A Comparative Study of Diabetic Retinopathy Detection using Machine Learning and Deep Learning Architectures

A thesis Report
Submitted in fulfillment of the requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

Submitted by

Silvia Sanjana	**170104142**
Nazmus Shakib Shadin	**170104119**
Md Sayeedur Rahman	**170104111**
Mallik Galib Shahriar	**160204079**

Supervised by

Dr.-Ing. Nusrat Jahan Lisa
Assistant Professor

Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Dhaka, Bangladesh

December 26, 2021

CANDIDATES' DECLARATION

We, hereby, declare that the thesis presented in this report is the outcome of the investigation performed by us under the supervision of Dr.-Ing. Nusrat Jahan Lisa, Assistant Professor, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE4100: Project and Thesis I and CSE4250: Project and Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

_____

Silvia Sanjana
ID: 170104142

_____

Nazmus Shakib Shadin
ID: 170104119

_____

Md Sayeedur Rahman
ID: 170104111

_____

Mallik Galib Shariar
ID: 160204079

CERTIFICATION

This thesis titled, "**A Comparative Study of Diabetic Retinopathy Detection using Machine Learning and Deep Learning Architectures**", submitted by the group as mentioned below has been accepted as satisfactory in fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in December, 2021.

Group Members:

Silvia Sanjana	**170104142**
Nazmus Shakib Shadin	**170104119**
Md Sayeedur Rahman	**170104111**
Mallik Galib Shahriar	**160204079**

_____

Dr.-Ing. Nusrat Jahan Lisa
Assistant Professor and Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

_____

Professor Dr. Mohammad Shafiul Alam
Professor and Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

First of all, we like to express our gratitude to Almighty Allah for His graces in keeping us well and healthy during the research study and in providing us with the capacity to accomplish this work.

Afterwards, we would like to convey our heartfelt appreciation to our thesis supervisor, "Dr.-Ing. Nusrat Jahan Lisa," for giving us with the opportunity of working with her, for her invaluable guidance at every level, and for her patience with us throughout our study work. This research would not have been accomplished without her vast knowledge, passion, new ideas, motivation, and determination. We would be eternally indebted to her for her guidance and unwavering support.

In addition, we would like to thank all the faculty and staff members of the Computer Science and Engineering Department for always being cordial and appreciative to us.

Following that, we are indebted to our families for their love and devotion, concern, efforts, and blessings, which enabled us to complete our degree path and consistently provided us with the necessary encouragement to accomplish this research. Additionally, we want to convey our gratitude to our friends for being there.

Silvia Sanjana
Nazmus Shakib Shadin
Md Sayeedur Rahman
Mallik Galib Shariar

Dhaka
December 26, 2021

ABSTRACT

The ailment of the eyes, Diabetic Retinopathy (DR) is on the rise around the world, causing diabetic patients to lose their vision and become partially or completely blind. The long-term health consequences associated with DR have a considerable impact on both the patient and economy, as the condition primarily affects young, productive persons. Early recognition and diagnosis can help individuals suffer less harm. This study proposes a computer-aided diagnosis system that is based on digital processing of retinal images and deep learning models to assist people in the early detection of diabetic retinopathy and also presents a comparative study on the performance of both machine learning and deep learning models. Fundus images of both healthy and affected retina are taken in this study. Here, image processing techniques such as extraction of the green channel, morphological operations, thresholding technique, contrast limited adaptive histogram equalization, top-hat transform, median filtering are used in machine learning models to detect defects in retinopathy affected images. In the deep learning portion, we have done auto-cropping, BGR to RGB conversion, Image Resizing, and Gaussian Blur as Image pre-processing techniques. Machine learning algorithms (Random Forest, Support vector machine, Decision tree) are used to classify these features and efficiently follow a cross-validation approach. In machine learning, we have used the Messidor-2 dataset which contains 1741 data images to detect diabetic retinopathy. In deep learning, we have also used a publicly available dataset named APTOS-2019 which contains 3662 data of retina images to train an ensemble of six deep Convolutional Neural Network models (CNN, VGG19, InceptionV3, Resnet50, Resnet50V2, Densenet169) to encode rich features and improve classification for different stages of DR. The experimental results show that, unlike current methods, the proposed model detects all stages of DR and a few of them outperforms the state-of-the-art methods.

Contents

ABSTRACT iv

List of Figures viii

List of Tables xii

1 Introduction 1
- 1.1 Motivation . . . 3
- 1.2 Objective . . . 3
- 1.3 Diabetic Retinopathy and it's Effect . . . 4
- 1.4 Challenges . . . 4
- 1.5 Types of Diabetic Retinopathy . . . 5
- 1.6 Data Collection . . . 5
- 1.7 Features of Diabetic Retinopathy . . . 7

2 Related Works 9

3 Background study 14
- 3.1 Image Processing Steps for Machine Learning . . . 14
  - 3.1.1 Green Channel . . . 14
  - 3.1.2 CLAHE (Contrast Limited Adaptive Histogram Equalization) . . . 14
  - 3.1.3 Morphological Operation . . . 14
  - 3.1.4 Shade Correction . . . 15
  - 3.1.5 Threshold . . . 15
- 3.2 Data Preprocessing for Machine Learning . . . 16
  - 3.2.1 Sampling . . . 16
- 3.3 Algorithms for Machine Learning . . . 17
  - 3.3.1 Random Forest . . . 17
  - 3.3.2 Decision Tree . . . 17
  - 3.3.3 Support Vector Machine (SVM) . . . 19
- 3.4 Learning Curves . . . 19
- 3.5 Data Image Pre-processing Steps for Deep Learning . . . 20
  - 3.5.1 Data Pre-Processing Technique for Deep Learning . . . 20

3.5.2 Data Augmentation for Deep Learning . . . 21
3.6 Deep learning Algorithms . . . 23
3.6.1 Convolutional Neural Network (CNN) Architecture . . . 23
3.6.2 VGG19 Architecture . . . 25
3.6.3 InceptionV3 Architecture . . . 27
3.6.4 ResNet50 Architecture . . . 29
3.6.5 ResNet50V2 Architectue . . . 31
3.6.6 DenseNet169 Architecture . . . 33

4 Proposed Method **35**
4.1 Proposed methodology for Machine Learning . . . 35
4.2 Proposed methodology for Deep Learning . . . 36

5 Implementation **39**
5.1 Feature extraction for Machine Learning . . . 39
5.1.1 Blood vessel extraction . . . 39
5.1.2 Exudate extraction . . . 40
5.1.3 Micro aneurysm extraction . . . 41
5.2 Feature scaling for Machine Learning . . . 41
5.3 Features for Machine Learning . . . 41
5.4 Feature selection for Machine Learning . . . 43
5.5 Implementation using Deep Learning . . . 44
5.5.1 Deep Learning based CNN Architecture . . . 44
5.5.2 Deep Learning based Transfer Learning Architecture . . . 45

6 Experimental Results **49**
6.1 Result Analysis for Machine Learning . . . 49
6.1.1 K-Fold cross validation: . . . 49
6.1.2 Accuracy for support vector machine . . . 49
6.1.3 Accuracy for Random Forest . . . 51
6.1.4 Accuracy for Decision tree classifier . . . 52
6.2 Result Analysis for Deep Learning . . . 53
6.3 Result analysis of Deep Learning Models in Binary Class Classification . . . 54
6.3.1 Result analysis for CNN Model in Binary Class Classification . . . 54
6.3.2 Result analysis for VGG19 in Binary Class Classification . . . 55
6.3.3 Result analysis for DenseNet169 in Binary Class Classification . . . 56
6.3.4 Result analysis for InceptionV3 in Binary Class Classification . . . 56
6.3.5 Result analysis for ResNet50 in Binary Class Classification . . . 57
6.3.6 Result analysis for ResNet50V2 in Binary Class Classification . . . 58
6.4 Result analysis of Deep Learning Models in Multi Class Classification . . . 59

6.4.1 Result analysis for CNN Model in Multi Class Classification 59
6.4.2 Result analysis for VGG19 in Multi Class Classification 60
6.4.3 Result analysis for DenseNet169 in Multi Class Classification 60
6.4.4 Result analysis for InceptionV3 in Multi Class Classification 61
6.4.5 Result analysis for ResNet50 in Multi Class Classification 62
6.4.6 Result analysis for ResNet50V2 in Multi Class Classification 63
6.5 Analysis and Comparison with DL Models in both Binary and Multi-Class Classification 64
6.5.1 Analysis and Comparison in Binary Class Classification 64
6.5.2 Analysis and Comparison in Multi Class Classification 65
6.6 Confusion Matrix for the Deep Learning Model 66
6.7 Prediction of Diabetic Retinopathy using Deep Learning Model 67

7 Outperforming State-of-the-Art Models **73**
7.1 Comparison on machine learning based models 73
7.2 Comparison on deep learning based models in binary class classification . 74
7.3 Comparison on deep learning based models in multi class classification . . 75
7.4 Analysis of the DL based outperforming models 75

8 Conclusion and Future Work **76**

References **77**

List of Figures

1.1 The fundus image of a normal eye vs. a Diabetic Retinopathy eye [1] 2
1.2 Diabetic Retinopathy Effect. 4
1.3 Types of Diabetic Retinopathy. 6
1.4 Visual Graph Representation of our used APTOS-2019 Dataset. 6
1.5 Visual Graph Representation of our used Messidor-2 Dataset. 7
1.6 Sample of Dataset's Retinal Images 7
1.7 Features of Diabetic Retinopathy. 8

3.1 Dataset before sampling 16
3.2 Dataset after sampling 16
3.3 How random forest algorithm works [2] 17
3.4 How Decision tree algorithm works [3] 18
3.5 Working flow of decision tree algorithm [3] 18
3.6 How Support vector machine algorithm works [4] 19
3.7 Sample Images of 5 DR Classes after applying the Pre-Processing Techniques 21
3.8 Basic Structure of Neural Network 24
3.9 Basic Structure of Convolutional Neural Network (CNN) 25
3.10 Structure of VGG19 Network 26
3.11 Basic Structure of InceptionV3 28
3.12 Basic Structure of ResNet50 30
3.13 Parametes for Various ResNet Architectures 31
3.14 Architecture diagram of ResNet50V2 32
3.15 ResNetV1 and ResNetV2 Variants 32
3.16 Difference between ResNet V1 and ResNet V2 33
3.17 DenseNet169 Architecture 33
3.18 DenseNet169 Parameters Architectures 34

4.1 Proposed methodology diagram using Machine Learning 35
4.2 Proposed methodology diagram using Deep Learning 36
4.3 Proposed methodology diagram using Deep Learning based CNN Model . . . 38
4.4 Proposed methodology diagram using Deep Learning based TL Model 38

5.1 Blood vessel extraction procedure 39

5.2 Exudate with optic disc . . . 40
5.3 optic disc extraction procedure . . . 40
5.4 Exdate detection . . . 40
5.5 Micro aneurysm extraction procedure . . . 41
5.6 Co relation among the features(Pearson correlation) . . . 43
5.7 Implemented CNN Model . . . 44
5.8 CNN Training Parameters . . . 45
5.9 Transfer Learning Implemented architecture . . . 46
5.10 VGG19 Training Parameters . . . 46
5.11 DenseNet169 Training Parameters . . . 47
5.12 InceptionV3 Training Parameters . . . 47
5.13 ResNet50 Training Parameters . . . 48
5.14 ResNet50V2 Training Parameters . . . 48

6.1 Learning curve for binary class classification . . . 50
6.2 Learning curve of for multiclass class classification . . . 50
6.3 Confussion matrix for binary class classification . . . 50
6.4 Confussion matrix for multiclass class classification . . . 50
6.5 Learning curve for binary class classification . . . 51
6.6 Learning curve of for multiclass class classification . . . 51
6.7 Confussion matrix for binary class classification . . . 51
6.8 Confussion matrix for multiclass class classification . . . 51
6.9 Learning curve for binary class classification . . . 52
6.10 Learning curve of for multiclass classification . . . 52
6.11 Confussion matrix for binary class classification . . . 53
6.12 Confussion matrix for multiclass class classification . . . 53
6.13 Training and Validation model loss for the CNN architecture in different epochs 55
6.14 Training and Validation model accuracy for the CNN architecture in different epochs . . . 55
6.15 Training and Validation model auc for the CNN Architecture in different epochs 56
6.16 Training and Validation model loss for the VGG19 architecture in different epochs . . . 56
6.17 Training and Validation model accuracy for the VGG19 architecture in different epochs . . . 57
6.18 Training and Validation model auc for the VGG19 Architecture in different epochs . . . 57
6.19 Training and Validation model loss for the DenseNet169 architecture in different epochs . . . 58

6.20 Training and Validation model accuracy for the DenseNet169 architecture in different epochs . . . 58
6.21 Training and Validation model auc for the DenseNet169 Architecture in different epochs . . . 59
6.22 Training and Validation model loss for the InceptionV3 architecture in different epochs . . . 59
6.23 Training and Validation model accuracy for the InceptionV3 architecture in different epochs . . . 60
6.24 Training and Validation model auc for the InceptionV3 Architecture in different epochs . . . 60
6.25 Training and Validation model loss for the ResNet50 architecture in different epochs . . . 61
6.26 Training and Validation model accuracy for the ResNet50 architecture in different epochs . . . 61
6.27 Training and Validation model auc for the ResNet50 Architecture in different epochs . . . 62
6.28 Training and Validation model loss for the ResNet50V2 architecture in different epochs . . . 62
6.29 Training and Validation model accuracy for the ResNet50V2 architecture in different epochs . . . 63
6.30 Training and Validation model auc for the ResNet50V2 Architecture in different epochs . . . 63
6.31 Training and Validation model loss for the CNN architecture in different epochs 64
6.32 Training and Validation model accuracy for the CNN architecture in different epochs . . . 64
6.33 Training and Validation model auc for the CNN Architecture in different epochs 65
6.34 Training and Validation model loss for the VGG19 architecture in different epochs . . . 65
6.35 Training and Validation model accuracy for the VGG19 architecture in different epochs . . . 66
6.36 Training and Validation model auc for the VGG19 Architecture in different epochs . . . 66
6.37 Training and Validation model loss for the DenseNet169 architecture in different epochs . . . 67
6.38 Training and Validation model accuracy for the DenseNet169 architecture in different epochs . . . 67
6.39 Training and Validation model auc for the DenseNet169 Architecture in different epochs . . . 68

6.40 Training and Validation model loss for the InceptionV3 architecture in different epochs . . . 68
6.41 Training and Validation model accuracy for the InceptionV3 architecture in different epochs . . . 68
6.42 Training and Validation model auc for the InceptionV3 Architecture in different epochs . . . 69
6.43 Training and Validation model loss for the ResNet50 architecture in different epochs . . . 69
6.44 Training and Validation model accuracy for the ResNet50 architecture in different epochs . . . 69
6.45 Training and Validation model auc for the ResNet50 Architecture in different epochs . . . 70
6.46 Training and Validation model loss for the ResNet50V2 architecture in different epochs . . . 70
6.47 Training and Validation model accuracy for the ResNet50V2 architecture in different epochs . . . 70
6.48 Training and Validation model auc for the ResNet50V2 Architecture in different epochs . . . 71
6.49 Confusion Matrix for DR detection using DL Model . . . 72
6.50 Prediction of DR detection for Single Class (Class 4: PDR) . . . 72

7.1 Comparison of proposed model with existing models in terms of Average recall 74

List of Tables

1.1 Types of Diabetic Retinopathy and their effects in the eyes 5

2.1 Summary of the works related to Diabetic Retinopathy Detection related to DL 13

3.1 Parameters for Data Augmentation 22

6.1 Performance metrics of SVM for binary class classification 50
6.2 Performance metrics of SVM for multiclass class classification 51
6.3 Performance metrics of Random Forest classifier for binary class classification 52
6.4 Performance metrics of Random Forest classifier for multi class classification 52
6.5 Performance metrics of Decision tree classifier for binary class classification . 53
6.6 Performance metrics of Decision tree classifier for multi class classification . . 53
6.7 Demo Confusion Matrix 54
6.8 Result Comparison and analysis for DL Models in Binary Class Classification . 71
6.9 Result Comparison and analysis for DL Models in Multi Class Classification . 71

7.1 Comparison with the existing techniques 73
7.2 Comparison with the existing models for Binary Class Classification in DL Models 74
7.3 Comparison with the existing models for Multi Class Classification in DL Models 75

Chapter 1

Introduction

Diabetic retinopathy (DR) is an eye disease originated by the difficulties of diabetes. It is a leading cause of vision impairment and one of the most common diabetic eye diseases. It is caused by injury to blood vessels and tissue which are light-sensitive in the retina [5]. DR can strike anyone with either type 1 or type 2 diabetes. The longer anyone has diabetes and the less well his blood sugar is controlled, the more likely this eye problem will develop. Fluid leaks from blood vessels into the retina cause damage to the retina. This tiny blood vessel will leak blood and fluid on the retina, forming features such as hemorrhages, micro-aneurysms, hard exudates, venous loops or cotton wool spots [6]. DR may present with no symptoms or only minor sight problems at first. It can eventually lead to blindness. As a result, early detection is crucial in order to protect a patient from permanent vision loss.

Until now, ophthalmologists have manually screened DR, which is a time-consuming procedure. Many undiagnosed and uncontrolled diabetic patients become blind as a result of a lack of proper screening and treatment facilities, solely at the primary and secondary care level. Diagnostic and treatment options are limited to urban tertiary care centers, which cannot fill the demands of the entire populace. Another stumbling block in the path to DR diagnosis is a scarcity of competent ophthalmologists. Within the recent past computer-aided diagnosis (CAD) systems are likely to influence a meticulous, consistent, well-founded, and timely DR diagnosis.

However, Fundus images are widely used to detect DR through automated system, and Ophthalmologists can use these to replicate their findings. In the literature, several machine learning (ML) and deep learning (DL)-based techniques for identifying DR have been presented. Machine learning models take the fundus eye images as input, and then image preprocessing is done. After that, the features of the fundus ocular image are retrieved, and a classification model is created using machine learning techniques. The categorization

model is then assessed. Deep learning is a multilayered extension of machine learning for extraction of features. The dataset for DL architecture's training and testing is initially acquired. It is then exposed to a number of preprocessing techniques in order for the network to learn the features from a better image. The preprocessed image is delivered to the DL architecture for feature extraction and classification. The layers of the DL architecture accept the output from the previous layer as input, process it, and then pass it on to the next layer. Finally, the last layer generates the end result. These can be used by doctors to back up their findings.

Over time, various ML methods have been applied to perform DR diagnosis. Machine learning classifying algorithms can be used to classify DR features extracted from the output of various retinal fundus images. In this proposal ML classifiers have been tested on (Messidor) dataset. Total 500 images are trained and tested in which 250 images are normal images and other 250 images have severe retinopathy. Decision making for predicting the presence of diabetic retinopathy is performed using Support Vector Machine,Random Forest, and Decision tree.

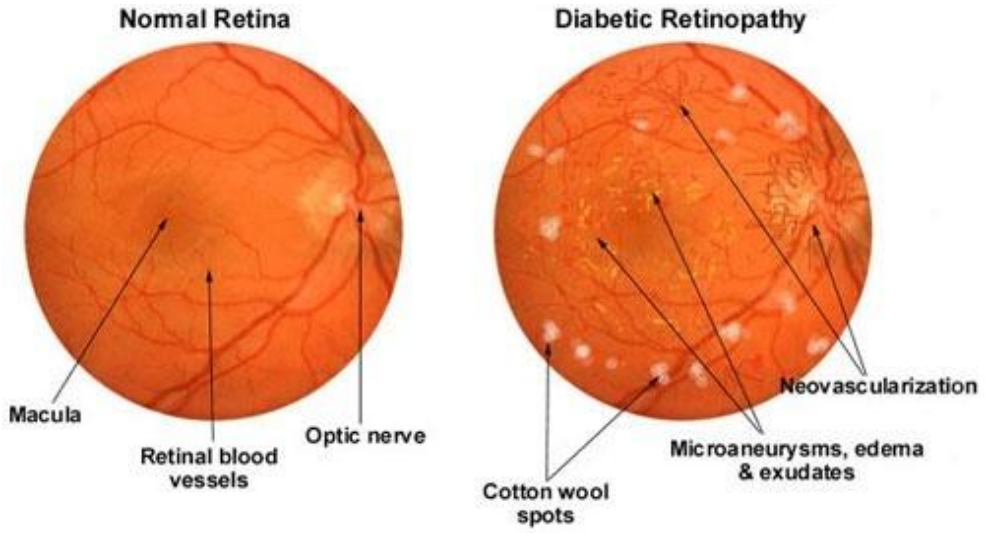


Figure 1.1: The fundus image of a normal eye vs. a Diabetic Retinopathy eye [1]

Here, Figure 1.1 demonstrates the difference between a healthy eye and an eye with Diabetic Retinopathy. Different deep learning methods had been used to detect DR with higher accuracy over the time period. Deep learning models can predict DR more precisely. In this proposal, we have proposed six deep learning based models, those are: CNN, VGG19, InceptionV3, Resnet50, Resnet50V2, Densenet169 . We used a binary class classification and multi class classification system to classify Diabetic Retinopathy (0: No DR, 1: DR) and (0: No DR, 1: Mild DR, 2: Moderate DR, 3: Severe DR, 4: PDR) on APTOS-2019 dataset. Our models have outperformed the existing state of the art in a number of ways.

However, a sufficient number of experiments were conducted to compare the performance of the above-mentioned machine learning and deep learning models. The latest state-of-the-art methods for detecting and classifying DR color fundus images using different techniques have been reviewed and analyzed for this article. In addition, the color fundus retina datasets available on DR have been examined.

The following is a summary of the paper's structure. Section 2 summarizes research findings on the diagnosis of diabetic retinopathy. Section 3 describes the study in depth, while Section 4 analyzes the outcomes. In Section 5, we addressed our research and its major findings, and also an evaluation of our current work for further exploration.

1.1 Motivation

Diabetic retinopathy affects an increasing number of people every day. By 2030, the population is assumed to increase from 126.6 million to 191.0 million. If no action is taken, the number of people with vision-threatening diabetic retinopathy (VTDR) will lift from 37.3 million to 56.3 million. It has been largely ignored in health care and in many backward countries due to insufficient medical services [6]. Because there are currently insufficient methods for detecting diabetic retinopathy, we will develop a system that will predict diabetic retinopathy. As a result, we chose to use Machine Learning and Deep Learning models to predict this disease.

1.2 Objective

Our main goal is to predict diabetic retinopathy and compare the performance of various machine learning and deep learning models for predicting diabetic retinopathy. Machine learning algorithms like Random forest, Support vector machine, logistic regression and Decision tree can be trained with a training dataset and can predict data based on that dataset. We also trained six deep Convolutional Neural Network models (CNN, VGG19, InceptionV3, Resnet50, Resnet50V2, Densenet169) to improve classification and encode the features for different stages of DR using a publicly available retina image dataset. Diabetic Retinopathy can cause vision loss and deterioration that is permanent. It cannot be treated effectively unless it is detected properly and in a timely manner. Our goal is to use an automated system to detect DR so that people of third-world countries can receive proper

treatment.

1.3 Diabetic Retinopathy and it's Effect

Diabetic retinopathy (DR) is a diabetes complication that affects eyes. It's caused by damage to the blood vessels of the light-sensitive tissue at the back of the eye (retina). Early detection and treatment are keys to preventing vision loss from Diabetic retinopathy (DR).

Here in Figure 1.2 we can see the effects of diabetic retinopathy in the eyes. The Normal eye vs. the eye effected with the diabetic retinopathy. How the eye is different from the normal eye to the diabetic retinopathay effected eye.

