DIABETIC RETINOPATHY DETECTION USING TRANSFER LEARNING

A PROJECT REPORT

Submitted by

ARUN KRISHNA KV (17C013)
SABARISH S (17C077)
SOUNDARRAJAN T (17C092)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI – 15
(A Government Aided Autonomous Institution Affiliated to Anna University)

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2021

THIAGARAJAR COLLEGE OF ENGINEERING
MADURAI-15

(A Government Aided Autonomous Institution Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report "DIABETIC RETINOPATHY DETECTION USING TRANSFER LEARNING" is the bonafide work of **ARUN KRISHNA KV (17C013) , SABARISH S (17C077), SOUNDARRAJAN T (17C092)** who carried out the project work under my supervision during the Academic Year 2020-2021.

SIGNATURE

Dr. P. Chitra M.E., PhD.
HEAD OF THE DEPARTMENT
COMPUTER SCIENCE AND ENGINEERING
THIAGARAJAR COLLEGE OF ENGG.
MADURAI – 625 015

SIGNATURE

Dr. S. Mercy Shalinie M.E., PhD, PDF
PRINCIPAL & PROFESSOR
COMPUTER SCIENCE AND ENGINEERING
THIAGARAJAR COLLEGE OF ENGG.
MADURAI – 625 015

Submitted for the VIVA VOCE Examination held at Thiagarajar College of Engineering on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Acknowledgement

I wish to express my deep sense of gratitude to **Dr. S. Mercy Shalinie,** Principal of Thiagarajar College of Engineering for her support and encouragement throughout this project work.

I wish to express my sincere thanks to **Dr.P.Chitra**, Head of the Department of Computer Science and Engineering for her support and ardent guidance.

I owe my special thanks and gratitude to **Dr. S. Mercy Shalinie**, Department of Computer Science and Engineering for her guidance and support throughout our project.

I am also indebted to all the teaching and non-teaching staff members of our college for helping us directly or indirectly by all means throughout the course of our study and project work.

I extremely thank my parents, family members and friends for their moral support and encouragement for our project amidst this pandemic situation.

ABSTRACT

Diabetic retinopathy (DR) is one of the most common reasons for human vision loss in the working-age population of the world. The disease gets severe if it is not treated properly at its early stages. If DR is diagnosed early enough, vision deterioration may be delayed or even prevented. In this study, we propose an automatic grading system to detect the presence and severity of the Diabetic Retinopathy directly from fundus images via transfer learning. The proposed methodology consists of building a deep convolutional neural network on the top of RESNET50 for the classification of fundal images based on severity of the image. The system is developed by using a high-quality dataset of DR medical images provided at Kaggles' APTOS (ASIA PACIFIC TELE OPHTHALMOLOGY SOCIETY) challenge in which all fundus images are taken by Aravind Eye Hospital. We evaluate the model designed using 5 metrics. Results show that our model has a high accuracy of 93% with Quadratic Kappa Score of 95%. Meanwhile it has precision of 88%, Recall of 85% and F1 score of 86%. We also have developed a Web Application where end users can upload the fundus images of both eyes and get tested with the severity of their eyes. Our model provides consistent detection results with high accuracy and specificity instantaneously. Hence, this work helps ophthalmologists by providing insights during the diagnostic process. The application developed is hosted at the url: http://18.205.216.30:5000/ *using amazonec2@aws*.

Table of Contents:

LIST OF FIGURES

TABLES

LIST OF ABBREVIATIONS

DR	Diabetic Retinopathy
CNN	Convolutional Neural Network
t-SNE	t-Distributed Stochastic Neighbour Embedding
SGD	Stochastic Gradient Descent
rmsProp	Root Mean Square Propagation
Adam	Adaptive Moment Estimation
CV	Computer Vision
OS	Operating System
GUI	Graphical User Interface
GD	Gradient Descent
LR	Learning Rate

CHAPTER 1:

INTRODUCTION

Diabetic retinopathy (DR) is a long-term complication of diabetes that damages the retina. Notably, the risk of blindness in people with DR is 25 times higher than in healthy people; as a result, DR is the leading cause of blindness among people aged 20 to 65 years around the world. The damage to the retinal blood vessels gradually prevents light from passing through the optic nerves, making the Diabetic patient blind. India is said to be the diabetic capital of the world by 2030 with over 90 million people affected by it. Unfortunately, nearly 60 percent of them are subaltern. Only a small percentage of them are aware that if they have diabetes for a long time, they may develop a diabetic complication. In India, patients with DR often receive late invasive care and develop severe illness, leading to a poor prognosis and high medical costs. As a result, India has a much higher rate of extreme proliferative DR than any other country.
Furthermore, the blindness caused by DR is permanent, putting a heavy strain on families and community.

Current system for detection and treatment of DR is slow and requires a lot of work to be done manually (explained in detail in 3.1). Also, during medical emergencies such as the persisting pandemic situation, it is becoming more and more tough to book an appointment to the hospital for regular--non-emergency--check-ups. Hence an automatic system for the early detection of DR is of an urgent need in today's world.

Recent advancements in the field of image processing, produced a major breakthrough in computer vision. Image processing techniques find its application in all kinds of industries. From medical images to astronomical images, image processing plays a major role. In fact, the field of medical image processing has gone uphill in its combination with the techniques of artificial intelligence.

Deep learning, which is a subfield of artificial intelligence, has some groundbreaking advanced research in medical images. From detection of skin disease to detection of tumors in the brain cells, there are an endless number of applications. Processing medical images using techniques like classification helps us acquire different

inferences and also makes us progress towards turning manual visual examinations into automation.

Despite this, there are still issues with using CNNs in medical research. To begin with, obtaining sufficient real-world medical images, especially for some specialised diseases, is difficult. In addition, the availability of labelled medical data is usually minimal. Secondly, since DR features are so complex, they are likely to interact with other lesions, and DR's minute lesions are impossible to discern if image quality is poor. Fundus images are labelled by a manual operating procedure that is subject to subjectivity, according to medical journals. Finally, achieving high disease detection accuracy by training a single model with a set of medical image data and unavoidable image noise is difficult. As a result we go for the concept of transfer learning. Knowledge reuse is the central principle of transfer learning;the migration of big data to small data fields to resolve the problem of data and knowledge shortage in small data fields.

This report is organized as follows: Chapter 2 shows some of the existing works by doing literature survey. Chapter 3 explains Problem Definition and background/scope of the project. Chapter 4 lists the hardware and software requirements. Chapter 5 discusses the Control Flow Graph/System Design and Chapter 6 discusses the proposed methodology in a detailed manner. Chapter 7 walks us through the implementation part. Chapter 8 examines the experimentation results and execution part. Chapter 9 concludes the study and dictates about the future work of the project.

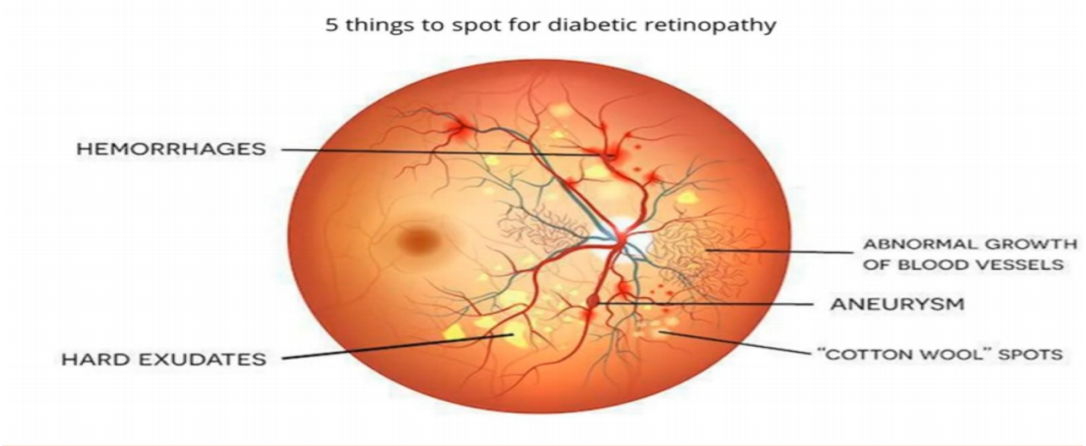
SEVERITY	STAGE NAME
No DR	No abnormalities
Mild DR	Microaneurysms only.
Moderate DR	More than just microaneurysms but less than severe NPDR
Severe DR	Severe intraretinal hemorrhages and microaneurysms in each of four quadrants ;definite venous beading in two or more quadrants;
Proliferative DR	neovascularization; vitreous/preretinal hemorrhage.

Table: 1 Severity and Stages of DR

Figure(1.1): People with NO DR

. Figure(1.2): People with DR

By seeing the above images one can see difference between image produced by normal eye and DR eye.



Figure(1.3): Features

CHAPTER 2:

LITERATURE SURVEY

(Xianglong Zeng, Haiquan Chen, Yuan Luo, Wenbin Ye)[3] The proposed model takes two binocular fundus images as inputs and learns their similarity to aid in prediction. The proposed binocular model achieves an area under the receiver operating curve of 0.951 with a training set of just 28 104 images and a test set of 7024 images, which is 0.011 higher than the current monocular model. It achieves a kappa rating score of 0.829.

(Jayant Yadav, Manish Sharma, Vikas Saxena)[4] The proposed model tries to solve the detection of diabetic retinopathy problem by using computer vision to not only diagnose the disease, but also to automate the process using a neural network to provide outcomes for a large number of patients in a limited amount of time. Overall testing accuracy of 75% was obtained.

(Karkhanis Apurva Anant, Tushar Ghorpade, Vimla Jethani)[5] proposes a technique for detecting texture and wavelet features using image processing and data mining. The results are obtained for an image from the DIARETDB1 standard database and analysed using sensitivity, specificity, and accuracy parameters. Their method has a 97.75 percent accuracy rate for binary classification of the disease, which can aid in the detection and prevention of diabetes..

(WIE ZHANG, JIE ZHANG, SHIJUN YANG, ZHENTAO GAO, JUNJIE HU, YUANYUAN CHEN)[2]proposes an automated DR identification and grading system called DeepDR is proposed.This detects the presence and severity of DR from fundus images using transfer learning and ensemble learning. Evaluation of the models is performed on the basis of validity and reliability using nine metrics.. Experiment results indicate the importance and effectiveness of the ideal number and combinations of component classifiers in relation to model performance.

(G. García, J. Gallardo, A. Mauricio, J. López, C. Del Carpio)[6]Proposed a method of applying CNN (Alexnet, VGGnet16, etc.) on right and left eye images separately.The preprocessing and augmentation phases were performed on the dataset to improve the contrast of images. They achieved the best results on VGG16 with no fully connected layer and achieved 93.65% specificity, 54.47% sensitivity, and 73.68% accuracy. However, DR stages were not explicitly classified in their work.

(S. Dutta, B. C. Manideep, S. M. Basha, R. D. Caytiles and N. C. S. N. Iyengar)[7] Here they used a dataset with three deep learning models (Feed Forward Neural Network (FNN), Deep Neural Network (DNN), and Convolutional Neural Network (CNN). A total of 35128 images were used in which 2000 images were taken for training(Validation split in the ratio 7:3). They applied many preprocessing steps (median, mean, Std deviation, etc.) and then trained their model on the training dataset. The Best training accuracy of 79.6% was obtained on DNN.

(Yuping Jiang, Huiqun Wu, Jiancheng Dong)[8]For this study, a total of 10,551 fundus image datasets were collected. First, histogram equalisation and image augmentation were applied to the images. The CNN was then developed and trained using the Caffe system. 8,626 photos were used to train our CNN models. Finally, the qualified CNN model's accuracy was checked by classifying 1,925 fundus images into DR and non-DR categories with accuracy of 75.70%

(Quang H. Nguyen, Ramasamy Muthuraman, Laxman Singh, Gopa Sen, Anh Cuong Tran, Binh P. Nguyen, Matthew Chua)[10] proposes to give an automated classification system that analyses fundus images and produces the severity of DR using ML models like VGG19, VGG16 and CNN. Their model gives 80 percent SENSITIVITY, 82 percent ACCURACY, 82 percent SPECIFICITY for classifying images into 5 levels from 0th level to 4th level, where 0th level is no DR and 4^{th} level is very high severity of DR.

(Athasart Markt Hewan, Noppadol Maneerat)[11] With the help of Image processing, this study focuses on detecting the retinal blood vessels - useful for diagnosis of diabetic retinopathy. Techniques like image enhancement, morphological operator, scaling, and filtering were applied to the green channel data to extract desired features. With a maximum specificity and accuracy of 99.6% and 96.8% respectively on the retinal images from the DRIVE database, the model does well to detect blood vessels.

(Zilin Zhang)[12] proposes a method where severity level of DR is identified by regression and extracting features from fundus images is achieved by Deep CNN based on EfficientNet-B3. The system shows the severity of DR with Quadratic Weighted Kappa 0.935.

(Aleshan Maistry, Anban Pillay, Edgar Jembere)[13] . Their last work did not address the imbalances in datasets that were used. In this idea, they have used data augmentation techniques for datasets balancing which improves the accuracy of automated diagnosis of DR. Finally the system produces an F-1 score of 0.8 and ACCURACY of 86.96 percent.

(Aiki Sakaguchi, Renjie Wu, Sei-ichiro Kamata)[14] proposes a Graph Neural Network based method that essentially has two features. The first feature focuses on image preprocessing to extract local regions capturing the lesions to filter out background noise. Second feature utilizes the GNNs unused for the fundus classification. The Dataset ‘Indian Diabetic Retinopathy Image Dataset (IDRiD)’ was taken from the "Diabetic Retinopathy: Segmentation and Grading Challenge" held at

the IEEE International Symposium in 2018. The model showed a 2.9% increase in accuracy over the conventional methods used in the contest.

(Chunyan Lian, Yixiong Liang, Rui Kang, Yao Xiang)[15] have used 4 factors of networking architectures, fine tuning, class imbalance and preprocessing. Their system generates 79 percent ACCURACY as opposed to Harry's scheme which have 75 percent ACCURACY

(Yuping Jiang, Huiqun Wu, Jiancheng Dong)[16] explores using CNNs to screen DR automatically. Around 11,000 images collected from the Kaggle fundus image dataset were first preprocessed with histogram equalization and image augmentation. Next, the neural network was modelled and trained with the Caffe framework. 80% of the dataset was used for training and the rest 20% for validating the performance of the model in classifying DR and non DR images. Their model achieved a validation accuracy of 75.70%.

(Yi-Wei Chen, Tung-Yu Wu, Wing-Hung Wong, Chen-Yi Lee)[17] proposes a deep CNN recognition pipeline which has a lightweight network called SI2DRNet-vl and also it has 6 functionality to enhance the detection performance. Our pipeline without fine tuning gives performs state on Messidor dataset associated with 2.48x fewer in total floating operations and 5.26x fewer in total parameters

(Abhishek Samantaa, AheliSahaa, Suresh ChandraSatapathya, Steven LawrenceFernandesb, Yo-DongZhangc)[18] surveys the use of transfer learning based CNN architecture. An imbalanced dataset of around 3000 training images and 400 validation images from color fundus photography was used. The model was trained on four classes namely no DR, mild DR, moderate DR and proliferative DR and achieved a Cohen's kappa score of 0.8836 on the validation set and 0.9809 on the training set.

CHAPTER 3

PROBLEM DEFINITION AND BACKGROUND

3.1 Existing approach:

At present, diabetic retinopathy (DR) is detected and treated among patients with diabetes by opthamologists with years of experience, since detecting DR, especially identifying the stage of the disease in the patient, is often not an easy task. Also, DR is tough to recognize among patients themselves, in the sense, an affected individual may not know he is affected until the condition becomes slightly alarming. Hence, voluntary hospital admission generally occurs very lately. So usually, large camps are organised to collect fundal images. In camps like these, thousands of people may turn out. After collecting the images, they will be referred to ophthalmologists who will have to manually go through these collected images. The people who have been identified to have diabetic retinopathy by the doctors will then be contacted for a further in depth diagnosis and treatment to hospitals. So as we can see here, the time needed for the completion of this entire process is

At present, there is no dashboard for patients or general doctors to upload ocular fundus images to detect and classify the stage of diabetic retinopathy in the patient, if he/she has it.

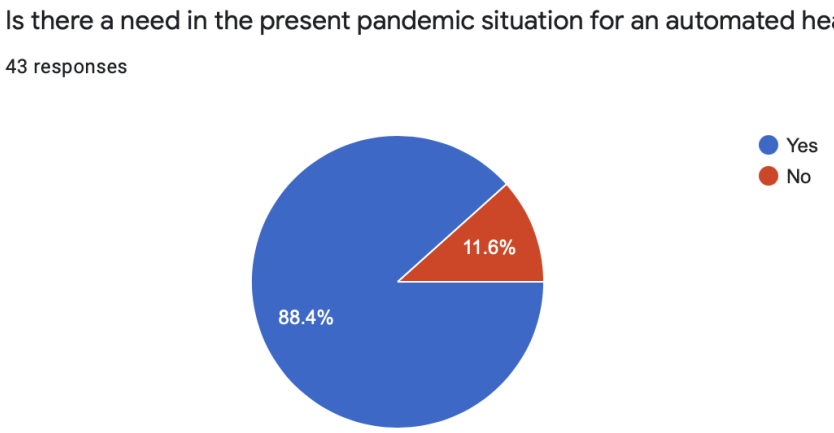
3.2 Proposed methodology:

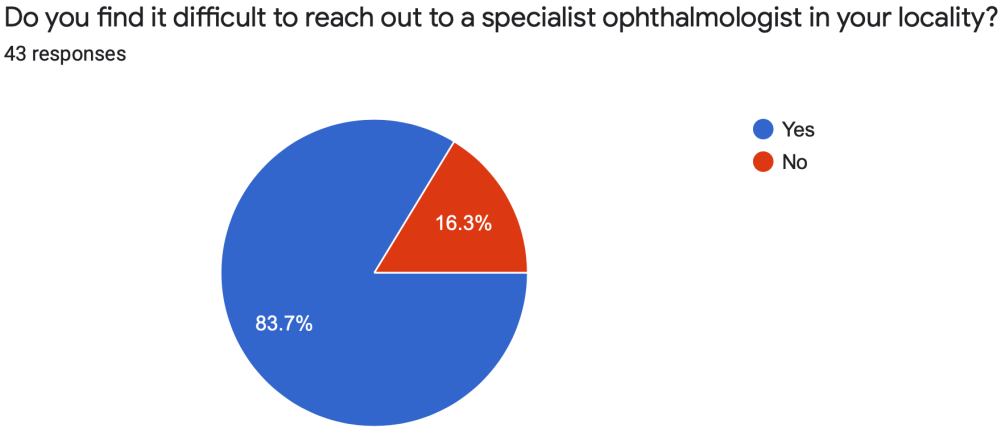
In this project, we have proposed and developed a methodology for the detection of DR from the fundal images by image processing and multi-stage-classification using Transfer Learning using CNN in which the system performs well in comparison with human evaluation metrics. The purpose of this project is to design a methodology which is **less complex to use** for the detection of DR and detects the disease affected images with high Accuracy, quadratic kappa score and Recall.Dataset for model training is taken from kaggle aptos challenge[1] ***Even the application is hosted on the Internet so that anyone anytime can make use of this solution.***

CHAPTER 4:
REQUIREMENT ANALYSIS AND SPECIFICATION

4.1 Requirements:

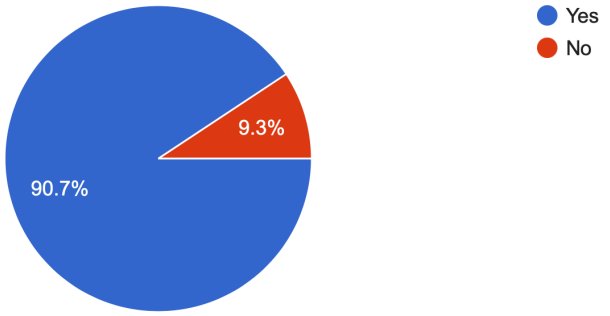
A survey was conducted to find out the requirements of end users of such an automated DR detection system.





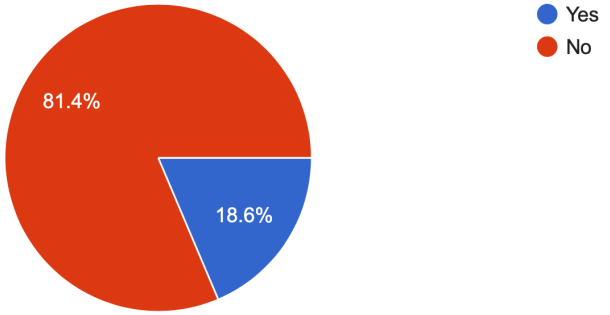
Do you find it time consuming to get diagnosed for any eye related diseases by manually going to a hospital?

43 responses



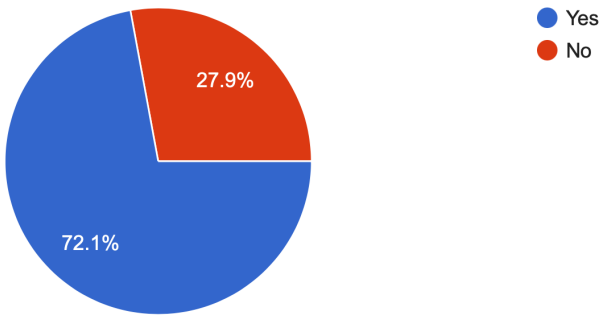
Have you come across any automated system to detect diabetic retinopathy?

43 responses



If there is a solution developed for detecting diabetic retinopathy online, will you be open to using it?

43 responses



Do you have any suggestions for an upcoming automated health disease prediction system ?

7 responses

In this pandemic situation, if we can provide maximum services to people through online,it will be very helpful.Besides, if we have a provision to collect (like an online survey through google forms) and store the health related datas of people and can store it securely,we can predict atleast some diseases and can make the people aware about and how to prevent it.

It should be free.

The automated system must be hosted publicly and easy accessibility must be provided for the people.

From the results obtained, it is evident that due to the less number of specialist ophthalmologists, the time consuming diagnosis methodology prevailing in hospitals, and the openness of people to usage of such systems, an Automated Grading system for Diabetic Retinopathy is very much useful for the general public in this arising pandemic condition.

4.2 Hardware Specification

This project has the following minimum hardware requirement
Processors : Intel(R) Core(TM)
Architecture : i5 or above.
CPU Speed : 2.30 – 2.40 GHz
Installed RAM : 8.00 GB
System type : 64-bit processor.

4.3 Software Specification

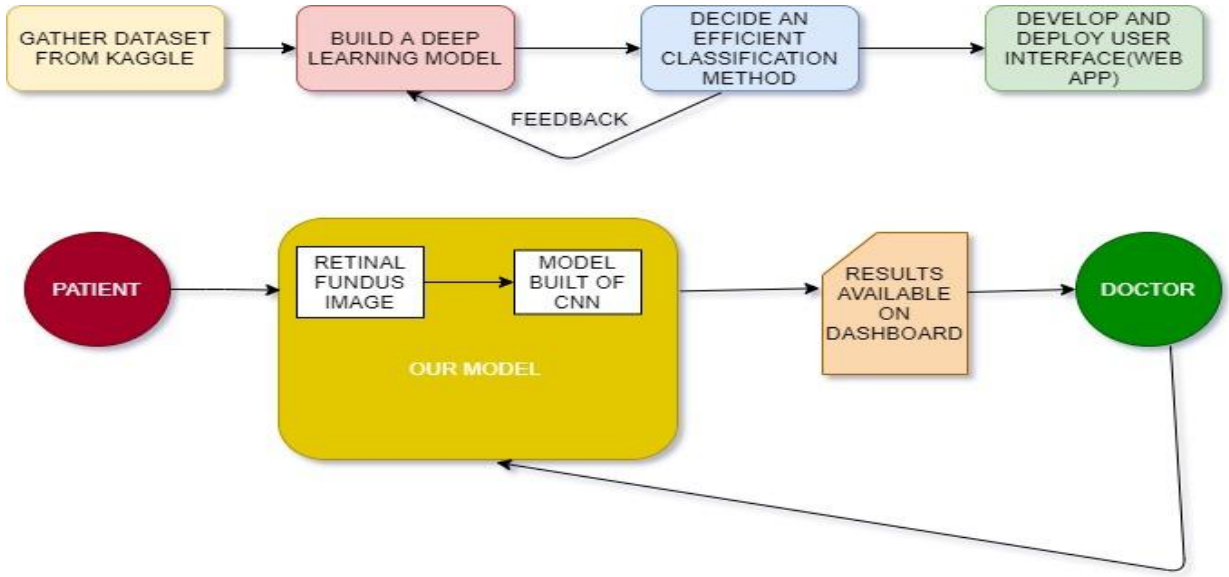
The software requirements of the project are : Operating System : Windows 8 and above/MAC OS/Linux
Tools : (i) Anaconda
(ii) Jupyter Notebook/ Google Colab.

4.4 Technologies used

Technologies used for this project are :
(i) Python
(ii) OpenCV
(iii) Keras – a library in python for implementing deep learning algorithms.
(iv)TensorFlow- model graphs, callback functions.
(v) Matplotlib and Seaborn for visualization.
(vi)Flask and its related libraries for backend development.
(vii)Html, CSS, JavaScript for frontend

CHAPTER 5
CONTROL FLOW GRAPH & SYSTEM DESIGN

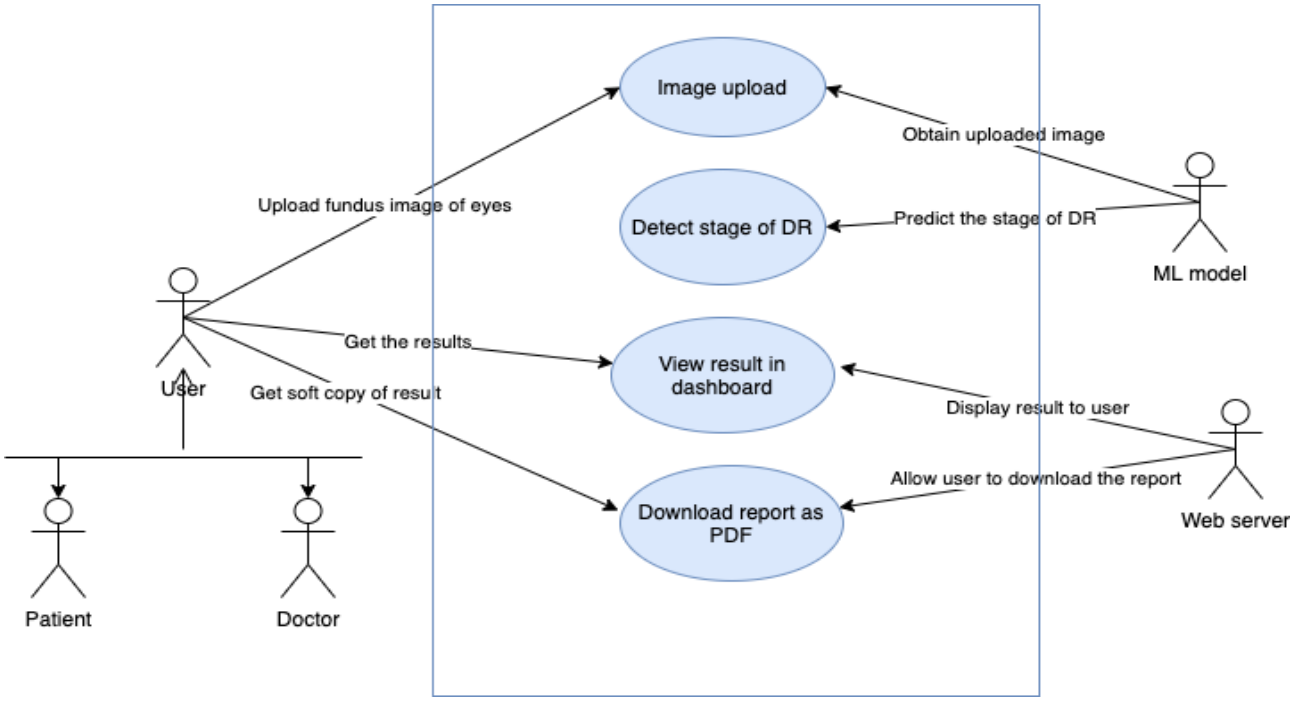
5.1 CONTROL FLOW DIAGRAM:



Figure(5.1) Control Flow Diagram

Fig(5.1) is the control flow diagram of our project. As we can see, patients themselves or doctors can upload the eye fundus image to our model, after which the model predicts the stage of DR and displays it to the patient. After that, the patient can download the result as a PDF if needed, and visit a qualified opthamologist. The work that went behind creating the model is depicted in the first part of the figure. As we can see, we started with the dataset gathering -> built a deep learning model ->decided a better classification method, and with any feedback received from this, modified our deep learning model, and finally created and deployed a web application for the end user to easily access the application.

