A Project Report On

DETECTION OF DIABETIC RETINOPATHY USING MACHINE LEARNING AND IMAGE PROCESSING

Submitted By :

Ritorshi Pal - 16900119089
Rakesh Dey - 16900119091
Gargee Roy - 16900119107
Debaleena Ghosh - 16900119108
Bimal Maity - 16900119109

Department- CSE, Semester- 8th
Subject: Project-III (PROJ- CS881)

Under the guidance of
Prof. Suman Bhattacharya

A Project Report
To be submitted in the partial fulfillment of the requirements
For the degree of
Bachelor of Technology in Computer Science and Engineering

Department of Computer Science and Engineering,
Academy of Technology

Affiliated to

Maulana Abul Kalam Azad University of Technology, West Bengal, 2023

CERTIFICATE

This is to certify that the project entitled: **DETECTION OF DIABETIC RETINOPATHY USING MACHINE LEARNING AND IMAGE PROCESSING** submitted to MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY in the partial fulfillment of the requirement for the award of the B.TECH degree in COMPUTER SCIENCE AND ENGINEERING of **Project-III(PROJ-CS881)** is carried out by

Ritorshi Pal - 16900119089
Rakesh Dey - 16900119091
Gargee Roy - 16900119107
Debaleena Ghosh - 16900119108
Bimal Maity - 16900119109

under my guidance. The matter embodied in this project is genuine work done by the students and has not been submitted whether to this University or to any other University/Institute for the fulfillment of the requirement of any course of study.

Prof. Suman Bhattacharya (Guide)
Department of Computer Science
and Business Systems
Academy of Technology**,** Aedconagar,
Hooghly-712121, West Bengal, India

Dated:
Countersigned By

Prof. Prasenjit Das
Head, Department of Computer Science and Engg.
Academy of Technology, Aedconagar,
Hooghly-712121, West Bengal, India

STATEMENT BY THE CANDIDATES

Ritorshi Pal , Roll - 16900119089
Rakesh Dey , Roll - 16900119091
Gargee Roy , Roll - 16900119107
Debaleena Ghosh , Roll - 16900119108
Bimal Maity , Roll - 16900119109

B. Tech 8th Semester
Dept. of Computer Science
Academy of Technology

We hereby state that the Project Report entitled "**Detection Of Diabetic Retinopathy using Machine Learning and Image Processing**" has been prepared by us to fulfill the requirements of **Project-III (PROJ-CS881) during the period January 2023 to June 2023.**

_____

_____

_____

_____

_____

Signature of the students

TABLE OF CONTENTS

Chapter 1 - Introduction

Diabetes is a chronic disease caused by either failure of insulin production in the body or inability to resist insulin in the body. Complications of Diabetes lead to heart disorders, vascular disease and stroke, Kidney disease, Neuropathy and Diabetic eye disease known as “diabetic retinopathy”. Diabetes for a prolonged time damages the blood vessels of the retina and thereby affecting seeing ability of a person and leading to diabetic retinopathy. Diabetic retinopathy is classified into two categories, non-proliferative diabetic retinopathy (NPDR) and proliferative diabetic retinopathy (PDR).

In this paper, detection of diabetic retinopathy in fundus images is done by image processing and machine learning techniques. Probabilistic Neural Network (PNN), Bayesian Classification and Support vector machines (SVM) are the three models adopted for detection of diabetic retinopathy in fundus image and their results analyzed and compared. The amount of the disease spread in the retina can be identified by extracting the features of the retina. The features like blood vessels, hemorrhages of NPDR image and exudates of PDR image are extracted from the raw images using the image processing techniques and fed to the classifier for classification. A total of 350 fundus images were used, out of which 100 were used for training and 250 images were used for testing. Experimental results show that PNN has an accuracy of 89.6 % Bayes Classifier has an accuracy of 94.4% and SVM has an accuracy of 97.6%.

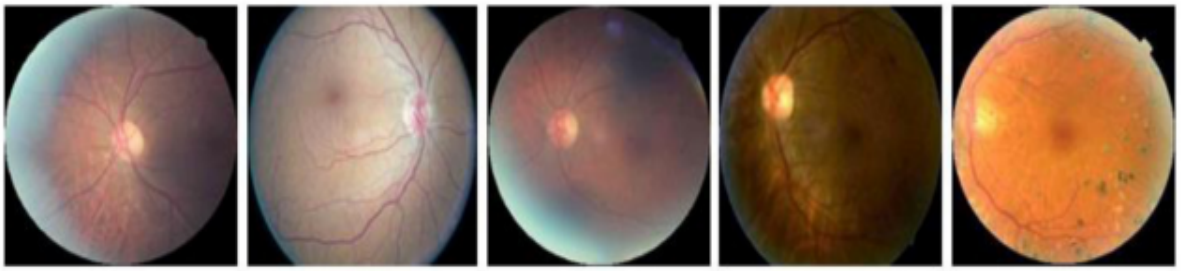
1.1. Purpose of This Study :

India is said to be the diabetic capital of the world by 2030 with over 80 million people affected by it. Unfortunately, more than 2/3rds of them are from the “subaltern”. If identified early, this is a blindness which can be avoided.

This project mainly focuses on the prediction of diabetic retinopathy disease. CNN models can be trained by using training datasets and CNN will give the probability of the eye infected with diabetics. Our objective is to train our model by providing training datasets to it and our goal is to detect the severity of diabetic retinopathy disease accurately. An ML model on the edge, with focus on privacy, scalability and interpretability. Matching cases based on severity and time for immediate diagnosis.

.

1.2. Brief Overview of the Project Report :



Different stages of DR: (a) no DR, (b) mild, (c) moderate, (d) severe, and (e) PDR.

By using ML algorithms, we train the machine with a dataset of images. We first separated the dataset of images into 70%-30% and sent the bigger set into training. We keep the smaller set for testing. The machine is first trained by the marked training images through classification. The algorithm then studies the markings. The more accurate markings we have, the better. After training is done, the testing image set is passed to validate whether the algorithm can accurately mark out the abnormalities on the test images. If yes, then good. If not, we train the algorithm with more images and run more epochs till we get the most accurate results.

Chapter 2-Literature Overview

In the last few years, several diabetic-related health issues have been rising worldwide. DR, caused by diabetes, may lead to blindness, and for preventing this, early diagnosis is necessary.The traditional measure to identify DR involves ophthalmologists for assessment and diagnosing capability, which is time-consuming and costly work. Hence, it became crucial to present efficient DL-based methods. DL has now become an interesting research area and has achieved superb performances in the domain of image processing, especially in DR identification. Innovative and intricate DNN frameworks are being designed to resolve several computerized works. In this review paper, first the collection of retinal datasets is briefly outlined and then DL methods are discussed.The adoption of various approaches has been explored to identify the retinal irregularity,then the performance evaluation metrics have been briefly reviewed for automated detection models. In the report, it was examined that almost a scholarly job has been carried out by utilizing CNN models to produced deep multilevel models for the detection of DR employing digital retinal photographs. Advantages of carrying out DL-based methods in DR-screening include reduced reliance on human power, expenses of screening, and concerns related to intra- and intergrader variability. Despite the fact that the significance of DL is rising and various positive outcomes in its research are reaching heights, there remain challenges that need to be addressed. Automated diagnosis of a DR image encounters two major challenges: technical fluctuation in the imaging procedure and patient-to-patient inconsistency in pathological indications of illness.

➢ Clinical challenges :

• Variation in DR classification systems leads to minor differences on each of the DR severance scales. As a result, models developed by employing one DR classification system cannot carry out as expected when tested on a database ranked utilizing a different classification system, though both systems apparently measure the equal clinical situation. Thus it is essential to assure the normalization of the accuracy among the training and testing databases to assure the robustness of the models developed.

• Various reference standards, with accuracy achieved from a type of retinal expert, general ophthalmologists, professional graders, and optometrists remain a challenge.