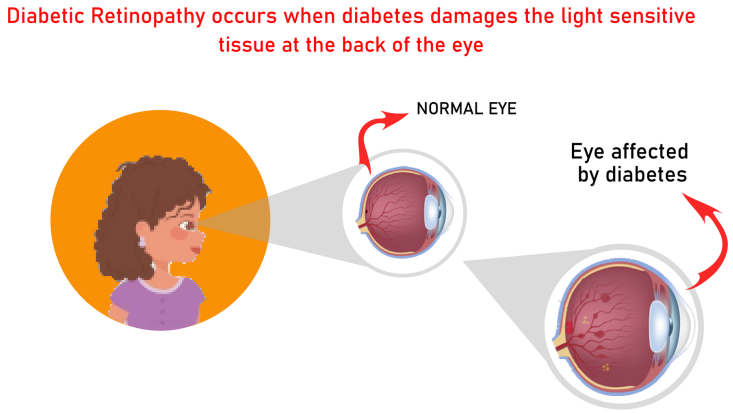


Figure 1.2: Diabetic Retinopathy Effect.

1.4 Challenges

The main challenge in conducting this research is the lack of a standard dataset. In our region, standard open datasets are rare. Another issue is non-representative data. To generalize well,The training data should be indicative of new instances, i.e., the data we use for training should include all cases that have happened and will happen in the future. The trained model is unlikely to make accurate predictions if the training set is non-representative. The data we gathered may not be suitable for training; some samples differ from others in terms of outliers or missing values, for example. Another issue in deep learning models is overfitting and underfitting the training data, both of which can lead to incorrect decisions.

1.5 Types of Diabetic Retinopathy

There are 5 types of Diabetic Retinopathy:
1. No Diabetic Retionopathy (No DR) [0]
2. Mild Diabetic Retionopathy [1]
3. Moderate Diabetic Retionopathy [2]
4. Severe Diabetic Retionopathy [3]
5. Proliferative Diabetic Retionopathy (PDR) [4]
Let's see the types of diabetic of and their effect in the eyes in the table 1.1.

Table 1.1: Types of Diabetic Retinopathy and their effects in the eyes

Level of severity	***Features***
No DR	No abnormalities
Mild DR	Only microaneurysms
Moderate DR	Less than severe NPDR more than microaneurysms
Severe DR	A lot of hemorrhage and microaneurysms are observed in 4 quadrants of the retina Venous beading in two or more quadrants Intraretinal microvascular abnormalities in at least one quadrant
Proliferative DR	One or both can be found among Neovascularization and pre-retinal/vitreous hemorrhage

Here, in our work we have done the binary class classification of diabetic retinopathy that means healthy eyes (0), and diabetic retinopathy (1-4).

As well as we have done the multi class classification of diabetic retinopathy that meas all the 5 types of diabtic retinopathy is detected separately.

Let's visualize the diabetic retinopathy classes with a figure. The types of Diabetic is given in figure 1.3.

1.6 Data Collection

For detecting Diabetic Retinopathy (DR), we have collected all our data images from multiple datasets which is APTOS-2019 (Asia Pacific Tele-Ophthalmology Society-2019) [7] and Messidor-2 [8].

1. APTOS-2019: There are total number of 3662 data images in APTOS-2019. Among them, 1805 images with No DR, 370 images with Mild DR, 999 Images with Moderate DR, 193 images with Severe DR, and 295 images with Proliferative DR. The visual graph representation of our used APTOS-2019 dataset is given below in Figure 1.4.

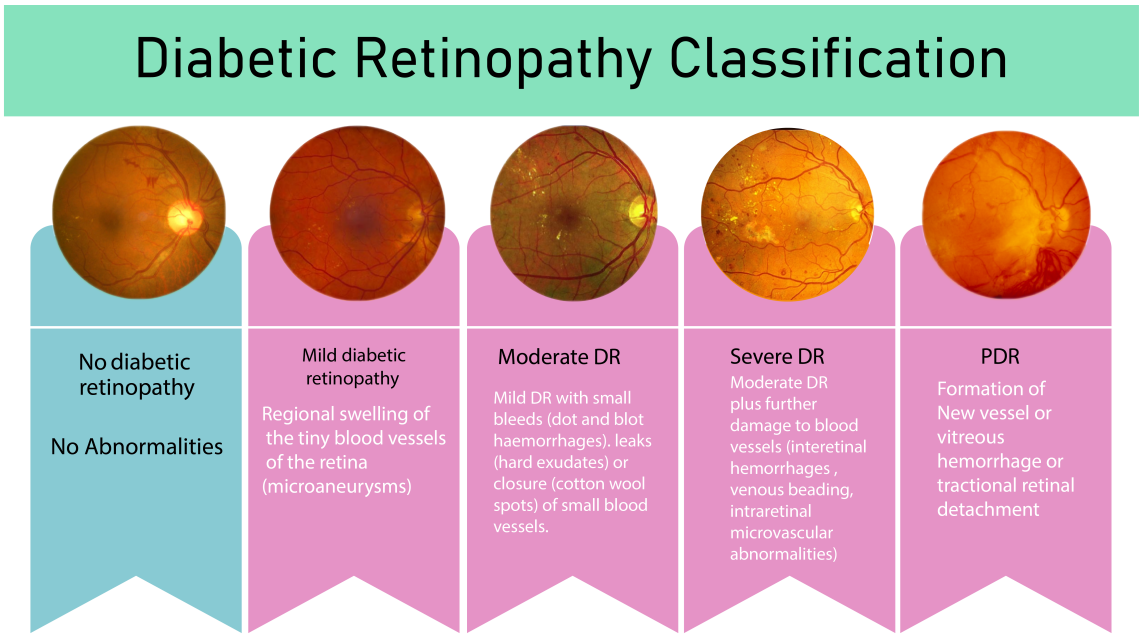


Figure 1.3: Types of Diabetic Retinopathy.

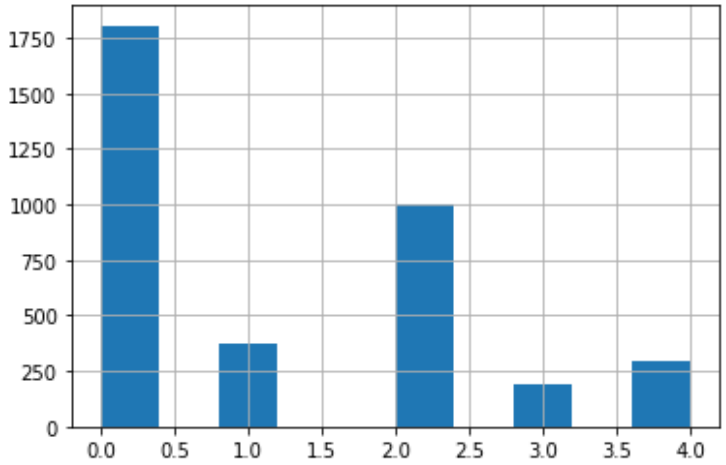


Figure 1.4: Visual Graph Representation of our used APTOS-2019 Dataset.

2. Messidor-2: This dataset contains 1741 images in total, taken with Topcon TRC NW6 non-mydriatic fundus camera having 45 degrees FOV. Data are Leveled from 0-3. The visual graph representation of our used Messidor-2 dataset is given below in Figure 1.5.

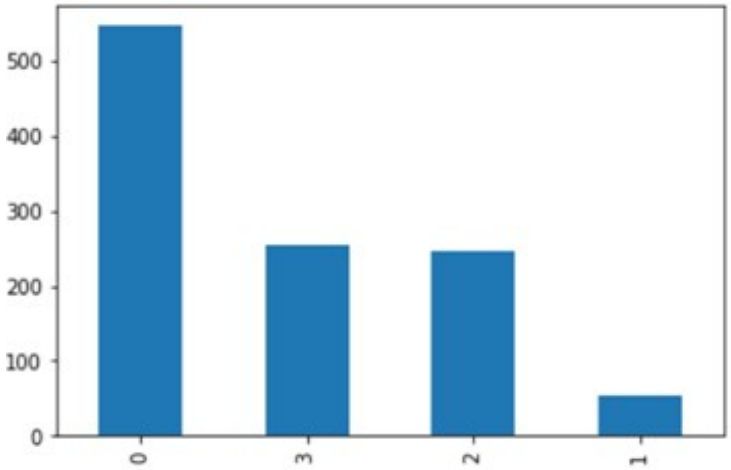


Figure 1.5: Visual Graph Representation of our used Messidor-2 Dataset.

Test Train Split:
We have 80% of our data images are used as the training data from this dataset, and the rest of the 20% of data images are used as testing data.

(a) Normal Eye (b) Eye with Diabetic Retinopathy

Figure 1.6: Sample of Dataset's Retinal Images

1.7 Features of Diabetic Retinopathy

There are total number of five features in Diabetic Retinopathy effected eyes. These are:
1. Retinal arterial macroaneurysm
2. Cotton Wool Spots

3. Blood Vessels

4. Hemorrhages

5. Hard Exudates

Now let's see the visual representation of these features in figure 1.7.

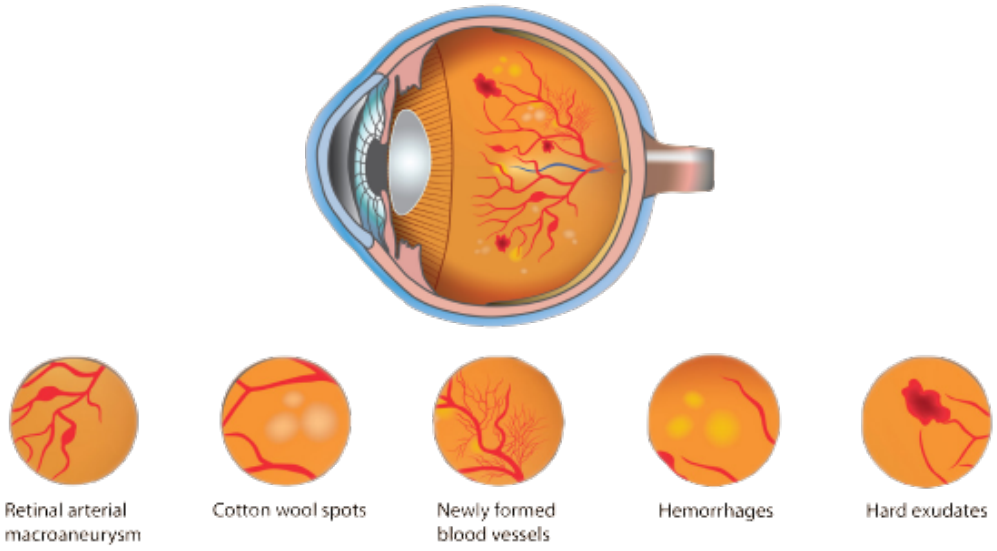


Figure 1.7: Features of Diabetic Retinopathy.

Chapter 2

Related Works

Diabetic Retinopathy has been the subject of extensive research in order to detect it accurately. In [9] a computer assisted diagnosis based on the digital processing of retinal images is proposed by Enrique V. Carrera et al. They isolated blood vessels, micro aneurysms and hard exudates and classified diabetic retinopathy by fitting these features into SVM classifiers. Several image processing techniques were used to isolate these features. They took 400 images in which 152 images without retinopathy (grade 0), 30 with mild NPDR (grade 1), 69 images with moderate NPDR (grade 2) and 149 images with severe NPDR (grade 3) from Messidor database that consists of 1200 retinal images. Their work has two subsections (NPDR detection and NPDR grade detection). In the NPDR detection step they achieved 92.4% accuracy using the SVM classifier in which they used 301 retinal images, 152 images with grade 0 and 149 images with grade 3. In the NPDR grade classification step they used 400 retinal images and they got 80% accuracy using the SVM classifier.

In [10] image pre-processing techniques and morphological operations are applied to detect statistical features. Five statistical features (Entropy, energy, standard deviation, mean, variance and contrast) are extracted after image pre-processing and blood vessel extraction. Histogram-based features are extracted by using Discrete Wavelet Transform (DWT). These features are classified by machine learning algorithms (KNN, SVM, Artificial neural network). SVM, KNN and ANN showed 78%, 74% and 80.5% accuracy using statistical features, On the other hand they showed 69%, 83% and 86% accuracy while using Histogram feature of DWT image.

In [11] two well-known feature extraction techniques scale invariant feature (SIFT) and Speedup robust features (SURF) were used to capture exudate regions. Image resizing, Green channel Extraction, Adaptive histogram Equalization and Edge enhancement techniques were used in the image preprocessing step. Each exudates region was stored in a feature matrix and the Support vector machine classifier used the feature matrix for the prediction of DR. SVM classifier gave 98% average specificity with 94% sensitivity. The al-

gorithm was tested on 100 images in which 50 images were normal and other 50 images were retinopathy affected.

In [12] they proposed a method for extracting blood vessels from the retinal fundus image. Image preprocessing techniques such as morphological operation, contour border technique, thresholding are used for this purpose.

Non-perfusion regions (NP), microaneurysms, leakages, and laser scars were among the four types of DR lesions annotated in a total of 4067 FFA images in [13]. Three CNNs, DenseNet, ResNet50, and VGG16, were trained to achieve multi-label classification, which implies the algorithms could recognize four retinal lesions at once. DenseNet's area under the curve (AUC) for detecting NP, microaneurysms, leakages, and laser scars were 0.8703, 0.9435, 0.9647, and 0.9653 respectively. AUC for NP was 0.8140, 0.9097 for microaneurysms, 0.9585 for leakages, and 0.9115 for laser scars in ResNet50. The AUC for NP was 0.7125, 0.5569 for microaneurysms, 0.9177 for leakages, and 0.8537 for laser scars for VGG16.

In [14], the (APTOS) 2019 dataset was used to train and test the DL models. Res-Net18, AlexNet, GoogleNet, VGG16, SqueezeNet, and VGG19 were the deep transfer models that were used in this study. There are only a few layers in these models compared to larger models like DenseNet and InceptionResNet, hence they were chosen for use in this study In order to strengthen the models and avoid the overfitting problem, we employed data augmentation approaches. The AlexNet model was the most accurate in testing, with a 97.9c success rate. In addition, our outcomes were bolstered by the success indicators that we attained.

For the identification and classification of diabetic retinopathy, a reformed capsule network [15] has been devised. The features from the fundus images are retrieved using the convolution and primary capsule layers, and the chances that the image belongs to a given class are evaluated using the class capsule layer and softmax layer. The Messidor dataset is used to validate the effectiveness of the proposed reformed network on four performance measures. On healthy retina, stage 1, stage 2, and stage 3 fundus pictures, the built capsule network achieves the accuracy of 97.98 %, 97.65 %, 97.65 %, and, 98.64 % respectively.

Hattiya et al [16] proposed an automated technique for DR screening. The objective was to develop a classifier capable of discriminating between DR and non-DR retinal images. The core concept is to segment retinal images in order to obtain a region of interest (ROI) that is consistent with the classification procedure. The data for the evaluation came from a Kaggle of 23,513 images. The best results were obtained using the AlexNet learning mechanism, which achieved an accuracy of 98.42 percent for the training set and 81.32 percent for the testing set, respectively.

Zhang et al [17] have presented the DeepDR system, an autonomous DR identification, and grading system. DeepDR employed transfer learning and ensemble learning to detect the

existence and extent of DR directly from fundus images. It is constructed using a combination of well-known convolutional neural networks and specialized conventional deep neural networks. It investigates the association between the frequency of ideal component classifiers and the number of class labels, and also the effect of various element classifier combinations on the best integration performance while building an optimal model. They used nine measures to assess the models' validity and dependability. The results indicate that the identification model operates optimally at 97.5% sensitivity, 97.7 % specificity, and 97.7 % area under the curve. However, the grading model obtains a 98.1% sensitivity and a 98.9% specificity.

The purpose of [18] is to use Convolution Neural Networks to systematize the diagnosis and classification of non-proliferative Diabetic Retinopathy from fundus images. The model was validated using two well-known datasets, MESSIDOR and IDRiD. The images were preprocessed and were modified prior to deploying the Convolution Neural Network (CNN) layers. The highest accuracy obtained with MESSIDOR images is 90.89%.

The performance of multiple extremely efficient and robust CNN architectures for Diabetic Retinopathy classification using Transfer Learning is examined in [19]. The study focused on models such as VGG16, Resnet50 V2, and EfficientNet B0. Multiple performance metrics are used to evaluate categorization performance, including True Positive Rate, False Positive Rate, and Accuracy. Additionally, many performance graphs are displayed to illustrate the architecture's efficiency, including the Confusion Matrix and the ROC Curve. The results reveal that Transfer Learning utilizing ImageNet weights and the VGG 16 model achieves the highest classification accuracy of 95%. It is accompanied by the ResNet50 V2 design, which has a 93% accuracy.

Mushtaq et al's [20] primary objective is to develop a systematic approach for accurately identifying DR. It perceives a deep learning methodology, specifically a Densely Connected Convolutional Network DenseNet-169, that is used to diagnose diabetic retinopathy earlier. It categorizes fundus images according to their severity as No DR, Mild, Moderate, Severe, or Proliferative DR. The datasets used are Diabetic Retinopathy Detection 2015 and Aptos 2019 Blindness Detection, both of which were acquired from Kaggle. The proposed method collects, preprocesses, augments, and models data. Our proposed model was 90% accurate.

In [21], to categorize DR image data into five categories, authors utilized two pretrained deep learning CNN models, ResNet50 and Xception, each with a unique optimizer and activation function. Both frameworks achieved an accuracy of 83% when tested on actual data. The proposed deep learning approach can also be used to categorize diabetic retinopathy phases based on fundus images.

To evaluate and optimize the model, a 10 fold cross-validation technique was used by Li et al [22]. Additionally, they chose the Messidor-2 dataset to evaluate the effectiveness of

the algorithm. Moreover, they evaluated accuracy, specificity, sensitivity, AUC, and value for discriminating between no referral (no apparent DR and mild NPDR) and referral (moderate NPDR, severe NPDR, and PDR). The suggested technique attained a classification accuracy of 93.49%, a sensitivity of 96.93%, and a specificity of 93.45%, with an AUC of up to 0.9905.

Vijayan et al [23] suggested employing deep learning (DL) techniques to identify Diabetic Retinopathy utilizing fundus images. The suggested fine-tunedVGG19 CNN architecture outperforms the Kaggle data set and is capable of solving this multi-class categorization issue successfully. The proposed model makes use of pre-trained weights from image net data, which reduces training time and enhances performance in terms of sensitivity, specificity, and accuracy in identifying DR from retinal images. Deep transfer learning was used in conjunction with technique fine-tuning to achieve the best test accuracy of 73.60%.

An 18-layer deep convolutional neural network (CNN) with three fully connected layers is suggested in [24] to evaluate image data and instantaneously differentiate between no DR, moderate and severe DR with a validation accuracy of 88-90%, a sensitivity of 87%-80%. A preprocessing stage included image scaling and data augmentation specific to a class of data.

In Table 2.1, we can see a summary of the linked research in the detection of Diabetic Retinopathy. This table 2.1 provides an overview of related works on the detection of diabetic retinopathy.

Table 2.1: Summary of the works related to Diabetic Retinopathy Detection related to DL

Ref	Dataset	Used Models	Performance Measure				
			Accuracy	Sensitivity	Specificity	ROC	AUC
[13]	Eye Center at the Second Affiliated Hospital of Zhejiang University School of Medicine	DenseNet, ResNet50, VGG16					0.8703 for NP, 0.9435 for Microaneurysms, 0.9647 for leakages, 0.9653 for laser scars (DenseNet), and 0.8140 for NP, 0.9097 for Microaneurysms 0.9585 for leakages, 0.9115 for laser scars (ResNet50) 0.7125 for NP, 0.5569 for Microaneurysms, 0.9177 for leakages, 0.8537 for laser scars (VGG16)
[14]	APTOS 2019	AlexNet, Squeezenet, GoogleNet, VGG16, Res-Net18, VGG19	97.9, 90.3, 96.3, 96.8, 97.8, 97.4				
[15]	Messidor	CNN	97.98 (stage1), 97.65 (stage2), 97.65 (stage3), and 98.64 (stage4)				
[16]	Kaggle (23,513 images)	AlexNet, ResNet50, DenseNet201, InceptionV3, MobileNet, NasNet, NASNetMobile	81.32, 67.04, 50.0, 58.69, 66.48, 71.18, 70.70				
[17]	Own dataset containing 13767 images	CNN (ResNet50, InceptionResNetV2, InceptionV3, Xception and DenseNets)	96.5	98.1	98.9		
[18]	MESSIDOR, IDRiD	CNN layers	90.89, 90.29	88.75, 88.75	96.30 , 96.89		
[19]	APTOS 2019	VGG16, Resnet50 V2,EfficientNet B0	95, 93, 96				
[20]	Diabetic Retinopathy Detection 2015, Aptos 2019 Blindness	DenseNet-169	90				
[21]	APTOS 2019	ResNet50,Xception	83, 83				
[22]	Messidor-2	Inception-V3	93.49	96.93	93.45		0.9905
[23]	Kaggle data set	VGG19	73.60				
[24]	(APTOS) 2019	Deep (CNN)	88-89	87-89	94-95		

Chapter 3

Background study

3.1 Image Processing Steps for Machine Learning

3.1.1 Green Channel

The combination of red,green and blue channel builds a RGB image. Blue channel holds low contrast and doesn't hold much information. In the red channel blood vessels are visible but it has too much noise and it is saturated. On the other hand, the green channel holds much information as well as it has better contrast. In the green channel blood vessels are darker and the background is bright. So we choose the green component of the image for the detection process.

3.1.2 CLAHE (Contrast Limited Adaptive Histogram Equalization)

We use CLAHE to enhance the contrast of the image. CLAHE operates on a small region of the image rather than the whole image. Contrast of small regions are enhanced with histogram equalization so images are equally illuminated [25]. Dark details such as blood vessels micro aneurysms are much darker after CLAHE operation.

3.1.3 Morphological Operation

Dilation:

This process can be applied on binary images as well as grayscale images. It increases the boundaries of regions of foreground pixels (White pixels). The areas of foreground pixels grow on the other hand holes within the regions become smaller. The areas that are smaller

than the structuring element and separated by space are joined using a dilation process. It is called the hole filling procedure [26].

Opening:

Opening process is the process in which dilation is followed by the erosion operation. Internal noises can be removed using opening operations [27].

3.1.4 Shade Correction

Due to photographic process and chordial fluorescence in an image unwanted changes in intensity can occur and shade correction process can help to remove the unwanted changes in intensity [28].

3.1.5 Threshold

Threshold is a technique in which assignment of the pixel value is determined by the provided threshold value. Each pixel value is compared with the provided threshold value. If the pixel value is less than the threshold value, 0 will be assigned to the pixel otherwise 1 will be set as pixel value to the pixel. Threshold produces a binary image. Thresholding process is done on grayscale image so before thresholding operation the image should be converted to the grayscale image.

3.2 Data Preprocessing for Machine Learning

3.2.1 Sampling

A dataset with an unequal distribution of classes is referred to as imbalanced classification. On imbalanced classification datasets, standard machine learning approaches do not perform well, which is a big concern.

This implies that the number of samples in the training dataset corresponding to each class varies significantly. A substantial skew in the class distribution, such as a 1:10, 1:1000, or even 1:1000 ratio of instances in the minority class to those in the majority class, is not uncommon.

Many machine learning methods are built to deal with data that has an equal number of observations in each class. When this isn't the case, algorithms can learn to ignore a tiny amount of input in order to give satisfactory results. A set of techniques for modifying a training dataset so that the class distribution is balanced or more balanced is known as data sampling. Following the balancing of the dataset, standard machine learning algorithms may be trained on it without any modifications. Even when class distributions are extremely unequal, a data preparation method can be utilized to manage the problem of imbalanced categorization. As because our dataset has imbalance property, we over sampled our dataset. For the sampling purpose we used synthetic minority oversampling technique.

