_{35}.q$$

We can generalize it to:

$$\frac{\partial E}{\partial Y_i} = \sum W_{ij} . \frac{\partial E}{\partial x_j}$$

where the ith node is in the Lth layer and the jth node is at the (L+1)th layer.

Now, once we find, the change in error with a change in weight for all the edges. We can update the weights and start learning for the next epoch using the formula.

$$W_{new} = W_{old} - \propto . \frac{\partial E}{\partial W_{old}}$$

Where alpha is the learning rate. This is how the backpropagation algorithm works.

4. Experimental work

4.1. Exploring data:

We are using the "Breast Ultrasound Images Dataset" (Dataset BUSI) which is collected by Dr. Aly Fahmy. The data collected in baseline include breast ultrasound images among women in ages between 25 and 75 years old. It was collected in 2018. The number of patients is 600 female patients, and it consists of 780 images. [22]

The data is categorized into three classes: Normal, benign, and malignant.

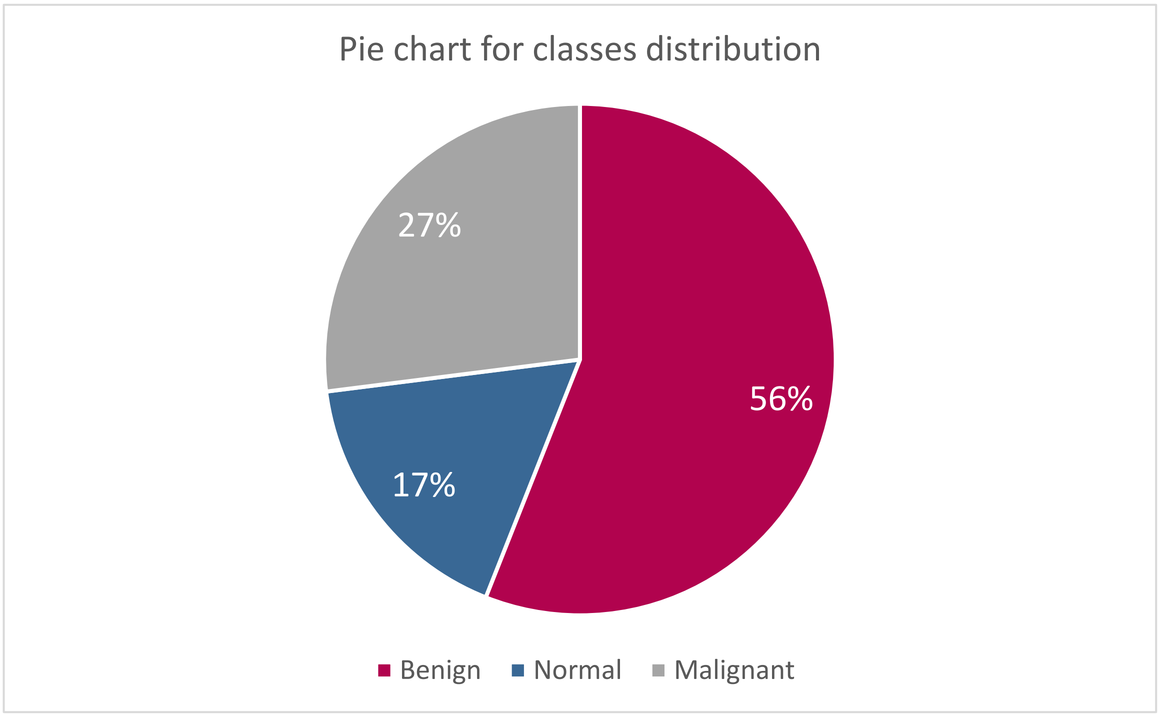


Figure 14 - Pie chart for classes distribution

4.2. Visualizing data:

As shown in the figure, each image has its segmentation mask, segmentation mask is a specific portion of an image that is isolated from the rest of an image, the output of the segmentation mask can be used to copy exact areas of an image that have been assigned a label in the deep learning model.

