DIABETIC RETINOPATHY DETECTION WITH ARTIFICIAL INTELLIGENCE

[1]Debraj Banerjee

[1] B.Tech Final Year (Student),
[1]Dept of Electronics and Telecommunication Engineering,
[1]KIIT University, Bhubaneswar, India)

Abstract : Diabetic retinopathy is a state of affairs that occurs as a result of vandalizing the blood vessels of the retina in people who have diabetes . Diabetic retinopathy can prosper if we have type 1 or 2 diabetes and a longhorn of uncontrolled high blood sugar levels . While we may start out with only mild vision problems, we can in the fullness of time lose our sight . Untreated diabetic retinopathy is one of the stereotypical causes of blindness in the United States, according to the National Eye Institute. It’s also the stereotypical eye disease in people with diabetes. In this paper, we will train a deep neural network model based on CNNs and Residual Blocks to detect the type of Diabetic Retinopathy from images. DR is a disease that results from the complication of type 1 and 2 diabetes . DR is the leading cause of blindness in the working-age population of the developed world and is estimated to affect over 2.6 million people in the world and is responsible for 2.6% of global blindness (0.84 million of 32.4 million people) worldwide according to the WHO. This paper design in such way that learner can understand the concept and implement in their real life.

***IndexTerms* - Convolutional Neural Networks , Diabetic Retinopathy , World Health Organization.**

I. INTRODUCTION

Humanoid with diabetes, whatever the type of diabetes, requires regular and repetitive annual retinal screening for early detection and auspicious treatment of diabetic retinopathy , awfully sight-threatening diabetic retinopathy . Screening for retinopathy is precisely done through fundus examination by optometric physician or retinal color photography using conventional mydriatic or nonmydriatic fundus cameras by trained eye artisans . Smartphone-based retinal imaging has popped up as one of the recent economical ways of putting out retinopathy in the community . However, till date disregarding the type of fundus camera cast-off , the retinal images had to be graded for the occupancy and severity of DR by retinal specialists . And most undeveloped states in India , facing problems in screening the DR . So AI is simulation of human intelligence by a software/machine. It is an esoteric field which is based on teaching the machine to apprehend specific patterns. It has been used for poles apart kinds of technical tasks including unambiguous classification of high-resolution images. AI for identification and categorization of DR happens by providing thousands of retinal images of diversifying grades of DR to the system for learning .

II. METHODOLOGY

Method of DR detection with AI is enacted in the podiums called jupyter notebook . Because on jupyter notebook , we do simulation and analysis in a hands-on manner in the browser. It will give podiums access to pre-configured cloud desktops that have all the software and data we will need. So, we can just focus on the learning and implementation the concept for this paper . This means instant access to a cloud desktop with Python and TensorFlow pre-installed.The hole methodology is categorized in six steps

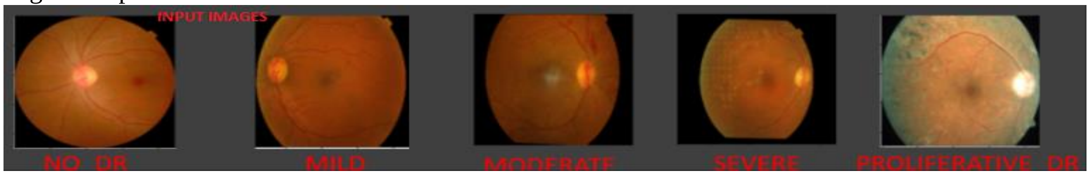
1. Recognize the theory and intuition behind Deep Neural Networks, Residual Nets, and Convolutional Neural Networks (CNNs). 2. Appeal Python libraries to import, pre-process and envisage images. 3. Effectuate data augmentation to improve model generalization capability. 4. Assemble a deep learning model based on Convolutional Neural Network and Residual blocks using Keras with Tensorflow 2.0 as a backend. 5. Compile and fit Deep Learning model to training data. 6. Assess the staging of trained CNN and ensure its rationalization using various KPIs such as accuracy, precision and recall.

III. MODELING AND ANALYSIS

While doing the analysis and modeling in jupyter notebook , I divided the steps in 5 ways for better understanding and implementation in the real world for DR.

