CHAPTER 1

INTRODUCTION

This chapter presents an overview of the Diabetic Retinopathy. This chapter also goes through the stages of Diabetic Retinopathy, as well as the clinical signs, symptoms, and risk factors. This chapter contains a list of research objectives. This chapter concludes with a discussion on chapter organisation.

1.1 OVERVIEW OF DIABETIC RETINOPATHY:

According to WHO (World Health Organization) more than 347 million people are suffering from diabetes and it will be seventh prominent reason of death worldwide in 2030. Over the years, patients with diabetes tend to show abnormality in retina, due to emerging obstacle called Diabetic Retinopathy. People above 30 years having diabetes for more than 15 years, carry 78% chance of developing Diabetic Retinopathy. Diabetic Retinopathy is due of long-term standing of diabetic mellitus. Retinopathy means - damage of retina and as a result, the blood vessels become choked, leaky and grow arbitrarily. Diabetic Retinopathy is asymptomatic; it does not affect with view until it reaches at progress stage. Diagnosis of DR using deep learning techniques to classifying the stages of Diabetic Retinopathy in Transfer learning.*(Yingfeng Zheng et al (2012))*

1.2 DIABETIC RETINOPATY:

Diabetic retinopathy is a complication of diabetes that affects the eyes. It causes progressive damage to the retina, the light-sensitive lining in the back of the eye. Diabetic retinopathy occurs when changes in the blood glucose levels cause changes in retinal blood vessels. In some cases, these vessels will swell and leak fluid into the back of the eye. In other cases, abnormal blood vessels will grow on the surface of the retina. Over time, it can cause retinal tissue to swell, resulting in cloudy or blurred vision. The condition usually affects both eyes, and the longer a person has diabetes, the more likely they will develop diabetic retinopathy. Left untreated, it can eventually cause blindness.*(Diabetic retinopathy | AOA)*

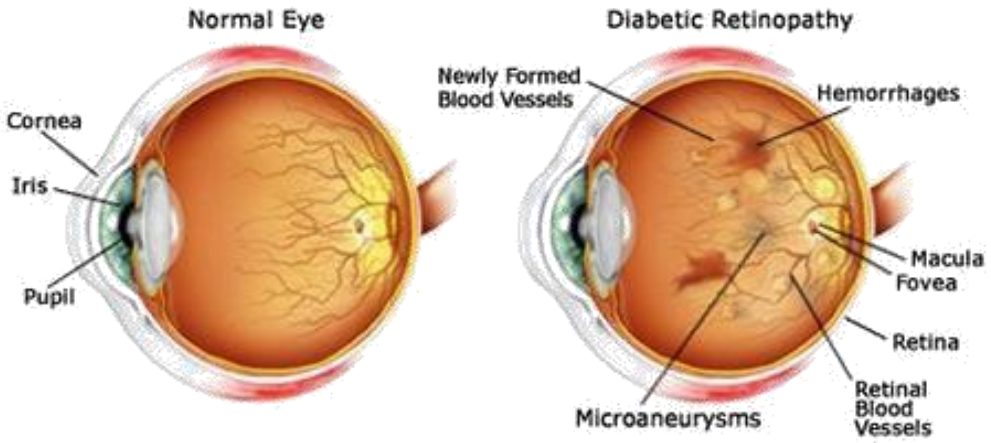

Figure 1.1 : Comparison of human retina of a healthy person and a person suffering from diabetic retinopathy *(eyedoctorophthalmologistnyc.com)*

1.3 STAGES OF DIABETIC RETINOPATHY:

There are two main types of diabetic eye disease.

- Non-Proliferative Diabetic Retinopathy (NPDR)
- Proliferative Diabetic Retinopathy (PDR)

Table 1.1 : Annotation of Diabetic Retinopathy[4]

Observable findings on fundus images	**Corresponding category**	**Suggested Treatment**
No abnormalities	Normal	Recheck in 12 months
Microaneurysm, exudation, but less than heavy	Mild or Moderate NPDR	Recheck in 6 months
Any of the following: 1. More than 20 intraretinal hemorrhagesin each of 4 quardrants 2. Define venus in 2+ quardarants 3. Prominent IRMA in 1+ quardrant 4. PDR but less than severe	Severe NPDR or Mild PDR	Scatter Laser Treatment
One or more of the following: Pre-macular hemorrhage, vitreous hemorrhage, severe retinal proliferative.	Severe PDR	Laser or vitrectomy treatment

1.3.1 FIVE CATEGORY OF DIABETIC RETINOPATHY:

The categorizes DR in terms of five stages *(eyedoctorophthalmologistnyc.com)*:

0. **No Diabetic Retinopathy**. No disease is yet apparent.

1. **Mild Diabetic Retinopathy**. Small areas of balloon-like swelling in the retinas, tiny blood vessels, called microaneurysms occur at the earliest stage of the disease. These microaneurysms may leak fluid into the retina. This is called *capillary leakage*.

2. **Moderate Diabetic Retinopathy**. As the disease progresses, blood vessels that nourish the retina may swell and distort. They may also lose their ability to transport blood. Both conditions cause characteristic changes to the appearance of the retina, may contribute to swelling of the macula and also cause further capillary leakage of blood or exudate.

3. **Severe Diabetic Retinopathy**. Many more blood vessels are blocked, depriving blood supply to areas of the retina. This is known as *capillary non-perfusion*. These areas secrete growth factors that signal the retina to grow new blood vessels.

4. **Proliferative Diabetic Retinopathy**. At this advanced stage, capillary non-perfusion triggers the proliferation of new blood vessels, which grow along the inside surface of the retina and into the vitreous gel, the fluid that fills the eye. The new blood vessels are fragile, which makes them more prone to capillary leakage. Accompanying scar tissue can contract and cause retinal detachment; the pulling away of the retina from underlying tissue (like wallpaper peeling away from a wall). Retinal detachment can lead to permanent vision loss.

1.3.2 CLINICAL SIGNS OF DIABETIC RETINOPATHY:

Clinical signs (depicted on Figure 1.1) include:

- **microaneurysms:** Small circular red lesions in the retina, represent vascular leakage, early signs of DR, not visually threatening, generally resolve within 3 to 4 months.