• There exist several opinions even when functioning under an equal classification structure; for instance, an intergrader agreement among professionals may dispute the medical ground truth.

➢ Technical challenges :

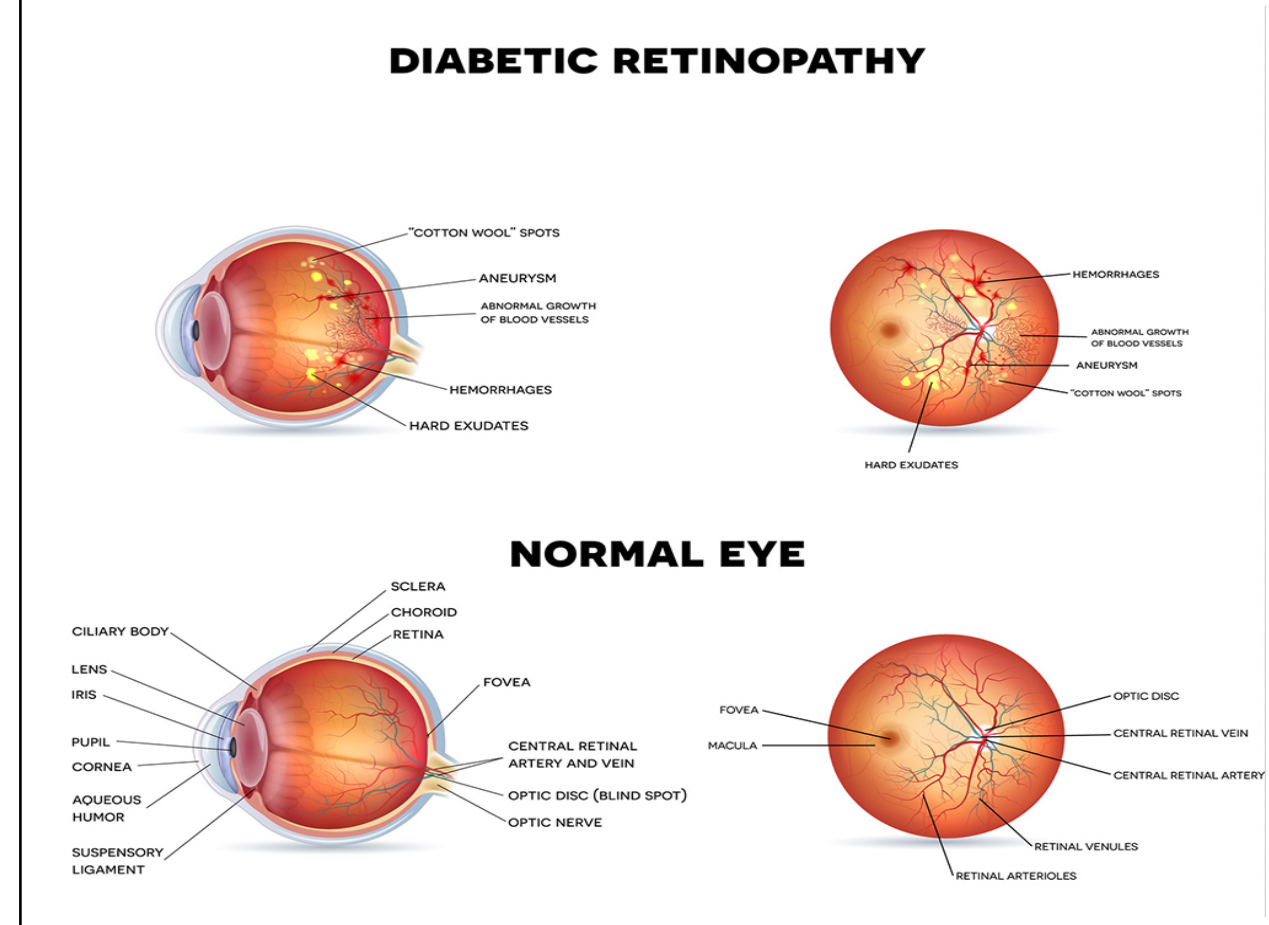
• Another constraint of usage of DL with health sector encounters is the size of retinal databases required for training of DL models. As the outcome of the DL model relies on the quality and size of the training data, the present public database sizes are required to be increased, whereas the big size database such as Kaggle is required to remove the unlabeled and low-quality data.

• Although DL systems may be very proficient and capable of developing an appropriate solution to a specific challenge after data training is done.

Chapter 3 : Problem Definition & Objectives

3.1 Problem Statement :

"Detection of diabetic retinopathy using machine learning and image processing."



3.2 Problem Elaboration :

Diabetic retinopathy (DR) is a disease which causes blindness in people having diabetes. Currently, to detect DR, medical staff has to thoroughly examine images of the retina manually taken by the technique of Fundus photography. This is time consuming. We proposed a model to detect DR using machine learning techniques such as Neural networks to make the detection process automated as well as accurate.

3.3 Proposed Methodology :

Machine learning consists of a number of stages to detect retinopathy in the fundus images that includes converting image to suitable input format, denoising and various preprocessing techniques. It also includes training a model with a training set and validating with a different testing set. Method proposed in this project can be listed in two steps: Image Preprocessing, and Supervised learning and Feature Extraction.

First, the images are preprocessed. Pooling is performed to resize the images. During preprocessing, the features of the object/objects that are to be detected are specified. As the images are heterogeneous they are compressed into a suitable size and format. Layer separation will also be performed. To make the image non-linear RELU is performed. For making intensity variations uniform histogram equalization to the image can be applied. Morphological operation will be done to remove the noise present in the background of the retinal image. We then plan to use CNN for feature extraction and prediction of the class of DR. RESNET model based on CNN is used. A CNN is able to capture the temporal and spatial dependencies in images and fits better due to the decrease in parameters used and weight reusability. It has the ability to train to understand the complexity of the image more efficiently.

Chapter 4 : Feasibility Study

Several tests are used for diabetic retinopathy screening. The sensitivity (the ability of the test to detect disease, if it is present) and specificity (the ability of the test to find there is nothing wrong, if there is no disease) are important factors in choosing a test, but policy-makers need to be aware that information on test performance for diabetic retinopathy is not straightforward to interpret, because:

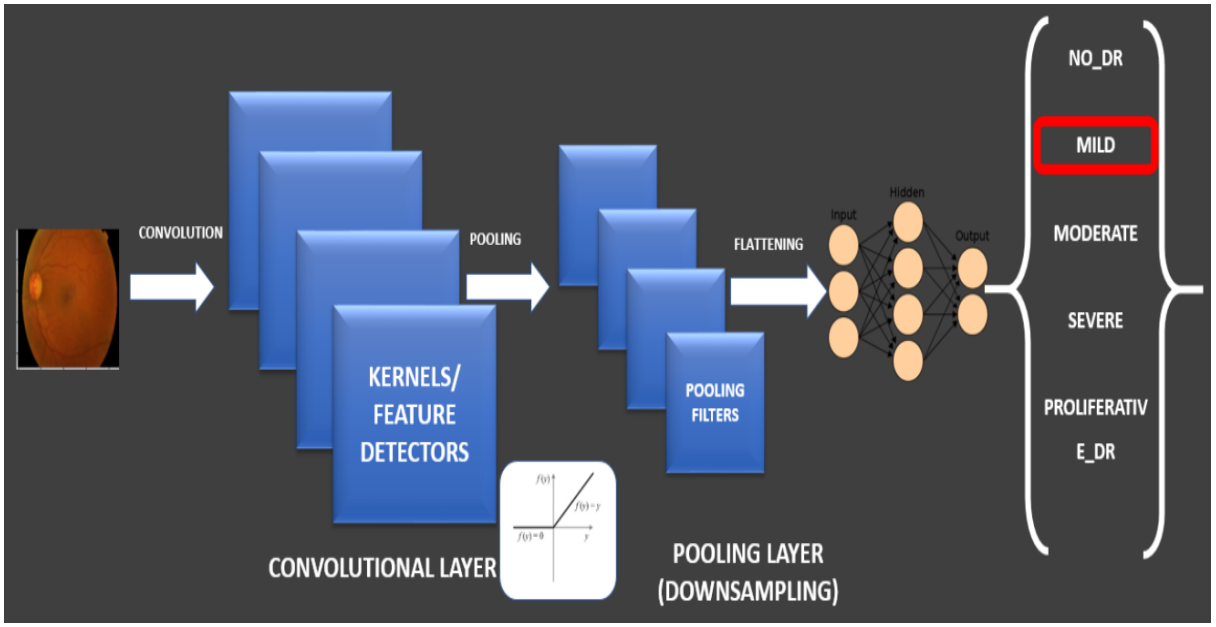
• Researchers use different outcomes to measure sensitivity, such as the ability of a test to pick up any retinopathy compared to sight-threatening diabetic retinopathy; and