1. UNDERSTAND THE PROBLEM STATEMENT

Diabetic retinopathy is the leading cause of blindness in the working age population of the developed world . The World Health Organization estimated that more than 347 million people have the disease worldwide.With the proper use of AI and Deep Learning , doctors will be able to detect blindness before it occurs . Retinopathy among people mostly living in rural areas where medical Screening and experienced Doctors are limited .



. Fig1 = Input images .

For analysis we need the strong data set for input and output . The data set consists of 3662 colour images belonging to 5 categories. Categories that present in the data are No_DR , Mild , Moderate , Severe and proliferative (rapidly growing).

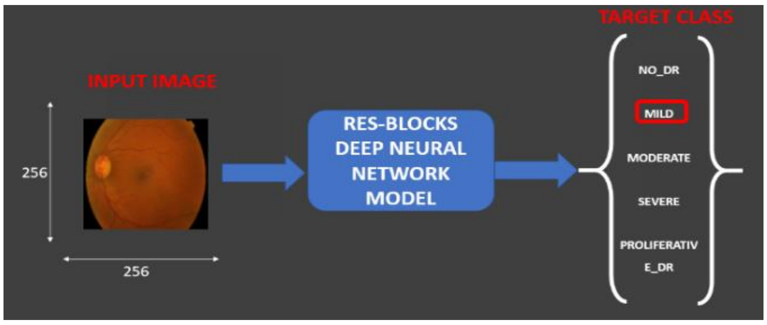
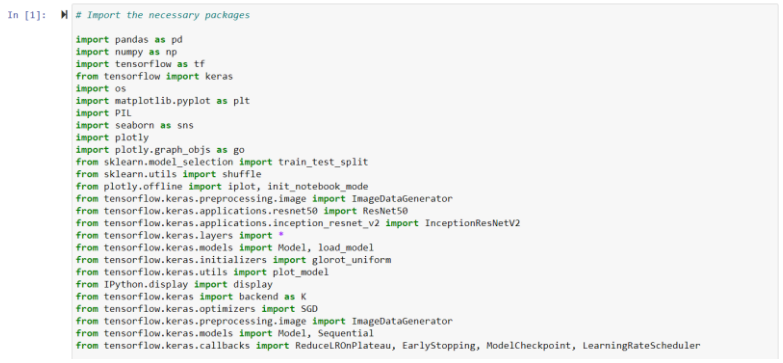
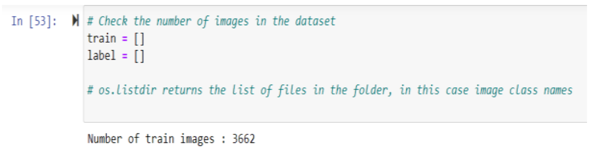


Fig2 = Block Diagram of Input Image to Target Class .

2 . IMPORT LIBRARIES

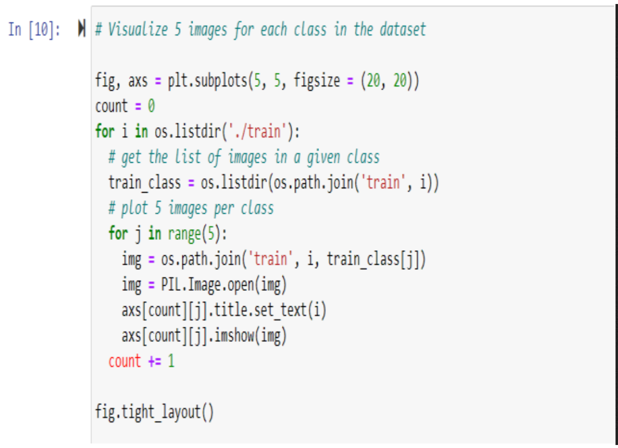
Importing the libraries required for DR analysis . Here we will import pandas , Numpy , tensorflow , os , matplotlib , PIL , Seaborn , Plotly , Sklearn and Keras , each library has its own different functions. After that we will analyze the class (No_DR , Mild , Moderate , Severe and proliferative) and Check the number of images in the dataset. And Print out the count plot for all classes using Seaborn using command sns.countplot(label) .


```
In [1]:  # 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
import plotly
import plotly.graph_objs as go
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from plotly.offline import iplot, init_notebook_mode
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
```


```
In [53]:  # 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

          Number of train images : 3662
```
Fig3 = Importing libraries & No. Of Images train .

