

**REPUBLIC OF TURKEY
ISTANBUL GELISIM UNIVERSITY
INSTITUTE OF GRADUATE STUDIES**

Department of Electrical-Electronic Engineering

**AN EFFICIENT FPGA IMPLEMENTATION OF CNN
SPECIALIZED IN IMAGE RECOGNITION
FOR BREAST CANCER**

Master Thesis

Omar Mhmood ABDULHADI

Supervisor

Assoc. Prof. Dr. Indrit MYDERRIZI

Istanbul – 2022

THESIS INTRODUCTION FORM

Name and Surname : Omar Mhmood Abdulhadi

Language of the Thesis : English

Name of the Thesis : An Efficient FPGA Implementation of CNN
Specialized in Image Recognition for
Breast Cancer

Institute : Istanbul Gelişim University Graduate Education
Institute

Department : Electrical-Electronic Engineering

Thesis Type : Post Graduate

Date of the Thesis : 27.01.2022

Page Number : 138

Thesis Supervisors : Assoc. Prof. Dr. INDRIT MYDERRIZI

Index Terms : Convolutional neural network (CNN), Field
Programmable Gate Arrays (FPGA), Image
recognition of breast cancer

Turkish Abstract : Görüntü işlemenin popüleritesi ve sağladığı avantajlar nedeniyle, medikal sektörlerde özellikle hastalıkların teşhisinde görüntü işleme teknolojisinin kullanıldığı yeni alanlar oluşmaya başlamıştır. tıbbi uygulamaların sınıflandırma sorunu, yalnızca gri kanala ilişkin bilgilerin bulunduğu ve hiçbir kronik bilginin bulunmadığı X-ray, CT taraması vb. gri tonlamalı görüntülerde kendini göstermektedir. Bu çalışmada, meme kanseri tespiti için bir evrimsel sinir ağı (CNN) kullanılmıştır; bu nedenle, CNN modelinin eğitimi için büyük, renkli biyopsi görüntüleri kullanılır. Rastgele orman, k-en yakın komşular, saf Bayes, destek vektör makinesi vb. gibi diğer modeller de kullanıldı. CNN sonunda kanser tahmininin doğruluğunu %97'ye kadar koruyabilir. Ayrıca VGG-16, AlexNet, ResNet-18, ShuffleNet ve

LeNet gibi önceden eğitilmiş derin öğrenme paradigmaları da aynı amaç için kullanıldı. Önerilen CNN ağı, birçok derin öğrenme ağından daha iyi performans gösterdi.

Distribution List

- : 1. To the Institute of Graduate Studies of Istanbul
Gelisim University
2. To the National Thesis Center of YÖK (Higher
Education Council)

Omar Mhmood ABDULHADI

**REPUBLIC OF TURKEY
ISTANBUL GELISIM UNIVERSITY
INSTITUTE OF GRADUATE STUDIES**

Department of Electrical-Electronic Engineering

**AN EFFICIENT FPGA IMPLEMENTATION OF CNN
SPECIALIZED IN IMAGE RECOGNITION
FOR BREAST CANCER**

Master Thesis

Omar Mhmood ABDULHADI

Supervisor

Assoc. Prof. Dr. Indrit MYDERRIZI

Istanbul – 2022

DECLARATION

I hereby declare that in the preparation of this thesis, scientific ethical rules have been followed, the works of other persons have been referenced in accordance with the scientific norms if used, there is no falsification in the used data, any part of the thesis has not been submitted to this university or any other university as another thesis.

Omar Mhmood ABDULHADI

.../.../2022



TO ISTANBUL GELISIM UNIVERSITY
THE DIRECTORATE OF INSTITUTE OF GRADUATE STUDIES

The thesis study of Omar Mhmood ABDULHADI titled as AN EFFICIENT FPGA IMPLEMENTATION OF CNN SPECIALIZED IN IMAGE RECOGNITION FOR BREAST CANCER has been accepted as MASTER THESIS in the department of ELECTRICAL-ELECTRONIC ENGINEERING by out jury.

Director *Assoc. Prof. Dr. Indrit MYDERRIZI*
(Supervisor)

Member *Asst. Prof. Dr. Ahmed Amin Ahmed SOLYMAN*

Member *Asst. Prof. Dr. AFM Shahen SHAH*

APPROVAL

I approve that the signatures above signatures belong to the aforementioned faculty members.

... / ... / 2022

Prof. Dr. İzzet GÜMÜŞ
Director of the Institute

SUMMARY

Due to the popularity of image processing and its advantages, new fields have begun to be established using image processing technology in the medical sector, especially for diagnosing diseases. The information used for the diagnosis is preserved within the image pixels, and the quality of the diagnosis is restricted by the amount or quality of information preserved in every pixel. The problem arises when there is not enough pixel information in some/specific images, making it unfit for processing under all mentioned applications. In other words, the value of information preserved by any image is a function of what each pixel preserved. This problem is manifested in grayscale images (that are widely populated in medical applications) where only information about grey channels is available, and no chromatic information exists. Because of that, grayscale images are unfit for image processing since the pixel information of the image is represented by one value only (grayscale channel), and no strong correlation can be ensured between any two images in grayscale (assumed for the same event/object). Convolution neural networks have achieved a wide range of classification challenges, including facial, object and activity recognition. However, short development time comes at the cost of performance and energy economy. Although PYNQ development boards have recently reached a level of maturity, where FPGA prototyping effort is equivalent to that of CPUs or GPUs, FPGA prototyping is still a solid alternative for developing embedded CNN applications. The research published here describes a design to speed up FPGA prototyping using an open-source framework; it offers a simplified platform for developing CNN-powered FPGA applications quickly. In this thesis, the proposed framework incorporates high-level convolutional networks, which are programmable for a wide range of network specifications and offer superior performance in a time-efficient manner. In this work, a convolutional neural network (CNN) is utilised for breast cancer detection; hence, big, coloured biopsy images are used for the training of the CNN model. Other models, such as random forest, k-nearest neighbors, naïve Bayes, support vector machine, etc., were also used. CNN could eventually preserve the accuracy of cancer prediction to 97%. Moreover, pre-trained deep learning paradigms were used for the same purpose, such as VGG-16, AlexNet, ResNet-18, ShuffleNet and LeNet. The proposed CNN network has outperformed many deep learning networks.

Keywords : Convolutional neural network (CNN), field-programmable gate array (FPGA), Python productivity for ZYNQ (PYNQ), Central processing units (CPUs), Graphics processing units (GPUs)



ÖZET

Görüntü işlemenin popüleritesi ve sağladığı avantajlar nedeniyle tıp sektöründe özellikle hastalıkların teşhisinde görüntü işleme teknolojisi kullanılarak yeni alanlar kurulmaya başlanmıştır. Teşhis için kullanılan bilgiler görüntü pikselleri içinde korunur ve teşhisin kalitesi, her pikselde korunan bilgi miktarı veya kalitesi ile sınırlandırılır. Sorun, bazı/belirli görüntülerde yeterli piksel bilgisi olmadığında ortaya çıkar, bu da onu belirtilen tüm uygulamalar altında işlemeye uygun hale getirmez. Başka bir deyişle, herhangi bir görüntü tarafından korunan bilginin değeri, her pikselin koruduğu şeyin bir fonksiyonudur. Bu sorun, yalnızca gri kanallarla ilgili bilgilerin mevcut olduğu ve hiçbir kronik bilginin bulunmadığı gri tonlamalı görüntülerde (tıbbi uygulamalarda yaygın olarak kullanılan) kendini gösterir. Bu nedenle, görüntünün piksel bilgisi yalnızca bir değerle temsil edildiğinden (gri tonlamalı kanal) gri tonlamalı görüntüler görüntü işleme için uygun değildir ve gri tonlamalı herhangi iki görüntü arasında (aynı olay/nesne için varsayılır) güçlü bir korelasyon sağlanamaz.). Evrişim sinir ağları, yüz, nesne ve aktivite tanıma dahil olmak üzere çok çeşitli sınıflandırma zorluklarına ulaştı. Ancak, kısa geliştirme süresi performans ve enerji ekonomisi pahasına gelir. PYNQ geliştirme kartları, son zamanlarda FPGA prototiplerinin CPU veya GPU'larına eşdeğer olduğu bir olgunluk düzeyine ulaşmış olsa da, FPGA prototipleme, gömülü CNN uygulamaları geliştirmek için hala sağlam bir alternatiftir. Burada yayınlanan araştırma, açık kaynaklı bir çerçeve kullanarak FPGA prototiplemeyi hızlandıracak bir tasarımı açıklamaktadır; CNN destekli FPGA uygulamalarını hızlı bir şekilde geliştirmek için basitleştirilmiş bir platform sunar. Bu tezde önerilen çerçeve, çok çeşitli ağ özellikleri için programlanabilen ve zaman açısından verimli bir şekilde üstün performans sunan yüksek seviyeli evrişimli ağları içermektedir. Bu çalışmada, meme kanseri tespiti için bir evrişimsel sinir ağı (CNN) kullanılmıştır; bu nedenle, CNN modelinin eğitimi için büyük, renkli biyopsi görüntüleri kullanılır. Rastgele orman, k-en yakın komşular, saf Bayes, destek vektör makinesi vb. gibi diğer modeller de kullanıldı. CNN sonunda kanser tahmininin doğruluğunu %97'ye kadar koruyabilir. Ayrıca VGG-16, AlexNet, ResNet-18, ShuffleNet ve LeNet gibi önceden eğitilmiş derin öğrenme paradigmaları da aynı amaç için kullanıldı. Önerilen CNN ağı, birçok derin öğrenme ağından daha iyi performans gösterdi.

Anahtar Kelimeler : Evriřimli sinir ađı (CNN), sahada programlanabilir kapı dizisi (FPGA), ZYNQ için Python üretkenliđi (PYNQ), Merkezi işlemler birimleri (CPU'lar), Grafik işleme birimleri (GPU'lar)



TABLE OF CONTENTS

SUMMARY	i
ÖZET	iii
TABLE OF CONTENTS	v
ABBREVIATIONS	ix
LIST OF TABLES	x
LIST OF FIGURES	xii
LIST OF ANNEXES	xv
PREFACE	ix
INTRODUCTION	1

CHAPTER ONE

PURPOSE OF THIS THESIS

1.1. Literature Survey	3
1.2. Problem Statement	10
1.3. Study Objectives.....	11
1.4. Thesis Organization.....	12

CHAPTER TWO

THEORETICAL AND BACKGROUND

2.1. Anatomy of the Breast.....	13
2.1.1. Anatomy and Physiology of the Breast.....	13
2.1.2. Structure of the Breast.....	13
2.1.3. Breast Conditions	14
2.2. Introduction to Machine Learning and Neural Networks	16
2.2.1. Neural Networks.....	16
2.2.2. A brief Introduction to Machine Learning	17
2.2.3. Machine Learning.....	18
2.2.4. Deep Learning	19
2.2.5. Convolutional Neural Networks.....	20

2.2.5.1.	Convolutional Layer	23
2.2.5.2.	Non-linearity (Activation Function).....	24
2.2.5.3.	Stride.....	25
2.2.5.4.	Padding.....	26
2.2.5.5.	Pooling layer.....	26
2.2.5.6.	Fully-Connected layer	27
2.2.5.7.	Dropout layer	28
2.2.5.8.	Classification layer	29
2.2.5.9.	Grouping layers	29
2.3.	Popular CNN Architectures.....	30
2.3.1.	LeNet	30
2.3.2.	AlexNet	31
2.3.3.	VGGNet.....	32
2.3.4.	Residual Network (ResNet).....	33
2.4.	Introduction to Field Programmable Gate Arrays.....	35
2.4.1.	Field-Programmable Gate Arrays.....	35
2.4.1.1.	Programmable logic.....	36
2.4.1.2.	Programmable interconnect.....	37
2.4.1.3.	Programmable I/O	37
2.4.1.4.	Customizable Functional Blocks.....	37
2.4.2.	FPGAs Versus Other Hardware Platforms.....	38
2.4.3.	FPGAs Versus ASICs	38

CHAPTER THREE

METHODOLOGY

3.1.	Overview	40
3.2.	Database	40
3.3.	Preprocessing.....	40
3.4.	Image Decomposition.....	42
3.5.	Neural Network Modelling	45
3.6.	Model Establishment.....	47
3.7.	Performance Metrics	48

3.7.1.	Mean Square Error	49
3.7.2.	Mean Absolute Error	49
3.7.3.	Root Mean Square Error	49
3.7.4.	Accuracy	49
3.7.5.	Confusion Matrix	50

CHAPTER FOUR

HARDWARE MODEL

4.1.	Overview	51
4.2.	PYNQ Development Board.....	52
4.3	Image Flashing.	55
4.4.	Image Configuration	55
4.5.	Prediction Models on FPGA	56
4.6.	Deep Learning on FPGA	56
4.6.1.	Model 1.....	57
4.6.2.	Model 2.....	59
4.6.3.	Implementation LeNet-5 - ARM Linux OS	60
4.6.4.	Implementation AlexNet – ARM Linux OS	63
4.6.5.	Implementation Prototypes: Test Phase	66

CHAPTER FIVE

RESULTS AND DISCUSSION

5.1.	Methods Overview	67
5.2.	Results and Discussion of Proposed CNN	70
5.3.	Pre-trained Models	76
5.4.	Confusion Matrix	82
5.5.	Machine Learning Models.....	82
5.6.	Regression Model.....	86
5.7.	ROC and AUC Measures	91
5.7.1	Proposed Model (CNN).....	92
5.7.2.	Machine Learning Models	92
5.8.	Comparisons Between Implementation (CPU and PYNQ)	93

5.9. Comparisons Between the Implementation of AlexNet and LeNet (CPU and PYNQ).....	96
--	----

CHAPTER SIX

CONCLUSION AND DISCUSSION

6.1. Conclusion.....	98
6.2. Discussion	99
6.3. Proposed CNN Over FPGA Development Board	100
REFERENCES.....	102
ANNEXES.....	109
RESUME.....	117

ABBREVIATIONS

CAD	:	Computer Aided Design
KNN	:	K-Nearest Neighbor
FPGA	:	Field Programmable Gate Array
CNN	:	Convolutional Neural Network
CPU	:	Central Processing Unit
SVM	:	Support Vector Machine
ANN	:	Artificial Neural Network
GPU	:	Graphics Processing Unit
PYNQ	:	Python Productivity for Zynq
BC	:	Breast Cancer
RF	:	Random Forests
ReLU	:	Rectified Linear Unit
API	:	application programming interface
DL	:	Deep Learning
ML	:	Machine Learning
MAE	:	Mean Absolute Error
MSE	:	Mean Square Error
RMSE	:	Root Mean Square Error

LIST OF TABLES

Table 1. Confusion Matrix Candidates' Formula.....	56
Table 2. PYNQ-Z2 Development Board Features.	56
Table 3. Required Libraries to be Installed on the PYNQ Development Board.	56
Table 4. The First Proposed Model of FPGA-based Breast Cancer Detection.	56
Table 5. Epoch-wise Results (MSE and Time Computation) for the First Proposed Model..	56
Table 6. The Second Proposed Model of FPGA Environments (Model 2)... ..	56
Table 7. Epoch-wise Results (MSE and Time Computation) for the Second Proposed CNN Model	56
Table 8. Lasagne LeNet Configuration.	56
Table 9. Epoch-wise Results (MSE and Time Computation) for the LeNet-5 Model... ..	56
Table 10. Lasagne AlexNet Configuration.....	56
Table 11. Epoch-wise Results (MSE and Time Computation) for the AlexNet Model.....	56
Table 12. The Configuration of the Proposed CNN Structure used in CPU-based Learning..	56
Table 13. CNN Model Training Coefficients.....	56
Table 14. Results of the First and Second Iterations of the CNN Model.....	63
Table 15. Results of the Third and Fourth Iterations of the CNN Model.	63
Table 16. Results of the Fifth and Sixth Iterations of the CNN Model.....	63
Table 17. Results of the Seventh and Eighth Iterations of the CNN Model.	63
Table 18. VGG-16 Neural Network Structure.	63
Table 19. AlexNet Neural Network Structure.	59
Table 20. LeNet Neural Network Structure	60
Table 21. Proposed Network Structure.....	63
Table 22. Pre-trained Deep Learning Classification Performance Measures.....	65
Table 23. Accuracy of Prediction Measures for all the Algorithms.....	68
Table 24. Regression Model Performance Metrics for the Machine Learning Algorithms as Compared with the Proposed CNN.	69
Table 25. AUC Values for the Machine Learning Algorithms	68

Table 26. Performance Comparison of the Proposed Models (CPU and FPGA) 68

Table 27. Performance Comparison of the AlexNet and LeNet Models (CPU and FPGA) 68



LIST OF FIGURES

Figure 1. Classification Categories of Breast Cancer	4
Figure 2. Comparison performance evaluation between VGG-16, ResNet-50 and Q....	5
Figure 3. ROC curve for the 5-folds of cross-validation and the mean ROC curve	7
Figure 4. The surrounding structure and the breast.....	13
Figure 5. Breast structure	14
Figure 6. Histologically distinct breast cancers	16
Figure 7. Artificial intelligence (AI), machine learning (ML), neural networks (NN), deep learning (DL) and spiking neural networks (SNN)	19
Figure 8. Features of extraction and classification of the visual image using convolution neural networks	23
Figure 9. An illustrative example of arithmetic operations performed in the convolutional layer of the CNN network	23
Figure 10. Activation functions: ReLU, tanh and sigmoid	25
Figure 11. Example of stride with convolution.....	26
Figure 12. Pooling output averages and maximums for a 2 x 2 filter with a stride of 2	27
Figure 13. A three-layer fully connected matrix	28
Figure 14. Representation of dropouts	28
Figure 15. CNN architecture as an example	30
Figure 16. Example of image classification using CNN architecture	31
Figure 17. LeNet mode's architecture	32
Figure 18. AlexNet mode's architecture.....	33
Figure 19. A structure of VGG network building blocks: convolution (Conv) and fully connected (FC)	34
Figure 20. Basis of the residual network architecture.....	34
Figure 21. Residual learning: block of the residual network architecture	35
Figure 22. FPGA internal architecture	36
Figure 23. Coloured and RGB image channels.....	42
Figure 24. Neural network fundamental structure	44
Figure 25. A demonstration of CNN training processes	47
Figure 26. CNN training model procedure.....	48

Figure 27. PYNQ-Z2 FPGA development board structure.....	56
Figure 28. Software to flash the image OS into the PYNQ development board.....	57
Figure 29. Sample application for breast cancer image	68
Figure 30. Preprocessing model flow diagram.....	70
Figure 31. Confusion matrix of proposed CNN.....	78
Figure 32. Pre-trained deep learning classifier results (graphical representation).....	84
Figure 33. Confusion matrix of VGG-16.....	85
Figure 34. Confusion matrix of AlexNet	85
Figure 35. Confusion matrix of LeNet-5.....	86
Figure 36. Confusion matrix of ResNet-18.....	87
Figure 37. Confusion matrix of ShuffleNet	88
Figure 38. Graphical representation of the results of machine learning approaches showing prediction measures for all the algorithms	89
Figure 39. Confusion matrix of Logistic Regression	90
Figure 40. Confusion matrix of Kernel Naïve Bayes.....	91
Figure 41. Confusion matrix of Linear Discriminant.....	91
Figure 42. Confusion matrix of KNN	92
Figure 43. Confusion matrix of Gaussian Naïve Bayes	92
Figure 44. Confusion matrix of Fine Tree.....	93
Figure 45. Confusion matrix of linear SVM	94
Figure 46. Demonstration of ROC and AUC region for proposed CNN.....	96
Figure 47. ROC and AUC demonstration for the Kernel Naïve Bayes algorithm.....	97
Figure 48. ROC and AUC demonstration for the Logistoc Regression algorithm	97
Figure 49. ROC and AUC demonstration for Fine Tree algorithm	98
Figure 50. ROC and AUC demonstration for the Gaussian Naïve Bayes algorithm....	98
Figure 51. ROC and AUC demonstration for KNN alorithm	98
Figure 52. ROC and AUC demonstration for the Linear Discriminant algorithm.....	98
Figure 53. ROC and AUC demonstration for the Linear SVM algorithm.....	99
Figure 54. Graphical comparison of AUC values between the proposed CNN and machine learning algorithms	100
Figure 55. Graphical representation of the training time comparison of the proposed models (CPU and FPGA).....	101
Figure 56. Graphical representation of the accuracy comparison of the proposed models (CPU and FPGA).....	101

LIST OF ANNEXES

ANNEXES A. Three main requirements are to be fulfilled upon reaching to this step	109
ANNEXES B. Deep Learning Libraries.....	111
ANNEXES C. Caffe Project.....	112
ANNEXES D. Uploading database file.....	115



PREFACE

During the process of preparing this thesis , I would like to express my gratitude and thanks to Assoc. Prof. Dr. Indrit MYDERRIZI, who supported and guided me throughout the period .

I do not forget my family, my wife who supported me until this moment .

Thanks to all of my friends who have always encouraged me.



INTRODUCTION

Cancer is a major problem for public health worldwide, causing many deaths and straining public and private health systems. For women, breast cancer is one of the most common types of cancer, and it is also one of the most fatal unless detected and treated early. More than 2.4 million new cases have been diagnosed, and 523,000 people are missing (Fitzmaurice et al., 2017). Breast carcinoma histopathology may generally be analysed by examining the different types of tumours present, including 20 major tumours and 18 minor subtypes (Fletcher et al., 2002). A biopsy is the only way to know for sure if cancer is present. The final diagnosis of breast cancer, including grading and staging, is still performed by pathologists who examine minute pictures under a microscope for visual findings. The histopathological analysis is a demanding and lengthy procedure requiring enormous skill, attention and patience. Computer-aided diagnosis (CAD) technologies, in turn, can enable pathologists to be more productive, objective and consistent with their diagnoses (Motlagh et al., 2018).

On the other hand, numerous advances in neural network technologies, including transfer learning, ensemble learning and many types of regularisation, have significantly enhanced neural networks' performance and have yielded tremendous success in numerous disciplines. Pathologists currently utilise the major method to assess cancers' stage, kind and subtype through visual evaluation of histopathology slides. Many years ago, as the medical databases increased, a larger number of histopathological slides were added. Advanced machine learning approaches were incorporated. This naturally entails introducing deep learning techniques to the medical problem picture categorisation (Gurcan et al., 2009). Deep neural networks have shown their ability to revolutionise existing technologies by delivering exceptional performance in both visual and audio processing. Over the last few years, research on a deep neural network (CNN) has seen a dramatic increase. While a neural network (NN) is like an average brain, CNN is an interconnected group of neurons that are subject to learnable weight and bias adjustments. CNN focuses on image processing. Thus, it may take advantage of certain aspects of images to boost its speed, allowing it to handle big picture databases more efficiently (Rahman et al., 2016). On embedded systems, the use of CNN allows real-time classification processes to be

done more dynamically. By off-line training, models can be established and used for application deployment, allowing the system to concern itself with improving the efficiency of forwarding propagation (i.e. deployment). Embedded CNN systems can have a huge impact on numerous fields of study. Field-programmable gate arrays (FPGAs) have the potential to make CNN acceleration more practical. FPGA is an integrated circuit that allows them to be reconfigured at the gate level. A great deal of bespoke logic (or look-up tables) is contained in it, which can be programmed into multiple and varied digital modules for countless and various purposes (Gschwend et al., 2020).

There are several ways to use FPGA to take advantage of the many benefits of embedded CNN applications.

- FPGA's powerful parallel processing capacity enables the incorporation of CNNs into embedded systems to significantly increase the rate of deployment.
- The reconfigurability of FPGA makes it possible to synthesise hardware accelerators for a particular purpose. Each CNN model offers better resource management and more customisation of the application.
- FPGA provides increased data processing capacity while using less electricity than other systems, where energy consumption is an essential factor in the use of devices.

The implementation side of the thesis aims to implement the CNN deployment design flow on FPGA in a manner comparable to how it is implemented on CPU or GPU platforms to enable FPGA-based CNN deployment with insights familiar to engineers. In order to run Theano, which is a popular AI framework, the framework utilises the Ubuntu OS on PYNQ's ARM core. Instead of packaging the FPGA application programming interface (API) as an ordinary Theano CNN layer function, it is packaged as a built-in Theano function, which could be instantiated in the same way as a Theano built-in function. In this work, breast cancer is being studied using deep learning and machine learning paradigms.

CHAPTER ONE

PURPOSE OF THE THESIS

1.1. Literature Survey

Outline

Substantial attempts have been made for the detection of histological and X-ray images of breast cancer (BC), where the two primary types of BC (benign and malignant) are more generally detected by CAD and FPGAs. The current project is about digital photographs of breast biopsy specimens, particularly open (surgical) biopsies. Therefore, this state-of-the-art study will focus on image databases scanned from breast biopsy samples. Several literary studies that focus on the histopathology field, where there has been a considerable surge in research, use the researchers' own datasets, various evaluation techniques and different performance measures (Irshad et al., 2013).

Classification Categories of Breast Cancer

Two basic types of automatic BC classification-related works can be created based on the feature representation. In the first type, efforts to automate the feature extraction technique are included. Mammograms, magnetic resonance imaging, biopsy thermography and digital imaging of biopsy slides, for example, could all be used to image breast pathology (Yan et al., 2020). The features of the second type, the Wisconsin Breast Cancer Dataset, have already been extracted. Figure 1 shows the types of classification categories of BC. In this chapter, the most recent and important procedures in the literature are presented, where technology is used to deal with different data types of BC images. The results of the literature review are listed below.

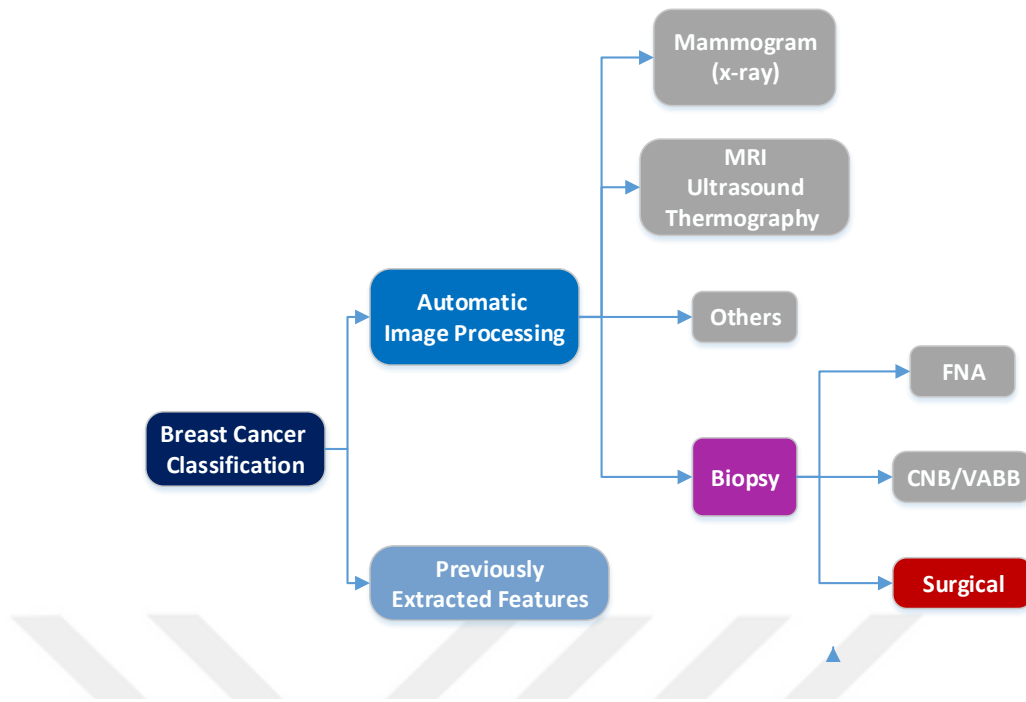


Figure 1. Classification Categories of Breast Cancer

Image Processing

Aly et al. (2021) proposed a CAD system based on YOLO deep learning approaches to discriminate between cancer and benign tumours in full-field digital imaging and images of mammograms from the INbreast dataset. By employing the suggested approach, the overall accuracy was 94.2%. The YOLO v3 algorithm, ResNet and Inception were used to detect masses. Feature extraction methods and k-means clustering for the creation of anchors that correspond to the original dataset.

Tsochatzidis et al. (2021) proposed a novel way for integrating segmentation information into a CNN. The strategy was used to improve the diagnosis performance of a CNN classifier by using both ground-truth and artificially generated segmentation maps. Because segmentation quality has been found to have an important impact on diagnosis performance, the current study should focus on enhancing automatic segmentation in the future. Two datasets were used: DDSM-400. There are 400 mammograms in this dataset (196 benign and 204 malignant), with masses chosen from the DDSM.

Rahman et al. (2020) presented a BC CT image diagnosis using pre-trained CNN models and classifying mammographic mass tumour images as benign or malignant. Transfer learning, special preprocessing and data increases were introduced to address

the restricted availability of training datasets. The proposed architectures were evaluated in the digital database for screening mammography (DDSM) and achieved a precision of 0.796, 0.754 accuracies and a reminder of 0.891 on a CNN-like InceptionV3 model. However, a 0.857 precision and a retrieval rate of 0.873 with a ResNet-50-like CNN network were achieved. Overall, proposed a model similar to the ResNet-50 network achieved ~5% better than the InceptionV3 network.

Omonigho et al. (2020) improved AlexNet deep CNN to classify BC into benign and malignant tumours, including the BreakHis dataset. The work included different convolutional 5 x 5 and 3 x 3 filter sizes in each of the first four layers. A preprocessing algorithm was used to eliminate noise factors and reduce calculation time. The results show that the completely connected layer used as a classifier in the updated architecture of AlexNet provided an overall rating accuracy of 95.70%, which was improved over the conventional techniques used for mammogram analysis.

Ismail et al. (2019) compared the detection of BC with two model networks for deep learning. The overall process included preprocessing images, classification and evaluation of performance. Using MIAs mammography datasets, researchers evaluated the efficacy of deep learning models VGG-16 and ResNet50 in distinguishing between benign and malignant cancers. The results show that VGG-16 produced 94% better results in accuracy than ResNet-50 with 91.7%.