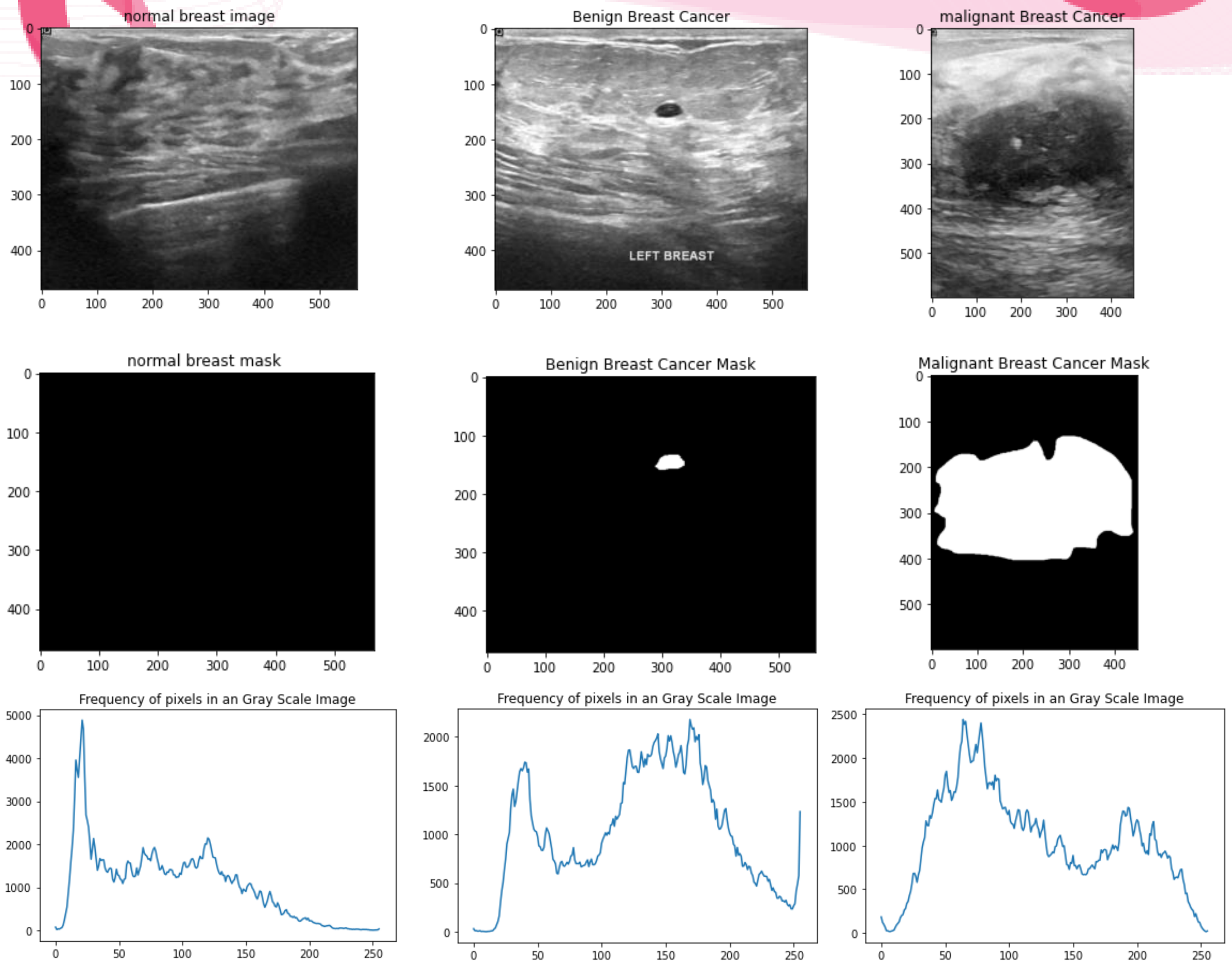

normal breast image
Benign Breast Cancer
malignant Breast Cancer
LEFT BREAST
normal breast mask
Benign Breast Cancer Mask
Malignant Breast Cancer Mask
Frequency of pixels in an Gray Scale Image
Frequency of pixels in an Gray Scale Image
Frequency of pixels in an Gray Scale Image

Figure 15 - Train samples with mask and frequency of pixels in a greyscale image

4.3. Preprocessing data:

While exploring the data, we found that the data is heavily unbalanced and very small in size, which may cause the model to underfit the data.

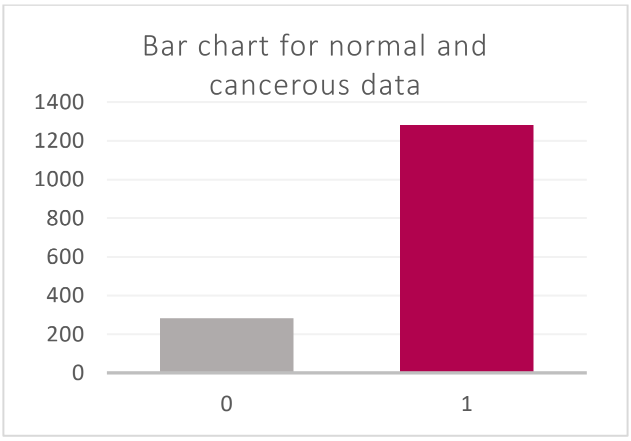


Figure 16 - Bar chart for normal and cancerous data

In order to solve this problem, we will use the image augmentation techniques using Keras, the image augmentation is a technique of altering the existing data images to create more data for the model training process using many simple processes such as flipping images and changing their sizes.[23]

In our dataset, we used the following processes:

- width_shift_range (It shifts the image horizontally shifts)
- height_shift_range (It shifts the image vertically shifts)
- shear_range (It distorts the image along an axis)
- zoom_range (It zooms in the image)
- horizontal_flip (It flips the image horizontally)
- rotation_range (It rotates the image)
- fill_mode (It fills points outside the boundaries according to a given mode)

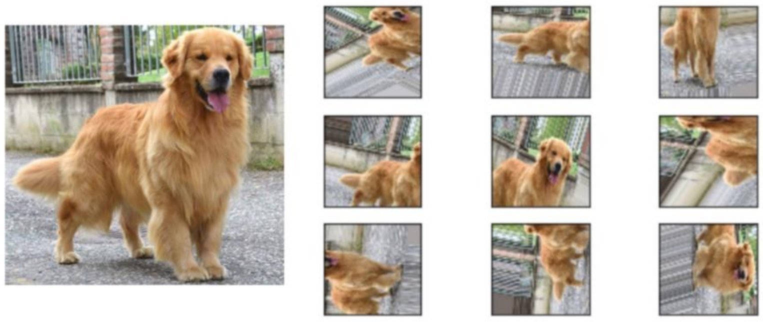


Figure 17 - Augmentation sample

After the augmentation process, the dataset is bigger in size as shown in the figure.

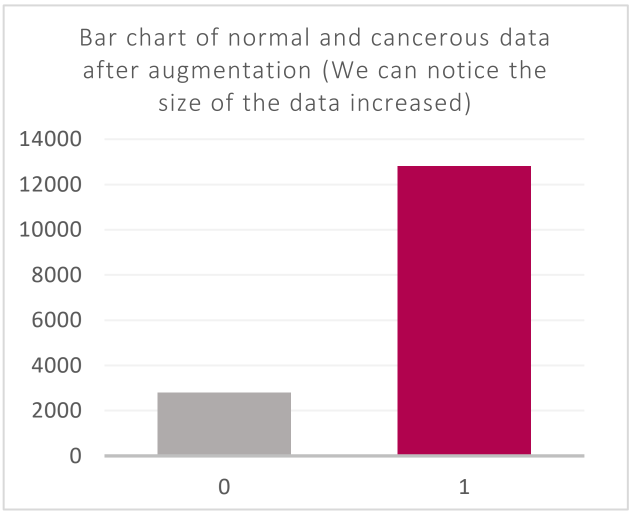


Figure 18 - Bar chart of normal and cancerous data after augmentation (We can notice the size of the data increased)

4.4. Building the model:

The input shape is resized to be (64, 64, 3). The size of each image is (64, 64), and the is the number of layers.
The architecture of the model consists of the following:

- 10 Conv2D layers:
  - Conv2D is a 2D convolutional layer, which creates a convolutional kernel that is wind with layers input which helps produce a tensor of outputs.
  - Each Conv2D layer has:
    - Activation layer (ReLU)
    - Maxpooling2D with pool size (2, 2)
    - Padding with the same size as the input size
  - The second, fourth, seventh, and tenth layers have stride with size (2, 2)

- Flatten layer:
  - Flatten layer is used to make the multidimensional input a one-dimensional output.

- Dense layer:
  - Dense layer is used to classify an image based on output from convolutional layers by changing the dimensions of the input vector.

- Dropout
  - Dropout refers to dropping out the nodes (inputs and hidden layers) in the neural network.

- Dense layers of 4096 units.
- Activation function (Sigmoid function) for the output layer