• Some tests are better than others for detecting diabetic macular oedema compared to the different grades of diabetic retinopathy. Test performance will need to be considered against other factors, such as cost and ease of use. Cost is not a barrier, the preferred method for screening is digital retinal photography, which provides quality images and an ability to store and audit images. Sensitivity of different tests can vary according to who does the examination and how well they are trained, which is important for tests such as direct ophthalmoscopy. A systematic review that compared sensitivity of test by operator found the sensitivity of direct ophthalmoscopy by general practitioners varied between 25% and 66% compared to 43% and 79% for ophthalmologists.

Diabetic retinopathy is a rapidly developing field. Policy-makers will need to ensure they are referring to the most up-to-date evidence and are planning a programme that can respond to changing evidence and new technology. New types of technologies, such as automated image-grading systems and hand-held cameras, are being developed alongside diabetic retinopathy screening techniques and may offer new methods for screening in the future. Diabetic retinopathy screening is also likely to benefit from further developments in artificial intelligence-based technologies for image capture and analysis, offering opportunities to improve the quality of imaging and grading. Policy-makers should work with clinicians and academics to review the evidence base of these emerging technologies and assess their cost–effectiveness and affordability.

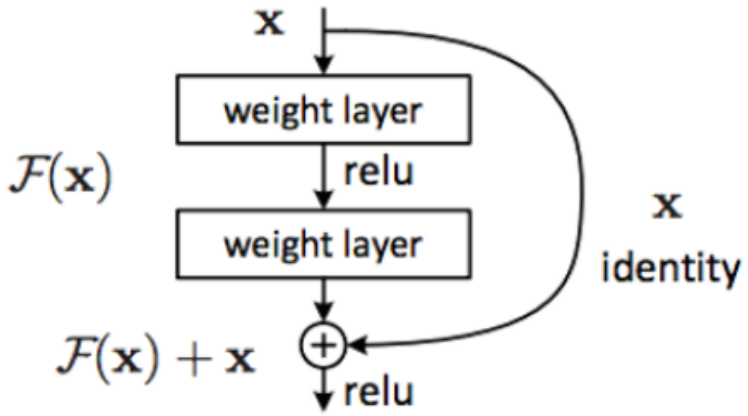
Chapter 5 : System Analysis

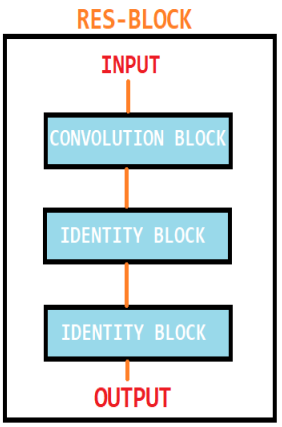
Convolutional Neural Network :

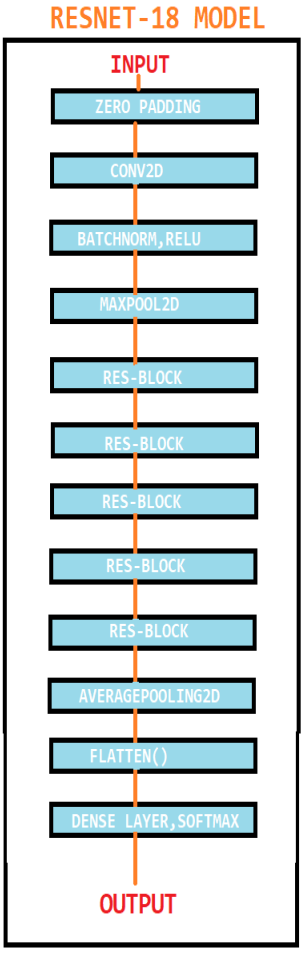


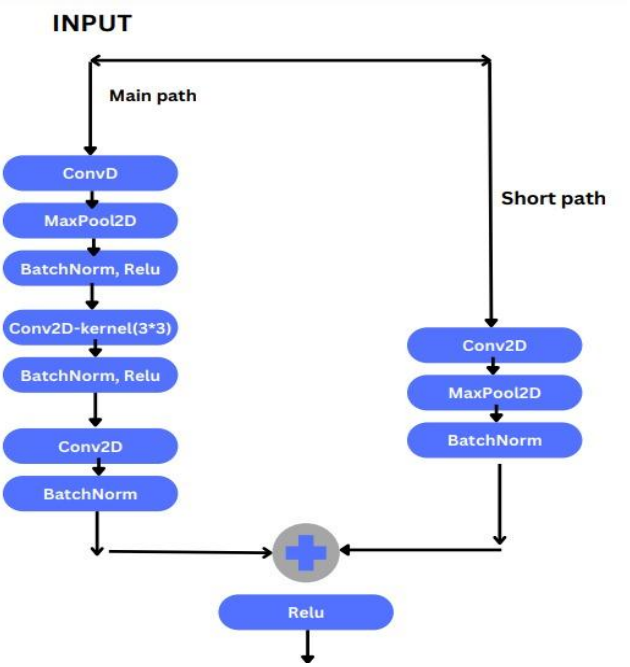
RESNET(Residual Network)

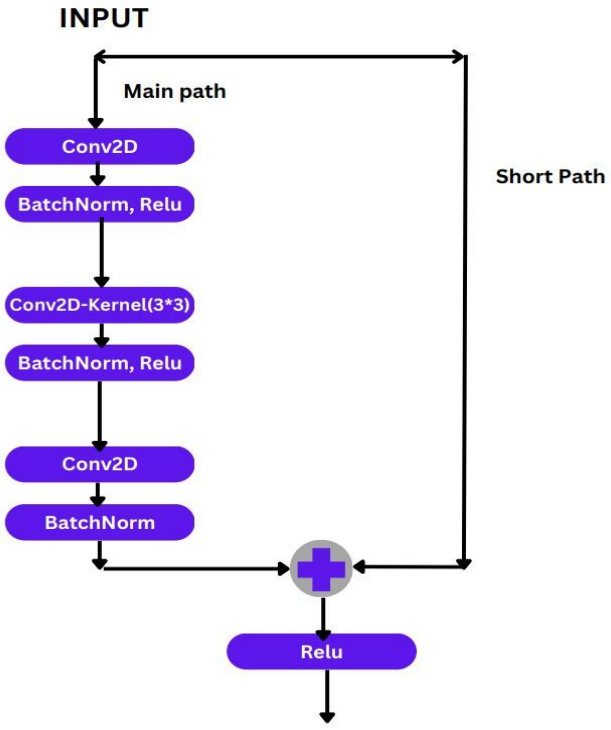
- As CNNs grow deeper, vanishing gradients tend to occur which negatively impact network performance.
- Vanishing gradient problem occurs when the gradient is back-propagated to earlier layers which results in a very small gradient.
- Residual Neural Network includes a “skip connection” feature which enables training of 152 layers without vanishing gradient issues.
- Resnet works by adding “identity mappings” on top of the CNN.
- ImageNet contains 11 million images and 11,000 categories. ImageNet is used to train ResNet deep networks.