- **retinal haemorages:** Heamorages in different layers of retina, can be of different sizes and shapes.

- **hard (lipid) exudates:** Yellow irregular shaped lesions with sharp edges, represent lipids

and protein depositions. When accompanied by retinal thickening, represent a feature of diabetic macular edema, usually around the macula.

- **cotton-wool spots (soft exudates):** White lesions with faded edges, represent nerve fibre layer infarct, usually around the optic disc, the sign of retinal ischemia

- **Intraretinal microvascular abnormalities (IRMA):** Abnormal branching or dilation of existing blood vessels, can either be caused by the generation of new vessels or by remodeling of the existing ones.

- **macular edema**: Disturbance of the blood-retinal boundary produces leakage of plasma into the retinal tissue and swelling. It can set in in any DR stage and may result in the detachment of the retinal wall.

- **venous beading:** Dilated, irregular, tortuous veins, a non-specific sign of retinal ischemia.

- **neovascularisation:** An abnormal proliferation of new blood vessels. These vessels can grow into the vitreous cavity inflicting obscured vision and may result in vitreous haemorrhages.
- **vitreous haemorrhage:** Bleeding into the vitreous cavity.

1.4 SYMTOMS OF DIABETIC RETINOPATHY:

Patients report various levels of symptoms while suffering from DR. This is caused by the fact, that unless the macula is degraded, the patient might not notice any subjective symptoms.*(Diabetic retinopathy | AOA)*.

Figure 1.2 : Comparison of human vision of a person suffering from diabetic retinopathy and healthy person vision(*(fayophthalmoloy.com)*

The most common symptoms are:

- Spots or dark strings floating in your vision
- blurred vision
- Fluctuating vision
- increased sensitivity on light
- troubles seeing in the dark

1.5 RISK FACTOR OF DIABETIC RETINOPATHY:

The risk of developing the eye condition can increase as a result of:

- Having diabetes for a long time
- Poor control of your blood sugar level
- High blood pressure
- High cholesterol
- Pregnancy
- Tobacco use
- Body Mass Index

1.6 OBJECTIVES OF THE RESEARCH:

The objectives of the research aims to classify diabetic retinopathy into five stages.
The Specific objectives of the thesis are:

1) To process color fundus retinal images for Diabetic Retinopathy detection.
2) To extract key features from the pre-processed images.
3) To build the Transfer Learning Techniques for DR Detection
4) To detect the stage of Diabetic Retinopathy.
5) To evaluate the performance metrics of various Transfer Learning Methodologies.

1.7 ORGANIZATION OF CHAPTERS:

The rest of the technical is structure as follows:

- The first chapter presents the introduction for Diabetic Retinopathy Detection
- The second chapter presents the reviews on pre-processing and classification algorithms.
- The third chapter presents the company profile.
- The fourth chapter presents the methodology of various existing classification algorithm.
- The fifth chapter presents the statistical and performance results for APTOS dataset.
- The sixth and seventh chapter summarizes the research and presents directives for future research.

1.8 SUMMARY AND DISCUSSION:

This chapter explained about the overview of Diabetic Retinopathy and its various stages, types of Diabetic Retinopathy, Symptoms of Diabetic Retinopathy and the risk factors for Diabetic Retinopathy. This chapter also explains the objectives of the research and the organization of chapters. The upcoming chapter explains about the literature reviews, methodologies and results

CHAPTER 2

REVIEW OF LITERATURE

This chapter presents a reviews on Image preprocessing techniques done in fundus image by classifying the five stages in diabetic rethinopathy are discussed in this chapter. And also this chapter presents the reviews on convolution neural network and transfer learning various algorithms used for the fundus image to classify.

2.1 REVIEWS ON DR DETECTION USING IMAGE PROCESSING TECHNIQUES

- Rubina Sarki et al (2021) proposed an automated classification framework for Diabetic Eye Diseases was achieved in several steps: image quality enhancement, image segmentation (region of interest), image augmentation (geometric transformation), and classification. The optimal results were obtained using traditional image processing methods with a new build convolution neural network (CNN) architecture. The new built CNN combined with the traditional image processing approach presented the best performance with accuracy for DED classification problems. The results of the experiments conducted showed adequate accuracy, specificity, and sensitivity.
- Harry Pratt et al (2016) proposed a CNN approach to diagnose DR from digital fundus images and to accurately classify its severity. And also developed a network with CNN architecture and data augmentation which can identify the intricate features involved in the classification task such as micro-aneurysms, exudate and haemorrhages on the retina and consequently provide a diagnosis automatically and without user input. The proposed CNN achieved a sensitivity of 95% and an accuracy of 75% on 5,000 validation images
- D. Jude Hemanth et al (2018) proposed includes employment of image processing with histogram equalization, and the contrast limited adaptive histogram equalization techniques. Next, the diagnosis is performed by the classification of a convolutional neural network. The method was validated using 400 retinal fundus images within the MESSIDOR database, and average values for different performance evaluation parameters were obtained as accuracy 97%, sensitivity (recall) 94%, specificity 98%, precision 94%, FScore 94%.

- Mohamed Chetoui, et al (2018) performed Local Ternary Pattern (LTP) and Local Energy-based Shape Histogram (LESH). We show that they outperform LBP extracted features. Support Vector Machines (SVM) are used for the classification of the extracted histogram. A histogram binning scheme for features representation is proposed. The experimental results show that LESH is the best performing technique with an obtained accuracy of 0.904 using SVM with a Radial Basis Function kernel Similarly, the analysis of the ROC curve shows that LESH with SVM-RBF gives the best Area Under Curve performance with 0.931.

2.2 REVIEWS ON CNN BASED DR DETECTION

- Zhentao Gao et al (2018) trained deep convolutional neural network models to grade the severity of DR fundus images. They achieve an accuracy of 88.72% for a four-degree classification task in the experiments. They deployed their models on a cloud computing platform and provided pilot DR diagnostic services for several hospitals; in the clinical evaluation, the system achieved a consistency rate of 91.8% with ophthalmologists, demonstrating the effectiveness on their work.