5.2 USE CASE DIAGRAM:



Figure(5.2) Use Case Diagram

ACTORS:

1. **User - further divided into two - Patients and Doctors:** They are the end users of the application. They will be able to upload the eye fundus image to detect the presence and severity of DR, and get the results.

2. **ML Model:** The developed model, which will get the image from the user and predict the presence and severity of DR in it.

3. **Web server:** Hosts the application in the website and allows the user to interact with it.

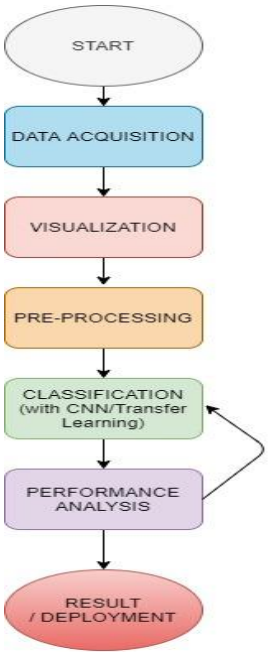
PROPOSED LIST OF FEATURES:

1. **Image Upload -** Allows the user to upload the retinal fundus image for prediction.
2. **Detect the stage of DR -** Predicts the stage of DR of the uploaded image.
3. **View result in dashboard -** The user can view his predicted stage of DR in the webpage.
4. **Download report as PDF -** User can download his result as PDF from the webpage.

CHAPTER 6
PROPOSED METHOD

The proposed methodology uses the techniques of image processing and deep learning. The methodology consist of the following steps:

1. Data acquisition
2. Visualization
3. Preprocessing
4. Classification
5. Performance analysis
6. Deployment

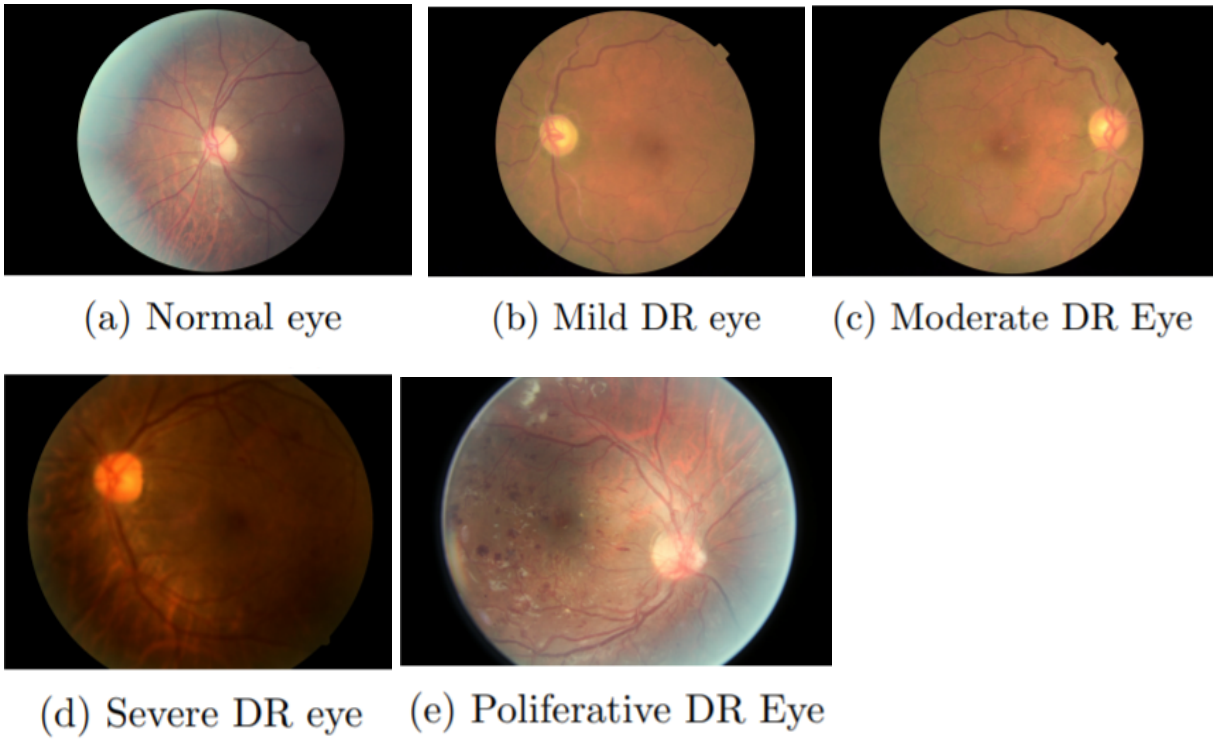


Figure(6.1): Flow diagram for procedure

6.1 DATA ACQUISITION

Ophthalmologists use different kinds of medical images like Fundoscopy(Ophthalmoscopy)images, A-scan images, B-scan images (both are eye ultrasound images) and Retinoscopy. Selection of a particular type of image is solely based on the purpose of treatment. For Diabetic Retinopathy Doctors make use of Fundoscopy images only. A funduscopy is a procedure that uses an ophthalmoscope to enable a health professional to see inside the eye's fundus and other structures (or funduscope). It can be performed as part of a regular physical examination or as part of an eye examination. It's important for assessing the retina's, optic disc's, and vitreous humor's wellbeing.

For our study, we considered fundus images only with which we did a multi-class classification on the severity. All fundus images were taken from Kaggle Aptos Challenge[1]. The label file consisted of eye_id and severity associated with it. Severity is ranged from 0-4 where 0 is No DR and 4 is Proliferic DR. Some of the fundus images are shown below in fig(6.2):



Figure(6.2) Stages of Eyes

6.2 DATA VISUALIZATION

Before starting the preprocessing techniques, it is necessary to visualize how many images are there under each severity stage. For visualization, we use two python libraries Matplotlib and Seaborn.

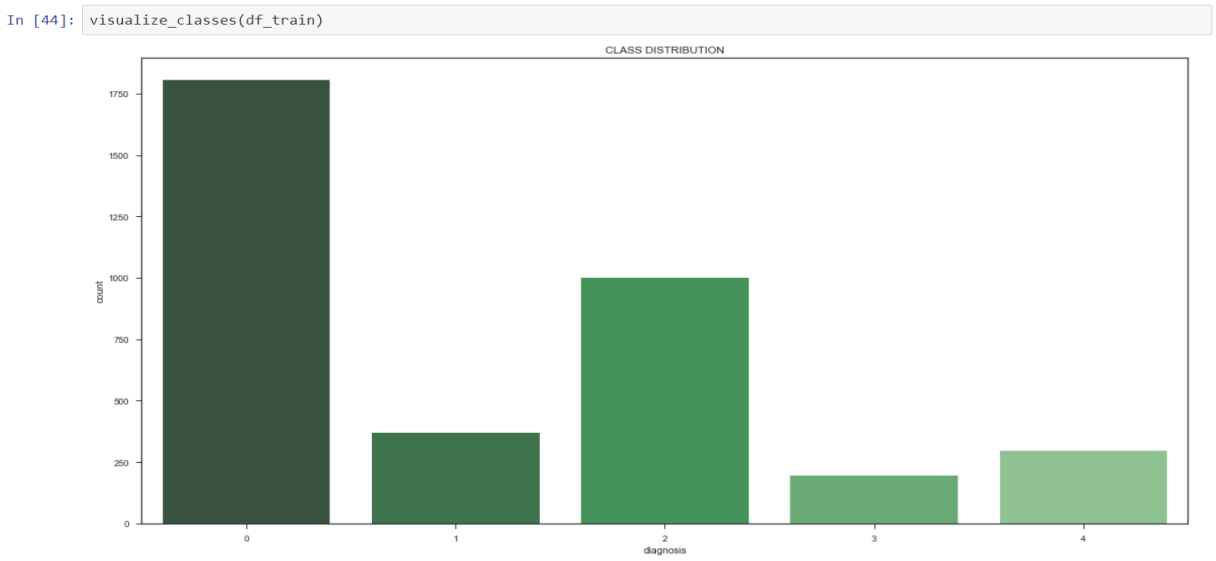


Figure:(6.3)- Visualization of dataset

From fig[6.4] we can infer details formulated in below table:

STAGE	Number of Images
stage-0	1760
stage-1	290
stage-2	1000
stage-3	250
stage-4	362

TABLE: 2 Number of images per stage

Total number of images available in dataset is : 3662

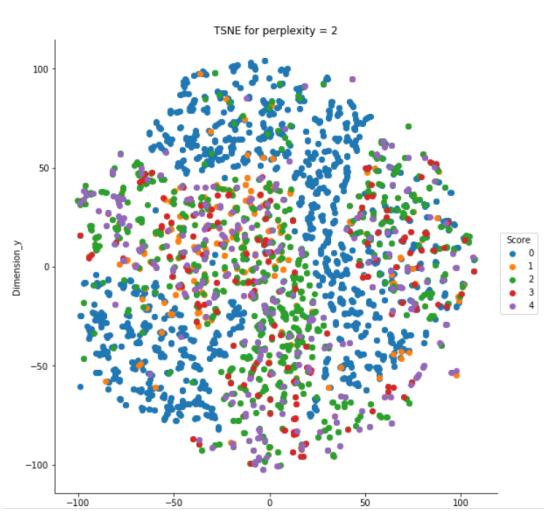
We use the library t-SNE[19] to visualize our high dimensional data. The t-Distributed Stochastic Neighbor Embedding (t-SNE) which is a non-linear as well as unsupervised technique is primarily used for exploration and visualization of high-dimensional data.

Non linearity in data can be found in this method. t-SNE gives a sense of how data is organised in a high-dimensional space.A second feature of t-SNE is the "perplexity" parameter, which can be tweaked to balance attention between local and global aspects of results. It is used in high-dimensional space to select the standard deviation σ_i of the Gaussian representing the conditional distribution.

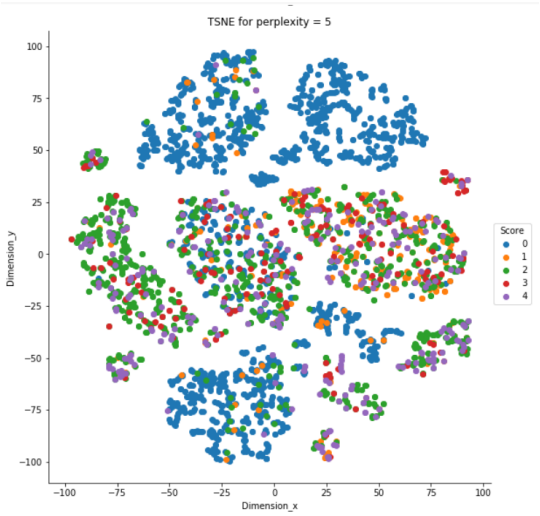
T-SNE[23] algorithm works as follows:

- Creating a joint probability distribution that reflects the data points' similarities
- Making a dataset of points in the target dimension and estimating their joint probability distribution.
- Changing the dataset in low-dimensional space using gradient descent so that the joint probability distribution describing it is as close as possible

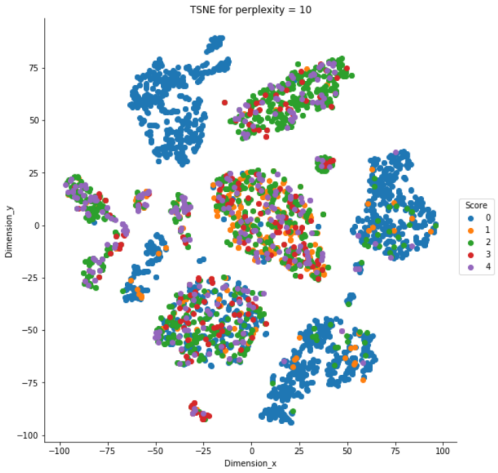
In certain ways, the parameter is a guess at the number of near neighbours each point has. The perplexity value has a complicated impact on the final images. Generally the perplexity value lies within the range of [2,50] and anything beyond that makes it more complex. Using t-SNE[23] we can visualize how easily multiple classes are easily separable.



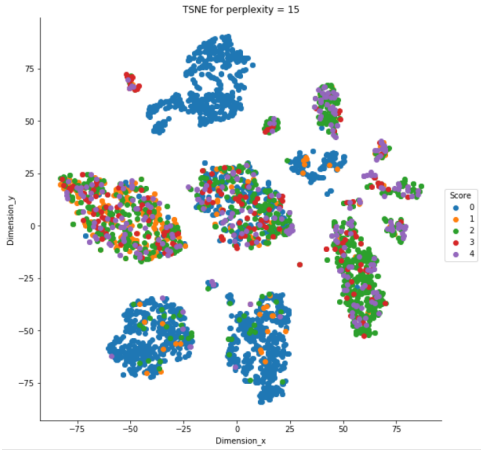
Figure(6.4 a)



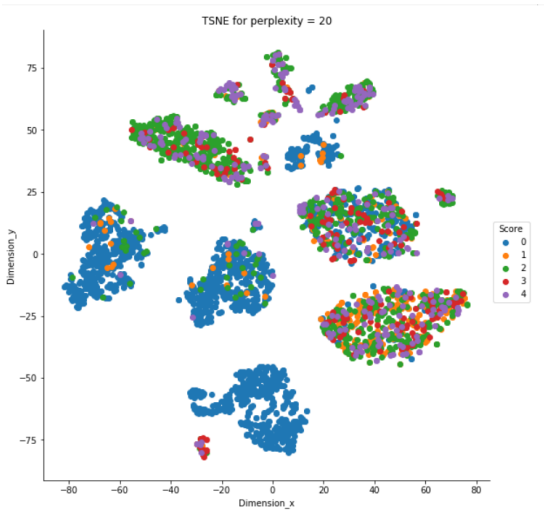
Figure(6.4 b)



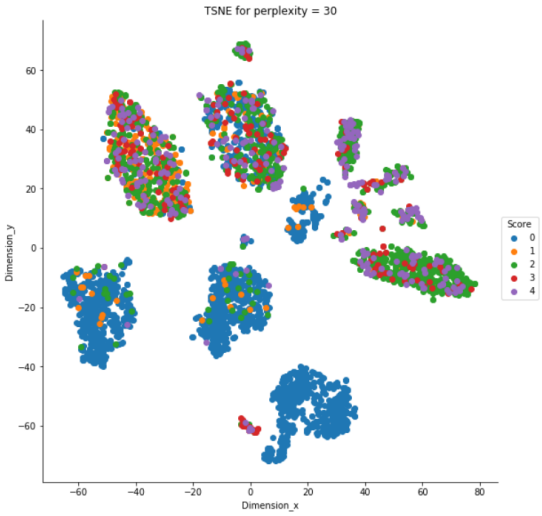
Figure(6.4 c)



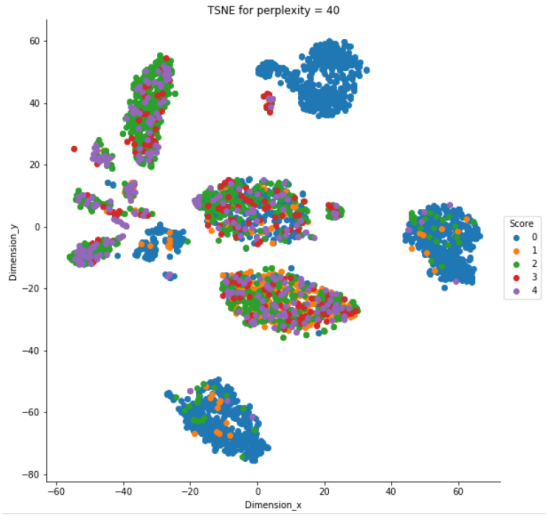
Figure(6.4 d)



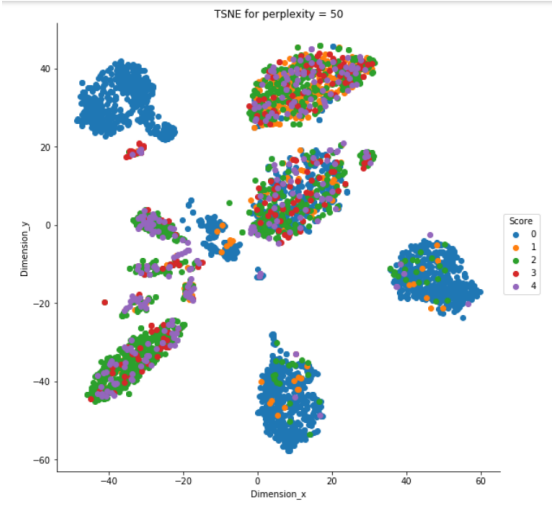
Figure(6.4 e)



Figure(6.4 f)



Figure(6.4 g)



Figure(6.4 h)

From these visualizations we can see at fig(6.4 a) where perplexity is 2, all classes are equally scattered, whereas at perplexity=50(fig 6.4 h) we infer that class-0 are easily separable from rest of classes as blue colored spots are distinguishable from rest of the colored spots quite easily and clearly. Which means, doing a binary classification(ie, whether a patient has DR or No DR) is pretty much easier with this dataset.

6.3 IMAGE PREPROCESSING

In most cases, the fundus images obtained are of low quality and noisy. The data that is fed into the machine learning model needs to be cleaned up and standardized and well suited as per our requirements. By doing so, the complexity of the learning model and performance metrics of the model can be increased.The preprocessing algorithm should be selected in such a way that the image does not lose any information.

Gaussian Blur

Gaussian Blur[21] is an image processing technique which uses Gaussian Function to blur an image. It has been majorly used to reduce noise and detailing. This blurring technique produces a smooth blur similar to that seen while viewing a picture through a transparent frame, as opposed to the bokeh effect created by an out-of-focus lens or the shadow cast by an object under normal lighting.

Mathematically, applying a Gaussian blur to a picture is the same as convolving the image with a Gaussian function. This can be conjointly called a two-dimensional Weierstrass transform. By contrast, convolving by a circle (i.e., a circular box blur) would accurately reproduce the bokeh effect. Since the Fourier transform of a Gaussian is another Gaussian, applying a Gaussian blur has the impact of reducing the photographs' high-frequency components.

In Gaussian blur, transformation is applied on each and every pixel using the Gaussian Function represented in equation(1,2):

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

Equation(1): for single dimension

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Equation(2): for two dimension

where x represents the distance from origin in horizontal axes, y represents the distance from origin in vertical and σ is the standard deviation. Time complexity of doing a Gaussian blur is :

$$O\left(w_{\text{kernel}} w_{\text{image}} h_{\text{image}}\right) + O\left(h_{\text{kernel}} w_{\text{image}} h_{\text{image}}\right)$$

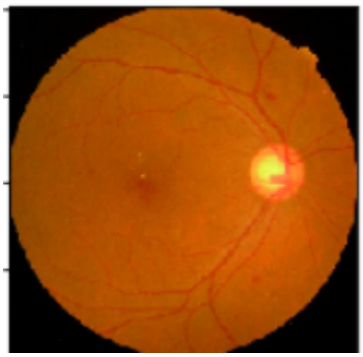
This gaussian blur is implemented using OpenCV library function cv.GaussianBlur().

The Function definition in python for Gaussian blur is shown in equation(3):

dst= cv.GaussianBlur(src,ksize,sigmaX,[dst],[sigmaY],[bias])

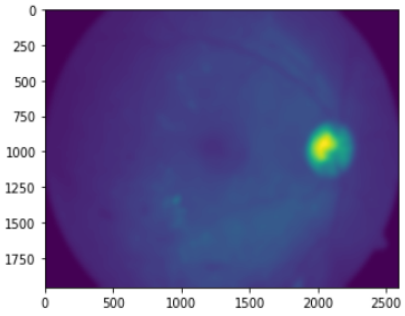
Equation(3): Gaussian blur python Function

Below image is typical implementation of GaussianBlur function on an image:



Figure(6.5): Normal image

```
In [98]: gaussian_image=cv2.GaussianBlur(img,(0,0),20)

In [99]: plot.imshow(gaussian_image)

Out[99]: <matplotlib.image.AxesImage at 0x28c78e5f4a8>
```



Figure(6.6) : Gaussian Blurred Image

But just with this Gaussian filtered image we cannot train our model because the image is very transparent and no features can be extracted. So we merge this gaussian blur image with the original image using opencv library in such a way that the image is now good to be fed into the neural network as internals are more visible.

The OpenCV library used for image blending is ***cv.addWeighted().*** This is an image addition technique in which the weights with which the image has to be combined/blended is specified unlike normal image addition. By making use of weights, transparency/ blending is achieved. Equation(4) used for image addition in this method is:

$$g(x) = (1 - \alpha) f_0(x) + \alpha f_1(x)$$

Equation(4): Image Addition with weights

Where $F_0(x)$ and $F_1(x)$ are the images that is provided and α denotes the weights that have been used. It's not necessary that the sum of weights should be 1. Below Equation(5) can be also applied.

$$dst = \alpha * img1 + \beta * img2 + \gamma$$

Equation(5): Image addition

Where γ is bias that is added.

6.4 CLASSIFICATION

Image classification is the process of classifying the image into the set of considered categories. In classification tasks, a set of classes are defined under which each image is classified. According to our study, we need to classify the fundus images into 5 classes of DR (No DR, Mild DR, Moderate DR, Severe DR, Proliferative DR). There are a number of machine learning algorithms available for classification tasks. According to recent advancements in the field of deep learning, CONVOLUTION NEURAL NETWORK (CNN) has an advantage over the machine learning algorithms. CNN extracts features automatically from the images fed using the convolution layers and classifies the image using the fully connected network at the end. Below we mentioned the layers present in our convolutional neural network.

1. **CONVOLUTION LAYER**

   This layer is used to extract features from the image. It is the first layer of CNN. It uses a number of filters associated with each layer. Each filter learns different features from the image and these features are fed to the subsequent layers. Convolution layer helps to decrease the dimensions of the input on the subsequent layers.

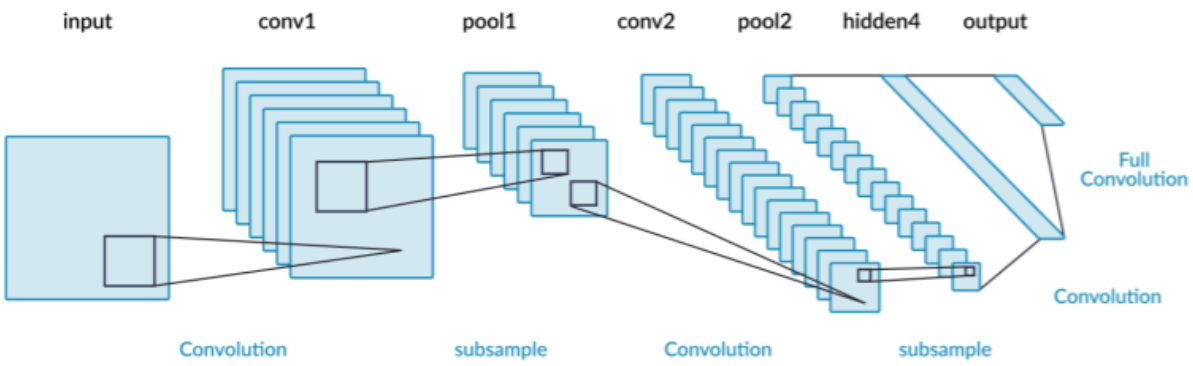
2. **POOLING LAYER**

   These layers help us greatly reduce the size of the features extracted from the previous layers. This layer often uses maximum function or average function to reduce the feature size. Mostly pooling layers occur after convolution layers in architectures.

3. **FULLY CONNECTED LAYER**

   This layer uses the outputs from the previous layers and feeds forwards it, just like a fully connected layer does. In the final layer, we get the probabilities of the image being in one of the target labels, from which we predict the category of the input image.

The CNN architecture is shown in Figure(6.7). This is how CNN works.



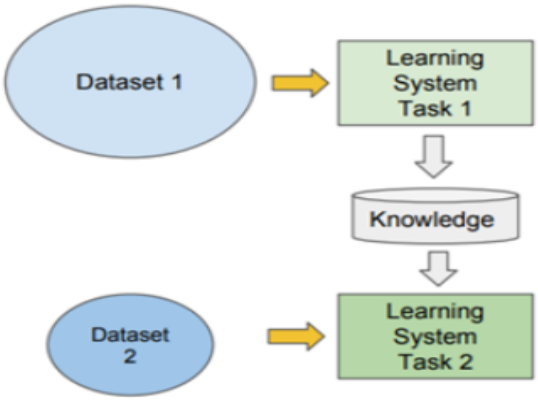
Figure(6.7) CNN Architecture

In addition to normal CNN architecture, the concept of TRANSFER LEARNING is also used in this study for better performance.

TRANSFER LEARNING

Transfer Learning[22] is a technique in which the knowledge gained while solving a problem is stored and is reused for different but related problems.