3.PERFORM DATA EXPLORATION AND DATA VISUALIZATION

Data exploration is the initial step in data analysis, where users explore a large data set in an unstructured way to uncover initial patterns, characteristics, and points of interest. This process isn't meant to reveal every bit of information a data set holds, but rather to help create a broad picture of important trends and major points to study in greater detail.Data exploration can use a combination of manual methods and automated tools such as data visualizations, charts, and initial reports[1] .while running the data exploration and data visualization applying the above code , the output in below fig .


```
In [10]:  # 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()
```

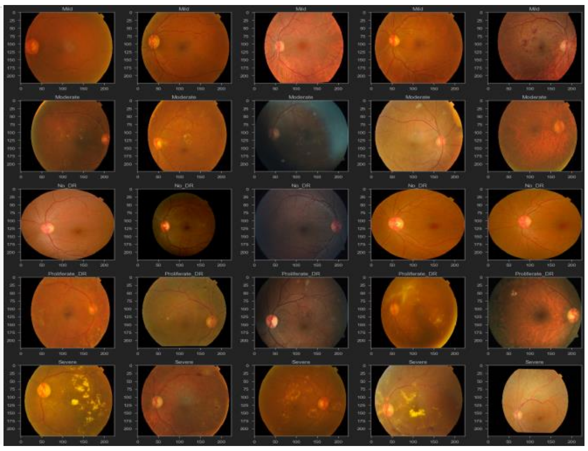
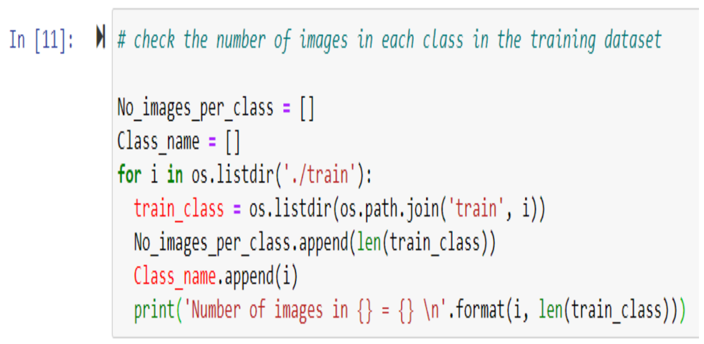


Fig4 = Data visualization code & visualization pic .

Simultaneously check the number of images in each class in the training dataset because a quotidian and distinctly effective approach to deep learning on small image datasets is to use a pretrained network. A pretrained network is an extricated network that was formerly trained on a prodigious dataset, generally on a large-scale image-classification task[2]. If this original dataset is large abundant and general enough, then the spatial hierarchy of features assimilated by the pretrained network can assimilated act as a generic model of the visual world, and hence its features can ratify useful for many different computer vision problems, even though these new problems may involve utterly different classes than those of the original task. For instance, you might train a network on ImageNet and then remodel this trained network for existence as remote as identifying furniture items in images.

Data exploration uses both blue-collar data analysis (repeatedly considered one of the determiner tedious and time gobbling tasks in data science) and motorized tools that back out data into inceptive reports that include data visualizations and charts. This technique enables deeper data analysis as motifs and trends are recognized. Data exploration helps hatch a more plain sailing view of datasets rather than rushing over thousands of figures in unregulated data .



```
In [11]:  # 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)))
```

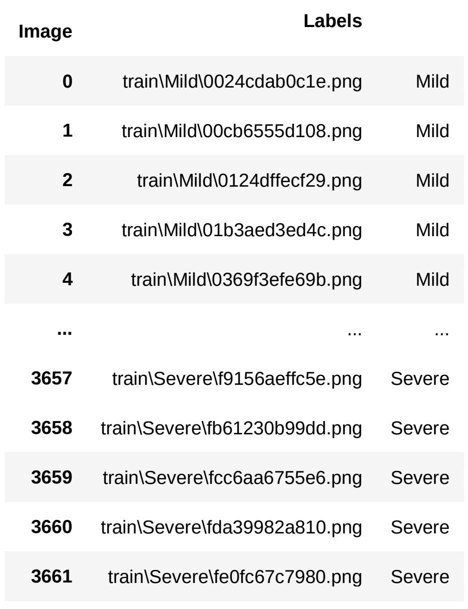
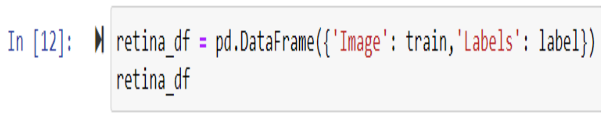