- Muhammad Mateen et al (2018) proposed DR classification system achieves a symmetrically optimized solution through the combination of a Gaussian mixture model, visual geometry group network, singular value decomposition and principle component analysis (PCA), and softmax, for region segmentation, high dimensional feature extraction, feature selection and fundus image classification, respectively. Utilization of PCA and SVD feature selection with fully connected (FC) layers demonstrated the classification accuracy of 92.21%, 98.34%, 97.96%, and 98.13% for FC7-PCA, FC7-SVD, FC8-PCA, and FC8-SVD, respectively.

- Qilei Chen et al (2019) analyzed the lesion-vs-image scale carefully and propose a large-size feature pyramid network to preserve more image details for mini lesion instance detection. Their method includes an effective region proposal strategy to increase the sensitivity. The experimental results shows that their proposed method was superior to the original feature pyramid network method and Faster RCNN.

- Darshit Doshi et al (2016)presents the design and implementation of GPU accelerated deep convolutional neural networks to automatically diagnose and thereby classify high-resolution retinal images into 5 stages of the disease based on severity. The single model accuracy of the convolutional neural networks presented in this paper is 0.386 on a quadratic weighted kappa metric and ensembling of three such similar models resulted in a score of 0.3996.

- T Chandrakumar et al (2016) proposed deep learning approach such as Deep Convolutional Neural Network (DCNN) gives high accuracy in classification of these diseases through spatial analysis. A DCNN is more complex architecture inferred more from human visual Prospects. Proposed architecture deployed with dropout layer techniques yields around 94-96 percent accuracy. Also, it tested with popular databases such as STARE, DRIVE, kaggle fundus images datasets are available publicly.

- Hager Khalil et al (2019) Convolutional Neural Networks (CNN) is performed to classify the retinal fundus images to normal, background and pre-proliferative retinopathy. The proposed model consists of 5 layers. Finally, a global average pooling is used. In this work they achieve accuracy reached 95.23%.

2.3 REVIEWS ON TRANSFER LEARNING BASED DR DETECTION

- SEHRISH QUMMAR et al (2019) used the publicly available Kaggle dataset of retina images to train an ensemble of five deep Convolution Neural Network (CNN) models (Resnet50,Inceptionv3, Xception, Dense121, Dense169) to encode the rich features and improve the classification for different stages of DR. The experimental results show that the proposed model detects all the stages of DR unlike the current methods and performs better compared to state-of-the-art methods on the same Kaggle dataset.

- Carson Lam et al (2019) proposed preprocessing with contrast limited adaptive histogram equalization and ensuring dataset fidelity by expert verification of class labels improves recognition of subtle features. Transfer learning on pretrained GoogLeNet and AlexNet models from ImageNet improved peak test set accuracies to 74.5%, 68.8%, and 57.2% on 2-ary, 3-ary, and 4-ary classification models, respectively.

2.5 SUMMARY AND DISCUSSION:

This chapter presents a reviews on Image preprocessing techniques done in fundus image by classifying the five stages in diabetic rethinopathy are discussed in this chapter. And also this chapter presents the reviews on convolution neural network and transfer learning various algorithms used for the fundus image to classify.

CHAPTER 3

ORGANIZATION PROFILE

This chapter presents the details of the organization. This chapter also represents the name, services of the company. The organization flow chart, and specialization of organization are also discussed in detail.

3.1 ORGANIZATION NAME:

Deep Medicine Labs

3.2 ABOUT THE COMAPANY

They are a team of public health scientists, epidemiologists, medical doctors, biodesign engineers and health technology entrepreneurs united with the idea to solve problems in public health using technology. Our core areas of interdisciplinary expertise include **Clinical Data Science, Clinical AI, Biodesign** and **Health Technology.**

3.3 SERVICES

They have a R&D department (Clinical R&D and Technology R&D) that identifies potential problems in public health and develop tailor made solutions as proof-of-concepts. The proof-of-concepts are translated into health technology products through an interdisciplinary effort of Deep Medicine Labs with its collaborative partners and stakeholders. In addition, They also engage actively in knowledge sharing through training/workshops, Webinars/Seminars, Lectures, Internships and Certificate Courses. At present They are working on a Depression Detection algorithm using vocal biomarkers for our product named “psychENHANCE”

3.4 ORGANIZATION CONTACT

FOUNDER AND CEO : Dr.ArunKumar Annamalai, Director of Clinical Research and Dr.Prakash, Principal Scientific Advisor

CHAPTER 4

METHODOLOGY

This chapter presents the system architecture of the compared algorithm for classifying of Diabetic Retinopathy. We divide the total process into 2 steps- preprocessing and Diabetic Retinopathy classification. Preprocessing includes green channel extraction, Contrast Limited Adaptive Histogram Equalization and Gaussian Blur. Finally, in the classification phase, we will detect whether Diabetic Retinopathy is present or not. Moreover, if present, whether it is Mild, Moderate, Severe and PDR. Finally the Compared algorithm pseudocodes are also pointed out in this chapter.