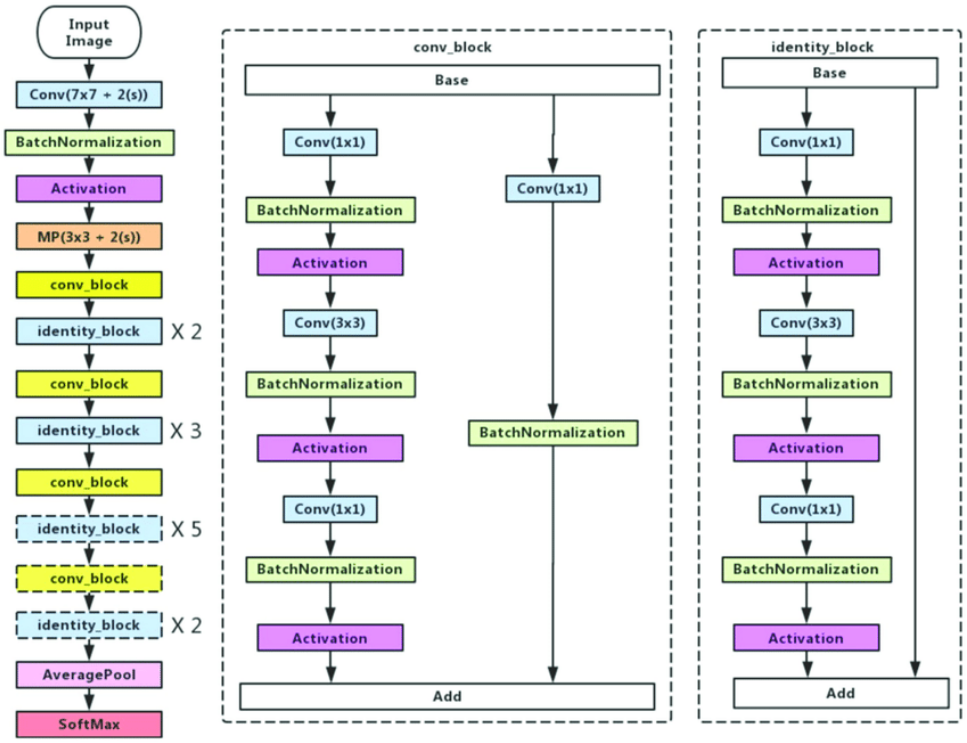
Below figure shows how transfer learning actually works.



figure(6.8) Transfer Learning Working

In the study performed, Dataset1 used is the IMAGENET dataset which has over 2Million images and over 1000 object categories. Learning System used is the RESNET 50 model.

ResNet-50 is a 50-layer deep convolutional neural network. It can be preloaded with a network that has been trained on over a million images from the ImageNet database. The network will classify images into 1000 different object categories, including keyboards, mice, pencils, and a variety of animals. Hence,the network has knowledge on a variety of rich feature representations. Below figure shows the architecture of the Resnet50 model.



Figure(6.9) RESNET ARCHITECTURE

As shown in the figure (6.9) , the knowledge obtained on training with ImageNet is stored and with the new dataset (fundus image), we retrain this ResNet50 with extra layers added to get indented results.

CHAPTER 7
IMPLEMENTATION

7.1 ALGORITHM

Step 1 : Collect the fundus dataset and categorize them as images with stages of DIABETIC RETINOPATHY accordingly.

Step 2: Apply preprocessing techniques (Gaussian Blur) along with the original image to generate a new preprocessed dataset.

Step 3: Apply Data Augmentation (ImageDataGenerator) techniques such as rotation, flipping, cropping etc.

Step 4: Split the dataset for training, testing and validation of the model adequately.

Step 5: Apply the filters and scalings to the images, so as to remove the noises present in the image, treating all images with the same pixel range and to balance the learning rate.

Step 6: Feed the preprocessed image to the convolutional neural network.

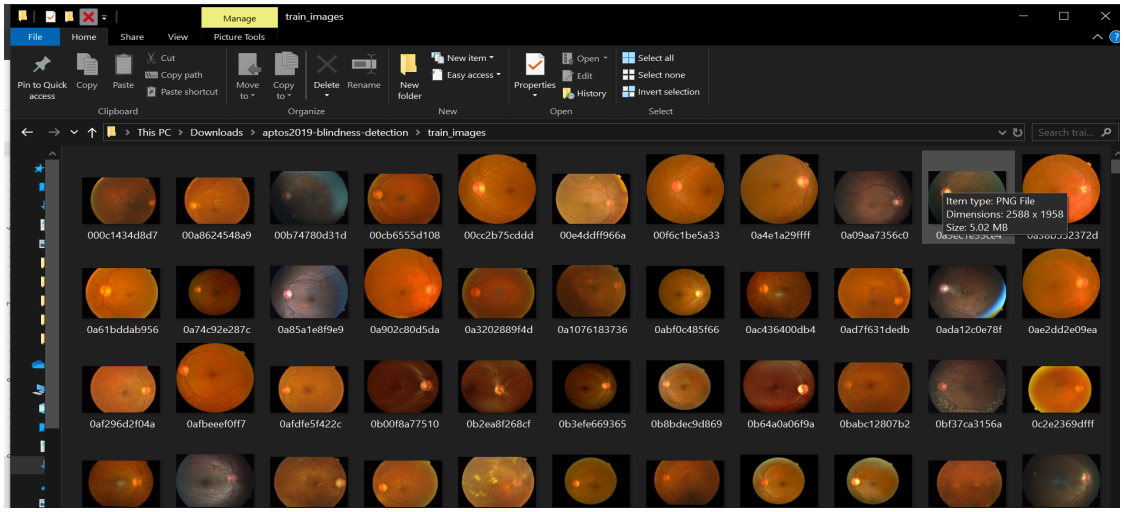
Step 7: Evaluate the model performance using metrics such as quadratic kappa, accuracy, precision, recall and F1-score.

Step 8: Repeat the training of model

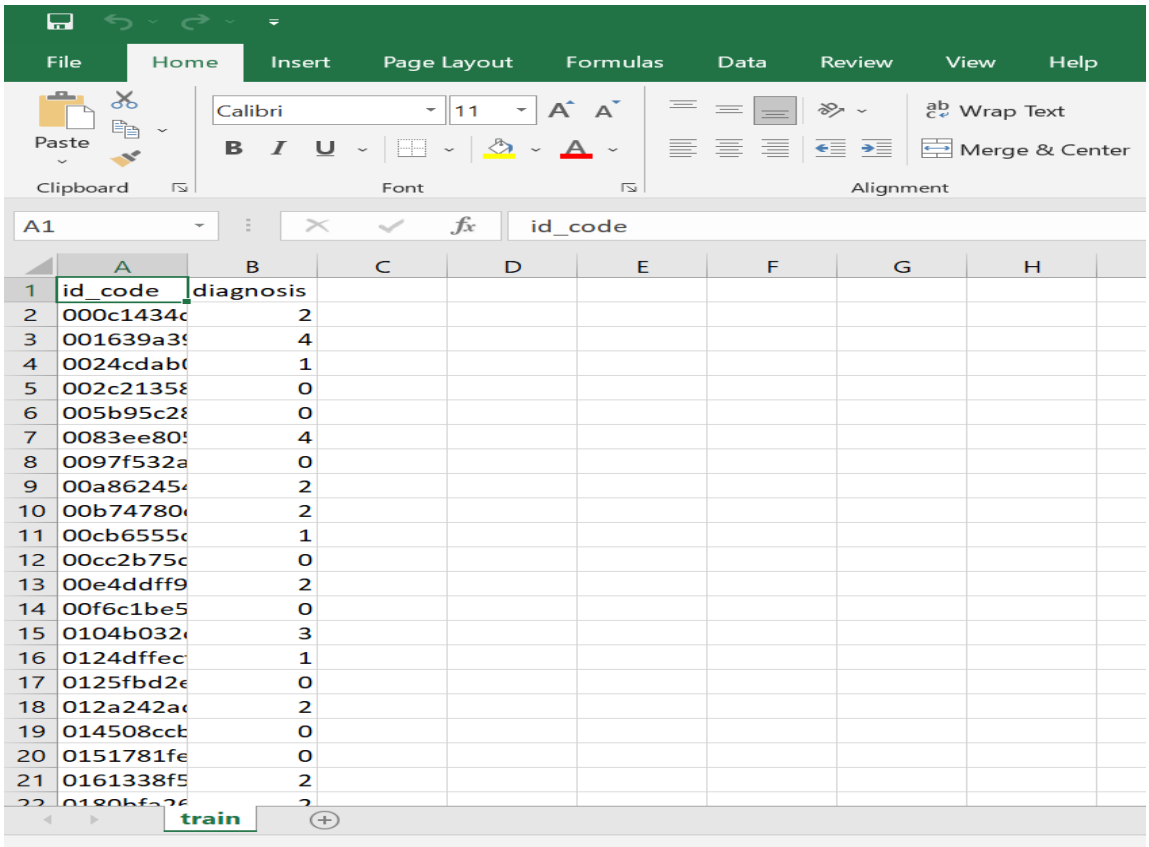
- without preprocessed images
- Transfer Learning using Resnet50 technique to get good comparative performance results.

7.2 DATASET PREPARATION

The dataset used in this study is collected from the Kaggle Asia Pacific Tele-Ophthalmology Society [APTOS][1] challenge. The dataset contains a total of 3662 fundus images. All the fundus images are labelled by Doctors of ARAVIND EYE hospital, Madurai, in accordance with their stages also in a csv file.



figure(7.1): DATASET



	A	B
1	id_code	diagnosis
2	000c1434c	2
3	001639a3	4
4	0024cdab	1
5	002c2135	0
6	005b95c2	0
7	0083ee80	4
8	0097f532a	0
9	00a86245	2
10	00b74780	2
11	00cb6555	1
12	00cc2b75c	0
13	00e4ddff9	2
14	00f6c1be5	0
15	0104b032	3
16	0124dffec	1
17	0125fbd2e	0
18	012a242a	2
19	014508ccb	0
20	0151781fe	0
21	0161338f5	2

Figure(7.2) : Labels

7.3 PREPROCESSING

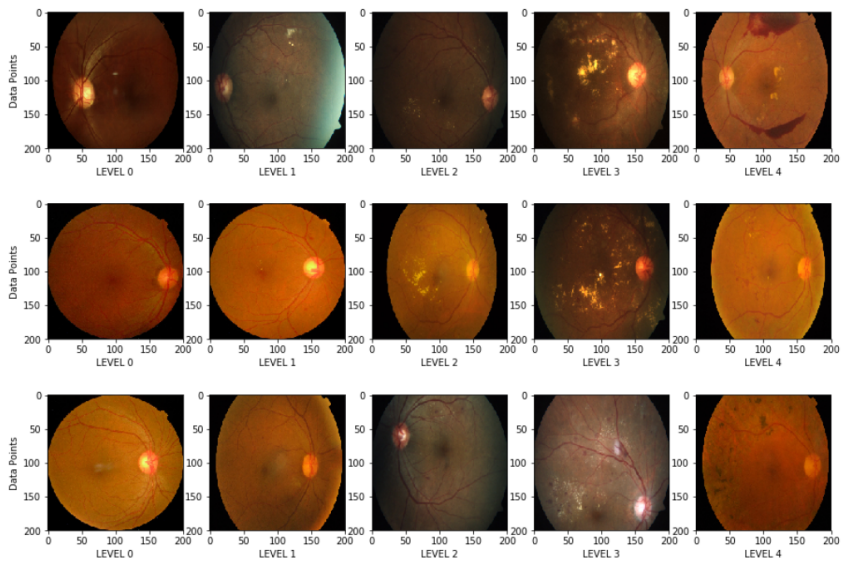
As it is discussed in Chapter 6.3, we use Gaussian Blur[19] as filtering technique and this filtered image is blended along with the original image to make a final preprocessed image for feeding into the model.

Initially the entire dataset was converted into gray scale images, as image noise can be minimized and there won't be a lot of pixel scaling. But the results (refer chapter 6) are not as good as expected.

```
In [28]: def showImages(df,classes,color_value):
             df=df.groupby('diagnosis',group_keys=False).apply(lambda df:df.sample(classes))
             df= df.reset_index(drop=True)

             plot.rcParams["axes.grid"]=False
             for i in range(classes):
                 x,y=plot.subplots(1,5,figsize= (15,15))
                 y[0].set_ylabel("Data Points")

                 df_arr=df[df.index.isin([i+(classes*0),i+(classes*1),i+(classes*2),i+(classes*3),i+(classes*4)])]

                 for j in range(5):
                     if color_value == 'gray':
                         image=grey(cv2.imread(df_arr.file_path.iloc[j]))
                         y[j].imshow(image,cmap=color_value)
                     else:
                         y[j].imshow(Image.open(df_arr.file_path.iloc[j]).resize((Size_of_image,Size_of_image)))
                     y[j].set_xlabel('LEVEL '+str(df_arr.diagnosis.iloc[j]))

                 plot.show()
```

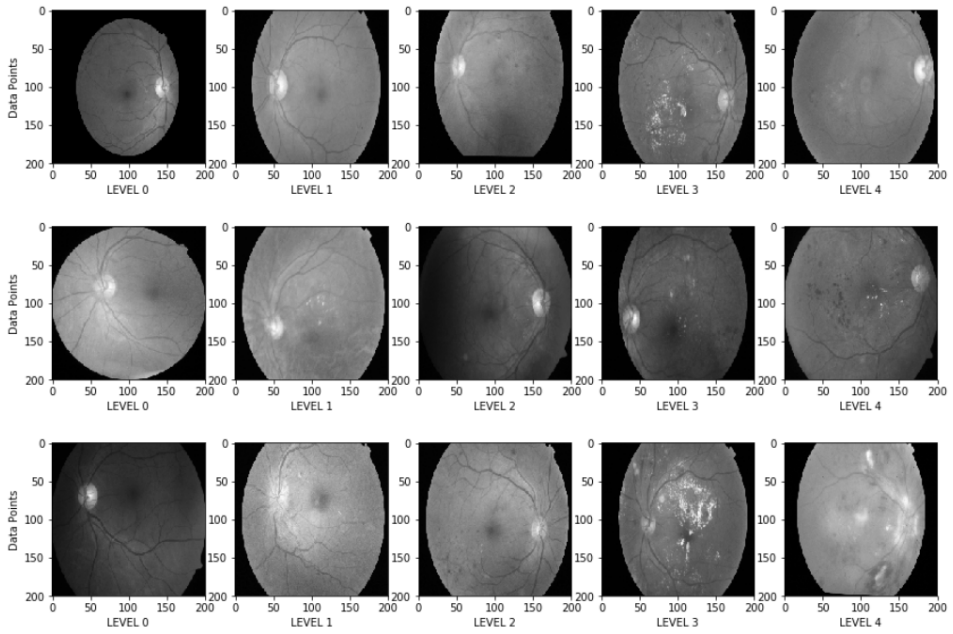
Figure(7.3) : code Snippet to list down images under each Class

```
In [29]: showImages(df_train,6,color_value=None)
```



Figure(7.4) : Normal Images that are cropped

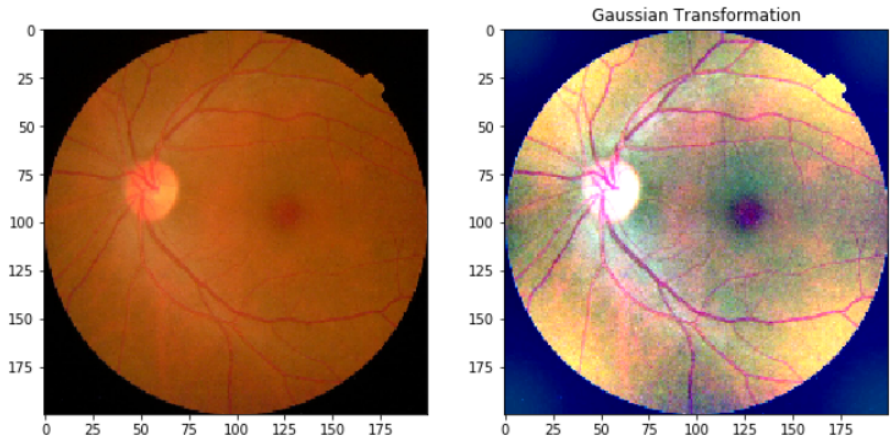
```
In [30]: showImages(df_train,6,color_value='gray')
```



Figure(7.5) : Grey scaled preprocessed image

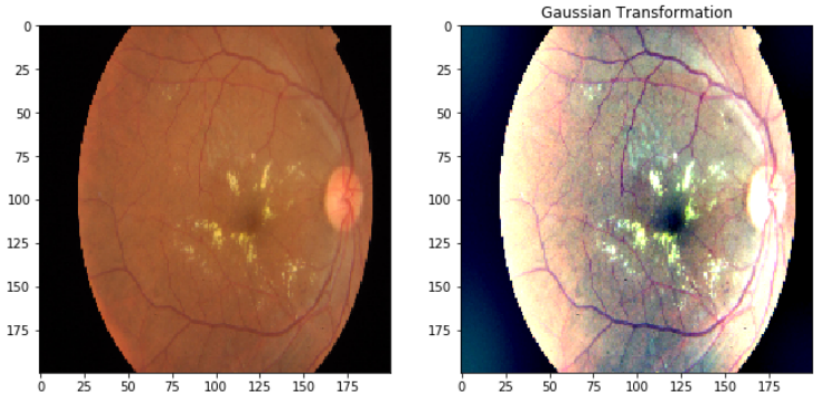
Since the results obtained with these 2 color-scales were not satisfactory, the above mentioned techniques were used.

```
In [40]: final_image=cv2.addWeighted(img,4,gaussian_image,-4,128)

In [44]: x,y=plot.subplots(1,2,figsize=(11,11))
         y[0].imshow(img)
         y[1].imshow(final_image)
         plot.title('Gaussian Transformation')
         plot.show()
```



Figure(7.6): Gaussian Transformed Image

```
In [34]: gaussian_image2=cv2.GaussianBlur(img,(0,0),30)
         final_image2=cv2.addWeighted(img,4,gaussian_image2,-4,128)

In [64]: x,y=plot.subplots(1,2,figsize=(11,11))
         y[0].imshow(img)
         y[1].imshow(final_image2)
         plot.title('Gaussian Transformation')
         plot.show()
```



Figure(7.7) : Uncropped Preprocessed Image

From figure 7.7, it can be noticed that due to cropping of images, some edges of the images are lost. This can significantly affect the result, as there could be some exudates, aneurysms etc. So to make a circular crop, the method suggested by Ratthachat Chatpatanasiri et al[20]was very helpful. Two major points were:

- Reducing lighting-condition effects
- Cropping uninformative area

In [15]:

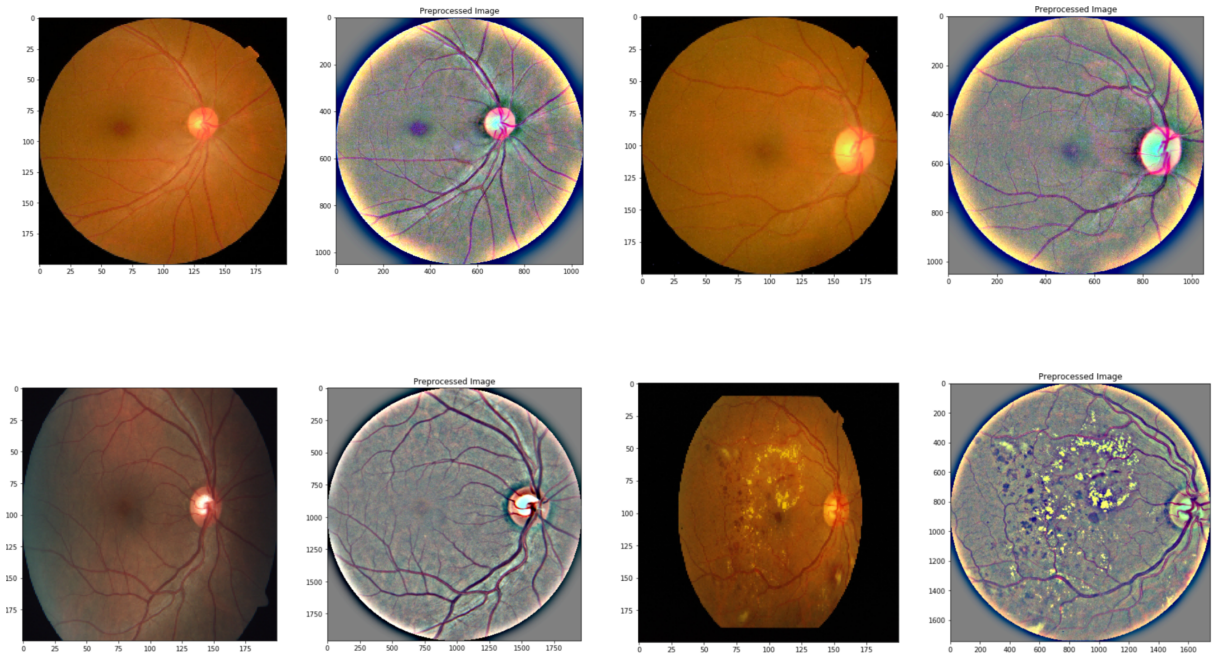
```
def crop(img,tolerance=7):
    if img.ndim==2:
        mask=img>tolerance
        return img[np_ix_(mask.any(1),mask.any(0))]
    elif img.ndim==3:
        gray_img=cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
        mask= gray_img>tolerance

        check_shape = img[:,:,0][np.ix_(mask.any(1),mask.any(0))].shape[0]
        if (check_shape == 0): # image is too dark so that we crop out everything,
            return img # return original image
        else:
            img1=img[:,:,0][np.ix_(mask.any(1),mask.any(0))]
            img2=img[:,:,1][np.ix_(mask.any(1),mask.any(0))]
            img3=img[:,:,2][np.ix_(mask.any(1),mask.any(0))]

            img = np.stack([img1,img2,img3],axis=-1)

        return img
```

Figure(7.8) Code snippet for cropping the fundus image

So the entire preprocessing includes,

1. Cropping edges
2. Gaussian Blur
3. Image Blending[26]

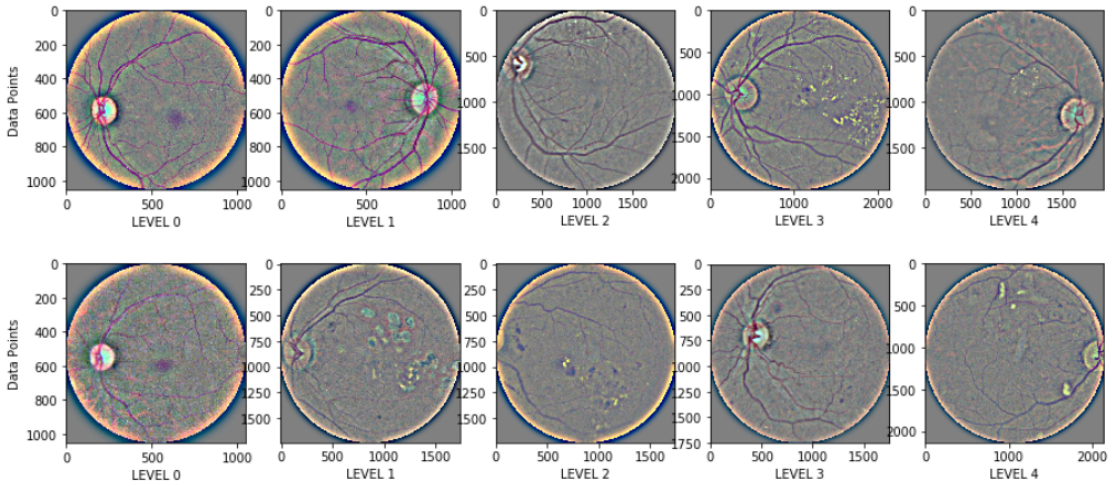
```
In [112]: def circular(img,sigmax):
              img=crop(img)
              img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
              h,w,d=img.shape
              x=int(w/2)
              y=int(h/2)
              z=np.amin((x,y))

              circle_img=np.zeros((h,w),np.uint8)
              cv2.circle(circle_img,(x,y),int(z),1,thickness=-1)
              img = cv2.bitwise_and(img, img, mask=circle_img)
              img = crop(img)
              img=cv2.addWeighted(img,4, cv2.GaussianBlur( img , (0,0) , sigmax) ,-3 ,128)
              return img
```



Figures(7.8a) Fully pre processed images

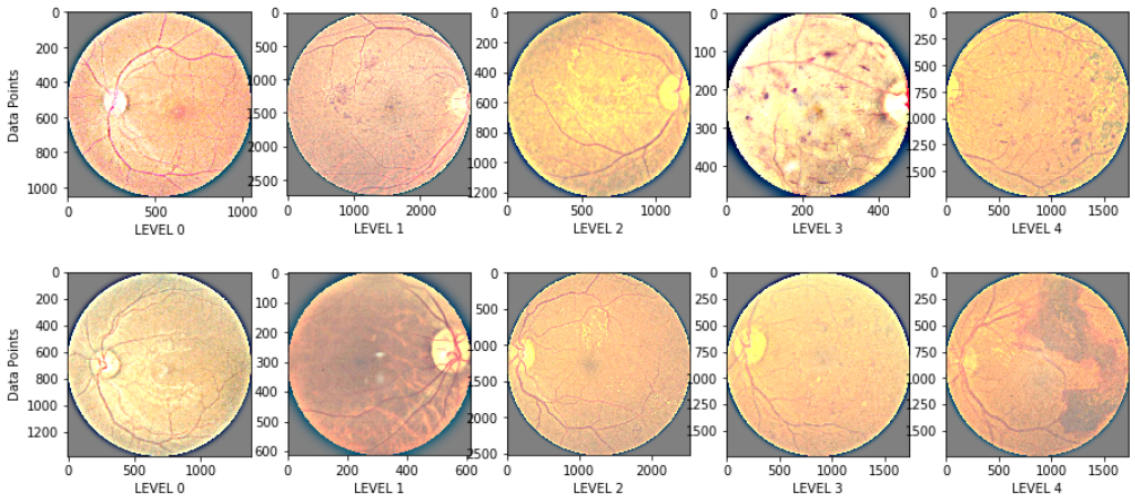
The preprocessed image varies with the values used in standard deviation for smoothening and also depends on the alpha beta values with the images being blended. So we need to find out the value of standard deviation and weights which can be used to get better results.

```
In [73]: showImagesPre(df_train,5,sigmax=30) #alpha=4 beta=-4
```



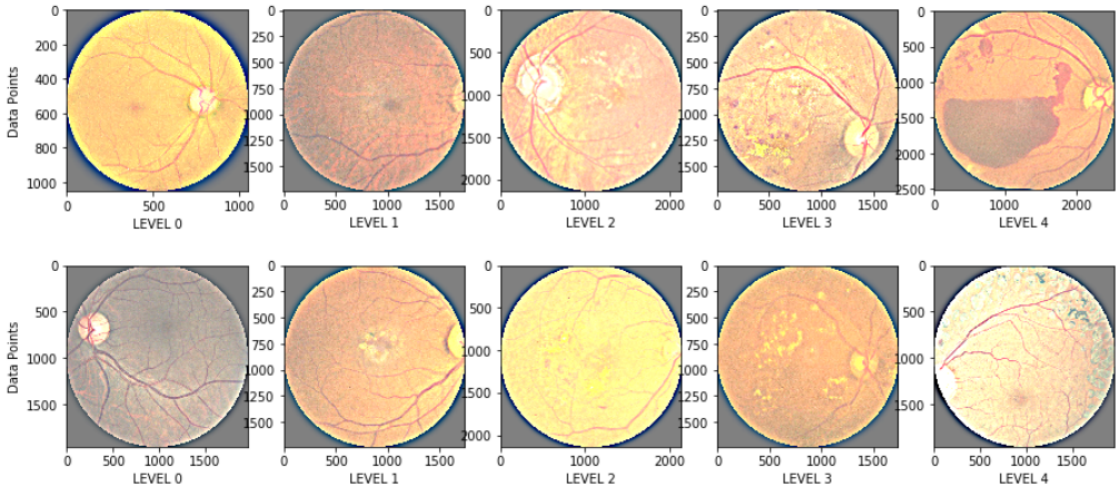
Figure(7.9) Images with standard deviation=30 , alpha=4, beta=-4

```
In [81]: showImagesPre(df_train,5,sigmax=20)  #alpha is 4 and beta is -3 defined at circular
```



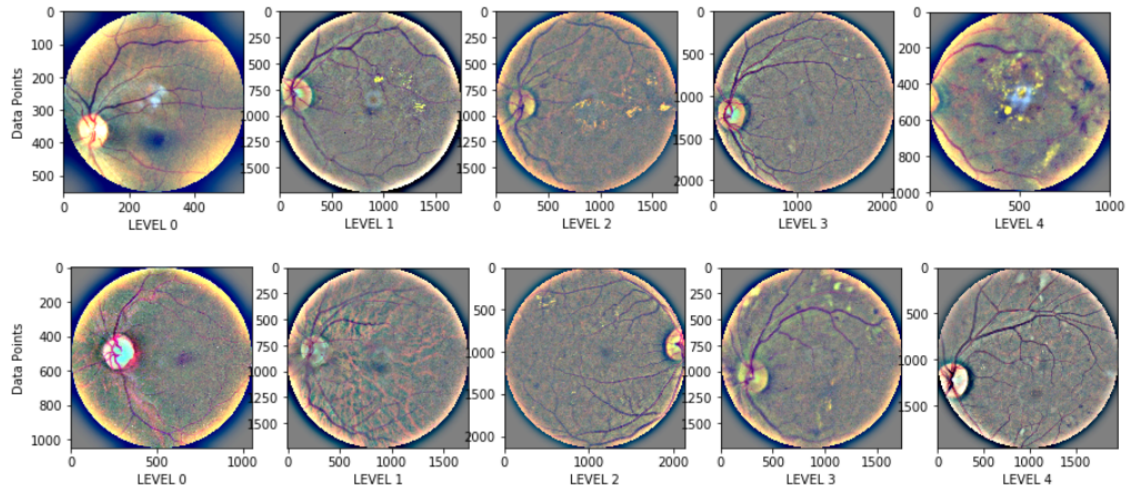
Figure(7.10) Images with standard deviation=20 , alpha=4, beta=-3

```
In [82]: showImagesPre(df_train,5,sigmax=30) #alpha is 4 and beta is -3 defined at circular
```



Figure(7.11) Images with standard deviation=30 , alpha=4, beta=-3

```
In [85]: showImagesPre(df_train,5,sigmax=50)#alpha is 4 and beta is -4 defined at circular
```



Figure(7.12) Images with standard deviation=50 , alpha=4, beta=-4

By doing a trial and error analysis using a base CNN model which was developed, we come to the conclusion that standard deviation:20 and alpha = 4, beta= -3 yields the best results.