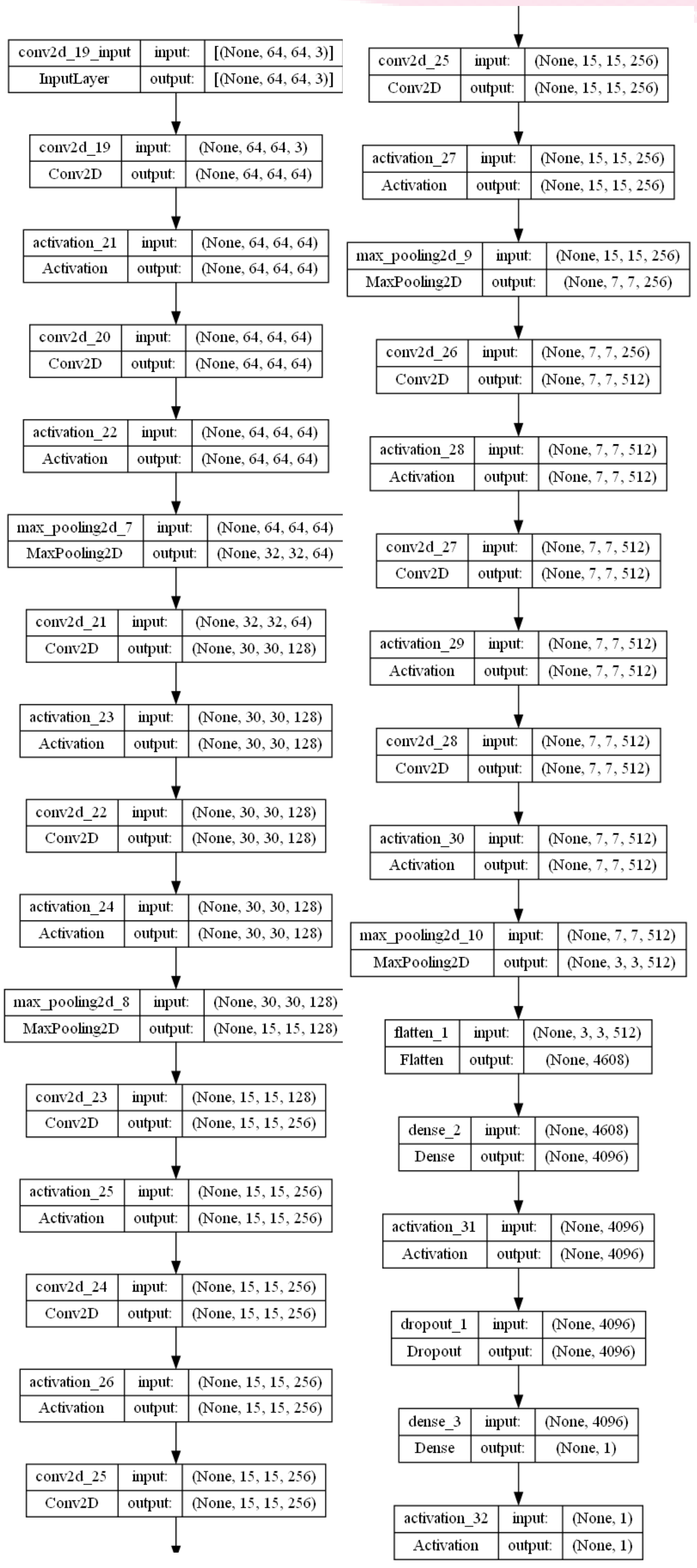

conv2d_19_input InputLayer input: [(None, 64, 64, 3)] output: [(None, 64, 64, 3)]
conv2d_19 Conv2D input: (None, 64, 64, 3) output: (None, 64, 64, 64)
activation_21 Activation input: (None, 64, 64, 64) output: (None, 64, 64, 64)
conv2d_20 Conv2D input: (None, 64, 64, 64) output: (None, 64, 64, 64)
activation_22 Activation input: (None, 64, 64, 64) output: (None, 64, 64, 64)
max_pooling2d_7 MaxPooling2D input: (None, 64, 64, 64) output: (None, 32, 32, 64)
conv2d_21 Conv2D input: (None, 32, 32, 64) output: (None, 30, 30, 128)
activation_23 Activation input: (None, 30, 30, 128) output: (None, 30, 30, 128)
conv2d_22 Conv2D input: (None, 30, 30, 128) output: (None, 30, 30, 128)
activation_24 Activation input: (None, 30, 30, 128) output: (None, 30, 30, 128)
max_pooling2d_8 MaxPooling2D input: (None, 30, 30, 128) output: (None, 15, 15, 128)
conv2d_23 Conv2D input: (None, 15, 15, 128) output: (None, 15, 15, 256)
activation_25 Activation input: (None, 15, 15, 256) output: (None, 15, 15, 256)
conv2d_24 Conv2D input: (None, 15, 15, 256) output: (None, 15, 15, 256)
activation_26 Activation input: (None, 15, 15, 256) output: (None, 15, 15, 256)
conv2d_25 Conv2D input: (None, 15, 15, 256) output: (None, 15, 15, 256)
conv2d_25 Conv2D input: (None, 15, 15, 256) output: (None, 15, 15, 256)
activation_27 Activation input: (None, 15, 15, 256) output: (None, 15, 15, 256)
max_pooling2d_9 MaxPooling2D input: (None, 15, 15, 256) output: (None, 7, 7, 256)
conv2d_26 Conv2D input: (None, 7, 7, 256) output: (None, 7, 7, 512)
activation_28 Activation input: (None, 7, 7, 512) output: (None, 7, 7, 512)
conv2d_27 Conv2D input: (None, 7, 7, 512) output: (None, 7, 7, 512)
activation_29 Activation input: (None, 7, 7, 512) output: (None, 7, 7, 512)
conv2d_28 Conv2D input: (None, 7, 7, 512) output: (None, 7, 7, 512)
activation_30 Activation input: (None, 7, 7, 512) output: (None, 7, 7, 512)
max_pooling2d_10 MaxPooling2D input: (None, 7, 7, 512) output: (None, 3, 3, 512)
flatten_1 Flatten input: (None, 3, 3, 512) output: (None, 4608)
dense_2 Dense input: (None, 4608) output: (None, 4096)
activation_31 Activation input: (None, 4096) output: (None, 4096)
dropout_1 Dropout input: (None, 4096) output: (None, 4096)
dense_3 Dense input: (None, 4096) output: (None, 1)
activation_32 Activation input: (None, 1) output: (None, 1)

Figure 19 - Deep learning model architecture

4.5. Training the model:

After building the deep learning model architecture, we are going to split the data into training and validation data using scikit-learn train test split, then we fit the training data to our model. [24]

We start the training process with the compilation process, which is a built-in function responsible for configuring the model for training with different arguments as follows:

- Loss:
  - Loss function is the function used for backpropagation. It computes the quantity that a model should seek to minimize during training.
  - binary_crossentropy is responsible for computing the cross-entropy loss between true labels and predicted labels.
  - Binary cross-entropy function is described by the following equation:

    $$J(\vec{w},\ b) = -\frac{1}{m_{train}} \sum_{i=1}^{m_{train}} y^{(i)}.\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right).\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)$$

    Where $J(\vec{w},\ b)$ is the loss function, m_{train} is the training examples, $y^{(i)}$ is the true label, $f_{\vec{w},b}$ is the sum of weight and bias, and $x^{(i)}$ is the input.
- Optimizer:
  - Optimizers are used to change the attributes of the neural network such as weight and learning rate to reduce the losses.
  - Adam works as a replacement optimization algorithm for stochastic gradient descent for training deep learning models.
- Metric:
  - Metric function is used to judge the performance of the model, it is similar to loss function, except that the results from evaluating a metric are not used when training the model. Note that you may use any loss function as a metric.