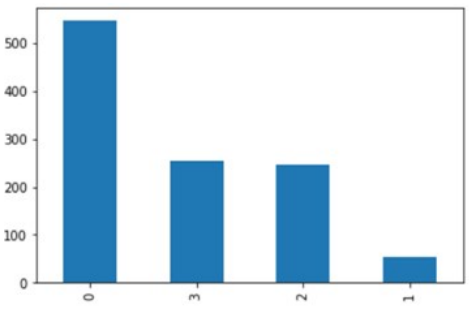


Figure 3.1: Dataset before sampling

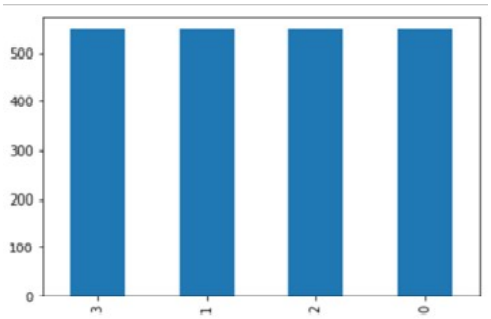


Figure 3.2: Dataset after sampling

3.3 Algorithms for Machine Learning

3.3.1 Random Forest

In classification related problems random forest algorithms can be used. It is a supervised classification algorithm which contains various numbers of decision trees. Usually the number of decision trees defines the robustness of the forest. In general, higher the number of decision trees in a forest gives better accuracy. Random forest classifiers have many advantages. Because it takes the average of all the predictions it doesn't suffer from overfitting problems. It can handle missing values. This algorithm can be used for classification and regression problems. Technically it is an ensemble learning method. Attribute selection indicators such as gain, entropy, GINI index for each attribute are used to generate the trees. This model counts the votes of each tree and takes the most popular class as the result. It is more accurate and powerful compared to the other non-linear classification algorithms [2].

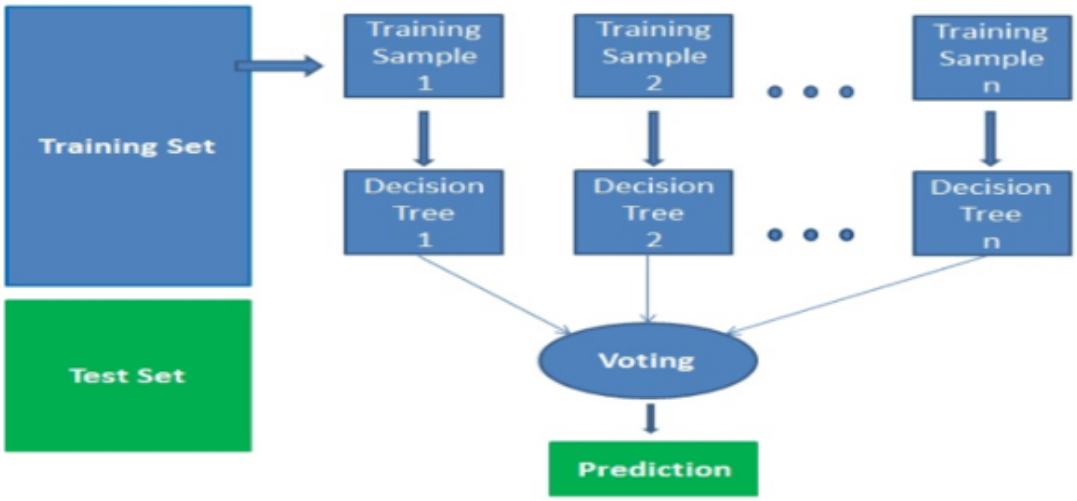


Figure 3.3: How random forest algorithm works [2]

3.3.2 Decision Tree

In the decision tree each internal node represents features (attributes), each branch represents the decision rule and each leaf node represents the outcome. Topmost internal node is called the root node.

It's training time is faster compared to the neural network algorithm. Number of records and the number of attributes determines the time complexity of the decision tree algorithm. Decision tree algorithms can give better accuracy when the data are high dimensional. The decision tree algorithm does not depend upon probability distribution assumptions, that's

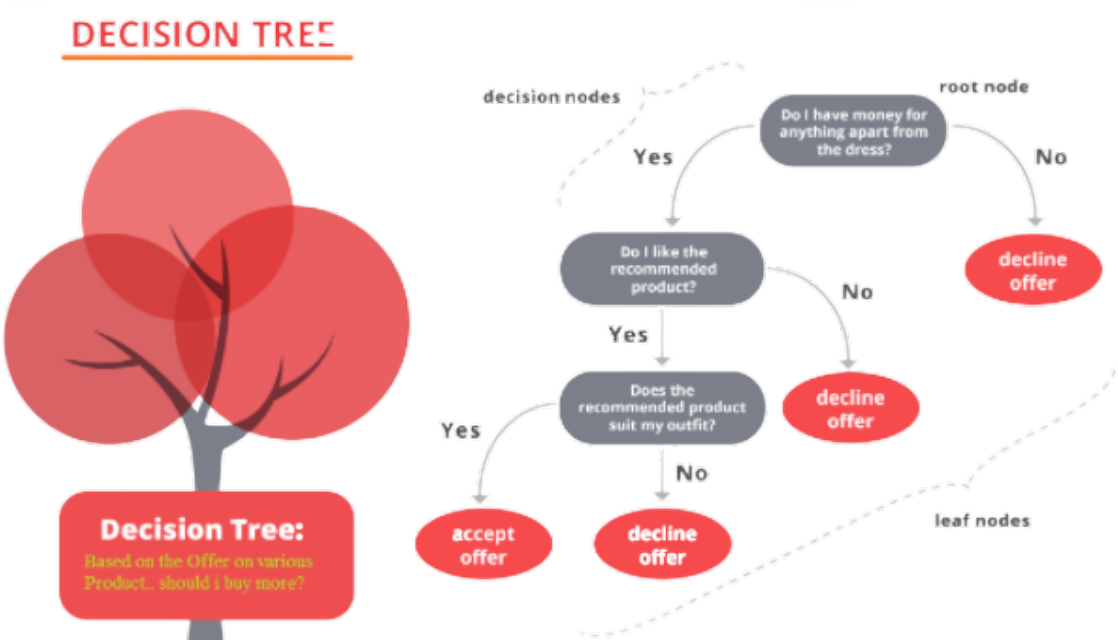


Figure 3.4: How Decision tree algorithm works [3]

why it is called a non-parametric method.

Working flow of decision tree algorithm [3].

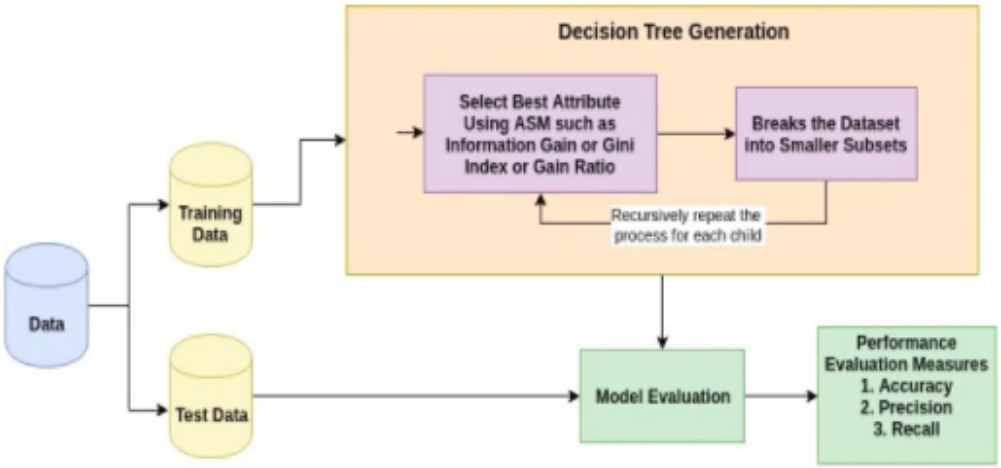


Figure 3.5: Working flow of decision tree algorithm [3]

3.3.3 Support Vector Machine (SVM)

Support vector machine is a classification approach. It can be used both for classification and regression problems. Multiple continuous and categorical variables can be handled by this algorithm. In multidimensional space to separate different classes SVM constructs a hyperplane. Optimal hyperplane is generated in an iterative manner, which is used to minimize an error. Maximizing marginal hyperplanes is the core idea of SVM that divides the dataset into classes. SVM algorithm is implemented using kernel. Kernel takes the non-separable problem and converts it into separable by adding more dimension to it. It is useful when the problem is non-separable. Kernel trick helps to build an accurate classifier. SVM classifiers have many advantages. SVM classifier offers good accuracy. SVM works well with a clear margin separation and with high dimensional space. As because they use a subset of training points in the decision phase, they need less memory space [4].

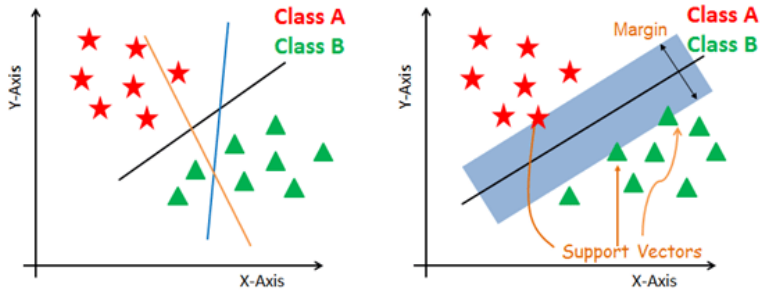


Figure 3.6: How Support vector machine algorithm works [4]

3.4 Learning Curves

The learning curve is well-known among data scientists. It demonstrates the effectiveness of the machine learning model and how it learns. Learning curves are frequently used as a diagnostic tool in machine learning for algorithms that gradually learn from training dataset. We extend our dataset by a certain amount, and then we test our model's performance on the training dataset and on the holdout validation dataset after each update during training. We can see how the amount of observations affects the performance matrices by looking at the learning curve.

3.5 Data Image Pre-processing Steps for Deep Learning

3.5.1 Data Pre-Processing Technique for Deep Learning

Green channel extraction from RGB images is a common preprocessing method since the green band has the most information compared to the red and blue bands.

The green channel of the picture is subjected to contrast enhancement in order to increase the image's contrast. Increasing the brightness and luminance of a picture is a common goal of illumination correction. In photography, Gaussian filtering is the process of smoothing out the picture and eliminating noise.

There is a tendency for researchers to remove the unnecessary and redundant black border from DR datasets so that they can focus on the area of interest (ROI).

Glow-in-the-dark filtering of the green channel; contrast enhancement; Illumination correction; Gaussian filtering; Resize; Optical disc and blood vessels removal; histogram equalization; amplification and gray-scale conversion; and cropping for the area of interest.

For a standardized training dataset, reduce the photos to 512 by 512 pixels.

Now, Let's see some data pre-processing technique that are used to detect DR:

1. BGR to RGB Conversion: There are some bluish pixels in our dataset. All the data are not in RGB mode, that's why, first we have proceeded through the process of converting the BGR (Blue, Green, and Red) shots to RGB (Red, Green, and Blue) images.

2. Auto Cropping: Following that, we have done auto cropping for our deep learning architectures. Auto cropping is used to eliminate the blank areas in images that don't transmit much information, such as in landscape images. For doing the cropping, we first estimated the pad width of the data image. After that, we used the auto cropping approach to adjust the image size based on the pad width. We used OpenCV python's auto cropping method. Because auto cropping is only effective on Gray Scale pictures, as a consequence of auto cropping, we needed to convert the data images from RGB to Grayscale, therefore, here we also utilized the OpenCV Python module.

3. Image Resizing: Images of our data have now been scaled to fit the desired shape: (256, 256). That means the height and width of the all images are converted to 256 pixels. The OpenCV python package was also utilized in this case.

4. Implementation of Gaussian Blur: We used the gaussian blur approach to eliminate

noise and other extraneous information from our data images. This technique was used to remove these undesired discoveries in our data that we had discovered. Gaussian blurring is a technique that is often used to compress images. Prior to downsampling a image, it is common practice to apply a low-pass filter on the image first. This is done in order to prevent the appearance of inaccurate high-frequency information in the downsampled image (aliasing), as well as to remove noise from the data pictures (noise reduction). Gaussian blur also utilizes the OpenCV python module.

Now let's see some sample Images of our 5 DR Classes after applying the all the Pre-Processing techniques in Figure 3.7.

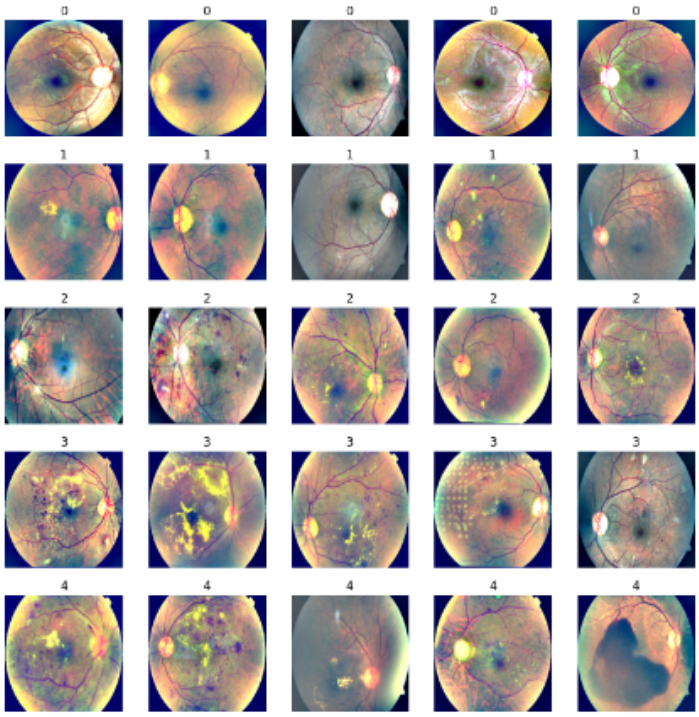


Figure 3.7: Sample Images of 5 DR Classes after applying the Pre-Processing Techniques

3.5.2 Data Augmentation for Deep Learning

As an artificial process, Augmentation creates equal numbers of instances for each class. When there is an unbalance between the pictures of various social classes, the augmentation approach is applied. When the number of DR photos is fewer than in the other classes, photographs are rotated, flipped, mirrored, or translated to create new examples of the chosen images.

It is very well known that all Deep Learning architectures are data hungry and Deep learning architectures perform well with higher amounts of data. For generating more data we have

used the ImageDataGenerator Keras library [29]. The features are mentioned below in table 3.1.

Table 3.1: Parameters for Data Augmentation

Rescale	**Shear Range**	**Zoom_range**	**Horizontal_flip**	**Fill_Mode**
1./255	0.2	0.2	True	constant

We reshaped all the images into (256,256,3). That means height and weight is 256, and Chanel number is: 3 (RGB images). We used the model loss in BinaryCrossetnropy for Binary Class, and Categorical_Crossentropy in Multi-Class classification of Diabetic Retinopathy detection. For faster calculation images are converted into NumPy arrays. Labeling of the images is done by the LabelBinarizer() method.

3.6 Deep learning Algorithms

Reasons of using Deep learning, and Transfer Learning models:

Machine Learning and Deep Learning are broad fields with many applications, including image recognition, speech recognition, recommendation and association systems, and so on. To create any model from the ground up, we'll need a lot of storage and computing power, which won't always be available. We may also encounter situations where we have methods for improving existing models, but the difficulties of retraining the models from scratch prevent us from doing so. We can use the concept of Transfer Learning to address such goals.

Transfer learning is a machine learning approach in which a model generated for one job is utilized as the foundation for another task's model. Instead than building models from scratch, this technique starts with pre-trained models for computer vision and natural language processing applications. This enables us to meet the challenge of developing Deep Learning models, which necessitates a large amount of computing and storage resources. Transfer learning in deep learning, on the other hand, only works if the model features learned from the first task are general.

3.6.1 Convolutional Neural Network (CNN) Architecture

To know Convolutional Neural Network. First lets know what is a neural network?

Neural Network: An important subgroup of deep learning algorithms are neural networks, which are also known as artificial neural networks (ANNs) and simulated neural networks (SNNs). Their name and structure are inspired by the human brain, replicating the way organic neurons communicate with one other.

An artificial neural network (ANN) is made up of a number of node levels, each of which contains an input layer, a hidden layer, and an output layer. Each artificial neuron, or node, has a certain weight and criterion related with it. When a node's output exceeds a certain threshold, it is activated and begins transferring data to the network's next layer. Otherwise, there is no way for the next layer of the network to receive any data.

Training data is essential for neural networks to learn and improve over time. While these algorithms are significant tools in computer science and artificial intelligence, they are only useful once they have been fine-tuned for accuracy. Tasks in speech or picture recognition can be completed in a matter of minutes rather than hours, compared to the manual identification of human experts. Google's search algorithm is one of the most well-known neural networks.

Now let's see a simple structure of Neural Network in Figure 3.8:

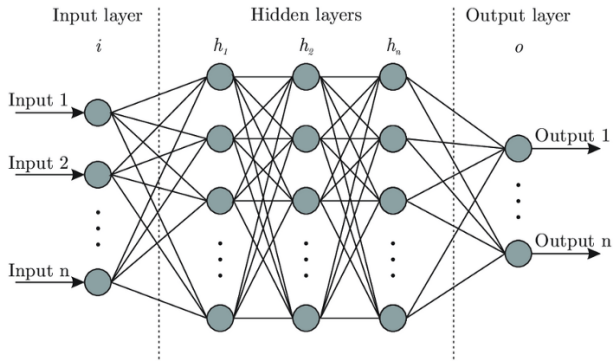


Figure 3.8: Basic Structure of Neural Network

Now let's discuss our main topic which is Convolutional Neural Network (CNN).

Convolutional Neural Network (CNN): A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take an image, give significance (learnable weights and biases) to various aspects/objects in the picture, and be able to discern one from the other.

Convolutional neural networks (also known as CNNs or ConvNets) are a type of artificial neural network often used to evaluate visual imagery in deep learning programs. As a result of the shared-weight architecture of convolution kernels and filters that slide along input features and produce translation equivariant responses known as feature maps, they are also known as shift invariant or space-invariant artificial neural networks (SIANN). Most convolutional neural networks are invariant to translation, which is the opposite of what one may expect. Image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series are just a few of the areas in which they can be used.

CNNs are regularized perceptrons that are based on multilayers. All neurons within one layer of a multilayer perceptron are connected to all neurons within a higher-layer perceptron. Networks with "full connection" are prone to overfitting data because of this. Regulating parameters during training (such as weight decay) and reducing connectivity are common approaches to prevent overfitting (skipped connections, dropout, etc.) A distinct method to regularization is used by CNNs, which take advantage of the hierarchical pattern in data and build larger patterns out of simpler ones. The bottom end of a scale of connectivity and complexity is CNNs.

Compared to other image classification methods, CNNs require very little pre-processing. These filters (or kernels) are optimized using automatic learning, whereas in traditional methods these filters are hand-crafted. Features can be extracted without any prior information or human involvement because of this independence.

Now let's see a simple structure of Neural Network in Figure 3.9:

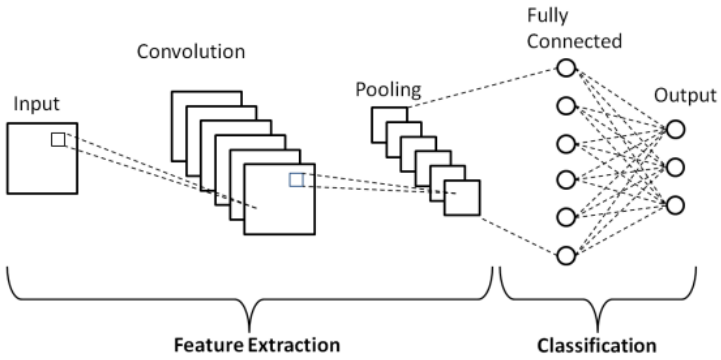


Figure 3.9: Basic Structure of Convolutional Neural Network (CNN)

3.6.2 VGG19 Architecture

The convolutional neural network, VGG19 was trained on over a million pictures from the ImageNet collection.

The VGG19 model is a variation of the VGG model that has 19 layers (16 convolution layers, 3 Fully connected layers, 5 MaxPool layers and 1 SoftMax layer). There are further VGG versions, including VGG11, VGG16, and others. VGG is a successor of AlexNet, although it was established by a separate group, the Visual Geometry Group at Oxford's, and thus the name VGG. It carries and utilizes some ideas from AlexNet and improves on them, and uses deep Convolutional neural layers to enhance accuracy. The VGG19 architecture consisted of sixteen layers of CNNs, three fully connected layers, and a final layer for the softmax function; the fully connected layers and final layer will remain consistent throughout all network architectures. Thus, VGG is a deep CNN that is used to classify images. The VGG19 model contains the following layers:

01. Conv3x3 (64), 02. Conv3x3 (64), 03. MaxPool, 04. Conv3x3 (128), 05. Conv3x3 (128), 06. MaxPool, 07. Conv3x3 (256), 08. Conv3x3 (256), 09. Conv3x3 (256), 10. Conv3x3 (256), 11. MaxPool, 12. Conv3x3 (512), 13. Conv3x3 (512), 14. Conv3x3 (512), 15. Conv3x3 (512), 16. MaxPool, 17. Conv3x3 (512), 18. Conv3x3 (512), 19. Conv3x3 (512), 20. Conv3x3 (512), 21. MaxPool, 22. Fully Connected (4096), 23. Fully Connected (4096), 24. Fully Connected (1000), 25. SoftMax

These layers can be shown by the following figure 3.10.

Architecture:

a. This network was fed a fixed-size (224 * 224) RGB image as input, indicating that the matrix was of shape (224,224,3).

Figure 3.10: Structure of VGG19 Network

b. The only preprocessing was to subtract the mean RGB value from each pixel, which was computed throughout the entire training set.
c. We used kernels of size (3 * 3) with a stride size of 1 pixel, which enabled them to cover the entire image concept. d. Spatial padding was utilized to maintain the image's spatial resolution.
e. Max pooling was accomplished with sride 2 over a 2 * 2 pixel window.
f. This was followed by the Rectified linear unit (ReLu) to include non-linearity into the model in order to increase classification and computing speed. Whereas previous models employed tanh or sigmoid functions, this proved to be significantly better.
g. Three fully linked layers were implemented, the first two of which were 4096 channels in size, followed by a layer of 1000 channels for 1000-way ILSVRC classification, and the final layer being a softmax function.

So, both VGG16 and VGG19 of VGGNet have two completely connected layers with 4096 channels each, followed by another fully connected layer with 1000 channels for label prediction. The final fully connected layer makes use of a softmax layer for classification.

3.6.3 InceptionV3 Architecture

Inception v3 is primarily concerned with conserving computational power through modifications to prior Inception architectures. This concept was advanced in the 2015 publication Rethinking the Inception Architecture for Computer Vision. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jonathon Shlens collaborated on it [30].

InceptionV3 is one of the updated versions of the initial inception family, which includes numerous features such as LabelSmoothing, Factorized 7x7 convolutions, and the use of an auxiliary classifier to distribute label information across the network. Additionally, batch normalization is used for the layers on the side head. It is mostly used for image analysis and object detection.

To begin, we froze the fundamental layer of our InceptionV3 architecture during our inquiry. This was accomplished by using the include top = false command and then superimposing our trained layer on top of this baseline. Additionally, all of our data images were formed into (225,225,3). It used the 'Relu Activation' function in this example, which is similar to the one used in the Adam stochastic gradient descent technique. We set the learning rate for backpropagation to 0.01 in our InceptionV3 architecture to enhance accuracy.