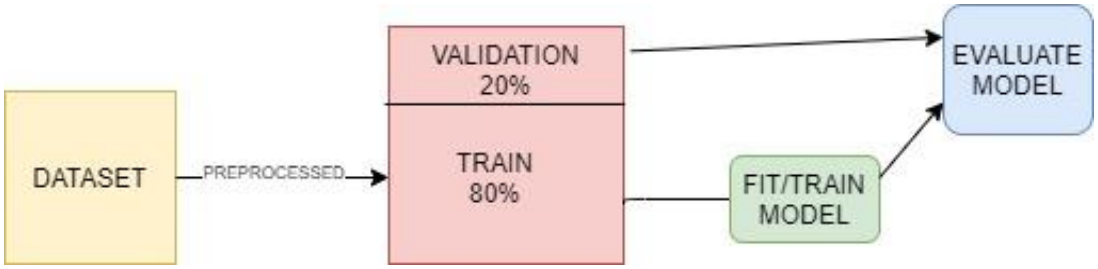
7.4 IMAGE AUGMENTATION

Image augmentation[27] is done using the keras library *IMAGEDATAGENERATOR.* Image data augmentation is a technique for artificially increasing the size of a training dataset by modifying images in the dataset. More data can help deep learning neural network models become more skilled, and image augmentation techniques can help suit models generalise what they've learned. The aim is to feature new, plausible examples to the training dataset. This refers to image variations from the training set that the model is likely to encounter.

Keras deep learning library provides data augmentation automatically when a model is being trained. Many techniques like flip,shear,zoom, rotation,brightness along with pixel rescaling are supported in the IMAGE DATA GENERATOR class.

First the instance for the datagenerator class is created, in which we specify the techniques that have to be applied. After an instance has been created, an iterator is created which returns a batch of augmented images. For iterator creation, *flow_from_dataframe()* is used. Generally we can use any flow() function.

flow_from_dataframe() gets the dataframe and the path to a directory + generates batches. The dataset's images aren't explicitly used. Instead, the model is only given augmented images. Since the augmentations are done at random, it is possible to create and use changed images as well as similar facsimiles of the original images.



Figure(7.13a) Dataset Split

Here we do a 4:1 train and test split and then again 4:1 train and validation split within the train images. Datagenerator is also used to specify the validation dataset and the test dataset.

```
In [9]: def img_generator(train,test):
            train_datagen=ImageDataGenerator(rescale=1./255, validation_split=0.2,horizontal_flip=True)

            train_generator=train_datagen.flow_from_dataframe(dataframe=df_train_train,
                                                              directory="C:/Users/Sabarish/Downloads/aptos2019-blindness-detection/train_
                                                              x_col="file_name",
                                                              y_col="diagnosis",
                                                              batch_size=BATCH_SIZE,
                                                              class_mode="categorical",
                                                              target_size=(HEIGHT, WIDTH),
                                                              subset='training')

            valid_generator=train_datagen.flow_from_dataframe(dataframe=df_train_train,
                                                              directory="C:/Users/Sabarish/Downloads/aptos2019-blindness-detection/train_
                                                              x_col="file_name",
                                                              y_col="diagnosis",
                                                              batch_size=BATCH_SIZE,
                                                              class_mode="categorical",
                                                              target_size=(HEIGHT, WIDTH),
                                                              subset='validation')

            test_datagen = ImageDataGenerator(rescale=1./255)
            test_generator = test_datagen.flow_from_dataframe(dataframe=df_train_test,
                                                              directory = "C:/Users/Sabarish/Downloads/aptos2019-blindness-detection/trai
                                                              x_col="file_name",
                                                              target_size=(HEIGHT, WIDTH),
                                                              batch_size=1,
                                                              shuffle=False,
                                                              class_mode=None
                                                             )

            return train_generator,valid_generator,test_generator
```

figure(13b): code snippet for dataset splitting

Here we use only rescale and horizontal_flip as techniques for augmentation.

Importance of rescaling[28]:

1. In a normal image, the pixel scale ranges from 0-255. Some images will be of high pixel density whereas others won't. So to bring up uniformity among the pixel ranges in images, we do rescaling by dividing the pixel value with 255 so that the resultant pixel scale of image ranges between 0-1.

2. Another reason is that due to pixel variations, the loss that is taken also varies ie, high pixel images create stronger loss and require only less learning rate compared to that of low pixel images which requires high learning rate. Since the images share a common model, learning rate and weights, these differences among their losses will contribute to back propagation update.

Horizontal_flip alone is used in augmentation techniques. In our scenario, there is no need for vertical flip as no input image will be provided in vertical flip. Horizontal flip is useful as left eye horizontal flip is similar to that of a right eye and vice versa. Similarly brightening techniques are also avoided as preprocessed image **clarity** gets affected due brightening,

7.5 CLASSIFICATION

After preprocessing and augmentation, the dataset is ready to be fed into the convolutional neural network. The dataset is split for training, testing and validating the model as follows:

```
from sklearn.model_selection import train_test_split

df_train_train,df_train_test = train_test_split(df_train,test_size = 0.2)
print(df_train_train.shape,df_train_test.shape)
```

```
(2929, 4) (733, 4)
```

Figure(7.14) Train and test split

```
train_generator,valid_generator,test_generator = img_generator(df_train_train,df_train_test)
```

```
Found 2344 validated image filenames belonging to 5 classes.
Found 585 validated image filenames belonging to 5 classes.
Found 733 validated image filenames.
```

Figure(7.15) : No of images under train, validation, test

Initially a base CNN model is designed to observe for which set of images(preprocessed with different standard deviation, gray scale) the performance metrics are high.

The CNN model has the following architecture:

1. It has three convolution layers and one hidden layer in the fully connected layer.
2. First convolutional layer has 2 filters followed by a max-pooling layer and batch normalization.
3. Second convolutional layer has 3 filters followed by a max-pooling layer and Flatten layer.
4. The hidden dense layer contains 1000 neurons.
5. The output layer contains 5 neurons each one corresponds to the 5 different classes.

```
m1d= Sequential()
m1d.add(Conv2D(3,kernel_size=(3,3),strides=(1,1),
               activation='relu',
              input_shape=(320,320,3)))
m1d.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
m1d.add(BatchNormalization())
m1d.add(Conv2D(3,(5,5),activation='relu'))
m1d.add(MaxPooling2D(pool_size=(3,3)))
m1d.add(Flatten())
m1d.add(Dense(1000,activation='relu'))
m1d.add(Dense(5,activation='softmax'))

m1d.compile(loss='categorical_crossentropy',optimizer=optimizers.SGD(lr=0.01),metrics=['accuracy'])
```

Figure(7.16): Code for architecture of CNN

```
m1d.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 318, 318, 3)       84
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 159, 159, 3)       0
_________________________________________________________________
batch_normalization_5 (Batch (None, 159, 159, 3)       12
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 155, 155, 3)       228
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 51, 51, 3)         0
_________________________________________________________________
flatten_5 (Flatten)          (None, 7803)              0
_________________________________________________________________
dense_9 (Dense)              (None, 1000)              7804000
_________________________________________________________________
dense_10 (Dense)             (None, 5)                 5005
=================================================================
Total params: 7,809,329
Trainable params: 7,809,323
Non-trainable params: 6
```

Figure(7.17): CNN Architecture.

The table below shows the hyper parameters used

HYPER PARAMETERS	VALUES
ACTIVATION FUNCTIONS	RELU, SOFTMAX
NUMBER OF EPOCHS	15
OPTIMIZER USED	ADAM OPTIMIZER
LEARNING RATE	0.01
LOSS FUNCTION	CATEGORICAL CROSS ENTROPY

Table 3: Hyperparameters for base CNN model

The training is done using *fit_generator()* function. The reason for using *fit_generator()* over *fit()* is that for augmentation, we used IMAGEDATAGENERATOR. Since the datagenerator function is used, corresponding we use *fit_generator()* for training as well. The parameter steps_per_epoch is to be specified inside the fit_generator() along with the train_generator, valid_generator.

```
STEP_SIZE_TRAIN = train_generator.n//train_generator.batch_size
STEP_SIZE_VALID = valid_generator.n//valid_generator.batch_size
print(STEP_SIZE_TRAIN,STEP_SIZE_VALID)
```

```
293 73
```

```
tune_gray1 = m1d.fit_generator(generator=train_generator,
                               steps_per_epoch=STEP_SIZE_TRAIN,
                               validation_data=valid_generator,validation_steps=STEP_SIZE_VALID,
                               epochs=15,
                               verbose=1).history
```

```
Epoch 1/15
293/293 [==============================] - 148s 507ms/step - loss: 0.8819 - accuracy: 0.6762 - val_loss: 1.4942 - val_accuracy:
0.7312
Epoch 2/15
293/293 [==============================] - 122s 417ms/step - loss: 0.7934 - accuracy: 0.7078 - val_loss: 0.8140 - val_accuracy:
0.7019
Epoch 3/15
293/293 [==============================] - 124s 422ms/step - loss: 0.7697 - accuracy: 0.7095 - val_loss: 0.5964 - val_accuracy:
0.7331
Epoch 4/15
293/293 [==============================] - 127s 432ms/step - loss: 0.7614 - accuracy: 0.7180 - val_loss: 0.7282 - val_accuracy:
0.7123
Epoch 5/15
293/293 [==============================] - 126s 429ms/step - loss: 0.7482 - accuracy: 0.7193 - val_loss: 0.0949 - val_accuracy:
0.7175
Epoch 6/15
293/293 [==============================] - 126s 429ms/step - loss: 0.7452 - accuracy: 0.7218 - val_loss: 1.0150 - val_accuracy:
0.7314
Epoch 7/15
293/293 [==============================] - 126s 431ms/step - loss: 0.7312 - accuracy: 0.7278 - val_loss: 0.5600 - val_accuracy:
0.7279
Epoch 8/15
293/293 [==============================] - 126s 429ms/step - loss: 0.7352 - accuracy: 0.7218 - val_loss: 1.5701 - val_accuracy:
0.7192
```

Figure(7.18) Training of model using base CNN

TRANSFER LEARNING

For the application of transfer learning, initially the resnet50 model weights are downloaded. Then these weights are loaded and we does following modifications:

1. Applying GlobalAveragePool on the loaded model
2. Applying DropOut of 50% for regularization
3. Creating a dense layer with 2048 neurons with activation function relu
4. Again Dropout of 50% for regularization
5. Finally creating final output layer using denser layer of 5 neurons and softmax as activation function

```
def create_model(input_shape, n_out):
    input_tensor = Input(shape=input_shape)
    base_model = applications.ResNet50(weights=None, include_top=False,input_tensor=input_tensor)
    base_model.load_weights('resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5')

    x = GlobalAveragePooling2D()(base_model.output)
    x = Dropout(0.5)(x)
    x = Dense(2048, activation='relu')(x)
    x = Dropout(0.5)(x)
    final_output = Dense(n_out, activation='softmax', name='final_output')(x)
    model = Model(input_tensor, final_output)
    return model
```

Figure(7.19) RESNET weight loaded and model built on the top of Resnet

```
model = create_model(input_shape=(HEIGHT, WIDTH, CANAL), n_out=N_CLASSES)

for layer in model.layers:
    layer.trainable = False

for i in range(-5, 0):
    model.layers[i].trainable = True
model.summary()
```

```
Model: "model_1"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         (None, 320, 320, 3)  0
_________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 326, 326, 3)  0           input_1[0][0]
_________________________________________________________________________________________________
conv1 (Conv2D)               (None, 160, 160, 64) 9472        conv1_pad[0][0]
_________________________________________________________________________________________________
bn_conv1 (BatchNormalization)  (None, 160, 160, 64) 256         conv1[0][0]
_________________________________________________________________________________________________
activation_1 (Activation)    (None, 160, 160, 64) 0           bn_conv1[0][0]
_________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)    (None, 162, 162, 64) 0           activation_1[0][0]
_________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 80, 80, 64)   0           pool1_pad[0][0]
_________________________________________________________________________________________________
res2a_branch2a (Conv2D)      (None, 80, 80, 64)   4160        max_pooling2d_1[0][0]
```

```
dropout_2 (Dropout)             (None, 2048)         0           dense_1[0][0]
__________________________________________________________________________________________________
final_output (Dense)            (None, 5)            10245       dropout_2[0][0]
==================================================================================================
Total params: 27,794,309
Trainable params: 4,206,597
Non-trainable params: 23,587,712
```

Figure(7.20) Resnet50 Architecture.

CALLBACKS

Callbacks[24] are kinds of triggers that are invoked under particular defined circumstances. The use of callback functions are to improve the performance of the model. The major advantage of callbacks is that it prevents models from being overtrained. Callbacks execute after each epoch. In our study we have used 2 callback functions.

1. EARLY STOPPING
2. REDUCE LR ON PLATEAU

EARLY STOPPING: The main drawback of training neural networks is with the number of epochs for which model needs to be trained. Depending on the value of epoch, the model can either underfit or overfit, but most of the time, the latter occurs. Due to overfitting the train accuracy linearly increases, whereas there will be large fluctuations in validation accuracy. To avoid this, EARLY STOPPING comes handy. This gets the input on the parameter to be monitored (mostly validation loss), and patience epochs which indicates the number of epochs the model should continue to be trained before the training stops.

REDUCE LR ON PLATEAU: This callback is also very similar to that of EARLY STOPPING. On overfitting, when the learning rate stagnants and reaches Plateau, this callback is triggered. This callback reduces the learning rate by a fraction of *decay_lr* which is specified in the function call. Here also the input on the parameter to monitor(mostly validation loss) and patience epochs which indicates the number of epochs the model should be trained before the LR is reduced.

OPTIMIZER: The optimization algorithm (also known as an optimizer) is the most common method for training a machine learning model to reduce its error rate. An optimizer's efficacy can be measured by using two metrics: *convergence and generalization* pace.

In our study two optimizers are used:

1. SGD [Stochastic Gradient Descent]
2. ADAM [Adaptive Moment Estimator]

The most popular approach for optimising deep learning networks is Gradient Descent. This method can be used to update each model parameter, observe how a shift affects the objective function, select a path that lowers the error rate, and iterate until the objective function converges to the minimum. But recent experiments conducted proved that GD is not an efficient approach.

SGD is a variant of Gradient descent. GD operates on the entire dataset whereas SGD performs computations in a subset of dataset as it is not efficient as well as redundant while computing the entire dataset. Adam is a gradient based stochastic objective function optimization algorithm. It computes individual adaptive learning rates for different parameters by combining the benefits of two SGD extensions :

- Root Mean Square Propagation (RMSProp)
- Adaptive Gradient Algorithm (AdaGrad).

```
from keras.layers import Input,GlobalAveragePooling2D,Dropout,Dense,Activation
from keras.callbacks import EarlyStopping,ReduceLROnPlateau
```

```
for layer in model.layers:
    layer.trainable = True

es = EarlyStopping(monitor='val_loss', mode='min', patience=ES_PATIENCE, restore_best_weights=True, verbose=1)
rlrop = ReduceLROnPlateau(monitor='val_loss', mode='min', patience=RLROP_PATIENCE, factor=DECAY_DROP, min_lr=1e-6, verbose=1)

callback_list = [es, rlrop]
optimizer = optimizers.Adam(lr=LEARNING_RATE)
model.compile(optimizer=optimizer, loss="categorical_crossentropy",  metrics=['accuracy'])
model.summary()
```

Figure(7.21): Model Compilation using callbacks

```
model.summary()
```

```
Model: "model_1"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         (None, 320, 320, 3)       0
_________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 326, 326, 3)       0           input_1[0][0]
_________________________________________________________________________________________________
conv1 (Conv2D)               (None, 160, 160, 64)      9472        conv1_pad[0][0]
_________________________________________________________________________________________________
bn_conv1 (BatchNormalization)(None, 160, 160, 64)      256         conv1[0][0]
_________________________________________________________________________________________________
activation_1 (Activation)    (None, 160, 160, 64)      0           bn_conv1[0][0]
_________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)    (None, 162, 162, 64)      0           activation_1[0][0]
_________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)(None, 80, 80, 64)       0           pool1_pad[0][0]
_________________________________________________________________________________________________
res2a_branch2a (Conv2D)      (None, 80, 80, 64)        4160        max_pooling2d_1[0][0]

global_average_pooling2d_1 (Glo (None, 2048)           0           activation_49[0][0]
_________________________________________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0           global_average_pooling2d_1[0][0]
_________________________________________________________________________________________________
dense_1 (Dense)              (None, 2048)              4196352     dropout_1[0][0]
_________________________________________________________________________________________________
dropout_2 (Dropout)          (None, 2048)              0           dense_1[0][0]
_________________________________________________________________________________________________
final_output (Dense)         (None, 5)                 10245       dropout_2[0][0]
=================================================================================================
Total params: 27,794,309
Trainable params: 27,741,189
Non-trainable params: 53,120
```

Figure(7.22) : Final Model summary

```
history_finetunning = model.fit_generator(generator=train_generator,
                                          steps_per_epoch=STEP_SIZE_TRAIN,
                                          validation_data=valid_generator,
                                          validation_steps=STEP_SIZE_VALID,
                                          epochs=EPOCHS,
                                          callbacks=callback_list,
                                          verbose=1).history
```

Figure(7.23): Code snippet for model training with callbacks

```
Epoch 3/40
366/366 [==============================] - 10875s 30s/step - loss: 0.1814 - accuracy: 0.9241 - val_loss: 0.1439 - val_accuracy:
0.9196
Epoch 4/40
366/366 [==============================] - 10156s 28s/step - loss: 0.1629 - accuracy: 0.9321 - val_loss: 0.0904 - val_accuracy:
0.9061
Epoch 5/40
366/366 [==============================] - 9950s 27s/step - loss: 0.1430 - accuracy: 0.9417 - val_loss: 0.6599 - val_accuracy:
0.9000
Epoch 6/40
366/366 [==============================] - 11019s 30s/step - loss: 0.1360 - accuracy: 0.9464 - val_loss: 0.2107 - val_accuracy:
0.8751
Epoch 7/40
366/366 [==============================] - 11573s 32s/step - loss: 0.1251 - accuracy: 0.9491 - val_loss: 0.2783 - val_accuracy:
0.9094

Epoch 00007: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
Epoch 8/40
366/366 [==============================] - 10065s 28s/step - loss: 0.0803 - accuracy: 0.9693 - val_loss: 0.0025 - val_accuracy:
0.9166
Epoch 9/40
366/366 [==============================] - 10666s 29s/step - loss: 0.0613 - accuracy: 0.9787 - val_loss: 0.3402 - val_accuracy:
0.9229
Epoch 10/40
366/366 [==============================] - 11452s 31s/step - loss: 0.0540 - accuracy: 0.9817 - val_loss: 0.2791 - val_accuracy:
0.9204
Epoch 11/40
366/366 [==============================] - 9949s 27s/step - loss: 0.0449 - accuracy: 0.9857 - val_loss: 0.2493 - val_accuracy:
0.9238

Epoch 00011: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.
Epoch 12/40
366/366 [==============================] - 9625s 26s/step - loss: 0.0330 - accuracy: 0.9902 - val_loss: 0.0215 - val_accuracy:
0.9287
Epoch 13/40
366/366 [==============================] - 9741s 27s/step - loss: 0.0288 - accuracy: 0.9906 - val_loss: 0.5448 - val_accuracy:
0.9166
Restoring model weights from the end of the best epoch
Epoch 00013: early stopping
```

Figure(7.24): Model training results.

Table below shows the hyper parameters used

HYPER PARAMETERS	VALUES
ACTIVATION FUNCTION	RELU, SOFTMAX
NO OF EPOCHS	40 | [15]
OPTIMIZER USED	ADAM OPTIMIZER [SGD+rmsProp]
LEARNING RATE	le-04
LOSS FUNCTION	CATEGORICAL CROSS ENTROPY
CALL BACKS USED	EARLY STOPPING, REDUCE LR ON PLATEAU
EARLY STOPPING PATIENCE	5 EPOCHS
REDUCE LR PATIENCE	3 EPOCHS
LEARNING DECAY AMOUNT	0.5 ON LR

Table 4: Hyperparameters for resenet-50 model

The model is evaluated using the metrics such as accuracy, precision, recall and f1-score.

Fig.(7.25) shows the code snippet for the model evaluation.

```
from sklearn.metrics import confusion_matrix, cohen_kappa_score,accuracy_score
print("Test Cohen Kappa score: %.3f" % cohen_kappa_score(test_preds, df_train_test['diagnosis'].astype('int'),
                                                         weights='quadratic'))
print("Test Accuracy score : %.3f" % accuracy_score(df_train_test['diagnosis'].astype('int'),test_preds))
cm = confusion_matrix(df_train_test['diagnosis'].astype('int'),test_preds)
test_recall = np.diag(cm) / np.sum(cm, axis = 1)
test_precision = np.diag(cm) / np.sum(cm, axis = 0)
print('Recall :%.3f' %np.mean(test_recall))
print('Precision : %.3f'%np.mean(test_precision ))
f1score= 2*(test_recall*test_precision)/(test_precision+test_recall)
print('F1 SCORE : %.3f' %np.mean(f1score))
```

Figure(7.25) : code snippet for model evaluation.

7.6 DEPLOYMENT

The developed model is integrated with Flask-application. This developed website is then hosted in Amazon's AWS EC2 instance.

```
if __name__ == "__main__":
    print(("* Loading Keras model and Flask starting server..."
           "please wait until server has fully started"))
    # Run app
    app.run(host="0.0.0.0",threaded=False, debug=False)
```

Figure(7.26): code snippet for flask integration

The above is the code snippet for integrating the python code with flask to get a web application.

```
@app.route('/')
def xno():
  return render_template("test.html")


@app.route('/uploader', methods = ['POST'])
def upload_file():
   if request.method == 'POST':
      f = request.files.getlist("file")

      results = []
      for i in range(2):
        file_base = "uploaded"+str(random.randint(1,1000))
        filename = file_base+".jpeg"
        print(filename)
        f[i].save(os.path.join(os.path.join(path,"static"), filename))

        print(f[i].filename)
        img = cv2.imread(os.path.join(os.path.join(path,"static"), filename))
        x,pre=predict(img)
        x=x.flatten()
        y=[0,1,2,3,4]
        plot.ylabel('PROBABILITY')
        plot.xlabel('STAGE')
        plot.bar(y,x,color='green', edgecolor = 'red')

        plot_name = "plot"+file_base+".jpeg"
        plot.savefig(os.path.join(os.path.join(path,"static"), plot_name))
        plot.savefig(os.path.join(path,plot_name))
        if i==0:
          img1=filename;plot1=plot_name;X.append(img1);X.append(plot1)
        else:
          img2=filename;plot2=plot_name; X.append(img2);X.append(plot2)
        plot.close()
        ab=plot.show()
        plot.close()
        plot.close()


        results.append(x)
        val.append(pre)
        global model
      print(os.path.join(os.path.join(path,"static"), str(img1)))
      return render_template("dr.html", im1= img1, im2=img2, plt1=plot1, plt2=plot2,res= val)
```

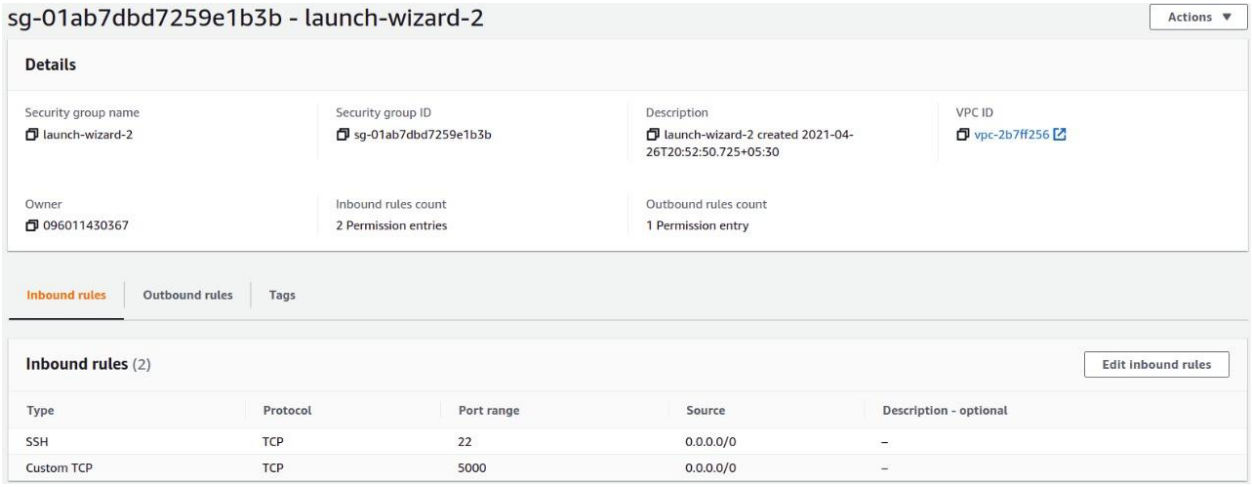
Figure(7.27): code snippet for getting user input of image and predicting it with developed model

The above code creates a basic web page <test.html> for uploading the fundus image and using the created model to predict the proper DR stage and display it in the subsequent web page <dr.html>.

```python
@app.route('/download', methods=['GET','POST'])
def downloadPdf():
    output=BytesIO()
    canva = canvas.Canvas(output, pagesize=letter)
    width,height = letter

    canva.setLineWidth(.3)
    canva.setFont('Helvetica', 20)
    canva.drawString(width/2-60,750,'MEDICAL REPORT')

    canva.drawImage( path+"static/"+str(X[0]), 50,500, 3*inch,3*inch)
    canva.drawImage( path+"static/"+str(X[2]), width-50-(3*inch),500, 3*inch,3*inch)
    canva.drawImage( path+"static/"+str(X[1]), 50,500-3*inch-50, 3*inch,3*inch)
    canva.drawImage(path+"static/"+str(X[3]), width-50-(3*inch),500-3*inch-50, 3*inch,3*inch)
    canva.drawString(100,500-3*inch-150,"LEFT EYE STAGE: "+str(val[0]))
    canva.drawString(width-(2*inch)-100,500-3*inch-150,"RIGHT EYE STAGE: "+str(val[1]))

    canva.save()
    pdf_out = output.getvalue()
    output.close()

    response = make_response(pdf_out)
    response.headers['Content-Disposition'] = "attachment; filename='REPORT1.pdf"
    response.mimetype = 'application/pdf'
    return response
```

Figure(7.28): code snippet for allowing the user to download the medical report as PDF.