4.1 SYSTEM ARCHITECTURE:

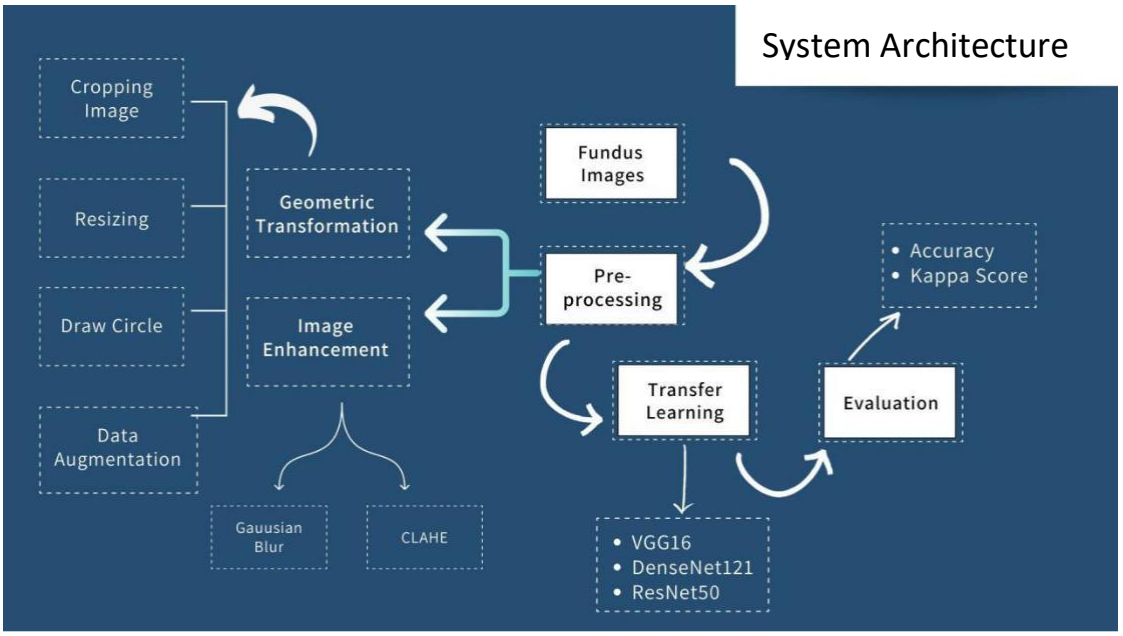

Figure 4.1: FlowChart of System Architecture

4.2 IMAGE PRE-PROCESSING METHODS:

There are many features required for the diagnosis of diabetic retinopathy using fundus images, as well as various other eye diseases which can be apparent in fundus images. These features are exudate, micro-aneurysms, haemorrhages and drusen all of which have varying contrast and smoothness in a fundus image. Therefore, the image pre-processing undertaken prior to training is important so that the network learns to identify the correct features for classification(*Harry Pratt*)

4.2.1 CROP IMAGE:

The some images were chopped at the top and bottom, they had to be standardized. Since the field of view (FOV) of an image (the section of the retina seen in the image), is circular and first cropped to a square of side equal to the diameter of the FOV. (*DILIP SINGH SISODIA et al (2017)*)**.**

4.2.2 RESIZING:

Since the original images vary widely in size, Resize the image to a new width and height.

4.2.3 DRAWING A CIRCLE:

Initialize the image with 255 and find the centre of the image. The variable 'sz' changes the size of the image. The image is a square matrix. Then 'rad' contains the **radius of the circle**(*Circle | IMAGE PROCESSING (imageeprocessing.com).*

4.2.4 IMAGE AUGMENTATION:

Image augmentation is a technique of altering the existing data to create some more data for the model training process. In other words, it is the process of artificially expanding the available dataset for training a deep learning model(*Image Augmentation for Deep Learning (analyticsvidhya.com).*

4.3 IMAGE ENHANCEMENT:

Image enhancement refers to the process of highlighting certain information of an image, as well as weakening or removing any unnecessary information according to specific needs. For example, eliminating noise, revealing blurred details, and adjusting levels to highlight features of an image.(*Image Processing 101 Chapter 2.1: Image Enhancement | Dynamsoft Blog)*

4.3.1 GREEN CHANNEL EXTRACTION:

Pre-processing of fundus image is performed in order to improve the contrast. In order to enhance the contrast of the retinal images, some information is commonly discarded before processing such asthe red and blue components of the image. Green channel is extensively used in pre-processing as it displays the best vessels/background contrast and greatest contrast between the optic disc and retinal tissue. Red channel is relatively bright and vascular structure of the choroid is visible. The retinal vessels are also visible but show less contrast than green channel. Blue channel is noisy and contains little information (*Tareq Mahmud et al (2020)*).

4.3.2 CONTAST LIMITED ADAPTIVE HISTROGRAM EQUALIZATION:

Contrast Limited Adaptive histogram equalization (CLAHE) is used for contrast enhancement. CLAHE computes several histograms of image and uses them to reallocate intensity value of image. Hence, CLAHE is more appropriate to improve regional contrast and edge enhancement in each region of image (*Tareq Mahmud et al (2020)*).

4.3.3 GAUSSIAN BLUR:

The Gaussian smoothing operator is used to `blur' images and remove noise

4.4 TRANSFER LEARNING APPROACHES:

Transfer learning is the reuse of deep learning models that are pre-trained on huge datasets such as subsets of the Image Net database to fit to a previously unseen dataset. Deep Learning (DL) is part of machine learning that works fine when there is huge training data and large computational power for solving classification and regression problems. Convolution neural network(CNN) is one of instances of DL architecture that works thriving with multi-dimensional data such as images and videos. DL requires huge extent of training data to correctly work. The lack of enough labeled training data is one of the key challenges of applying DL in health industry. Transfer learning, on deep convolution neural networks, has gained attention due to the lack of enough training data. Transfer learning helps to reduce the deep learning model training time.*(Chaitanyanarava | Medium)*

All the pretrained model architectures are extended with following three layers.

1. **GlobalAveragePooling2D:**

   From the last convolutional layer of any pretained model which generates as many feature maps as the number of target classes, and applies global average pooling to each in order to convert each feature map into one value by taking average of it.

2. **Dropout**

   Dropout works by randomly setting the outgoing edges of hidden units to 0 at each update of the training phase. Which will be used as a sort of regularization in order to avoid overfitting during model training.

3. **Dense**

   This layer contains 5 units each represents the individual class with sigmoid activation.

   Optimizer: Adam

   loss: binary cross entropy

   metric: accuracy and weighted kappa

4.4.1 VGG16:

Visual Geometry Group(VGG).

VGG has 16 or 19 layers starting with simple 3x3 filters in first convolutional layer. Number of channels increased with term of 2 and Nh, Nw decreased with factor of 2. Though it contains lots of parameters, the model is simple with fixed 3x3 filters and stride=1 and same padding in convolutional layers.

2x2 filters with stride=2 in max pooling layers.