x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu


RES-BLOCK
INPUT
CONVOLUTION BLOCK
IDENTITY BLOCK
IDENTITY BLOCK
OUTPUT


RESNET-18 MODEL
INPUT
ZERO PADDING
CONV2D
BATCHNORM,RELU
MAXPOOL2D
RES-BLOCK
RES-BLOCK
RES-BLOCK
RES-BLOCK
RES-BLOCK
AVERAGEPOOLING2D
FLATTEN()
DENSE LAYER,SOFTMAX
OUTPUT


INPUT
Main path
ConvD
MaxPool2D
BatchNorm, Relu
Conv2D-kernel(3*3)
BatchNorm, Relu
Conv2D
BatchNorm
Short path
Conv2D
MaxPool2D
BatchNorm
Relu


INPUT
Main path
Conv2D
BatchNorm, Relu
Conv2D-Kernel(3*3)
BatchNorm, Relu
Conv2D
BatchNorm
Short Path
Relu

Code Demonstration:

1. Import libraries/datasets

```
# Import the necessary packages
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
import os
import matplotlib.pyplot as plt
import PIL
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from tensorflow.keras.preprocessing.image
 import ImageDataGenerator
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.inception_resnet_v2
import InceptionResNetV2
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
from IPython.display import display
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.preprocessing.image
import ImageDataGenerator
from tensorflow.keras.models import Model, Sequential
from     tensorflow.keras.callbacks     import     ReduceLROnPlateau,
EarlyStopping, ModelCheckpoint, LearningRateScheduler
```

```
from jupyterthemes import jtplot
jtplot.style(theme='monokai',     context='notebook',     ticks=True,
grid=False)
# setting the style of the notebook to be monokai theme
# this line of code is important to ensure that we are able to see
the x and y axes clearly
```

```
os.listdir('./train')
```

```
os.listdir(os.path.join('train', 'Mild'))
```

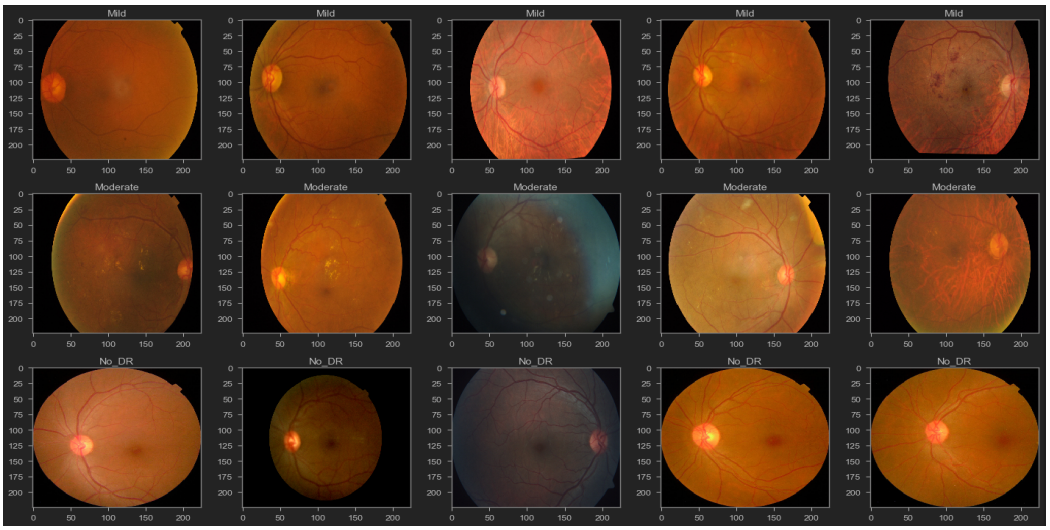
```
# Check the number of images in the dataset
train = []
label = []
# os.listdir returns the list of files in the folder, in this case
image class names
for i in os.listdir('./train'):
  train_class = os.listdir(os.path.join('train', i))
  for j in train_class:
    img = os.path.join('train', i, j)
    train.append(img)
    label.append(i)
print('Number of train images : {} \n'.format(len(train)))
```

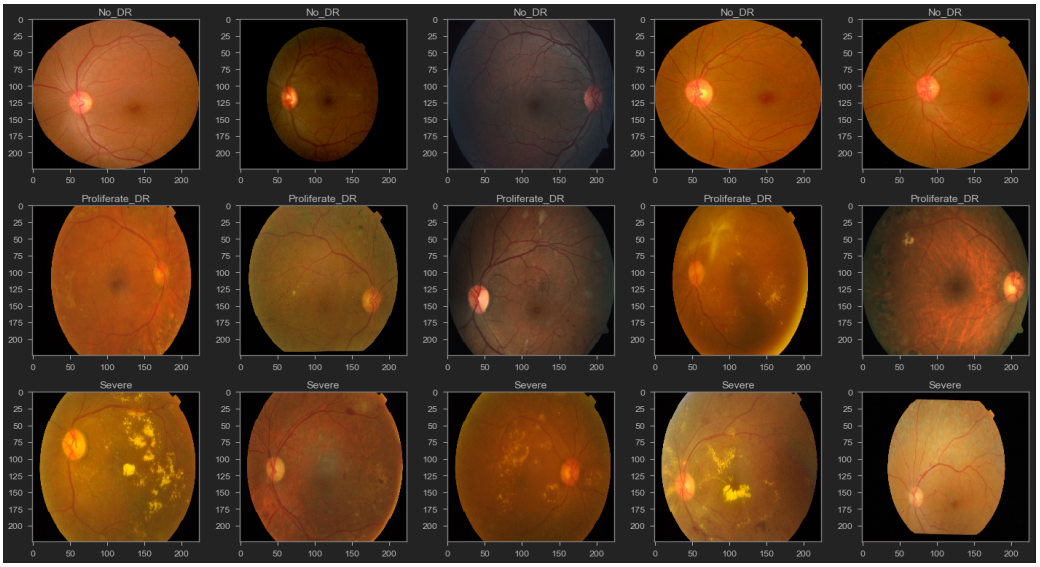
2. Data Exploration and Data Visualization :

```
# Visualize 5 images for each class in the dataset
fig, axs = plt.subplots(5, 5, figsize = (20, 20))
count = 0
for i in os.listdir('./train'):
  # get the list of images in a given class
  train_class = os.listdir(os.path.join('train', i))
  # plot 5 images per class
  for j in range(5):
    img = os.path.join('train', i, train_class[j])
    img = PIL.Image.open(img
    axs[count][j].title.set_text(i)
    axs[count][j].imshow(img)
  count += 1
fig.tight_layout()
#check the number of images in each class in the training dataset
No_images_per_class = []
Class_name = []
for i in os.listdir('./train'):
  train_class = os.listdir(os.path.join('train', i))
  No_images_per_class.append(len(train_class))
  Class_name.append(i)
 print('Number of images in {} = {} \n'.format(i, len(train_class)))
```

```
Number of images in Mild = 370
Number of images in Moderate = 999
Number of images in No_DR = 1805
Number of images in Proliferate_DR = 295
Number of images in Severe = 193
```

```
retina_df = pd.DataFrame({'Image': train,'Labels': label})

retina_df
```





3. Data Augmentation is performed and data generator is created

```
# Shuffle the data and split it into training and testing
retina_df = shuffle(retina_df)
train, test = train_test_split(retina_df, test_size = 0.2)
```

```
# Create run-time augmentation on training and test dataset
# For training datagenerator, we add normalization, shear angle, zooming range
and horizontal flip
train_datagen = ImageDataGenerator(
```

```
        rescale = 1./255,
        shear_range = 0.2,
        validation_split = 0.15)

# For test datagenerator, we only normalize the data.
test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
# Creating datagenerator for training, validation and test dataset.

train_generator = train_datagen.flow_from_dataframe(
    train,
    directory='./',
    x_col="Image",
    y_col="Labels",
    target_size=(256, 256),
    color_mode="rgb",
    class_mode="categorical",
    batch_size=32,
    subset='training')

validation_generator = train_datagen.flow_from_dataframe(
    train,
    directory='./',
    x_col="Image",
    y_col="Labels",
    target_size=(256, 256),
    color_mode="rgb",
    class_mode="categorical",
    batch_size=32,
    subset='validation')

test_generator = test_datagen.flow_from_dataframe(
    test,
    directory='./',
    x_col="Image",

      y_col="Labels",
    target_size=(256, 256),
    color_mode="rgb",
    class_mode="categorical",
    batch_size=32)
```

```
Found 2490 validated image filenames belonging to 5 classes.
Found 439 validated image filenames belonging to 5 classes.
Found 733 validated image filenames belonging to 5 classes.
```