	Image	Labels
0	train\Mild\0024cdab0c1e.png	Mild
1	train\Mild\00cb6555d108.png	Mild
2	train\Mild\0124dffecf29.png	Mild
3	train\Mild\01b3aed3ed4c.png	Mild
4	train\Mild\0369f3efe69b.png	Mild
...	...	...
3657	train\Severe\f9156aeffc5e.png	Severe
3658	train\Severe\fb61230b99dd.png	Severe
3659	train\Severe\fcc6aa6755e6.png	Severe
3660	train\Severe\fda39982a810.png	Severe
3661	train\Severe\fe0fc67c7980.png	Severe



```
In [12]:  retina_df = pd.DataFrame({'Image': train,'Labels': label})
          retina_df
```

.

Fig5 = Training dataset and calling label data

4.UNDERSTAND THE THEORY AND INTUITION BEHIND CONVOLUTIONAL NEURAL NETWORKS (CNN) AND RESIDUAL BLOCKS

Deep learning, is an approach of AI. Specifically, it is a type of machine learning, a technique that allows computer systems to improve with experience and data. It achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts. It has been successfully used in commercial applications since the 1990s. Deep learning algorithms have completely changed our perception of information processing through different applications such as; NLP, computer vision, speech and audio, social network analysis, and healthcare, this includes application area the computer-aided diagnosis [3]. Some key enabler deep learning algorithms such as generative adversarial networks, convolutional neural networks, and model transfers [4].

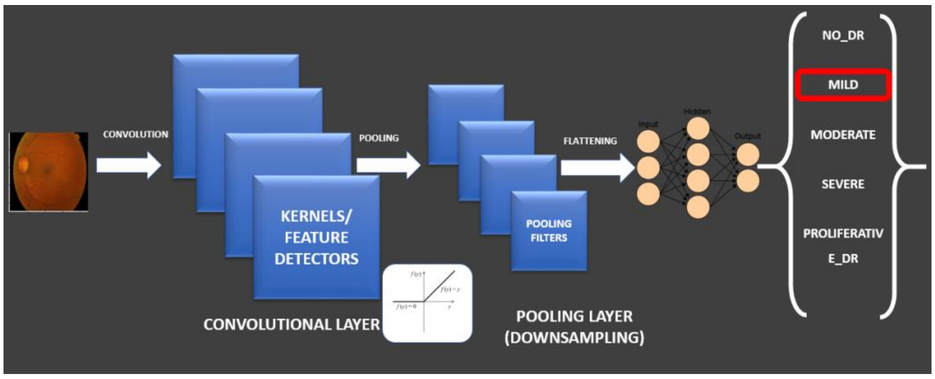


Fig6 = Convolutional Layer

Convolutional Neural Network (CNN model) is influenced by the human visual system [5]. CNN architecture consists of multi-layer. Low-level features and attributes can have extracted from earlier layer for example edge detection. And the higher layer or final fully connected layer gives the object features of the entire image [6] [7] [8]. Convolutional Layer is central of CNN [9] [10]. It takes a filter(kernel) and pass it through all the points in the image (Input) and passing at any point into a single position (Output). Pooling layer down-sampling of an image, It takes sub-samples of convolutional layer output and produces a single output. Then it uses different pooling techniques such as max pooling, mean pooling, average pooling etc. Later, fully connected layers take input from all neurons in the previous layer and perform operation with an individual neuron in the current layer to generate output.