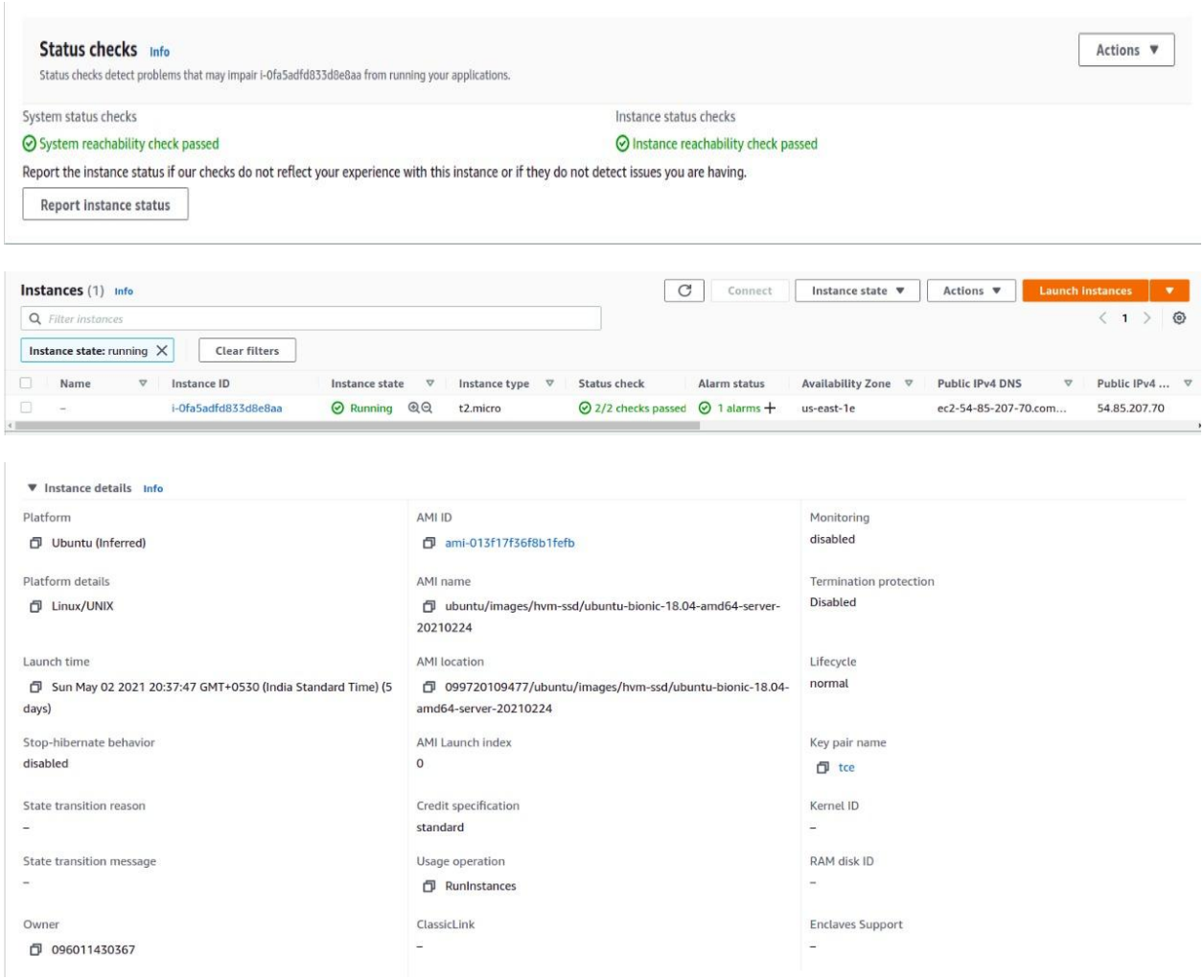
The above is the code snippet which allows the user to download his medical report as a PDF file.



Figure(7.29) :AWS Dashboard

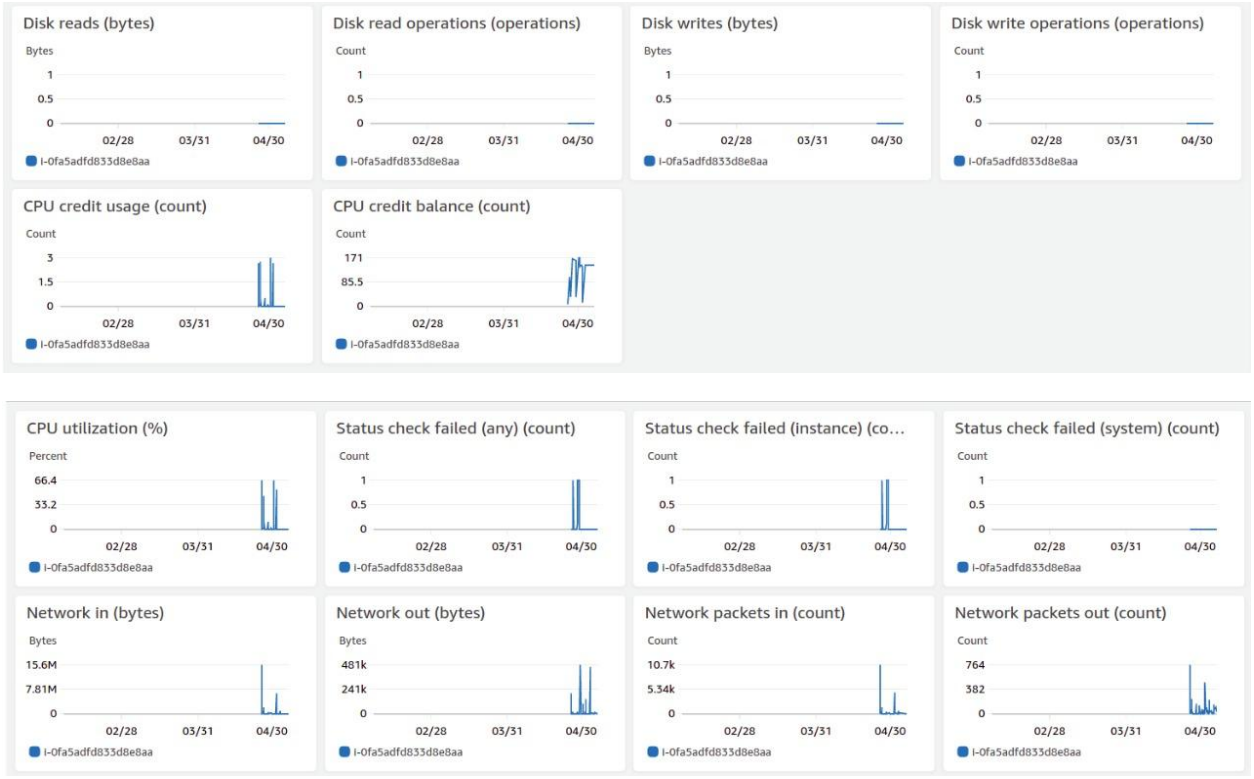
The above figure is the AWS EC2 dashboard for the hosted instance.

Instance Details:



figure(7.30) Details of the EC2 resource

Performance stats of EC2:



Figure(7.31) Performance report of EC2 instance

CHAPTER 8

EXPERIMENTS AND RESULTS

8.1 DataSet Loading and Visualization.

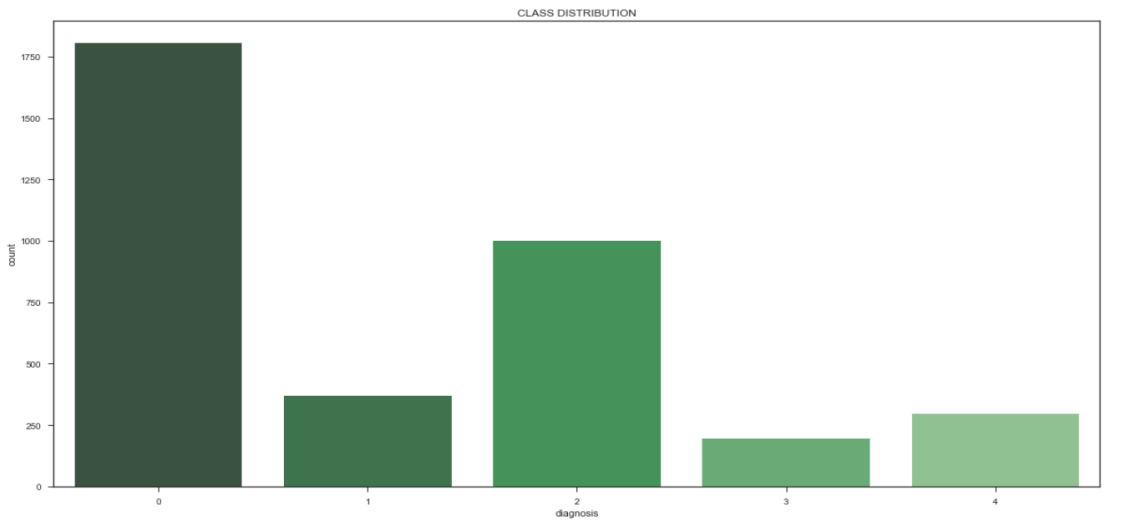
```
In [8]: #now we need to read data from test and train dir

def load_data_dir():
    train=pd.read_csv('train.csv')
    test=pd.read_csv('test.csv')

    test_dir=os.path.join('./','test_images/')
    train_dir=os.path.join('./','train_images/')
    train['file_path']=train['id_code'].map(lambda x:os.path.join(train_dir,'{}.png'.format(x)))
    test['file_path']=test['id_code'].map(lambda x:os.path.join(test_dir,'{}.png'.format(x)))

    train['file_name']=train['id_code'].apply(lambda x: x+ ".png")
    test['file_name']=test['id_code'].apply(lambda x: x+ ".png")
    train['diagnosis']= train['diagnosis'].astype(str)

    return train,test
```

```
In [10]: df_train,df_test=load_data_dir()
print(df_train.shape,df_test.shape)
```

```
(3662, 4) (1928, 3)
```

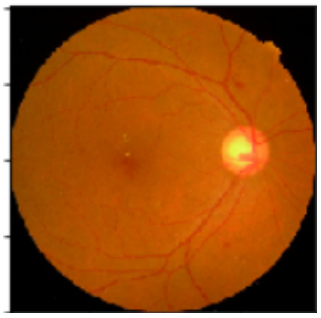
Figure(8.1.a) : Code snippet for dataset loading

```
In [44]: visualize_classes(df_train)
```

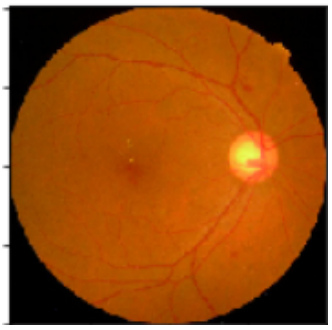


Figure(8.1.b) Dataset visualisation

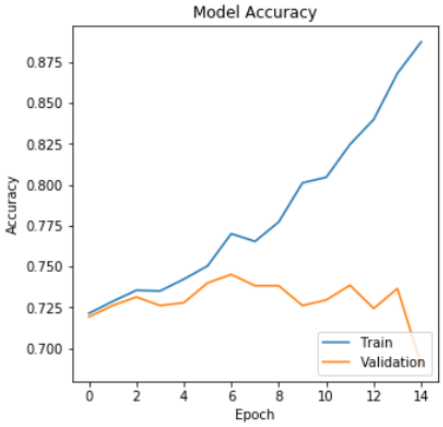
8.2 Base CNN model + Un-preprocess images



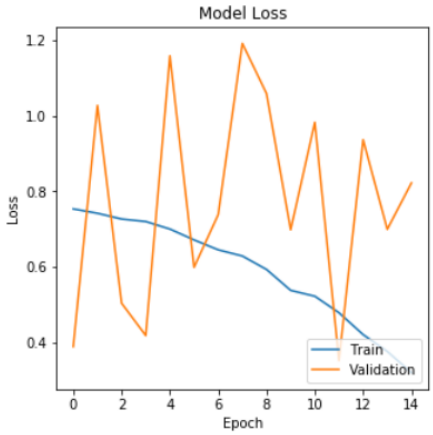
Figure(8.2.1): Normal image



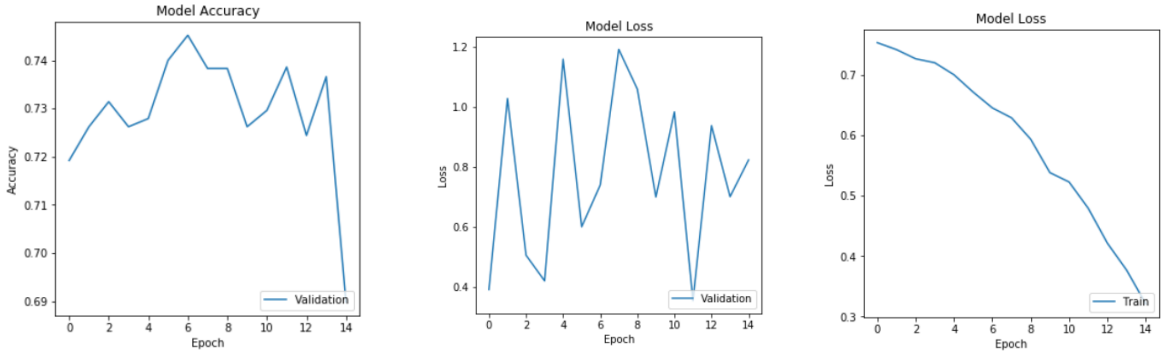
Figure(8.2.2): Image Fed into Neural Ntw



Figure(8.2.3): Accuracy Curve



Figure(8.2.4): Loss curve



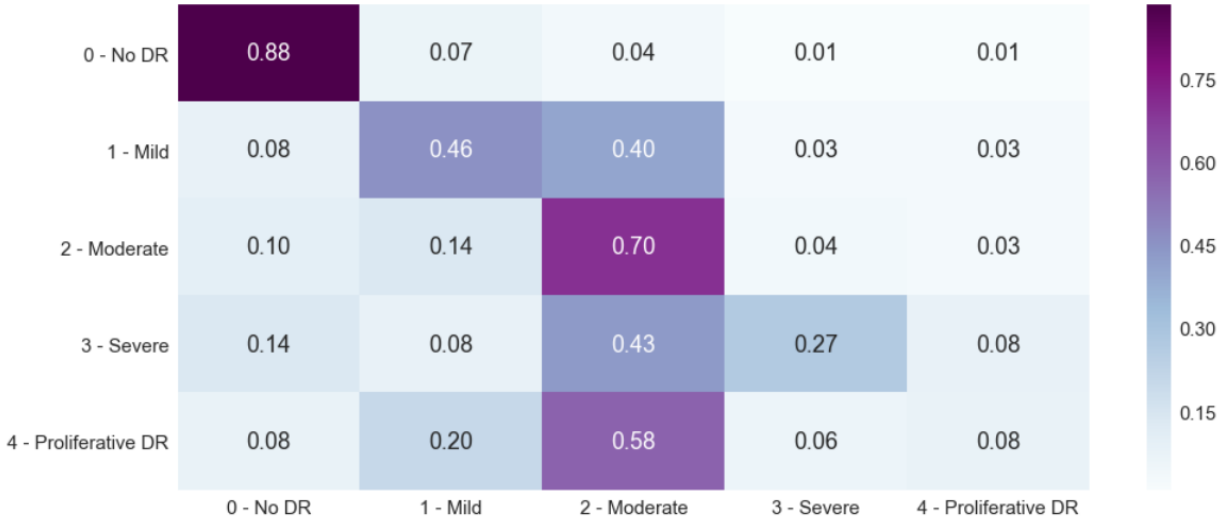
Figure(8.2.5) Validation Accuracy, Validation Loss & Training Loss

With the results, obtained a confusion matrix representing the percentage values are plotted.

Below is the code snippet of confusion matrix generated:

```
labels = ['0 - No DR', '1 - Mild', '2 - Moderate', '3 - Severe', '4 - Proliferative DR']
plot_conf_matrix(list(df_train_test['diagnosis'].astype(int)),test_preds,labels)

cnf_matrix = confusion_matrix(df_train_test['diagnosis'].astype('int'), test_preds)
cnf_matrix_norm = cnf_matrix.astype('float') / cnf_matrix.sum(axis=1)[:, np.newaxis]
df_cm = pd.DataFrame(cnf_matrix_norm, index=labels, columns=labels)
plot.figure(figsize=(16, 7))
sns.heatmap(df_cm, annot=True, fmt='.2f', cmap="BuPu")
plot.show()
```

Figure(8.2.6): code snippet for confusion matrix



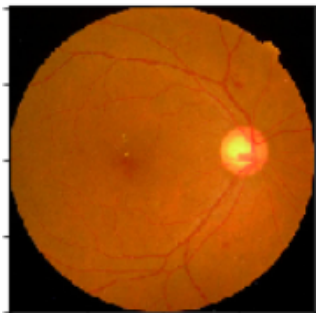
Figure(8.2.7) Confusion matrix

```
print("Test Cohen Kappa score: %.3f" % cohen_kappa_score(test_labels, df_train_test['diagnosis'].astype('int'), weights='quad
print("Test Accuracy score : %.3f" % accuracy_score(df_train_test['diagnosis'].astype('int'),test_labels))
cm = confusion_matrix(df_train_test['diagnosis'].astype('int'),test_labels)
test_recall = np.diag(cm) / np.sum(cm, axis = 1)
test_precision = np.diag(cm) / np.sum(cm, axis = 0)
print('Recall :%.3f' %np.mean(test_recall))
print('Precision : %.3f'%np.mean(test_precision ))
f1score= 2*(test_recall*test_precision)/(test_precision+test_recall)
print('F1 SCORE : %.3f' %np.mean(f1score))
```

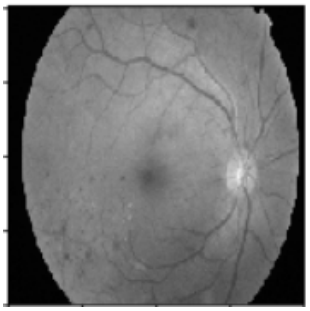
```
Test Cohen Kappa score: 0.643
Test Accuracy score : 0.693
Recall :0.478
Precision : 0.497
F1 SCORE : 0.471
```

Figure(8.2.8) Performance Metrics

8.3 Base CNN + Grey scaled images

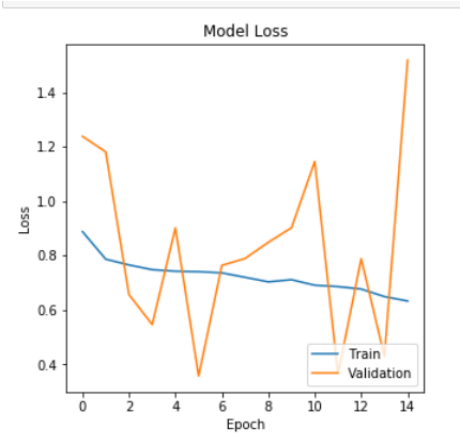


Figure(8.3.1): Normal image

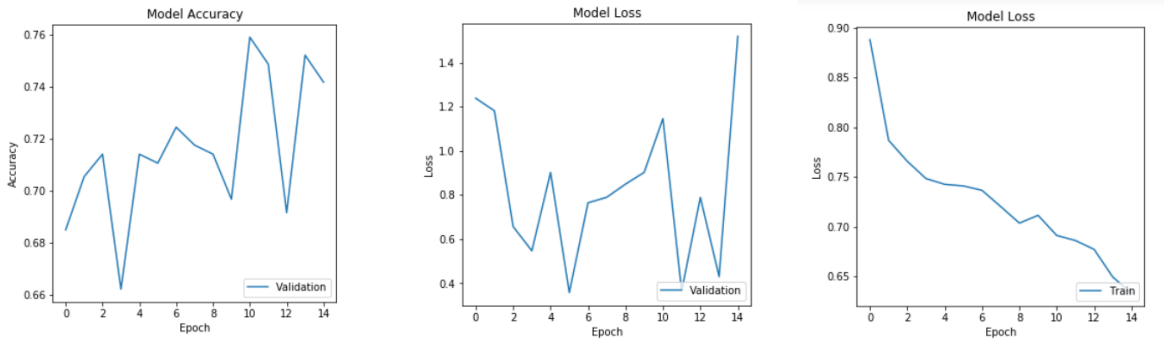


Figure(8.3.2): Image Fed into Neural Ntw

Figure(8.3.3): Accuracy Curve



Figure(8.3.4): Loss curve



Figure(8.3.5) Validation Accuracy, Validation Loss & Training Loss

With the results obtained, a confusion matrix representing the percentage values are plotted. Below is the confusion matrix generated:

Figure(8.3.6) Confusion matrix

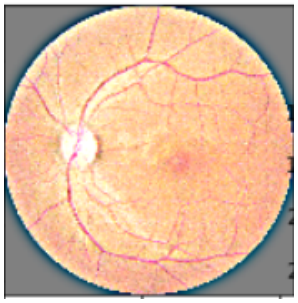
```
print("Test Cohen Kappa score: %.3f" % cohen_kappa_score(test_labels, df_train_test['diagnosis'].astype('int'), weights='quadr
print("Test Accuracy score : %.3f" % accuracy_score(df_train_test['diagnosis'].astype('int'),test_labels))
cm = confusion_matrix(df_train_test['diagnosis'].astype('int'),test_labels)
test_recall = np.diag(cm) / np.sum(cm, axis = 1)
test_precision = np.diag(cm) / np.sum(cm, axis = 0)
print('Recall :%.3f' %np.mean(test_recall))
print('Precision : %.3f'%np.mean(test_precision ))
f1score= 2*(test_recall*test_precision)/(test_precision+test_recall)
print('F1 SCORE : %.3f' %np.mean(f1score))
```

```
Test Cohen Kappa score: 0.663
Test Accuracy score : 0.726
Recall :0.474
Precision : 0.544
F1 SCORE : 0.483
```

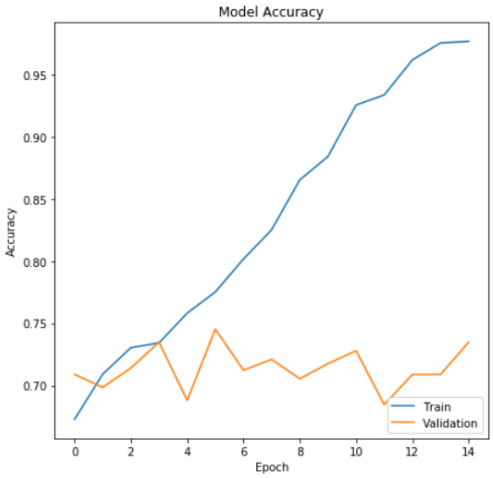
Figure(8.3.7) Performance Metrics

8.4 Base CNN + Preprocess(Standard Deviation=20)

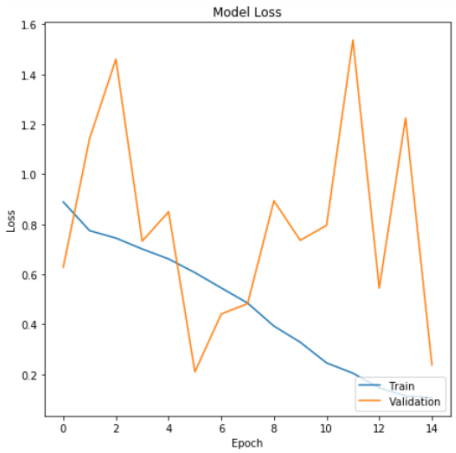
Figure(8.4.1): Normal image



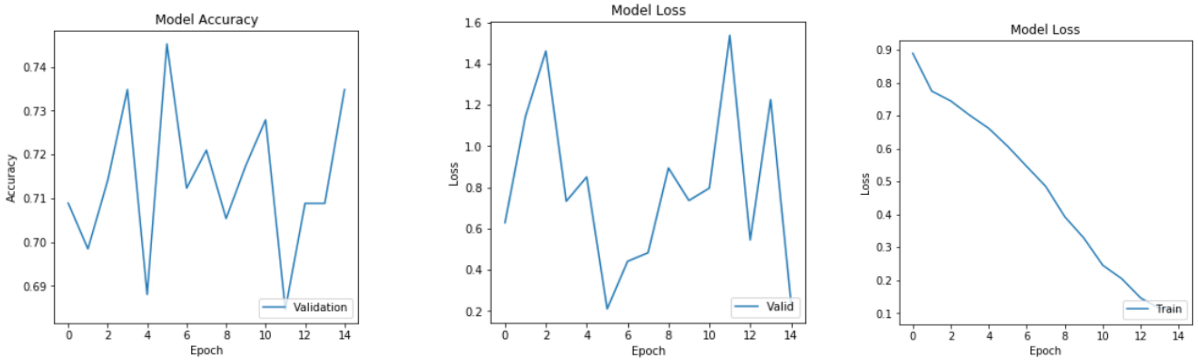
Figure(8.4.2): Image Fed into Neural Ntw