The following figure illustrates our study model, which is built on the InceptionV3 architecture is in Figure 3.11:

Figure 3.11, depicts the basic structure of the InceptionV3 model. It has created an inception

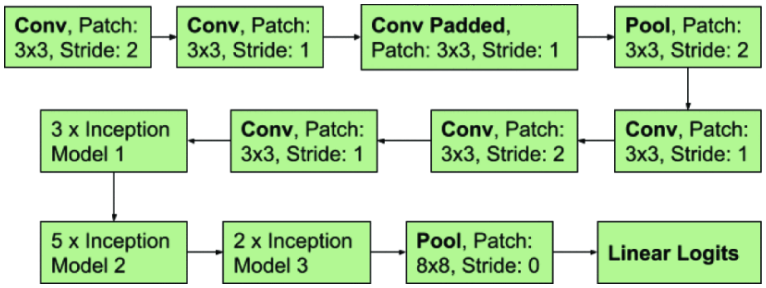


Figure 3.11: Basic Structure of InceptionV3

model that combines a number of different convolutional filters of varying sizes together into a single new filter, which is explained in greater detail below. There are reduced parameters to master and the computational complexity is reduced by using this architecture.

The details architecture of an Inception v3 network is constructed incrementally, as described below:

1. Factorized Convolutions: This contributes to increased computational efficiency by reducing the number of parameters in a network. Additionally, it monitors the network's efficiency.

2. Smaller convolutions: Training becomes much more efficient when larger convolutions are reduced in size in favor of smaller ones. If a 5 x 5 convolution filter has 25 parameters, two 3 x 3 filters replacing it have just 18 parameters (3 x 3 + 3 x 3).

3. Asymmetric convolutions: 3 x 3 convolution can be replaced by a 1 x 3 convolution followed by a 3 x 1 convolution. The number of parameters would increase somewhat if a 3 x 3 convolution was replaced with a 2 x 2 convolution.

4. Auxiliary classifier: Auxiliary classifiers are small CNNs that are introduced between layers during training, and their loss is contributed to the loss of the main network. Auxiliary classifiers were utilized in GoogLeNet to create a deeper network, whereas an auxiliary classifier works as a regularizer in Inception v3.

5. Grid size reduction: Typically, grid size reduction is accomplished through pooling procedures. However, in order to circumvent computational cost limitations, a more efficient technique is provided.

The finished architecture incorporates all of the aforementioned concepts.

3.6.4 ResNet50 Architecture

ResNet50 is a ResNet version that has 48 Convolutional layers, one MaxPool layer, and one Average Pool layer. It supports floating-point operations upto $3.8x10^9$. It is a frequently used ResNet model, and we have extensively examined the ResNet50 design. In part because of ResNets' design, it was able to train super deep neural networks with hundreds or even thousands of layers and still obtain excellent results. The ResNet framework was originally designed for image recognition, but it may also be used to improve the accuracy of non-computer vision activities. Before ResNet, deep neural networks were difficult to train because of the vanishing gradient problem. Each level of the ResNet-50 model has an identity block and convolution. The convolution in each block has three levels, and the convolution in each block of identity is three tiers deep. More than 23 million training parameters are available in the ResNet-50. Now let's see the basic achtiecture of ResNet50 in Figure 3.12

The ResNet50 architecture is composed of a convolution with a kernel size of 7 * 7 and 64 distinct kernels, each with a stride size of 2, resulting in a single layer. Following that, we see maximum pooling with a stride size of two. Next there comes a 1 * 1,64 kernel, followed by a 3 * 3, 64 kernel, and finally a 1 * 1,256 kernel. These three layers are repeated three times in total, giving us nine levels in this stage. Continuing that, we see a kernel of 1 * 1,128 followed by a kernel of 3 * 3,128 and finally a kernel of 1 * 1,512. This phase was performed four times, totaling 12 layers. Alongside that, there is a 1 * 1,256 kernel and two further kernels with 3 * 3,256 and 1 * 1,1024, which are repeated six times for a total of 18 layers. And then a 1 * 1,512 kernel was combined with two additional 3 * 3,512 and 1 * 1,2048 kernels, which was repeated three times for a total of nine layers. Following that, we perform an average pool and conclude with a fully linked layer having 1000 nodes and a softmax function, which results in a single layer. Thus, when we add these together, we get a $1 + 9 + 12 + 18 + 9 + 1 = 50$ layer Deep Convolutional network.

Bottleneck architecture is used in ResNet50, ResNet152, and other networks with more complex networks. Three stacked layers are used to represent each residual function F. The three layers are composed of single, triple, and single convolutions. The 11 convolution layers are responsible for decreasing the size and subsequently re-inflating it. Due to the fact that the 3x3 layer was retained as a bottleneck, it was shortened to fit the lower input/output dimensions.

In the following layer, there are 1000 neurons in a completely linked pooling layer. The Figure below in Figure 3.13 summarizes the layers and parameters of the various ResNet architectures.

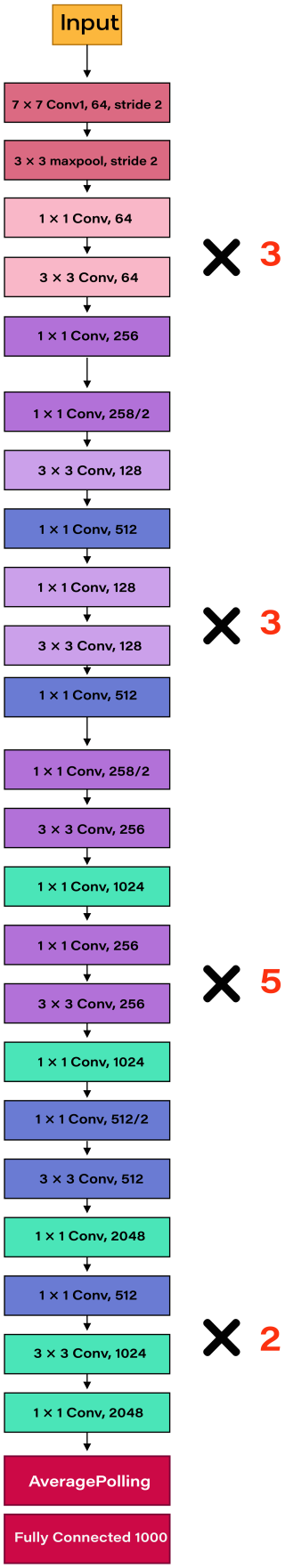
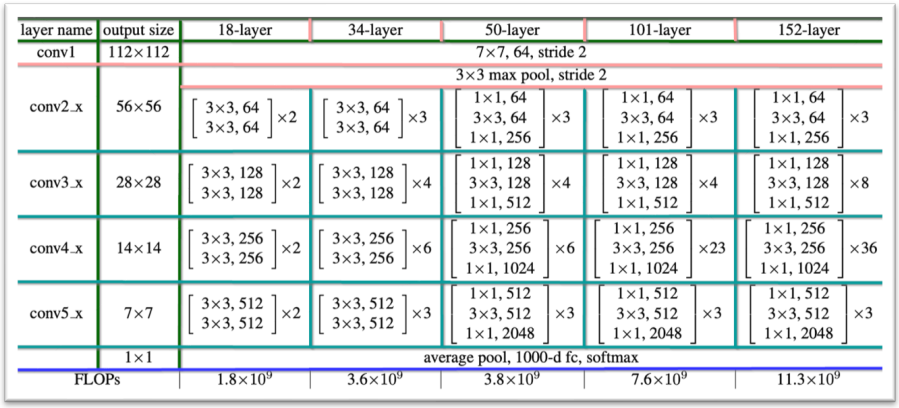


Figure 3.12: Basic Structure of ResNet50



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 3.13: Parametes for Various ResNet Architectures

3.6.5 ResNet50V2 Architectue

ResNet50V2 [31] is a revised form of ResNet50 that outperforms the ImageNet dataset than ResNet50 or ResNet101. The propagation concept of the connections between blocks has been altered in ResNet50V2. On the ImageNet dataset, ResNet50V2 also performs well. Figure 3.14 is showing the architectural diagram of ResNet50V2.

4 ResNet50 version 2 is all about pre-activating weight layers rather than post-activating them. The image below illustrates the fundamental design of ResNet's post-activation (original version 1) and pre-activation (version 2) variants. Figure 3.15 is showing it.

The following are the primary distinctions between ResNet – V1 and ResNet – V2:

1. ResNet V1 introduces the second non-linearity following the addition operation between x and F. (x). ResNet V2 has eliminated the last non-linearity, so establishing an identity relationship between the input and output.
2. Prior to multiplication with the weight matrix, ResNet V2 applies Batch Normalization and ReLU activation to the input (convolution operation). Convolution is performed via ResNet V1, followed by Batch Normalization and ReLU activation. Figure 3.16 is showing the difference between ResNetV1 and ResNetV2.

The ResNet V2 algorithm is primarily concerned with implementing the second non-linearity as an identity mapping, which means that the output of the addition operation between the identity and residual mappings should be transmitted directly to the next block for processing. However, in ResNet V1, the output of the addition operation is passed from ReLU activation to the next block as the input.

When 'f' is an identity function, the signal can be relayed directly between any two units. Additionally, without affecting the signal, the gradient value generated at the output layer can easily reach the beginning layer.

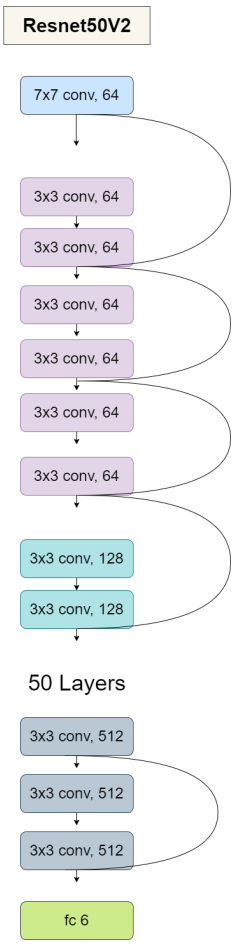


Figure 3.14: Architecture diagram of ResNet50V2

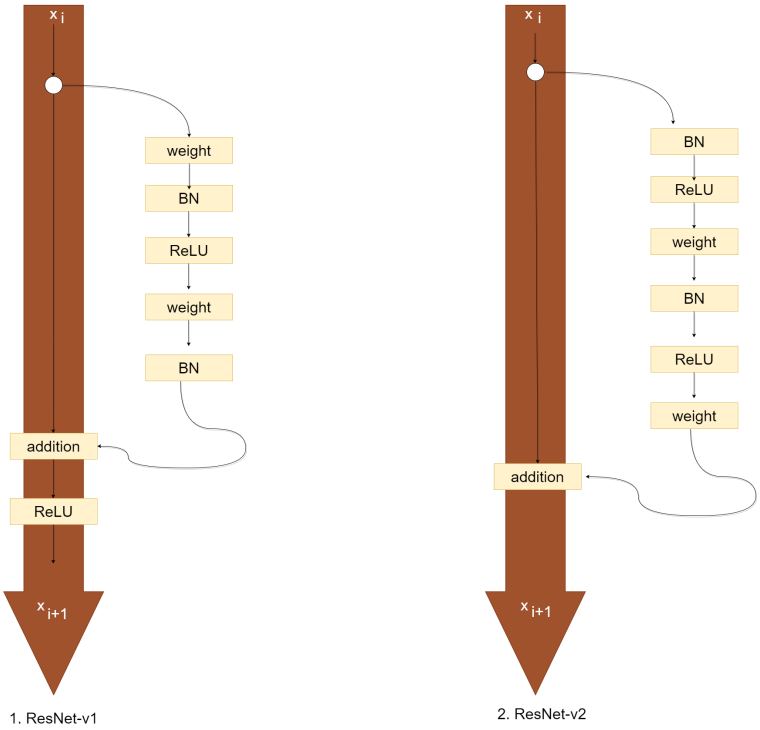


Figure 3.15: ResNetV1 and ResNetV2 Variants

ResNet - V1	ResNet - V2
$y = x_l + F(x_l, \{W_i\})$ $x_{l+1} = H(x) = ReLU(y)$	$y = h(x_l) + F(x_l, \{W_i\})$ $x_{l+1} = H(x) = f(y)$
y = Addition Output x_{l+1} = Input to Next Block	y = Addition output $h(x_l)$ = Generalized form of input. For ResNet V1, $h(x_l) = x_l$. f = Function applied to 'y'. For ResNet V1, f = ReLU. For ResNet V2, f is an identity mapping.

Figure 3.16: Difference between ResNet V1 and ResNet V2

3.6.6 DenseNet169 Architecture

As the name suggests, each layer of a DenseNet is connected to every other layer. There are L(L+1)/2 direct connections between the L levels. All preceding layers feature maps are utilized as inputs for each succeeding layer, and the layer's own feature maps are used as inputs to the next layer. DenseNets connect every layer to every other layer in a hierarchical fashion. This is the most important point to grasp. Levels in DenseNet are created by concatenating the feature maps of preceding layers and feeding that data into the system. The vanishing-gradient problem is alleviated, feature propagation is strengthened, feature reuse is encouraged, and the number of parameters is greatly reduced by dense networks. Figure 3.17 is showing the DenseNet169 Architecture.

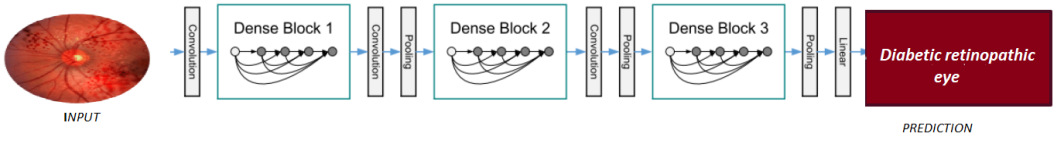


Figure 3.17: DenseNet169 Architecture

DenseNet-169 was considered as, although its 169-layer depth, it has a low parameter count in comparison to other models and the design well tackles the vanish gradient problem. The densenet-169 model is a member of the DenseNet family of image classification models. The primary difference between this model and the densenet-121 model is the model's size and accuracy. The densenet-169 is slightly larger, at around 55MB, than the densenet-121, which is approximately 31MB in size. The four dense blocks of DenseNet-169 have [6, 12,

32, 32] layers.

The details of the DenseNet169 model architectures are given here in Figure 3.18.

Layers	Output Size	DenseNet 169
Convolution	112×112	7×7 conv, stride 2
Pooling	56×56	3×3 max pool, stride 2
Dense Block [1]	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer [1]	56×56	1×1 conv
	28×28	2×2 average pool, stride 2
Dense Block [2]	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer [2]	28×28	1×1 conv
	14×14	2×2 average pool, stride 2
Dense Block [3]	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Transition Layer [3]	14×14	1×1 conv
	7×7	2×2 average pool, stride 2
Dense Block [4]	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Classification Layer	1×1	7×7 global average pool
	1000	1000D fully-connected, softmax

Figure 3.18: DenseNet169 Parameters Architectures

Chapter 4

Proposed Method

4.1 Proposed methodology for Machine Learning

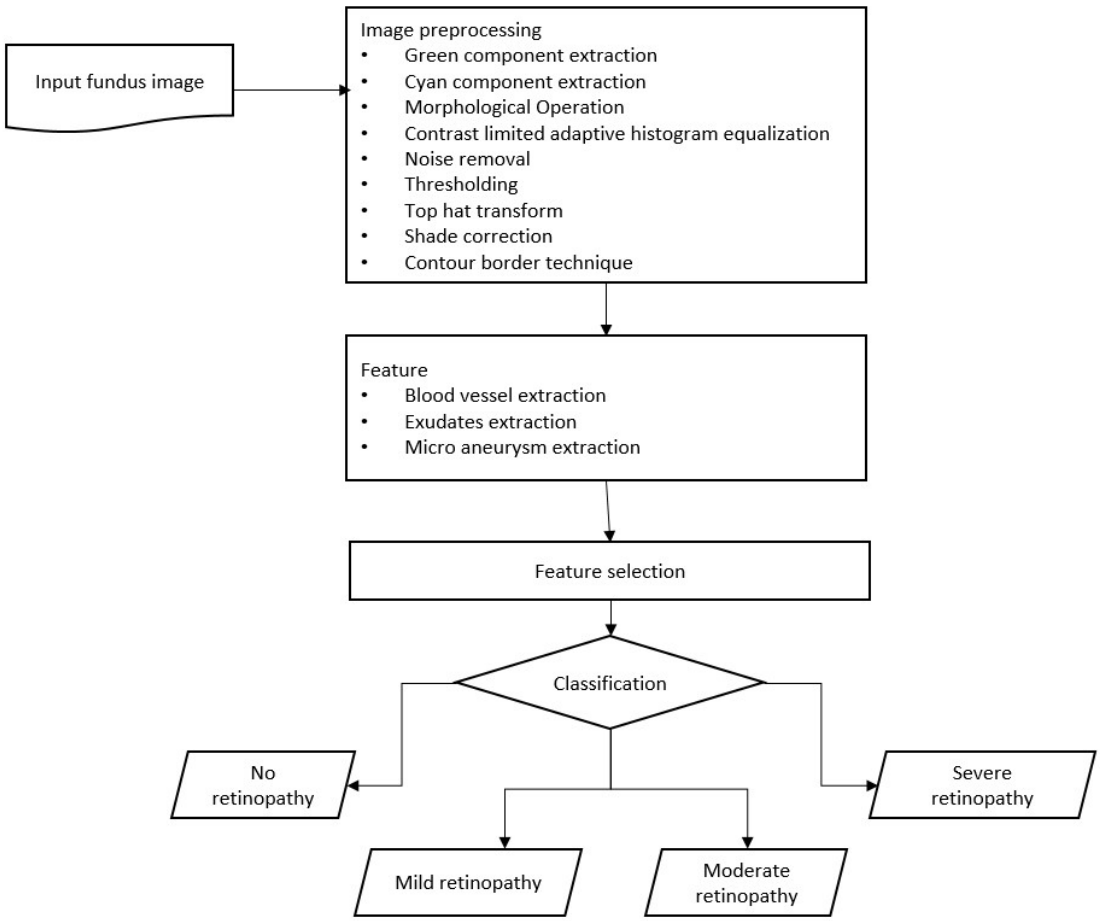


Figure 4.1: Proposed methodology diagram using Machine Learning

Summary of Machine Learning based proposed method for Detecting DR:

In the figure 4.1, at the first step we have collected retinal fundus images from Messidor. Then before extraction of the features, first we preprocess the images so that the features that needed to be extracted can be extracted properly. Some image enhancement techniques such as CLAHE, Thresholding, Gaussian filter, Morphological operation and shade correction are done in image preprocessing step. After preprocessing Blood vessels, Exudates and microaneurysms are extracted. Then feature reduction techniques are applied in the feature selection step. After that the dataset is divided into two sets, where one for training purposes and the rest one is for testing purposes. We used a Support vector machine, Random forest algorithm and Decision tree algorithm for the classification. After the model has learned the dataset, newer data is provided without output for testing purposes. In the final phase we get the accuracy of each algorithm we used and get to know which particular algorithm will give me a more accurate result for the prediction of diabetic retinopathy.

4.2 Proposed methodology for Deep Learning

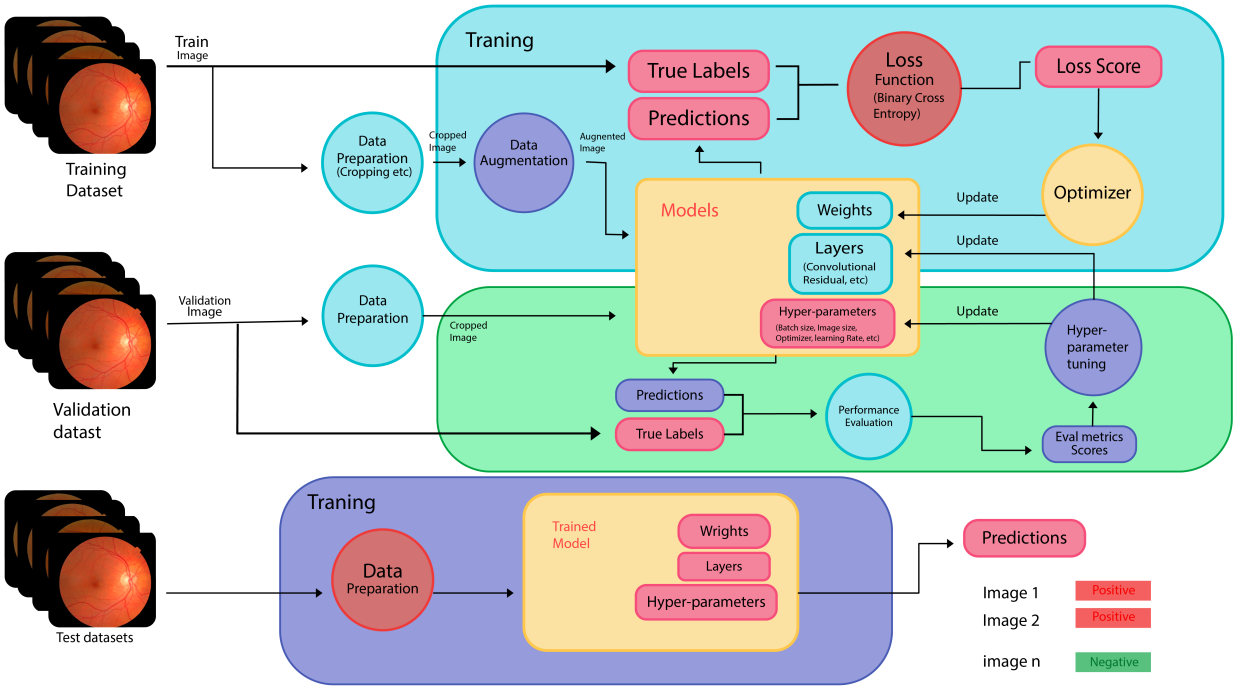


Figure 4.2: Proposed methodology diagram using Deep Learning

Summary of Deep Learning based proposed method for Detecting DR:

In the figure 4.2, First we have pre-processed the input images of Train and validation data (Using auto cropping, BGR to RGB conversion, Image Resizing, Implementation of Gaussian Blur) using OpenCV python module from APTOS-2019 dataset. Then We have done the Data Augmentation using ImageDataGenerator Keras Library. We used Binary Cross Entropy for loss function. Then using the optimizer we have updated the weights, layer, and Hyperparameters (Batch_size, Image_Size, Optimizer, Learning_Rate etc). Then we have added the classification layer. After we have evaluated the performance metrices (Accuracy, Precision, Recall, F1-Score, AUC) from validation dataset. The we have predicted our model using our Test Data.

These deep learning based architectures are pre-trained models. Here, we freeze the base layer of the deep learning models by using the command (IncludeTop= false) and at the top of that we have added our trainable layer and used 'ImageNet' as weights. We have taken ImageShpae=(256,256,3) We have used the ClassifierActivation ('softmax' activation) function if requires.

For full connection we have used two dense layers. In the first dense layer we have used the 'Relu' activation function and in the second dense layer we have used the 'Sigmoid' activation function. Between the dense layers we have used the dropout of 0.5 value.

In our Deep Learning based models we used one Convolutional Neural Network (CNN) and Five Transfer Learning Models (VGG19, InceptionV3, ResNet50, ResNet50V2, and DenseNet169). Now see their Respective Flow Charts.

Convolutional Neural Network(CNN): Now let's see the flowchart diagram of our CNN model. Here in Figure 4.3 is showing the flowchart of our Deep Learning based CNN model.

Here we can observe in the figure 4.3 that, first from the fundus photography we have done the image preprocessing, then after dataset compilation, then we have extracted the features and moved it forward to the classification model with meta data information.

Transfer Learning: Now let's see the flowchart diagram of our Transfer Learning model. Here in Figure 4.4 is showing the flowchart of our Deep Learning based Transfer Learning (TL) model.