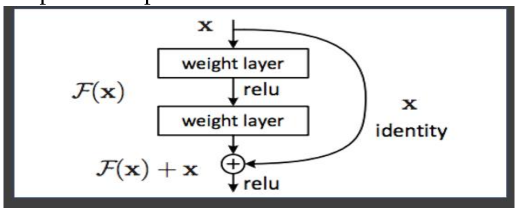
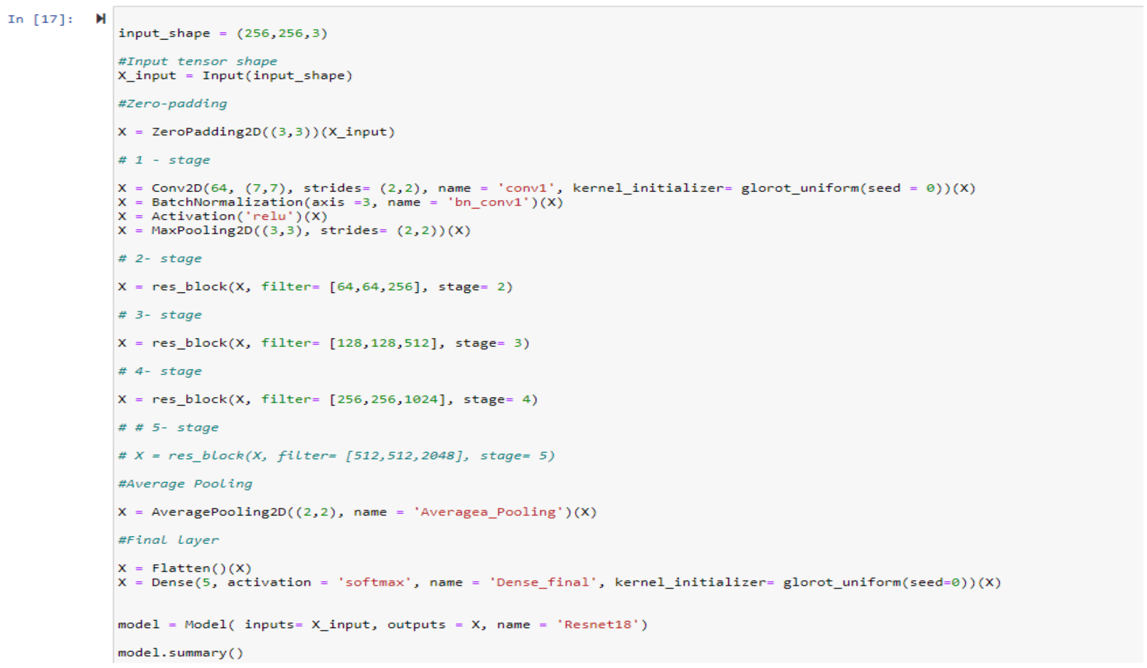

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig7 = Weight Layer



```
In [17]:
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()
```

Fig8 = Build Res-Block Based Deep Learning Model Code

```
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]

_______________________________________________________________________________________

________________________________________________________________________________________
____________
flatten (Flatten)               (None, 9216)         0          Averagea_Pooling[0][0]

________________________________________________________________________________________
____________
Dense_final (Dense)             (None, 5)            46085      flatten[0][0]

========================================================================================
============
Total params: 4,987,525
Trainable params: 4,967,685
Non-trainable params: 19,840
```

Fig9 = Model Data

The proposed model of automatic Diabetic Retinopathy's detection, are dividing into three phases. Starting from preparing the data, then extract the features of the entire images based on CNN, finally classification phase to recognize healthy and unhealthy retina image. Shows the block diagram of the proposed model schema .