VGG-16 architecture:

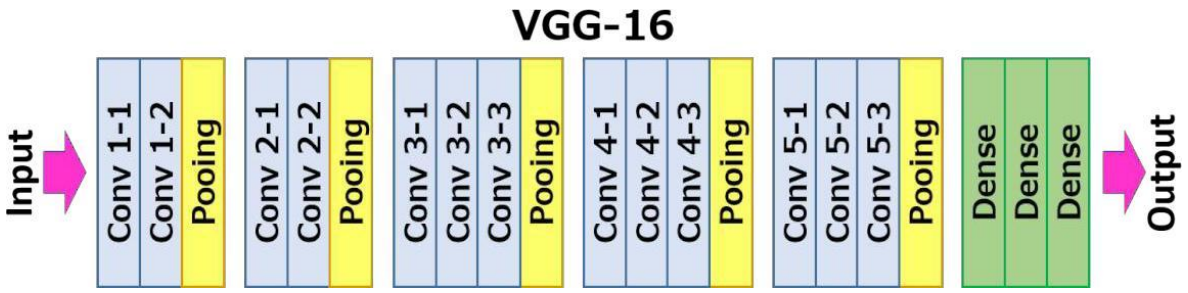

Figure 4.2: VGG16 Architecture

Arguments

include_top: whether to include the 3 fully-connected layers at the top of the network.
weights: one of None (random initialization), 'imagenet' (pre-training on ImageNet), or the path to the weights file to be loaded.
input_tensor: optional Keras tensor (i.e. output of layers.Input()) to use as image input for the model.
input_shape: optional shape tuple, only to be specified if include_top is False (otherwise the input shape has to be (224, 224, 3) (with channels_last data format) or (3, 224, 224) (with channels_first data format). It should have exactly 3 input channels, and width and height should be no smaller than 32. E.g. (200, 200, 3) would be one valid value.

4.4.2 DenseNet 121:

Densenet was developed specifically to improve the declined accuracy caused by the vanishing gradient in in high-level neural networks. Due to the longer path between the input and output layer, the information vanishes before reaching its destination. It is a 5-layer dense block with a growth rate of k = 4. An output of the previous layer acts as an input of the second layer by using composite function operation. This composite operation consists of the convolution layer, pooling layer, batch normalization, and non-linear activation layer.

DenseNet with 121 layers:

=> 5+(6+12+24+16)*2 = 121

5 - Convolutional and Pooling

layer 3 - Transition layers(6,12,24)

1 - Classification layer(16)

2 - DenseBlock(1x1 and 3x3 conv)

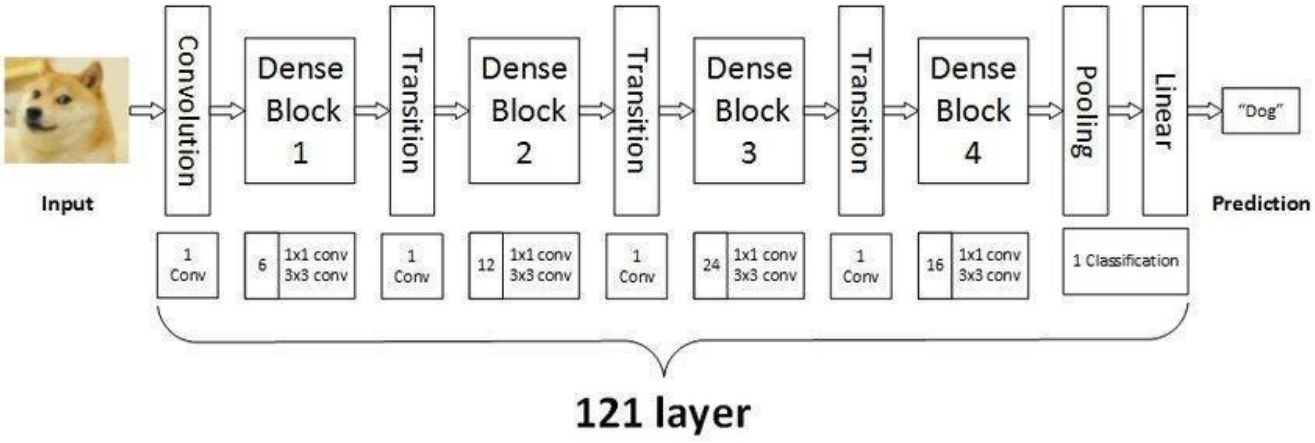

Figure 4.3: DenseNet121 Architecture

4.4.3 ResNet50:

ResNet has many variants that run on the same concept but have different numbers of layers.Resnet50 is used to denote the variant that can work with 50 neural network layers. Instead
of directly stacking few layers to fit an underlying H(x) mapping. Explicitly let these layers to fit a residual mapping F(x).

Now,

$F(x) = H(x) - x$ (4.1)

$H(x) = F(x) + x$ (4.2)

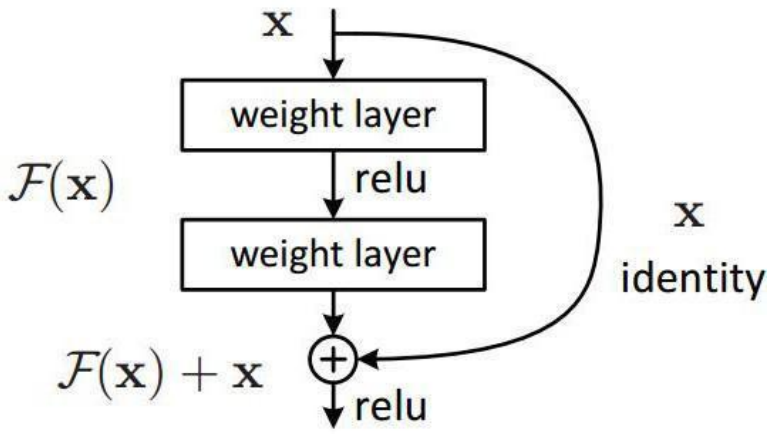

Figure 4.4: ResNet mapping

- Now this residual mapping (F(x)+x) are called skip connections which skips one or more layers.
- These connections perform identity mapping and there outputs are added to the outputs of the stacked layers.
- These connections add neighter extra parameters nor computational complexity.

ResNet50, the building block was modified into a bottleneck design due to concerns over the time taken to train the layers. This used a stack of 3 layers instead of the earlier 2.

Therefore, each of the 2-layer blocks in Resnet34 was replaced with a 3-layer bottleneck block, forming the Resnet 50 architecture. This has much higher accuracy than the 34-layer ResNet model. The 50-layer ResNet achieves a performance of 3.8 bn FLOPS.