Here we can observe in the figure 4.4 that, from the dataset we have pre processed and augmented the dataset, and implemented the the transfer learning model with convolutional, and pooling layer. After that using the dense layer we have given the full connection. Finally we evaluated the performance metriecs.

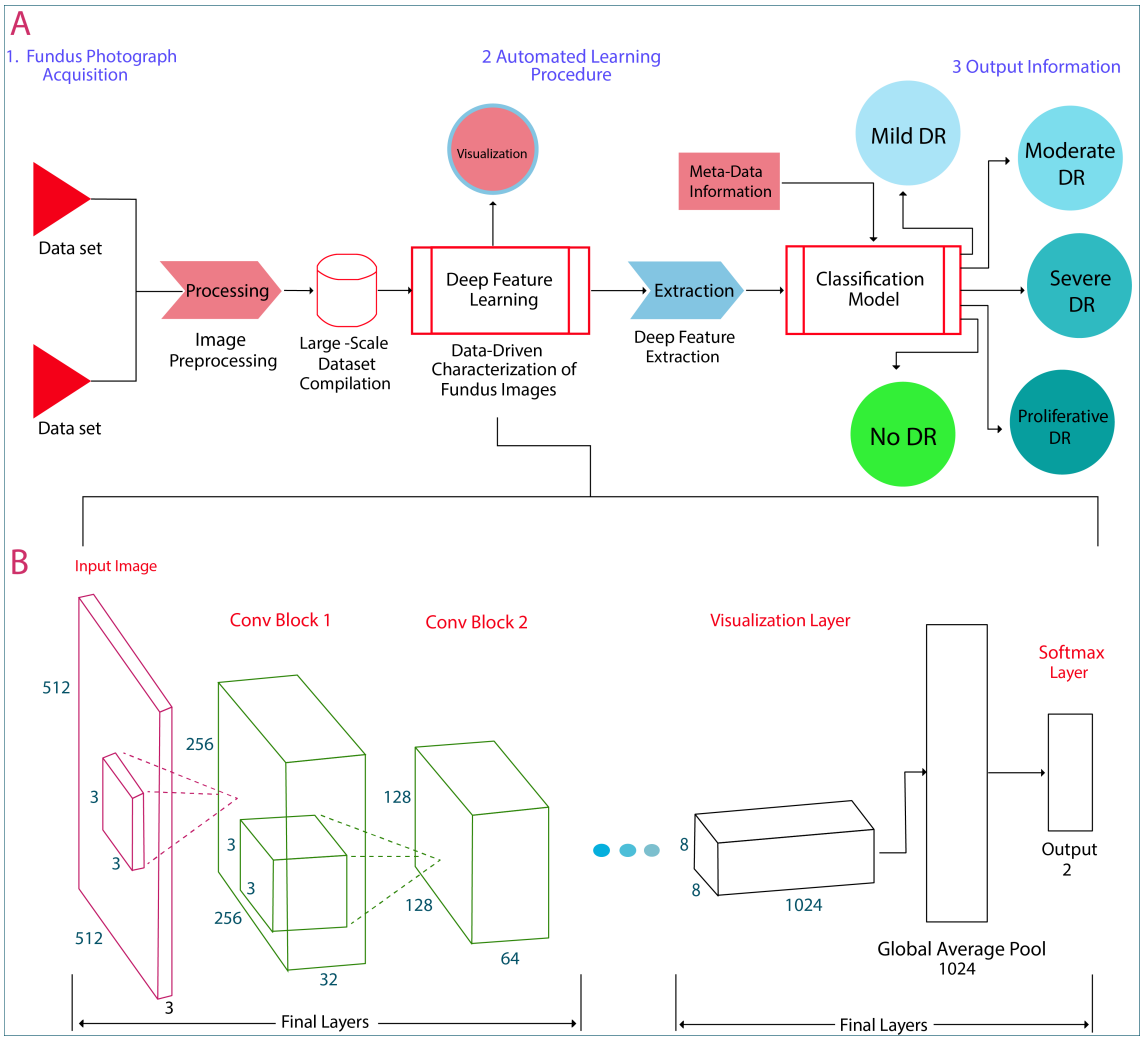


Figure 4.3: Proposed methodology diagram using Deep Learning based CNN Model

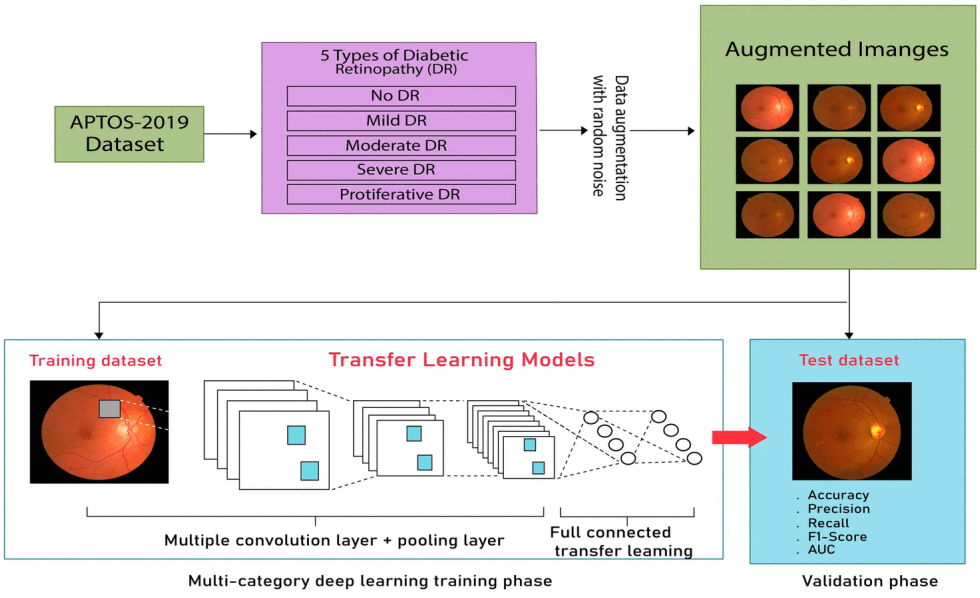


Figure 4.4: Proposed methodology diagram using Deep Learning based TL Model

Chapter 5

Implementation

5.1 Feature extraction for Machine Learning

5.1.1 Blood vessel extraction

As because green component holds better contrast and has good details first Green component is isolated from RGB retinal fundus image.After that CLAHE (Contrast limited adaptive histogram equalization) is performed on the green component. CLAHE makes the dark region (Blood vessel) darker and enhances the contrast of the image. On histogram equalized image morphological Opening operation is performed. Then to hide the blood vessels we applied median filter. Then difference between the median filtered image and opened image is binarize using threshold technique. To remove extra salt paper noise we perform gaussianblur [12].

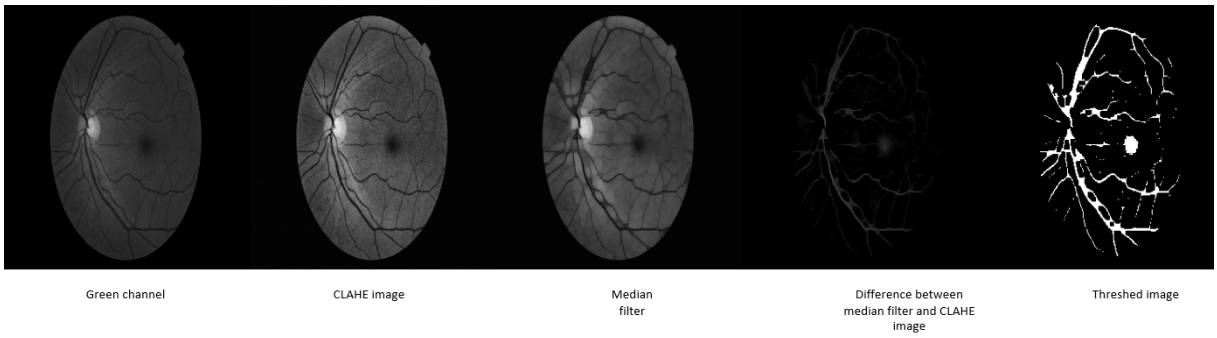


Figure 5.1: Blood vessel extraction procedure

5.1.2 Exudate extraction

First the RGB image is transformed to its CMY representation and the magenta component is extracted. On magenta component morphological dilation operation is performed to hide blood vessels exudates and optic disc. Difference between the dilated image and magenta component is binarize using threshold operation. Median filter is applied to the resultant image to reduce the noise. After removing noise the resultant images show the exudates and the optic disc portion. So to remove the optic disc from the resultant image we have to detect the optic disc. In order to remove the disc portion from retinal image, the magenta component is subtracted from the gray scale image. After that largest contour is extracted using contour border technique and that largest contour is the optic disc. After that the optic disc is subtracted from the resultant image and the subtracted image shows the exudates region.

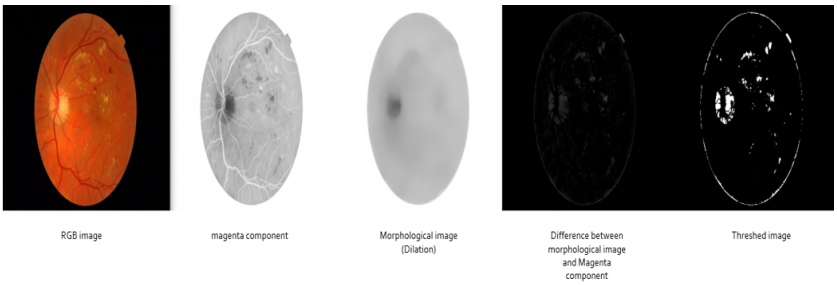


Figure 5.2: Exudate with optic disc

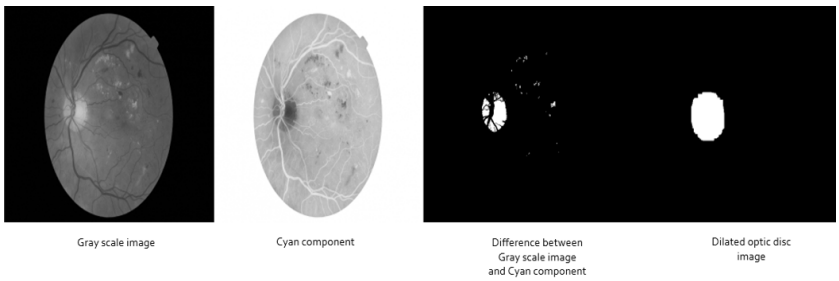


Figure 5.3: optic disc extraction procedure

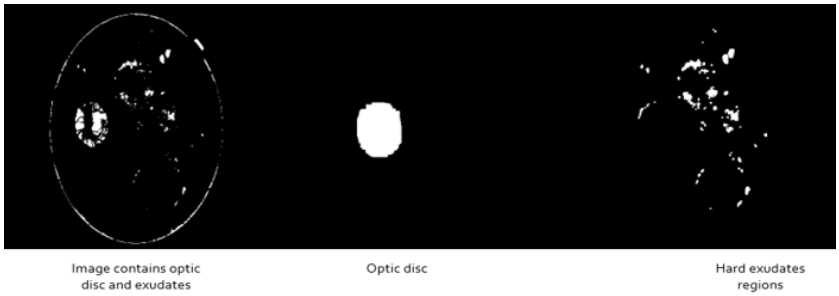


Figure 5.4: Exdate detection

5.1.3 Micro aneurysm extraction

To detect micro aneurysm, the first green channels of the color fundus image are dis- noised by a median filter to remove salt pepper noise that are the main noise effect of the retinal fundus image. Contrast limited adaptive histogram equalization is applied on the filtered image. And the resultant image shows better contrast. This contrast enhancement method makes the dark regions such as micro aneurysm . To remove unwanted changes in intensity that occurred across the image due to the images obtained in the previous step, a shade correction method is applied. Morphological closing with disc as a structuring element is performed which fills the hole inside the dark circular details [28].

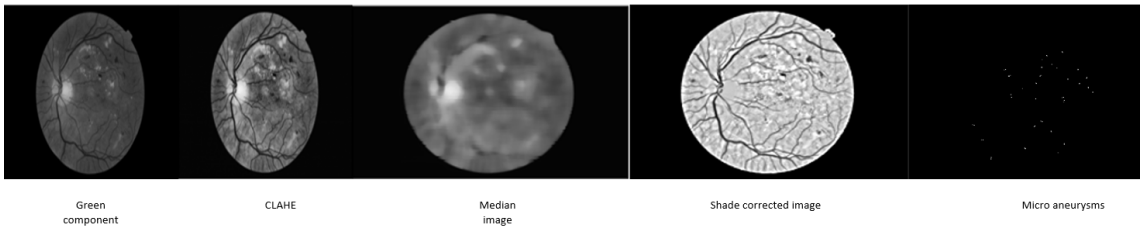


Figure 5.5: Micro aneurysm extraction procedure

5.2 Feature scaling for Machine Learning

Feature scaling is a data-preprocessing process that normalises the range of data in the dataset. In a dataset features may vary in degrees of magnitude,range and units. Machine learning algorithms such as logistic regression, linear regression are sensitive to feature scaling So to handle highly varying magnitudes of units feature scaling is performed.

5.3 Features for Machine Learning

We took 7 features for Diabetic retinopathy detection

Texture features:

- Standard deviation of red channel
- Standard deviation of green channel
- Standard deviation of blue channel
- Green channel entropy

Defects:

- Blood vessel density
- Exudates density
- Micro aneurysms density

5.4 Feature selection for Machine Learning

Before training a classifier we analyze the dependencies between different features. From the correlation matrix we see that there is a positive correlation between standard deviation of green and standard deviation of blue. So we dropped the standard deviation of the blue channel.

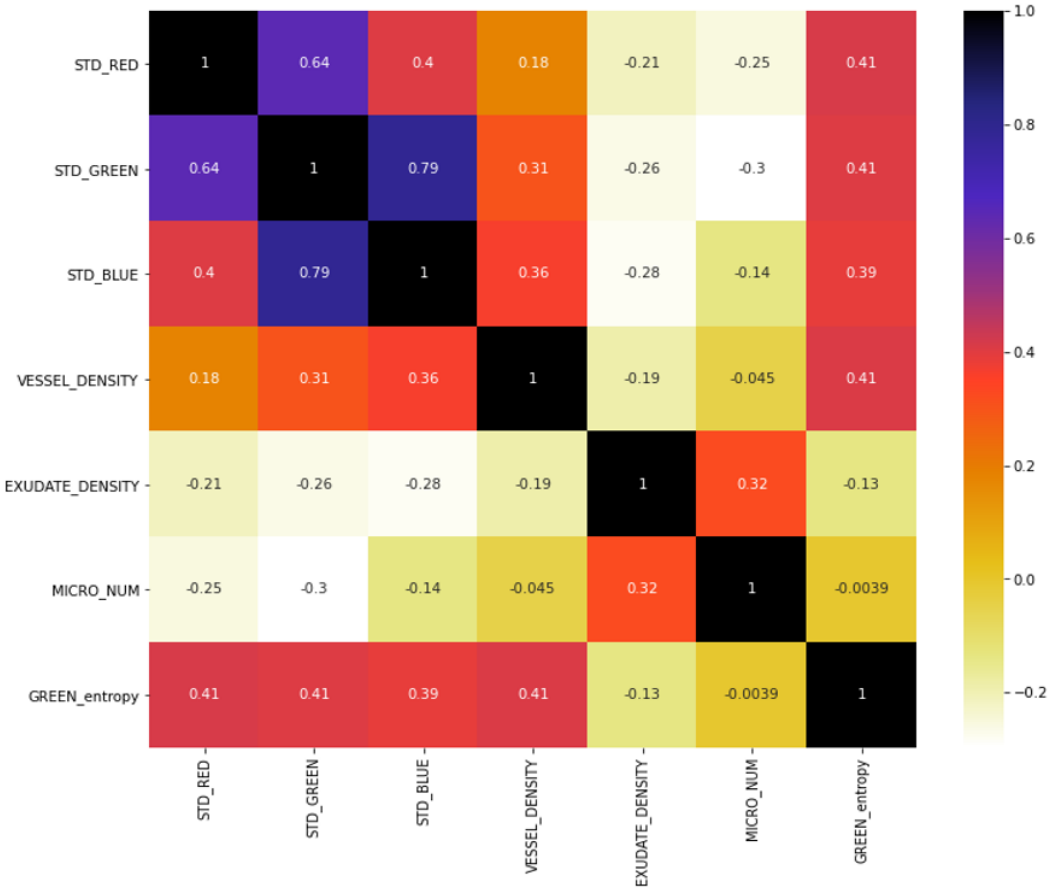


Figure 5.6: Co relation among the features(Pearson correlation)

5.5 Implementation using Deep Learning

5.5.1 Deep Learning based CNN Architecture

Now in this section we will observe our implemented structure of Convolutional Neural Network (CNN) architecture in our research. The implemented structure is shown below in Figure 5.7. Our CNN model includes two convolutional layers. Throughout the convo-

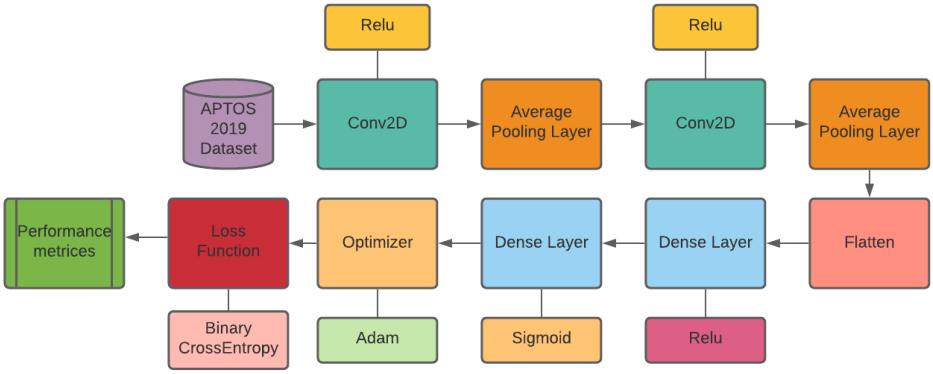


Figure 5.7: Implemented CNN Model

lutional network, Convolutional2D was employed. The 'ReLu activation function was used in both convolutional2D layers. Two Dense Layers were used for full connectivity. Relu activation function and Sigmoid activation function were employed to activate the first and second dense layers, respectively. Additionally, there are a few hidden layers, as well as an input layer, that are related to the layers themselves. The average pooling2D layer has been employed in our investigation.

CNN architecture using average pooling Operation: Since the average pooling method takes into account all values, it is a very generalized computation that may be used to translate features to outputs. Lower layer pooling collects all the values, which are transferred to the next layer where features are mapped and output is generated. The images have been sculpted to take on a variety of (256,256,3). In other words, the images are 256 by 256 pixels and have a channel value of 3, indicating they are RGB images. Located within the convolution layer, the pool has a dimension of (2,2). A total of 32 nodes are activated during Relu activation, each of which has a specific purpose. The activation of Relu requires the employment of 32 nodes, each of which serves a specific purpose. Using average pooling for input characteristics, we implemented a deep learning-based CNN architecture in the Figure in 5.7.

Here we used two convolutional layer and between each convolutinal we haved used Average Pooling Layer as Pooling Layer. Then we have flatten our dataset and converted it 1-D. Then we have used two Dense Layers for 'Full Connection'. In the first dense layer we

have used 'Relu Activation' function, and in the second dense layer we have used 'Sigmoid Activation Function'. As optimizer we have used 'adam' optimizer. For Calculating the loss function we have used 'BinaryCrossEntropy' function. Then we have evaluated performance matrices like: Accuracy, Precision, Recall, F1-Score.

Here in Figure 5.8 we can observe that, our Deep Learning based Convolutional Neural Network (CNN) has total number of 15,755,425 parameters and there is no non trainable parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 average_pooling2d (AverageP  (None, 127, 127, 32)     0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      9248

 average_pooling2d_1 (Averag  (None, 62, 62, 32)       0
 ePooling2D)

 flatten (Flatten)           (None, 123008)            0

 dense (Dense)               (None, 128)               15745152

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 15,755,425
Trainable params: 15,755,425
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.8: CNN Training Parameters

5.5.2 Deep Learning based Transfer Learning Architecture

Now in this section we will observe our implemented structure of Transfer Learning architecture in our research. The implemented structure is shown below in Figure 5.9. Here we can observe from 3.9 that, first from the dataset we get the input layer. Then we have done the all the required pre-processing steps, then we do the data augmentation for increasing the data. Then we have employed the pre-trained transfer learning model (VGG19, DenseNet169, InceptionV3, ResNet50, ResNet50V2). These are pre-trained models which are previously trained on more than 2 million dataset of the ImageNet database. Then we have employed Global Average Pooling 2D as pooling Layer. Then for avoiding the overfitting we have used the Dropout of 0.5. The We have used a DenseLayer with Relu activation

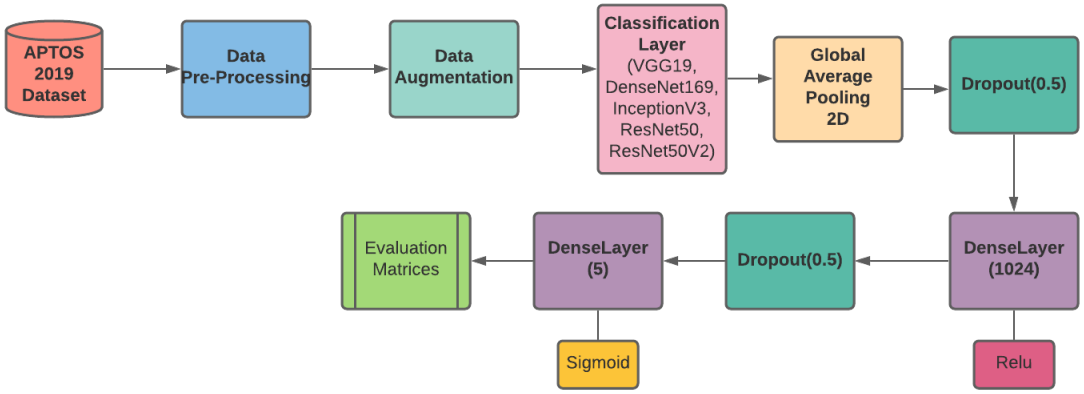


Figure 5.9: Transfer Learning Implemented architecture

function for the Full Conncetion, then we have used another dropout of 0.5, then another DenseLayer with Sigmoid activation function for the final full connection. The we have done the performance matrices on the basis of this model.