Figure(8.4.3): Accuracy Curve

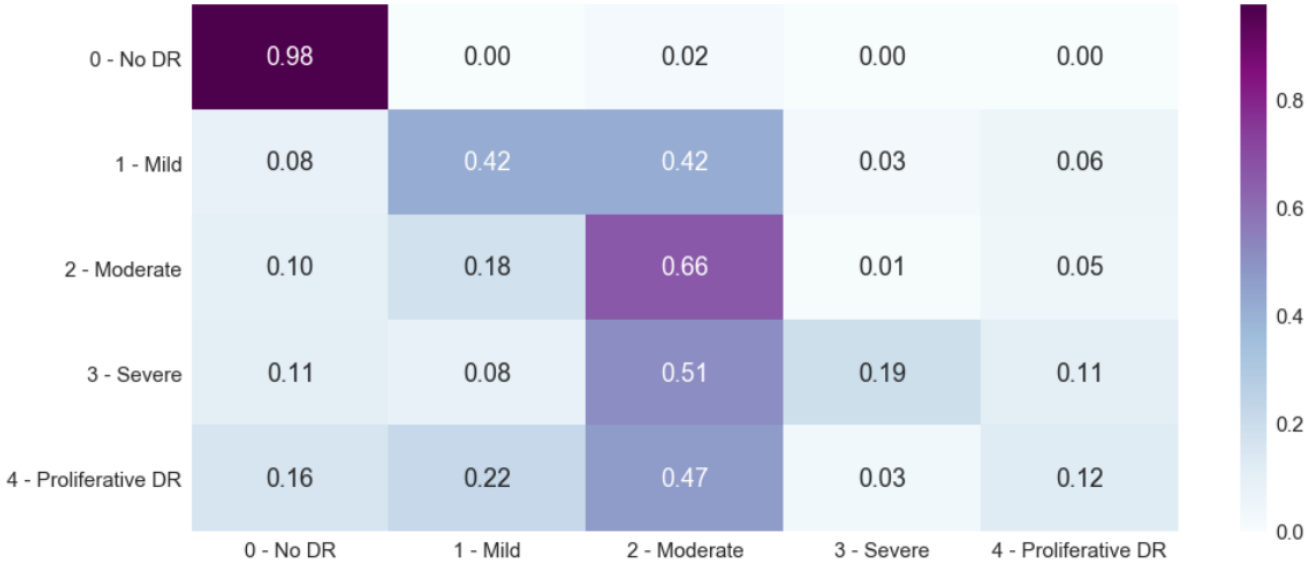


Figure(8.4.4): Loss curve



Figure(8.4.5) Validation Accuracy, Validation Loss & Training Loss

With the results obtained, a confusion matrix representing the percentage values are plotted. Below is the confusion matrix generated:



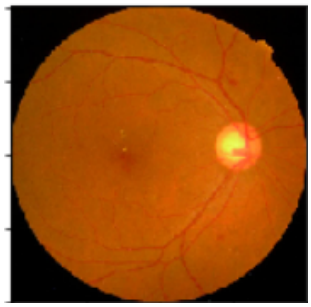
Figure(8.4.6) Confusion matrix

```
print("Test Cohen Kappa score: %.3f" % cohen_kappa_score(test_labels, df_train_test['diagnosis'].astype('int'), weights='quadrati
print("Test Accuracy score : %.3f" % accuracy_score(df_train_test['diagnosis'].astype('int'),test_labels))
cm = confusion_matrix(df_train_test['diagnosis'].astype('int'),test_labels)
test_recall = np.diag(cm) / np.sum(cm, axis = 1)
test_precision = np.diag(cm) / np.sum(cm, axis = 0)
print('Recall :%.3f' %np.mean(test_recall))
print('Precision : %.3f'%np.mean(test_precision ))
f1score= 2*(test_recall*test_precision)/(test_precision+test_recall)
print('F1 SCORE : %.3f' %np.mean(f1score))
```

```
Test Cohen Kappa score: 0.696
Test Accuracy score :0.752
Recall :0.566
Precision : 0.513
F1 SCORE :0.547
```

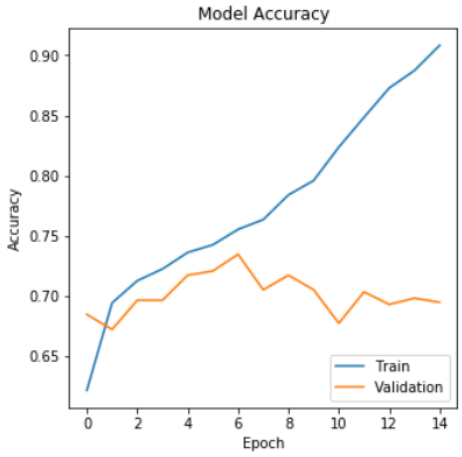
Figure(8.4.7) Performance Metrics

8.5 Base CNN + Preprocess(Standard Deviation=30, alpha=4, beta=-3)

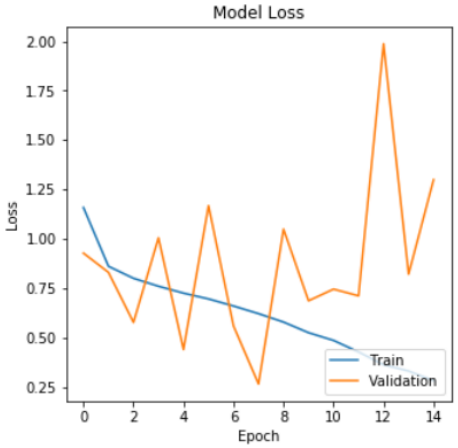


Figure(8.5.1): Normal image

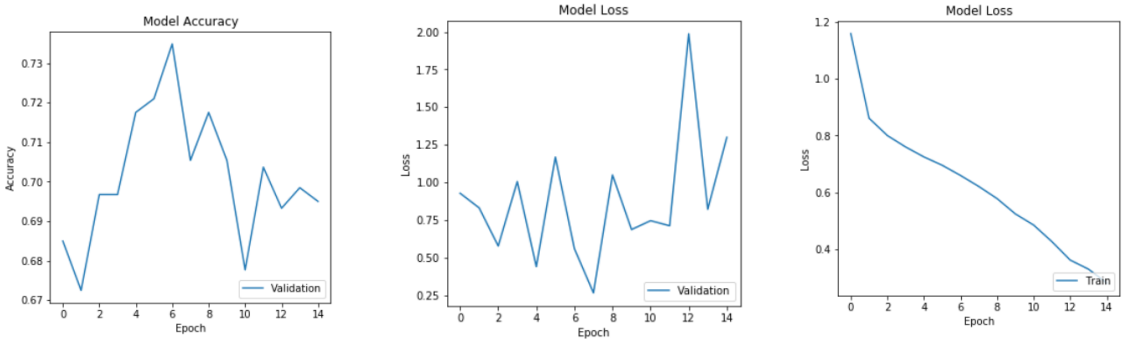
Figure(8.5.2): Image Fed into Neural Ntw



figure(8.5.3): Accuracy Curve

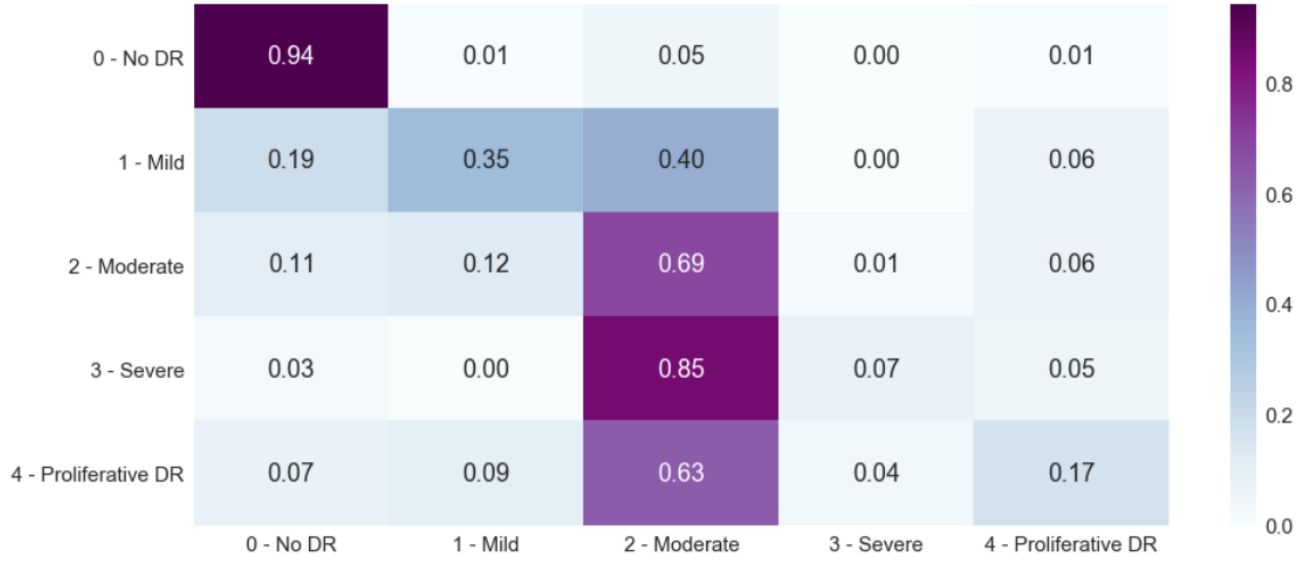


figure(8.5.4): Loss curve



Figure(8.5.5) Validation Accuracy, Validation Loss & Training Loss

With the results obtained, a confusion matrix representing the percentage values are plotted.



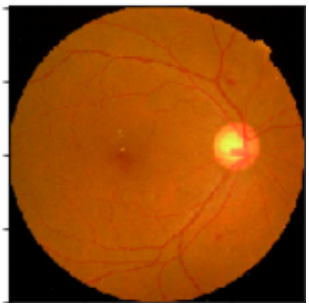
figure(8.5.6) Confusion matrix

```
from sklearn.metrics import confusion_matrix, cohen_kappa_score,accuracy_score
print("Test Cohen Kappa score: %.3f" % cohen_kappa_score(test_preds, df_train_test['diagnosis'].astype('int'),
                                                         weights='quadratic'))
print("Test Accuracy score : %.3f" % accuracy_score(df_train_test['diagnosis'].astype('int'),test_preds))
cm = confusion_matrix(df_train_test['diagnosis'].astype('int'),test_preds)
test_recall = np.diag(cm) / np.sum(cm, axis = 1)
test_precision = np.diag(cm) / np.sum(cm, axis = 0)
print('Recall :%.3f' %np.mean(test_recall))
print('Precision : %.3f'%np.mean(test_precision ))
f1score= 2*(test_recall*test_precision)/(test_precision+test_recall)
print('F1 SCORE : %.3f' %np.mean(f1score))
```

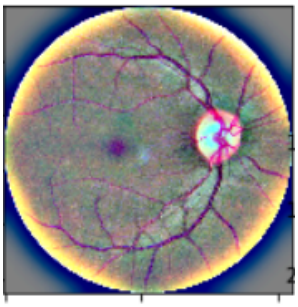
```
Test Cohen Kappa score: 0.715
Test Accuracy score : 0.716
Recall :0.463
Precision : 0.523
F1 SCORE : 0.471
```

Figure(8.5.7) Performance Metrics

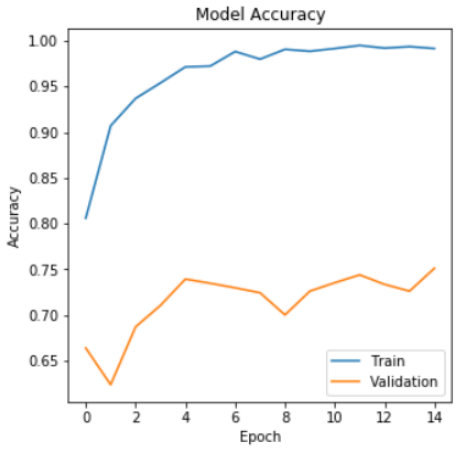
8.6 Base CNN + Preprocess(Standard Deviation=50 alpha=4, beta=-4)



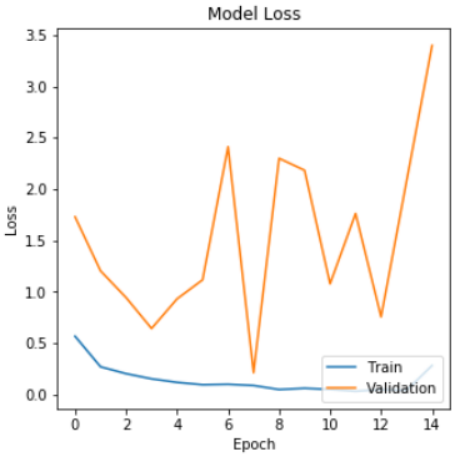
Figure(8.6.1): Normal image



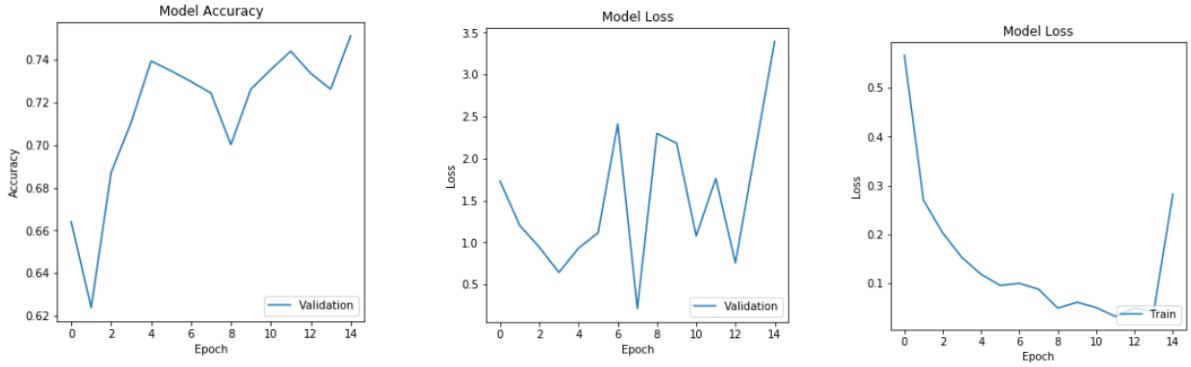
Figure(8.6.2): Image Fed into Neural Ntw



figure(8.6.3): Accuracy Curve



figure(8.6.4): Loss curve



Figure(8.6.5) Validation Accuracy, Validation Loss & Training Loss

With the results obtained, a confusion matrix representing the percentage values are plotted.

Figure(8.6.6) Confusion matrix

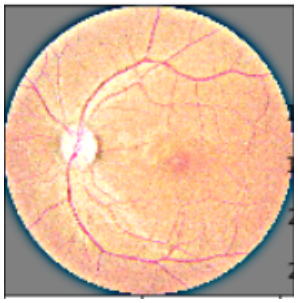
```
print("Test Cohen Kappa score: %.3f" % cohen_kappa_score(test_labels, df_train_test['diagnosis'].astype('int'),
                                                         weights='quadratic'))
print("Test Accuracy score : %.3f" % accuracy_score(df_train_test['diagnosis'].astype('int'),test_labels))
cm = confusion_matrix(df_train_test['diagnosis'].astype('int'),test_labels)
test_recall = np.diag(cm) / np.sum(cm, axis = 1)
test_precision = np.diag(cm) / np.sum(cm, axis = 0)
print('Recall :%.3f' %np.mean(test_recall))
print('Precision : %.3f'%np.mean(test_precision ))
f1score= 2*(test_recall*test_precision)/(test_precision+test_recall)
print('F1 SCORE : %.3f' %np.mean(f1score))
```

```
Test Cohen Kappa score: 0.671
Test Accuracy score :0.738
Recall  :0.557
Precision :0.531
F1 SCORE :0.543
```

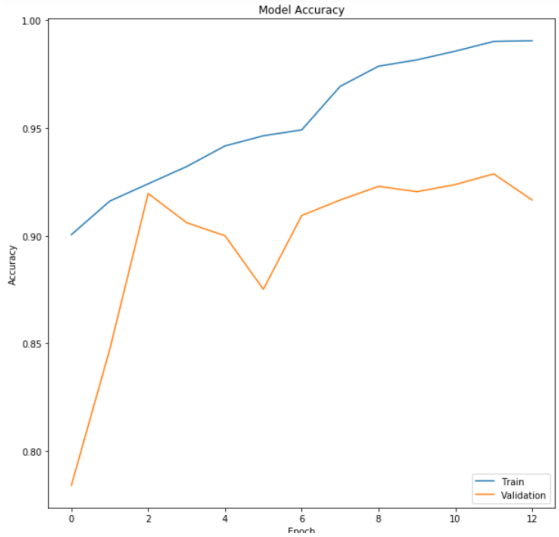
Figure(8.6.7) Performance Metrics

8.7 RESNET50 + Preprocess(Standard Deviation=20 alpha=4, beta=-3)

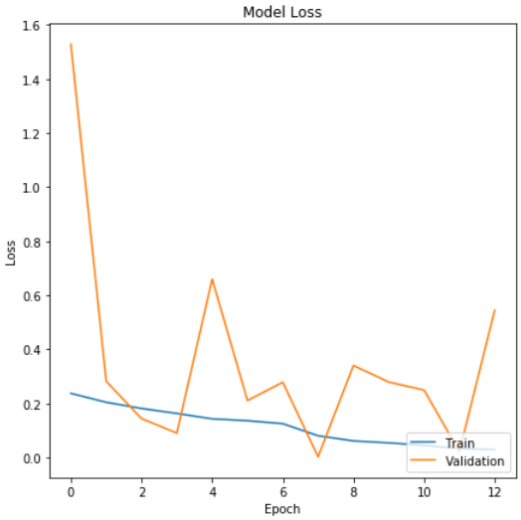
Figure(8.7.1): Normal image



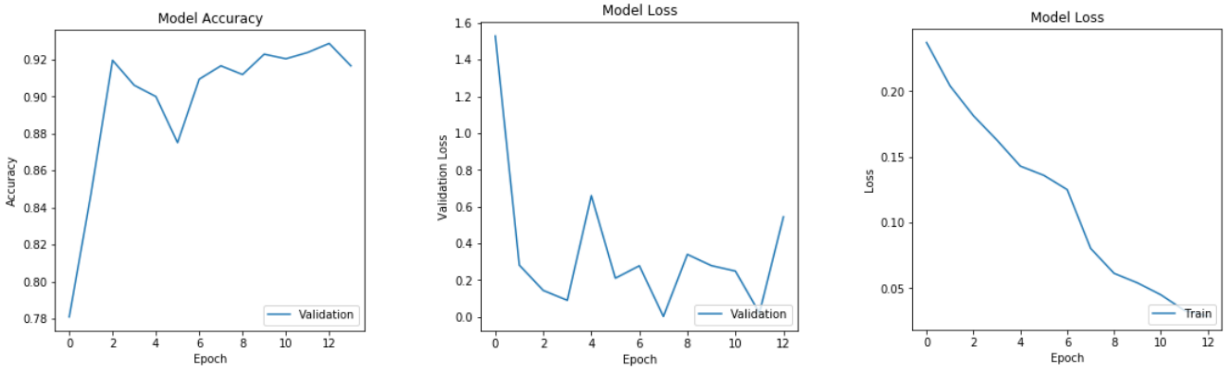
Figure(8.7.2): Image Fed into Neural Ntw



figure(8.7.3): Accuracy Curve

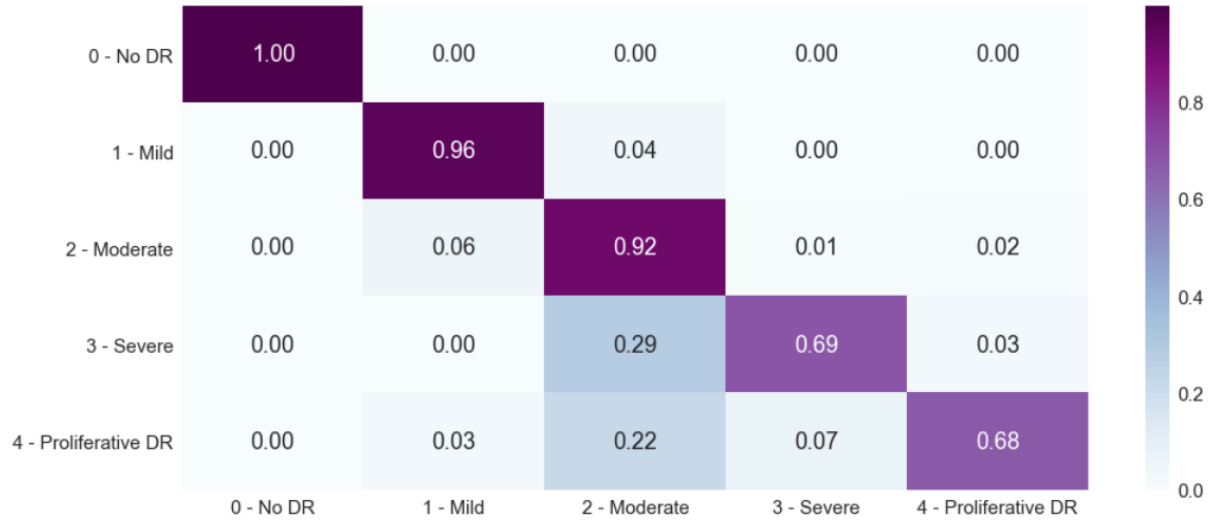


figure(8.7.4): Loss curve



Figure(8.7.5) Validation Accuracy, Validation Loss & Training Loss

With the results obtained, a confusion matrix representing the percentage values are plotted.



Figure(8.7.6) Confusion matrix

```
print("Test Cohen Kappa score: %.3f" % cohen_kappa_score(test_preds, df_train_test['diagnosis'].astype('int'),
                                                         weights='quadratic'))
print("Test Accuracy score : %.3f" % accuracy_score(df_train_test['diagnosis'].astype('int'),test_preds))
cm = confusion_matrix(df_train_test['diagnosis'].astype('int'),test_preds)
test_recall = np.diag(cm) / np.sum(cm, axis = 1)
test_precision = np.diag(cm) / np.sum(cm, axis = 0)
print('Recall :%.2f' %np.mean(test_recall))
print('Precision : %.2f'%np.mean(test_precision ))
f1score= 2*(test_recall*test_precision)/(test_precision+test_recall)
print('F1 SCORE : %.3f' %np.mean(f1score))
```

```
Test Cohen Kappa Score: 0.949
Test Accuracy Score: 0.928
Recall : 0.85
Precision :0.88
F1 SCORE : 0.858
```

Figure(8.7.7) Performance Metrics

Results

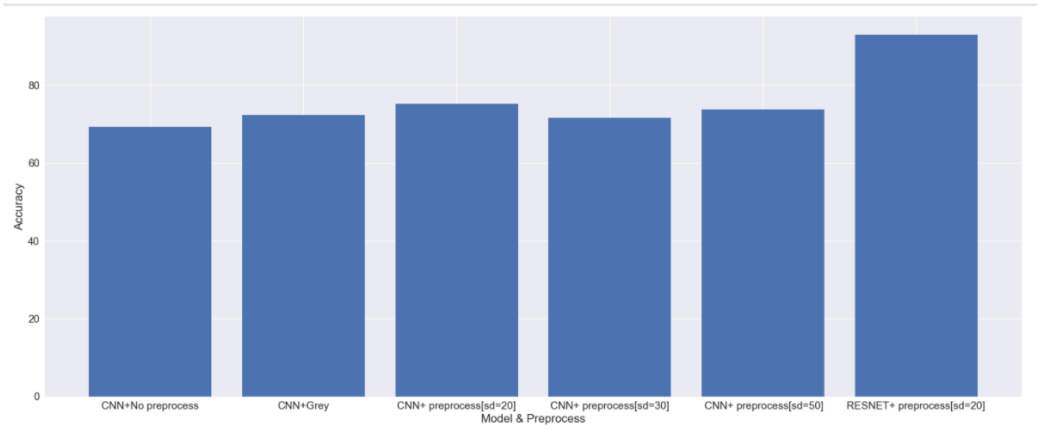
TP: 680
Overall Accuracy: 92.77%

Class	n (truth)	n (classified)	Accuracy	Precision	Recall	F1 Score
1	357	358	99.86%	1.0	1.0	1.0
2	85	74	97.68%	0.96	0.84	0.89
3	210	198	94%	0.92	0.87	0.89
4	31	35	97.54%	0.69	0.77	0.73
5	50	68	96.45%	0.68	0.92	0.78

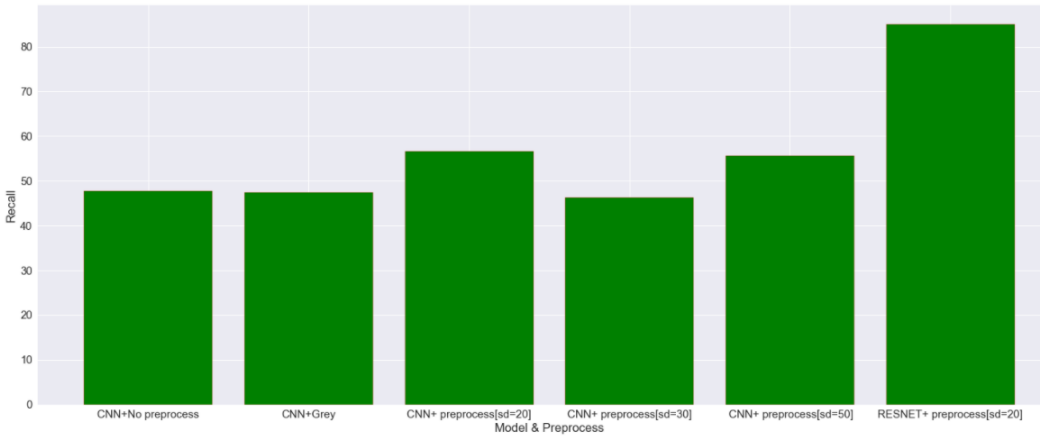
Figure(8.7.8) Each class performance metrics.

8.8 COMPARISON OF MODELS:

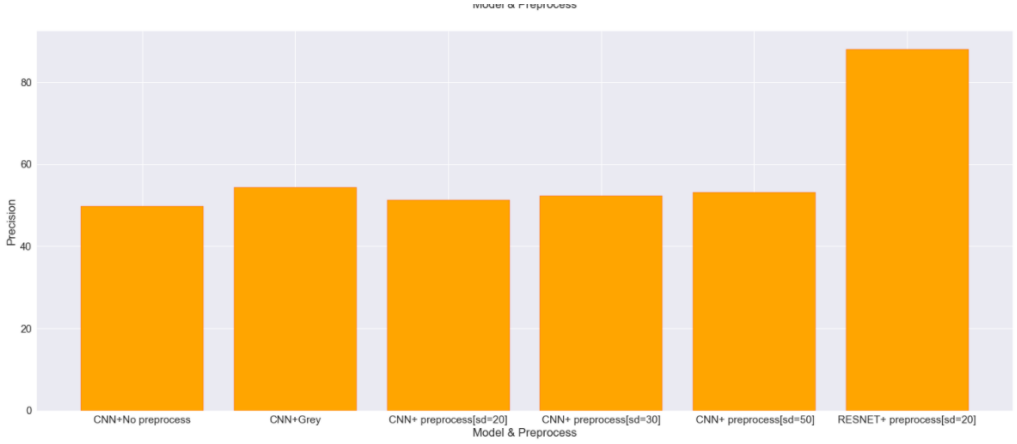
Figures 8.8.1 through 8.8.5 gives us an insight of how each model performed in comparison to each other across the various performance metrics.