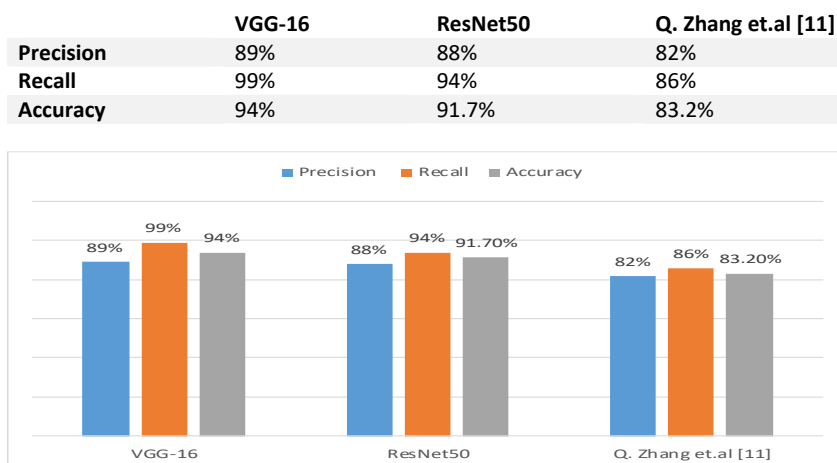


Figure 2. Comparison performance evaluation between VGG-16, ResNet-50 and Q (Ismail, N. S., & Sovuthy, C., 2019)

Zhang et al. (2019) presented the detailed learning approach linked with object identification to identify and classify lesions in order to detect BC. Transfer learning was also applied based on the faster R-CNN network. Also, the five network extractors ResNet101, InceptionV2, InceptionV3, MobileNet and Inception ResNetV2, were analysed to examine the model's effect (DDSM) on the mammography screening dataset. The accuracy of the neural networks used in this study was demonstrated (MobileNet 71.1%, InceptionV2 73.9%, ResNet-101 76.4%, InceptionV3 78% and Inception ResNetV2 80%).

Motlagh et al. (2018) employed multiple types of ResNet networks on BC digital pictures, including the BreakHis dataset. Colour map selection and data augmentation were used as preprocessing techniques to prepare data for further extraction and categorisation. Deep CNNs were then used to identify and detect malignant and benign instances using pre-trained and fine-tuned deep CNNs. Approximately 85% of the shots were chosen at random, with the remaining images used for testing to create a training set. For benign and malignant subtype classifications, this was done at 94.8 and 96.4%, respectively. The binary classification job had a precision of 98.7% when it came to benign and malignant classes.

Salem et al. (2018) presented the study of DCNN multi-view for the classification of mammograms. Despite the small size of the BC image dataset, the ability of DCNN appeared to be superior to even a small dataset. Also, the recall metric was considered an important factor in network performance. The recall metric should be maximised to demonstrate the significance of using cross-validation to ensure that the learning process is not overfitting. The results show an average rating accuracy of 80.10% for 5-fold cross-validation and an average AUC of 0.78.

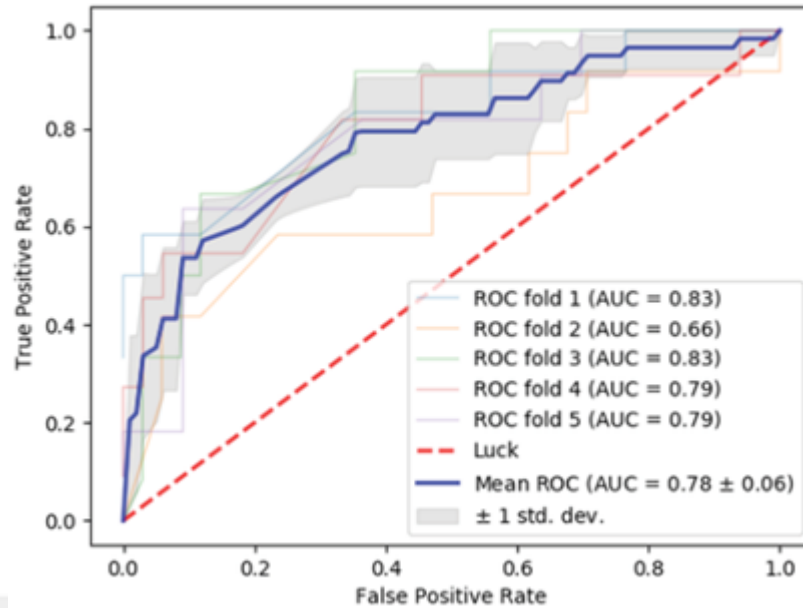


Figure 3. ROC curve for the 5-folds of cross-validation and the mean ROC curve (Salem, M. A. M., 2018, December)

Akbar et al. (2017) presented a new strategy for CNN regularisation. The key notion was the transition from convolutional to fully connected layers (FC). The modified transition model learned filters of different dimensions and then compressed them using average pooling. According to the authors, the total test accuracy of the BreakHis dataset was 82.7%.

Han et al. (2017) proposed an additional deep learning multi-classification model. The class-structured deep convolutional neural network (CSDCNN) was an end-to-end recognition approach that used a hierarchical representation and feature space constraints to maximise the Euclidean distance between inter-class labels. The original BreakHis dataset, which includes both oversampled imbalanced classes and the same database augmented on the training set, was employed to evaluate the technique's performance. Using the BreakHis dataset boosted the achieved performance for the picture- and patient-level analysis. Overall, the accuracy was 96.07 and 96.25% for image- and patient-level binary classification, respectively.

Wei et al. (2017) have developed an approach called BiCNN, which was inspired by the CNN network. Two datasets were used, the original and an augmented version, incorporating class and sub-class labels. A total of 50, 25 and 25% of the photos were

in the training, validation and test sets, respectively. The stated patient- and image-level identification rate was 97%.

Kahya et al. (2017) suggested an adaptive sparse support vector intended to accomplish the feature selection and classifications on histopathology pictures. The algorithm they created was based on the smooth support vector machine (SSVM) with L1-norm and Wilcoxon rank sum test. A 10-fold cross-validation was utilised in the proposed adaptive SSVM in the experimental context and resulted in enhanced classification accuracy across all magnification factors, with an average value of 94.97 (40×), 93.62 (100×), 94.54 (200×) and 94.42 (400×).

Spanhol et al. (2017) presented a study on the deep convolutional activation feature (DeCAF) use of the BreakHis dataset for BC identification, the results of the image-based evaluation and the results of the correction-based evaluation. The CNN-based method had an accuracy of approximately 77.8% when running a four-class experiment and 83.3% binary category test accuracy when validating the 2015 dataset for the BC classification challenge. Also, the CSDCNN standardised deep learning model was used to classify histopathological images of BC.

Araújo et al. (2017) proposed an additional deep learning CNN model for classifying haematoxylin biopsy images of the BC dataset. Images were placed in four categories: normal tissues, lesions, malignant neoplastic tumours or invasive carcinomas. The network's architecture was set up to focus on both nuclei and overall tissue organisation, depending on the scale of the details. This architecture allowed for whole-slide pathology images to be expanded upon. Some of the features extracted by CNNs were fed into the classifier to train it. Accuracies of 77.8% for four different types of class and 83% for carcinoma/non-carcinoma were reached. Within that small statistical error, the cancer detection system was about 96% sensitive.

Gupta et al. (2017) proposed two NNs of BC using DL methods, in which CNN variants were being classified. The data on the two proposed NNs were trained by the BreakHis dataset. An independent magnification classification of BC was suggested in 2016 based on a CNN with different sizes of convolution kernels (seven, five and three kernels). Multi-task CNN (MTCNN) classification used the BreakHis dataset for BC and recorded an 83.25% recognition accuracy.

Spanhall et al. (2016) presented how a CNN (a network of computers known as a cluster neural network) trained directly from histopathological images to produce the described results. The patch method used by CNNs takes advantage of extracting sub-images during training and testing phases. It was important to extend the training set through randomly defined extraction of patches from the data in training. After classifying each patch, their classification findings were brought together, and the overall findings of the patch analysis were produced. Baseline accuracy increased by between 4 and 6% due to the methodology used in this study.

Chan and Tuszynski (2016) presented an approach that sought to use fractal dimension as a descriptor and SVM as a classifier to examine BCs. In terms of their experimental approach, the authors implemented their methodology and tried to categorise the tumour subtypes they found in the BreakHis dataset. Despite the F1 score of 97.9%, the reported accuracy was only 55.6%.

Spanhol et al. (2015) compiled several combinations of six different visual feature descriptors, each tested with classifiers, and average accuracies were 80 to 85%. The BreakHis dataset was used for training 70% of the data, while the remaining 30% was used for testing. It was instead used to create the training set. For example, in order to employ the five divisions of a practice test as well as the relevant results, a procedure must be established that includes both the five divisions of the practice test and the results. The possibility to make a fair comparison of approaches was enabled because the samples for each trial are publicly available.

Filipczuk et al. (2013) employed a database including 737 pictures of cytological samples taken from Fine Needle Aspiration (FNA) biopsies done on 67 individuals at Polish Regional Hospital. This breast cytological material was transformed into virtual slides, and a pathologist then manually selected 8-bit RGB TIFF files of $1,538 \times 828$ pixels generated from each of the selected locations. This dataset contained 25 benign (with an aggregate total of 275 photos) and 42 malignant (an aggregate count of 462 images) samples. Four separate classifiers in 25 characteristics represented the nuclei. The proposed approach correctly classified 98.51% of samples as benign or malignant.

George et al. (2013) presented a method for accurately identifying and tracking nuclei in breast cytological images using an automated cell nuclei identification and segmentation method. Classifying images as malignant or benign breast tumours was

done using the extracted features from this method. This study's dataset was collected in partnership with the Early Cancer Detection Unit at the Ain Shams University Hospital in Egypt. Breast lumps, also known as fibroadenomas, were taken for the fine needle aspiration cytology (FNAC) procedure, where samples were stained with May-Grünwald-Giemsa³ or Diff⁴ stain. There were 92 photographs in this set of RGB JPEG microscopic photos with a resolution of 2,560 x 1,920 pixels, 45 of which depict benign lesions and 47 of which depict malignant tumours.

1.2. Problem Statement

According to the literature, medical applications-based image processing is an evolving topic that is still under the reliability improvement phase. Researches are competing to implement a reliable structure for computer vision for medical purposes.

As per the literature survey illustrated in this chapter, machine learning-based medical applications can predict the conditions of patients or, in other words, diagnose the case objects for infections in various diseases. The current application of machine learning-based medical applications is still under development and has not yet come into practical fields.

Some works seen in the literature are deploying the NNs with a large number of hidden layers for classification of the data, impacting the performance of the overall model by consuming a big amount of computational cost and, hence, increasing the processing time.

Today, images have become vital for various applications, such as pattern recognition, security systems, face recognition, remote sensing, etc. Medical applications of image processing are important fields aimed for human living enhancement.

Due to the popularity of image processing applications, new fields have begun to be established using the technology of image processing called medical image processing for diagnosis.

Considering that noise interference with the image could manipulate the information of every channel, and the noise impact is different in each channel, the need for multiple channels while processing the image digitally is a must and is the

only way for matching two images with each other in applications like face or pattern recognition. Noise impacting the signal channel, such as a grayscale image, may corrupt all information preserved in the pixel, while the noise impact is lesser while dealing with the multi-channel image such as coloured images. However, this poses another challenge because of the amount of information and the difference between the states of the image data, which affects the system's accuracy.

1.3. Study Objectives

A machine learning-based predictor is used to predict BC in a large image dataset. The model is aimed to provide a high accuracy prediction of the disease by satisfying the following points.

1. To conduct FPGA-based learning for BC diagnosis by training the model in FPGA environments to enhance the model's accuracy.
2. Predict cancer occurrence in picture data using sophisticated NNs such as CNN. This model is predicted to forecast with high accuracy. This type of technology was created recently to address the problems associated with learning and error when training classic NN models.
3. The prediction performance of the CNN is evaluated in different model hyperparameters to meet the optimum optimisation of the aforementioned outcomes to justify its performance.
4. To understand the prediction performance, other paradigms, such as deep learning and machine learning, i.e. machine learning tools such as k-nearest neighbor (KNN), random forest (RF), etc., were also used for performance comparison. Pre-trained deep learning paradigms were used for the same purpose, such as VGG-16, AlexNet, ResNet-18, ShuffleNet and LeNet.
5. The mean square error, mean absolute error, root mean square error, prediction time and prediction accuracy were used to evaluate the performance of the models employed in this study.
6. Implementation of smart and consistent data preprocessing models that tackles the drawbacks, such as data variance.

1.4.Thesis Organization

- There are six technical chapters in this thesis report.
- Chapter One presents a collection of literature surveys and related works, including general information, related work, the problem statement and the objective of the thesis.
- Chapter Two provides a theoretical background of BC disease, explains the deep neural network structures and introduces field-programmable gate arrays (FPGA).
- Chapter Three, entitled ‘methodology’, illustrates the methods and theoretical parts with coding parts that form the project paradigms, such as the preprocessing model, developing the algorithms with the details and evaluating developed algorithms.
- Chapter Four presents the practical implementation of the FPGA board with experimental results.
- Chapter Five is meant to demonstrate the results and the outcomes of this study, along with the discussion and comparison of the results with the experiments.
- Chapter Six presents the conclusion and discussion.

CHAPTER TWO

THEORETICAL AND BACKGROUND

2.1. Anatomy of the Breast

2.1.1. Anatomy and Physiology of the Breast

The breast is the tissue of the thoracic muscles. Women's breasts are specialised tissue containing milk (glandular) and fatty tissue. The fat quantity determines the breast size. The milk-producing breast is arranged in 15 to 20 parts, known as lobes. Each lobe contains smaller structures, called lobules, in which milk is produced. The milk moves through a network of narrow tubes known as ducts. The ducts attach and form larger ducts, which essentially leave the skin in the nipple. The dark skin region around the nipple is known as the areola (Ellis, H., & Mahadevan, V., 2013). Connective tissue and ligaments sustain and form the breast. Nerves give the breast feelings. Figure 4 shows the collarbone, pectoral muscle, armpit and breastbone.

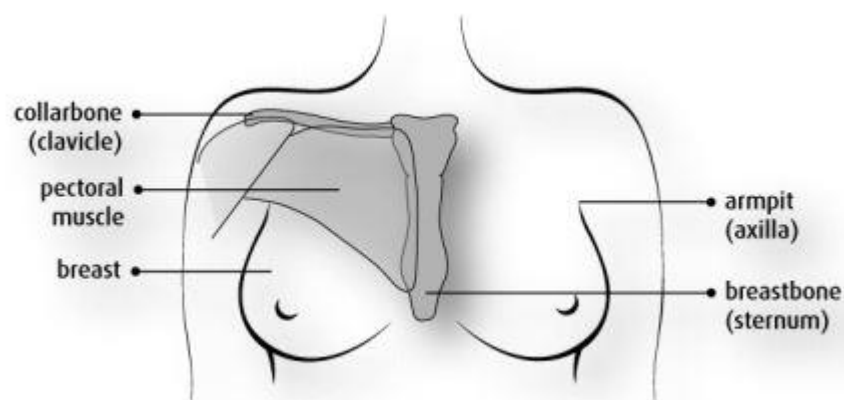


Figure 4. The surrounding structure and the breast (Zhang, H., 2015)

2.1.2. Structure of the Breast

The breast consists of a mass of glandular, fatty and connective tissue. The breast is lying on the chest muscle called the pectoral muscle. The female breast, as shown in Figure 4, stretches from below the collarbone (clavicle), the axilla (axilla) and over to the breastbone (sternum) (Ellis, H., & Mahadevan, V., 2013). The breast consists of:

Lobules: glands whose primary function is to make milk, and a portion of the breast lobe connects to the nipple through ducts. During pregnancy, the lobules of the breast enlarge naturally; this is an example of normal adenosis. After pregnancy and nursing, the lobules partially retreat.

Ducts: the milk ducts, also known as lactiferous ducts, are the tubes that bring breast milk from the breast tissue to the nipple. In the breast, there are between 15 to 20 milk ducts.

Fatty and connective tissue: these are terms used for describing breast tissue composed of nearly all fatty tissue. Fatty breast tissue does not appear dense in a mammogram, making identifying tumours or other breast changes simpler. Fatty breast tissue in older women is more prevalent than in younger women.

Areola: is the pigmented region around the nipple on the breast. More commonly, the areola is a small circular region on the body, with a different histology from the surrounding tissue or other small circular areas, such as inflamed skin. The adult human female nipple has numerous small openings arranged radially at the tip of a nipple (lactiferous duct). Other small openings in the areola, known as Montgomery's drums, are sebaceous glands, as shown in Figure 5.

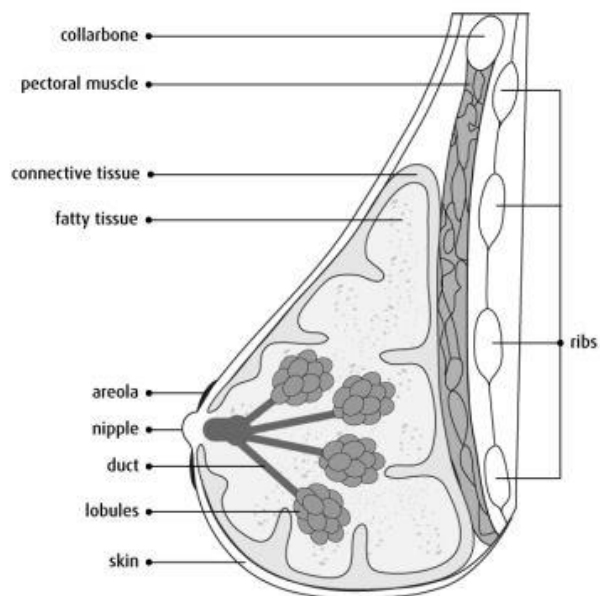


Figure 5. Breast structure (Zhang, H., 2015)

2.1.3. Breast Conditions

Breast cancer: Malignant (cancerous) cells that grow abnormally in the breast and, if left unchecked, spread to the rest of the body are known as BC. BC is a disease that affects almost primarily women, although men can be afflicted as well. Lumpiness, swelling or changes in the skin are all signs of BC, according to Dixon, J. M. (Ed.) (2012).

Ductal carcinoma in situ: Ductal carcinoma in situ (DCIS) of the breast refers to lesions made up of aberrant epithelial cells that are entirely contained within breast ducts and do not spread beyond the basement membrane. BC in duct cells has not penetrated or disseminated across the body. Women diagnosed with DCIS are likely to be healed, as shown in Figure 6 (A). Women with DCIS diagnoses are exceptionally likely to be cured since BC in duct cells has not infiltrated or spread farther into the body.

Lobular in situ carcinoma (LISC): While it is known as LCIS, it does not invade or spread and is not real cancer in the milk-producing lobular cells. However, in the future, women with LCIS are more likely to develop invasive BC, as shown in Figure 6 (B).

Invasive ductal carcinoma: This is a BC that starts in the cells of the duct but then invades the breast more deeply and is able to spread to the rest of the body (metastasising). The most common type of invasive BC is invasive ductal carcinoma, as shown in Figure 6 (C).

Invasive lobular carcinoma: A BC that starts in cells that produce milk but then deepens in the breast, which can spread (metastasise) to the rest of the body. Invasive lobular carcinoma is a rare type of cancer of the breast, as shown in Figure 6 (D).

Tubular carcinoma: A tubular form of invasive BC that is extremely rare. The most common cancer found in the colon is tubular in form. Tubular cancer is a kind of ductal carcinoma that originates in the tubular tissues of the body (IDC). The IDC's progress to other locations in the duct and invade other tissues, as shown in Figure 6 (E).

Medullary carcinoma: This is a type of breast carcinoma. BC that starts in the milk ducts is known as ductal carcinoma. The tumour on this BC is named for the medulla.

Medullary carcinoma of the breast accounts for 3 to 5% of all confirmed BC cases, as shown in Figure 6 (F).

Mucinous carcinoma: Also known as mucinous, it makes up just 2% of all BCs. Mucinous BC occurs in the breast's milk ducts, much as other forms of invasive ductal cancer, before spreading to the tissues around the duct, as shown in Figure 6 (G).

Papillary Carcinoma: This is a rare form of BC that only occurs in the ducts. The name derives from the papules, or finger-like projections, that can be seen as cells are examined under a microscope, as shown in Figure 6 (H).

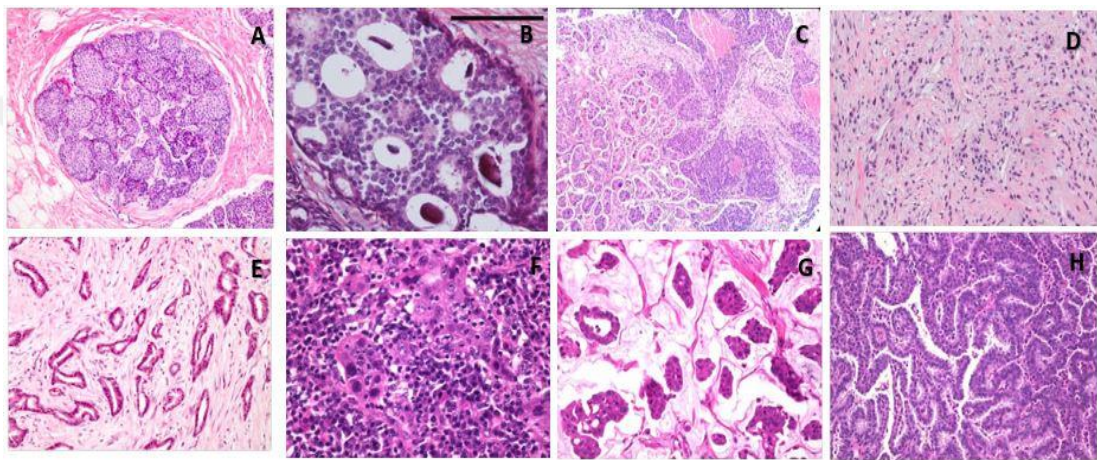


Figure 6. Histologically distinct breast cancers (Gour, M., Jain, S., & Sunil Kumar, T., 2020)

2.2. Introduction to Machine Learning and Neural Networks

2.2.1. Neural Networks

It was in the early 19th century that the first NN architecture was created. NNs, which were derived from neuroscience, have influenced ANN, modelling their formation, which often involves many artificial neurons. The neurons are arranged in a network to form a neural feed network. The synaptic dendrites are modelled by the weighted sum of the input received from other neurons. The neuron's input value (X_i) is based on the total weighted inputs (W_i), which are impacted by a constant value of 1. This input is then fed into a non-linear function that yields the neuron's output signal equation.

$$OUT_{neuron}^i = \sum_{j=1}^{K_{input}} INPUT^i \times weight^{ij} + Bias^i \quad (1)$$

The idea behind NNs is that they are designed to learn with minimal guidance from humans. When training is completed, the performance of the NN improves. The network knows how to connect data points that include problems with solutions, putting in the input and getting out the output to help find solutions to new problems. Before implementing any system, the network must be trained on limited data collection, as it is impossible to set up parameters like weights and biases without a training system. The initial weights are all tiny and have a haphazard weight distribution. Non-linearity supplies weighted inputs that have been multiplied when calculating the difference between the actual output (labelled samples) and the output produced by non-linearity. Because parameters (weights and biases) are not picked manually in the NN but learned during this process, the network must be trained on a specific collection of instances before it can be set up. A network of labelled training samples is provided, small random weights are used to begin the workout. Non-linearity provides inputs that are multiplied by weights. Weight values are optimised to reduce error. The backpropagation algorithm (Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. Y., 2011) is used to spread the results output across the network and reduce error. The stochastic gradient descent (SGD) (Bottou, L., 2010) is frequently used to solve this.

2.2.2. A brief Introduction to Machine Learning

Machine learning is an approach to artificial intelligence whereby ‘computers’, like humans, learn. Machine learning discusses how program structures can learn and develop experience automatically. Learning in this sense is not heart learning but identifying problematic trends and smart decisions based on data. The challenge arises from the fact that all conceivable options are too difficult to define given all possible inputs. Machine learning develops algorithms that discover information based on sound statistical and computational principles from real data and experience to tackle this issue. Machine learning is generally divided into two main types: supervised learning and unsupervised learning (Mohri, M., Rostamizadeh, A., & Talwalkar, A., 2018).

Supervised learning: For training, the learner is given different samples labeled, and then guesses what is wrong; this is the most common scenario in which problems with categorisation, regression and classification develop. If supervised training is employed, the network receives input and output, comparing the processed output to the desired output. Errors are then propagated back through the network to change the network's weights. This process repeatedly happens as the weights are changed continuously to reduce the error. A training dataset is called the training package. While improving the link weights, the data set gets processed numerous times by utilising the current framework (Mohri, M., Rostamizadeh, A., & Talwalkar, A., 2018).

Unsupervised learning: The learner only receives unlabelled training data and forecasts all unknown areas. As no labelled example is commonly available in this context, it can be difficult to measure a learner's output quantitatively. Clustering and reduced dimensionality are examples of unsupervised learning issues. In unsupervised training, input is given but not desired output. Then the device itself must determine what features it uses to group input data. This is also called adaptation. Clustering typically uses unsupervised learning (Mohri, M., Rostamizadeh, A., & Talwalkar, A., 2018).

2.2.3. Machine Learning

One way to understand machine learning is to think of it as a collection of algorithms that make decisions without being explicitly programmed. When a computer programme learns from experience E concerning some task T and some success measure P, it is said to learn from experience E if its performance on T, as calculated by P, increases with experience E (Mitchell et al., 1997). The experience E will be watching the software classify tens of thousands of images since this term pertains to image classification. Assignment T would be image classification, and the success measure P would be the likelihood that the image is correctly classified. More often than not, learning algorithms have been effective in fields such as:

- 1- Medical diagnosis
- 2- Computational biology applications
- 3- Speech recognition
- 4- Natural language processing

- 5- Computer vision tasks
- 6- Text or document classification
- 7- Optical character recognition (OCR)
- 8- Games
- 9- Unassisted vehicle control (robots, navigation)
- 10- Recommendation systems, search engines, information extraction systems

2.2.4. Deep Learning

An extensive list of fields has been changed by a subset of AI called machine learning in recent decades. Deep learning originated from NN, a type of machine learning (ML). Since its inception, DL has become increasingly effective, improving its performance greatly in nearly every application sector. Figure 7 depicts the taxonomy of AI. Deep learning is an ML class that was mostly developed in 2006.

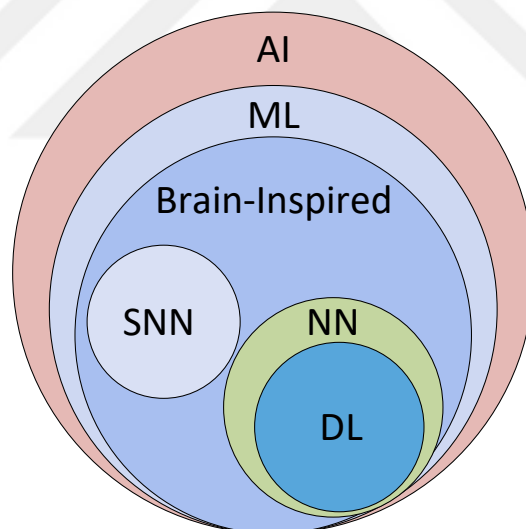


Figure 7. Artificial intelligence (AI), machine learning (ML), neural networks (NN), deep learning (DL) and spiking neural networks (SNN) (Chandra, A. L., Desai, S. V., Guo, W., & Balasubramanian, V. N., 2020)

DL takes place using an artificial NN, which comprises a number of levels organised in a hierarchy. At the initial level in the hierarchy, the network discovers something easy and transfers this information to the next level. This basic knowledge is taken

from that level, merged into something a little more complicated and passed on to the third level. This process continues as each level in the hierarchy builds on the feedback it receives from the previous level. For example, a deep study network's initial level of an image of a cat might use differences in a picture's light and dark areas to learn where the edges or lines are in a cat. The first level transfers this knowledge about the edges to the second level, combining the edges in simple forms like a right angle or diagonal line. The third stage blends basic types with complex shapes such as ovals or rectangles. The next step might merge the ovals and the rectangles into rudiments. The method continues until the top hierarchy is reached, where the network can classify cats. The network also learns to classify all other species it encounters with the cats when learning about them. Architectures of core DL, as we saw in the examples above of what DL and ML are, are the essential part of this form of AI data set used to train the NN model. For the image classification mission, many researchers used the well-known 'ImageNet' database. ImageNet is a WordNet hierarchy-organised image dataset. WordNet is a broad English lexical database. Nouns, verbs, adjectives and adverbs are grouped into a collection of cognitive synonyms (synsets). Synsets are interconnected through conceptual-semantic and linguistic relationships. A browser will navigate the resulting network of meaningfully linked terms and concepts (Fellbaum, 1998). A meaningful concept in WordNet, probably represented in several words or sentences, is called a 'synonym collection' or a 'synnet'. (Fellbaum, 1998). WordNet has more than 100,000 synsets, with more than 80,000 nouns. (Fellbaum, 1998).

Classification: Predicting a data point's class is known as classification. In other contexts, classes are referred to as goals/labels or divisions. For example, it might assign items to categories like politics, business, sports or weather, while picture classification might assign items to landscape, portrait or animal. The number of categories in such tasks is typically limited, but in certain complicated tasks, such as OCR, text labelling or speech recognition, it may be high and even unbounded (Mohri, M., Rostamizadeh, A., & Talwalkar, A., 2018).

Ranking: Object ranking is a relatively new field of study within information retrieval that is concerned with ranking objects, such as named entities, in the context of a user query or application. The canonical ranking example is web search, which returns web pages specific to a search query. In the framework of the implementation of knowledge

extraction or natural language processing systems, several other rating issues occur (Mohri, M., Rostamizadeh, A., & Talwalkar, A., 2018).

Clustering: The objective of clustering is to divide a population or set of data points into a number of categories. Data points in the same groupings are more comparable to one another than data points in other groups. Put another way, the goal is to separate groups with similar characteristics and assign them to clusters for analysing very large data sets. Clustering algorithms, for example, seek to classify ‘communities’ within large numbers of individuals in the sense of social network research (Mohri, M., Rostamizadeh, A., & Talwalkar, A., 2018).

Manifold Learning and Dimensionality Reduction: Carry out an operation on the items, thus transforming their original representation into a lower-dimensional representation. One of the key difficulties in image, video and signal processing, in general, as well as ML, is dimensionality reduction of high-dimensional data. The digital image preprocessing method can include multiple elements (Mohri, M., Rostamizadeh, A., & Talwalkar, A., 2018).

2.2.5. Convolutional Neural Networks

In the 1990s, the CNN algorithm was applied to the manuscript digit classification problem, and it was successful (LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P., 1998). This led to the advancement of CNNs, which led to additional advancements and superior results in recognition tasks. Deep neural networks (DNNs) may have the advantage over CNNs in terms of the ability to provide facial recognition, but they fall short of human visual processing ability and require greater processing. The process learns and extracts 2D functionality abstractions, but it also structures 2D and 3D images. The maximum pooling layer of CNNs absorbs changes in shape. Also, the CNNs have far fewer parameters than a fully connected network of comparable size, where each connection is sparse, meaning it is only active in a subset of nodes and has fixed weights. In particular, CNNs can better overcome the gradient problem by decreasing gradient-based learning. CNNs are known to develop an extremely well-tuned weighting algorithm if a gradient-based algorithm can trace the complete network and minimise an error criterion (LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P., 1998).