Transfer Learning based VGG19 Architecture

Here in Figure 5.10 we can observe that, our Transfer Learning based VGG19 has total number of 28,413,505 parameters, where there are 17,828,353 trainable parameters and there is 10,585,152 non trainable parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 8, 8, 512)         20024384

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 256)               8388864

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 28,413,505
Trainable params: 17,828,353
Non-trainable params: 10,585,152
_________________________________________________________________
None
```

Figure 5.10: VGG19 Training Parameters

Transfer Learning based DenseNet169 Architecture

Here in Figure 5.11 we can observe that, our Transfer Learning based DenseNet169 has total number of 39,906,369 parameters, where there are 27,303,681 trainable parameters and there is 12,602,688 non trainable parameters.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet169 (Functional)    (None, 8, 8, 1664)        12642880

 flatten_1 (Flatten)         (None, 106496)            0

 dense_2 (Dense)             (None, 256)               27263232

 dropout_1 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 1)                 257

=================================================================
Total params: 39,906,369
Trainable params: 27,303,681
Non-trainable params: 12,602,688
_________________________________________________________________
None
```

Figure 5.11: DenseNet169 Training Parameters

Transfer Learning based InceptionV3 Architecture

Here in Figure 5.12 we can observe that, our Transfer Learning based InceptionV3 has total number of 40,677,665 parameters, where there are 18,874,881 trainable parameters and there is 21,802,784 non trainable parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 6, 6, 2048)        21802784

 flatten (Flatten)           (None, 73728)             0

 dense (Dense)               (None, 256)               18874624

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 40,677,665
Trainable params: 18,874,881
Non-trainable params: 21,802,784
_________________________________________________________________
None
```

Figure 5.12: InceptionV3 Training Parameters

Transfer Learning based ResNet50 Architecture

Here in Figure 5.13 we can observe that, our Transfer Learning based ResNet50 has total number of 57,142,657 parameters, where there are 34,609,665 trainable parameters and there is 22,532,992 non trainable parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 flatten (Flatten)           (None, 131072)            0

 dense (Dense)               (None, 256)               33554688

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 57,142,657
Trainable params: 34,609,665
Non-trainable params: 22,532,992
_________________________________________________________________
None
```

Figure 5.13: ResNet50 Training Parameters

Transfer Learning based ResNet50V2 Architecture

Here in Figure 5.14 we can observe that, our Transfer Learning based ResNet50V2 has total number of 57,119,745 parameters, where there are 34,609,665 trainable parameters and there is 22,532,992 non trainable parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50v2 (Functional)     (None, 8, 8, 2048)        23564800