Classification Algorithm

Require: Fundus Images (X, Y); where Y = {y/y ∈ {Normal, Mild, Moderate, Severe, PDR}}

Output : The trained model that classifies the fundus image x ∈
X Perform Preprocessing:

- Resize the image to dimension 256 x 256
- Perform augmentation: Randomly crop five patches, of size 256 x 256, of each image and perform flip flop, zoom = 0.2 and 180o rotation

Import a set of pre-trained models H = {Resnet50,Xception, Dense121, VGG16}.
Replace the last fully connected layer of each model by a layer of (5 × 1) dimension.

```
foreach ∀h ∈ H do
    α = 0.0005
    for epochs=1 to 5 do
        foreach mini batch (Xi, Yi) ∈ (Xtrain, Ytrain) do
            Update the parameters of the model h(·)
            if the validation error is not improving for five epochs then
                α = α × 0.0005
            end
        end
    end
end
foreach x ∈ Xtest do
    Ensemble the output of all models, h ∈ H
end
```

Figure 4.5: Pseudocode of proposed approach for DR detection

4.5 SUMMARY AND DISCUSSION:

In this chapter, the detail explanation about the data pre-processing and various transfer learning algorithms performed in APTOS dataset. In the upcoming chapter, the results are discussed.

CHAPTER 5

RESULTS AND DISCUSSION

This chapter presents the result and discussion about the dataset and the various pre-processsing techniques used. The results of the transfer learning, after preprocessing are described.

5.1 Datasets used for Experimental Study

A Data collected from APTOS dataset which provides by kaggel. The dataset consists of colour fundus photographs collected from various sources. The images are classified based on the severity of DR, where each image was assigned to a class by a trained clinician

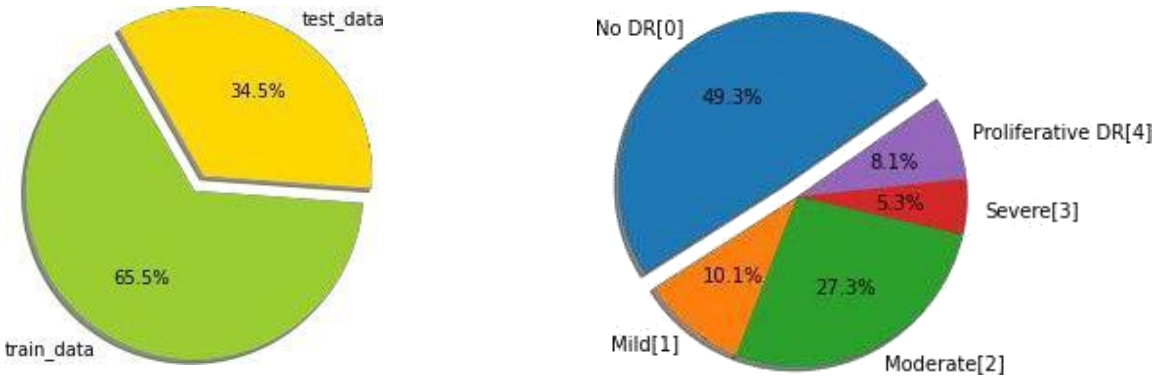

Figure 5.1 *Pie-chart Analysis of size of train and test dataset and Pie-chart Analysis of Number of images on each target label*

The Dataset which consist of(figure 5.1)

- Train_images: directory containing training images(total 3662 images)
- Test_images : directory containing test images(total 1928 images)

5.2 Pre-processing used in this study

5.2.1 Crop Image:

All the images contain a black background. And we can observe some of the images have extra dark pixels at the sides of the images. So we will crop of those extra dark pixels.

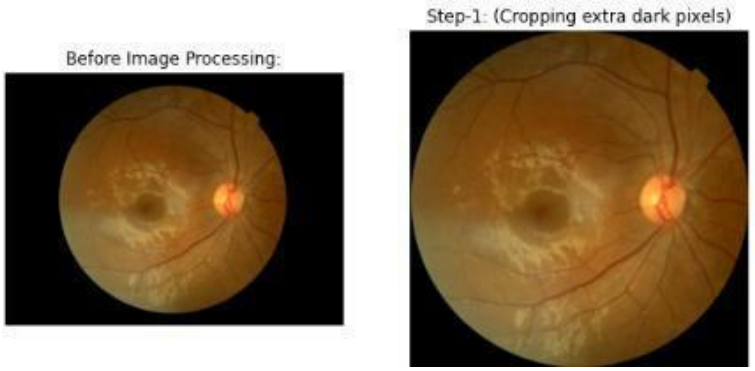

Figure 5.2: *original image (left) and cropping the extra dark pixels(right)*

5.2.2 Resizing

As the dataset is huge in size, the images are stored in .png format with size 256x256 pixels for reducing the computational time.

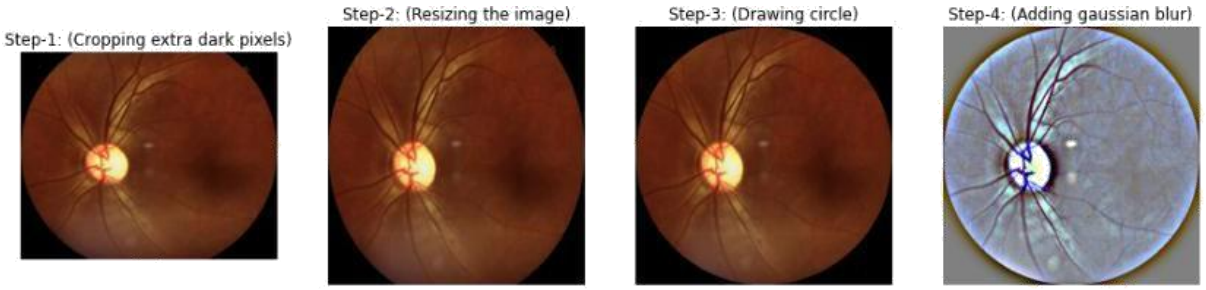

Figure 5.3: *second image:resizing, third image: locate a retina in a circle and finally applied gaussian blur*

5.2.3 Contrast Limited Adaptive Histogram Equalization:

CLAHE is a variant of Adaptive histogram equalization (AHE) which takes care of over-amplification of the contrast. This algorithm can be applied to improve the contrast of images.