4. Build RES - BLOCK Based Deep Learning Model

```
def res_block(X, filter, stage):

  # Convolutional_block
  X_copy = X
  f1 , f2, f3 = filter
  # Main Path

  X = Conv2D(f1, (1,1),strides = (1,1), name
='res_'+str(stage)+'_conv_a', kernel_initializer= glorot_uniform(seed =
0))(X)X = MaxPool2D((2,2))(X)

  X = BatchNormalization(axis =3, name = 'bn_'+str(stage)+'_conv_a')(X)

  X = Activation('relu')(X)

  X = Conv2D(f2, kernel_size = (3,3), strides =(1,1), padding = 'same',
name ='res_'+str(stage)+'_conv_b', kernel_initializer=
glorot_uniform(seed = 0))(X)

  X = BatchNormalization(axis =3, name = 'bn_'+str(stage)+'_conv_b')(X)

  X = Activation('relu')(X)

  X = Conv2D(f3, kernel_size = (1,1), strides =(1,1),name
='res_'+str(stage)+'_conv_c', kernel_initializer= glorot_uniform(seed =
0))(X)

  X = BatchNormalization(axis =3, name = 'bn_'+str(stage)+'_conv_c')(X)

# Short path

  X_copy = Conv2D(f3, kernel_size = (1,1), strides =(1,1),name
='res_'+str(stage)+'_conv_copy', kernel_initializer= glorot_uniform(seed
= 0))(X_copy)

  X_copy = MaxPool2D((2,2))(X_copy)

  X_copy = BatchNormalization(axis =3, name =
'bn_'+str(stage)+'_conv_copy')(X_copy)
```

```
# Short path

  X_copy = Conv2D(f3, kernel_size = (1,1), strides =(1,1),name
='res_'+str(stage)+'_conv_copy', kernel_initializer= glorot_uniform(seed
= 0))(X_copy)

  X_copy = MaxPool2D((2,2))(X_copy)

  X_copy = BatchNormalization(axis =3, name =
'bn_'+str(stage)+'_conv_copy')(X_copy)

  X = Add()([X,X_copy])

  X = Activation('relu')(X)

  # Identity Block 1

  X_copy = X

  # Main Path

  X = Conv2D(f1, (1,1),strides = (1,1), name
='res_'+str(stage)+'_identity_1_a', kernel_initializer=
glorot_uniform(seed = 0))(X)

X = BatchNormalization(axis =3,name= 'bn_'+str(stage)+'_identity_1_a')(X)

X = Activation('relu')(X)

  X = Conv2D(f2, kernel_size = (3,3), strides =(1,1), padding = 'same',
name ='res_'+str(stage)+'_identity_1_b', kernel_initializer=
glorot_uniform(seed = 0))(X)

  X = BatchNormalization(axis =3, name =
'bn_'+str(stage)+'_identity_1_b')(X)

  X = Activation('relu')(X)

  X = Conv2D(f3, kernel_size = (1,1), strides =(1,1),name
='res_'+str(stage)+'_identity_1_c', kernel_initializer=
glorot_uniform(seed = 0))(X)

  X = BatchNormalization(axis =3, name =
'bn_'+str(stage)+'_identity_1_c')(X)

  # ADD

  X = Add()([X,X_copy])
```

```
  X = Activation('relu')(X)

  # Identity Block 2

  X_copy = X

  # Main Path

  X = Conv2D(f1, (1,1),strides = (1,1), name
='res_'+str(stage)+'_identity_2_a', kernel_initializer=
glorot_uniform(seed = 0))(X)

  X = BatchNormalization(axis =3, name =
'bn_'+str(stage)+'_identity_2_a')(X)

  X = Activation('relu')(X)

  X = Conv2D(f2, kernel_size = (3,3), strides =(1,1), padding = 'same',
name ='res_'+str(stage)+'_identity_2_b', kernel_initializer=
glorot_uniform(seed = 0))(X)

  X = BatchNormalization(axis =3, name =
'bn_'+str(stage)+'_identity_2_b')(X)

  X = Activation('relu')(X)

  X = Conv2D(f3, kernel_size = (1,1), strides =(1,1),name

='res_'+str(stage)+'_identity_2_c', kernel_initializer=

glorot_uniform(seed = 0))(X)


  X = BatchNormalization(axis =3, name =
'bn_'+str(stage)+'_identity_2_c')(X)

  # ADD

  X = Add()([X,X_copy])

  X = Activation('relu')(X)

  return X
input_shape = (256,256,3)
#Input tensor shape
X_input = Input(input_shape)
#Zero-padding
X = ZeroPadding2D((3,3))(X_input)
```

```
# 1 - stage
X = Conv2D(64, (7,7), strides= (2,2), name = 'conv1', kernel_initializer=
glorot_uniform(seed = 0))(X)
X = BatchNormalization(axis =3, name = 'bn_conv1')(X)
X = Activation('relu')(X)
X = MaxPooling2D((3,3), strides= (2,2))(X)
# 2- stage
X = res_block(X, filter= [64,64,256], stage= 2)
# 3- stage
X = res_block(X, filter= [128,128,512], stage= 3)
# 4- stage
X = res_block(X, filter= [256,256,1024], stage= 4)
# # 5- stage
# X = res_block(X, filter= [512,512,2048], stage= 5)
#Average Pooling
X = AveragePooling2D((2,2), name = 'Averagea_Pooling')(X)
#Final layer
X = Flatten()(X)
X = Dense(5, activation = 'softmax', name = 'Dense_final',
kernel_initializer= glorot_uniform(seed=0))(X)
model = Model( inputs= X_input, outputs = X, name = 'Resnet18')
model.summary()
```

```
Output exceeds the size limit. Open the full output data in a text
editor
Model: "Resnet18"
_____________________________________________________________________
Layer (type)                 Output Shape              Param #
Connected to

=====================================================================
input_1 (InputLayer)         [(None, 256, 256, 3) 0

_____________________________________________________________________
zero_padding2d (ZeroPadding2D)      (None,  262,  262,  3)      0
input_1[0][0]

_____________________________________________________________________
conv1 (Conv2D)                      (None, 128, 128, 64) 9472
zero_padding2d[0][0]

_____________________________________________________________________


bn_conv1 (BatchNormalization)       (None,  128,  128,  64)  256
conv1[0][0]

_____________________________________________________________________
activation (Activation)                (None,  128,  128,  64)  0
```

```
bn_conv1[0][0]

__________________________________________________________________________________________
max_pooling2d   (MaxPooling2D)          (None,   63,   63,   64)       0
activation[0][0]

__________________________________________________________________________________________
res_2_conv_a  (Conv2D)                   (None,  63,  63,  64)     4160
max_pooling2d[0][0]

__________________________________________________________________________________________
max_pooling2d_1   (MaxPooling2D)      (None,   31,   31,   64)       0
res_2_conv_a[0][0]

__________________________________________________________________________________________
bn_2_conv_a   (BatchNormalization   (None,   31,   31,   64)       256
max_pooling2d_1[0][0]

__________________________________________________________________________________________
activation_1   (Activation)              (None,  31,  31,  64)       0
bn_2_conv_a[0][0]

__________________________________________________________________________________________
res_2_conv_b  (Conv2D)                   (None,  31,  31,  64)     36928
activation_1[0][0]
...
Total params: 4,987,525
Trainable params: 4,967,685
Non-trainable params: 19,840

__________________________________________________________________________________________
```