After the compilation process, we will use the fit method that is responsible for training the model for a fixed number of epochs (iteration on a dataset) with different arguments as follows:

- x_train:
  - Numpy arrays of input processed images.
- y_train:
  - Output labels.
- batch_size:
  - The number of examples to be trained utilized in one iteration.
- verbose:
  - It represents the choice that how you want to see the output of your neural network while it is training, when it's set to 1 it is represented as a progress bar
- epochs:
  - The number of how many times you go iterate on the training dataset.
- validation_data:
  - Data on which to evaluate the loss and any model metrics at the end of each epoch. The model will not be trained on this data.
- shuffle:
  - A parameter that is used to determine whether to shuffle the training data before each epoch or not.

4.6. Test cases:

We tried two architectures: the one mentioned in figure (19), and another one with only three Conv2D layers, and we used the data before doing the augmentation techniques, which resulted in the following learning curves (Accuracy of 91.3%, and loss of 0.18 at the best epoch)

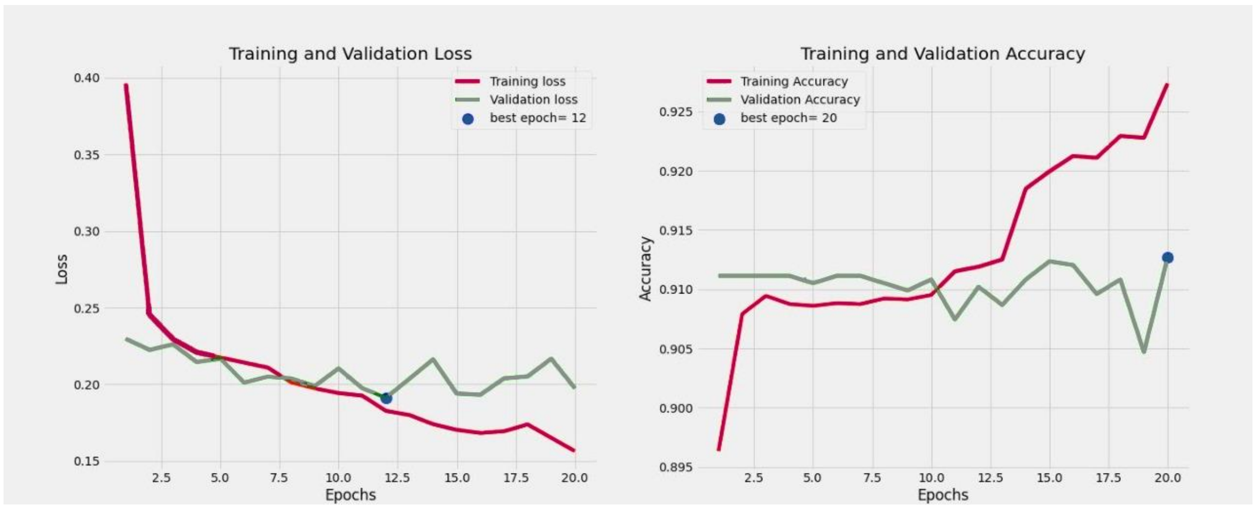


Figure 20 - Learning curves of test case (1)

The problem here is that the training accuracy and the validation accuracy were not converging, this may be due to the high variance of the data as a result of using training on small-size data as shown in figure (16).

After that, we did the augmentation techniques to increase the size of the data to avoid underfitting, which resulted in the following learning curves (Accuracy of 93.7%, and loss of 0.3 at the best epoch)

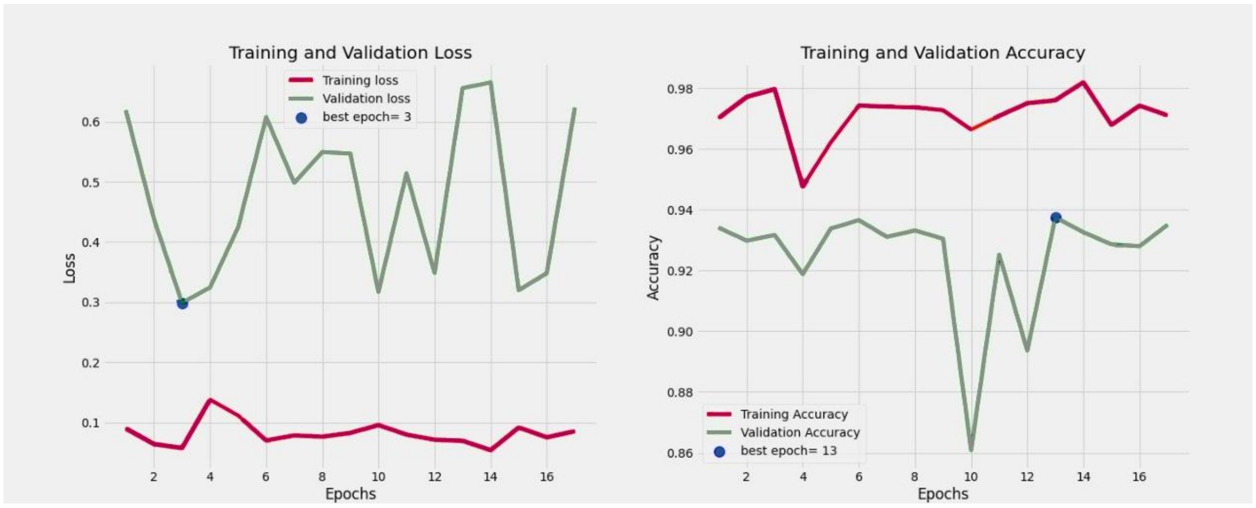


Figure 21 - Learning curves of test case (2)

The problem here is the same as above with the non-convergence of the training accuracy and validation accuracy, also the loss in the validation data is not converging, this may be due to the unbalanced dataset with the bias to the cancerous image dataset as shown in figure (18).

After that, we did the augmentation techniques to only the normal breast images dataset to balance the dataset as shown in the following figure:

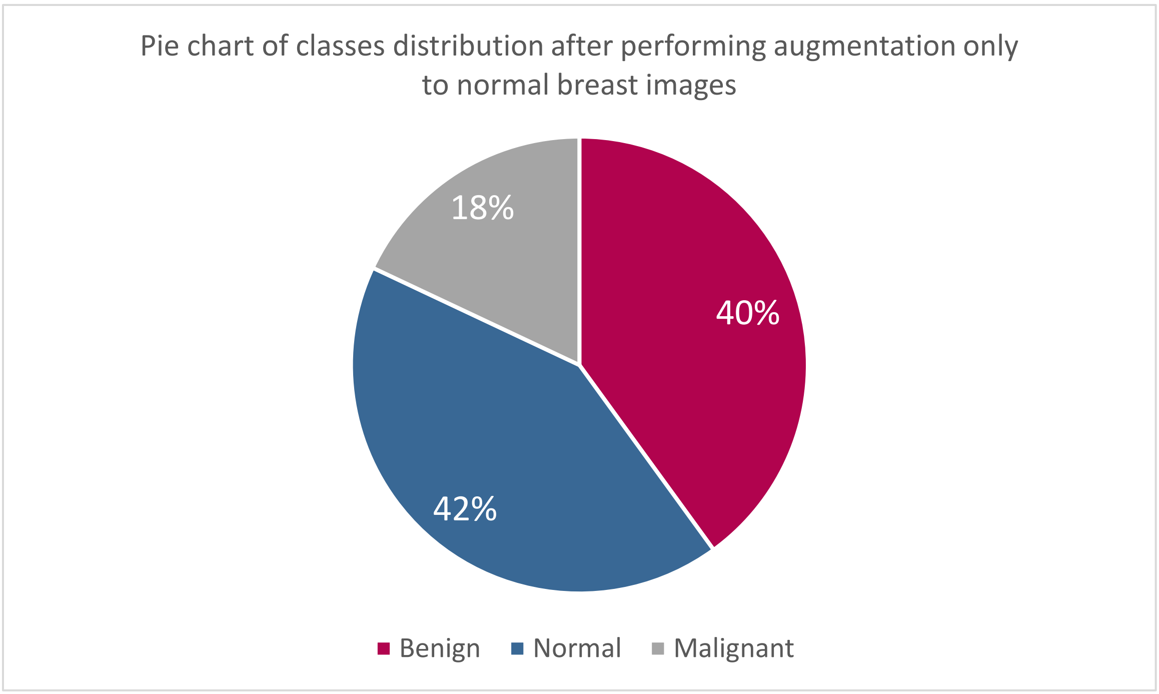


Figure 22 - Pie chart of classes distribution after performing augmentation only to normal breast images

This technique resulted in the following learning curves (Accuracy of 94.5%, and loss of 0.18 at the best epoch)

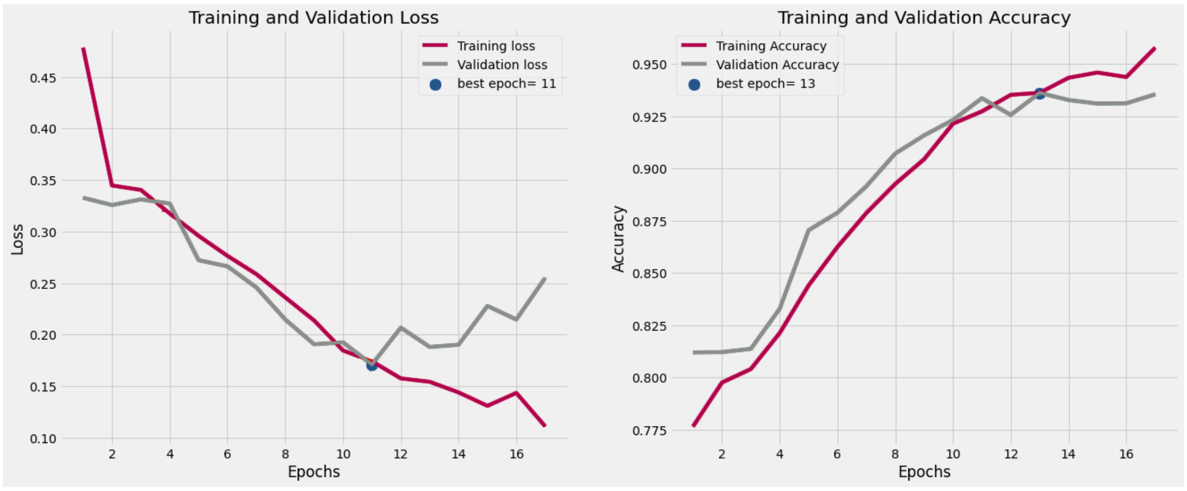


Figure 23 - Learning curves of test case (3)

As shown in the figure, the loss curves are converging to each other, and the training and validation accuracy curves are increasing and converging to each other. These results were positive, but we were seeking better training and validation accuracy, and we wanted the last portion of the training and validation loss curves to converge.

We then implemented the VGG16 architecture shown in figure (19), which used 10 Conv2D layers instead of only 3 Conv2D layers, we also trained the model for 20 epochs instead of 17 epochs. This new architecture resulted in the following learning curves (Accuracy of 95.2%, and loss of 0.15)

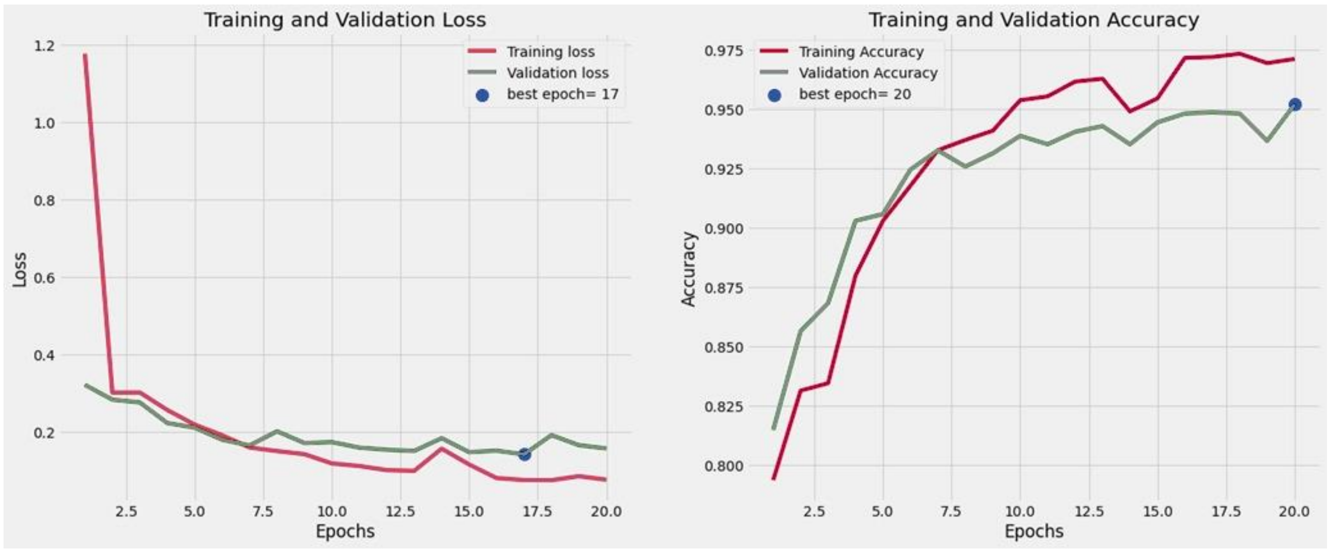
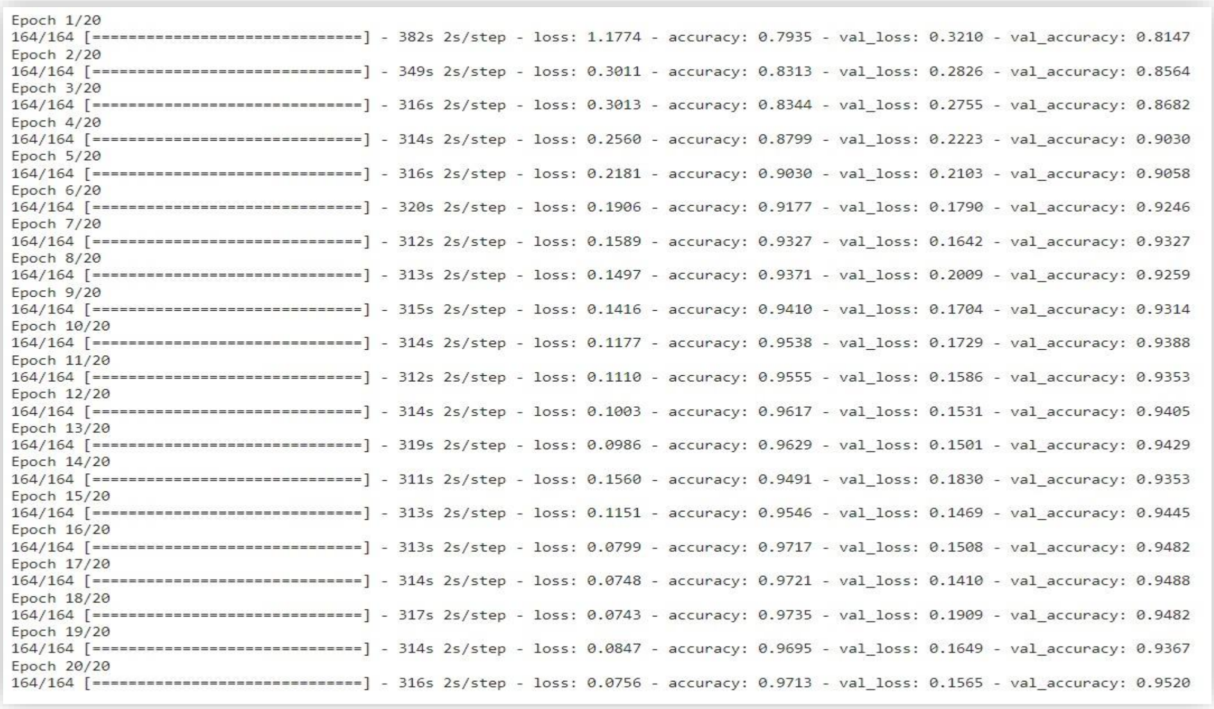