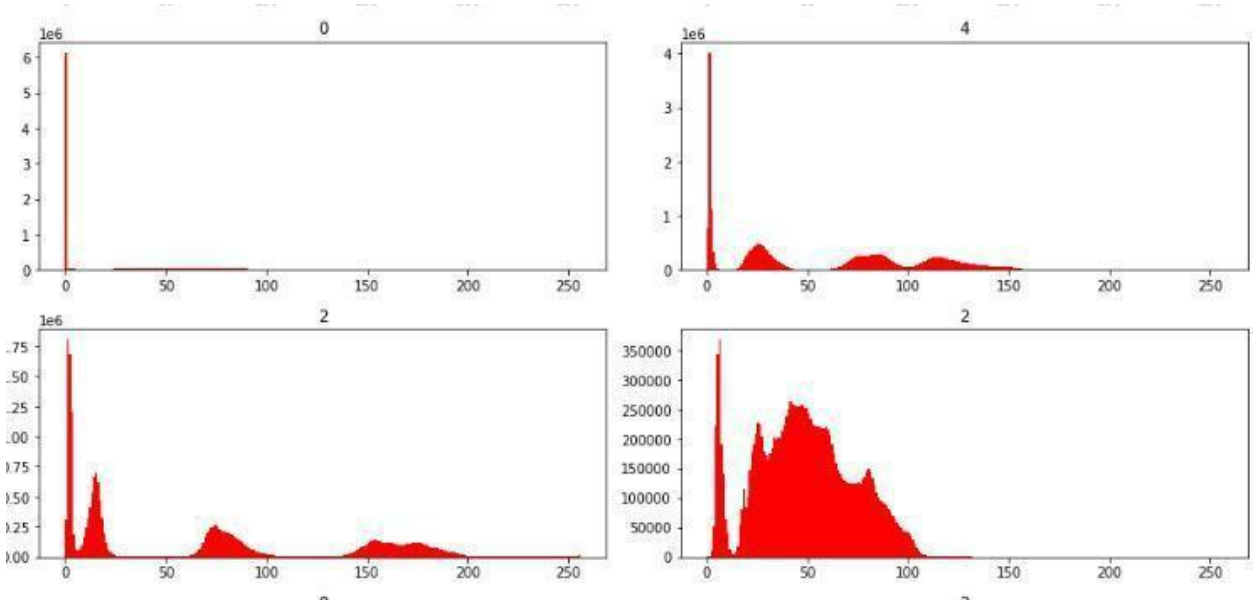

Figure 5.4: *four retina images histogram(before CLAHE)*

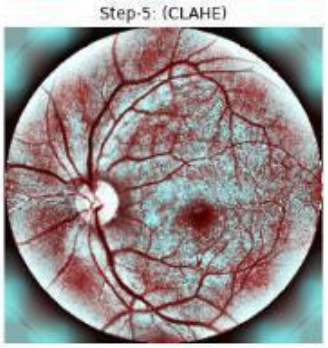

Figure 5.5: *CLAHE used*

It was observed that there was an improvement in the contrast of the fundus images after CLAHE was used. Figure 5.5 and Figure 5.6 show the difference of the retinal images.

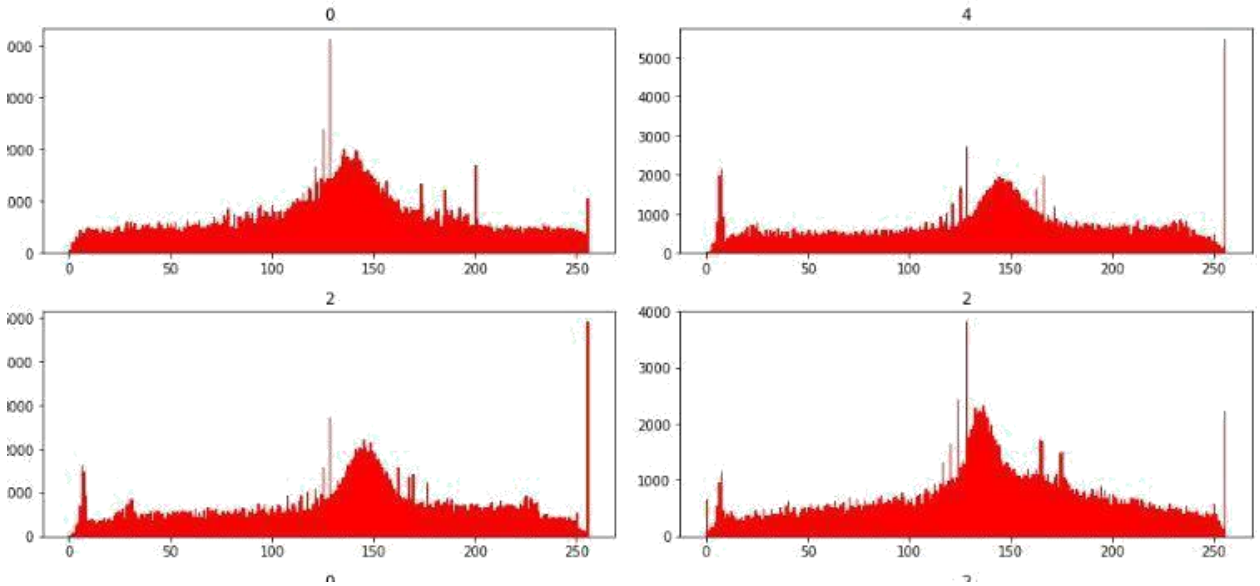

Figure 5.6: *four pre-processed retina images histogram(after CLAHE)*

5.2.4 Data Augmentation:

We hardly have ~3k images on training and ~500 images for validation. Any deep learning model with small dataset will prone to overfit easily.

By augmentation we can increase the size of our dataset at runtime without actually storing them in our system. I used the following four augmentation techniques which will increase our dataset by 4 times.