Figure 8 depicts two components of the CNNs architecture: feature extractors and a classifier. Every function extraction layer has layers for each network layer that receives information from another layer. In the previous layer, the origin is output and automatically transferred to the next layer as input. The three types of layers in the CNN architecture are concentration, max pooling and classification (Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... & Asari, V. K., 2018). Convolution and max pooling layers exist in the lower and middle layers of the network. The levels numbered one through ten represent 'turns' (in a game of pool, one would consider ten turns) for max pooling, while odd-numbered layers are for turning and bending. The feature mapping is made up of max pooling layer output and convertible layer output nodes. Each layer is produced by combining two or more prior layers. Each layer is associated with a limited section of the connected plane, and the next layer's nodes are linked to these regions. The input node is translated into a feature extraction point via the convolution layer node. The organisation of features occurs from the basic components. The size of a feature decreases when the feature extends to the top layer or stage. Depending on the kernel's size, procedures like convolution and max pooling depend on how long they take. To better represent the classification of an input image, the number of feature maps should increase (Schmidhuber, J., 2015). The last layer of the CNN is utilised as a path to create a grading layer, which is a fully connected network. It has come about thanks to better performance and the use of neural feed networks as a classification. The necessary number of variables (in the form of weights) are used to populate the dimensions of the weight matrix in the final NN, which is used in the classification layer. This is quite complex but also quite expensive concerning network or learning parameters. Numerous options exist as an alternative to networks, such as average pooling and worldwide average pooling, which are entirely connected. The classification of a particular class is made in the top classification layer, where a SoftMax layer is used to evaluate the score. The classifier categorises the findings in groups based on its highest-ranking class. Following the next part, details on the mathematical aspects of each layer of CNNs are presented.

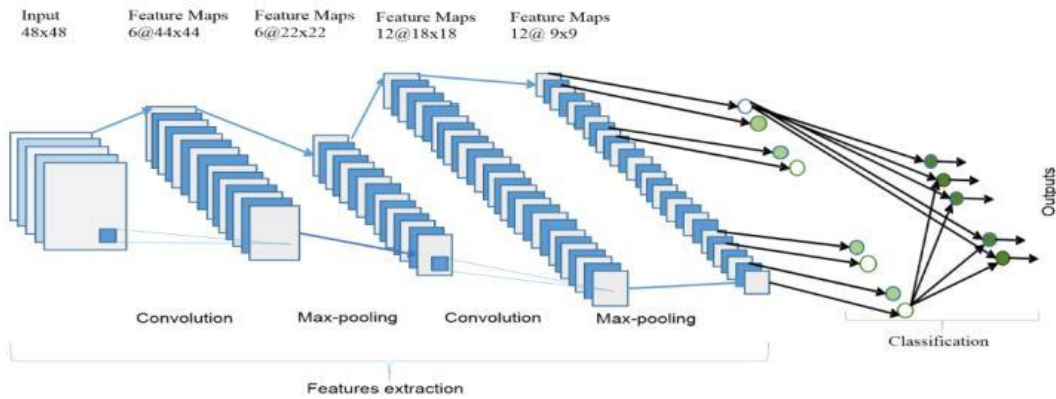


Figure 8. Features of extraction and classification of the visual image using convolution neural networks (Alom et al., 2018)

2.2.5.1. Convolutional Layer

A CNN's key component is the convolutional layer, which holds most of the model's operations. A three-dimensional multiply-accumulation (MACC) operation is used in the convolutional layer. This layer also executes the mathematical method known as convolution, which includes the operation. Figure 9 depicts a kernel/filter with a given input multiplied by an input set (the receptive area) and then combining weighted inputs with a given input multiplied by an input set (the receptive area).

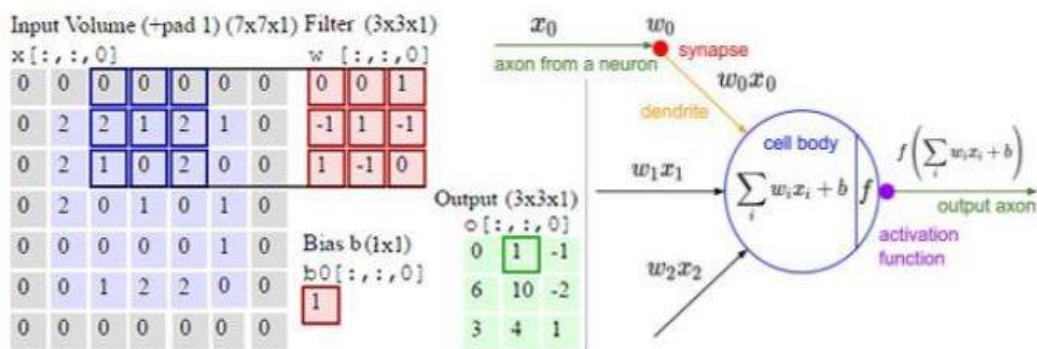


Figure 9. An illustrative example of arithmetic operations performed in the convolutional layer of the CNN network (Hamdan, M. K., & Rover, D. T., 2017)

In NNs, where the constant value effectively transposes the line, bias is akin to the role of a constant in a linear function. The input to the activation function in a scenario with

no bias is 'x' multiplied by the connection weight 'w0'. The activation function introduces non-linearity and restricts the output to an appropriate range by an activating function like the rectified linear unit (ReLU). Effects of the activation feature are transmitted in the next layer to corresponding neurons. The spatial output computation size is shown in Equation 2.

$$Output_{size} = \frac{(Input_{width} - Filter_{size} + 2 \times padding)}{Stride} + 1 \quad (2)$$

Three hyperparameters occur when determining the size of the output in Equation 2:

1. Depth. It is the number of filters we want to use, an input learning to look for something different.
2. Stride. The move. When we pass the filter along with the input, we must determine how to do several pixels every time.
3. Zero padding. Often it is simple to pad the volume of input types; a border of one-pixel size is used with zeros around the edge, summarising a convolutional layer.

2.2.5.2. Non-linearity (Activation Function)

The most frequent activation functions are sigmoid, tanh and ReLU. Tanh = $\tanh(x)$ activation functions must be applied to each pixel in the input to ensure non-linearity in the network and remove unneeded information. Unlike ReLU, which converges faster during training, it requires a longer CNN setup time (Krizhevsky, A., Sutskever, I., & Hinton, G. E., 2012). In addition, ReLU is generally defined as a zero-threshold operation $ReLU = \max(0, x)$. The various forms of activation functions are shown in Figure 10.

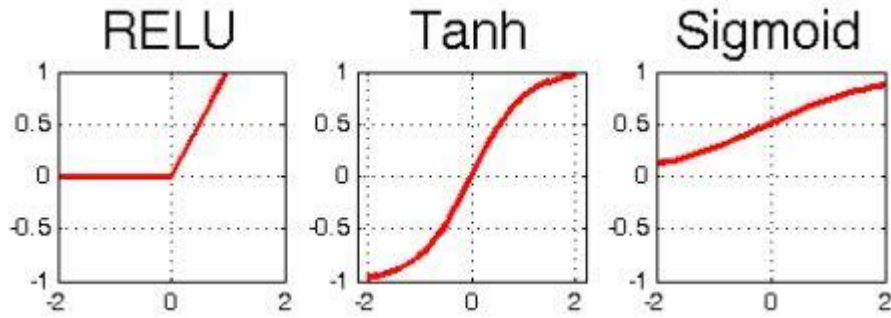


Figure 10. Activation functions: ReLU, tanh and sigmoid (Dureja, A., & Pahwa, P., 2019)

2.2.5.3. Stride

Another basic building component of convolution utilised in CNNs is a strided convolution. This can be thought of as downsampling of the complete convolution function's contribution. If every s pixel is sampled in every direction in the display, a convolution function can be created down the sample c that is shown in Equation 3.

$$Z_{i,j,k} = c(K, V, s)_{i,j,k} = \sum_{l,m,n} [V_{l(j-1) \times s + m, (k-1) \times s + n} K_{i,l,m,n}] \quad (3)$$

The stride of this downsampled convolution is referred to as s . It is also possible to designate a different stride for each motion direction. Figure 11 shows a visual illustration of the concept (stride with convolution). Figure 11 shows an example of how to employ a double stride. A two-stepped convolution with a phase of one-pixel is the same as a one-pixel forward skip, and a stride of one-pixel can be seen as convolution accompanied by downsampling. Obviously, the two-step solution requires the computation of several values and then discarding them (Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y., 2016).

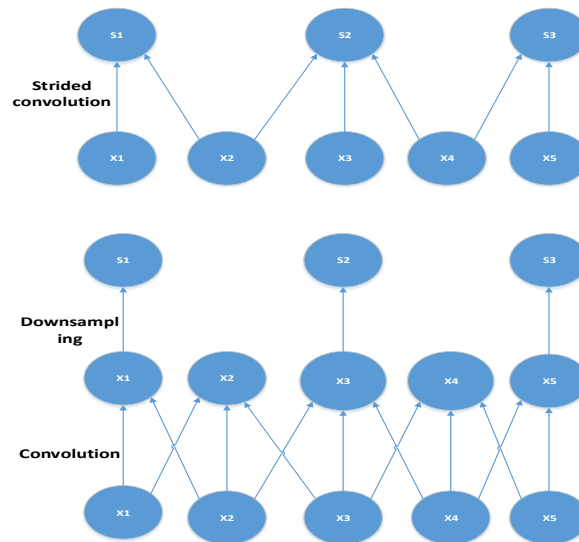


Figure 11. Example of stride with convolution (Zhou, Y., Zhao, H., Chen, J., & Pan, X., 2019)

2.2.5.4. Padding

The single most important characteristic of any convolution network is the capacity to initialise the input to a uniform V . Without this explicitly, the user interfaces element, the image's width would be one pixel less for each level of the representation. Zero-padded feedback to monitor the kernel width and the scale of the resulting picture independent of input growth is required of the network, which shrinks the network's spatial extent (Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y., 2016).

2.2.5.5. Pooling layer

Another component of a CNN is the pooling layer. Its purpose is to gradually shrink the representation's spatial size in order to reduce the number of parameters and computations in the network. Each feature map is treated separately by the pooling layer. The most popular pooling strategy is maximum pooling. Non-linear subsampling, which reduces the number of network layers in our function, is known as spatial pooling. Common ways to pool include max and average pooling. Max pooling will take the maximum value of the neurons and move it to the next layer,

leaving just the neurons whose values have the maximum in the pooling filter. In average pooling, every neuron in the filter contributes to the output of each neuron in the next layer, as seen in Figure 12.

2.2.5.6. Fully-Connected layer

The fully connected (FC) layer typically comes before the classification layer, which includes the largest number of parameters, as each neuron in this layer is connected to all neurons in the layer before it, and parameters on the relation between

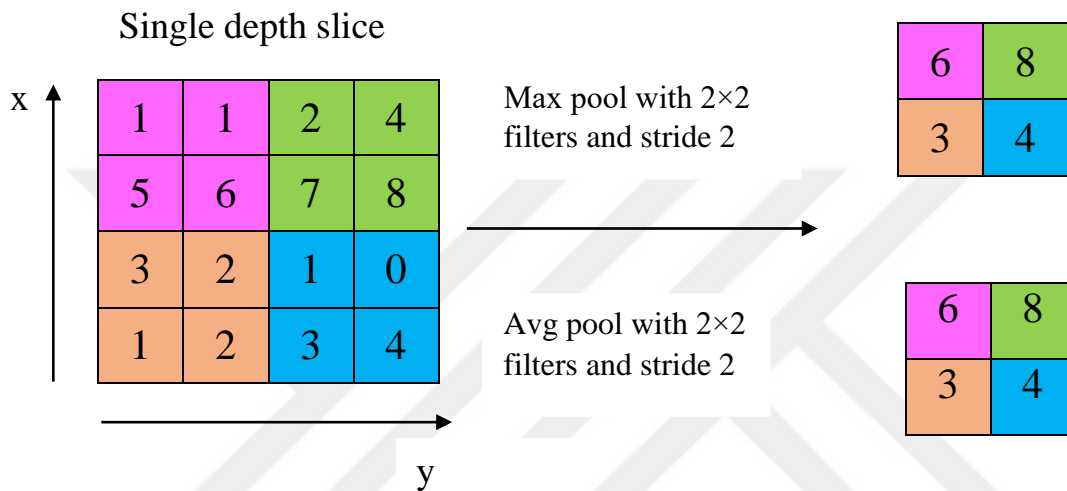


Figure 12. Pooling output averages and maximums for a 2×2 filter with a stride of 2 (Akhtar, N., & Ragavendran, U., 2020)

those neurons are translated. Input in this layer is multiplied by corresponding weights, biases added and nonlinearity introduced as convolutional layers, as shown in Figure 13.

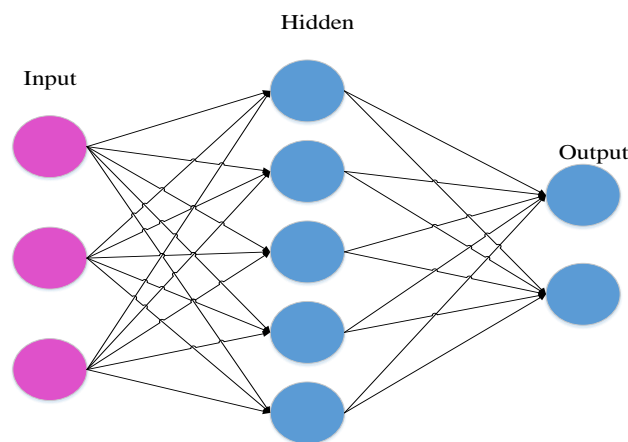


Figure 13. A three-layer fully connected matrix (Rørmann Olsen, I., 2018)

2.2.5.7. Dropout layer

The dropout layer is a method used in large CNNs to prevent overfitting during training. This layer drops a selectable percentage of its connections randomly not to learn specific mappings and imposes redundancies into the weights learned. Figure 14 is an example showing the use of the dropout method used in CNN networks.

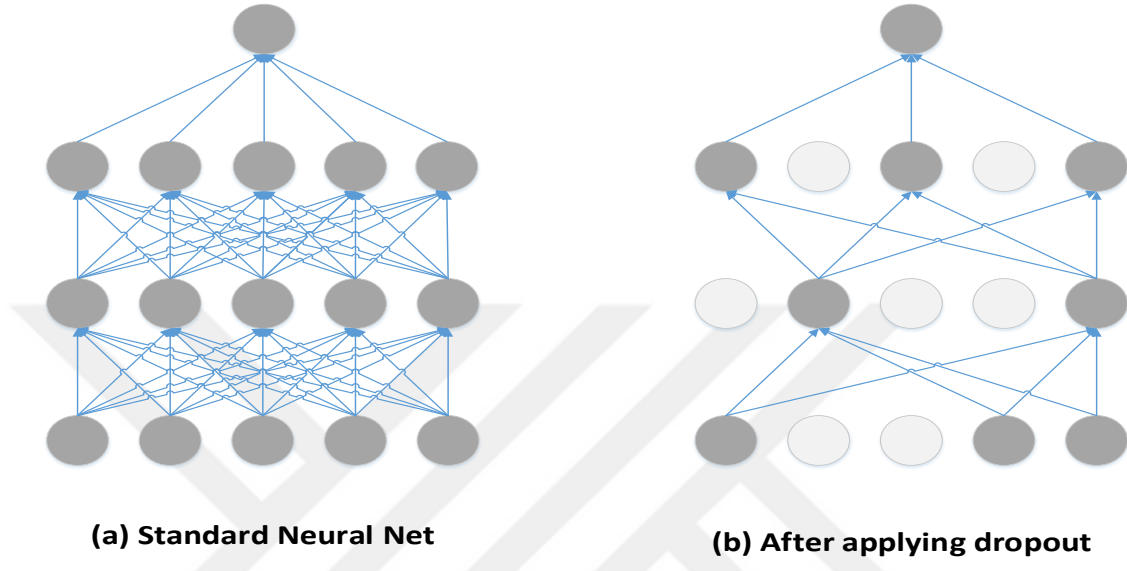


Figure 14. Representation of dropouts (Jørgensen, H., 2017)

2.2.5.8. Classification layer

The final class output of the preceding layers needs to be classified by the last layer of a CNN, and this layer's primary responsibility is to do so. This layer uses SoftMax for the classification process. In short, the SoftMax classification takes a previously computed value of z_i (in the range of 0 to 1) and changes it according to P_i to fulfil the common scaling method as shown in Equation 4. The accuracy of each class's label comparison is matched to the results of the SoftMax probabilities to check the quality of the model.

$$P_i = \frac{e^{z_i}}{\sum_k e^{z_k}} \quad (4)$$

2.2.5.9. Grouping layers

Now that the fundamental principles of CNNs have been defined, we can look more closely at how a system with CNNs employs those layers. As defined in the

classification layer section, a simple convolutional network is a sequence of layers and transforms a volume of activation by a differentiated feature in each network layer.

The fundamental layers of constructing a CNN

- i Convolutional Layer
- ii Rectified Linear Activation Layer
- iii Pooling Layer
- iv Fully Connected Layer

The complete architecture of the CNN stacks these layers. Look briefly at an illustration of how feasible CNNs are designed. If there is an input image for the CIFAR-10 vehicle dataset that is 32 x 32 x 3, the size is 32 pixels high and 32 pixels wide, and in this case, three channels or colours. The CIFAR-10 dataset is a much smaller image database, with each 32 x 32 x 3 image and only ten groups, like the ImageNet database. As shown in Figure 15, the layered architecture may have a simple CIFAR-10 classification CNN (Sinha, T., Verma, B., & Haidar, A., 2017).

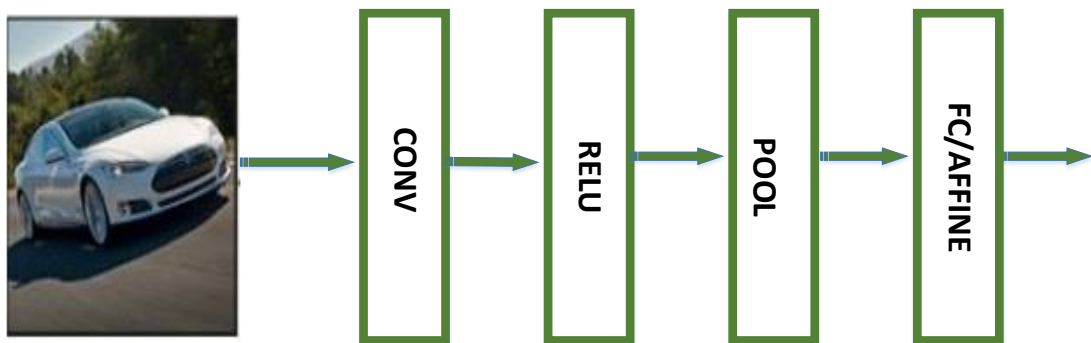


Figure 15. CNN architecture as an example (Sinha, T., Verma, B., & Haidar, A., 2017)

CONV Layer: The CONV layer assesses the output result as $32 \times 32 \times 12$ inches if 12 filters are employed.

ReLU Layer: In this layer, the size is not changed ($32 \times 32 \times 12$).

Maxpool Layer: The POOL layer performs spatial sampling by reducing the dimensions that convert the image dimensions to $16 \times 16 \times 12$.

Affine / Fully Connected Layer: The class values are decided in the FC layer, meaning the result is a volume size ($1 \times 1 \times 10$) in which each of the ten integers equals a class score, a 10-image CIFAR-10 image set classifies to a single class. This NN is typical in that each neuron is connected to the names of the number that came before it, as its name suggests. This occurs when CNN reads through the image layer by layer, as shown in Figure 16.

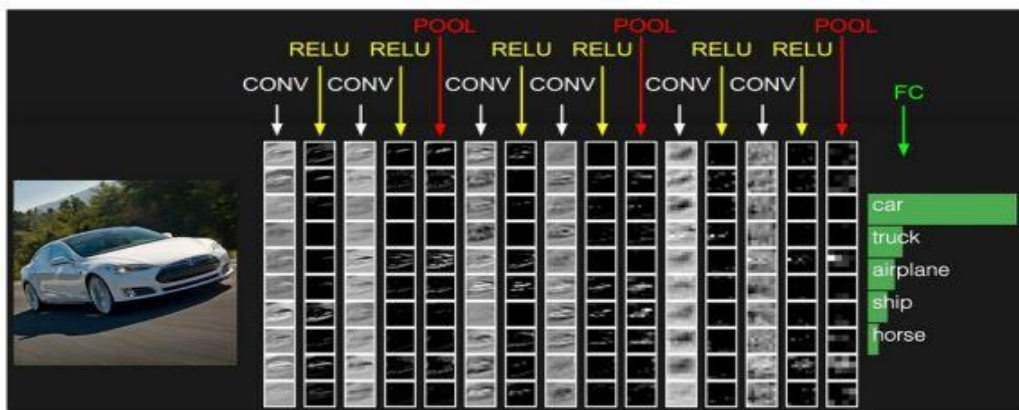


Figure 16. Example of image classification using CNN architecture (Jiang, Z. 2019, December)

2.3. Popular CNN Architectures

This section examines several important cutting-edge CNN architectures. Basic layers, such as the convolution, sub-sampling, dense and SoftMax layers, are generally utilised as the basic collections for the most significant CNNs. Convolutional and pooling layers are combined with linked and SoftMax layers in the architectural stack of many convolutional layers for all these networks.

2.3.1. LeNet

Although LeNet was proposed in the 1990s, the algorithm's implementation by 2010 was hampered by factors, such as processing ability and memory capacities. LeCun proposed backpropagation techniques for CNNs and experimented with handwritten numbers to obtain state-of-the-art accuracy. LeNet-5 (LeCun et al., 1998) is a well-known architecture. The LeNet-5 has two convolution layers, two subsampling layers and two fully sampled layers, as shown in Figure 17, and layers linked with a Gaussian relation output layer. The total weights and accumulations (MACs) are 431 thousand and 2.3 million, respectively.

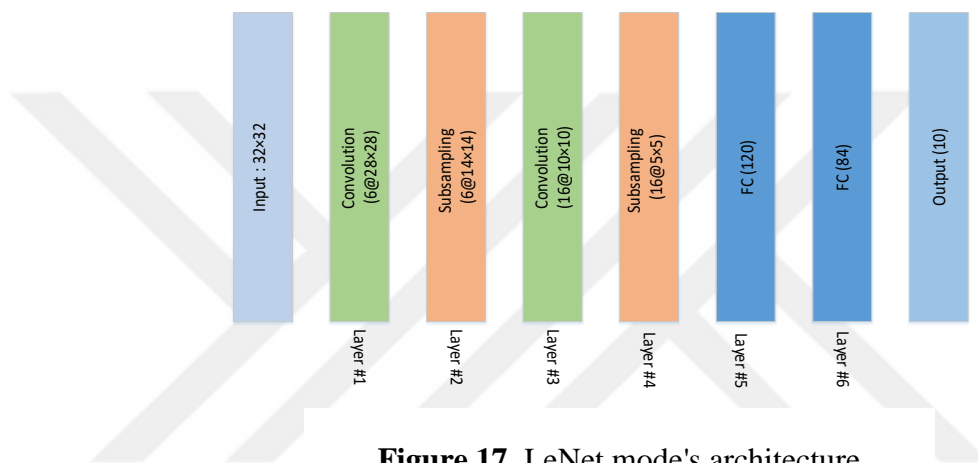


Figure 17. LeNet mode's architecture

2.3.2. AlexNet

AlexNet is a pioneer in computer vision, excelling at both ML and traditional computer programming (Krizhevsky, A., Sutskever, I., & Hinton, G. E., 2012). DL developed importance for image recognition and classification in ML and computer vision. Figure 18 depicts AlexNet's architecture. The first convolutional layer is modified and combined with 96 distinct layers using local response standardisation (LRN). The signal is received by 11 11 squares. The 3 x 3 filters use two stride measures, one small and one large, and 5 * 5 filters finish the second layer filtering procedure. The third, fourth and fifth convolution layers use 384, 384 and 296 map filter types, respectively, with three and three filters. This leaves the network with two fully linked layers to finish the SoftMax layer. This model is made up of two networks with the same structure and maps. This network uses LRN and dropout. LRN can be used in two ways. First, when normalising a single channel, either the N to N patch or

feature maps are applicable. A character set or a keyboard can be added to the LRN (a third dimension neighbourhood with just one pixel or location) (Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012).

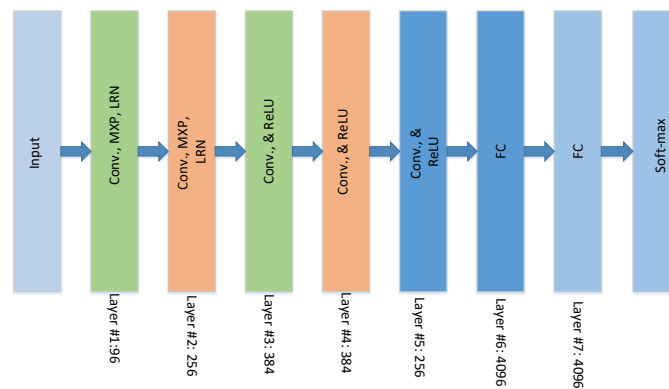


Figure 18. AlexNet model's architecture

AlexNet has three convolution layers, two linked. When handling the ImageNet dataset, averaging the total number of parameters for each layer for AlexNet yields 17.7 thousand parameters. The input sample is 224x224x3, the receptive field or filter (kernels or masks) is 11, the stride is 4, the output is 55x55x96, the first layer weight was 290400 neurons (55 total) and 364 weights. They are $290400 \times 364 = 105,705,600$ of the first convolution layer's 105,705,600 parameters. The network has 61 million weight and 724 million MACs.

2.3.3. VGGNet

Simonyan and Zisserman (2014) proposed the visual geometry group (VGG), a CNN model. The most important contribution of this research is that it demonstrates the importance of network depth and improves the accuracy of CNN classification or identification. Each of VGG's convolutional layers uses ReLU activation. In a single max pooling layer, numerous completely connected layers use ReLU activation. The model's conclusion is SoftMax. VGG-19 is a 3 x 3 two-staged filter. Simonyan and Zisserman (2014) suggested VGG-11, VGG-16 and VGG-19 models with 11, 16 and 19 layers, as shown in Figure 19.

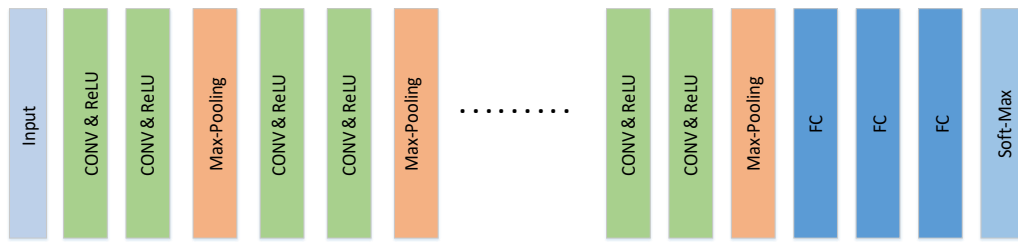


Figure 19. A structure of VGG network building blocks: convolution (Conv) and fully connected (FC)

The VGG-E variants all feature three connection layers. VGG-11, VGG-16 and VGG-19 have eight to sixteen convolution layers. VGG-19 has 138 million weights and 15.5 M.

2.3.4. Residual Network (ResNet)

ResNets are a type of NN that is commonly utilised as the backbone for computer vision applications. In 2015, this model won the ImageNet challenge (Szegedy et al., 2015). Kaiming created ResNet to create new ultra-deep networks without the problems of disappearing gradients. The ResNet has layer numbers from 34 to 1,202. Real-world applications frequently use ResNet-50, a network with 49 convolution layers and one fully linked layer at the end. Figure 20 depicts the ResNet architecture's foundation. In total, the network employs 25.5 M weights.

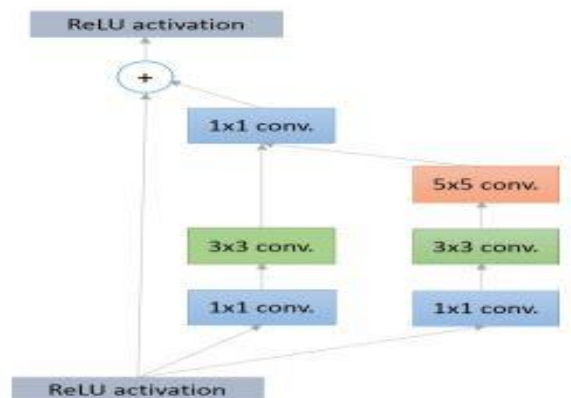


Figure 20. Basis of the residual network architecture

The ResNet design is based on VGGNet; however, convolutional layers only employ 3 x 3 filter kernels. ResNet designed a 34-layer flat network with and without shortcut connections, as shown in Figure 21, also creating networks from 34 to 152 layers. Overall, ResNet's 34-layer network defeated ImageNet's 152-layer network in the 2015 ImageNet LSVRC competition. ImageNet 2015's LSVRC network architecture won.

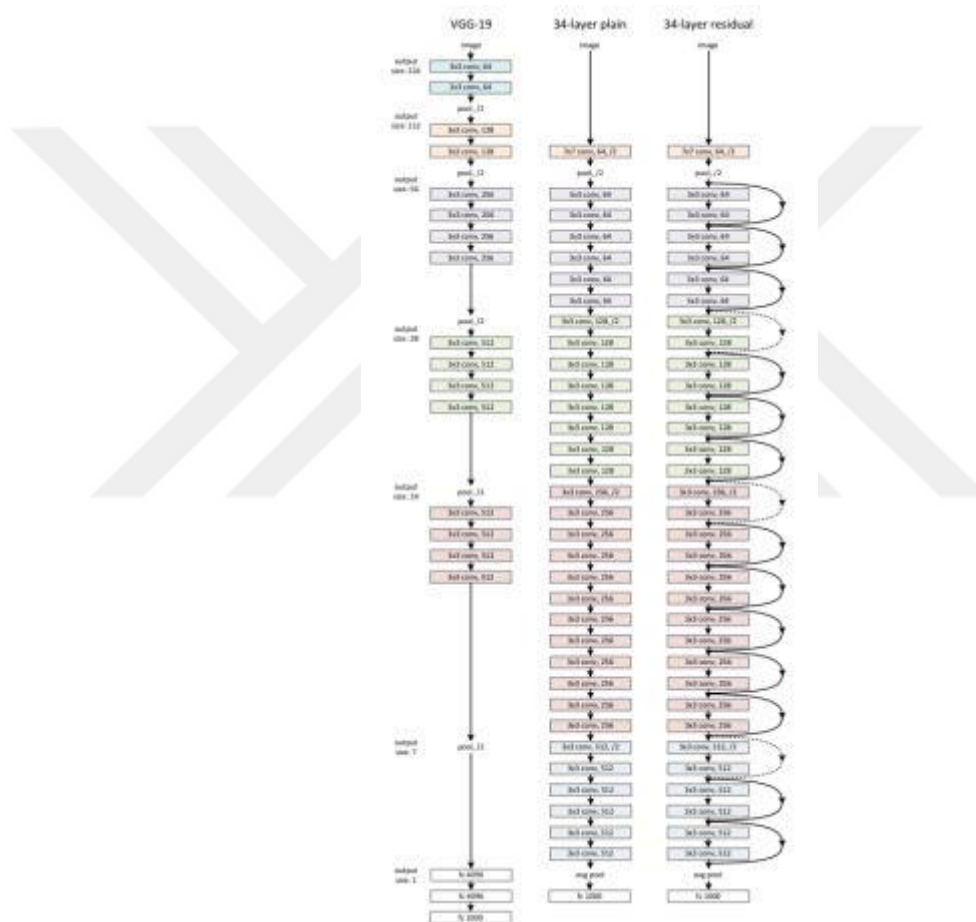


Figure 21. Residual learning: block of the residual network architecture

2.4. Introduction to Field Programmable Gate Arrays

This section offers a brief introduction to the Field-Programmable Gate Arrays(FPGAs), where FPGAs are shown to demonstrate characteristics, strengths, and limitations compared to other hardware platforms such as central processing units

(CPUs), application specific integrated circuits (ASICs) and graphics processing units (GPUs).