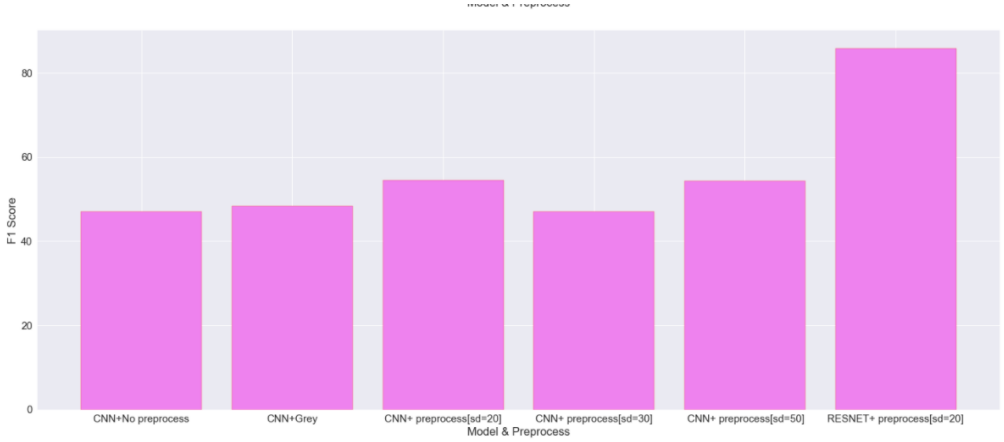
Figure(8.8.1) Accuracy of different models



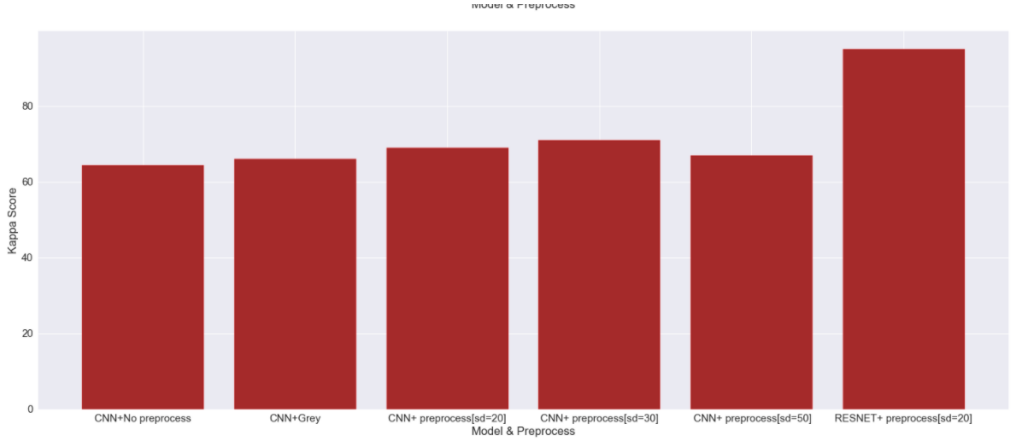
Figure(8.8.2) Recall of different models



Figure(8.8.3) Precision of different models



Figure(8.8.4) F1 score of different models



Figure(8.8.5) Kappa scores of different models

MODELS	KAPPA SCORE	ACCURACY	RECALL	PRECISION	F1 SCORE
CNN+No preprocess	64.3	69.3	47.8	49.7	47.1
CNN+Grey	66	72.4	47.4	54.4	48.3
CNN + preprocess[sd=20]	69	75.2	56.6	51.3	54.5
CNN + preprocess[sd=30]	71	71.6	46.3	52.3	47.1
CNN + preprocess[sd=50]	67	73.8	55.7	53.1	54.3
RESNET+ preprocess[sd=20]	95	93	85	88	85.8

Table 5: Comparison overview of all the models

With the above results, we can conclude that the model which was trained on RESNET50 produced the best results in terms of all the performance parameters with a promising accuracy and kappa scores of 93% and 95% respectively.

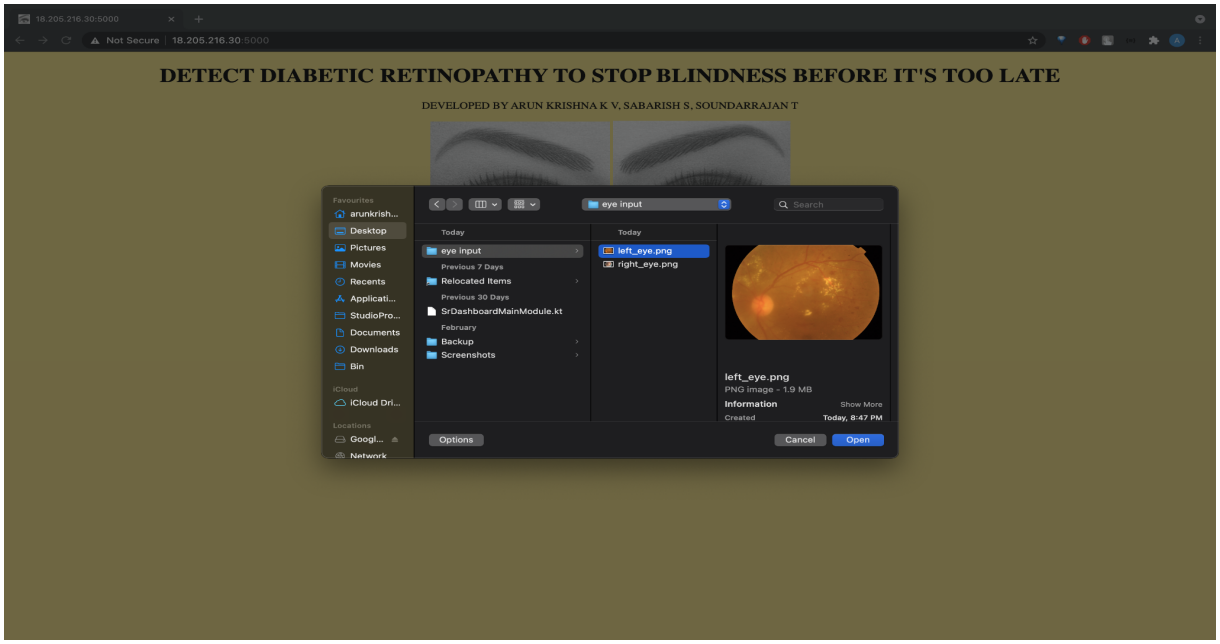
8.9 DEPLOYMENT & TESTING:

The application developed was hosted in Amazon AWS EC2 (Amazon Web Services, Elastic Compute Cloud), the process of which was discussed above in section 7.6

Figure(8.9.1) Homepage of the hosted web app

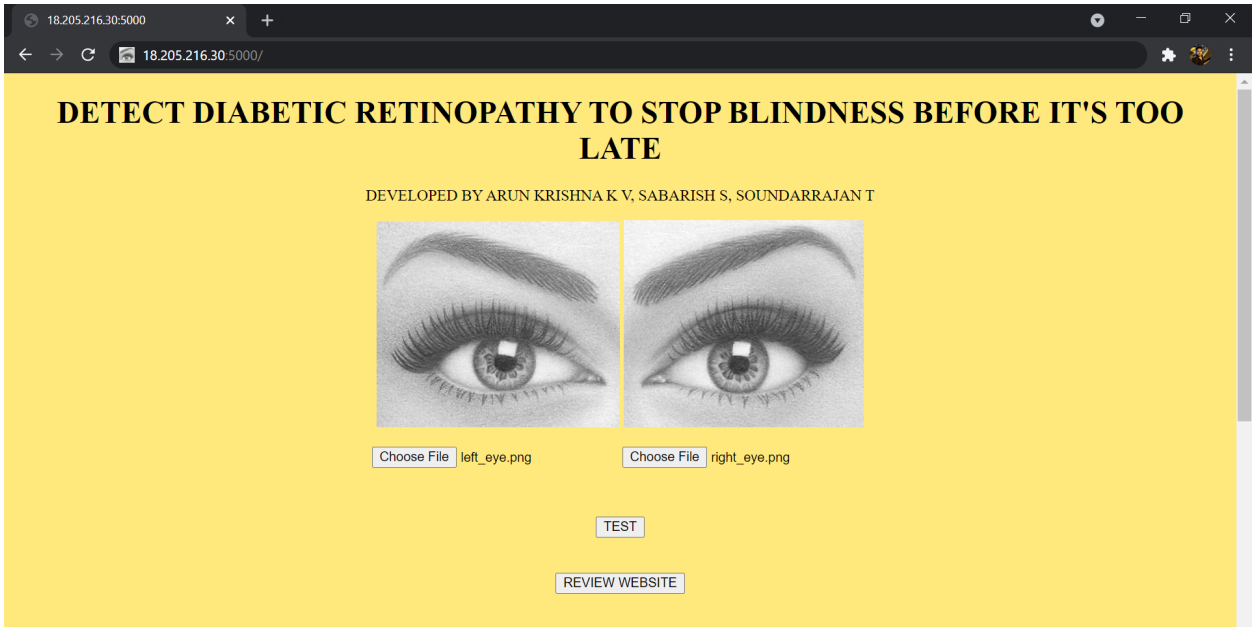
This is the homepage of the app where the user will be able to upload the fundus image of his both left and right eyes for prediction.

.



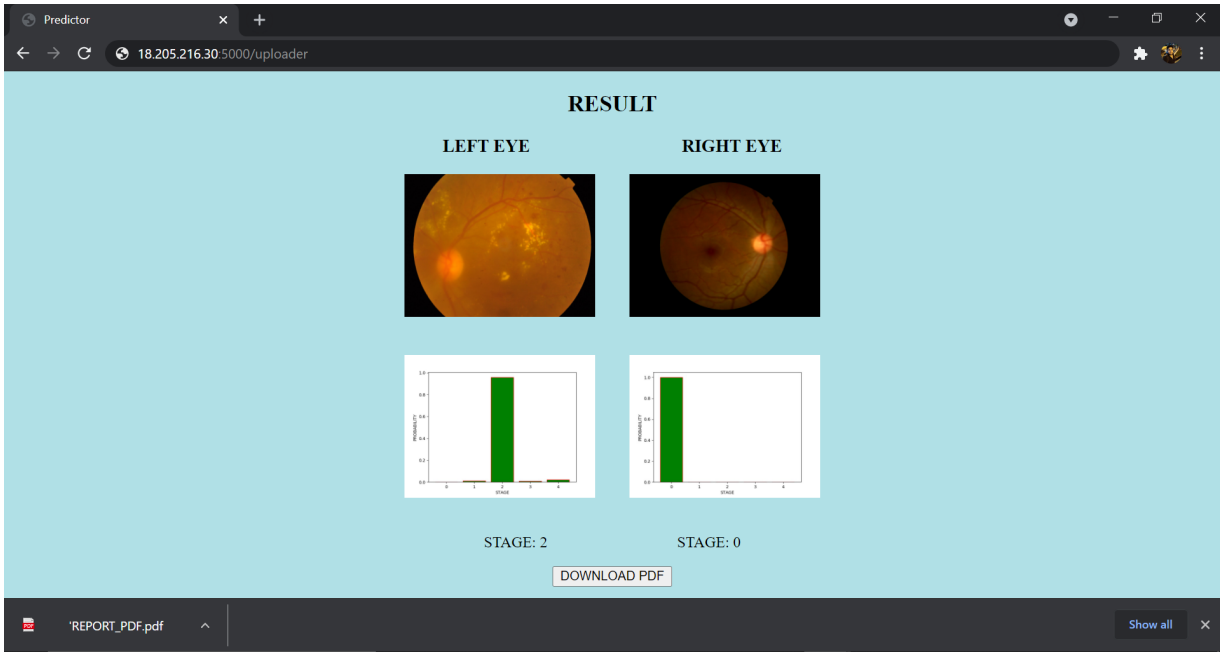
Figure(8.9.2) Upload screen

The above is the window popping up for the user to upload his images.



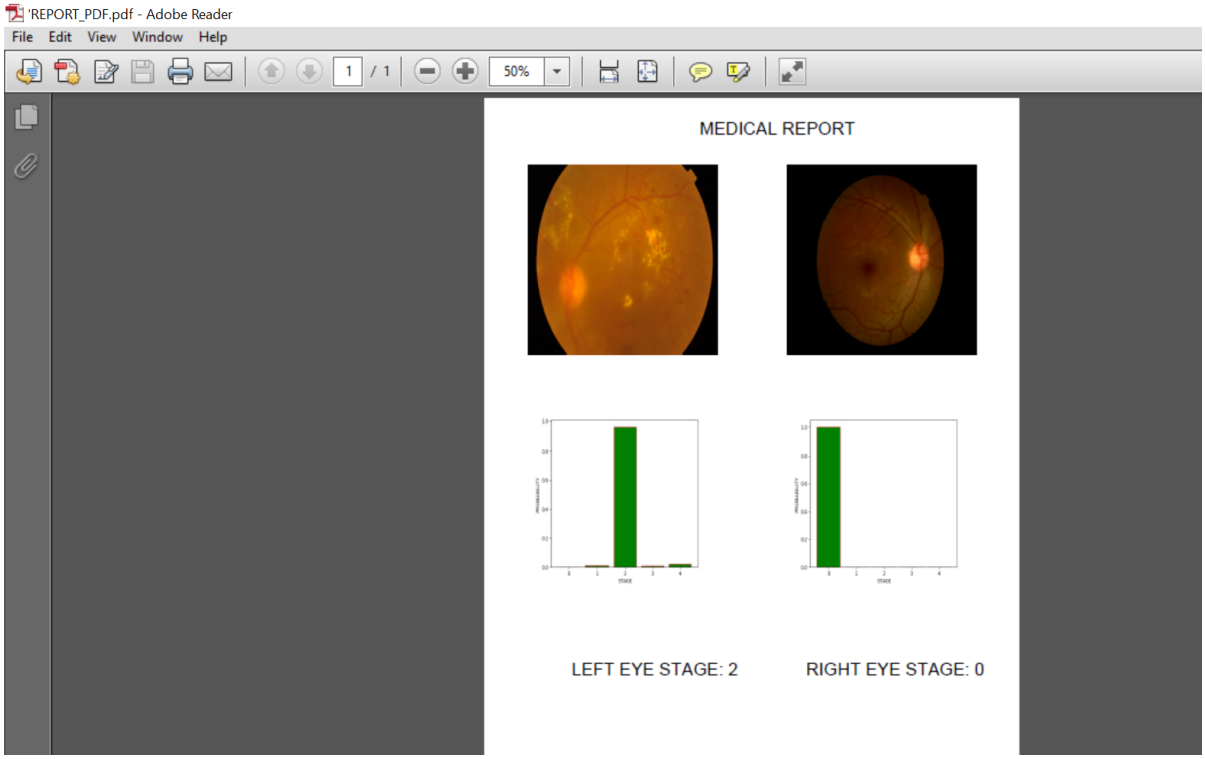
Figure(8.9.3) : Home page with images loaded

Once the user uploads both the images, they can give a Test, which executes the model and predicts the presence and stage of DR.



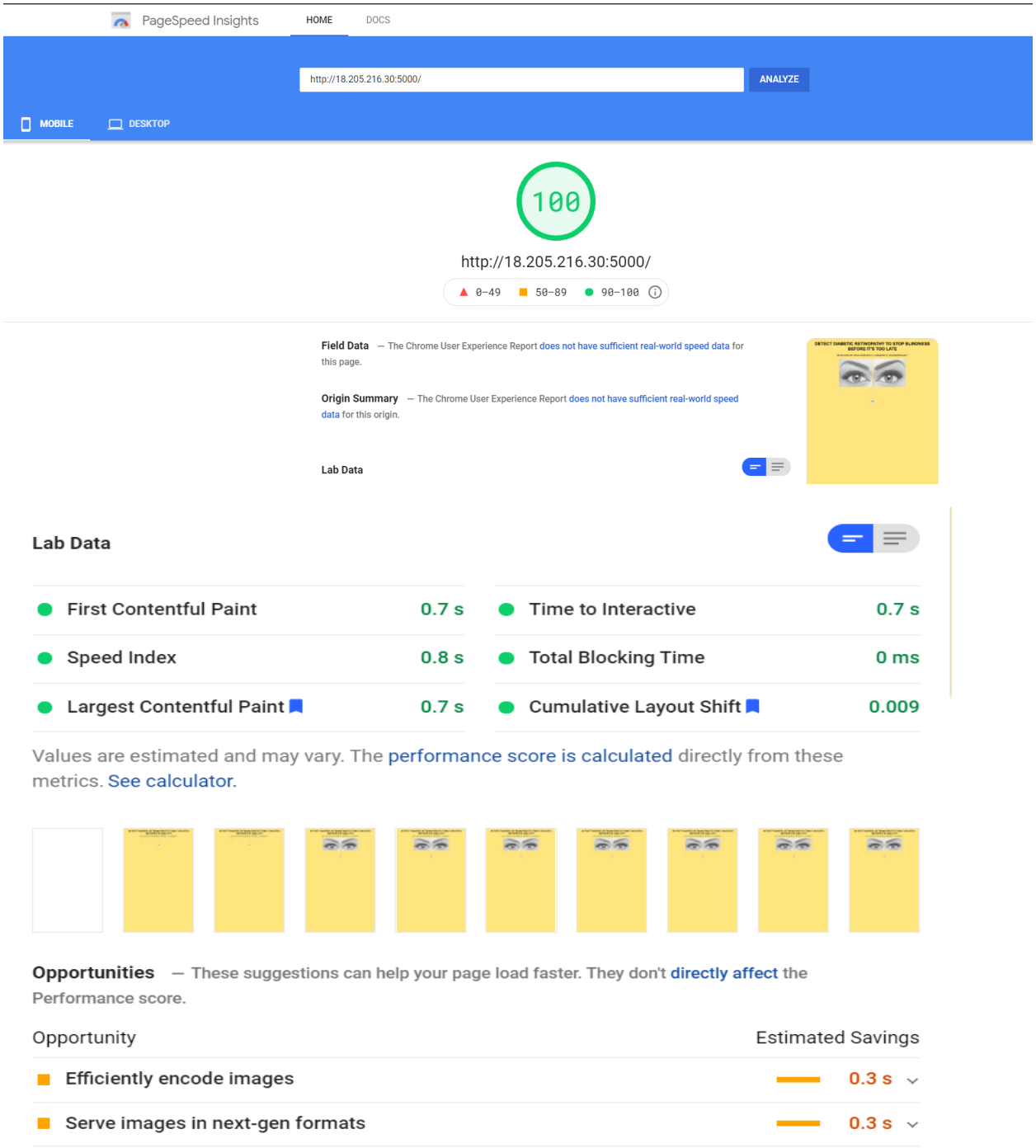
Figure(8.9.4) Result page

Figure(8.9.4) is the result page where the user is displayed with their corresponding result of the predictions. From here, the user is also given a choice to download their report in PDF format as shown in Figure(8.9.5), to approach doctors or ophthalmologists.

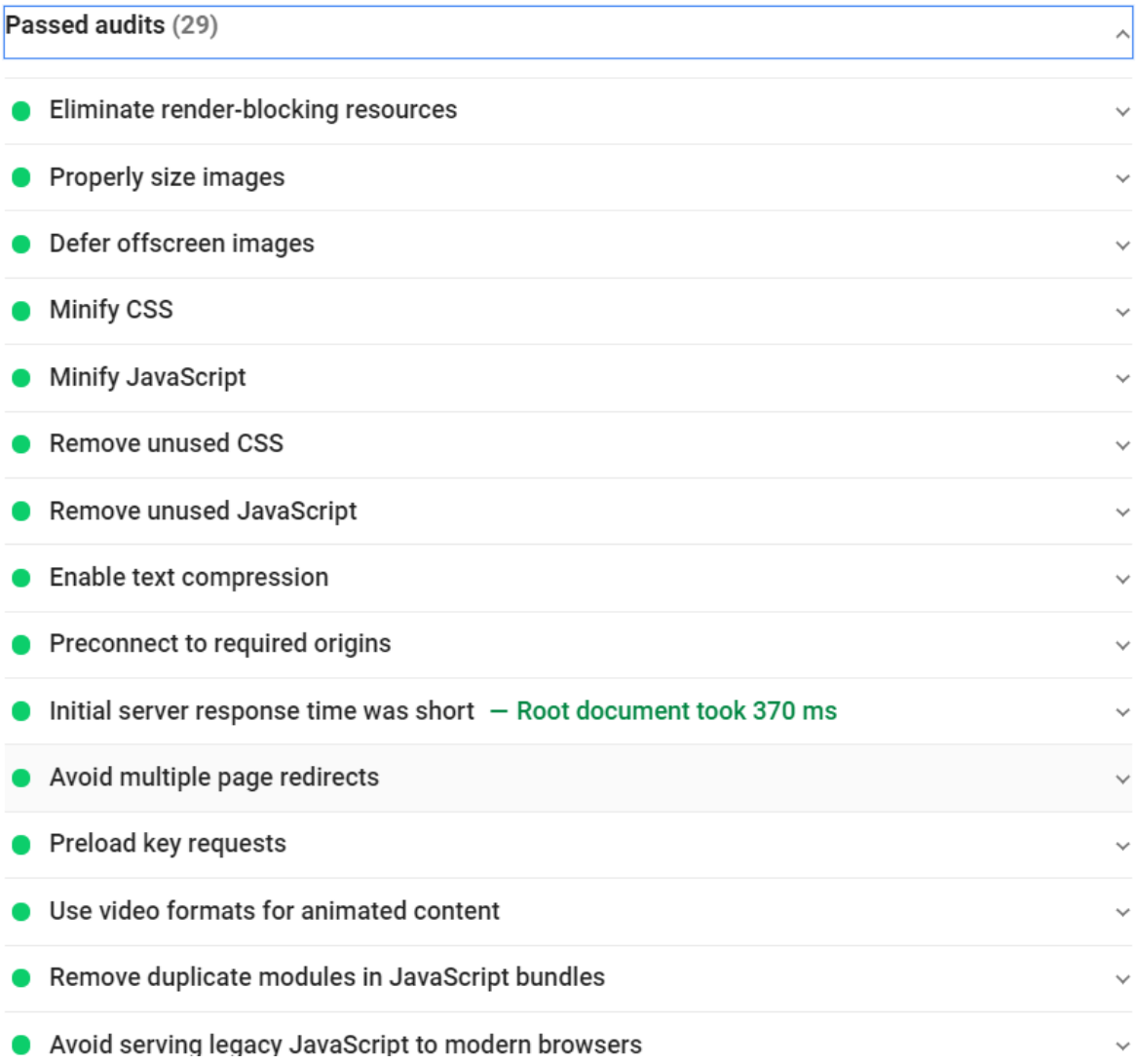


Figure(8.9.5) Report PDF that is downloaded

TESTING OF THE HOSTED WEBSITE

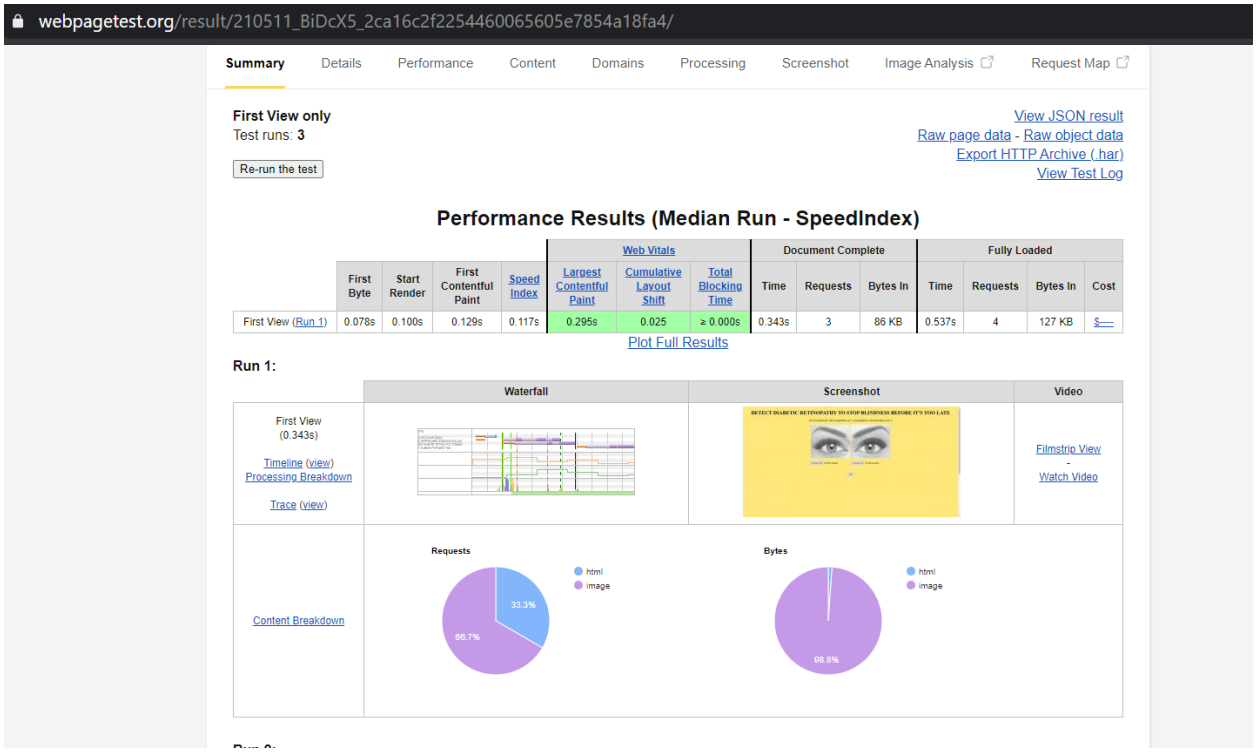


Figure(8.9.6a) Google pagespeed analytics for the hosted website



Figure(8.9.6b) Google pagespeed analytics for the hosted website

The figures 8.9.6a, 8.9.6b are the reports of our website's performance parameters as measured by Google's pagespeed.