 flatten (Flatten)           (None, 131072)            0

 dense (Dense)               (None, 256)               33554688

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 57,119,745
Trainable params: 34,609,665
Non-trainable params: 22,510,080
_________________________________________________________________
None
```

Figure 5.14: ResNet50V2 Training Parameters

Chapter 6

Experimental Results

6.1 Result Analysis for Machine Learning

6.1.1 K-Fold cross validation:

In the field of machine learning cross validation is widely used. Original dataset is partitioned amoung k equal size subsamples. From K subsamples, one of the subsamples is taken as the validation data for the testing model and the remaining k-1 subsamples are used as training data. In our thesis work we used 10 fold cross validation. The advantage of this procedure is that each sample is given the opportunity to be used [6].

6.1.2 Accuracy for support vector machine

We obtained training accuracy of 83% for binray class classification and 77% for multiclass classification using the SVM method, which is satisfactory. Support Vector Machines (SVMs) are a type of classifier that uses a hyperplane to separate the classes. We chose SVM because it is capable of classification and regression, as well as capturing considerably more intricate correlations between data points.

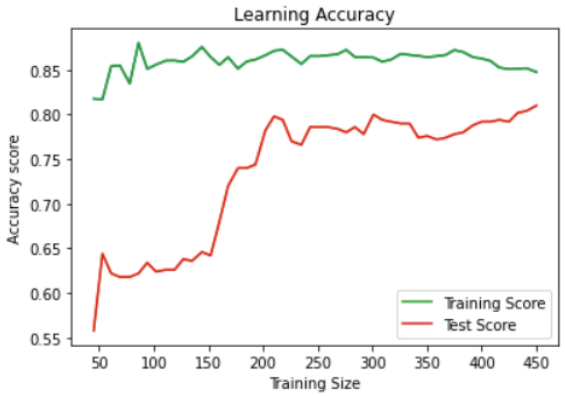


Figure 6.1: Learning curve for binary class classification

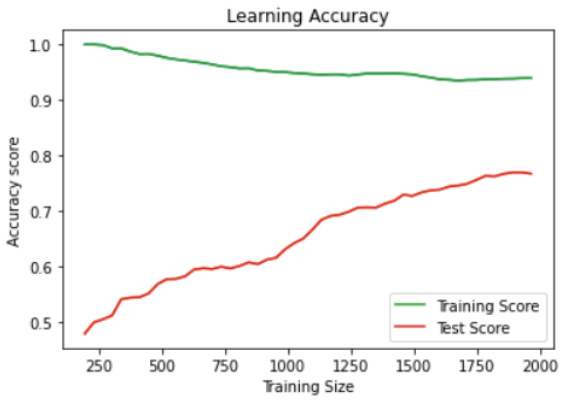


Figure 6.2: Learning curve of for multiclass class classification

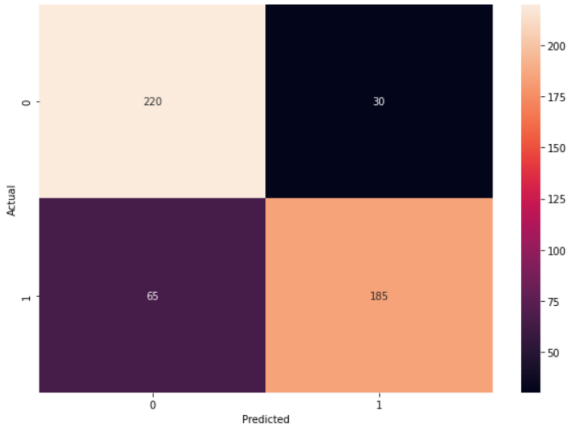


Figure 6.3: Confussion matrix for binary class classification

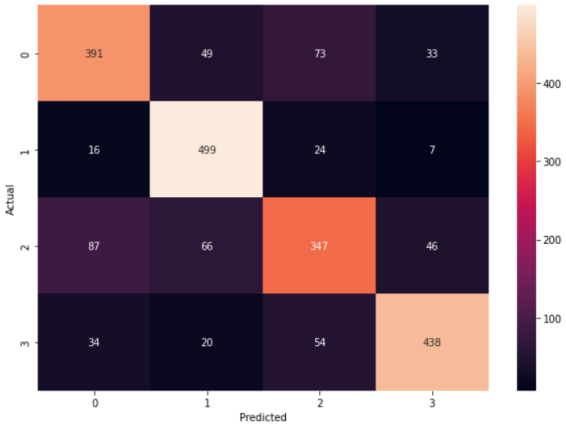


Figure 6.4: Confussion matrix for multi-class class classification

***Result:* Classification report**

Table 6.1: Performance metrics of SVM for binary class classification

Model	Classes	Accuracy	Precision	Recall	F1-Score
SVM	0	83%	78%	90%	84%
	1		88%	75%	81%

Table 6.2: Performance metrics of SVM for multiclass class classification

Models	Classes	Accuracy	Precision	Recall	F1-Score
SVM	0	77%	74%	72%	73%
	1		79%	91%	85%
	2		70%	64%	66%
	3		84%	80%	82%

6.1.3 Accuracy for Random Forest

We have an accuracy of 83.92 % and 76% for binary and multiclass classification when it comes to Random Forest. In the same way that SVM can be used for classification and regression, Random Forest can be used for both. We can observe that the SVM and Random Forest accuracy results are very similar. Random forest has an accuracy of around 84 % on binary class classification, which is likewise acceptable for our purposes and also 76% accuracy for multiclass which is quite satisfactory.

Figure 6.5: Learning curve for binary class classification

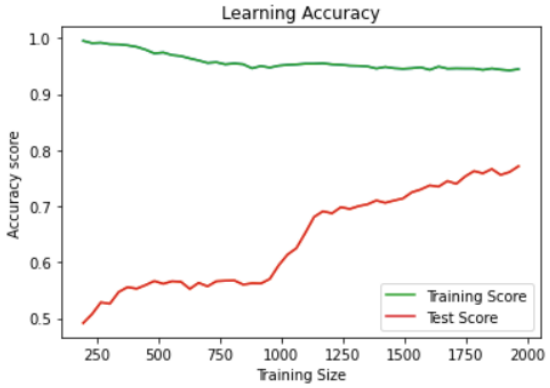


Figure 6.6: Learning curve of for multiclass class classification

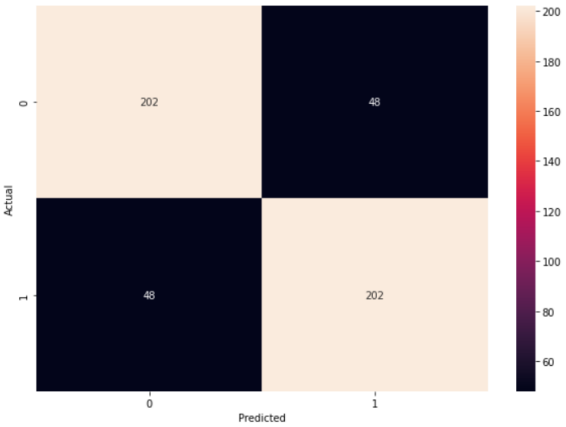


Figure 6.7: Confussion matrix for binary class classification

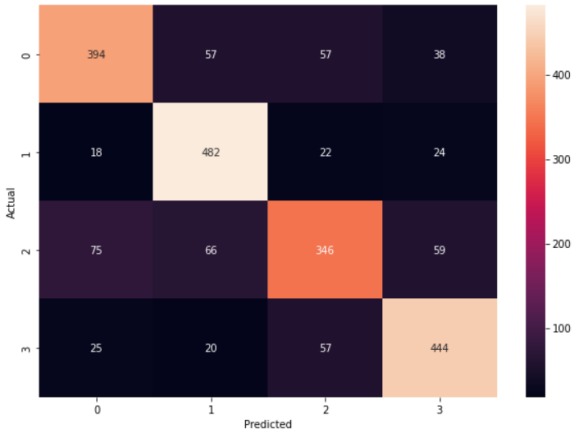


Figure 6.8: Confussion matrix for multi-class class classification

Result: **Classification report**

Table 6.3: Performance metrics of Random Forest classifier for binary class classification

Model	Classes	Accuracy	Precision	Recall	F1-score
Random Forest	0	84%	84%	84%	84%
	1		84%	84%	84%

Table 6.4: Performance metrics of Random Forest classifier for multi class classification

Models	Classes	Accuracy	Precision	Recall	F1-Score
Random Forest	0	76%	77%	72%	74%
	1		77%	88%	82%
	2		72%	63%	67%
	3		79%	81%	80%

6.1.4 Accuracy for Decision tree classifier

Decision Tree is a well-known machine learning algorithm. Decision Tree addresses the problem of machine learning by converting data into a tree representation. In the tree model, each attribute is represented by an internal node, and each class label is represented by a leaf node.A decision tree strategy may be used to tackle both regression and classification problems.Decision trees need less work for data preparation during pre-processing than other methods..Data does not need to be normalized when using a decision tree.Furthermore, missing values in the data have no effect on the decision tree-building process. We got around 80% accuracy when it comes to binary class classification and 69% in multiclass classification, which is acceptable.

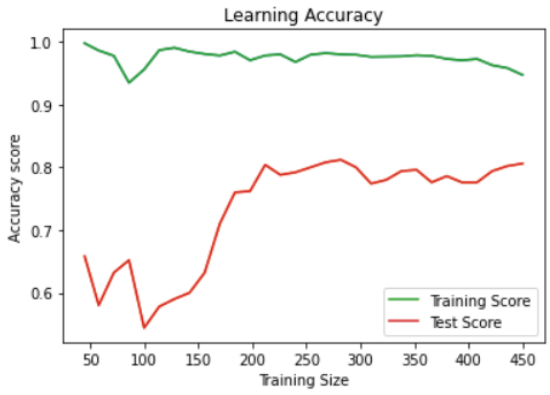


Figure 6.9: Learning curve for binary class classification

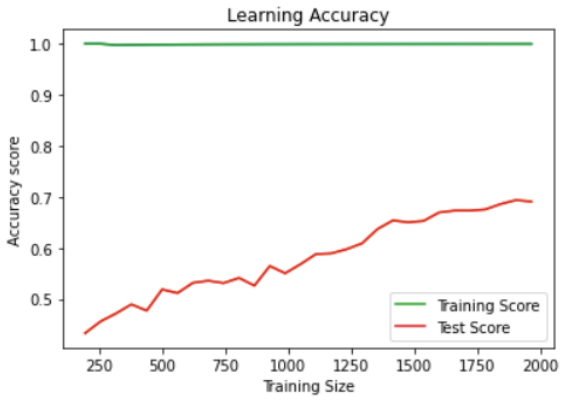


Figure 6.10: Learning curve of for multi-class classification

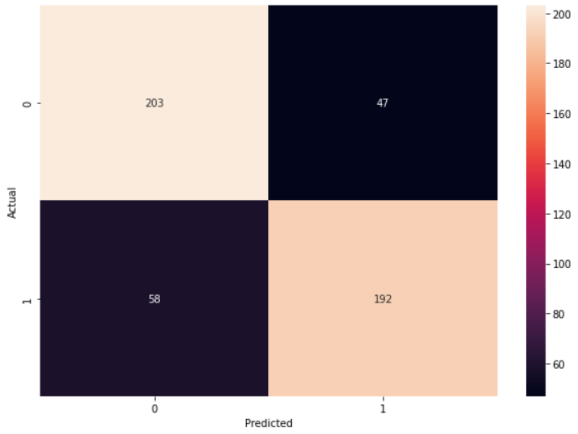


Figure 6.11: Confussion matrix for binary class classification

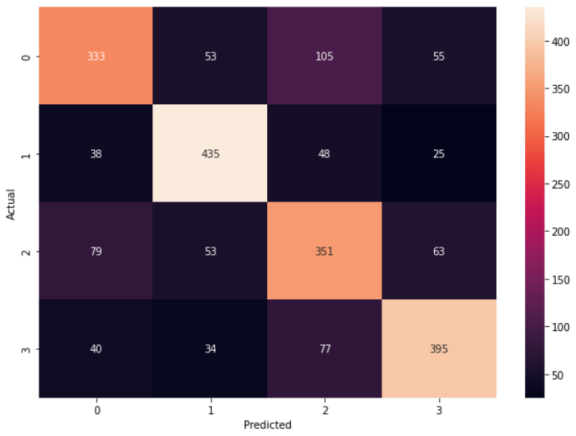


Figure 6.12: Confussion matrix for multi-class class classification

***Result:* Classification report**

Table 6.5: Performance metrics of Decision tree classifier for binary class classification

Model	Classes	Accuracy	Precision	Recall	F1-score
Decision tree	0	80%	79%	82%	80%
	1		81%	78%	80%

Table 6.6: Performance metrics of Decision tree classifier for multi class classification

Models	Classes	Accuracy	Precision	Recall	F1-Score
Decision tree	0	69%	68%	61%	64%
	1		76%	80%	78%
	2		60%	64%	62%
	3		73%	72%	73%

6.2 Result Analysis for Deep Learning

After finishing the training and validation stage on the basis of TP, TN, FP, and FN, we measured the performance of the five deep models using Accuracy, Recall/Sensitivity, Precision, F1-score, and AUC. The total number of true-positives, false-positives, true-negatives, and false-negatives is represented by TP, FP, TN, and FN, respectively.

Now,

1. TP (True-Positives) = The number of patients are affected by DR and the machine is capable of determining them as DR

2. TN (True-Negative) = The number of patients are not affected by DR and the machine is capable of determining them as NO-DR

3. FP (False-Positives) = The number of patients are not affected by DR but the machine is determining them as DR

4. FN (False-Negatives) = The number of patients are affected by DR but the machine is determining them as NO-DR

And the formulas for accuracy, precision, recall/sensitivity, specificity, and f1-score are given below:

$$Accuracy = \left(\frac{TP + TN}{TP + TN + FP + FN} \right) \tag{6.1}$$

$$Precision = \left(\frac{TP}{TP + FP} \right) \tag{6.2}$$

$$Recall/Sensitivity = \left(\frac{TP}{TP + FN} \right) \tag{6.3}$$

$$F1 - Score = \left(\frac{2xPrecisionxRecall}{Precision + Recall} \right) \tag{6.4}$$

we also made a confusion matrix on the basis of TP, TN, FP, and FN for all the deep learning models (CNN, VGG19, InceptionV3, ResNet50, ResNet50V2, and DenseNet169).

Table 6.7: Demo Confusion Matrix

	Predicted 0	**Predicted 1**
Actual 0	TN	FP
Actual 1	FN	TP

6.3 Result analysis of Deep Learning Models in Binary Class Classification

6.3.1 Result analysis for CNN Model in Binary Class Classification

CNN Model Binary Class Classification: our deep learning based CNN achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.32, 0.88, 0.86, 0.91, 0.89, and 0.95 respectively.

Accuracy, Loss, and AUC Diagram for CNN in Binary Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying

CNN Architecture.

The Loss, Accuracy, and AUC diagram of CNN architecture is shown below in Figure 6.13, 6.14, and 6.15 respectively.

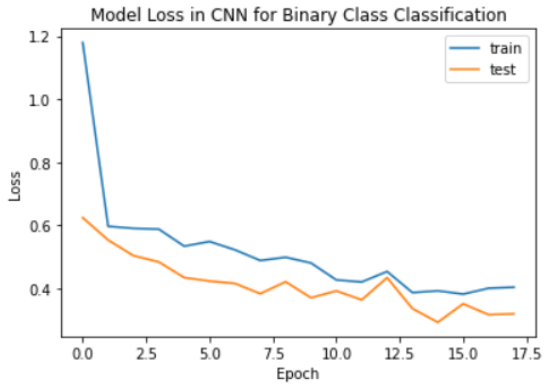


Figure 6.13: Training and Validation model loss for the CNN architecture in different epochs

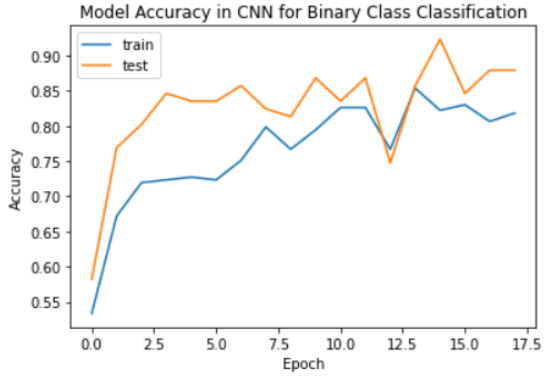


Figure 6.14: Training and Validation model accuracy for the CNN architecture in different epochs

6.3.2 Result analysis for VGG19 in Binary Class Classification

VGG19 Model Binary Class Classification: our deep learning based VGG19 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.20, 0.98, 0.98, 0.99, 0.99, and 0.987 respectively.

Accuracy, Loss, and AUC Diagram for VGG19 in Binary Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying VGG19 Architecture.

The Loss, Accuracy, and AUC diagram of CNN architecture is shown below in Figure 6.16, 6.17, and 6.18 respectively.

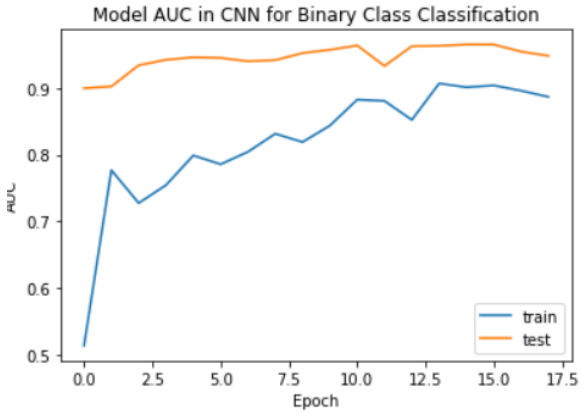


Figure 6.15: Training and Validation model auc for the CNN Architecture in different epochs

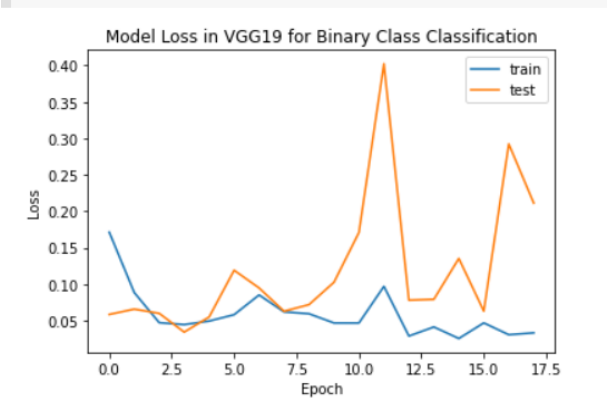


Figure 6.16: Training and Validation model loss for the VGG19 architecture in different epochs

6.3.3 Result analysis for DenseNet169 in Binary Class Classification

DenseNet169 Model Binary Class Classification: our deep learning based DenseNet169 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.069, 0.993, 0.996, 0.992, 0.994, and 0.996 respectively.

Accuracy, Loss, and AUC Diagram for DenseNet169 in Binary Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying DenseNet169 Architecture.

The Loss, Accuracy, and AUC diagram of CNN architecture is shown below in Figure 6.19, 6.20, and 6.21 respectively.

6.3.4 Result analysis for InceptionV3 in Binary Class Classification

InceptionV3 Model Binary Class Classification: our deep learning based InceptionV3 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.106, 0.994,

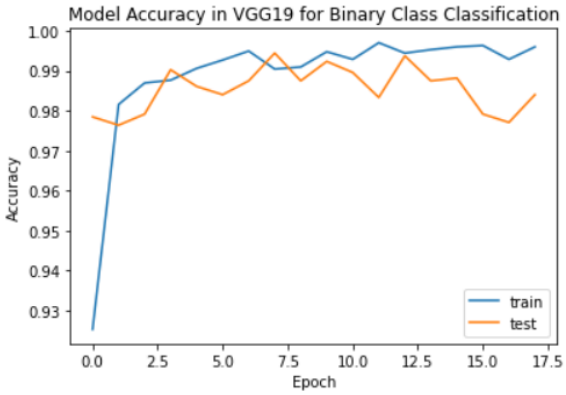


Figure 6.17: Training and Validation model accuracy for the VGG19 architecture in different epochs

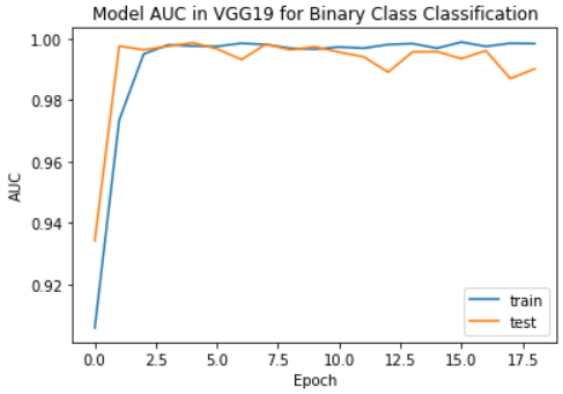


Figure 6.18: Training and Validation model auc for the VGG19 Architecture in different epochs

0.996, 0.993, 0.994, and 0.995 respectively.

Accuracy, Loss, and AUC Diagram for InceptionV3 in Binary Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying InceptionV3 Architecture.

The Loss, Accuracy, and AUC diagram of Inception architecture is shown below in Figure 6.22, 6.23, and 6.24 respectively.

6.3.5 Result analysis for ResNet50 in Binary Class Classification

ResNet50 Model Binary Class Classification: our deep learning based ResNet50 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.117, 0.994, 0.995, 0.995, 0.996, and 0.996 respectively.

Accuracy, Loss, and AUC Diagram for ResNet50 in Binary Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying

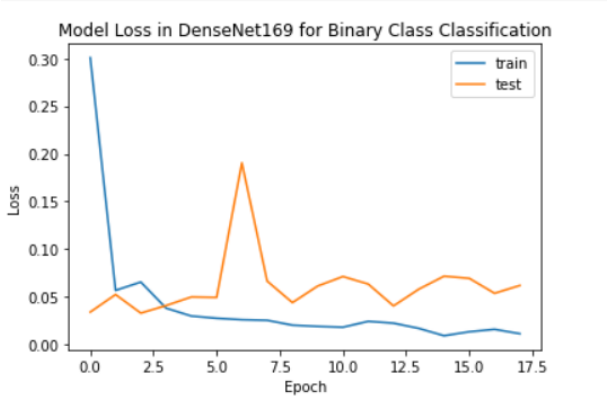


Figure 6.19: Training and Validation model loss for the DenseNet169 architecture in different epochs

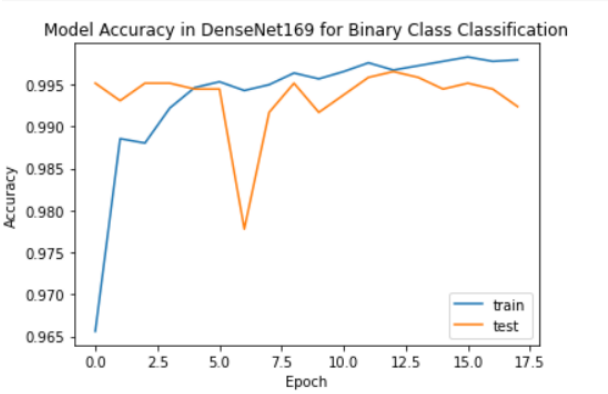


Figure 6.20: Training and Validation model accuracy for the DenseNet169 architecture in different epochs

ResNet50 Architecture.

The Loss, Accuracy, and AUC diagram of Inception architecture is shown below in Figure 6.25, 6.26, and 6.27 respectively.

6.3.6 Result analysis for ResNet50V2 in Binary Class Classification

ResNet50V2 Model Binary Class Classification: our deep learning based ResNet50V2 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.239, 0.991, 0.989, 0.993, 0.991, and 0.992 respectively.

Accuracy, Loss, and AUC Diagram for ResNet50V2 in Binary Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying ResNet50V2 Architecture.

The Loss, Accuracy, and AUC diagram of Inception architecture is shown below in Figure 6.28, 6.29, and 6.30 respectively.

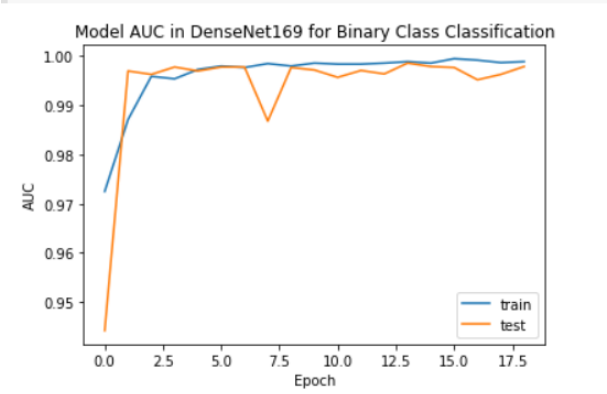


Figure 6.21: Training and Validation model auc for the DenseNet169 Architecture in different epochs

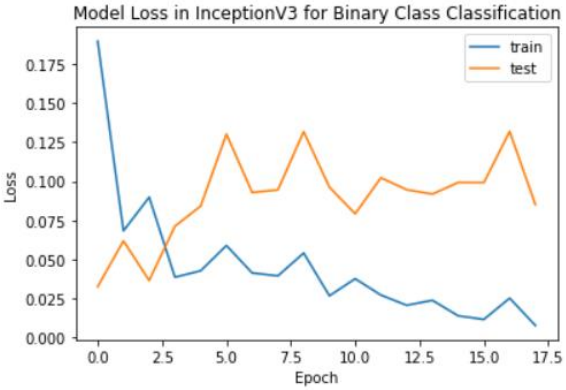


Figure 6.22: Training and Validation model loss for the InceptionV3 architecture in different epochs

6.4 Result analysis of Deep Learning Models in Multi Class Classification

6.4.1 Result analysis for CNN Model in Multi Class Classification

CNN Model Multi Class Classification: our deep learning based CNN achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.174, 0.946, 0.952, 0.946, 0.949, and 0.983 respectively.

Accuracy, Loss, and AUC Diagram for CNN in Multi Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying CNN Architecture.

The Loss, Accuracy, and AUC diagram of CNN architecture is shown below in Figure 6.31, 6.32, and 6.33 respectively.

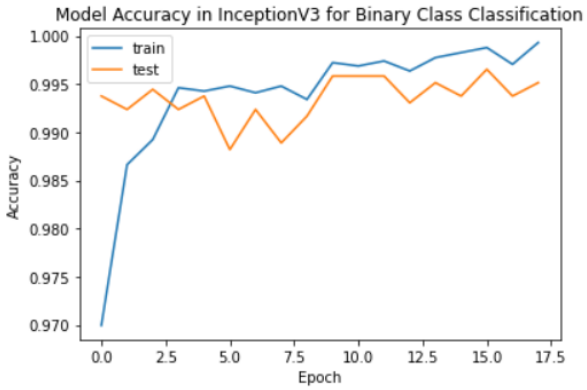


Figure 6.23: Training and Validation model accuracy for the InceptionV3 architecture in different epochs

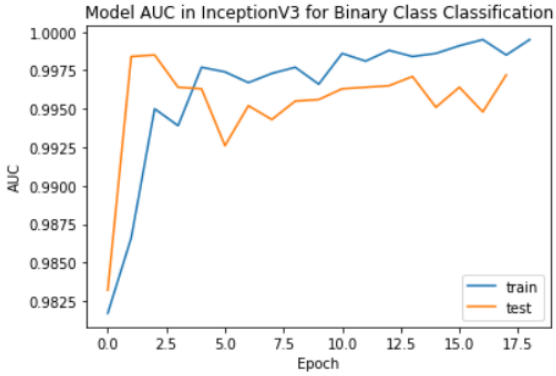


Figure 6.24: Training and Validation model auc for the InceptionV3 Architecture in different epochs

6.4.2 Result analysis for VGG19 in Multi Class Classification

VGG19 Model Multi Class Classification: our deep learning based VGG19 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.114, 0.980, 0.969, 0.928, 0.948, and 0.990 respectively.

Accuracy, Loss, and AUC Diagram for VGG19 in Multi Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying VGG19 Architecture.

The Loss, Accuracy, and AUC diagram of CNN architecture is shown below in Figure 6.34, 6.35, and 6.36 respectively.

6.4.3 Result analysis for DenseNet169 in Multi Class Classification

DenseNet169 Model Multi Class Classification: our deep learning based DenseNet169 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.108, 0.904,

Figure 6.25: Training and Validation model loss for the ResNet50 architecture in different epochs

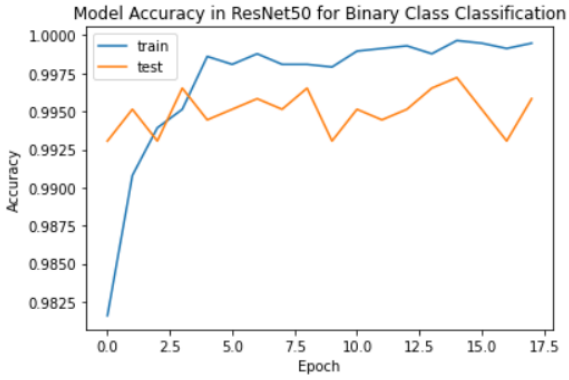


Figure 6.26: Training and Validation model accuracy for the ResNet50 architecture in different epochs

0.959, 0.951, 0.955, and 0.991 respectively.

Accuracy, Loss, and AUC Diagram for DenseNet169 in Multi Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying DenseNet169 Architecture.

The Loss, Accuracy, and AUC diagram of CNN architecture is shown below in Figure 6.37, 6.38, and 6.39 respectively.

6.4.4 Result analysis for InceptionV3 in Multi Class Classification

InceptionV3 Model Multi Class Classification: our deep learning based InceptionV3 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.120, 0.978, 0.938, 0.951, 0.945, and 0.990 respectively.

Accuracy, Loss, and AUC Diagram for InceptionV3 in Multi Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after

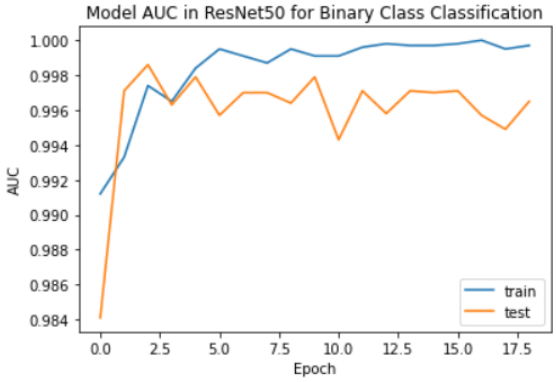


Figure 6.27: Training and Validation model auc for the ResNet50 Architecture in different epochs

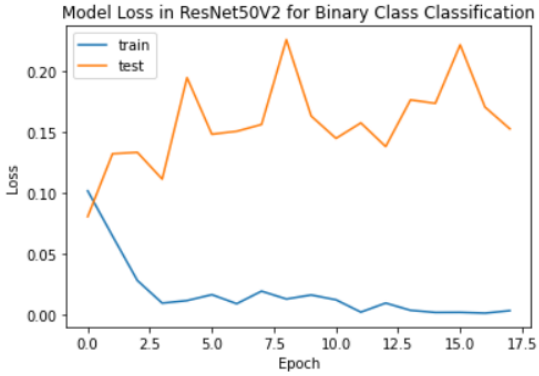


Figure 6.28: Training and Validation model loss for the ResNet50V2 architecture in different epochs

applying InceptionV3 Architecture.

The Loss, Accuracy, and AUC diagram of InceptionV3 architecture is shown below in Figure 6.40, 6.41, and 6.42 respectively.

6.4.5 Result analysis for ResNet50 in Multi Class Classification

ResNet50 Model Multi Class Classification: our deep learning based ResNet50 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.258, 0.998, 0.981, 0.850, 0.911, and 0.974 respectively.

Accuracy, Loss, and AUC Diagram for ResNet50 in Multi Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying ResNet50 Architecture.

The Loss, Accuracy, and AUC diagram of Inception architecture is shown below in Figure 6.43, 6.44, and 6.45 respectively.

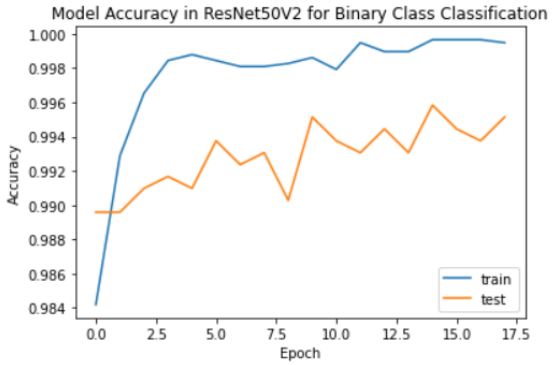


Figure 6.29: Training and Validation model accuracy for the ResNet50V2 architecture in different epochs

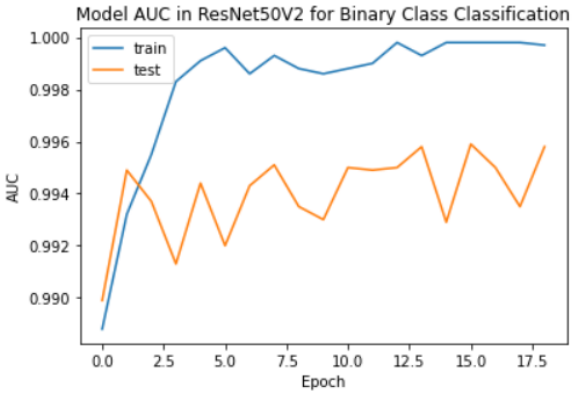


Figure 6.30: Training and Validation model auc for the ResNet50V2 Architecture in different epochs

6.4.6 Result analysis for ResNet50V2 in Multi Class Classification

ResNet50V2 Model Multi Class Classification: our deep learning based ResNet50V2 achieved the model loss, accuracy, precision, recall, f1-score, and AUC of 0.123, 0.974, 0.962, 0.924, 0.943, and 0.989 respectively.

Accuracy, Loss, and AUC Diagram for ResNet50V2 in Multi Class Classification: We achieved the following Accuracy, Loss, and AUC in both training set and validation set after applying ResNet50V2 Architecture.

The Loss, Accuracy, and AUC diagram of Inception architecture is shown below in Figure 6.46, 6.47, and 6.48 respectively.

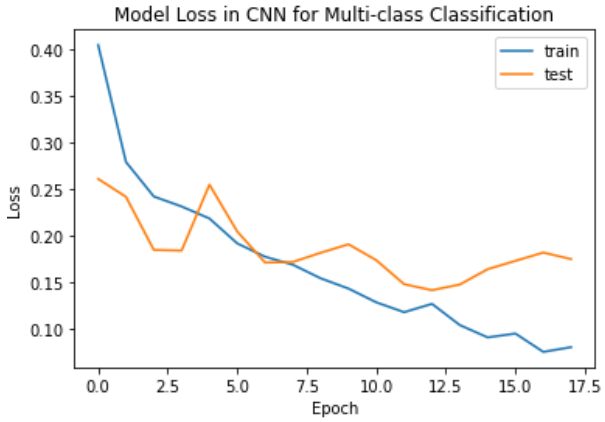


Figure 6.31: Training and Validation model loss for the CNN architecture in different epochs

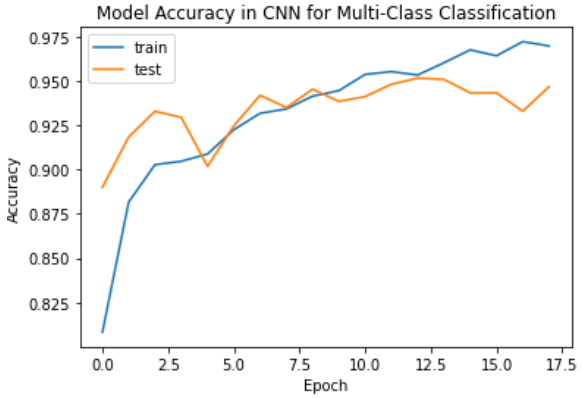


Figure 6.32: Training and Validation model accuracy for the CNN architecture in different epochs

6.5 Analysis and Comparison with DL Models in both Binary and Multi-Class Classification

6.5.1 Analysis and Comparison in Binary Class Classification

Here, from table 6.8 we can observe that, in the Binary Class Classification of DR detection in the models for ***Accuracy:*** ResNet50 is performing better than other Deep Learning models which is 99.45%, for ***Precision:*** DenseNet169 and InceptionV3 is performing better than other deep learning models which is 99.61%, for ***Recall:*** ResNet50 is performing better than other deep learning models which is 99.48%, for ***F1-Score:*** InceptionV3, and ResNet50 is performing better than other deep learning models which is 99.48%, and finally for ***AUC:*** DenseNet169, and ResNet50 is outperforming the other deep learning models which is 0.9964.

Then after analyzing the models for the binary class classification we can say that, the Model ResNet50 is overall better than the other deep learning models.

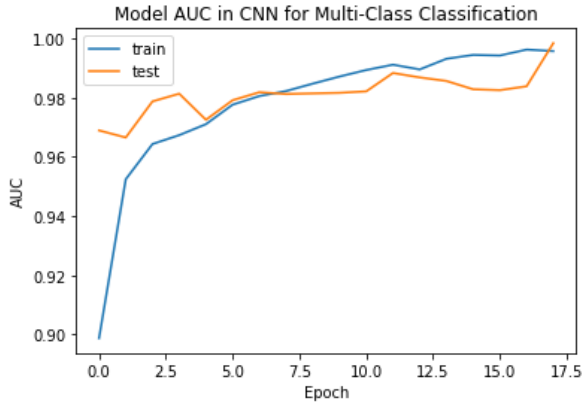


Figure 6.33: Training and Validation model auc for the CNN Architecture in different epochs

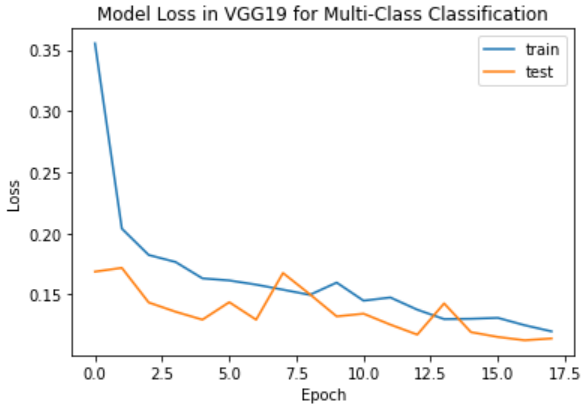


Figure 6.34: Training and Validation model loss for the VGG19 architecture in different epochs

6.5.2 Analysis and Comparison in Multi Class Classification