2.4.1. Field-Programmable Gate Arrays

FPGAs are prefabricated semiconductor devices that are coupled to 2D arrays of adjustable logic blocks via programmable logic (CLBs, or logic slices). Interconnect resembles a wired network that connects logic blocks horizontally and vertically with switch boxes (matrices of switches) at any crossroads with horizontal and vertical bundles, as shown in Figure 22. The logic blocks, fixed-function units and connections are electronically programmed by creating a bitstream configuration to add some digital design into the unit. With the reprogramming of FPGAs numerous times, the configuration activity is frequently recorded in SRAM memory cells (Trimberger, S. M. (Ed.), 2012). In 1967, Wahlstrom proposed the first static memory-based (SRAM) FPGA, which used configuration bits stream to enable logic and connectivity settings. In 1984, Xilinx released the first commercial modern-day FPGA. Arrays of adjustable input/output logic blocks were incorporated. Hundreds of thousands of customisable logic blocks, as well as a plethora of hard-held functional units, are available in today's high-end FPGA generations, allowing for quick and efficient deployment of common (Wahlstrom, S. E., 1967).

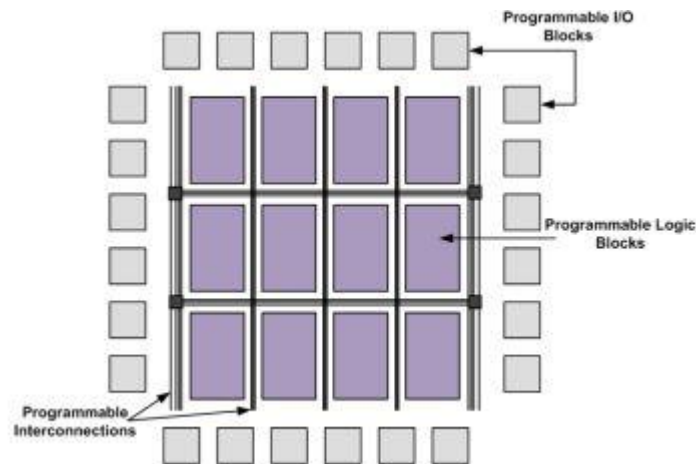


Figure 22. FPGA internal architecture (Wahlstrom, S. E., 1967)

2.4.1.1. Programmable Logic

In FPGA, programmable logic blocks provide the hardware resources for supporting basic computation and storage elements in digital systems. Combinational logic, a flip-flop and rapid carrying logic to save costs can be found in a standard simple logical element. Some modern FPGAs have memory and processing blocks with different capabilities in a complex combination that also contain partition blocks, multipliers and DPS blocks (Trimberger, S. M. (Ed.), 2012).

2.4.1.2. Programmable Interconnect

FPGAs contain configurable routing to connect logic blocks and I/O blocks. Pass transistors, tri-state buffers and multiplexers are used to create the needed link. In general, a logical cluster is connected via a multiplexer or pass transistor, while all three are employed for connecting larger overall routing structures. Many FPGA routing structures are employed worldwide, including 'island-style', 'cellular', 'bus-based' and 'registered' architectures.

2.4.1.3. Programmable I/O

Internal components of an FPGA are connected via external input/output pads or programmable I/O interfaces with an FPGA's functional blocks and routing topologies. I/O pads and their corresponding logic circuitry compose the I/O cell, a key component of any chip. The specification differences for supply voltage and reference voltage are making it hard to build I/O programmable blocks. Choosing an accepted standard is among the most important considerations in the I/O part. In any design project, if more I/O cells are going to be used (as is customary), then the silicon cell area must be increased (Trimberger, S. M. (Ed.), 2012).

2.4.1.4. Customizable Functional Blocks

More relevant programmable functional blocks, such as embedded memory, arithmetic logic (ALUs), multipliers (MUXs), digital signal transformers (DSP48), and embedded (DSP48) Microprocessors, were added to the FPGA design over time. As a result, heterogeneous FPGA systems were created.

2.4.2. FPGAs Versus Other Hardware Platforms

FPGA-based systems, which offer the capability to freely programme general logic blocks, win over conventional systems, which offer limited programmability in their general logic blocks. FPGAs can be programmed to perform at a high level. Accelerators are used for extremely particular tasks, resulting in greater processing speed and efficiency. FPGAs suit mobile device applications better than GPUs. Hence, they are more effective as a tool. This added complexity comes at the cost of longer development time as designers must account for using the available hardware and mapping target algorithms to the FPGA architecture efficiently. The computational intensity of the DSPs is surpassed by the FPGAs, which break the sequential executable model and process more per clock cycle. They take full advantage of hardware parallelism. Hardware-level input and output control (I/O) has improved performance, and it is specialised, which is what an application needs. Unlike conventional microprocessors, FPGAs use software that does not have operating system-induced difficulties in delivering real-time performance and uses deterministic hardware for every task (Olivito, J., Gran, R., Resano, J., González, C., & Torres, E., 2015).

2.4.3. FPGAs versus ASICs

Customised semiconductor devices are ASICs. In contrast to FPGAs, ASICs have no overhead area or time due to configuration logic and generic interconnections, thus creating the fastest, most energy-efficient and smallest systems. However, the sophisticated manufacturing processes for ASICs lead to a very long and complicated development round and very high repetitive engineering costs, which require a first-time design methodology and extensive design verification. ASICs are thus mainly suitable for high volume, cost-sensitive applications. With their reprogrammability FPGAs are best suited to prototype and short development cycles, in which concepts can be tested in hardware without the long manufacturing process of custom design. FPGA chips are upgradable in the field and do not require the time and cost of ASIC redesign. Interfaces based on ASICs may cause maintenance challenges and future compatibility, while FPGAs can be reconfigured to comply with future modifications that may be necessary (Kuon, I., & Rose, J., 2010).

CHAPTER THREE

METHODOLOGY

3.1. Overview

Medical data science applications involve using data analysis to diagnose patients for different diseases. However, the data collection technique is vital to these processes, and, hence, it needs to be selected with care. Data can be collected from hospitals using data entry techniques. When a battery of tests is performed, hospitals usually record the results and store the information in their systems so that it is available if needed in the future. A specialist doctor is always available to manually diagnose the case and accurately predict the disease.

Eventually, data is collectively used from different cases and their target (diagnosis) to be analysed using an NNs approach. Prediction of the disease can be made using ML and DL tools. This may provide high accuracy and time-efficient mean of disease diagnosis (Chen, K., Huang, L., Li, M., Zeng, X., & Fan, Y., 2018).

In this chapter, a detailed discussion of ML approaches, more likely DL algorithms, is discussed. Big data from various cases were referred to for training the algorithms. A BC disease prediction system was implemented using seven different machine learning algorithms, namely KNNs, Gaussian naive Bayes, kernel naive Bayes, linear discriminant, logistic regression, SVM and decision tree algorithms. AlexNet, VGG-16, LeNet-5, ResNet-18 and ShuffleNet are five deep learning algorithms that have been implemented for predicting cancer. The approach covers the so-called performance enhancement of the prediction from the view of the accuracy of disease prediction, the time required for the prediction and the amount of error presented in the results after the training process (Abhigna, P., Jerritta, S., Srinivasan, R., & Rajendran, V., 2017).

The performance metrics were also initiated in order to evaluate the difference in the prediction performance of the tools. The details of performance measures utilised for model performance assessment are covered in the following sections of this chapter.

3.2. Database

Images in the BreacKHis (Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. 2015) database of breast tumors, both benign and malignant. A hematoxylin-eosin-stained breast tissue biopsy slide is used to generate the samples. A surgical biopsy is used to collect the samples, which are then prepared for histological analysis and labelled by pathologists in Brazil. If a lesion does not meet any of the criteria for malignancy, it is considered histologically benign. In most cases, benign tumors are harmless, slow-growing, and limited in scope. As with cancer, a malignant tumor can infiltrate and destroy nearby tissues and then spread, resulting in death.

3.3. Preprocessing

In order to predict the disease occurrence using biopsy images, the record from every patient was studied after the data preprocessing. The overall process can be summarised as the following:

- 1) Image data were prepared by labelling each image with the exact class name. Thus, two classes were made, 'zero' (which indicates no infection/cancer occurrence) and 'one' (which indicates infection/cancer occurrence).
- 2) Image size was reduced to mitigate the load on the classifiers and expedite the training process. Image resizing was vital for accuracy, and a large amount of irrelevant data were omitted.
- 3) Image normalisation was made to minimise the variance between the images for optimising the training performance. Each image was normalised by dividing each pixel within each image by 225, which is the maximum pixel value for a coloured image. In order to ensure high performance of disease prediction, a smart classifier deploying a CNN was used. This model configuration deployed in this thesis is illustrated in Tables 12 and 13. This model is considered a smart version of the artificial NN, and it uses the same architecture of recurrent NNs except for some changes in the weights. It is capable of processing a large number of data values at once. It differs from the classical feedforward NN model by its large feedback loops between its layers. The main terminology of the long short-term NN is the word 'gates', which is given to the layers. More likely, the input and output gates represent the input and output layers in a normal (classical) feedforward NN, while the forget layer represents the hidden layer popular in classical feedforward NNs.

3.4. Image Decomposition

Digital images are populated in the science and engineering domains due to their efficiency in prescribing natural objects. However, the word ‘digital’ refers to those images that allot a value to each dot. As a result, images consist of a vast number of dots of varying intensity (value), generating the final value of the image, which gives it the final appearance that can be perceived by the naked eye.

Images, in general, can be classified into four types: red, blue, green and under infrared. On the other hand, a binary image represents an image with just black and white colours and pixels with binary values, most commonly zeros and ones. Figure 23 shows the four bands discussed above, as well as the united colours image, often known as a colourful image, which incorporates all four channels.

It is noteworthy that in the under infrared images channel, the same image can be used with special applications, such as remote sensing in satellite image processing, which can supply the other information that helps recognise the geological objects from satellite images.

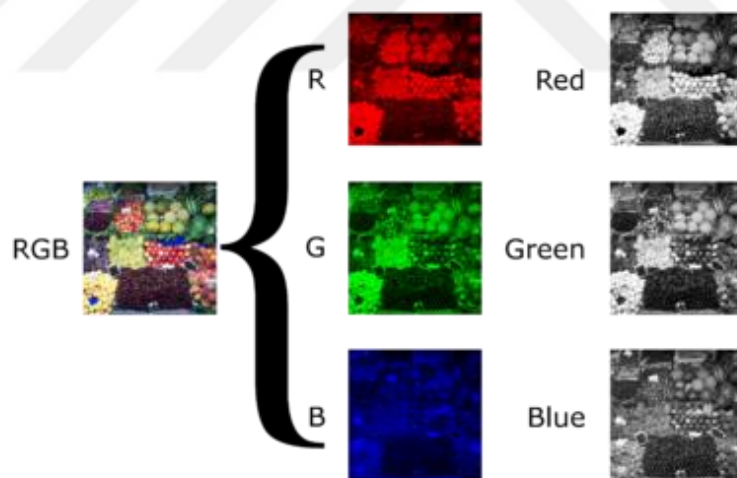


Figure 23. Coloured and RGB image channels channels
(Mandwi, I., Bhondge, B., Thakre, K., Dhande, G., & Patiye,
I., 2016)

In general, the main point of image processing is understanding what the images equal in mathematical scale. Images can be represented by a combination of their (three/four)

channel information that forms the final image colours. The colour image can be represented with a three-dimensional matrix with dimensions of X, Y and Z.

The final shade of a binary image is formed by a mixture of zeroes and ones based on the pixel location and intensity. Assuming that image S is a binary image of nine pixels, the representation can be seen in the following matrix.

$$s = \begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{matrix} \quad (5)$$

Whereas, the colored image is a combination of three channels namely, red, blue and green. The colour red is represented by the matrix 6, whereas the colors blue and green are represented by the matrices 7 and 8.

$$R = \begin{matrix} 1.2 & 0.5 & 1 \\ 1.25 & 0.98 & 1.8 \\ 1.32 & 2.1 & 0.84 \end{matrix} \quad (6)$$

$$B = \begin{matrix} 1.8 & 0.89 & 0.25 \\ 0.112 & 0.285 & 1.69 \\ 1.12 & 1.1 & 1.07 \end{matrix} \quad (7)$$

$$G = \begin{matrix} 1.879 & 0.213 & 0.748 \\ 0.396 & 0.203 & 1.369 \\ 1.102 & 1.279 & 0.258 \end{matrix} \quad (8)$$

Therefore, the combination of the three channels can produce the final result of the image, the representation can be seen in the following matrix 9.

$$Colour = \begin{matrix} R \\ B \\ G \end{matrix} \quad (9)$$

3.5. Neural Network Modelling

With the advancement of ML algorithms, new approaches are being implemented to provide superior learning capability and improved prediction performance. The NN is inspired by the human neural system and how neuron cells

work. It reflects the same fundamentals of the human learning system. The term artificial is allotted to those algorithms representing the smart generation of ML technology; basically, the NN works in two stages: the training and testing stages.

During the training stage, the NN analyses the data and the target to arrive at a notion for linking the input to the correct target (label) (or the expected results). The following steps can be considered during the training stage.

1. The NN is basically constructed of three parts: input, hidden, and output layers. The number of hidden layers varies depending on the requirements of the design.

Figure 24 demonstrates the construction of a single hidden layer NN.

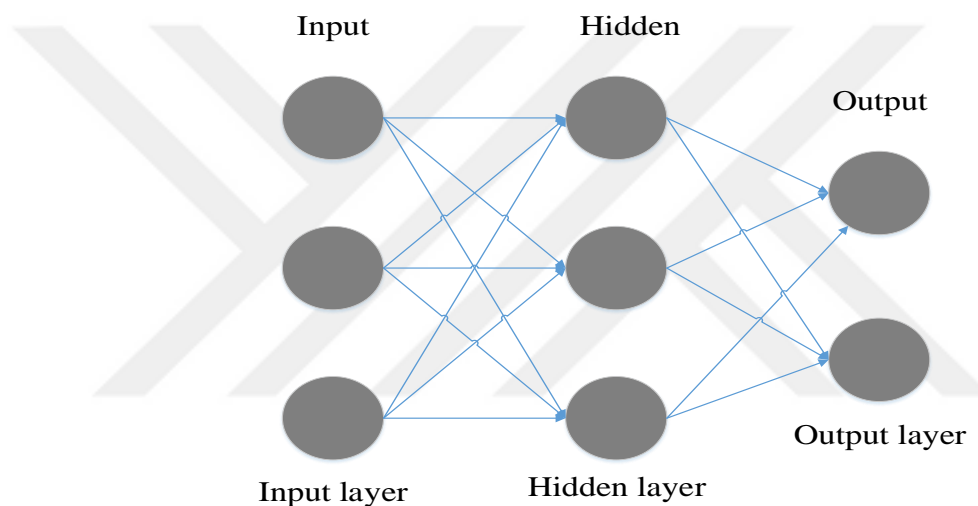


Figure 24. Neural network fundamental structure (Kumar, R. A., 2020)

Each layer in the NN is constructed with several (multiple) nodes. The node number is decided according to the input length. Generally, large node numbers might increase the execution time of the network.

2. Nodes are connected by means of weights, and the weights act like neurons in the human nervous system. These weights are represented by a numerical value, representing the number that scales the input when the input passes from one layer to another.

- At the input layer, $x[n]$ is a single-dimensional array with one column and n rows, which is passed into the NN model. The output of the hidden layer is expressed by $y[n]$ and can be represented in the Equation 10.

$$y[n] = \sum_{n=1}^{n=N} x[n].w_i \quad (10)$$

whereas $z[n]$ can stand for the output of the hidden layer and can be expressed, as in the following Equation 11.

$$z[n] = \sum_{N=1}^{n=N} y[n].w_h \quad (11)$$

Three separate weight factors, the input, hidden and output layer weights, must be presented on the network. However, the final output $m[n]$ of the NN model can be represented in the Equation 12.

$$m[n] = \sum_{n=1}^{n=N} z[n].w_o \quad (12)$$

- Training stage: an NN is first trained on specific data, and the purpose of the training stage is to provide a network with all information related to the data structure and nature; moreover, how data elements are related to each other and how they relate to the target. This process is an essential task in NN operation. The aim accuracy of the NN depends on how accurate the weight allotment is accurate.
- Weight is the main role player in the process of output generation or, in other words, the process of mapping the input to its particular class. Weight allotment is performed to satisfy the following Equation 13.

$$W = \frac{x[n]}{y[n]} \quad (13)$$

The general weight formula is represented by W , the input is $x[n]$ and the output is $y[n]$ after passing through the weight.

Knowing that a weight vector can include thousands of pieces and that the number of weights in the vector is always proportional to the number of dataset elements. Weight is allotted to the NN model using optimisation algorithms, more likely the LM algorithm, which acts as the standard weight generation algorithm in the NN model.

3.6. Model establishment

In order to perform BC diagnosis through biopsy image classification, a model is established using a DL approach and inspired by CNNs. Hence, the model is made to diagnose by predicting the case results by analysing each image. The model is first trained with a large image database that includes many coloured images to supply the NN with complete knowledge about the object colours to deal with various images.

The training of each layer in the model allows it to get familiar with images so that the network can recognise the image according to the accumulated information from each layer. Thus, the model can predict the accurate diagnosis during the training stage accordingly, as shown in Figure 25.

The training process controls how well the model performs. Therefore, a training algorithm integrated with the NN model might monitor the network's performance by evaluating the accuracy of the results. Accordingly, the optimisation algorithm might regenerate the weight in order to minimise the error in the results until reaching the minimal error in the output.

The training process is started by generating random weight coefficients and allotting those numbers to the NN weight values. Hence, after evaluating the model's performance using the fitness function, final weight coefficients can be decided. Therefore, the training process is repeated periodically during the training stages until reaching the weight values that minimise the fitness function. The fitness function, in this case, is the mean square error. The entire procedure of the mentioned training model is depicted in Figure 26.

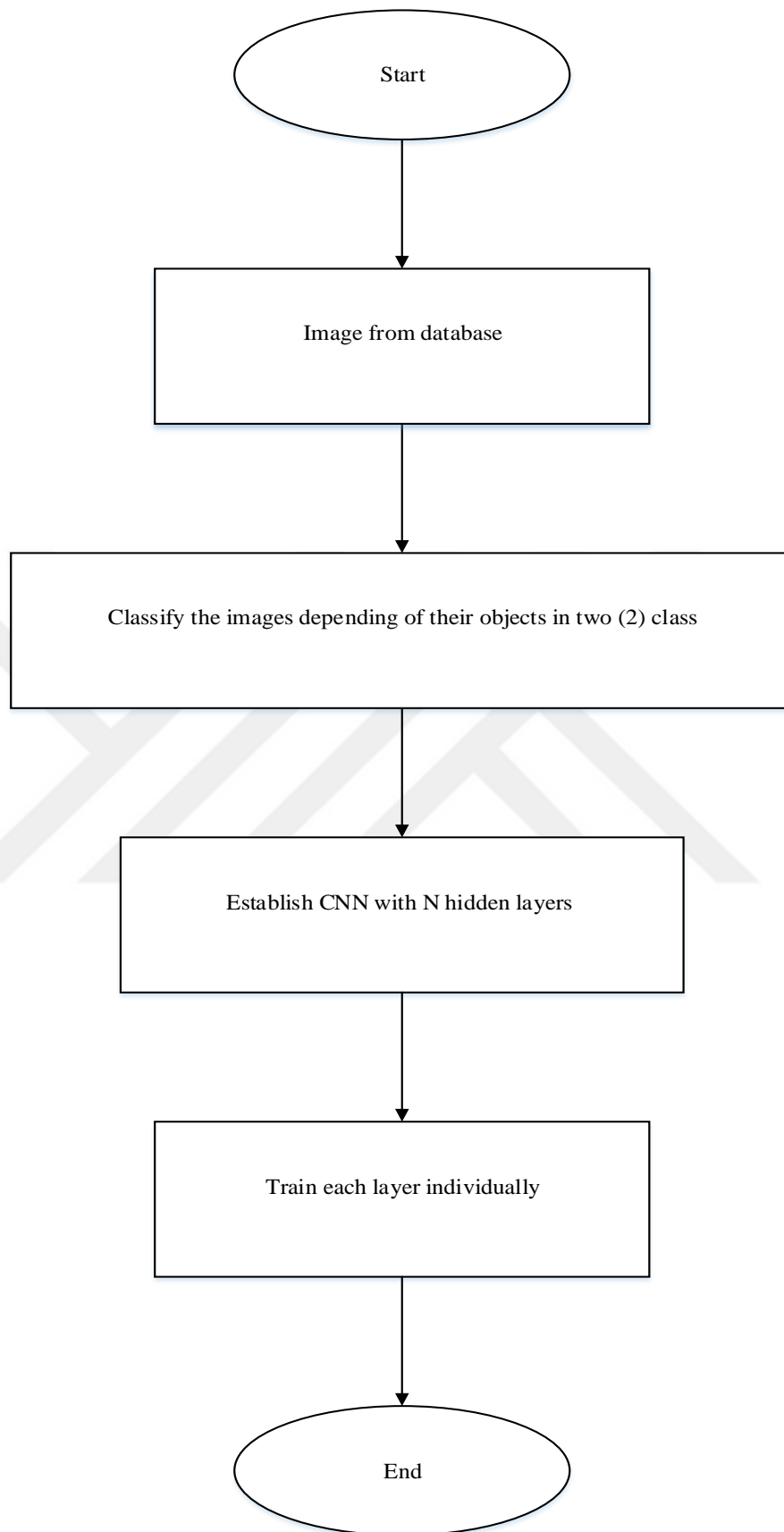


Figure 1. A demonstration of CNN training processes

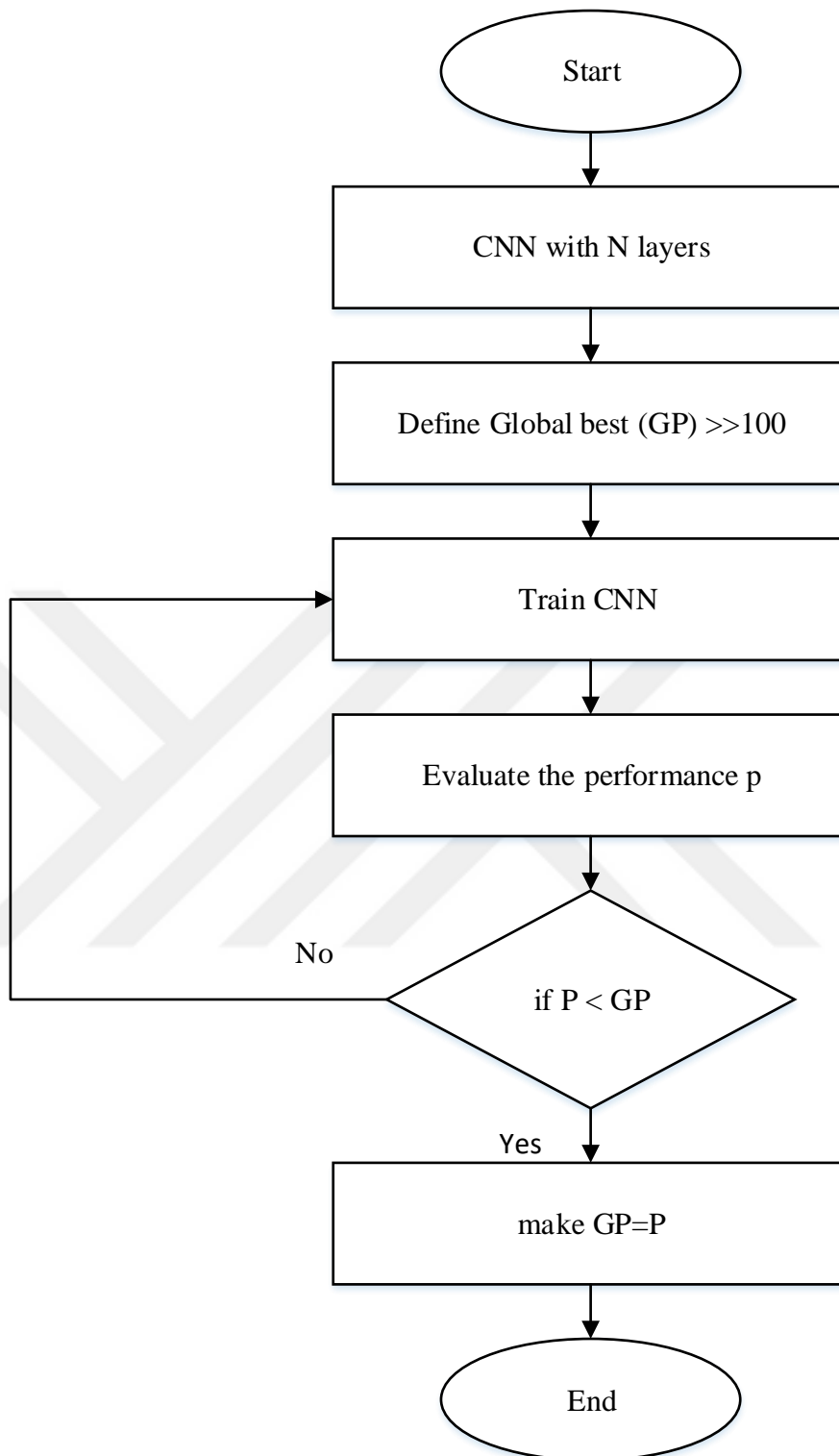


Figure 25. CNN training model procedure

The testing stage will begin by applying test data at the input of the NN model in order to obtain the diagnosis, i.e. zero or one. The NN will study the input data and accordingly will predict image class. However, this model is designed to intake the coloured biopsy images and provide the diagnosis report.

3.7. Performance Metrics

In order to evaluate the performance of the system during the testing, performance metrics were being studied in order to evaluate each algorithm deployed in this thesis. The following performance metrics were used for testing the system performance. If the input image is 10×10 pixels $x[n]$, the following representation can be written in matrix form, as shown in the following matrix 14.

$$x[n, m] = \begin{bmatrix} 1.2 & \cdots & 2.14 \\ \vdots & \ddots & \vdots \\ 5.13 & \cdots & 7.12 \end{bmatrix} \quad (14)$$

and $x[n,m]$ is the target image that is to be predicted by the NN. However, the actual prediction of the NN seems to be $p[n,m]$ which is represented by the following matrix 15.

$$p[n, m] = \begin{bmatrix} 1.2 & \cdots & 2.58 \\ \vdots & \ddots & \vdots \\ 5.74 & \cdots & 7.12 \end{bmatrix} \quad (15)$$

The error matrix, which may be calculated using the formula actual minus real, is directly dependent on the performance matrices. However, performance matrices are directly dependent on the error matrix, which is $E[n,m]$, represented by the actual minus real formula, as shown in the following formula 16 .

$$E[n, m] = x[n, m] - p[n, m] = \begin{bmatrix} 0 & \cdots & 0.44 \\ \vdots & \ddots & \vdots \\ 0.61 & \cdots & 0 \end{bmatrix} \quad (16)$$

Performance measures were evaluated using the following formulas given in the subsections below.

3.7.1. Mean Square Error

Mean square error (MSE) is calculated by applying the equation below, knowing that N represents the total number of errors in the error matrix. That number is recognised by calculating non-zero elements in the error matrix, represented by Equation 17.

$$MSE = \sum_{n=1}^N \left(\frac{E[n]}{N} \right)^2 \quad (17)$$

3.7.2. Mean Absolute Error

Mean absolute error (MAE) is another metric that is calculated; the full form of this metric is the MAE, which is given by Equation 18.

$$MAE = \sum_{n=1}^N |E[n]/N| \quad (18)$$

3.7.3. Root Mean Square Error

The root mean square error (RMSE) can be evaluated using the square root of the MAE function, as shown in Equation 19.

$$RMSE = \sqrt{\sum_{n=1}^N \left(\frac{E[n]}{N} \right)^2} \quad (19)$$

3.7.4. Accuracy

The accuracy is defined as the number of correct predictions divided by the total number of predictions multiplied by 100% and can be represented in Equation 20.

$$Accuracy = \frac{\text{number of correctly predicted pixels}}{\text{total number of pixels}} * 100\% \quad (20)$$

3.7.5. Confusion matrix

A confusion matrix is a special form of table design used in ML and associated disciplines. It aids in demonstrating prediction and recall in a system where the test data values are known.

In order to calculate the confusion matrix, the following points are to be fulfilled:

- The true positive rate (TPR) is the number of correct positive predictions divided by the total number of positives. It is also called recall or sensitivity (SN). The worst possible true positive rate is (0.0), and the best possible true positive rate is (1.0).
- The true negative rate (TNR) is the number of correct negative predictions divided by the total number of negatives. It is also called specificity (SP). The worst possible true negative rate is (0.0), and the best possible true negative rate is (1.0).
- The false positive rate (FPR) is the number of incorrect positive predictions divided by the total number of negatives. It can also be calculated as 1– specificity. The worst possible false positive rate is (1.0), and the best possible false positive rate is (0.0).
- The false negative rate (FNR) is the number of incorrect positive predictions divided by the total number of positives. The worst possible false negative rate is (1.0), and the best possible false negative rate is (0.0).

In order to calculate the confusion matrix, the following calculations are required. The formulas for calculating each of the above values are given in Table 1.

Table 1. Confusion matrix candidates' formula

Name	Formula
True Positive Rate (TPR)	$TPR = TP/(TP+FP)$
True Negative Rate (TNR)	$TNR = TN/(TN+FN)$
False positive rate (FPR)	$FPR = FP/(FP+TP)$
False Negative rate (FNR)	$FNR = FN/(FN+TN)$

CHAPTER FOUR

HARDWARE MODEL

4.1. Overview

Wide advancement in DL technology is motivating a variety of applications for adopting this technology in many fields of life. Thanks to open-source software, existing DL libraries can be modified and improved on a regular basis. As a result, current research focuses on improving a library's performance. Because of open-source software, existing DL libraries may be modified and enhanced on a regular basis to maintain the stability of DL systems used in important human-related sectors like health care. BC photos are collected from a large BC database and processed using a DL model known as a CNN on Mac and Windows platforms using the Python programming language (PyCharm simulator). CNN works with the structure mentioned in the previous chapters in order to perform cancer image classification. Two classes are used, cancerous and non-cancerous. Several preprocessing steps are preceded to the said classification, where each image is downsampled into (50 rows) and (50 columns) dimensions.