5. Compile and Train Deep Learning Model

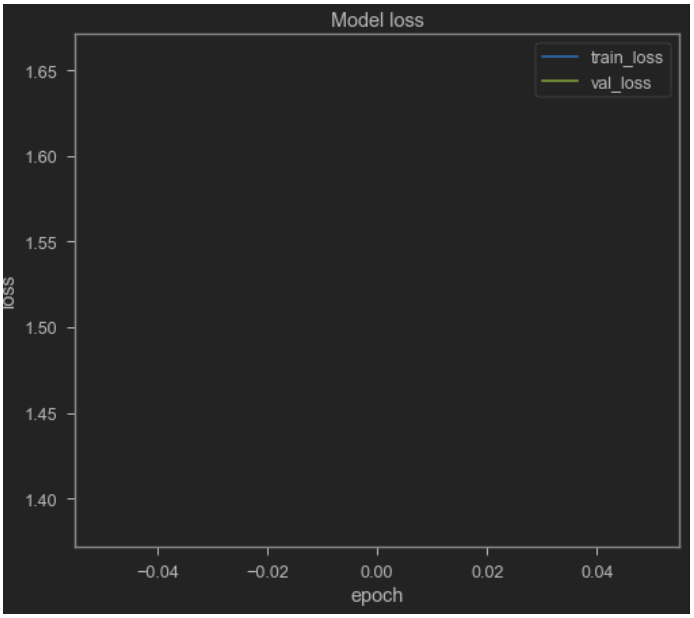
```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy',
metrics= ['accuracy'])
```

```
#using early stopping to exit training if validation loss is not
decreasing even after certain epochs (patience)
earlystopping = EarlyStopping(monitor='val_loss', mode='min', verbose=1,
patience=15)
#save the best model with lower validation loss
checkpointer = ModelCheckpoint(filepath="weights.hdf5", verbose=1,
save_best_only=True)
```

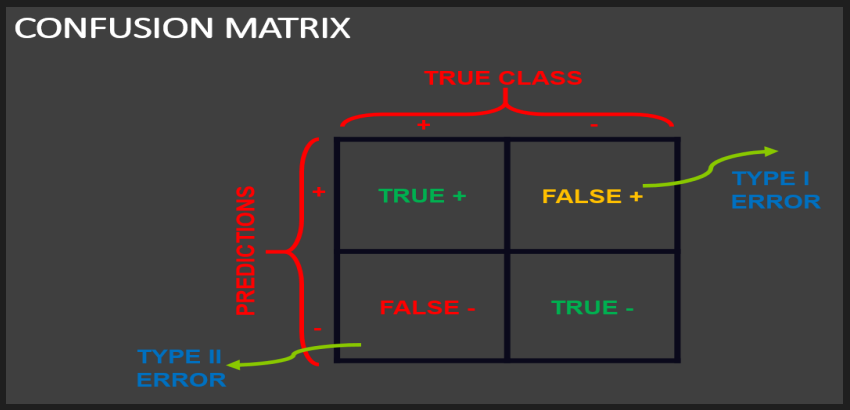
```
history = model.fit(train_generator, steps_per_epoch = train_generator.n
// 32, epochs = 1, validation_data= validation_generator,
validation_steps= validation_generator.n // 32, callbacks=[checkpointer ,
earlystopping])
```

```
Train for 77 steps, validate for 13 steps
76/77 [============================>.] - ETA: 13s - loss: 1.3915 -
accuracy: 0.6575
Epoch 00001: val_loss improved from inf to 1.65759, saving model to
weights.hdf5
77/77 [==============================] - 1080s 14s/step - loss:
1.3841 - accuracy: 0.6591 - val_loss: 1.6576 - val_accuracy: 0.2740
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train_loss','val_loss'], loc = 'upper right')
plt.show()
```



6. Assess the Performance of The Trained Model




```
model.load_weights("retina_weights.hdf5")

# Evaluate the performance of the model

evaluate = model.evaluate(test_generator, steps = test_generator.n // 32,
verbose =1)

print('Accuracy Test : {}'.format(evaluate[1]))
```

```
22/22 [==============================] - 51s 2s/step - loss: 0.4700
- accuracy: 0.8310

Accuracy Test : 0.8309659361839294
```

```
# Assigning label names to the corresponding indexes

labels = {0: 'Mild', 1: 'Moderate', 2: 'No_DR', 3:'Proliferate_DR', 4:
'Severe'}

# Loading images and their predictions

from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score

# import cv2

prediction = []

original = []

image = []

count = 0
```

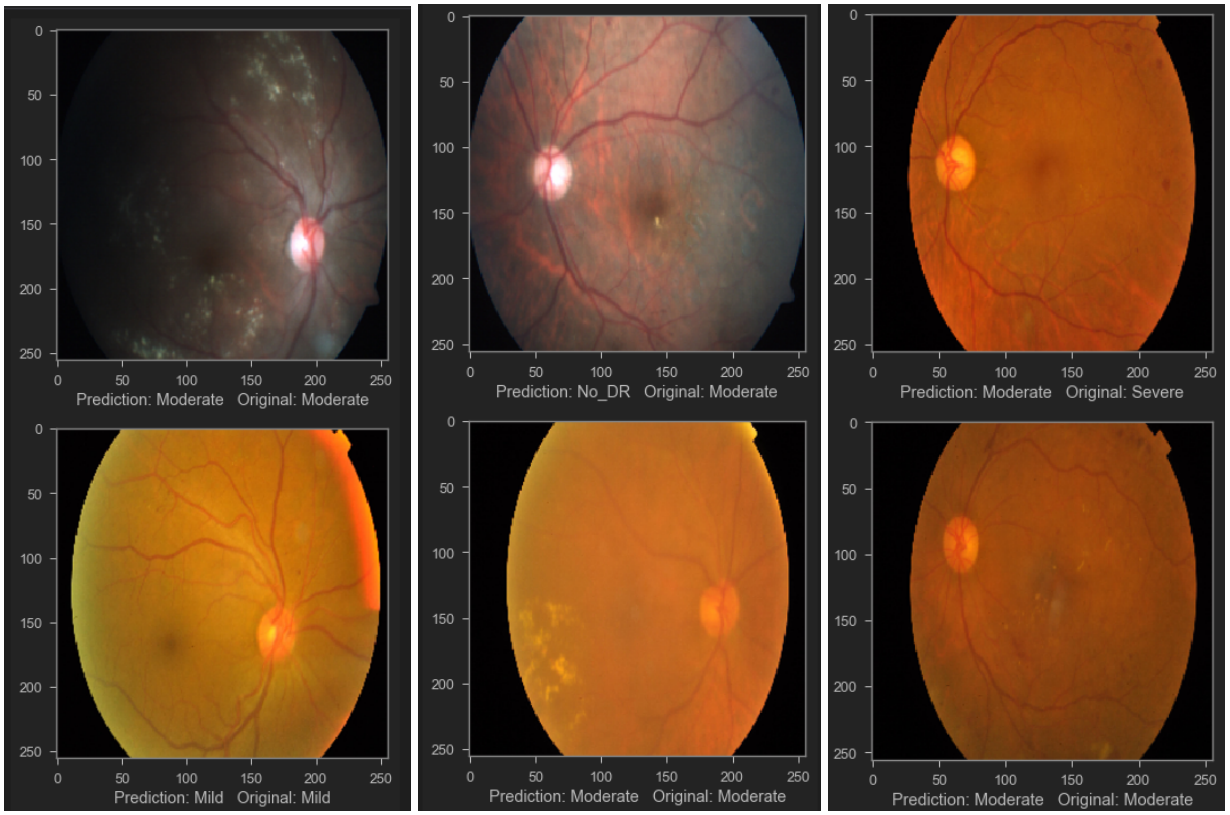
```
for item in range(len(test)):
  # code to open the image
  img= PIL.Image.open(test['Image'].tolist()[item])
  # resizing the image to (256,256)
  img = img.resize((256,256))
  # appending image to the image list
  image.append(img)
  # converting image to array
  img = np.asarray(img, dtype= np.float32)
  # normalizing the image
  img = img / 255
  # reshaping the image in to a 4D array
  img = img.reshape(-1,256,256,3)
  # making prediction of the model
  predict = model.predict(img)
  # getting the index corresponding to the highest value in the
prediction
  predict = np.argmax(predict)
  # appending the predicted class to the list
  prediction.append(labels[predict])
  # appending original class to the list
  original.append(test['Labels'].tolist()[item])
```

```
# Getting the test accuracy
score = accuracy_score(original, prediction)
print("Test Accuracy : {}".format(score))
```

```
Test Accuracy : 0.8294679399727148
```