Here, from table 6.9 we can observe that, in the Multi Class Classification of DR detection in the models for ***Accuracy:*** ResNet50 is performing better than other Deep Learning models which is 99.86%, for ***Precision:*** DenseNet169 is performing better than other deep learning models which is 99.61%, for ***Recall:*** DenseNet169 is performing better than other deep learning models which is 95.95%, for ***F1-Score:*** DenseNet169 is performing better than other deep learning models which is 95.57%, and finally for ***AUC:*** ResNet50 is outperforming the other deep learning models which is 0.9964.

Then after analyzing the models for the Multi class classification we can say that, the Model DenseNet169 is overall better than the other deep learning models.

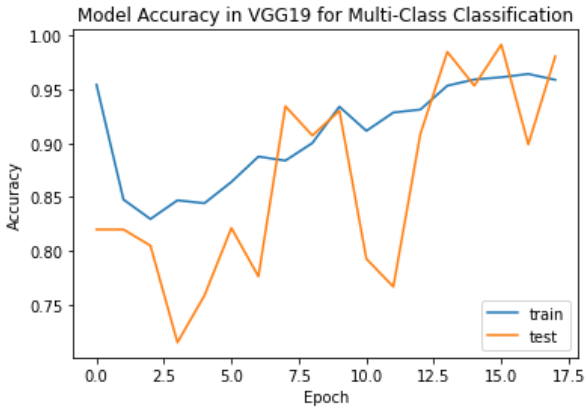


Figure 6.35: Training and Validation model accuracy for the VGG19 architecture in different epochs

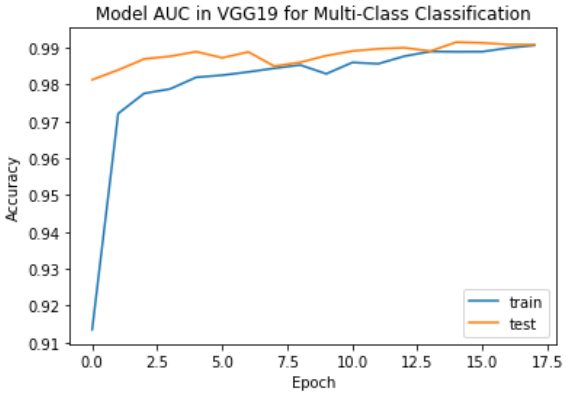


Figure 6.36: Training and Validation model auc for the VGG19 Architecture in different epochs

6.6 Confusion Matrix for the Deep Learning Model

As we have seen that for multi class classification DenseNet169 achieved better performance than the other Deep Learning models. That's why made our confusion matrix on the basis of DenseNet169. The figure of Confusion is shown below in figure 6.49.

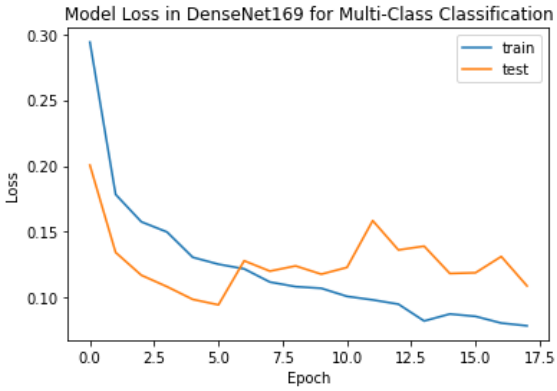


Figure 6.37: Training and Validation model loss for the DenseNet169 architecture in different epochs

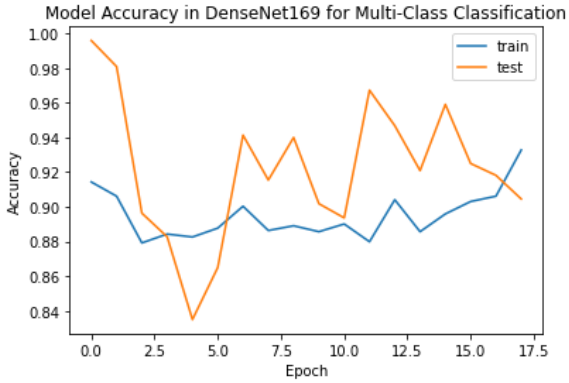


Figure 6.38: Training and Validation model accuracy for the DenseNet169 architecture in different epochs

6.7 Prediction of Diabetic Retinopathy using Deep Learning Model

Our models are also capable of finding the single class of diabetic retinopathy detection. In the following image we will see the diabetic retinopathy detection of Proliferative Diabetic Retinopathy (PDR), which belongs to Class 4 in figure 6.50. This prediction is done using the the best performing model of Diabetic retinopathy multi class classification which is DenseNet169.

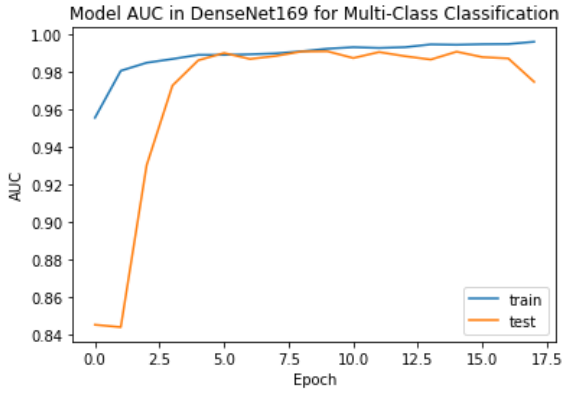


Figure 6.39: Training and Validation model auc for the DenseNet169 Architecture in different epochs

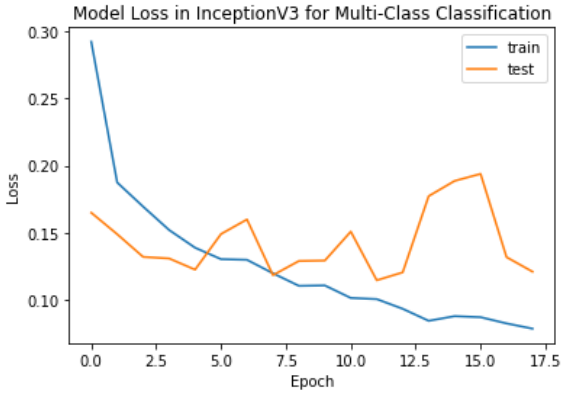


Figure 6.40: Training and Validation model loss for the InceptionV3 architecture in different epochs

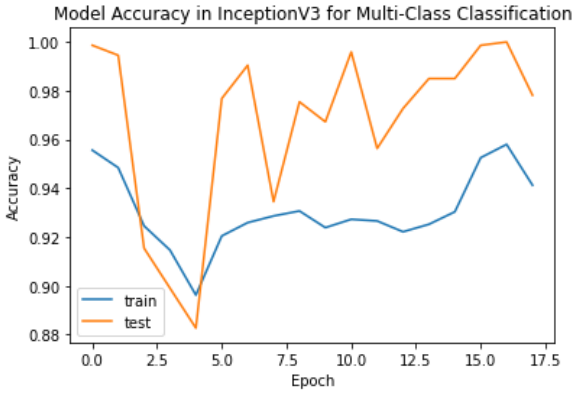


Figure 6.41: Training and Validation model accuracy for the InceptionV3 architecture in different epochs

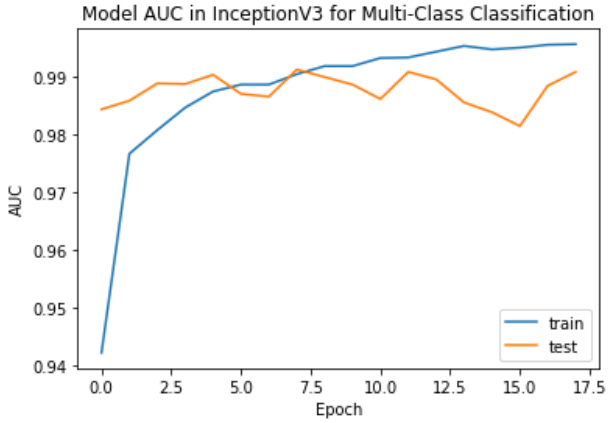


Figure 6.42: Training and Validation model auc for the InceptionV3 Architecture in different epochs

Figure 6.43: Training and Validation model loss for the ResNet50 architecture in different epochs

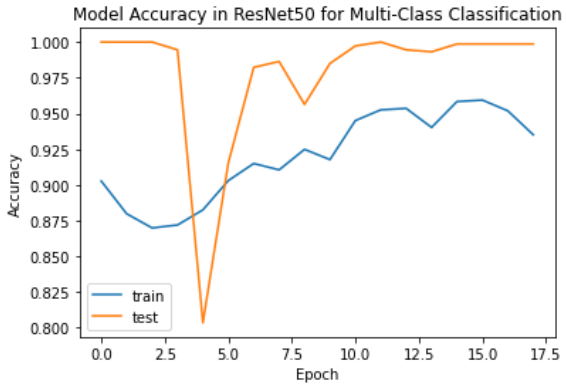


Figure 6.44: Training and Validation model accuracy for the ResNet50 architecture in different epochs

Figure 6.45: Training and Validation model auc for the ResNet50 Architecture in different epochs

Figure 6.46: Training and Validation model loss for the ResNet50V2 architecture in different epochs

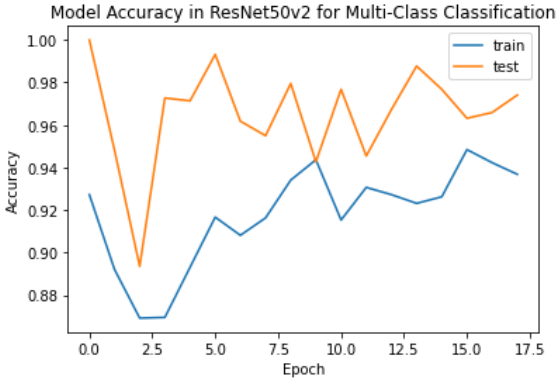


Figure 6.47: Training and Validation model accuracy for the ResNet50V2 architecture in different epochs

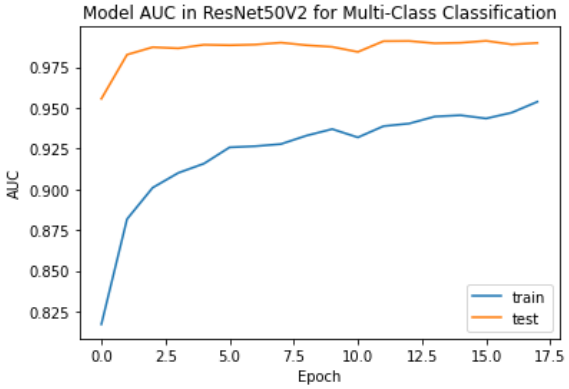


Figure 6.48: Training and Validation model auc for the ResNet50V2 Architecture in different epochs

Table 6.8: Result Comparison and analysis for DL Models in Binary Class Classification

Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*	*AUC*
CNN	87.91%	86.00%	91.49%	88.66%	0.9480
VGG19	98.48%	98.18%	98.95%	98.57%	0.9875
DenseNet169	99.38%	99.61%	99.22%	99.41%	0.9964
InceptionV3	99.446%	99.61%	99.35%	99.48%	0.9958
ResNet50	99.45%	99.48%	99.48%	99.48%	0.9964
ResNet50V2	99.10%	98.96%	99.35%	99.15%	0.9922

Table 6.9: Result Comparison and analysis for DL Models in Multi Class Classification

Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*	*AUC*
CNN	94.67%	95.26%	91.49%	94.64%	0.9838
VGG19	96.86%	97.19%	93.54%	95.33%	0.9908
DenseNet169	95.95%	99.61%	95.95%	95.57%	0.9914
InceptionV3	97.82%	93.89%	95.13%	94.50%	0.9909
ResNet50	99.86%	98.10%	85.05%	91.108%	0.9964
ResNet50V2	97.41%	96.29%	92.42%	94.31%	0.9899

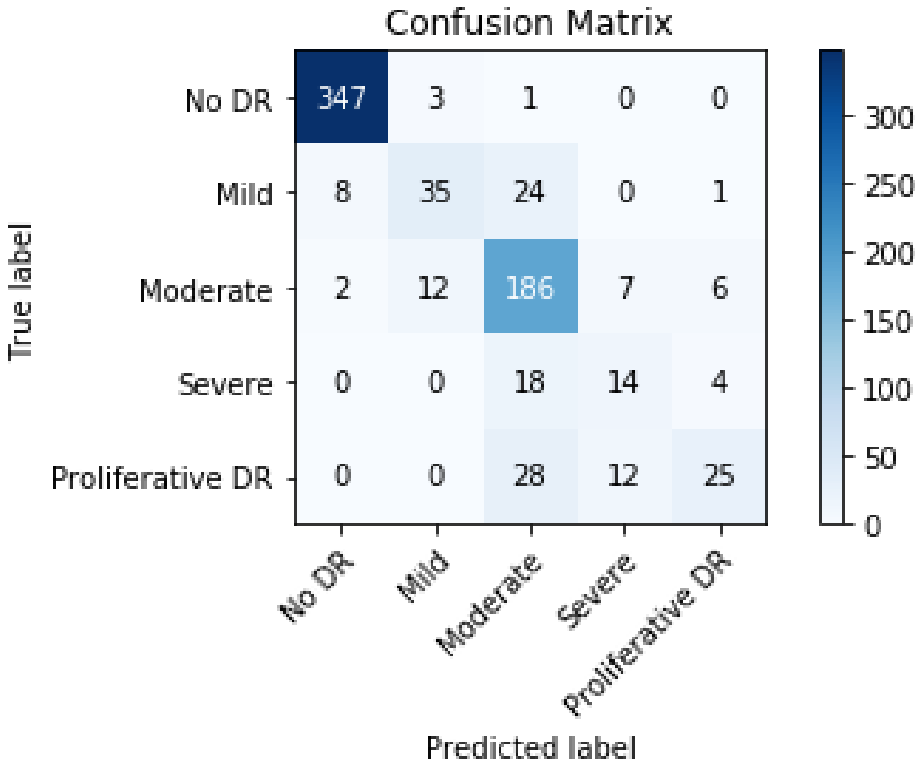


Figure 6.49: Confusion Matrix for DR detection using DL Model

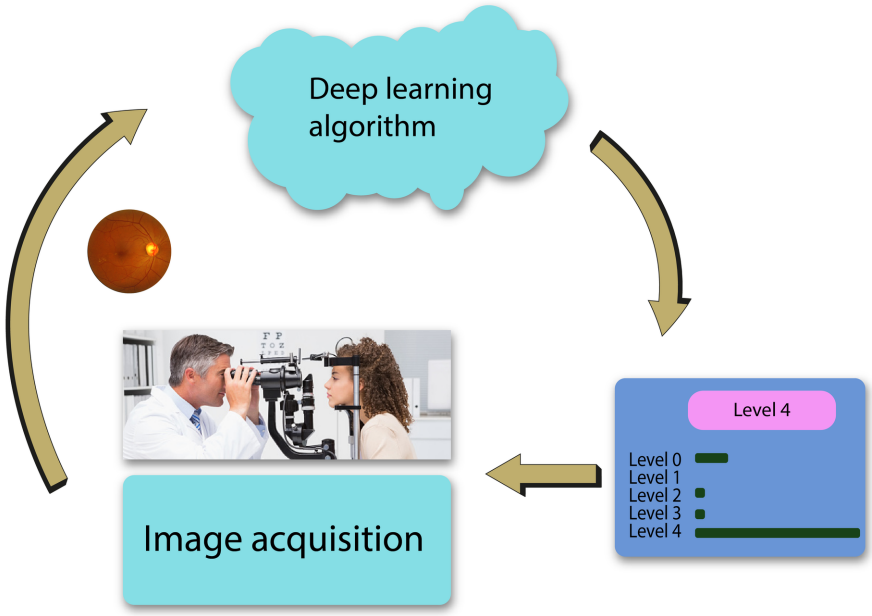


Figure 6.50: Prediction of DR detection for Single Class (Class 4: PDR)

Chapter 7

Outperforming State-of-the-Art Models

7.1 Comparison on machine learning based models

Table 7.1: Comparison with the existing techniques

Reference	Model	Accuracy	Average Recall	Average Precision	Average F1-Score
GT Reddy [32]	Random Forest	68%	67.5%	67.5%	67%
	KNN	65%	64.5%	64.5%	64.5%
	Decision Tree	55%	55%	55%	55%
Maliha et al [6]	Random Forest	63.63%	64%	64%	64%
	SVM	57.07%	57%	62%	53%
	KNN	64.50%	65%	65%	65%
Nur Izzati [33]	SVM	76.62%	80.81%	76.488%	78.42%
Proposed Model	Random Forest	84%	84%	84%	84%
	SVM	83%	82.5%	83%	82.5%
	Decision Tree	80%	80%	80%	80%

The suggested model is compared to three current models in terms of average sensitivity, precision, accuracy, and F1-score in table 7.1. The suggested model outperforms model in terms of sensitivity, accuracy, precision, and f1 score, demonstrating the resilience and breadth of our strategy. The suggested model is compared to the above reference models in terms of average sensitivity in Figure 7.1.

Our machine learning models got favourable performance than the current existing state-of-the art models because we have focused more on pre-processing than the other existing models. It is one of our prime reason to achieve the favourable performance in the machine learning section.

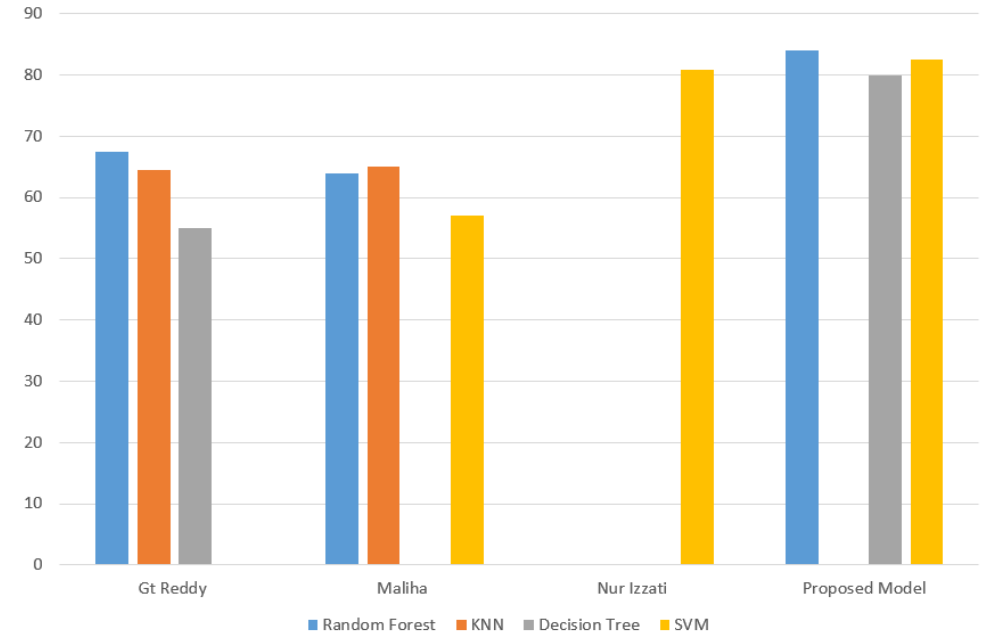


Figure 7.1: Comparison of proposed model with existing models in terms of Average recall

7.2 Comparison on deep learning based models in binary class classification

First we will see the binary class classification that means Healthy eye vs. DR effected eye model outperforming the existing state of the art models. which is shown in table 7.2. Here in table 7.2, A means Accuracy, Sn means Sensitivity/Recall. Here the state of the is shown where our models are outperforming the current existing models.

Table 7.2: Comparison with the existing models for Binary Class Classification in DL Models

Methods	*[18]*	*[14]*	*[13]*	*[16]*	*[19]*	*Our Result*
CNN	90.8% (A) 88.7% (Sn)	–	–	–	–	87.9% (A) 91.4% (Sn)
VGG19	–	97.4% (A)	–	–	–	98.4% (A)
DenseNet169	–	–	0.8703 for NP (AUC) 0.9435 for Microaneurysms (AUC) 0.9647 for leakages (AUC) 0.9653 for laser scars (AUC)	–	–	0.9964 (AUC)
InceptionV3	–	–	–	58.69% (A)	–	99.44% (A)
ResNet50	–	–	–	–	0.8140for NP (AUC) 0.9097 for Microaneurysms (AUC) 0.9585 for leakages (AUC) 0.9115 for laser scars (AUC)	0.9964 (AUC)

7.3 Comparison on deep learning based models in multi class classification

First we will see the multi class classification that means No DR, Mild DR, Moderate DR, Severe DR, Proliferative DR model outperforming the existing state of the art models. which is shown in table 7.3. Here in table 7.3, A means Accuracy, Sn means Sensitivity/Recall. Here the state of the is shown where our models are outperforming the current existing models.

Table 7.3: Comparison with the existing models for Multi Class Classification in DL Models

Methods	[24]	[23]	[20]	[22]	[21]	[19]	**Our Result**
CNN	88%-89%(A), 87%-89%(Sn)	–	–	–	–	–	94.67 %(A), 94.64 %(Sn)
VGG19	–	73.60% (A)	–	–	–	–	96.86%(A)
DenseNet169	–	–	90 % (A)	–	–	–	90.45 % (A)
InceptionV3	–	–	–	93.49 % (A), 96.93 %(Sn), 0.9905 (AUC)	–	–	97.817%(A), 93.88 %(Sn), 0.9909(AUC)
ResNet50					83%(A)		99.86%(A)
ResNet50V2						93%(A)	97.40%(A)

7.4 Analysis of the DL based outperforming models

From the section 7.2, and 7.3 we can see that, our models (CNN, VGG19, DenseNet169, ResNet50, ResNet50V2) is outperforming the existing state-of-art in both binary and multi class classifications of DR in terms of Accuarcy, Recall/Sensistivity, and AUC.

Hence, this improved models we improvised to detect diabetic retinopathy (DR) from retinal fundus images. This high accuracy, precision, recall/sensitivity, f1-score, and auc made these models to detect DR more efficiently.

Here, our deep learning based architecture and the transfer learning architectures have got better performance than the current existing state-of-the art model because we all know that, the deep learning models are self trained models but we are not only stacked on the self training of the deep learning models, we have also done a few pre-processing techniques before doing the model training using all the deep learning architectures. That's why we have achieved better performance than other state-of-the art models.

Chapter 8

Conclusion and Future Work

Early identification of DR is critical to preventing people from becoming impaired. Since the conventional way of detecting DR is time consuming, difficult, and expensive, numerous studies have been conducted to automate the detection process utilizing machine learning and deep learning techniques. This study provides a comprehensive analysis of Diabetic Retinopathy screening, assisting ophthalmologists in detecting DR early.The performance of the provided models are remarkable when compared to other current state of the arts. It has proposed ML models i.e. Random forest, Support Vector Machine ,and Decision tree along with DL based CNN architecture, VGG19 architecture, MobileNetV2 architecture, DenseNet169 architecture, InceptionV3 architecture, ResNet50 architecture, and ResNet50V2 architecture. We have showed comparison of various ML and DL models with their existing state of the arts. The proposed strategy paves the way for the development of an automated follow-up system for diagnosis and screening.

We will try to train our model with a larger dataset in the future. We will attempt to apply these models to a variety of Diabetic Retinopathy datasets. We'll also make an effort to improve the accuracy of the models we've already implemented and will try to outperform the existing state of the arts. We will extend the work and new features can be added to get better accuracy and for better classification. Detecting DR from fundus images has some limitations, including diagnosis with hazy retinal fundus images and detection in poor images. We will attempt to overcome these constraints through the use of more effective preprocessing techniques. In the near future, we will work to train our systems on a large sample size. We will try and apply the above models to a diverse array of dataset to detect Diabetic Retinopathy. Additionally, we will try to improve the performance of the models, and additional features may be introduced to boost performance and categorization.

Bibliography

[1] V. Pittu, S. R. Avanapu, and J. Sharma, ""diabetic retinopathy - can lead to complete blindness".," vol. 2, pp. 254–265., 01 2013.

[2] A. Navlani, "Random forest classification algorithm @ONLINE." https://www.datacamp.com/community/tutorials/random-forests-classifier-python, (Accessed: 20- Jun- 2021).

[3] A. Navlani, "Decision tree classification algorithm @ONLINE." https://www.datacamp.com/community/tutorials/decision-tree-classification-python, (Accessed: 20- Jun- 2021).

[4] A. Navlani, "Support vector machine classification algorithm @ONLINE." https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python, (Accessed: 20- Jun- 2021).

[5] "Diabetic retinopathy: Causes, symptoms, treatment @ONLINE." https://www.mayoclinic.org/diseases-conditions/diabetic-retinopathy/symptoms-causes/syc-20371611, (Accessed: 24- Aug - 2021).

[6] M. Maliha, A. Tareque, and S. S. Roy, *Diabetic retinopathy detection using machine learning*. PhD thesis, BRAC University, 2018.

[7] "Aptos 2019 blindness detection | kaggle @ONLINE." https://www.kaggle.com/c/aptos2019-blindness-detection/data, 2019 (Accessed: 23- Jun- 2021).

[8] "Messidor-2 dr grades @ONLINE." https://www.kaggle.com/google-brain/messidor2-dr-grades?select=messidor_readme.txt., 2018 (Accessed: 23- Jun- 2021).

[9] E. V. Carrera, A. González, and R. Carrera, "Automated detection of diabetic retinopathy using svm," in *2017 IEEE XXIV international conference on electronics, electrical engineering and computing (INTERCON)*, pp. 1–4, IEEE, 2017.

[10] K. K. Mohanty, P. K. Barik, R. C. Barik, and K. C. Bhuyan, "An efficient prediction of diabetic from retinopathy using machine learning and signal processing approach," in *2019 International Conference on Information Technology (ICIT)*, pp. 103–108, IEEE, 2019.

[11] M. K. Behera and S. Chakravarty, "Diabetic retinopathy image classification using support vector machine," in *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pp. 1–4, IEEE, 2020.

[12] "Quantification of retinal tissue damage @ONLINE." https://github.com/getsanjeev/retina-features/blob/master/report.pdf, (Accessed: 21- Jun- 2021).

[13] X. Pan, K. Jin, J. Cao, Z. Liu, J. Wu, K. You, Y. Lu, Y. Xu, Z. Su, J. Jiang, *et al.*, "Multi-label classification of retinal lesions in diabetic retinopathy for automatic analysis of fundus fluorescein angiography based on deep learning," vol. 258, pp. 779–785, Springer, 2020.

[14] N. E. M. Khalifa, M. Loey, M. H. N. Taha, and H. N. E. T. Mohamed, "Deep transfer learning models for medical diabetic retinopathy detection," vol. 27, p. 327, The Academy of Medical Sciences of Bosnia and Herzegovina, 2019.

[15] G. Kalyani, B. Janakiramaiah, A. Karuna, and L. N. Prasad, "Diabetic retinopathy detection and classification using capsule networks," pp. 1–14, Springer, 2021.

[16] T. Hattiya, K. Dittakan, and S. Musikasuwan, "Diabetic retinopathy detection using convolutional neural network: A comparative study on different architectures," vol. 7, pp. 50–60, 2021.

[17] W. Zhang, J. Zhong, S. Yang, Z. Gao, J. Hu, Y. Chen, and Z. Yi, "Automated identification and grading system of diabetic retinopathy using deep neural networks," vol. 175, pp. 12–25, Elsevier, 2019.

[18] P. Saranya and S. Prabakaran, "Automatic detection of non-proliferative diabetic retinopathy in retinal fundus images using convolution neural network," pp. 1–10, Springer, 2020.

[19] R. S. Salvi, S. R. Labhsetwar, P. A. Kolte, V. S. Venkatesh, and A. M. Baretto, "Predictive analysis of diabetic retinopathy with transfer learning," in *2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE)*, pp. 1–6, IEEE, 2021.

[20] G. Mushtaq and F. Siddiqui, "Detection of diabetic retinopathy using deep learning methodology," in *IOP Conference Series: Materials Science and Engineering*, vol. 1070, p. 012049, IOP Publishing, 2021.

[21] P. Lakshmi, P. NidhiBartakke, G. R. C. Rao, and D. Srikanth, "Diagnosis of diabetic retinopathy with transfer learning from deep convolutional neural network,"

[22] F. Li, Z. Liu, H. Chen, M. Jiang, X. Zhang, and Z. Wu, "Automatic detection of diabetic retinopathy in retinal fundus photographs based on deep learning algorithm," vol. 8, pp. 4–4, The Association for Research in Vision and Ophthalmology, 2019.

[23] T. Vijayan, S. M, K. A., and K. Balaguru, "Fine tuned vgg19 convolutional neural network architecture for diabetic retinopathy diagnosis," vol. 11, pp. 615–622, 10 2020.

[24] M. Shaban, Z. Ogur, A. Mahmoud, A. Switala, A. Shalaby, H. Abu Khalifeh, M. Ghazal, L. Fraiwan, G. Giridharan, H. Sandhu, *et al.*, "A convolutional neural network for the screening and staging of diabetic retinopathy," vol. 15, p. e0233514, Public Library of Science San Francisco, CA USA, 2020.

[25] "Clahe histogram eqalization – opencv @ONLINE." https://www.geeksforgeeks.org/clahe-histogram-eqalization-opencv/, (Accessed: 22- Jun- 2021).

[26] "Dilation @ONLINE." https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm, (Accessed: 20- Jun- 2021).

[27] "Opening @ONLINE." https://homepages.inf.ed.ac.uk/rbf/HIPR2/open.htm, (Accessed: 20- Jun- 2021).

[28] R. Lahoti, "Automatic detection of micro aneurysms from retinal image @ONLINE." https://cutt.ly/vme0DHG, (Accessed: 24- Jun- 2021).

[29] "Team, k. keras documentation: Image data preprocessing @ONLINE." https://keras.io/api/preprocessing/image/#imagedatagenerator-class, (Accessed: 23- Jun- 2021).

[30] V. Kurama, "A review of popular deep learning architectures: Resnet, inceptionv3, and squeezenet," (Accessed: 24- Nov - 2021).

[31] M. Rahimzadeh and A. Attar, "A modified deep convolutional neural network for detecting covid-19 and pneumonia from chest x-ray images based on the concatenation of xception and resnet50v2," vol. 19, p. 100360, 2020.

[32] G. T. Reddy, S. Bhattacharya, S. S. Ramakrishnan, C. L. Chowdhary, S. Hakak, R. Kaluri, and M. P. K. Reddy, "An ensemble based machine learning model for diabetic retinopathy classification," in *2020 international conference on emerging trends in information technology and engineering (ic-ETITE)*, pp. 1–6, IEEE, 2020.

[33] N. I. Ab Kader, U. Yusof, and S. Naim, "A study of diabetic retinopathy classification using support vector machine," *International Journal of Engineering Technology*, vol. 7, pp. 521–527, 01 2018.

Generated using Undegraduate Thesis LaTeX Template, Version 1.4. Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh.

This thesis was generated on Tuesday 4th January, 2022 at 5:21pm.