Changing the number of iterations, batch size, the number of filters within the layers and the impact of those configuration variations is measured and recorded. In this section, FPGAs were used to implement the proposed CNN, as well as to implement AlexNet and LeNet-5 DL networks. The performance results of networking implemented on FPGAs were compared with the CPU. FPGA is designed to provide flexible programmable environments that meet the needs of designers in various applications.

This chapter discusses image preprocessing and classification using the Lasagne model over the PYNQ FPGA development board. Two smart models were used by changing the layer's structure, and each model was attempted for the same problem where cancerous images are classified. The same performance metrics (i.e. accuracy, MSE and time) were used for evaluating the classification performance for both proposed models. They are referred to in this chapter.

4.2. PYNQ Development Board

Two development boards, PYNQ-Z1 and PYNQ-Z2, are produced by Xilinx for supporting the so-called system on chip (SoC). These development boards benefit from the power of the processor integrated into ZYNQ boards and the programming flexibility of Python in order to perform powerful system designs. These boards are enabled with only programming features without needing to adopt any circuit diagrams or implementation or a system-level programming language, such as hardware description language (HDL). Table 2 represents the features of the PYNQ development board.

API technology allows applications in hardware to interact with each other through software. API technology adaptation on PYNQ development boards makes it possible to configure the logic gates of the FPGA chip using Python codes. Other features are adopted in the PYNQ development board, such as input and output devices, including HDMI, USB, microphone input, Arduino ports, Pmod headers and Raspberry Pi interface. It is equipped using LED as an indications mean and sliding switches as control support means.

Most importantly, this board is integrated with network connectivity using an Ethernet chip with RJ-45 interfacing. The following features in Table 2 are integrated with the PYNQ-Z2 development board. Figure 27 represents the PYNQ development board structure.

Table 2. PYNQ-Z2 development board features

Chip	
○ 650 MHz Arm® Cortex®-A9 Dual-core Processor	
○ Programmable	logic
▪ 13,300 logic slices, each with four 6-input LUTs and 8 flip-flops	
▪ 630KB block RAM	
▪ 220 DSP slices	

<ul style="list-style-type: none"> ▪ On-chip Xilinx analog-to-digital converter (XADC)
<ul style="list-style-type: none"> ○ Programmable from JTAG, Quad-SPI flash, and MicroSD card
<ul style="list-style-type: none"> • Memory and storage
<ul style="list-style-type: none"> ○ 512 MB DDR3 with 16-bit bus @ 1050Mbps
<ul style="list-style-type: none"> ○ 16 MB Quad-SPI Flash with factory programmed 48-bit globally unique EUI-48/64™ compatible identifier
<ul style="list-style-type: none"> ○ MicroSD slot
<ul style="list-style-type: none"> • Power
<ul style="list-style-type: none"> ○ USB or 7V to 15V external power regulator
<ul style="list-style-type: none"> • USB and Ethernet
<ul style="list-style-type: none"> ○ Gigabit Ethernet PHY
<ul style="list-style-type: none"> ○ Micro USB-JTAG programming circuitry
<ul style="list-style-type: none"> ○ Micro USB-UART bridge
<ul style="list-style-type: none"> ○ USB 2.0 OTG PHY (supports host only)
<ul style="list-style-type: none"> • Audio and Video
<ul style="list-style-type: none"> ○ 2x HDMI ports (input and output)
<ul style="list-style-type: none"> ○ 24-bit I²S DAC with 3.5mm TRRS jack
<ul style="list-style-type: none"> ○ Line-in with 3.5mm jack
<ul style="list-style-type: none"> • Switches, Push-Buttons and LEDs
<ul style="list-style-type: none"> ○ 4x push-buttons
<ul style="list-style-type: none"> ○ 2x slide switches

<ul style="list-style-type: none"> ○ 4x LEDs
<ul style="list-style-type: none"> ○ 2x RGB LEDs
<ul style="list-style-type: none"> • Expansion Connectors
<ul style="list-style-type: none"> ○ 2x Pmod ports
<ul style="list-style-type: none"> ▪ 16 Total FPGA I/O (8 pins on Pmod A are shared with Raspberry Pi connector)
<ul style="list-style-type: none"> ○ Arduino Shield compatible connector
<ul style="list-style-type: none"> ▪ 24 Total FPGA I/O
<ul style="list-style-type: none"> ▪ 6 Single-ended 0V to 3.3V Analog inputs to XADC
<ul style="list-style-type: none"> ○ Raspberry Pi connector
<ul style="list-style-type: none"> ▪ 28 Total FPGA I/O (8 pins are shared with Pmod A)
<ul style="list-style-type: none"> • Board dimensions: 87 mm x 140 mm (3.43" x 5.51")

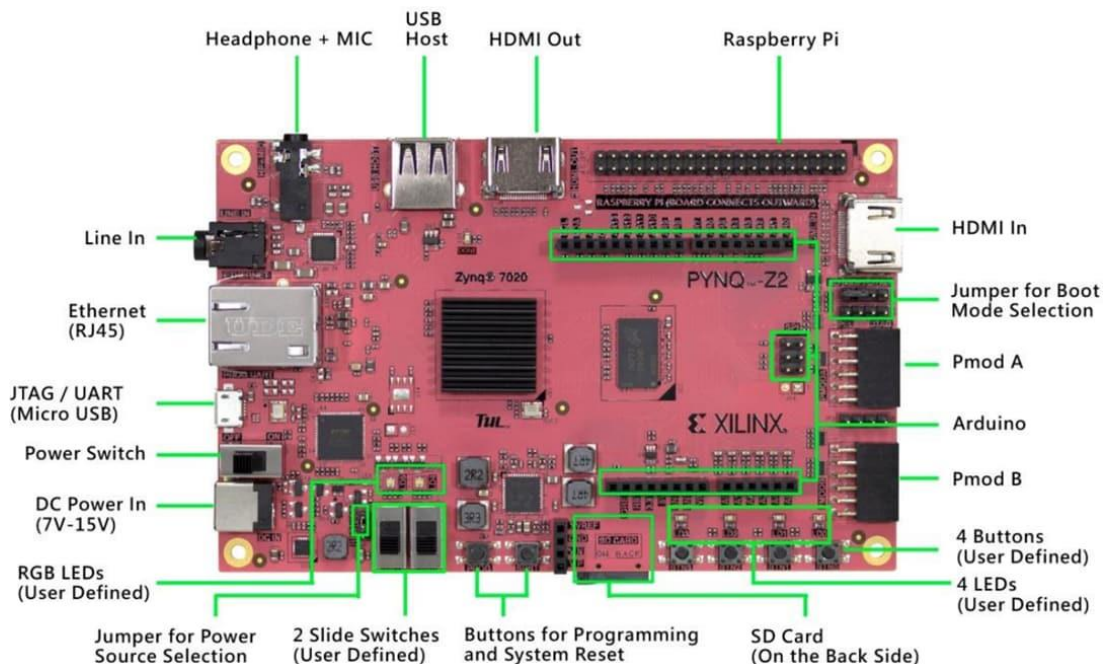


Figure 26. PYNQ-Z2 FPGA development board structure (Daniel, E.,31 January 2020)

4.3. Image Flashing

Technically, the PYNQ-Z2 board is equipped with an SD card slot to upload the same board's operating system (OS). Xilinx is providing all OS(s) officially through their web portal; it can be downloaded and burned onto a sufficient-sized SD card in order to begin the process of PYNQ. There are several third-party software's that can help in burning the OS (termed hereinafter as Image) onto SD cards. balenaEtcher software, as shown in Figure 28, is one of the top software programs used in this project. The SD card must first be prepped by formatting it in order to delete any files that may be stored on it before using the balenaEtcher software.

The image (OS) of PYNQ-Z2 can be downloaded first from the Xilinx web portal and stored on the computer. balenaEtcher can be used to access/clone the location where the image is stored and flash it onto the SD card. This flashing operation might take up to five minutes. The flashing operation is followed by the verification operation, which may last for one minute. At the end of verification, the SD card will be automatically ejected from the computer.

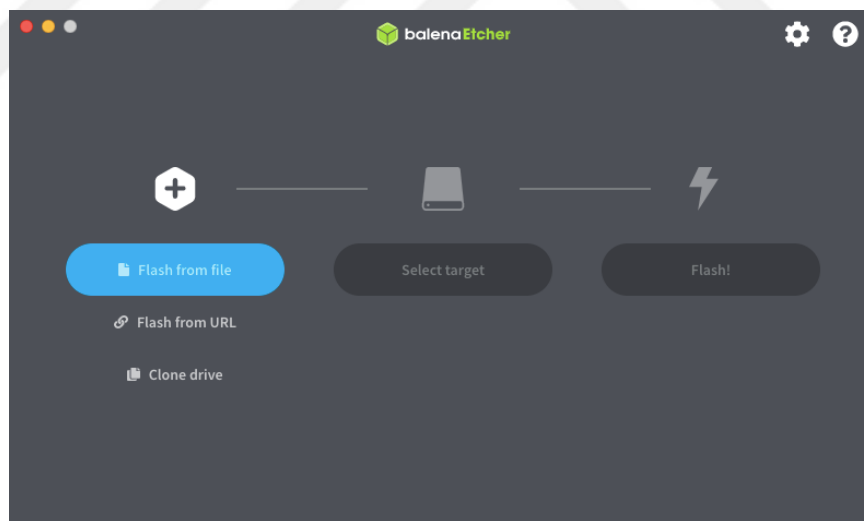


Figure 27. Software to flash the image OS into the PYNQ development board

4.4. Image Configuration

It is noteworthy that the image that was flashed onto the SD card provided Linux-based development environments where Python can be installed. Moreover, the DL libraries needed to accomplish this project will be installed through this platform.

Table 3 represents the list of libraries installed on the PYNQ board. One of the most insistent challenges the developers face using the PYNQ-Z2 board is the installation of DL libraries. In most cases, error messages appear, revealing something like ‘source cannot be found’; in order to tackle most similar errors, the PYNQ-Z2 development board is linked through terminal emulator software, such as Putty.

Table 3. Required libraries to be installed on the PYNQ development board

SN.	Source name	Installation
1	Update	Pip install update Pip install upgrade
2	NumPy	Pip install numpy
3	SCI kit	pip install -U scikit-learn (virtual environment installation is recommended)
4	Cython	Pip install cython
5	SciPy	Pip install scipy
6	pandas	Pip install pandas
7	Pillow	Pip install Pillow
8	Theano	Pip install Theano

4.5. Prediction Models on FPGA

Dealing with the PYNQ-Z2 development board is quite different from the CPU case in regular computers. Many challenges might be encountered in the software installation stage or during the code implementation.

4.6. Deep Learning on FPGA

Two DL models are mainly implemented on FPGA for cancer image diagnosis. Each model consists of a set of layers detailed in the following. The first model, ‘MODEL 1’, is considered to have a light structure, where the prediction process is

expected to take place in lesser time, whereas the second model, ‘MODEL 2’, is structured using larger payloads to enhance the prediction performance.

4.6.1. Model 1

After uploading the database file into the notebook, all photos must be read and placed on a separate array; the same approach as in the simulation stage was followed here unless the image dimensions were resampled to match the DL model (50 x 50). A new dimension was created, equivalent to (1x 50 x 50). The structures and layers listed in Table 4 were used to create a DL model.

Table 4. The first proposed model of FPGA-based breast cancer detection

Layer	Configurations
InputLayer	Shape = (None, 1,50,50)
Conv2DLayer	num_filters = 32 filter_size = (3, 3)
Gain layer	ReLU
MaxPool2DLayer	Size = (2, 2)
Conv2DLayer	num_filters = 64 filter_size = (3, 3)
Gain layer	ReLU
MaxPool2DLayer	pool_size = (2, 2)
DenseLayer	num_units = 128
Gain layer	ReLU
DropoutLayer	Probability = 0.5
DenseLayer	num_units = 3

Gain Layer	Softmax
------------	---------

This model was trained using the Adam algorithm with 20 epochs and a batch size of 20 samples.

Following that, the model was trained for error minimisation based on the detection results shown in Table 5.

Table 5. Epoch-wise results (MSE and time computation) for the first proposed model

Epoch	1/20	Train error:	0.404127	Time	39.87
Epoch	2/20	train error:	0.396219	Time	39.84
Epoch	3/20	train error:	0.393183	Time	39.73
Epoch	4/20	train error:	0.390402	Time	39.81
Epoch	5/20	train error:	0.387714	Time	39.85
Epoch	6/20	train error:	0.385068	Time	39.87
Epoch	7/20	train error:	0.382966	Time	39.76
Epoch	8/20	train error:	0.383447	Time	39.83
Epoch	9/20	train error:	0.377513	Time	39.87
Epoch	10/20	train error:	0.374564	Time	39.9
Epoch	11/20	train error:	0.371899	Time	39.74
Epoch	12/20	train error:	0.369401	Time	39.87
Epoch	13/20	train error:	0.366815	Time	39.88
Epoch	14/20	train error:	0.364147	Time	39.76
Epoch	15/20	train error:	0.361642	Time	39.86

Epoch	16/20	train error:	0.359146	Time	39.87
Epoch	17/20	train error:	0.357675	Time	39.87
Epoch	18/20	train error:	0.354271	Time	39.73
Epoch	19/20	train error:	0.351365	Time	39.87
Epoch	20/20	train error:	0.348738	Time	39.84

4.6.2. Model 2

With the intention of reducing error at the prediction results and minimising the processing time, Model 2 was created with three convolutional layers. The structure of this model is quite similar to that proposed in the CPU (CNN) model. The detailed configurations of Model 2 are given in Table 6.

Table 6. The second proposed model of FPGA environments (Model 2)

Layer	Configurations
InputLayer	Shape = (None, 1,50,50)
Conv2DLayer	num_filters = 5 filter_size = (2, 2)
Gain layer	ReLU
MaxPool2DLayer	Size = (2, 2)
Conv2DLayer	num_filters = 5 filter_size = (2, 2)
Gain layer	ReLU
MaxPool2DLayer	pool_size = (2, 2)
Conv2DLayer	num_filters = 5 filter_size=(2, 2)

Gain layer	ReLU
MaxPool2DLayer	pool_size = (2, 2)
Flatten layer	---
Gain layer	ReLU
DenseLayer	num_units=5
Gain Layer	ReLU
DenseLayer	num_units=5
Gain layer	Softmax

Results obtained from Model 2 are detailed in Table 7. This model was trained using the Adam algorithm with 15 epochs and a batch size of 15 samples.

Table 7. Epoch-wise results (MSE and time computation) for the second proposed CNN model

Epoch	1/15	Train error:	0.868096.	time:	16	seconds
Epoch	2/15	Train error:	0.594069.	time:	16.33	seconds
Epoch	3/15	Train error:	0.558680.	time:	15.87	seconds
Epoch	4/15	Train error:	0.528674.	time:	15.93	seconds
Epoch	5/15	Train error:	0.499050.	time:	16.65	seconds
Epoch	6/15	Train error:	0.453302.	time:	16.01	seconds
Epoch	7/15	Train error:	0.433587.	time:	15.87	seconds
Epoch	8/15	Train error:	0.423658.	time:	16.04	seconds
Epoch	9/15	Train error:	0.417319.	time:	16.29	seconds
Epoch	10/15	Train error:	0.411838.	time:	15.87	seconds

Epoch	11/15	Train error:	0.406916.	time:	15.91	seconds
Epoch	12/15	Train error:	0.402118.	time:	16.38	seconds
Epoch	13/15	Train error:	0.397430.	time:	15.89	seconds
Epoch	14/15	Train error:	0.393335.	time:	15.88	seconds
Epoch	15/15	Train error:	0.389542.	time:	16.07	seconds

4.6.3. Implementation LeNet-5 - ARM Linux OS

Theano was the framework used to implement the LeNet-5 and AlexNet networks. Apt-get was used to install Theano on PYNQ with a line of command (pip install Lasagne = 0.1). To further improve the CNN deployment's user interface, Lasagne was added on top of Theano. Lasagne is a lightweight Theano library for building and training NNs, with a more concise and understandable code presentation than Theano. A LeNet-5 CNN was implemented using the Lasagne syntax, as shown in Table 8.

In conclusion, CNNs can be declared using simple, legible Python codes, resulting in user design interfaces. Layers may easily be used with Lasagne because it supports user-designed customisable layers. As a high-level framework interface, Lasagne customised CNN layers. These layers can be created the same way as Theano's built-in layer functions, giving them similar speed. The framework's functionality was tested using a testbench project. It was made up of a Python notebook script and a modified Lasagne CNN layer with API.

Table 8. Lasagne LeNet Configuration

Layer	Configurations
InputLayer	Shape = (None, 1,32,32)
Conv2DLayer	num_filters = 6 filter_size = (5, 5)

Gain layer	tanh
AveragePooling2dLayer	Size = (2, 2)
Conv2DLayer	num_filters = 16 filter_size = (5, 5)
Gain layer	tanh
AveragePooling2dLayer	num_filters = 120 filter_size = (2, 2)
Conv2DLayer	num_filters = 120 filter_size = (5, 5)
Gain layer	tanh
DenseLayer	num_units = 2
Gain Layer	ReLU
DenseLayer	num_units = 2
Gain layer	Softmax

Results obtained from LeNet-5 are detailed in Table 9. This model is trained using the Adam algorithm with 20 epochs and a batch size of 20 samples.

Table 9. Epoch-wise results (MSE and time computation) for the LeNet-5 model

Epoch	1/20	train error:	0.731353	Time	53.88	seconds
Epoch	2/20	train error:	0.701256	Time	53.87	seconds
Epoch	3/20	train error:	0.586131	Time	53.85	seconds
Epoch	4/20	train error:	0.524314	Time	53.88	seconds
Epoch	5/20	train error:	0.510745	Time	53.78	seconds

Epoch	6/20	train error:	0.475931	Time	53.85	seconds
Epoch	7/20	train error:	0.451468	Time	53.89	seconds
Epoch	8/20	train error:	0.437517	Time	53.77	seconds
Epoch	9/20	train error:	0.425106	Time	53.87	seconds
Epoch	10/20	train error:	0.416713	Time	53.75	seconds
Epoch	11/20	train error:	0.413156	Time	53.88	seconds
Epoch	12/20	train error:	0.401532	Time	53.87	seconds
Epoch	13/20	train error:	0.401135	Time	53.83	seconds
Epoch	14/20	train error:	0.385213	Time	53.81	seconds
Epoch	15/20	train error:	0.374721	Time	53.78	seconds
Epoch	16/20	train error:	0.364182	Time	53.87	seconds
Epoch	17/20	train error:	0.356233	Time	53.88	seconds
Epoch	18/20	train error:	0.354361	Time	53.78	seconds
Epoch	19/20	train error:	0.341521	Time	53.83	seconds
Epoch	20/20	train error:	0.33583	Time	53.77	seconds

4.6.4. Implementation AlexNet – ARM Linux OS

As indicated in Table 10, an AlexNet CNN was created using Lasagne syntax. Finally, CNNs can be declared with simple, readable Python code, resulting in user-friendly interfaces. Lasagne makes it simple to use layers because it allows users to create their own unique layers. Lasagne adapts CNN layers as a framework for a high-level interface. These layers can be constructed the same way as Theano's built-in layer functions, which means they will be just as quick. A testbench project was created to test the framework's functioning. It consisted of a Python notebook script and a modified Lasagne CNN layer with API.

Table 10. Lasagne AlexNet Configuration

Layer	Configurations
InputLayer	shape=(None, 3, 227, 227)
Conv2DLayer	num_filters = 96 filter_size = (11, 11)
Gain layer	relu
MaxPool2DLayer	Size = (2, 2)
Conv2DLayer	num_filters=256 filter_size = (11, 11)
Gain layer	relu
MaxPool2DLayer	Size = (2, 2)
Conv2DLayer	num_filters = 384 filter_size = (3, 3)
Gain layer	relu
Conv2DLayer	num_filters = 384 filter_size = (3, 3)
Gain layer	relu
Conv2DLayer	num_filters = 256 filter_size = (3, 3)
Gain layer	relu
MaxPool2DLayer	Size = (2, 2)
DenseLayer	num_units = 2
Gain Layer	ReLU
DenseLayer	num_units = 2

Gain layer	Softmax
------------	---------

Results obtained from AlexNet are detailed in Table 11. This model was trained using the Adam algorithm with 20 epochs and a batch size of 20 samples.

Table 11. Epoch-wise results (MSE and time computation) for the AlexNet model

Epoch	1/20	train error:	0.872133	Time	240.33	seconds
Epoch	2/20	train error:	0.832314	Time	240.92	seconds
Epoch	3/20	train error:	0.721052	Time	240.21	seconds
Epoch	4/20	train error:	0.715296	Time	240.01	seconds
Epoch	5/20	train error:	0.635122	Time	240.35	seconds
Epoch	6/20	train error:	0.622754	Time	240.87	seconds
Epoch	7/20	train error:	0.574103	Time	240.89	seconds
Epoch	8/20	train error:	0.564352	Time	240.31	seconds
Epoch	9/20	train error:	0.551213	Time	240.89	seconds
Epoch	10/20	train error:	0.536389	Time	240.88	seconds
Epoch	11/20	train error:	0.526521	Time	240.32	seconds
Epoch	12/20	train error:	0.513386	Time	240.08	seconds
Epoch	13/20	train error:	0.485241	Time	240.87	seconds
Epoch	14/20	train error:	0.463533	Time	240.81	seconds
Epoch	15/20	train error:	0.435174	Time	240.91	seconds
Epoch	16/20	train error:	0.410176	Time	240.87	seconds
Epoch	17/20	train error:	0.384265	Time	240.05	seconds

Epoch	18/20	train error:	0.383412	Time	240.71	seconds
Epoch	19/20	train error:	0.374821	Time	240.01	seconds
Epoch	20/20	train error:	0.371483	Time	240.38	seconds

4.6.5. Implementation Prototypes: Test Phase

Following the training phase, the proposed model was tested using a set of photos in the testing phase. The accuracy of these photos was computed for each of the classes, and the overall accuracy was calculated. The most interesting part of the proposed CNN algorithm was that the proposed CNN structure extracted the features of an image locally, which means that the network learned specific patterns within the image and could recognise them anywhere in the image. The steps were repeated until the image was scanned. During the testing phase, 20% of the database photos were used to evaluate CNN's performance, and the performance was then quantified by the number of images accurately predicted. Applications that perform can be created to demonstrate the possibilities of my framework by showing the visual result that the user will get. By saving the training weights in layers after the network result reached a high accuracy, the network performance could be tested by entering an image into the network, and the expected result was tested by classifying the image as cancerous or non-cancerous, then the accuracy of this classification appeared, as shown in Figure 29.

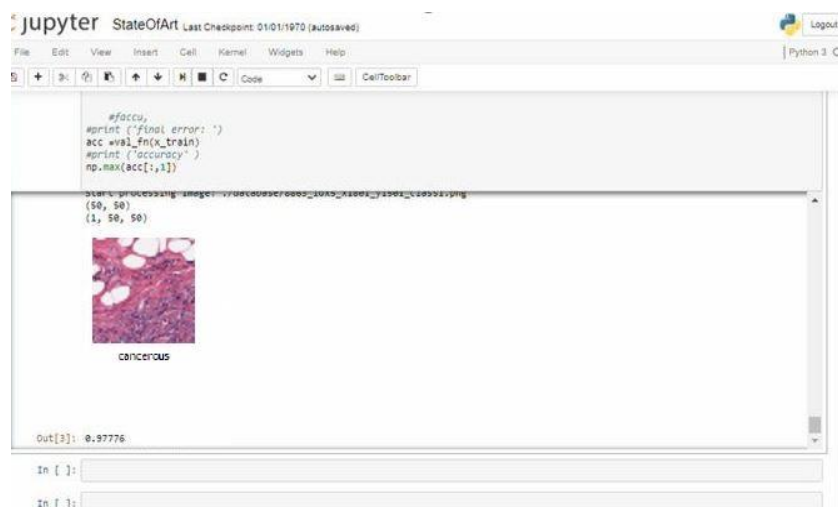


Figure 28. Sample application for breast cancer image

CHAPTER FIVE

RESULTS AND DISCUSSION

5.1. Overview

A CNN is mainly used with different hyperparameters (configurations) to predict cancer in the database. A total of 2,215 coloured biopsy images were used to train the FPGA-based DL models. The process was performed in the SKLearn model over Python environments. Images were first resized to new dimensions of 50 x 50 pixels. After that, the following steps were applied to the images before they were fed into CNN:

- a> Images resized to new dimensions, i.e. 50 x 50 pixels, in order to reduce the payload on the proposed classifier by removing the unwanted parts from the images.
- b> New reshaping was performed to convert the images (each) into three-dimensional arrays properly to supply them into CNN. The new shape of each image became 50 x 50 x 1.
- c> Image pixel normalisation, so that the value of each pixel would be a crack of one, and the peak value of the images' pixels would have one (e.g. 1). This process was for variance reduction in order to improve the classification results. The steps performed on the images before the training are demonstrated in Figure 30.

The model training was configured by employing only 20% of the images used for testing, while the other 80% are used for training. The CNN structure used in the configurations is listed in Table 12, and the training coefficients are shown in Table 13.

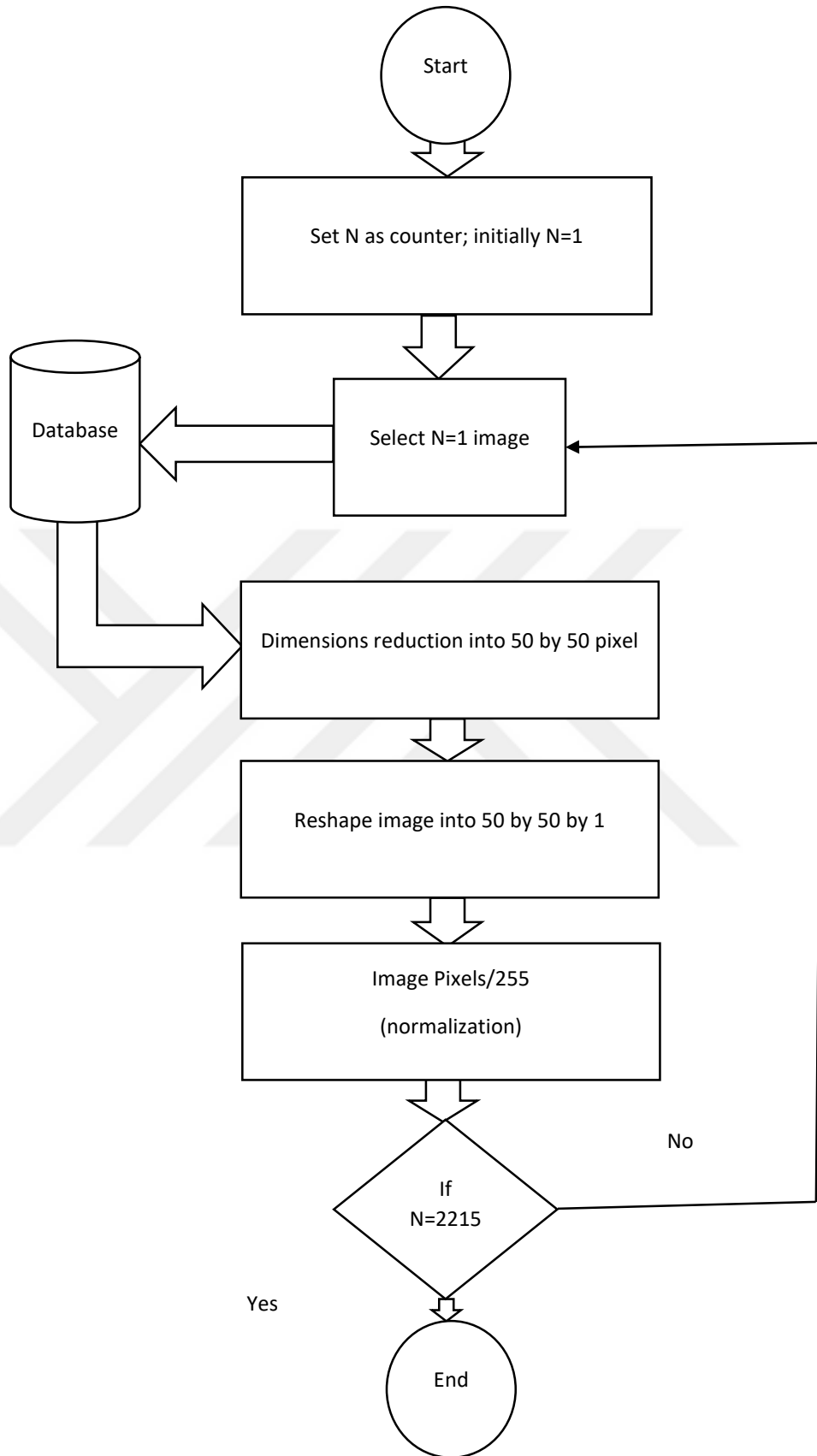


Figure 29. Preprocessing model flow diagram

Table 12. The configuration of the proposed CNN structure used in CPU-based learning

Layer	Information
Sequential CNN	Main model type
Conv2D	First layer with 32 filters and (3,3) kernel size and “linear” activation
LeakyReLU	Second layer with alpha transfer function
MaxPooling2D	Third layer with (2, 2) kernel size
Conv2D	Fourth layer with 64 filters and (3,3) kernel size and “linear” activation
LeakyReLU	Fifth layer with alpha transfer function
MaxPooling2D	Sixth layer with a pool size of (2, 2)
Conv2D	Seventh layer with 128 filters, (3,3) kernel size and “linear” activation
LeakyReLU	Eighth layer with alpha transfer function
MaxPooling2D	Ninth layer with a pool size of (2, 2)
Flatten	Tenth layer
Dense	Eleventh layer with 128 filters
LeakyReLU	Twelfth layer with alpha transfer function
Dense	Last layer with three filters

Table 13. CNN model training coefficients

Parameter	Value
Training method	Adam
Batch size	64
Epochs	10
Verbose	1

5.2. Results and Discussion of Proposed CNNs

Different iterations were performed for the CNN model to determine the optimum configurations. That is, the varying kernel size, layers and filter numbers of the main prementioned model were conducted. The results of each model, along with their description, are mentioned in Tables 14 to 17.