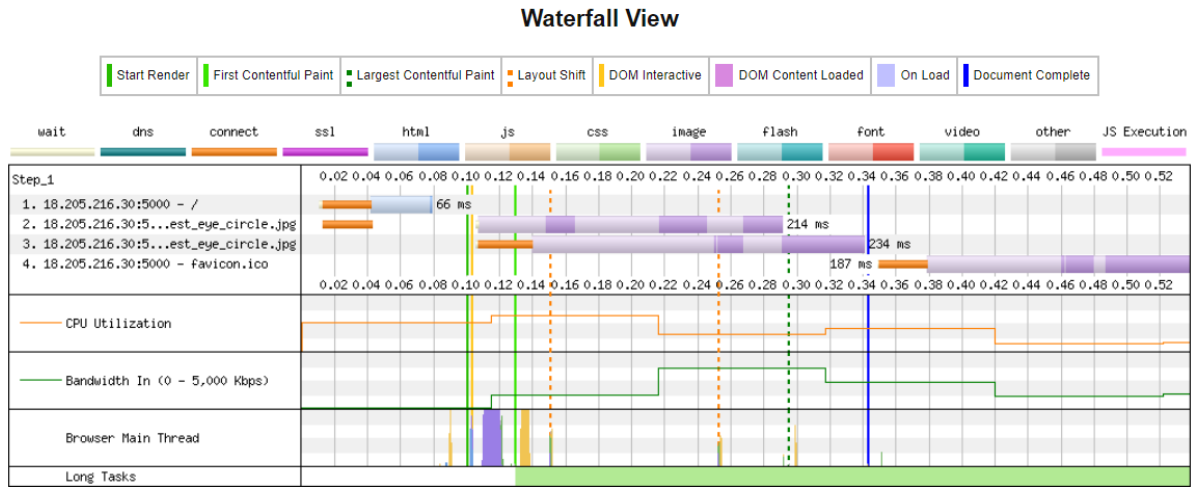


First View only
Test runs: **3**

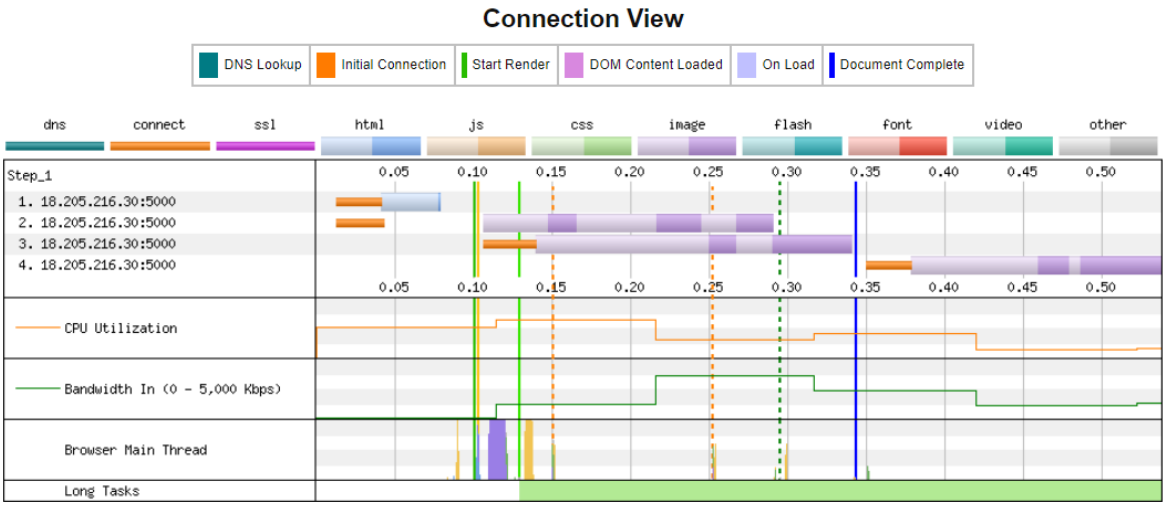
Export HTTP Archive (.har)
Custom Metrics

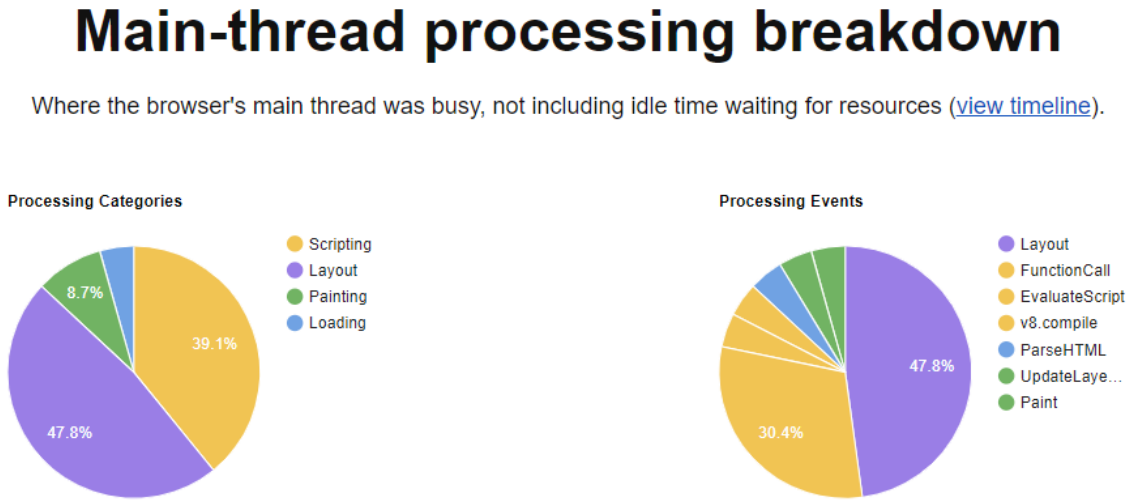
	First Byte	Start Render	First Contentful Paint	Speed Index	Result (error code)	Web Vitals: Largest Contentful Paint	Web Vitals: Cumulative Layout Shift	Web Vitals: Total Blocking Time	Document Complete: Time	Document Complete: Requests	Document Complete: Bytes In	Fully Loaded: Time	Fully Loaded: Requests	Fully Loaded: Bytes In
First View (Run 1)	0.078s	0.100s	0.129s	0.117s	0	0.295s	0.025	≥ 0.000s	0.343s	3	86 KB	0.537s	4	127 KB

Colordepth	dominteractive	domContentLoaded	loadEvent
24	0.103s	0.103s - 0.103s (0.000s)	0.343s - 0.343s (0.000s)



Figure(8.9.7a): webpagetest.org analytics of hosted website





Figure(8.9.7b): webpagetest.org analytics of hosted website

The figures 8.9.7a, 8.9.7b are the reports of our website's performance parameters as measured by webpagetest.org.

CHAPTER 9
CONCLUSION AND FUTURE WORKS

In this study, we have analyzed the effects of various deep learning models and preprocessing techniques in the detection and classification of diabetic retinopathy (DR). Considering our problem of detection of diabetic retinopathy, our proposed method of using Gaussian Blur filtering and subsequently blending the filtered image with the original, as input to the model yields an overall high accuracy. Initially a CNN model was created and trained with un-processed images. Further, to improve the accuracy, the model was trained with greyscale and preprocessed images. The latter produced higher accuracies. Finally, transfer learning technique was used in which those preprocessed images were used to train a pre-trained resnet-50 model. This approach resulted in the highest overall accuracy for classification of DR of 93%.

As future works, live capturing and predicting of DR can be focussed on, where the model is integrated into the optical fundus camera, thereby giving instantaneous results of disease. Also, ensemble learning methods may be used to further improve the test accuracy for the project.

CHAPTER 10
APPENDIX

1. **Quadratic Kappa(cohen Kappa) Score**: The agreement between two ratings is measured using quadratic weighted kappa. This metric usually ranges from 0 (random agreement between raters) to 1 (perfect agreement between raters) (complete agreement between raters). It is calculated as in equation(6).

$$\kappa \equiv \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e},$$

Equation(6): Formula for calculating Quadratic Kappa scores

Where P_0 is probability of relative observed agreement among raters, Pe is hypothetical probability of chance agreement.

2. **Accuracy:** Accuracy is the ratio of correctly predicted observation to the total observations. It is calculated as in equation(7).

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

Equation(7): Formula for calculating Accuracy

3. **Precision:** Precision is defined as the ratio of correctly predicted positive observations to the total predicted positive observations. It is calculated as in equation(8).

$$Precision = \frac{TP}{(TP+FP)}$$

Equation(8): Formula for calculating Precision

4. **Recall (Sensitivity)** - Recall is defined as the ratio of correctly predicted positive observations to the all observations in actual class - yes It is calculated as in equation(9).

$$Recall = \frac{TP}{(TP+FN)}$$

Equation(9): Formula for calculating Recall

5. **F1 score** - F1 Score is defined as the weighted average of Precision and Recall. It is calculated as in equation(10).

$$F1 - score = \frac{2*(Precision*Recall)}{(Recall+Precision)}$$

Equation(10): Formula for calculating F1 score

CHAPTER 11

REFERENCES

1. **https://www.kaggle.com/c/aptos2019-blindness-detection/data**

2. **AUTOMATED IDENTIFICATION AND GRADING SYSTEMS OF DIABETIC RETINOPATHY USING DEEP NEURAL NETWORKS-** ELSEVIER: KNOWLEDGE-BASED Systems 22ND MARCH 2019 Volume 175, 1 July 2019, Pages 12-25. https://doi.org/10.1016/j.knosys.2019.03.016

3. 10.1109/ACCESS.2019.2903171 **Automated Diabetic Retinopathy Detection Based on Binocular Siamese-Like Convolutional Neural Network**

4. **Diabetic retinopathy detection using feedforward neural network** - 10.1109/IC3.2017.8284350

5. **Diabetic retinopathy detection through image mining for type 2 diabetes** 10.1109/ICCCI.2017.8117738

6. **Detection of diabetic retinopathy based on a convolutional neural network using retinal fundus images-** International Conference of Artificial . Neural Netw. New York, NY, USA: Springer, 2017, pp. 635–642 DOI: https://doi.org/10.1007/978-3-319-68612-7_72

7. **Classification of diabetic retinopathy images by using deep learning models**. International Journal of Grid and Computing, vol. 11, no. 1, pp. 89-106, Jan. 2018

8. **Automatic Screening of Diabetic Retinopathy Images with Convolution Neural Network Based on Caffe Framework -**ICMHI '17: Proceedings of the 1st International Conference on Medical and Health Informatics 2017May 2017 Pages 90–94 https://doi.org/10.1145/3107514.3107523

9. **An Automated Model using Deep Convolutional Neural Network for Retinal Image Classification to Detect Diabetic Retinopathy-** ICCA 2020: Proceedings of the International Conference on Computing Advancements January 2020 Article No.: 25 Pages 1–8 https://doi.org/10.1145/3377049.3377067

10. **Diabetic Retinopathy Detection using Deep Learning - Quang H. Nguyen, et al** ICMLSC 2020: Proceedings of the 4th International Conference on Machine Learning and Soft Computing January 2020 Pages 103–107 https://doi.org/10.1145/3380688.3380709

11. **Retina Blood Vessel Detection for Diabetic Retinopathy Diagnosis - Athasart Narkthewan, Noppadol Maneerat** ICBET' 19: Proceedings of the 2019 9th International Conference on Biomedical Engineering and Technology March 2019 Pages 149–152 https://doi.org/10.1145/3326172.3326203

12. **Deep-Learning-Based Early Detection of Diabetic Retinopathy on Fundus Photography Using EfficientNet - Zilin Zhang** ICIAI 2020: Proceedings of the 2020 the 4th International Conference on Innovation in Artificial Intelligence May 2020 Pages 70–74 https://doi.org/10.1145/3390557.3394303

13. **Improving the accuracy of diabetes retinopathy image classification using augmentation - (Aleshan Maistry, Anban Pillay, Edgar Jembere)**SAICSIT '20: Conference of the South African Institute of Computer Scientists and Information Technologists 2020 September 2020 Pages 134–140 https://doi.org/10.1145/3410886.3410914

14. **Fundus Image Classification for Diabetic Retinopathy Using Disease Severity Grading - Aiki Sakaguchi, Renjie Wu, Sei-ichiro Kamata** ICBET' 19: Proceedings of the 2019 9th International Conference on Biomedical Engineering and Technology March 2019 Pages 190–196 https://doi.org/10.1145/3326172.3326198

15. **Deep Convolutional Neural Networks for Diabetic Retinopathy Classification -** ICAIP '18: Proceedings of the 2nd International Conference on Advances in Image Processing June 2018 Pages 68–72 https://doi.org/10.1145/3239576.3239589

16. **Automatic Screening of Diabetic Retinopathy Images with Convolution Neural Network Based on Caffe Framework** ICMHI '17: Proceedings of the 1st International Conference on Medical and Health Informatics 2017 https://doi.org/10.1145/3107514.3107523

17. **Diabetic Retinopathy Detection Based on Deep Convolutional Neural Networks - Yi-Wei Chen, Tung-Yu Wu, Wing-Hung Wong, Chen-Yi Lee IEEE** https://ieeexplore.ieee.org/abstract/document/8461427

18. **Automated detection of diabetic retinopathy using convolutional neural networks on a small dataset - Abhishek Samantaa, AheliSahaa, Suresh ChandraSatapathya, Steven LawrenceFernandesb, Yo-DongZhangc https://www.sciencedirect.com/science/article/abs/pii/S0167865520301483**

19. **https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1**

20. **https://codereview.stackexchange.com/questions/132914/crop-black-border-of-image-using-numpy/132934**

21. **https://medium.com/jun-devpblog/cv-2-gaussian-and-median-filter-separable-2d-filter-2d11ee022c66**

22. **https://www.coursera.org/lecture/machine-learning-projects/transfer-learning-WNPap**

23. **https://medium.com/swlh/t-sne-explained-math-and-intuition-94599ab164cf**

24. https://www.tensorflow.org/guide/keras/custom_callback

25. https://keras.io/api/callbacks/

26. https://docs.opencv.org/3.4/d5/dc4/tutorial_adding_images.html

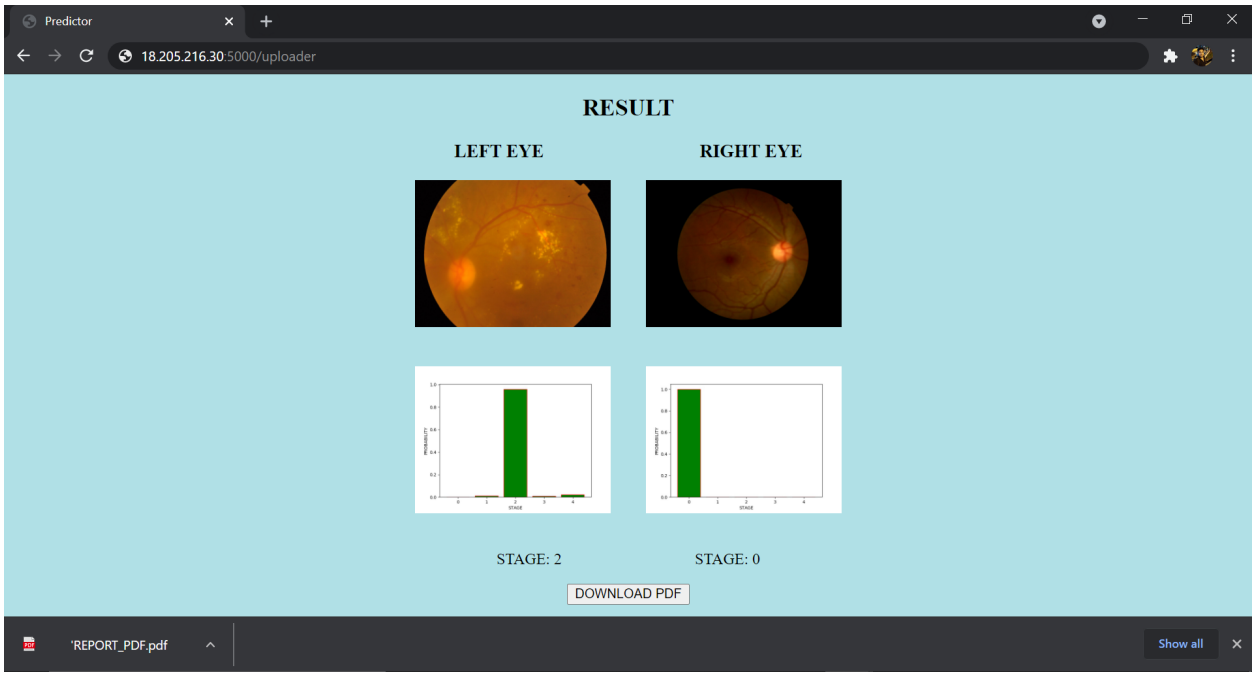
27. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

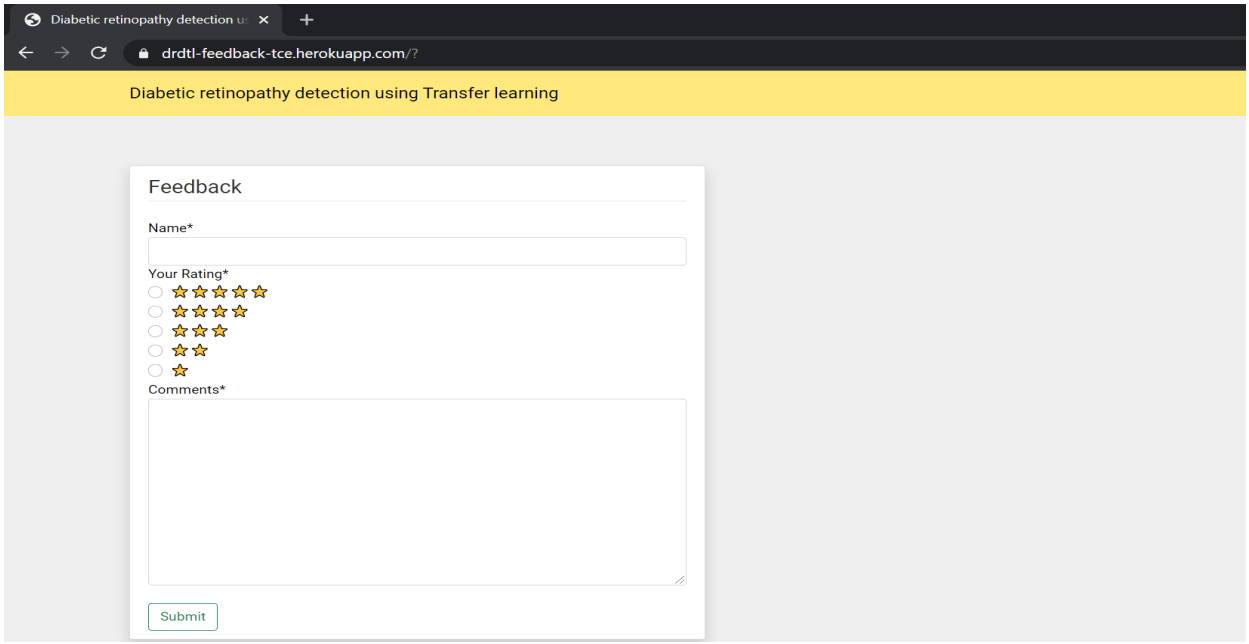
28. https://www.linkedin.com/pulse/keras-image-preprocessing-scaling-pixels-training-adwin-jahn

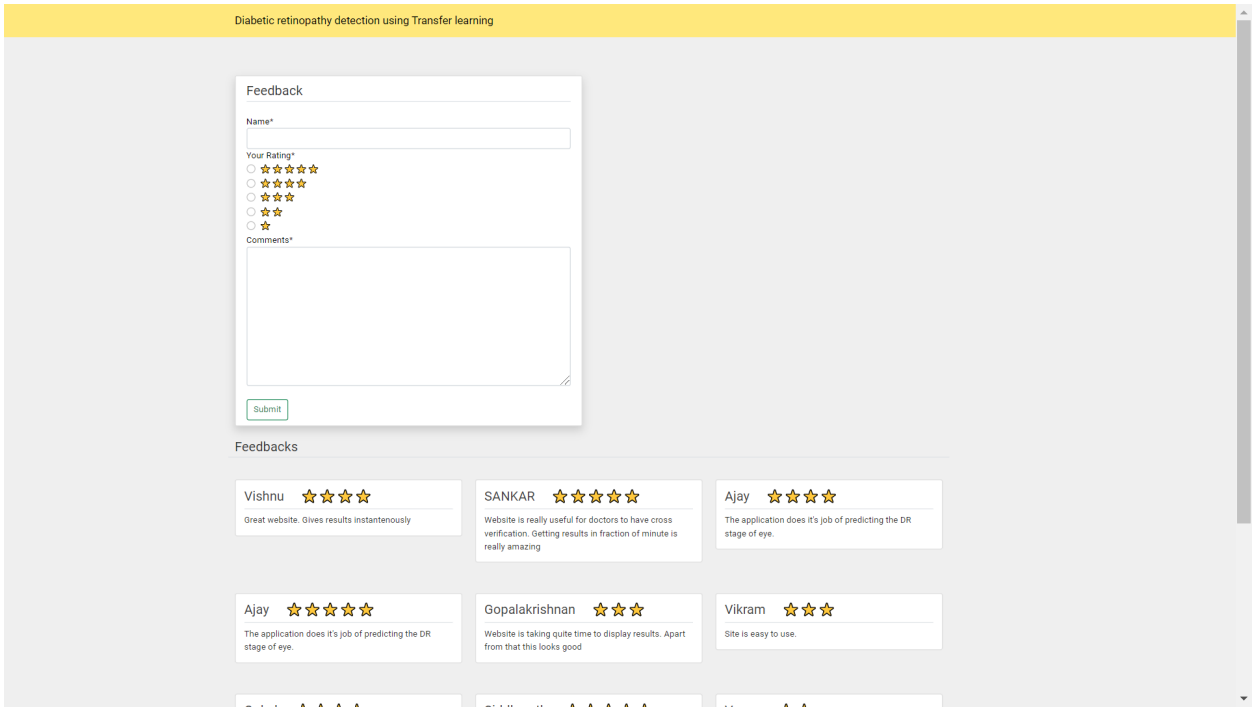
29. https://towardsdatascience.com/an-introduction-to-computer-vision-using-transfer-learning-in-fast-ai-aircraft-classification-a2685d266ac

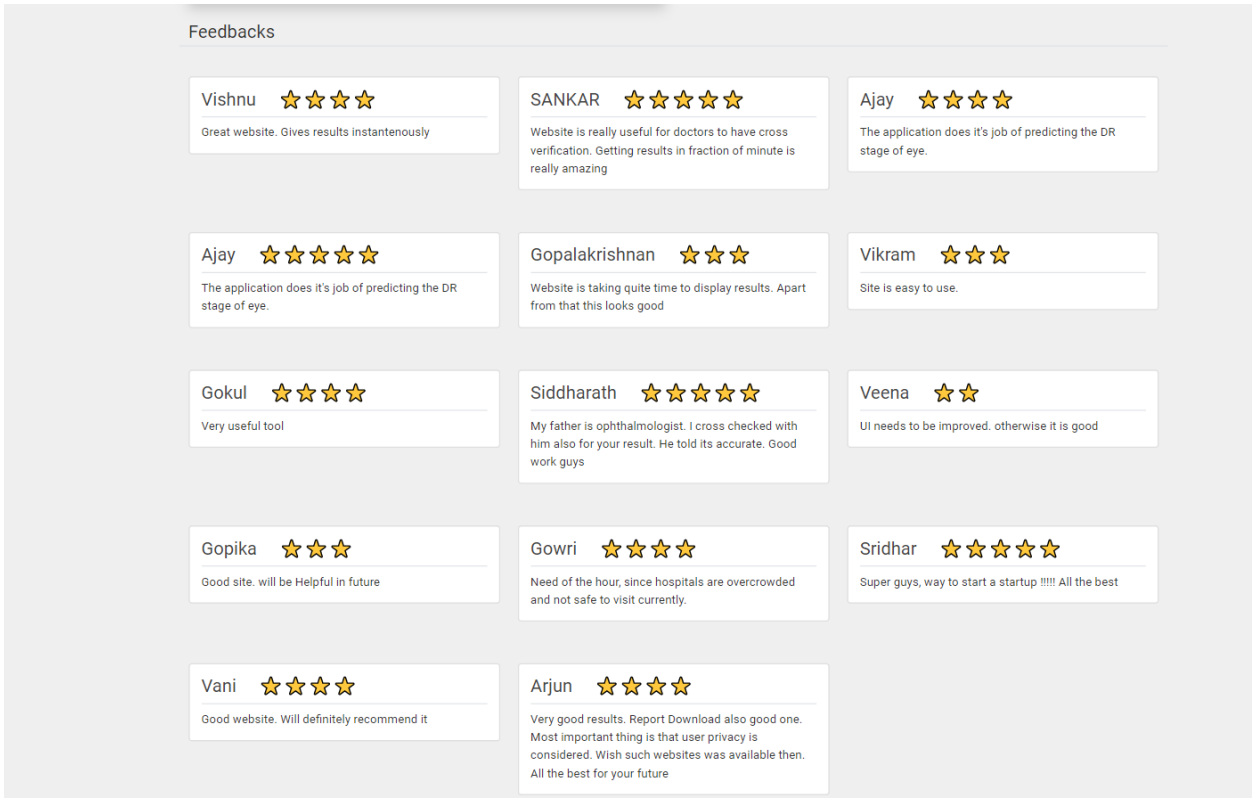
CHAPTER 12

PUBLICATIONS




Diabetic retinopathy detection us
drdtl-feedback-tce.herokuapp.com/?
Diabetic retinopathy detection using Transfer learning
Feedback
Name*
Your Rating*
Comments*
Submit


Diabetic retinopathy detection using Transfer learning
Feedback
Name*
Your Rating*
Comments*
Submit
Feedbacks
Vishnu
Great website. Gives results instantenously
SANKAR
Website is really useful for doctors to have cross verification. Getting results in fraction of minute is really amazing
Ajay
The application does it's job of predicting the DR stage of eye.
Ajay
The application does it's job of predicting the DR stage of eye.
Gopalakrishnan
Website is taking quite time to display results. Apart from that this looks good
Vikram
Site is easy to use.


Feedbacks
Vishnu
Great website. Gives results instantenously
SANKAR
Website is really useful for doctors to have cross verification. Getting results in fraction of minute is really amazing
Ajay
The application does it's job of predicting the DR stage of eye.
Ajay
The application does it's job of predicting the DR stage of eye.
Gopalakrishnan
Website is taking quite time to display results. Apart from that this looks good
Vikram
Site is easy to use.
Gokul
Very useful tool
Siddharath
My father is ophthalmologist. I cross checked with him also for your result. He told its accurate. Good work guys
Veena
UI needs to be improved. otherwise it is good
Gopika
Good site. will be Helpful in future
Gowri
Need of the hour, since hospitals are overcrowded and not safe to visit currently.
Sridhar
Super guys, way to start a startup !!!!! All the best
Vani
Good website. Will definitely recommend it
Arjun
Very good results. Report Download also good one. Most important thing is that user privacy is considered. Wish such websites was available then. All the best for your future

CHAPTER 13

PLAGIARISM REPORT

Document Information

Analyzed document	FINAL YEAR PROJECT REPORT BATCH 3.pdf (D105087440)
Submitted	5/16/2021 11:42:00 AM
Submitted by	
Submitter email	sabarish@student.tce.edu
Similarity	6%
Analysis address	kskcse.tcoe@analysis.ouriginal.com