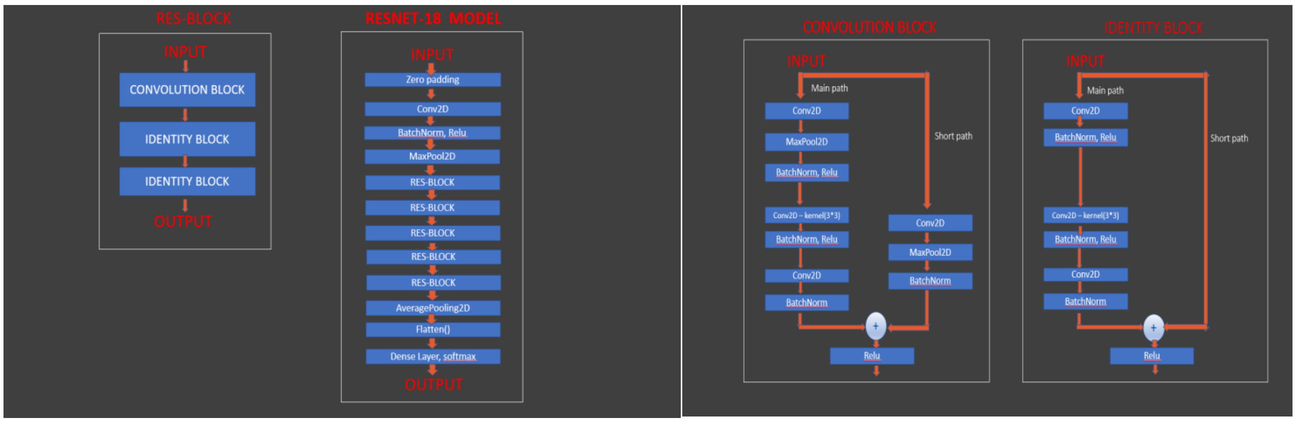
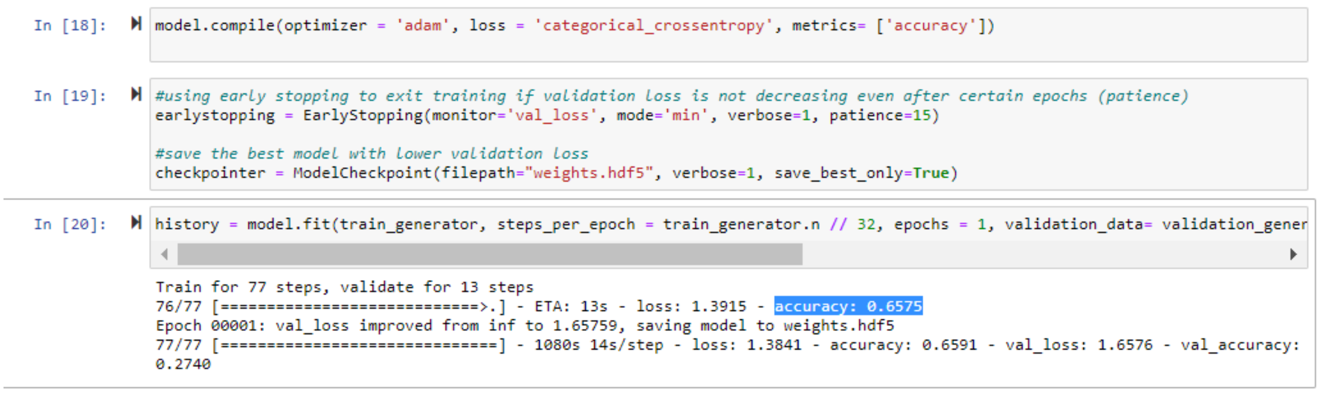


Fig10 =Model Schema

5.COMPILE AND TRAIN DEEP LEARNING MODEL

The proposed CNN architecture consists of three phases of a arrangement of convolution layer, homogenize layer with epsilon 0.00001, then enlivening function Rectified Linear Unit (ReLU) and max-pooling layer. The features maps are intended within convolution layer are normalized then based on ReLU for transmogrify the sum of weighted input into the activation of the node of the output of the current phase, this kind of activation function are easy to train and achieve a good performance. In the pooling layer, the topmost or average of the neighboring values to the feature maps are calculated to speed up the approaches and condense the size.

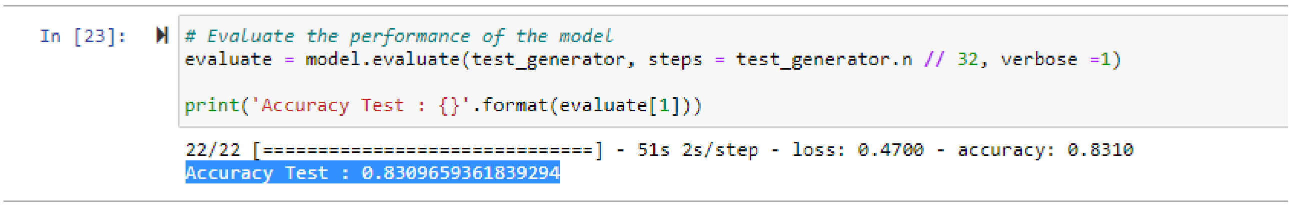


```
In [18]:  model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics= ['accuracy'])

In [19]:  #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)

In [20]:  history = model.fit(train_generator, steps_per_epoch = train_generator.n // 32, epochs = 1, validation_data= validation_gener

          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
```

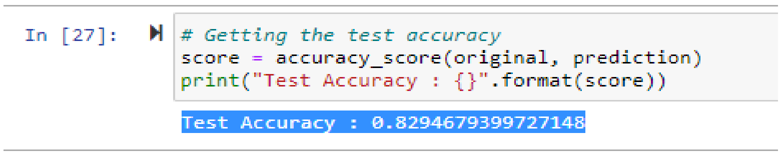
Fig11 = epochs accuracy .