Table 14. Results of the first and second iterations of the CNN model

Case	Con2D (1) layer parameter				Pooling layer parameter				Accuracy %
	Filter size	No of filter	Stride	Padding	Filter size	Type	Stride	Padding	
1	3*3	32	1	1	2*2	MAX	1	1	88.55
	5*5	32	1	1	2*2	MAX	1	1	86.21
	7*7	32	1	1	2*2	MAX	1	1	86.04
	9*9	32	1	1	2*2	MAX	1	1	85.11
	11*11	32	1	1	2*2	MAX	1	1	81.03
	13*13	32	1	1	2*2	MAX	1	1	81.00
Case	Con2D (1) layer parameter				Pooling layer parameter				Accuracy

	Con2D (1) layer parameter								Accuracy %
	Filter size	No of filter	Stride	Padding	Filter size	Type	Stride	Padding	
2	3*3	2	1	1	2*2	MAX	1	1	79.13
	3*3	4	1	1	2*2	MAX	1	1	83.33
	3*3	8	1	1	2*2	MAX	1	1	85.76
	3*3	16	1	1	2*2	MAX	1	1	80.41
	3*3	20	1	1	2*2	MAX	1	1	82.37
	3*3	32	1	1	2*2	MAX	1	1	88.55

Table 14 illustrates the accuracy results of the proposed CNN, such as when the number of filters of each layer was set to (32), and the filter size was iterated from 3 x 3 through 13 x 13 with a step size of 2, the maximum prediction accuracy was reported while the filters were (32) and each filter size was 3 x 3. It was observed that the more the size of the filters degrades the prediction accuracy. In other words, when the number of filters was iterated among (2, 4, 8, 16, 20 and 32), the prediction accuracy improved with a higher number of filters when the filter size was fixed at 3 x 3. That was manifested when 32 filters were used with a 3 x 3 filter size.

Table 15. Results of the third and fourth iterations of the CNN model

Case	Con2D (1) layer parameter				Pooling layer parameter				Accuracy %
	Filter size	No of filter	Stride	Padding	Filter size	Type	Stride	Padding	
3	3*3	32	1	1	2*2	MAX	1	1	88.55
	3*3	32	1	1	4*4	MAX	1	1	85.25
	3*3	32	1	1	6*6	MAX	1	1	81.01

	3*3	32	1	1	8*8	MAX	1	1	80.48
	3*3	32	1	1	10*10	MAX	1	1	84
	3*3	32	1	1	12*12	MAX	1	1	84.21
Case	Con2D (2) layer parameter				Pooling2 layer parameter				Accuracy %
	Filter size	No of filter	Stride	Padding	Filter size	Type	Stride	Padding	
	3*3	64	1	1	2*2	MAX	1	1	90.54
	5*5	64	1	1	2*2	MAX	1	1	89.37
	7*7	64	1	1	2*2	MAX	1	1	89.21
4	9*9	64	1	1	2*2	MAX	1	1	85.77
	11*11	64	1	1	2*2	MAX	1	1	84.32
	13*13	64	1	1	2*2	MAX	1	1	80.11

Table 15 illustrates the occurrence when two experiments were conducted. In the first case, the number of filters of the convolutional layer, as well as the filters' size, was fixed to 32 and 33, consequently iterating the filters' size in the pooling layer. In this case, the best accuracy was obtained when the filter size of the pooling layer was minimum, i.e., 2 x 2; that led to the prediction accuracy of 88.55%. The other case involved iterating the filters' size of the convolutional layer and uplifting the number of filters in it to 64; however, the filter size of the pooling layer was fixed to 2 x 2, which enhances the prediction accuracy to 90.54%.

Table 16. Results of the fifth and sixth iterations of the CNN model

Case	Con2D (2) layer parameter				Pooling layer parameter				Accuracy %
	Filter size	No of filter	Stride	Padding	Filter size	Type	Stride	Padding	
5	3*3	16	1	1	2*2	MAX	1	1	80.16
	3*3	26	1	1	2*2	MAX	1	1	79.33
	3*3	32	1	1	2*2	MAX	1	1	84
	3*3	44	1	1	2*2	MAX	1	1	86.11
	3*3	54	1	1	2*2	MAX	1	1	85.21
	3*3	64	1	1	2*2	MAX	1	1	90.54
Case	Con2D (2) layer parameter				Pooling2 layer parameter				Accuracy %
	Filter size	No of filter	Stride	Padding	Filter size	Type	Stride	Padding	
6	3*3	64	1	1	2*2	MAX	1	1	90.54
	3*3	64	1	1	4*4	MAX	1	1	86.32
	3*3	64	1	1	6*6	MAX	1	1	85.11
	3*3	64	1	1	8*8	MAX	1	1	87.57
	3*3	64	1	1	10*10	MAX	1	1	80.11

3*3	64	1	1	12*12	MAX	1	1	80.00
-----	----	---	---	-------	-----	---	---	-------

In Table 16, the convolutional and pooling layer filter sizes were set to 3 x 3 and 2 x 2, respectively, and the number of convolutional layer filters repeated. The findings of this experiment have shown that 64 filters with a size of 3 x 3 at the convolutional layer and a pooling layer with a 2 x 2 filter size preserved the best prediction accuracy, i.e. 90.54%.

The other event, shown in the same Table, included iterating the filters' size of the pooling layer while preserving the similar configuration, leading to the same prediction accuracy of 90.54%.

Table 17. Results of the seventh and eighth iterations of the CNN model

Case	Con2D (3) layer parameter				Pooling layer parameter				Accuracy %
	Filter size	No of filter	Stride	Padding	Filter size	Type	Stride	Padding	
7	3*3	128	1	1	2*2	MAX	1	1	95.12
	5*5	128	1	1	2*2	MAX	1	1	92.43
	7*7	128	1	1	2*2	MAX	1	1	92.21
	9*9	128	1	1	2*2	MAX	1	1	89.41
	11*11	128	1	1	2*2	MAX	1	1	90
	13*13	128	1	1	2*2	MAX	1	1	91.88

Case	Con2D (3) layer parameter				Pooling2 layer parameter				Accuracy
	Filter size	No of filter	Stride	Padding	Filter size	Type	Stride	Padding	%
8	3*3	32	1	1	2*2	MAX	1	1	79.54
	3*3	42	1	1	2*2	MAX	1	1	86.32
	3*3	62	1	1	2*2	MAX	1	1	85.11
	3*3	82	1	1	2*2	MAX	1	1	87.57
	3*3	100	1	1	2*2	MAX	1	1	91.01
	3*3	128	1	1	2*2	MAX	1	1	96.87

Table 17 shows how to set the number of filters in the convolutional layer to 128 and the pooling layer filter size to 2 x 2 while iterating the convolutional layer filter size. The accuracy of the prediction degraded as the filter size increased; the best accuracy was obtained with the smallest filter size, 3 x 3. The maximum accuracy that was reported from the aforementioned configuration was 96.87%.

The other scenario demonstrated in the same Table is when the number of convolutional layer filters increased to 128, and the filter size of both the convolutional and pooling layers was fixed to 3 x 3 and 2 x 2. Consequently, the model with these configurations was outperformed by a scoring accuracy of 96.87%.

Observation: It may be deduced that having a larger number of filters in the convolutional layer can enhance accuracy unless the filter size is too enormous. The number of filters in the convolutional layer directly influences prediction accuracy. The prediction accuracy could be improved by increasing the number of filters. On the

other hand, using a large filter size has a detrimental influence on performance; the same principle applies to increasing the pooling layer's filter size.

The proposed CNN confusion matrix shows that the number of negative cases predicted as positive was 1.7% of the total, while the number of positive cases predicted as negative was 1.4%. Figure 31 shows the proposed CNN confusion matrix.

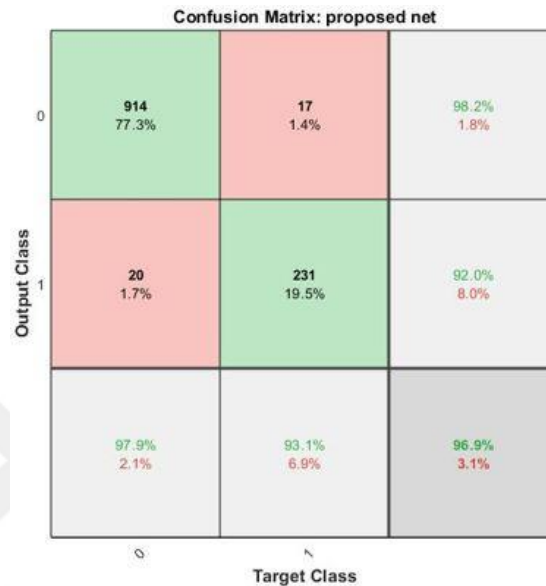


Figure 30. Confusion matrix of proposed CNN

5.3. Pre-trained Models

DL pre-trained models, such as VGG-16, AlexNet, LeNet, ResNet-18 and ShuffleNet, were used in order to predict BC in the given database. The analysis result of the mentioned models is shown in Tables 18 to 21.

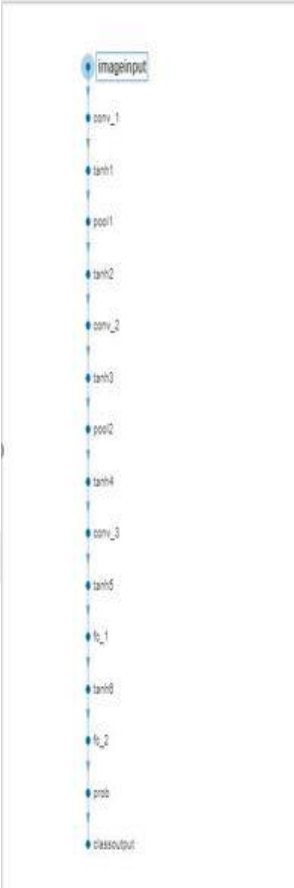
Table 18. VGG-16 neural network structure.

ANALYSIS RESULT					
	Name	Type	Activations	Learnables	Total Learnab...
1	input 224x224x3 images with 'zerocenter' normalization	Image Input	224x224x3	-	0
2	conv1_1 64 3x3x3 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	224x224x64	Weights 3x3x3x64 Bias 1x1x64	1792
3	relu1_1 ReLU	ReLU	224x224x64	-	0
4	conv1_2 64 3x3x3 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	224x224x64	Weights 3x3x64x64 Bias 1x1x64	36928
5	relu1_2 ReLU	ReLU	224x224x64	-	0
6	pool1 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	112x112x64	-	0
7	conv2_1 128 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	112x112x128	Weights 3x3x64x128 Bias 1x1x128	73856
8	relu2_1 ReLU	ReLU	112x112x128	-	0
9	conv2_2 128 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	112x112x128	Weights 3x3x128x128 Bias 1x1x128	147584
10	relu2_2 ReLU	ReLU	112x112x128	-	0
11	pool2 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	56x56x128	-	0
12	conv3_1 256 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	56x56x256	Weights 3x3x128x256 Bias 1x1x256	295168
13	relu3_1 ReLU	ReLU	56x56x256	-	0
14	conv3_2 256 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	56x56x256	Weights 3x3x256x256 Bias 1x1x256	590080
15	relu3_2 ReLU	ReLU	56x56x256	-	0
16	conv3_3 256 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	56x56x256	Weights 3x3x256x256 Bias 1x1x256	590080
17	relu3_3 ReLU	ReLU	56x56x256	-	0
18	pool3 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	28x28x256	-	0
19	conv4_1 512 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28x28x512	Weights 3x3x256x512 Bias 1x1x512	1180160
20	relu4_1 ReLU	ReLU	28x28x512	-	0
21	conv4_2 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28x28x512	Weights 3x3x512x512 Bias 1x1x512	2359808
22	relu4_2 ReLU	ReLU	28x28x512	-	0
23	conv4_3 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28x28x512	Weights 3x3x512x512 Bias 1x1x512	2359808
24	relu4_3 ReLU	ReLU	28x28x512	-	0
25	pool4 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	14x14x512	-	0
26	conv5_1 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14x14x512	Weights 3x3x512x512 Bias 1x1x512	2359808
27	relu5_1 ReLU	ReLU	14x14x512	-	0
28	conv5_2 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14x14x512	Weights 3x3x512x512 Bias 1x1x512	2359808
29	relu5_2 ReLU	ReLU	14x14x512	-	0
30	conv5_3 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14x14x512	Weights 3x3x512x512 Bias 1x1x512	2359808
31	relu5_3 ReLU	ReLU	14x14x512	-	0
32	pool5 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	7x7x512	-	0
33	fc6 4096 fully connected layer	Fully Connected	1x1x4096	Weights 4096x25088 Bias 4096x1	102764544
34	relu6 ReLU	ReLU	1x1x4096	-	0
35	drop6 50% dropout	Dropout	1x1x4096	-	0
36	fc7 4096 fully connected layer	Fully Connected	1x1x4096	Weights 4096x4096 Bias 4096x1	16781312
37	relu7 ReLU	ReLU	1x1x4096	-	0
38	drop7 50% dropout	Dropout	1x1x4096	-	0
39	fc8 1000 fully connected layer	Fully Connected	1x1x1000	Weights 1000x4096 Bias 1000x1	4097000
40	prob softmax	Softmax	1x1x1000	-	0
41	output crossentropyex with 'lench' and 999 other classes	Classification Output	-	-	0

Table 19. AlexNet neural network structure.

ANALYSIS RESULT				
	Name	Type	Activations	Learnables
1	data 227x227x3 images with 'zerocenter' normalization	Image Input	227x227x3	-
2	conv1 96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]	Convolution	55x55x96	Weights 11x11x3x96 Bias 1x1x96
3	relu1 ReLU	ReLU	55x55x96	-
4	norm1 cross channel normalization with 5 channels per element	Cross Channel Nor...	55x55x96	-
5	pool1 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	27x27x96	-
6	conv2 2 groups of 128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]	Grouped Convolution	27x27x256	Weigh... 5x5x48x128... Bias 1x1x128x2
7	relu2 ReLU	ReLU	27x27x256	-
8	norm2 cross channel normalization with 5 channels per element	Cross Channel Nor...	27x27x256	-
9	pool2 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	13x13x256	-
10	conv3 384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	13x13x384	Weights 3x3x256x384 Bias 1x1x384
11	relu3 ReLU	ReLU	13x13x384	-
12	conv4 2 groups of 192 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Grouped Convolution	13x13x384	Weigh... 3x3x192x192... Bias 1x1x192x2
13	relu4 ReLU	ReLU	13x13x384	-
14	conv5 2 groups of 128 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Grouped Convolution	13x13x256	Weigh... 3x3x192x128... Bias 1x1x128x2
15	relu5 ReLU	ReLU	13x13x256	-
16	pool5 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	6x6x256	-
17	fc6 4096 fully connected layer	Fully Connected	1x1x4096	Weights 4096x9216 Bias 4096x1
18	relu6 ReLU	ReLU	1x1x4096	-
19	drop6 50% dropout	Dropout	1x1x4096	-
20	fc7 4096 fully connected layer	Fully Connected	1x1x4096	Weights 4096x4096 Bias 4096x1
21	relu7 ReLU	ReLU	1x1x4096	-
22	drop7 50% dropout	Dropout	1x1x4096	-
23	fc8 1000 fully connected layer	Fully Connected	1x1x1000	Weights 1000x4096 Bias 1000x1
24	prob softmax	Softmax	1x1x1000	-
25	output crossentropyx with 'tench' and 999 other classes	Classification Output	-	-

Table 20. LeNet neural network structure.



ANALYSIS RESULT				
	Name	Type	Activations	Learnables
1	imageinput 32x32x1 images with 'zerocenter' normalization	Image Input	32x32x1	-
2	conv_1 6 5x5x1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28x28x6	Weights: 5x5x1x6 Bias: 1x1x6
3	tanh1 Hyperbolic tangent	Tanh	28x28x6	-
4	pool1 2x2 average pooling with stride [2 2] and padding [0 0 0 0]	Average Pooling	14x14x6	-
5	tanh2 Hyperbolic tangent	Tanh	14x14x6	-
6	conv_2 16 5x5x1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	10x10x16	Weights: 5x5x6x16 Bias: 1x1x16
7	tanh3 Hyperbolic tangent	Tanh	10x10x16	-
8	pool2 2x2 average pooling with stride [2 2] and padding [0 0 0 0]	Average Pooling	5x5x16	-
9	tanh4 Hyperbolic tangent	Tanh	5x5x16	-
10	conv_3 120 5x5x16 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	1x1x120	Weights: 5x5x16x120 Bias: 1x1x120
11	tanh5 Hyperbolic tangent	Tanh	1x1x120	-
12	fc_1 2 fully connected layer	Fully Connected	1x1x2	Weights: 2x120 Bias: 2x1
13	tanh6 Hyperbolic tangent	Tanh	1x1x2	-
14	fc_2 2 fully connected layer	Fully Connected	1x1x2	Weights: 2x2 Bias: 2x1
15	prob softmax	Softmax	1x1x2	-
16	classoutput crossentropy with classes '0' and '1'	Classification Output	-	-

Table 21. Proposed network structure.

1	imageinput 32x32x1 images with 'zerocenter' normalization	Image Input	32x32x1
2	conv_1 6 5x5x1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28x28x6
3	tanh1 Hyperbolic tangent	Tanh	28x28x6
4	pool1 2x2 average pooling with stride [2 2] and padding [0 0 0 0]	Average Pooling	14x14x6
5	tanh2 Hyperbolic tangent	Tanh	14x14x6
6	conv_2 16 5x5x0 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	10x10x16
7	tanh3 Hyperbolic tangent	Tanh	10x10x16
8	pool2 2x2 average pooling with stride [2 2] and padding [0 0 0 0]	Average Pooling	5x5x16
9	tanh4 Hyperbolic tangent	Tanh	5x5x16
10	conv_3 120 5x5x16 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	1x1x120
11	tanh5 Hyperbolic tangent	Tanh	1x1x120
12	fc_1 2 fully connected layer	Fully Connected	1x1x2
13	tanh6 Hyperbolic tangent	Tanh	1x1x2
14	fc_2 2 fully connected layer	Fully Connected	1x1x2
15	prob softmax	Softmax	1x1x2
16	classoutput crossentropyex with classes '0' and '1'	Classification Output	-

Table 22 shows the classification performance report for pre-trained deep learning networks. According to that, it can be understood that **ResNet-18** was reported as the best pre-trained algorithm that outperformed for prediction accuracy; 99% of prediction accuracy could be obtained from the **ResNet-18** model. The ResNet-18 sensitivity measure was calculated, and it was found to be as high as 99%, implying that all true (positive cancer cases) were accurately predicted in the model. Similarly, a specificity measure of 98% was obtained, meaning that only 98% of the negative cancer cases were successfully detected. In order to understand how well the classifier was predicting the positive cancer cases with respect to the total number of cases

(positives and negatives), precision was calculated; 99% of the precision was reported for the **ResNet-18** model. The percentage of positive examples calculated with respect to actually anticipated positives and mistakenly predicted positives might provide a different view of the model's performance; this is known as recall, and it was reported with a 99% accuracy. The F1 score represents the inter-relationship between both precision and recall. The **ResNet-18** produced 99% of recall performance measures. The geometric mean was calculated to understand how well the classifier performance could measure the classification for both positive and negative cases in terms of the majority and minority; it was reported as 99%. The final misclassification rate was calculated, which was only 1% for the **ResNet-18**; this measure was the opposite of accuracy, 99%. The same metrics were calculated for the VGG-16, ShuffleNet, AlexNet and LenNet; those metrics for the mentioned algorithms were less than those of the AlexNet.

Table 22. Pre-trained deep learning classification performance measures

Classifier Report						
Measure	Proposed Network	AlexNet	VGG-16	LeNet-5	ResNet-18	ShuffleNet
Accuracy	97%	93%	91%	85%	99%	98%
Sensitivity	98%	97%	95%	94%	99%	98%
Specificity	93%	80%	79%	52%	98%	98%
Precision	98%	95%	94%	88%	99%	99%
Recall	98%	97%	95%	94%	99%	98%
F1 Score	98	96	95	91	99	99
G Mean	95%	88%	87%	70%	98%	98%
Misclassification Rate	3%	7%	9%	15%	1%	2%

Figure 32 represents a graphic of the pre-trained DL classification performance metrics for the tested DNNs, represented by the parameters of the classifier.

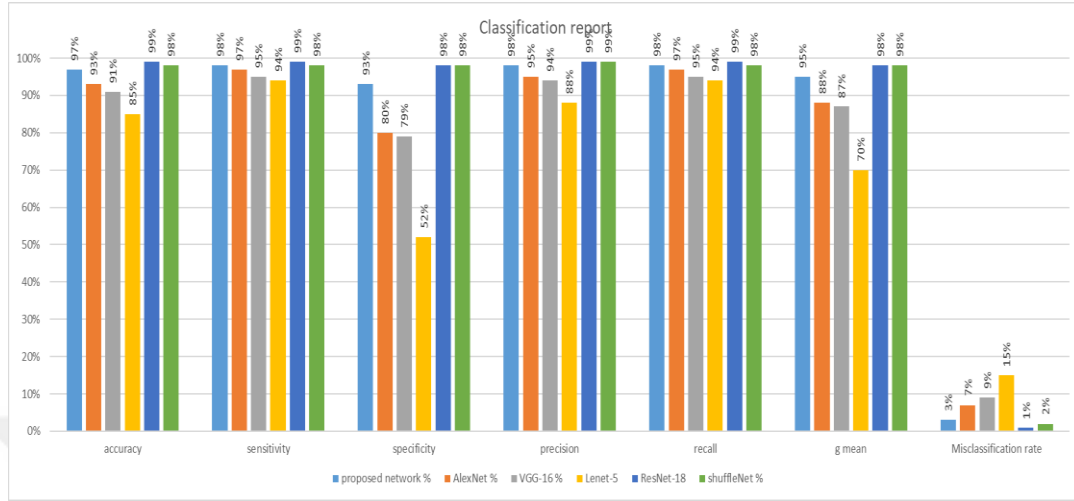


Figure 31. Pre-trained deep learning classifier results (graphical representation)

5.4. Confusion Matrix

The confusion matrix shows the number of positive cancer cases that are correctly predicted as positive by the respected classifier and the number of negative cases that are predicted as negatives. Also, the number of negative cases predicted as positive and the number of positive cases predicted as negative. The confusion matrixes of the above pre-trained algorithms are illustrated below, and the green highlighted areas in the confusion matrix plot present the correctly predicted cases (i.e. positives as positives and negatives as negatives). In order to calculate the confusion matrix, the following calculations are required.

Confusion matrix of VGG-16

For VGG-16, the number of negative cases that were predicted as positives was 4.1%, while the number of positive cases that were predicted as negative was 4.4%, as shown in Figure 33.

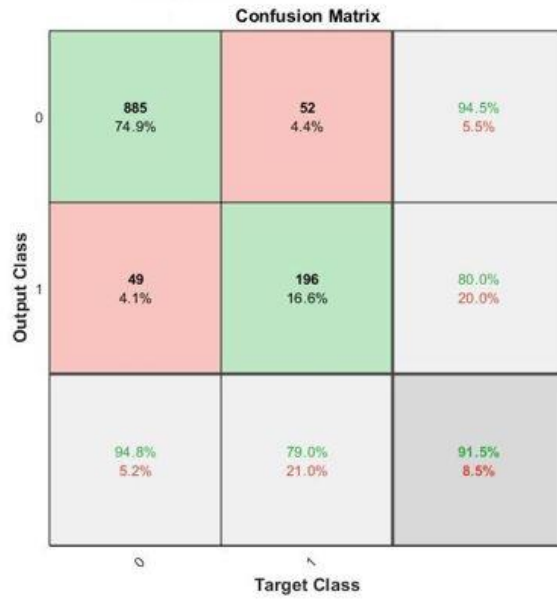


Figure 32. Confusion matrix of VGG-16

- **Confusion matrix of AlexNet**

For AlexNet, the number of negative cases that were predicted as positives was 2.5%, while the number of positive cases that were predicted as negative was 4.1%, as shown in Figure 34.

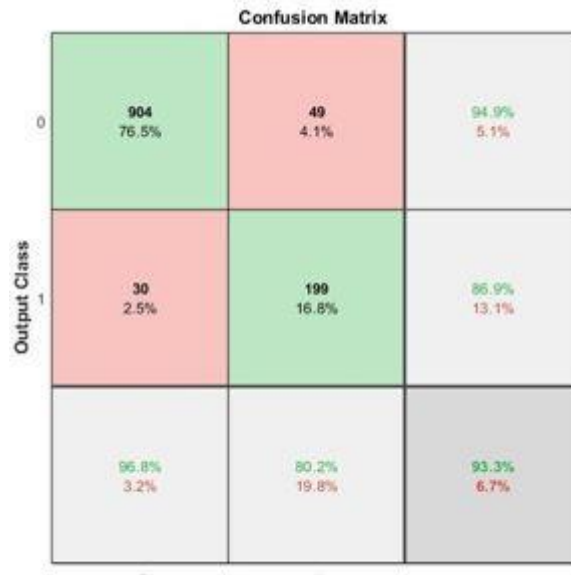


Figure 33. Confusion matrix of AlexNet

- **Confusion matrix of LeNet-5**

For LeNet, the number of negative cases that were predicted as positives was 4.8%, while the number of positive cases that were predicted as negative was 10%, as shown in Figure 35.

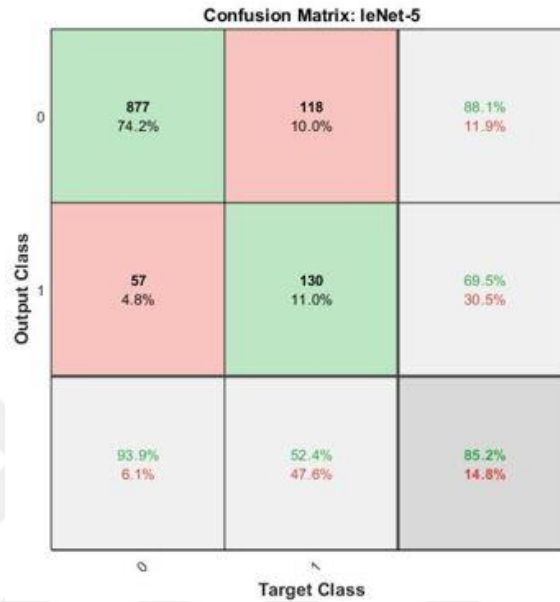


Figure 34. Confusion matrix of LeNet-5

- **Confusion matrix of ResNet-18**

For ResNet-18, the number of negative cases that were predicted as positives was 0.6%, while the number of positive cases that were predicted as negative was 0.5%, as shown in Figure 36.

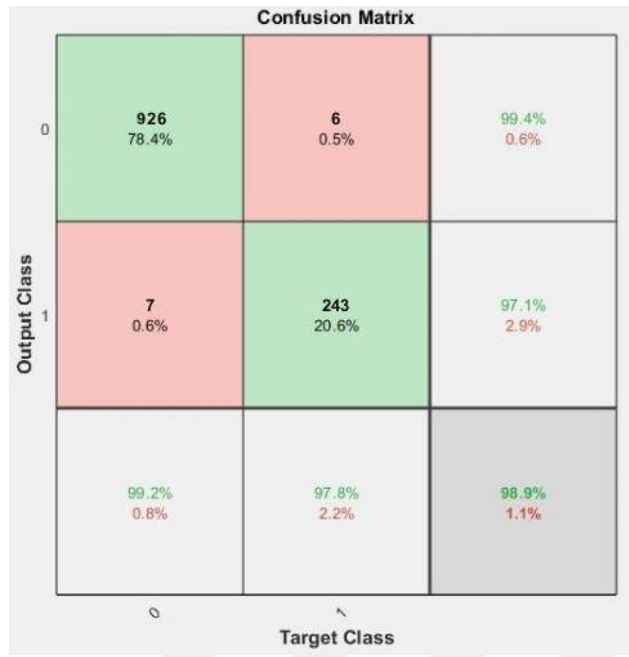


Figure 35. Confusion matrix of ResNet-18

- **Confusion matrix of Shufflenet**

For ShuffleNet, the number of negative cases that were predicted as positives was 1.1%, while the number of positive cases that were predicted as negative was 0.2%, as shown in Figure 37.

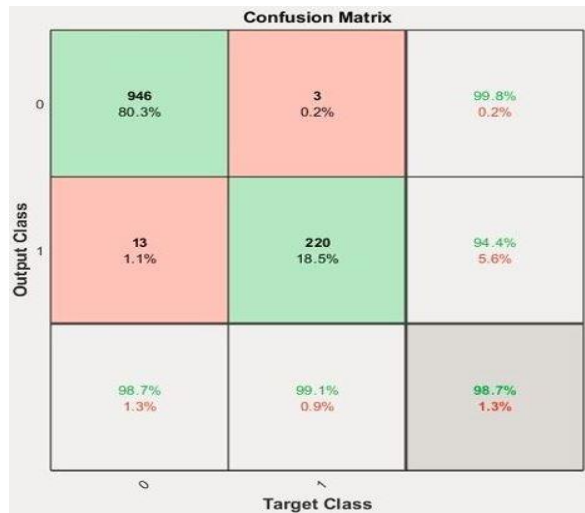


Figure 36. Confusion matrix of ShuffleNet

It can be seen that ResNet-18 made the minimum erroneous predictions, according to the percentage of positive as negative predictions and negative as positive predictions.

5.5. Machine Learning Models

Following the examination of the CNN models, each image was converted to a signal row array (single-dimensional array) and fed into algorithms similar to those described in Table 23 of all algorithms in disease prediction, with performance metrics recorded and tabulated.

Table 23. Accuracy of prediction measure for all the algorithms

Measure	Proposed Network	Gaussian Naive Bayes	kernel Naive Bayes	KNN	Linear Discriminant	Logistic Regression	SVM	Decision Tree
Accuracy	97%	80%	83%	86%	90%	86%	90%	85%
Sensitivity	98%	94%	94%	92%	94%	92%	93%	91%
Specificity	93%	51%	57%	67%	77%	64%	80%	65%
Precision	98%	79%	84%	91%	94%	89%	95%	91%
Recall	98%	94%	94%	92%	94%	92%	93%	91%
F1 Score	98%	86%	89%	91%	94%	91%	94%	91%
G Mean	95%	70%	73%	79%	85%	77%	86%	77%
Misclassification Rate	3%	20%	17%	14%	10%	14%	10%	15%

By comparing the suggested network to the basic evaluation measures that define the efficiency of NNs, it was discovered that the proposed network beats all networks. The networks were trained on the same collected data, where the values were calculated through the MATLAB software. According to Table 23, both SVM and linear discrimination algorithms were scored with 90% accuracy. Actually, none of them outperformed the other, and as we can see, if the sensitivity was slightly higher on one, the specificity might be higher on the other. Hence, it can be said that both algorithms

have a similar performance. Figure 38 shows a graphic of the results of the ML classification performance metrics for the tested ML algorithm, represented by the parameters of the classification report.