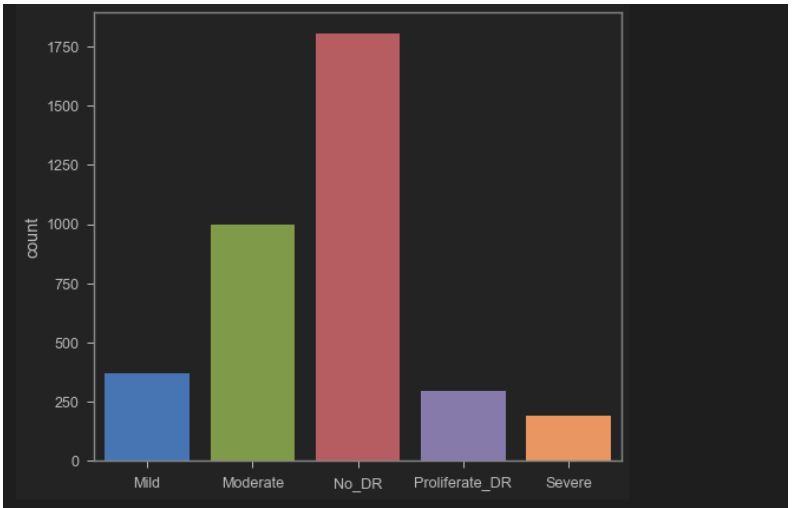
```
# Visualizing the results
import random
```

```
fig=plt.figure(figsize = (100,100))

for i in range(20):

    j = random.randint(0,len(image))

    fig.add_subplot(20, 1, i+1)

    plt.xlabel("Prediction: " + prediction[j] +"   Original: " +
original[j])

    plt.imshow(image[j])

fig.tight_layout()

plt.show()
```



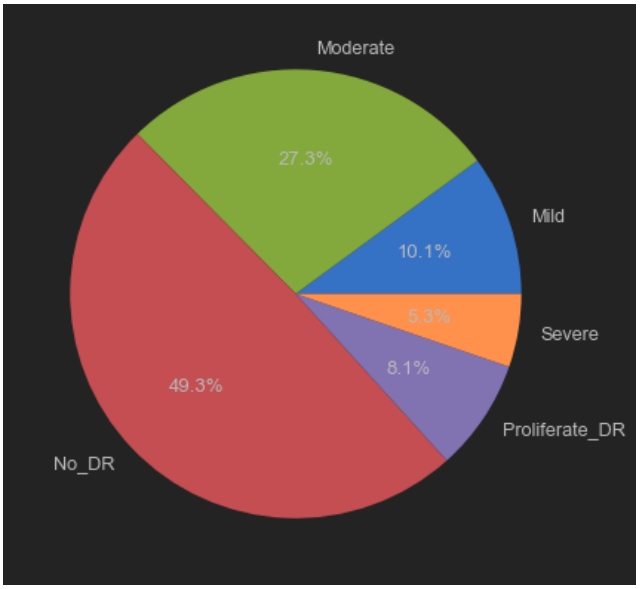
```
sns.countplot(label)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f10cd75ac8>
```

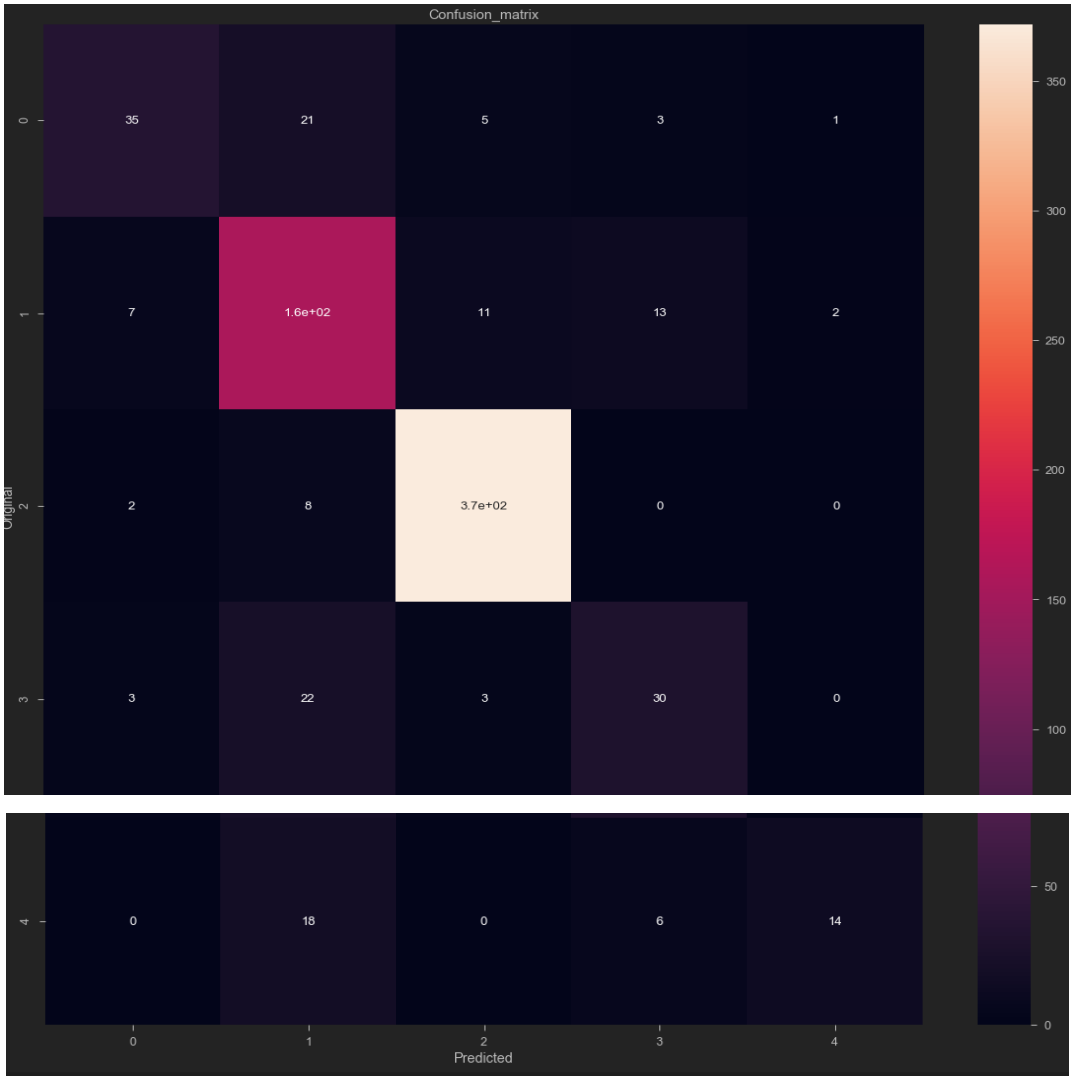


#plot a pie chart showing the percentage of samples per class.