IV. RESULTS AND DISCUSSION

The proposed model is tested on three different datasets, 80% of the retina images are selected for training and the 20% is for testing, a random selection of training images set for better evaluation. The experiments were performed on a computer with jupyter notebook a in Windows 10 64 bit operating system environment. The training options of the proposed CNN were using sgdm and learning rate schedule piecewise, with drop factor 0.2 to reduce the learning rate by a factor every 5 epochs, Set the maximum number of epochs for training to 20, shuffle each epoch and getting Evaluate the performance of the model accuracy 83% . And getting the test accuracy 82.9% .Attaching the fig in below .



```
In [23]:  # 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
```



```
In [27]:  # Getting the test accuracy
          score = accuracy_score(original, prediction)
          print("Test Accuracy : {}".format(score))

          Test Accuracy : 0.8294679399727148
```

Fig12 = Model accuracy & Test accuracy

V.ACKNOWLEDGMENT

I feel immense pleasure and feel privileged in expressing my deepest and most sincere gratitude to jupyter notebook & Convolutional Neural Network and Residual blocks using Keras with Tensorflow 2.0 as a backend , for its excellent guidance throughout my paper work .

REFERENCES

[1] https://www.sisense.com/glossary/data-exploration
.
[2] https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

[3] SAMIRA POUYANFAR, SAAD SADIQ , YILIN YAN, HAIMAN TIAN, YUDONG TAO, MARIA PRESA REYES, MEI-LING SHYU, SHU-CHING CHEN and S. S. IYENGAR, " A Survey on Deep Learning: Algorithms, Techniques, and Applications", ACM Computing Surveys, Vol. 51, No. 5, Article 92. Publication date: September 2018.
[4] F. Sultana, A. Sufian, and P. Dutta, "Advancements in image classification using convolutional neural network," in 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Nov 2018, pp. 122–129.
[5] Kemal Adem , Exudate detection for diabetic retinopathy with circular Hough transformation and convolutional neural networks , Expert Systems With Applications 114 (2018) 289–295
[6] L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: semantic image segmentation with deep convolutional nets, atrousconvolution, and fully connected CRFs, IEEE Trans. Pattern Anal. Mach. Intell.40 (4) (2018) 834–848, http://dx.doi.org/10.1109/TPAMI.2017.2699184.
[7] Mohammed Hamzah Abed , Atheer Hadi Issa Al-Rammahi , Mustafa Jawad Radif , REAL-TIME COLOR IMAGE CLASSIFICATION BASED ON DEEP LEARNING NETWORK , Xinan Jiaotong Daxue Xuebao/Journal of Southwest Jiaotong University 54(5) 2019. DOI: 10.35741/issn.0258-2724.54.5.23
[8] Masoumeh Zareapoor, Pourya Shamsolmoali, Jie Yang. Learning depth super-resolution by using multi-scale convolutional neural network. Journal of Intelligent and Fuzzy Systems 36(2): 1773-1783 (2019)
[9] Ümit Budak, Zafer Cömert, Musa Çıbuk, Abdulkadir Şengür , DCCMED-Net: Densely Connected and Concatenated Multi Encoder-Decoder CNNs for Retinal Vessel Extraction from Fundus Images , Medical Hypotheses, Volume 134, January 2020, 109426. https://doi.org/10.1016/j.mehy.2019.109426
[10] Mesut TOĞAÇAR, Burhan ERGEN, Zafer CÖMERT , Classification of Flower Species by Using Features Extracted from the Intersection of Feature Selection Methods in Convolutional Neural Network Models , Measurement , Volume 158, 1 July 2020, 107703. https://doi.org/10.1016/j.measurement.2020.107703