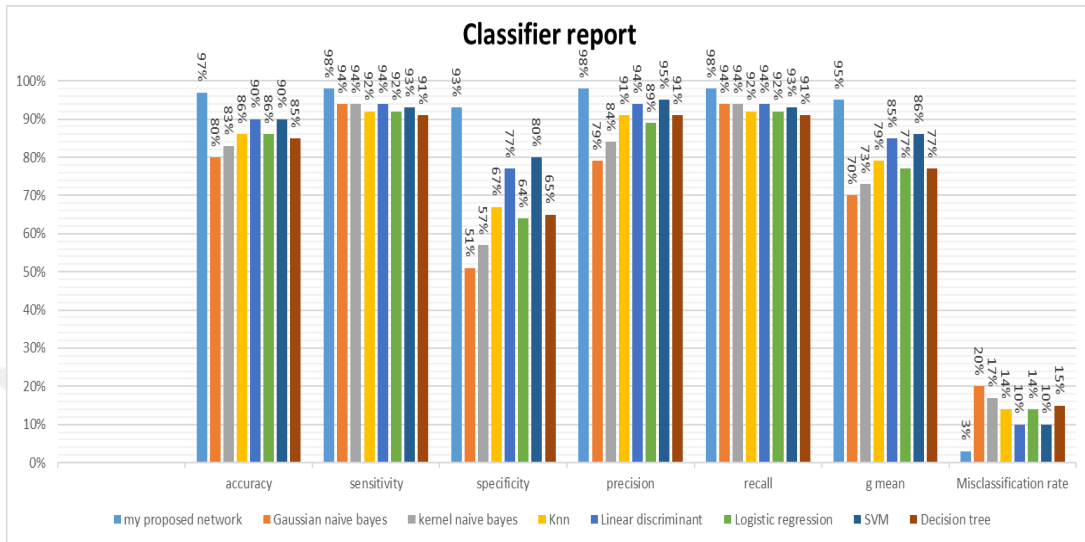


Figure 37. Graphical representation of the results of machine learning approaches showing prediction measures for all the algorithms

- **Confusion Matrix of Logistic Regression**

For logistic regression, the number of negative cases that were predicted as positives was 27%, while the number of positive cases that were predicted as negative was 11%, as shown in Figure 39.

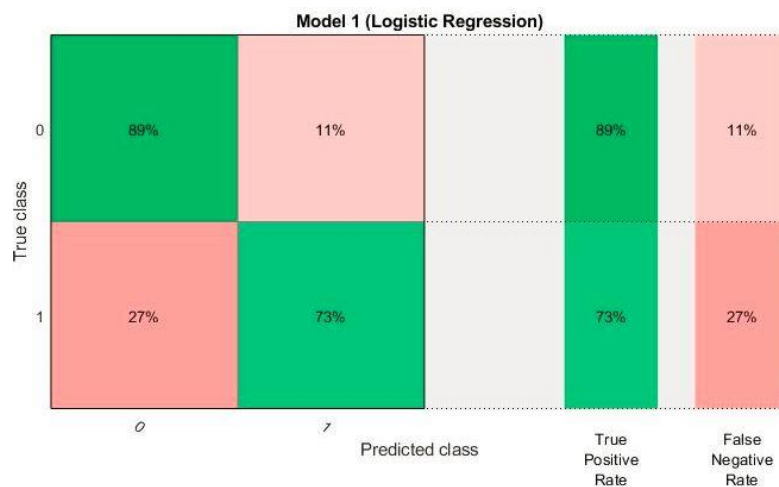


Figure 38. Confusion matrix of Logistic Regression

- **Confusion Matrix of Kernel Naïve Bayes**

For kernel naive Bayes, the number of negative cases that were predicted as positives was 19%, while the number of positive cases that were predicted as negative was 16%, as shown in Figure 40.

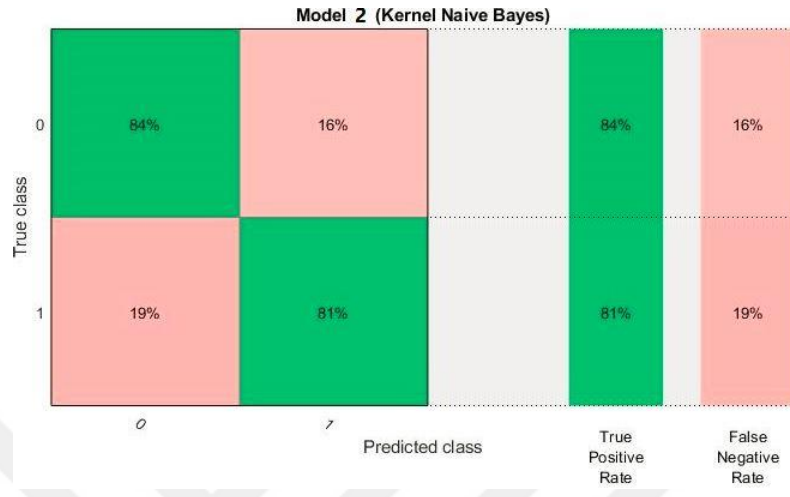


Figure 39. Confusion matrix of Kernel Naïve Bayes

- **Confusion Matrix of Linear Discriminant**

For linear discriminant, the number of negative cases that were predicted as positives was 24%, while the number of positive cases that were predicted as negative was 6%, as shown in Figure 41.

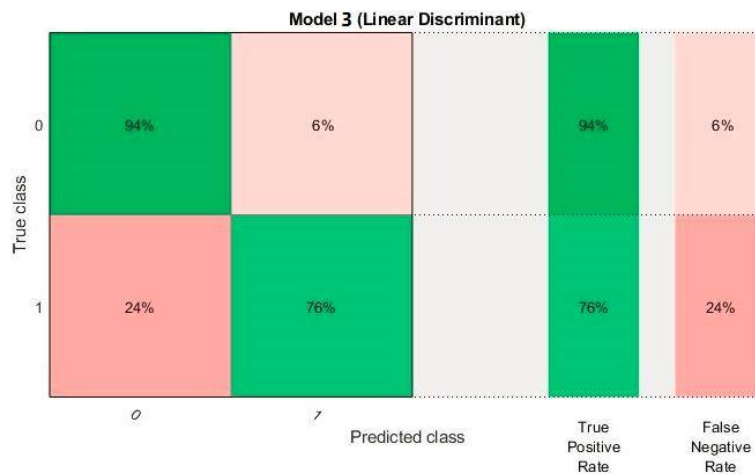


Figure 40. Confusion matrix of Linear Discriminant

- **Confusion Matrix of Fine KNN**

For fine KNN, the number of negative cases predicted as positives was 31%, while the number of positive cases predicted as negative was 9%, as shown in Figure 42.

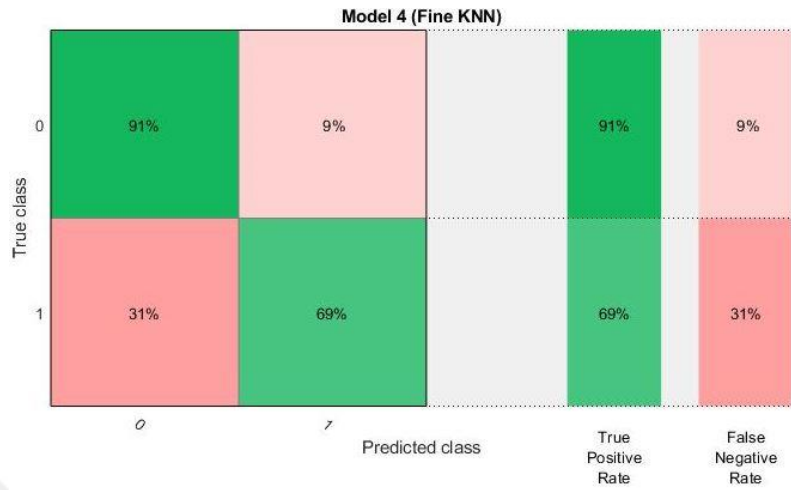


Figure 41. Confusion matrix of KNN

- **Confusion Matrix of Gaussian Naïve Bayes**

For Gaussian naïve Bayes, the number of negative cases that were predicted as positives was 17%, while the number of positive cases that were predicted as negative was 21%, as shown in Figure 43.

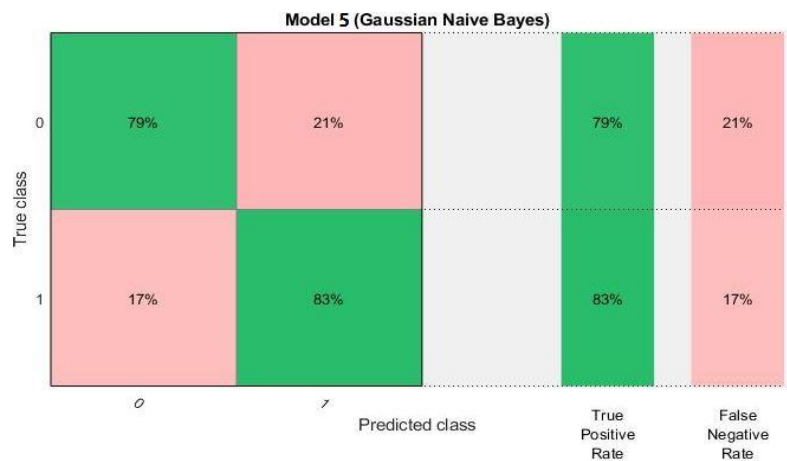


Figure 42. Confusion matrix of Gaussian Naïve Bayes

- **Confusion Matrix of Fine Tree**

For fine tree, the number of negative cases that were predicted as positives was 36%, while the number of positive cases that were predicted as negative was 9%, as shown in Figure 44.

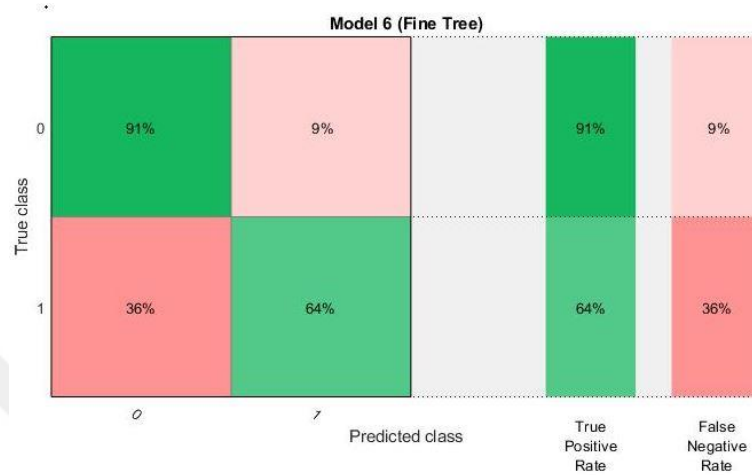


Figure 43. Confusion matrix of Fine Tree

- **Confusion Matrix of Linear SVM**

For linear SVM, the number of negative cases that were predicted as positives was 27%, while the number of positive cases that were predicted as negative was 5%, as shown in Figure 45.

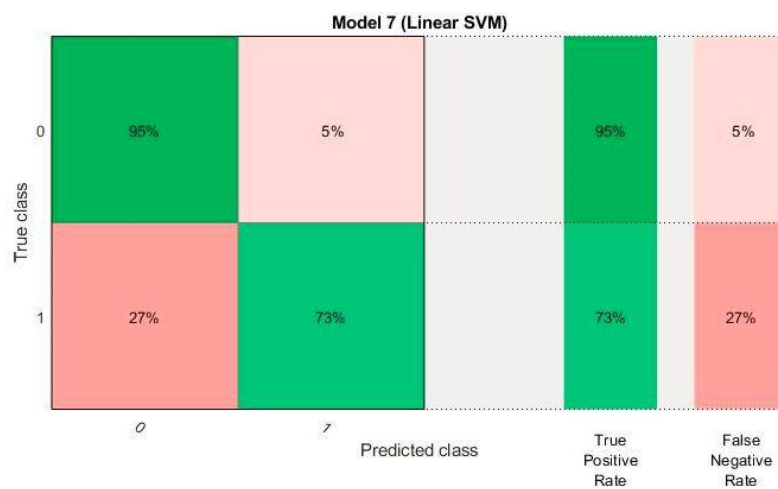


Figure 44. Confusion matrix of linear SVM

For SVM, the number of negative cases predicted as positives was 27%, while positive cases predicted as negative was 5%. As a result, the SVM and linear discrimination had the fewest error predictions.

5.6. Regression Model

Other metrics, such as MSE, MAE and RMSE, were measured for all the ML algorithm approaches. Table 24 illustrates the above metrics of all the ML algorithms. MSE gives the distance between the regression line and the classification model output. This number is always a positive, as it takes the square of the errors. MSE holds a bigger value than the other regression model metrics, such as MAE and RMSE. On the other hand, MAE is another metric for determining the distance of the prediction output from the regression line. MAE is lesser than the MSE in value but gives a similar interpretation.

Table 24. Regression model performance metrics for the machine learning algorithms as compared with proposed CNN

MODEL	MSE	RMSE	MAE
CNN	0.0726	0.269393	0.0726
SVM	0.082864	0.287861	0.144866
Linear Discriminant	0.083817	0.289513	0.1666842
Logistic Regression	0.09212	0.26134	0.23642
KNN	0.103295	0.3213958	0.103295
Decision Tree	0.152328	0.390293	0.161808
kernel Naive bayes	0.31450	0.52115	0.30461
Gaussian Naive Bayes	0.60400	0.757535	0.64400

Observations: The outperformed algorithms, according to the previous performance metrics, i.e. the accuracy measure, F1 score, etc., are SVM and linear discrimination algorithms. However, according to regression model metrics, i.e. MAE, MSE and RMSE, both SVM and linear discrimination algorithms had the lowest MAE (0.144866), MSE (0.082864) and RMSE (0.082864). (0.287861). Compared with the results of the performance measures of the proposed CNN regression model, MAE = 0.0726, MSE = 0.0726 and RMS = 0.269393, the proposed CNN outperformed the ML techniques.

5.7. ROC and AUC Measures

The receiver operating characteristic curve (ROC) and the area under the curve (AUC) are other classification performance measures. These measures are meant for detecting how well the model is classifying (predicting) the zero class as zero and one class as one. The higher value of AUC corresponds to the high capability of the classification model to distinguish between the classes. Both measures were calculated for the models made in this thesis hereafter.

5.7.1. Proposed Model (CNN)

As shown in Figure 46, the proposed CNN's AUC was 0.9906, representing a number close to one among all classifier algorithms.

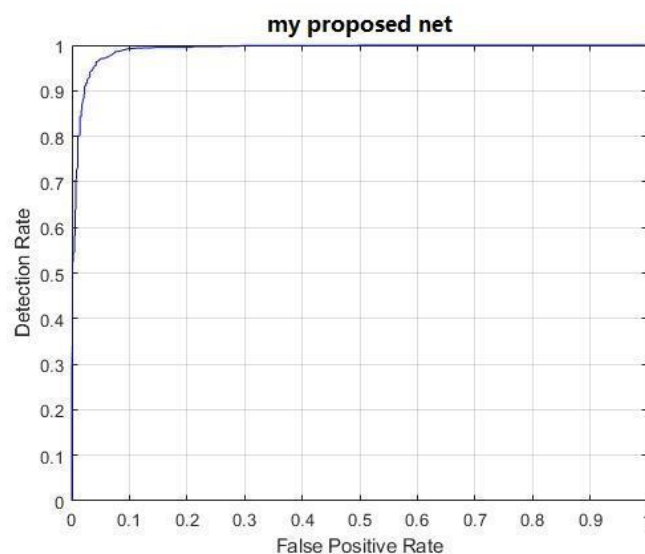


Figure 45. Demonstration of ROC and AUC region for proposed CNN

5.7.2. Machine Learning Models

The ROC and AUC values for the ML algorithms used in this thesis were measured. The AUC values for the ML methods are shown in Table 25 and Figures 47 to 53.

Table 25. AUC values for the machine learning algorithms

Measure	Decision Tree	Gaussian Naïve Bayes	kernel Naïve Bayes	KNN	Linear Discriminant	Logistic Regression	SVM
AUC	0.73	0.83	0.84	0.80	0.94	0.83	0.95

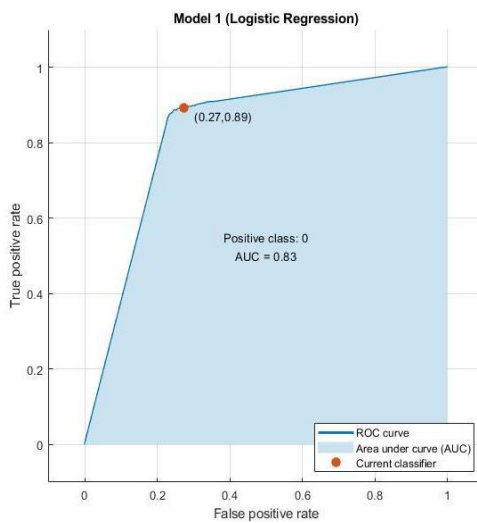


Figure 47. ROC and AUC demonstration for the Logistic Regression algorithm

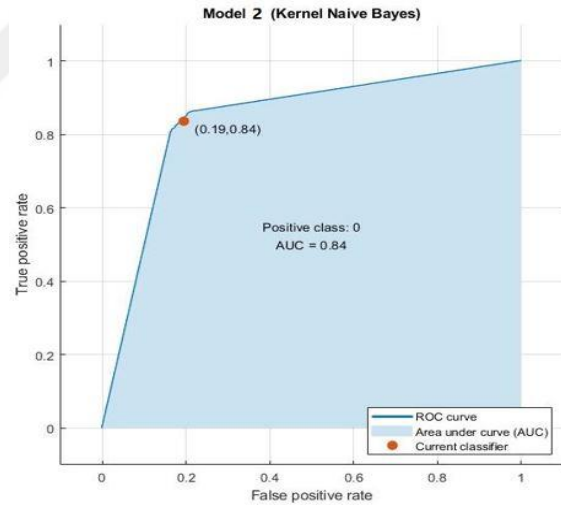


Figure 46. ROC and AUC demonstration for the Kernel Naïve Bayes algorithm

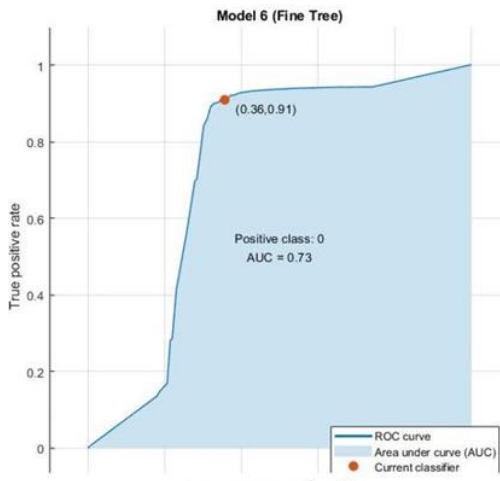


Figure 48. ROC and AUC demonstration for Fine Tree algorithm

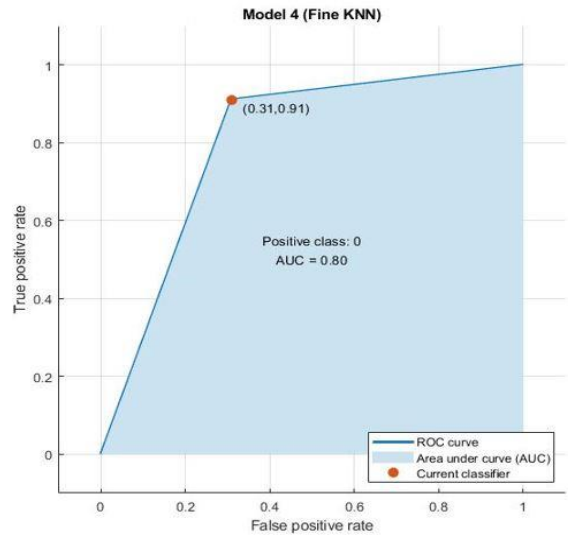


Figure 50. ROC and AUC demonstration for KNN algorithm for the Fine KNN algorithm

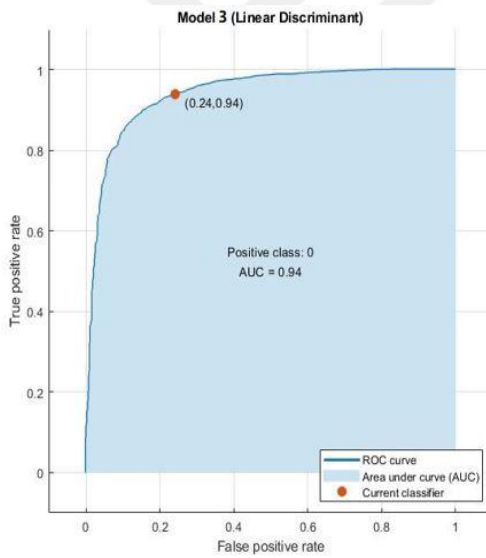


Figure 51. ROC and AUC demonstration for the Linear Discriminant algorithm

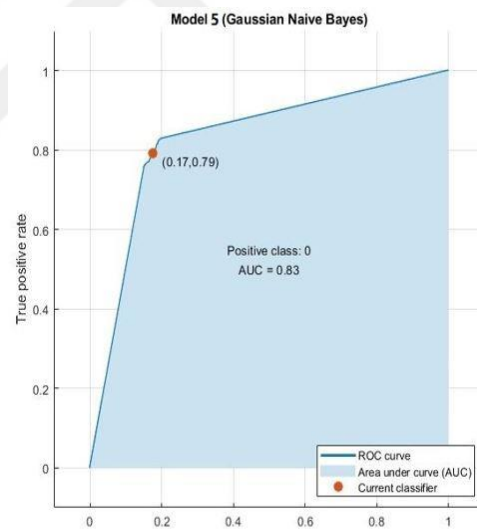


Figure 49. ROC and AUC demonstration for the Gaussian Naïve Bayes algorithm

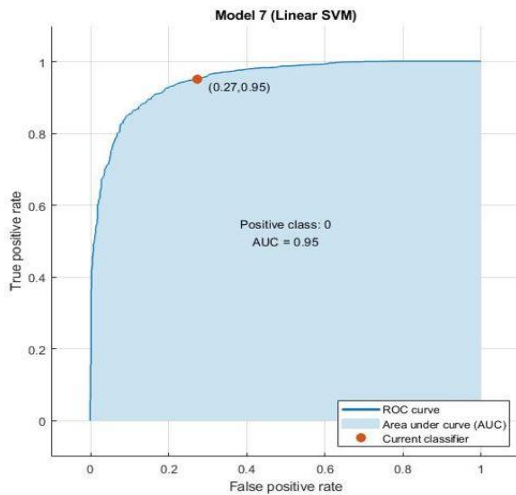


Figure 52. ROC and AUC

demonstration for the Linear SVM algorithm

Observations: From the results above, both the SVM and linear discrimination algorithms outperformed by scoring the biggest AUC values, i.e. 0.95 and 0.94. That is consistent with previous scoring results, such as accuracy, recall, etc. However, when these scores were compared to the proposed CNN, it was clear that CNN could ingest images and provide the best classification performance, as measured by the AUC value of 0.99 and the other classification performance metrics mentioned in the previous sections. The AUC values for the ML methods and the proposed CNN are shown in Figure 54.

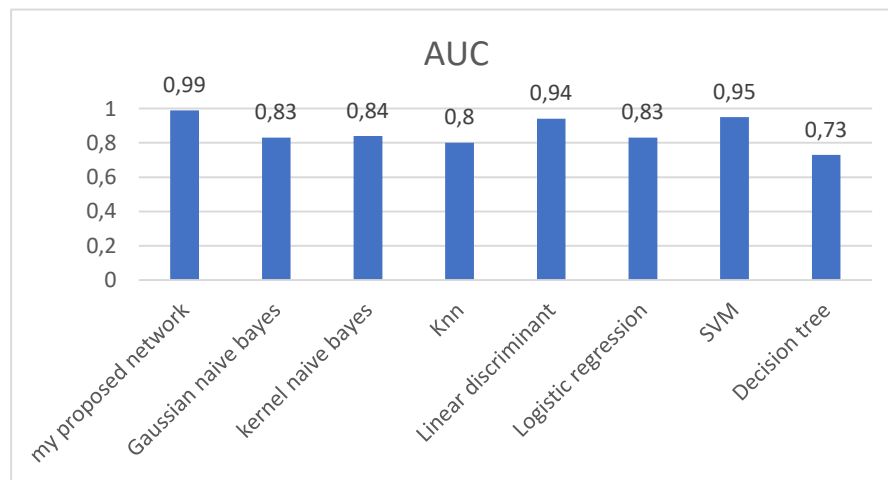


Figure 53. Graphical comparison of AUC values between the proposed CNN and machine learning algorithms

5.8. Comparisons Between Implementation (CPU and PYNQ)

On the implementation side of the thesis, a comparison was made in implementing the suggested CNN models on both the CPU and the PYNQ board using the identical CNN model training settings. The execution time on the PYNQ board was noticeably faster than that of the CPU. Table 26 and Figure 55 show the results. In terms of precision, however, the proposed FPGA Model 2 outperformed the CPU. As shown in Figure 56, the precision value of the first model converged between the CPU and the PYNQ board.

Table 26. Performance Comparison of the proposed models (CPU and FPGA)

Model Name	Accuracy	Time (seconds)
CPU-CNN model	96.87	58.2
FPGA-Model 2	97.776	16.07
FPGA-Model 1	96.576	39.73

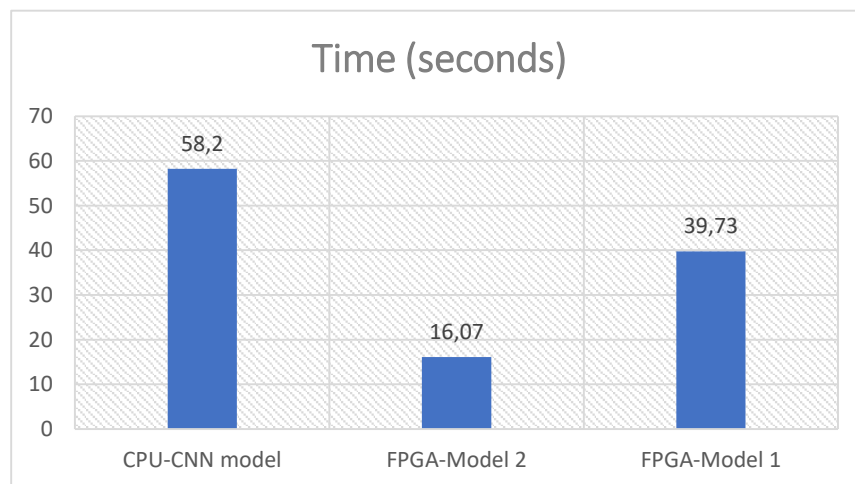


Figure 54. Graphical representation of the training time comparison of the proposed models (CPU and FPGA)

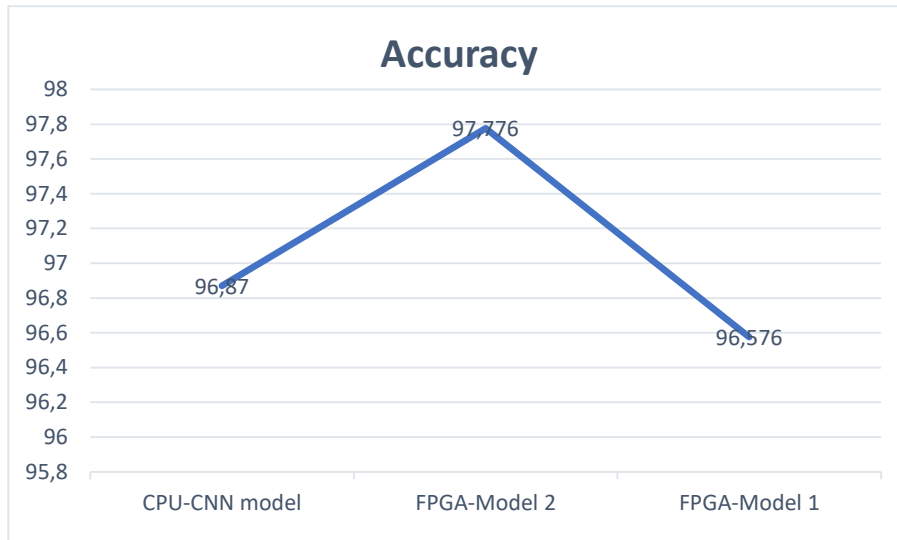


Figure 55. Graphical representation of the accuracy comparison of the proposed models (CPU and FPGA)

5.9. Comparisons Between the Implementation of AlexNet and LeNet (CPU and PYNQ)

A comparison was made in implementing the AlexNet and LeNet models on both the CPU and the PYNQ board using the identical CNN model training settings. The PYNQ board's execution time was noticeably faster than the CPU's. Table 27 shows the results. In terms of accuracy, the AlexNet and LeNet models outperformed the FPGA over the CPU.

Table 27. Performance Comparison of the AlexNet and LeNet models (CPU and FPGA)

Model Name	Accuracy	Time (seconds)
CPU-AlexNet	93.32	5566.11
CPU- LeNet	85.13	245
FPGA- AlexNet	94.56	240.87
FPGA- LeNet	85.73	53.75

CHAPTER SIX

CONCLUSION AND DISCUSSION

5.1. Conclusion

DL approaches are acquired increasingly important in today's life technology. DL techniques have been used for a wide range of scientific and engineering applications. ML and AI are being used in medical applications as well. Both are used for disease or sickness diagnostic prediction using historical data ranging from raw values (numbers or characters), images or even videos.

In this work, a proposed reliable model for BC diagnosis was generated using a CNN architecture, and the performance of the proposed CNN was compared across two types of education. The first learning was performed via the CPU using a set of ML algorithms, such as random forest, KNN, etc. Moreover, pre-trained classifiers of DL algorithms on CPU were also used, AlexNet, VGG-16, ResNet, etc. The second learning was performed via the FPGA development board using the proposed CNN as well as DL algorithms, AlexNet and LeNet-5, focusing on improving BC diagnosis, preventing overfitting and obtaining the highest possible prediction accuracy.

CNNs can be used to analyse big data with high efficiency and less training error due to their flexible and functional structure. Performance metrics such as accuracy, MSE, MAE and RMSE were deployed for the model's quality assessment.

In order to evaluate each algorithm's performance for the required task, the following results were obtained using the indicated algorithms (cancer diagnosis). The proposed CNN outperformed ML algorithms, according to CPU-based results. Methods, as well as DL algorithms. A prediction accuracy equal to 95.12% was observed in the results while using CNN over the other algorithms.

The random forest algorithm also achieved a good prediction accuracy, but its process took a long time, which is considered the main drawback of its performance. The other algorithms also achieved different accuracy measures, and all were less than the proposed CNN.

Moreover, pre-trained DL models, such as VGG-16, AlexNet, ResNet-18, ShuffleNet and LeNet, were used for the same goal. The ResNet-18 algorithm also achieved a

high prediction accuracy, but the process took longer, which was the main source of performance loss. Despite this, the prediction accuracy of the proposed algorithm was very high with a shorter time compared with other algorithms.

5.2. Discussion

The CNN model outperformed the other deployed models because it did not require an extra step to convert a multi-dimensional image into a single-dimensional array, and it also included extraction, which is often overlooked when employing deep learning paradigms. The results of the proposed CNN method were more dependable than those of other access methods.