Figure 24 - Learning curves of test case (4)

As shown in the figure, the loss curves are converging to each other even in the last portion, and the training and validation accuracy are increasing and converging to each other. Also, the validation accuracy is higher than in test case (3), as well as for the training accuracy that reached 97.1%.

5. Results & Analysis

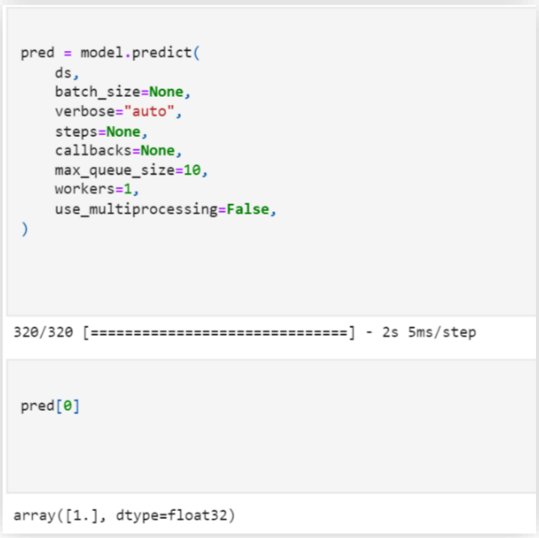
As shown in figure (24), after the third test case, we managed to reach an accuracy of 94.5% which is relatively the highest accuracy among different papers as mentioned in the literature review section. We also managed to make both loss curves and training and validation accuracy curves to be converging to each other.



```
Epoch 1/20
164/164 [==============================] - 382s 2s/step - loss: 1.1774 - accuracy: 0.7935 - val_loss: 0.3210 - val_accuracy: 0.8147
Epoch 2/20
164/164 [==============================] - 349s 2s/step - loss: 0.3011 - accuracy: 0.8313 - val_loss: 0.2826 - val_accuracy: 0.8564
Epoch 3/20
164/164 [==============================] - 316s 2s/step - loss: 0.3013 - accuracy: 0.8344 - val_loss: 0.2755 - val_accuracy: 0.8682
Epoch 4/20
164/164 [==============================] - 314s 2s/step - loss: 0.2560 - accuracy: 0.8799 - val_loss: 0.2223 - val_accuracy: 0.9030
Epoch 5/20
164/164 [==============================] - 316s 2s/step - loss: 0.2181 - accuracy: 0.9030 - val_loss: 0.2103 - val_accuracy: 0.9058
Epoch 6/20
164/164 [==============================] - 320s 2s/step - loss: 0.1906 - accuracy: 0.9177 - val_loss: 0.1790 - val_accuracy: 0.9246
Epoch 7/20
164/164 [==============================] - 312s 2s/step - loss: 0.1589 - accuracy: 0.9327 - val_loss: 0.1642 - val_accuracy: 0.9327
Epoch 8/20
164/164 [==============================] - 313s 2s/step - loss: 0.1497 - accuracy: 0.9371 - val_loss: 0.2009 - val_accuracy: 0.9259
Epoch 9/20
164/164 [==============================] - 315s 2s/step - loss: 0.1416 - accuracy: 0.9410 - val_loss: 0.1704 - val_accuracy: 0.9314
Epoch 10/20
164/164 [==============================] - 314s 2s/step - loss: 0.1177 - accuracy: 0.9538 - val_loss: 0.1729 - val_accuracy: 0.9388
Epoch 11/20
164/164 [==============================] - 312s 2s/step - loss: 0.1110 - accuracy: 0.9555 - val_loss: 0.1586 - val_accuracy: 0.9353
Epoch 12/20
164/164 [==============================] - 314s 2s/step - loss: 0.1003 - accuracy: 0.9617 - val_loss: 0.1531 - val_accuracy: 0.9405
Epoch 13/20
164/164 [==============================] - 319s 2s/step - loss: 0.0986 - accuracy: 0.9629 - val_loss: 0.1501 - val_accuracy: 0.9429
Epoch 14/20
164/164 [==============================] - 311s 2s/step - loss: 0.1560 - accuracy: 0.9491 - val_loss: 0.1830 - val_accuracy: 0.9353
Epoch 15/20
164/164 [==============================] - 313s 2s/step - loss: 0.1151 - accuracy: 0.9546 - val_loss: 0.1469 - val_accuracy: 0.9445
Epoch 16/20
164/164 [==============================] - 313s 2s/step - loss: 0.0799 - accuracy: 0.9717 - val_loss: 0.1508 - val_accuracy: 0.9482
Epoch 17/20
164/164 [==============================] - 314s 2s/step - loss: 0.0748 - accuracy: 0.9721 - val_loss: 0.1410 - val_accuracy: 0.9488
Epoch 18/20
164/164 [==============================] - 317s 2s/step - loss: 0.0743 - accuracy: 0.9735 - val_loss: 0.1909 - val_accuracy: 0.9482
Epoch 19/20
164/164 [==============================] - 314s 2s/step - loss: 0.0847 - accuracy: 0.9695 - val_loss: 0.1649 - val_accuracy: 0.9367
Epoch 20/20
164/164 [==============================] - 316s 2s/step - loss: 0.0756 - accuracy: 0.9713 - val_loss: 0.1565 - val_accuracy: 0.9520
```