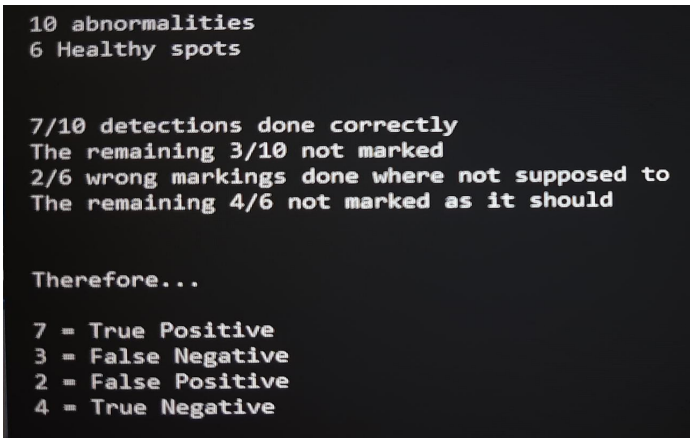
```
No_images_per_class

Class_name

fig1, ax1 = plt.subplots()

ax1.pie(No_images_per_class, labels = Class_name, autopct = '%1.1f%%')

plt.show
```



	precision	recall	f1-score	support
Mild	0.74	0.54	0.62	65
Moderate	0.69	0.83	0.75	190
No_DR	0.95	0.97	0.96	382
Proliferate_DR	0.58	0.52	0.55	58
Severe	0.82	0.37	0.51	38
accuracy			0.83	733
macro avg	0.76	0.64	0.68	733
weighted avg	0.83	0.83	0.82	733



Evaluation Metrics : Performance of the model is evaluated on the basis of these metrics.True Positive is the correct positive prediction by the model.True Negative is the correct negative prediction by the model.False Positive is the wrong prediction of the positive by the model.False Negative is the wrong prediction of the negative by the model.



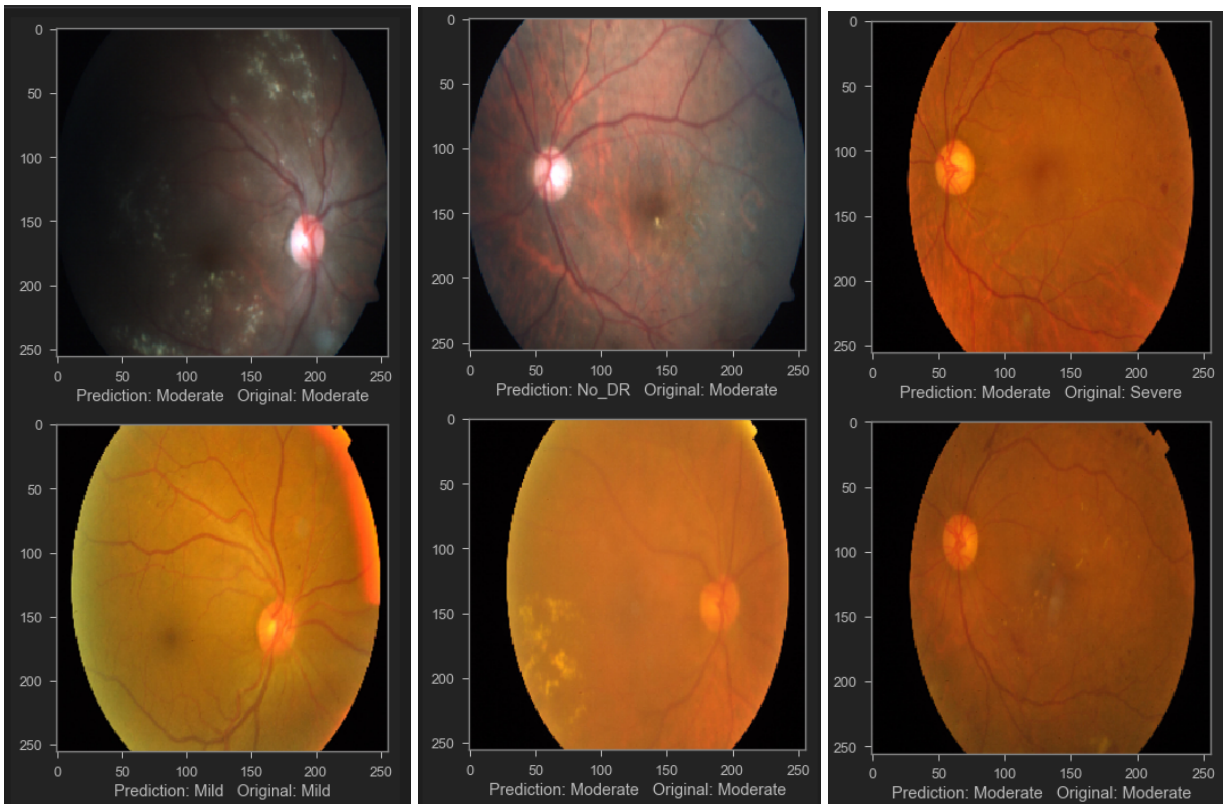
1. **Classification Accuracy :** (True Positive + True Negative)/(True Positive + True Negative + False Positive + False Negative) = 97.9 %
2. **Sensitivity :** (True Positive)/(True Positive + False Negative) = 0.7
3. **Specificity :** (True Negative)/(True Negative + False Positive) = 0.66
4. **Precision :** (True Positive)/(True Positive + False Positive) = 0.77
5. **F1-Score :** (2 * True Positive)/(2 * True Positive + False Positive + False Negative) = 0.73

Chapter 6 : Software and Hardware Requirement Specifications

Software Requirement Specifications :

- Jupyter Notebook
- VS Code
- Python

Chapter 7 : Results and Output



Chapter 8 : Conclusion

Among other existing supervising algorithms, most of them are requiring more pre-processing or post-processing stages for identifying the different stages of the diabetic retinopathy. Also, other algorithms mandatorily require manual feature extraction stages to classify the fundus images. In our proposed solution, Deep convolutional Neural Network is a wholesome approach to all levels of diabetic retinopathy stages. No manual feature extraction stages are needed. Our network architecture with dropout techniques yielded significant classification accuracy. True positive rates (or recall) are also improved. This architecture has some setbacks: An additional stage augmentation is needed for the images taken from different cameras with different fields of view. Also, our network architecture is complex and computation-intensive requiring a high-level graphics processing unit to process the high resolution images when the level of layers stacked more.

Chapter 9 : Future Works

- ➢ Validate the different abnormalities according the different types of Retinopathy
- ➢ Find the percentages of the different types of abnormalities present in a particular image
- ➢ In future we want to make an web pages which will detect “DIABETIC RETINOPATHY OF EYE”

Chapter 10 : Appendices

List of Technical Terms :

- ★ convolutional neural network
- ★ deep learning
- ★ diabetic retinopathy
- ★ machine learning
- ★ retinal fundus image

Abbreviations used:

- ★ **DR:** Diabetic Retinopathy
- ★ **NPDR:** Non-proliferative Diabetic Retinopathy
- ★ **PDR:** Proliferative Diabetic Retinopathy
- ★ **DNN:** Deep Neural Network
- ★ **SVM:** Support Vector Machine
- ★ **CNN:** Convolution Neural Network
- ★ **DL:** Deep Learning
- ★ **RGB:** Red-Green-Blue

Chapter 11 : Bibliography

1. International Diabetes Federation. *IDF Diabetes Atlas* 7th edn (International Diabetes Federation, Brussels, Belgium, 2015).
2. American Diabetes Association. 11. Microvascular complications and foot care: standards of medical care in diabetes-2020. *Diabetes Care* 43, S135–S151 (2020).
3. Wang, L. Z. et al. Availability and variability in guidelines on diabetic retinopathy screening in Asian countries. *Br. J. Ophthalmol.* 101, 1352–1360 (2017)
4. https://github.com/FDU-VTS/Awesome-Diabetic-Retinopathy-Detection
5. https://github.com/nkicsl/Fundus_Review