Discussions.

- 1.** While changing the filter size (window) and fixing the number of filters to 32 and the pooling layer window, the accuracy results were observed changing randomly within the 85–88% range.
- 2.** By fixing the window size, increasing the number of filters of the Con2D layer and fixing the window size of the pooling layer, a gradual increase was observed in the accuracy results.
- 3.** While fixing the filter size (window) of Con2D, fixing the number of filters in the same layer and changing the pooling layer window, the performance accuracy results were observed changing randomly within the 84–88% range.
- 4.** While changing the filter size (window) of the second Con2D layer and fixing the number of filters to 64, as well as the pooling layer window, the performance of the accuracy results were observed changing randomly within the 80–90% range.
- 5.** By changing the number of the filter of the second Con2D layer from 16 to 54 and fixing the window size of both the pooling and Con2D layers, the accuracy results changed randomly within the 80–85% range.
- 6.** Fixing the number of filters to 64 on the second Con2D layer, as well as its window to 3 x 3, and varying the second pooling layer window from 2 x 2 to 12 x 12, results dropped from 90 to 80% randomly within the intermediate stages.

7. Varying the window size of the third Con2D layer from 3 x 3 to 13 x 13 and fixing the number of filters of the same layer to 128 with a constant pooling window of 2 x 2, the accuracy results dropped randomly from 95% to 91%.

8. Varying the number of filters of the third Con2D from 32 to 128 and fixing the pooling window to 2 x 2, results enhanced from 95–97% randomly. The biggest number of filter deployments on the Con2D layers within the CNN model was the key factor for accuracy increments. However, the size of the filter must be chosen appropriately, so that model overfitting is prevented.

Comparison with pre-trained neural networks: Irrespective of the wide structure (complexity) of the pre-trained NNs, such as ResNet-18, ShuffleNet, VGG-16, AlexNet and LeNet, the proposed CNN yielded the best prediction accuracy over the other NNs. ResNet-18 outperformed ShuffleNet, VGG-16, AlexNet and LeNet by scoring 99% of BC prediction accuracy.

Comparison with machine learning tools: The linear discrimination algorithm recorded 90% of BC prediction accuracy. However, the kernel naive Bayes algorithm had the lowest recorded accuracy. The proposed CNN also outperformed over all the ML algorithms.

5.3. Proposed CNN Over FPGA Development Board

According to Table 26 and Figures 55 and 56, the proposed FPGA-based DL model outperformed the other two models in terms of accuracy of cancer detection and the time of training. The implemented model on FPGA was similar to the CPU–CNN model from the structural point of view. On the other hand, the CPU model had a higher layer budget, such as a higher number of filters in each layer and higher filter dimensions, to meet an accuracy of 96.87%. The same impacted the classifier's performance, as training time crossed 86 seconds. The FPGA-based proposed CNN model provided higher accuracy and less computational costs for a similar CPU structure-proposed model. From that, it can be concluded that FPGA (PYNQ-Z2) can ensure a good alternative for those applications demanding less processing delay and higher accuracy, so it has a good scope on replacing the traditional CPUs and on paving

the road for task-oriented boards, which is set to perform a particular task with higher accuracy and less time.



REFERENCES

- Abhigna, P., Jerritta, S., Srinivasan, R., & Rajendran, V. (2017, April). Analysis of feed forward and recurrent neural networks in predicting the significant wave height at the moored buoys in Bay of Bengal. In 2017 International Conference on Communication and Signal Processing (ICCSP) (pp. 1856-1860). IEEE.
- Akbar, S., Peikari, M., Salama, S., Nofech-Mozes, S., & Martel, A. (2017). Transitioning between convolutional and fully connected layers in neural networks. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support* (pp. 143-150). Springer, Cham.
- Akhtar, N., & Ragavendran, U. (2020). Interpretation of intelligence in CNN-pooling processes: a methodological survey. *Neural computing and applications*, 32(3), 879-898.
- Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... & Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*.
- Aly, G. H., Marey, M., El-Sayed, S. A., & Tolba, M. F. (2021). YOLO Based Breast Masses Detection and Classification in Full-Field Digital Mammograms. *Computer Methods and Programs in Biomedicine*, 200, 105823.
- Araújo, T., Aresta, G., Castro, E., Rouco, J., Aguiar, P., Eloy, C., ... & Campilho, A. (2017). Classification of breast cancer histology images using convolutional neural networks. *PloS one*, 12(6), e0177544.
- Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186). Physica-Verlag HD.
- Chan, A., & Tuszynski, J. A. (2016). Automatic prediction of tumour malignancy in breast cancer with fractal dimension. *Royal Society open science*, 3(12), 160558.

- Chandra, A. L., Desai, S. V., Guo, W., & Balasubramanian, V. N. (2020). Computer vision with deep learning for plant phenotyping in agriculture: A survey. arXiv preprint arXiv:2006.11391.
- Chen, K., Huang, L., Li, M., Zeng, X., & Fan, Y. (2018, October). A compact and configurable long short-term memory neural network hardware architecture. In 2018 25th IEEE International Conference on Image Processing (ICIP) (pp. 4168-4172). IEEE.
- Dixon, J. M. (Ed.). (2012). ABC of breast diseases (Vol. 226). John Wiley & Sons.
- Dureja, A., & Pahwa, P. (2019). Analysis of non-linear activation functions for classification tasks using convolutional neural networks. *Recent Patents on Computer Science*, 12(3), 156-161.
- Ellis, H., & Mahadevan, V. (2013). Anatomy and physiology of the breast. *Surgery (Oxford)*, 31(1), 11-14.
- Fellbaum, C. (1998). Towards a representation of idioms in WordNet. In *Usage of WordNet in Natural Language Processing Systems*.
- Filipczuk, P., Fevens, T., Krzyżak, A., & Monczak, R. (2013). Computer-aided breast cancer diagnosis based on the analysis of cytological images of fine needle biopsies. *IEEE transactions on medical imaging*, 32(12), 2169-2178.
- Fitzmaurice, C., Allen, C., Barber, R. M., Barregard, L., Bhutta, Z. A., Brenner, H., & Dicker, D. J. (2017). A systematic analysis for the global burden of disease study. *JAMA Oncol*, 3(4), 524-548.
- Fletcher, C. D., Unni, K., & Mertens, F. (2002). World Health Organization classification of tumours. Pathology and genetics of tumours of soft tissue and bone. IARC press.
- George, Y. M., Zayed, H. H., Roushdy, M. I., & Elbagoury, B. M. (2013). Remote computer-aided breast cancer detection and diagnosis system based on cytological images. *IEEE Systems Journal*, 8(3), 949-964.

- Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). Cambridge: MIT press.
- Gour, M., Jain, S., & Sunil Kumar, T. (2020). Residual learning based CNN for breast cancer histopathological image classification. *International Journal of Imaging Systems and Technology*, 30(3), 621-635.
- Gschwend, D. (2020). Zynqnet: An fpga-accelerated embedded convolutional neural network. arXiv preprint arXiv:2005.06892.
- Gupta, V., & Bhavsar, A. (2017). Breast cancer histopathological image classification: is magnification important?. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 17-24).
- Gurcan, M. N., Boucheron, L. E., Can, A., Madabhushi, A., Rajpoot, N. M., & Yener, B. (2009). Histopathological image analysis: A review. *IEEE reviews in biomedical engineering*, 2, 147-171.
- Hamdan, M. K., & Rover, D. T. (2017, December). VHDL generator for a high performance convolutional neural network FPGA-based accelerator. In *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)* (pp. 1-6). IEEE.
- Han, Z., Wei, B., Zheng, Y., Yin, Y., Li, K., & Li, S. (2017). Breast cancer multi-classification from histopathological images with structured deep learning model. *Scientific reports*, 7(1), 1-10.
- Irshad, H., Veillard, A., Roux, L., & Racoceanu, D. (2013). Methods for nuclei detection, segmentation, and classification in digital histopathology: a review—current status and future potential. *IEEE reviews in biomedical engineering*, 7, 97-114.
- Ismail, N. S., & Sovuthy, C. (2019, August). Breast cancer detection based on deep learning technique. In *2019 International UNIMAS STEM 12th Engineering Conference (EnCon)* (pp. 89-92). IEEE.

- Jiang, Z. (2019, December). A novel crop weed recognition method based on transfer learning from VGG16 implemented by keras. In IOP Conference Series: Materials Science and Engineering (Vol. 677, No. 3, p. 032073). IOP Publishing.
- Jørgensen, H. (2017). Automatic license plate recognition using deep learning techniques (Master's thesis, NTNU).
- Kahya, M. A., Al-Hayani, W., & Algamal, Z. Y. (2017). Classification of breast cancer histopathology images based on adaptive sparse support vector machine. *Journal of Applied Mathematics and Bioinformatics*, 7(1), 49.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 1097-1105.
- Kumar, R. A (2020). Survey on Memetic Algorithm and Machine learning Approach to Traveling Salesman Problem.
- Kuon, I., & Rose, J. (2010). *Quantifying and exploring the gap between FPGAs and ASICs*. Springer Science & Business Media.
- Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. Y. (2011, January). On optimization methods for deep learning. In ICML.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- Mandwi, I., Bhondge, B., Thakre, K., Dhande, G., & Patiye, I. (2016). *Cancer Identification by Analysis of WBC*.
- Mitchell, T. M. (1997). Does machine learning really work?. *AI magazine*, 18(3), 11-11.

- Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). Foundations of machine learning. MIT press.
- Motlagh, M. H., Jannesari, M., Aboulkheyr, H., Khosravi, P., Elemento, O., Totonchi, M., & Hajirasouliha, I. (2018). Breast cancer histopathological image classification: A deep learning approach. *BioRxiv*, 242818.
- Olivito, J., Gran, R., Resano, J., González, C., & Torres, E. (2015). Performance and energy efficiency analysis of a Reversi player for FPGAs and General Purpose Processors. *Microprocessors and Microsystems*, 39(2), 64-73.
- Omonigho, E. L., David, M., Adejo, A., & Aliyu, S. (2020, March). Breast Cancer: Tumor Detection in Mammogram Images Using Modified AlexNet Deep Convolution Neural Network. In 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS) (pp. 1-6). IEEE.
- Rahman, A. S. A., Belhaouari, S. B., Bouzerdoun, A., Baali, H., Alam, T., & Eldaraa, A. M. (2020, February). Breast Mass Tumor Classification using Deep Learning. In 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIOT) (pp. 271-276). IEEE.
- Rahman, A., Lee, J., & Choi, K. (2016, March). Efficient FPGA acceleration of convolutional neural networks using logical-3D compute array. In 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 1393-1398). IEEE.
- Rørmann Olsen, I. (2018). Dealing with word ambiguity in NLP. Building appropriate sense representations for Danish sense tagging by combining word embeddings with wordnet senses.
- S. E. Wahlstrom, "Programmable logic arrays cheaper by the millions," *Electronics*, vol. 40, pp. 90–95, December 1967.
- Salem, M. A. M. (2018, December). Mammogram-Based cancer detection using deep convolutional neural networks. In 2018 13th International Conference on Computer Engineering and Systems (ICCES) (pp. 694-699). IEEE.

- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
- Sinha, T., Verma, B., & Haidar, A. (2017). Optimization of convolutional neural network parameters for image classification. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-7). IEEE.
- Spanhall, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2016, July). Breast cancer histopathological image classification using convolutional neural networks. In *2016 international joint conference on neural networks (IJCNN)* (pp. 2560-2567). IEEE.
- Spanhol, F. A., Oliveira, L. S., Cavalin, P. R., Petitjean, C., & Heutte, L. (2017, October). Deep features for breast cancer histopathological image classification. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 1868-1873). IEEE.
- Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2015). A dataset for breast cancer histopathological image classification. *Ieee transactions on biomedical engineering*, 63(7), 1455-1462.
- Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2015). A dataset for breast cancer histopathological image classification. *Ieee transactions on biomedical engineering*, 63(7), 1455-1462.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).
- Trimberger, S. M. (Ed.). (2012). *Field-programmable gate array technology*. Springer Science & Business Media.
- Tsochatzidis, L., Koutla, P., Costaridou, L., & Pratikakis, I. (2021). Integrating segmentation information into CNN for breast cancer diagnosis of

mammographic masses. *Computer Methods and Programs in Biomedicine*, 200, 105913.

Wei, B., Han, Z., He, X., & Yin, Y. (2017, April). Deep learning model based breast cancer histopathological image classification. In 2017 IEEE 2nd international conference on cloud computing and big data analysis (ICCCBDA) (pp. 348-353). IEEE.

Yan, R., Ren, F., Wang, Z., Wang, L., Zhang, T., Liu, Y., ... & Zhang, F. (2020). Breast cancer histopathological image classification using a hybrid deep neural network. *Methods*, 173, 52-60.

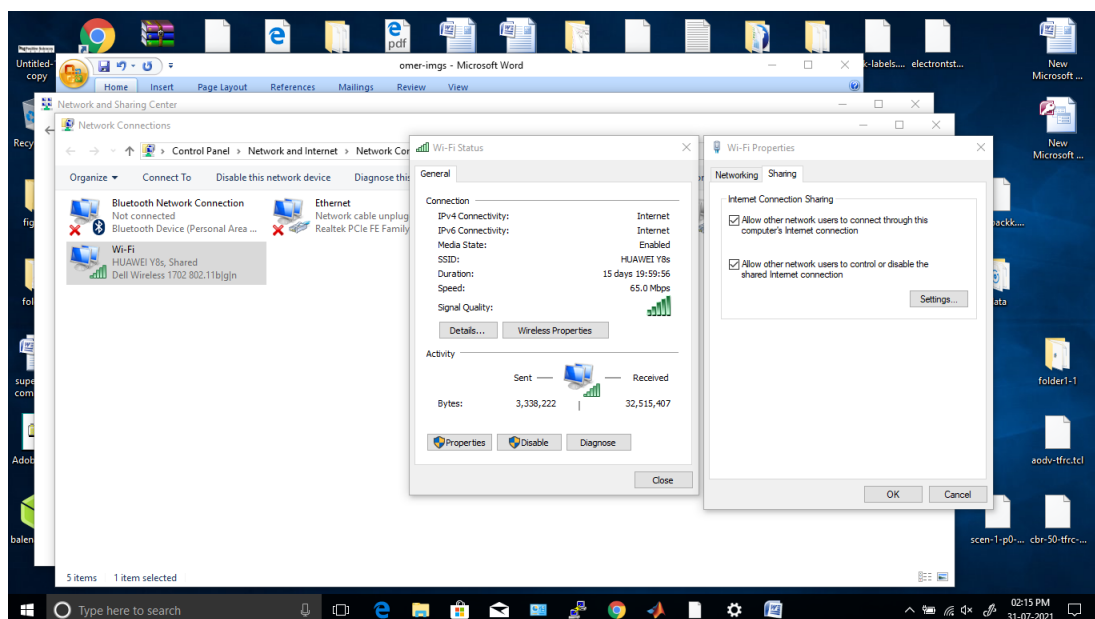
Zhang, H. (2015). Microwave imaging for ultra-wideband antenna based cancer detection.

Zhang, Z., Wang, Y., Zhang, J., & Mu, X. (2019, October). Comparison of multiple feature extractors on Faster RCNN for breast tumor detection. In 2019 8th International Symposium on Next Generation Electronics (ISNE) (pp. 1-4). IEEE.

ANNEXES

ANNEXES A. Three main requirements are to be fulfilled upon reaching to this step:

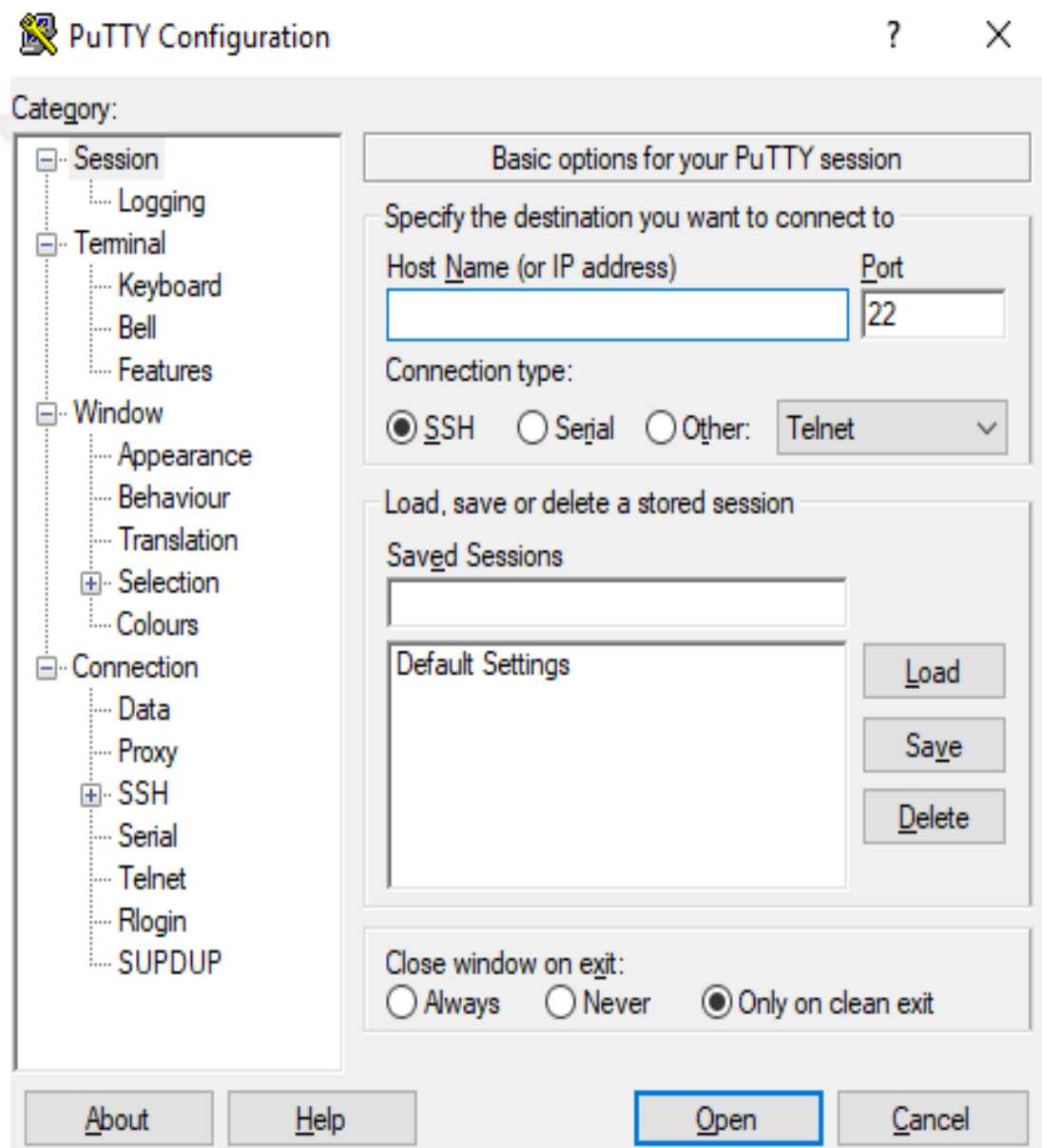
1. insertion SD care to the development board, connecting the development board with both micro-USB cable and ethernet cable with the computer. Development board is to be empowered through micro-USB cable with +5v DC from computer itself, while the other connection (ethernet connection) is to be used for data exchanging between the FPGA network card and the computer network card. Using of ON-OFF sliding switch, board can be started, it is important to keep the bin connector (on the board itself) to the position of SD card boot where FPGA can understand to boot the system flashed on the SD card.
2. before any further steps, FPGA need to get linked with internet, this can be availed using internet sharing from PC network card. Simply by opening “network and sharing center” → adaptor settings→ selection of main network card (local area network) and from there sharing of internet can be set. It is important to realize that no IP addressing is made so far and after sharing of the internet by local area network card, computer will automatically assign IP address to FPGA board i.e. 192.168.132.1.



3. Putty configuration in order to get access to the OS of FPGA (empowered with internet) where the python configurations can be made. Two important things to be taken care in order to successfully access the terminal of PYNQ.

a. Hostname and Clock size: in order to find the host name, accessing to the “Device Manager” where all USB ports are appearing. From that, name of USB port can be identified. Transfer the setting of Putty into Telnet.

b. Clock size must be set into 115200 Hz. is demonstrating Putty software front end.



ANNEXES B. Deep Learning Libraries

Shell of PYNQ-Z2 which has been accessed using Putty is providing Linux environments for establishing Python related tools used in this project. Such process can be initiated by fulfilling of the following steps:

1. since Linux environments is already installed in the flashed image (on SD card) and boot of PYNQ-Z2 board is set to be from SD card; system is ready now for implementation of further process. First important action is to install the Python; at the time this thesis is made, Python 3.8 is available so-to-say, same version can be installed on the board. The following command can be typed on the terminal in order to download and install the required source.

command: \$ sudo apt-get install python

2. most of the required python libraries can be installed using PIP tool; however, Python3.8 is providing PIP3. Get-pip.py is installed by running the following command:
`$ curl -sSL https://bootstrap.pypa.io/get-pip.py -o get-pip.py`

3. many of Python sources cannot be installed even when the prementioned two steps are accomplished. Thus, the so called virtual environments can be used for safe installing. For that, firstly virtual environment tool need to be installed as hereinafter:

command: \$sudo pip install virtualenv

The term SUDO is used for giving the authenticity of root user for this installation which acts good in many installations so, it is highly recommended to finish the installation easily. In order to use the virtual environments, following commands can be typed in the terminal:

Creation of new virtual environments:

command:\$virtualenv venv

Changing the directory is must; here in order to use the being created virtual environment, it is necessary to access the location of this virtual environment by using CD command; thereafter; virtual environment can be used by typing the following:

Code function: \$source venv/bin/activate

Soon after the required task gets over, the virtual environments can be stopped using the command: `$ deactivate`

All the required libraries that are needed for accomplishing the project are given below

ANNEXES C. Caffe Project

Caffe (Convolutional Architecture for Fast Feature Embedding) provides multimedia scientists and practitioners with a clean and modifiable framework for state-of-the-art deep learning algorithms and a collection of reference models. The framework is a BSD-licensed C++ library with Python and MATLAB bindings for training and deploying generalpurpose convolutional neural networks and other deep models efficiently on commodity architectures. Caffe fits industry and internet-scale media needs by CUDA GPU computation, processing over 40 million images a day on a single K40 or Titan GPU (≈ 2.5 ms per image). By separating model representation from actual implementation, Caffe allows experimentation and seamless switching among platforms for ease of development and deployment from prototyping machines to cloud environments. Caffe is maintained and developed by the Berkeley Vision and Learning Center (BVLC) with the help of an active community of contributors on GitHub. It powers ongoing research projects, large-scale industrial applications, and startup prototypes in vision, speech, and multimedia.

1. Install dependencies:

Commands:

```
$sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev  
libhdf5-serial-dev protobuf-compiler
```

```
$sudo apt-get install --no-install-recommends libboost-all-dev
```

```
$sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev
```

2. Install Protobuf 3

Commands:

```
$pip3 install protobuf
```

3. Install Caffe

Commands:

```
$cd /home/xilinx
```

```
$git clone https://github.com/BVLC/caffe.git
```

Next is accessing into the Caffe directory and into “make” folder and the following can be run:

Commands:

```
$make all
```

```
$make test
```

```
$make runtest
```

After completing that, CD command can be used in order to exit from Caffe directory to the main directory. Next is to install Pycaffe:

Commands:

```
$Install pycaffe with Python
```

```
$cd python
```

```
$for req in $(cat requirements.txt); do sudo pip install $req; done
```

```
echo "export PYTHONPATH=$(pwd):$PYTHONPATH " >> ~/.bash_profile # to be able to call "import caffe" from Python after reboot
```

```
$source ~/.bash_profile # Update shell
```

```
$cd .
```

```
$export PYTHONPATH=/home/xilinx/caffe/python
```

4. Install Theano with Lasagne on PYNQ

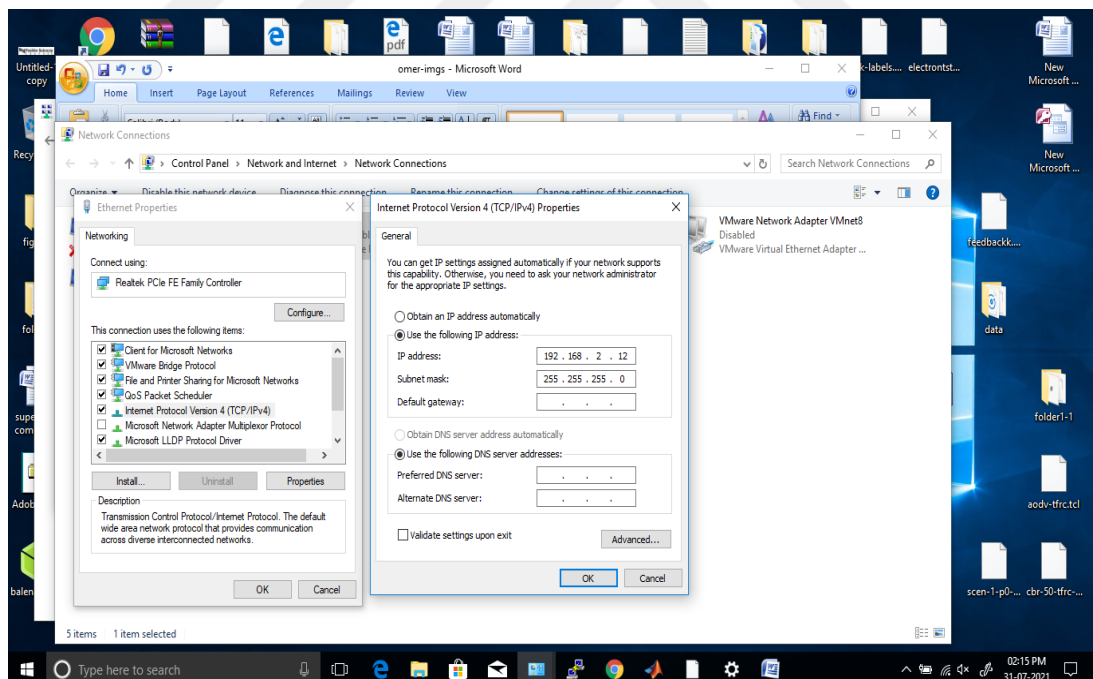
Commands:

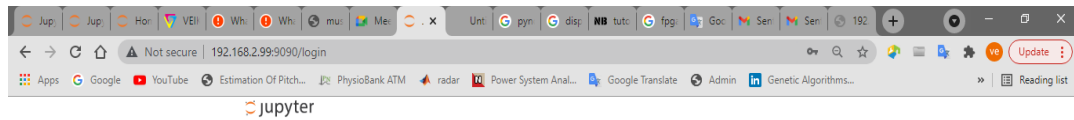
```
$pip=install-r
```

```
https://raw.githubusercontent.com/Lasagne/Lasagne/v0.1/requirements.txt
```

```
$pip install Lasagne==0.1
```

1. PUNQ-Z2 is providing Jupyter notebook where python programming can be conducted; firstly, all the database images (around 2156 image) must be uploaded to the notebook directory on the PYNQ-Z2 FPGA development board. Firstly, accessing the Jupyter notebook over the PYNQ-Z2 need to be performed after IP assignment. Static IP addressing to be given to the PYNQ-Z2 network card. The default IP address for PYNQ-Z2 development board is 192.168.2.99, however computer network card and this card to be linked together by assigning IP such as 192.168.2.14 to the PYNQ-Z2 development board, Figure below is demonstrating the IP addressing procedure.



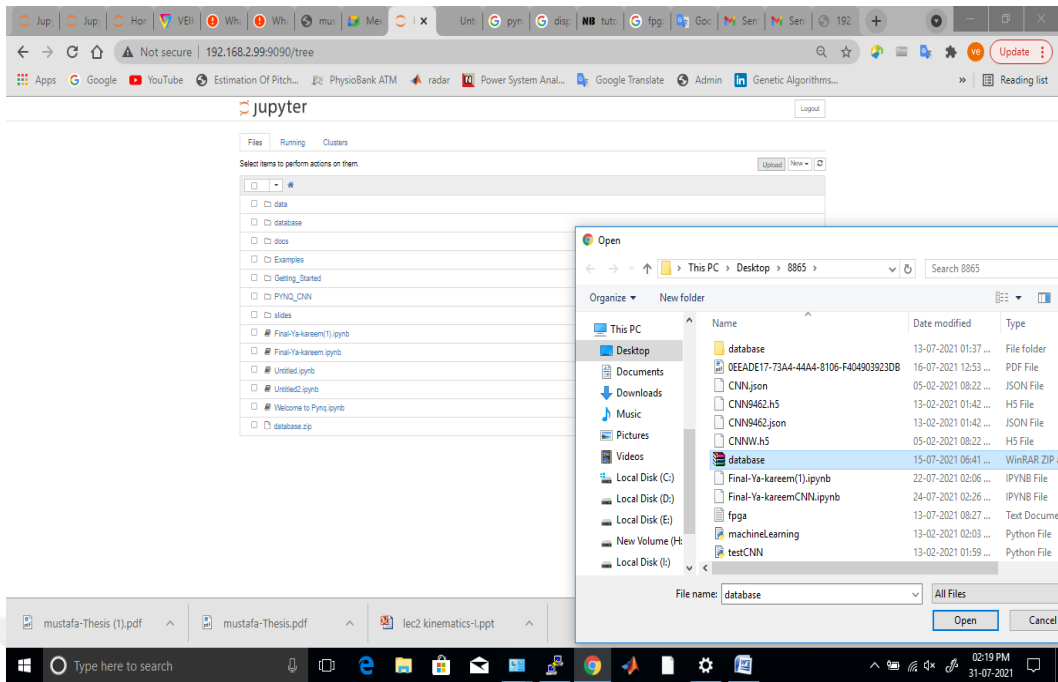


After assignment of the static IP to the development board, the last can be accessed by typing the same IP on any internet explorer application such as Google Chrome. Password of the PYNQ-Z2 will be the first thing that must enter into in order to open the jupyter notebook.

ANNEXES D. Uploading database file

Loading large number of images in jupyter notebook using method (upload file one by one) which takes unnecessary time and efforts, However, there are more efficient methods that can be employed, such as the following:

- 1.** compressing all the images into one archive file (.zip);
- 2.** uploading this archive file into the jupyter notebook;
- 3.** running of the following commands in the jupyter notebook itself in order to unarchive the uploaded file:



RESUME

Personal Information

Surname, name : Omar Mhmood ABDULHADI

Nationality : Iraq

Education

Degree	Education Unit	Graduation Date
Master	Electrical-electronic engineering	2021
Bachelor	Electronic engineering	2006/2007
High School	Al-motamezin school	2004

Work Experience

Year	Place	Title
2012-2013	Mosul-Iraq	engineering

Foreign Language

Arabic - English

Publications