Figure 25 - Training results

But we are not done yet, we trained our model on several classes of data which is why the model can predict the presence of a tumor in any type of data whether it is ultrasound or x-ray data. We kept looking for x-ray datasets to validate our claims and we managed to find a mammography dataset on Kaggle which could be used to validate the deep learning model working for different sources of images such as x-ray images.[25] We used the prediction method to predict whether the images are cancerous or not (All images were in fact cancerous), and the result was that the model predicted all data were cancerous as follows:



```
pred = model.predict(
    ds,
    batch_size=None,
    verbose="auto",
    steps=None,
    callbacks=None,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False,
)
```

```
320/320 [==============================] - 2s 5ms/step
```

```
pred[0]
```

```
array([1.], dtype=float32)
```

Figure 26 - The results of the model prediction to X-Ray dataset

6. Conclusion

It can be concluded that using artificial intelligence in detecting and classifying breast cancer in a more efficient way is a strong need for the biomedical field. As we have gone through this project, we found that using CNNs to analyze visual imagery data, then pooling the data to reduce images size by mapping a patch of pixels to a single value, and then using forward propagation and backpropagation to detect masks and filters in different images are all strong deep learning techniques and algorithms that can be further used in the diagnosis and detection of other different types of cancers and diseases.

7. Future work

We reached quite acceptable results. However, many things can be done to reach better results and validation accuracy, the following steps may be considered:

1. Collecting more data: One of the major problems that faced us is the limited amount of data which lead to high variance, we managed to solve this problem by using image augmentation techniques. However, collecting more segmented data will show more improvement in the learning curves and accuracy.
2. Using DCGANs (Deep Convolutional Generative Adversarial Networks): Using image augmentation is a technique to solve the problem of the limitation of the data, but a more efficient way to solve this problem is by using artificial intelligence to create new data (new images) using DCGANs. Generative Adversarial Networks (GANs) concept is one of the most interesting concepts in computer science nowadays. Two models are trained simultaneously by an adversarial process, and a generator (The artist) learns to create new images that look real and similar to the original images, while a discriminator (The art critics) learns to detect whether an image is real or fake.

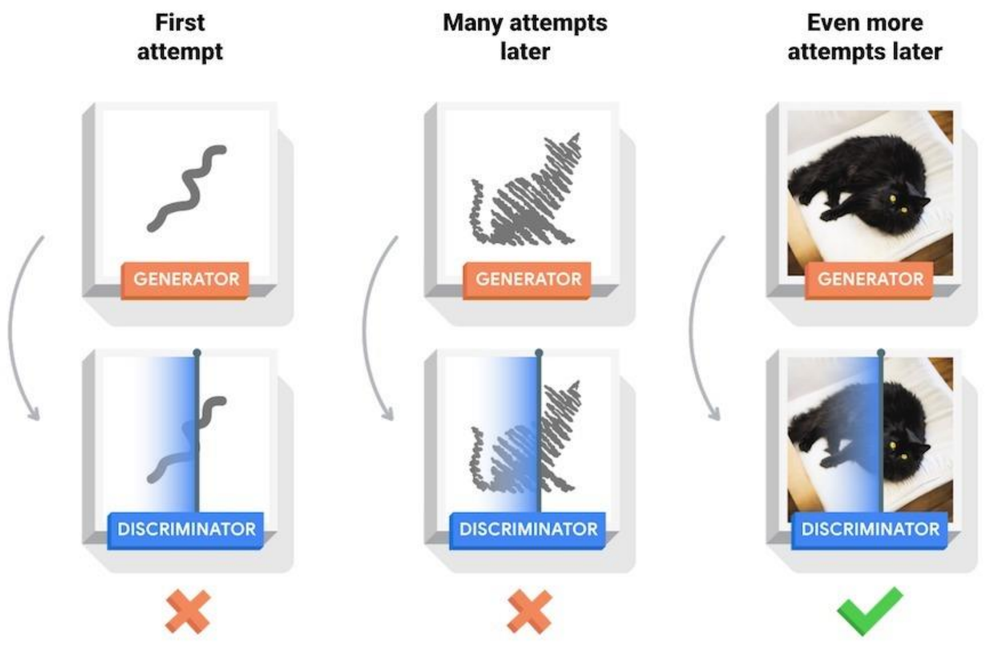


Figure 27 - Figure showing DCGANs process to create new data based on

3. Using transfer learning: Transfer learning, in a nutshell, is done by searching in the data science and artificial intelligence communities on the web for a model trained on data similar to our data which might be images of other types of cancers. This model is trained on a huge amount of data, we use the weights obtained from training on this huge dataset, and then fine-tune those weights using our neural networks on our small data set to increase the validation accuracy.

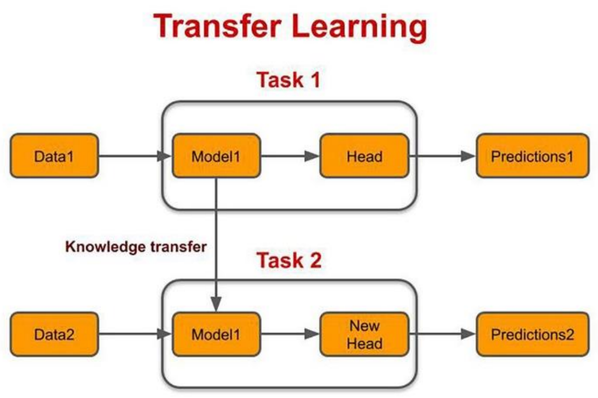
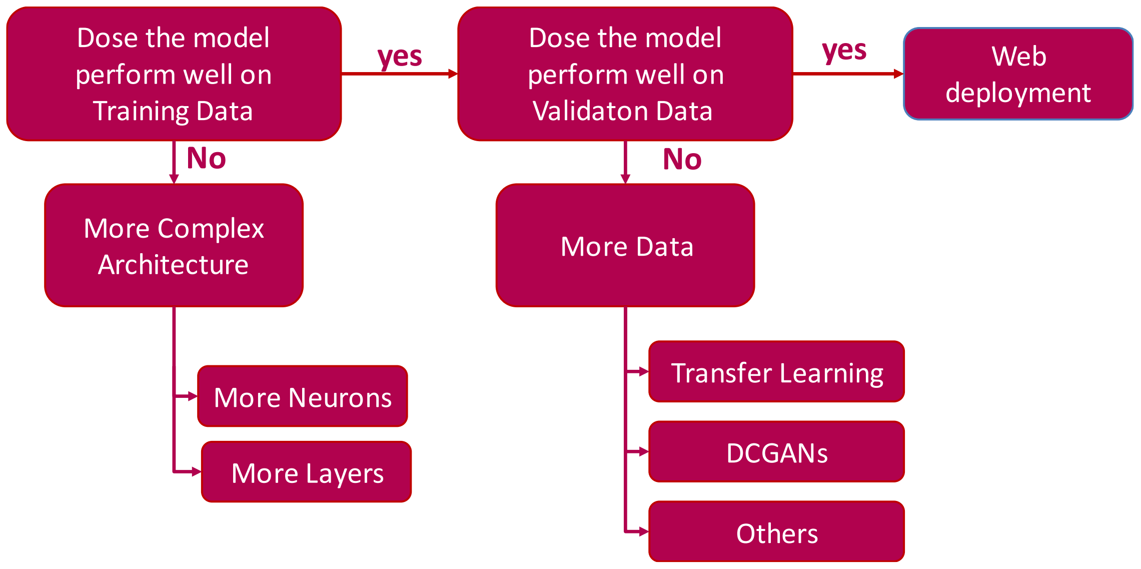


Figure 28 - Diagram showing transfer learning process

4. Model deployment: After conducting the research and building the model while reaching the highest accuracy we can reach, we will need to use the model as a final product, we can do this by deploying it to the web, by building a web application that is based on our model, where patients can upload their breast medical images and it will use our model to detect whether the patient has breast cancer or not, this service will be applicable using ultrasound images and x-ray images, further updates might include MRI images as well.


Dose the model perform well on Training Data
yes
Dose the model perform well on Validaton Data
yes
Web deployment
No
More Complex Architecture
More Neurons
More Layers
No
More Data
Transfer Learning
DCGANs
Others
Figure 29 - Model development life cycle

8. References

[1] Breast Cancer. 2021. [(accessed on Nov 2022)]. Available online: https://www.who.int/news-room/fact-sheets/detail/breast-cancer

[2] Sung H., Ferlay J., Siegel R.L., Laversanne M., Soerjomataram I., Jemal A., Bray F. Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA Cancer J. Clin 2022*

[3] International Cancer Control Partnership: WHO Cancer Country Profiles 2020. March 9, 2020. Available at https://www.iccp-portal.org/news/who-cancer-country-profiles-2020. Accessed NOV , 2022.

[4] Schlichting JA, Soliman AS, Schairer C, et al: Breast cancer by age at diagnosis in the Gharbiah, Egypt, population-based registry compared to the United States Surveillance, Epidemiology, and End Results Program, 2004–2008. BioMed Res Int 2022.

[5] Elghazaly H, Aref AT, Anderson BO, et al: The first BGICC consensus and recommendations for breast cancer awareness, early detection and risk reduction in low- and middle-income countries and the MENA region. Int J Cancer. Nov, 2022

[6] Science Direct. (2022, Nov. 11). Biomedical Signal Processing and Control [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1746809421008764?via%3Dihub

[7] MPI**.** (13 February 2022). Application of Deep Learning to Construct Breast Cancer Diagnosis Model [Online]. Available: https://www.mdpi.com/2076-3417/12/4/1957

[8] IGI Global**.** (2022). Optimized Breast Cancer Premature Detection Method With Computational Segmentation: A Systematic Review Mapping [Online]. Available: https://www.igi-global.com/chapter/optimized-breast-cancer-premature-detection-method-with-computational-segmentation/298105

[9] IEEEexplore**.** (30 October 2015). A Dataset for Breast Cancer Histopathological Image Classification [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7312934

[10] Link Springer**.** (28 February 2022). Quantum transfer learning for breast cancer detection [Online]. Available: https://link.springer.com/article/10.1007/s42484-022-00062-4

[11] MPI**.** (4 January 2022) Automated Breast Cancer Detection Models Based on Transfer Learning [Online]. Available: https://www.mdpi.com/1424-8220/22/3/876

[12] Science Direct**.** (2020). Chapter 3 - Applications of machine learning to brain disorders [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128157398000031

[13] Science Direct**.** (2020). Breast cancer detection by leveraging Machine Learning [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959520300801

[14] U. Kose, D. Gupta, D. A. V. H. C., and A. Khanna, “Chapter 7 - Deep convolutional neural network–based image classification for COVID-19 diagnosis,” in *Data Science for covid-19*, Amsterdam: Academic Press, 2021.

[15] “Volume 85, Issue 5, Part 2 - Kernels, in a nutshell,” in *Journal of logical and algebraic methods in programming*, Elsevier, 2016, pp. 921–930.

[16] “7.3. padding and stride” *7.3. Padding and Stride - Dive into Deep Learning 1.0.0-alpha1.post0 documentation*. [Online]. Available: https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html. [Accessed: 06-Dec-2022].

[17] S. K. Zhou, H. Greenspan, and D. Shen, “Chapter 14 - Deep Learning Models for Classifying Mammogram Exams Containing Unregistered Multi-View Images and Segmentation Maps of Lesions,” in *Deep learning for medical image analysis*, London, United Kingdom: Academic Press, 2017.

[18] I. V. A. N. DJORDJEVIC, “Chapter 14 - Quantum Machine Learning,” in *Quantum Information Processing, quantum computing,and quantum error correction: An engineering... approach*, S.l.: ELSEVIER ACADEMIC PRESS, 2021.

[19] “Forward propagation,” *What is Forward Propagation? | H2O.ai*. [Online]. Available: https://h2o.ai/wiki/forward-propagation/. [Accessed: 06-Dec-2022].

[20] V. Luhaniwal, “Forward propagation in Neural Networks-simplified math and code version,” *Medium*, 24-Oct-2020. [Online]. Available: https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250. [Accessed: 06-Dec-2022].

[21] A. Roy, “An introduction to gradient descent and backpropagation,” *Medium*, 14-Jun-2020. [Online]. Available: https://towardsdatascience.com/an-introduction-to-gradient-descent-and-backpropagation-81648bdb19b2. [Accessed: 06-Dec-2022].

[22] Al-Dhabyani W, Gomaa M, Khaled H, Fahmy A. Dataset of breast ultrasound images. Data in Brief. 2020 Feb;28:104863. DOI: 10.1016/j.dib.2019.104863.

[23] E. Lashgari, D. Liang, and U. Maoz, “Data augmentation for deep-learning-based electroencephalography,” *Journal of Neuroscience Methods*, vol. 346, p. 108885, 2020.

[24] K. Team, “Keras Documentation: Model training apis,” *Keras*. [Online]. Available: https://keras.io/api/models/model_training_apis/. [Accessed: 07-Dec-2022].

[25] Awsaf, “CBIS-DDSM: Breast Cancer Image Dataset,” *Kaggle*, 24-Jan-2021. [Online]. Available: https://www.kaggle.com/datasets/awsaf49/cbis-ddsm-breast-cancer-image-dataset. [Accessed: 07-Dec-2022].

Appendix

Link to the deep learning model repository on Github:
https://github.com/MohamedAlaaAli/BIOGENESIS