1. Horizontal flipping
2. Vertical flipping
3. Rotation_range = 180
4. Zoom range = 0.2

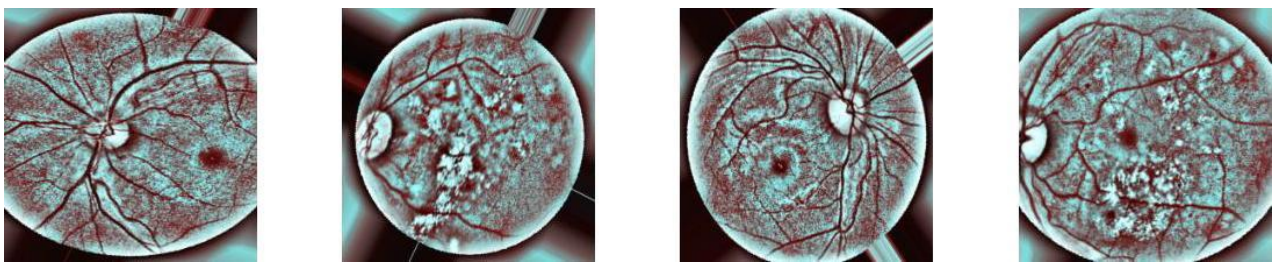

Figure 5.7: *Data Augmentation applied in a image*

5.3 Performance measures used in this study

5.3.1 Kappa Score

Cohen's kappa coefficient is a statistic which measures inter-rater agreement for qualitative (categorical) items. It is generally thought to be a more robust measure than simple percent agreement calculation, since k takes into account the agreement occurring by chance. Cohen's kappa measures the agreement between two raters who each classify N items into C mutually exclusive categories.*(Statistics - Cohen's kappa coefficient (tutorialspoint.com)*

Cohen's kappa coefficient is defined and given by the following function −

$$k = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e} \tag{5.1}$$

Where −

p0 = relative observed agreement among raters.

pe = the hypothetical probability of chance agreement.

p0 and pe are computed using the observed data to calculate the probabilities of each observer randomly saying each category. If the raters are in complete agreement then k = 1. If there is no agreement among the raters other than what would be expected by chance, k ≤ 0.

5.3.2 Confusion Matrix

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

Let's now define the most basic terms, which are whole numbers (not rates):

- **True positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
- **True negatives (TN):** We predicted no, and they don't have the disease.
- **False positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **False negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

5.3.3 Accuracy

Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.(*Accuracy, Precision, and Recall in Deep Learning | Paperspace Blog)*

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}} \quad (5.2)$$

5.3.4 Precision

The precision is calculated as the ratio between the number of Positive samples correctly classified to the total number of samples classified as Positive (either correctly or incorrectly). The precision measures the model's accuracy in classifying a sample as positive.(*Accuracy, Precision, and Recall in Deep Learning | Paperspace Blog)*

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}} \quad (5.3)$$

5.3.5 Recall

The recall is calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect Positive samples. The higher the recall, the more positive samples detected.(*Accuracy, Precision, and Recall in Deep Learning | Paperspace Blog)*

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}} \quad (5.4)$$

5.4 TRANSFER LEARNING MODELS:

VGG16

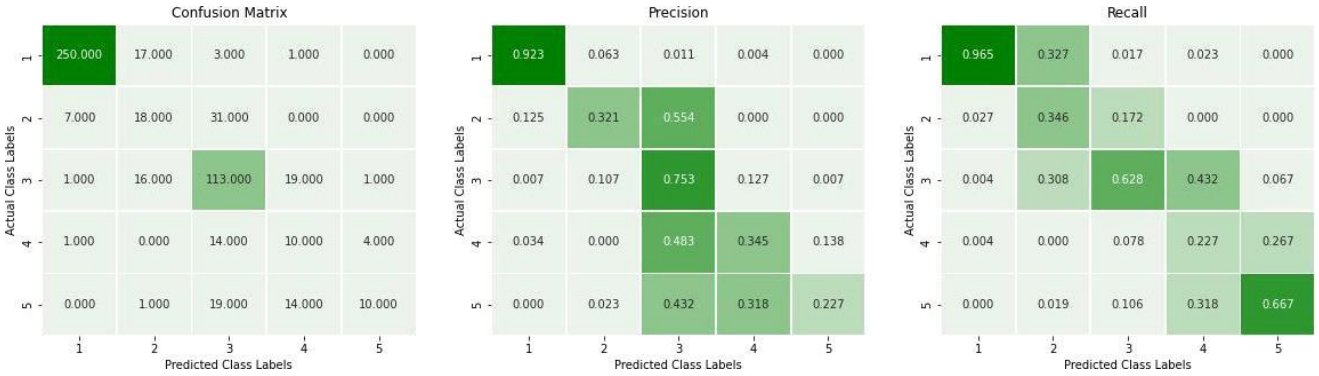

Figure 5.8: *Classification report of VGG16*

DenseNet121

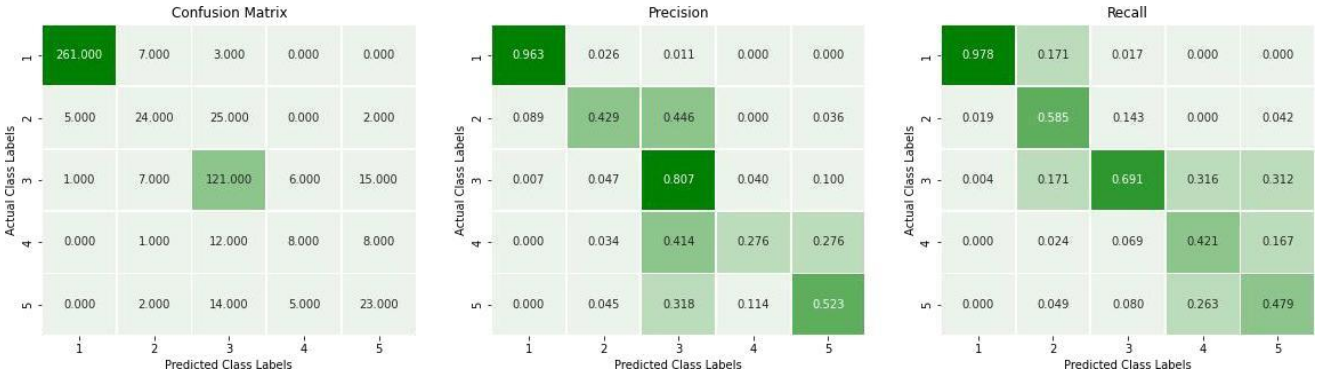

Figure 5.9: *Classification report of DenseNet121*

ResNet50

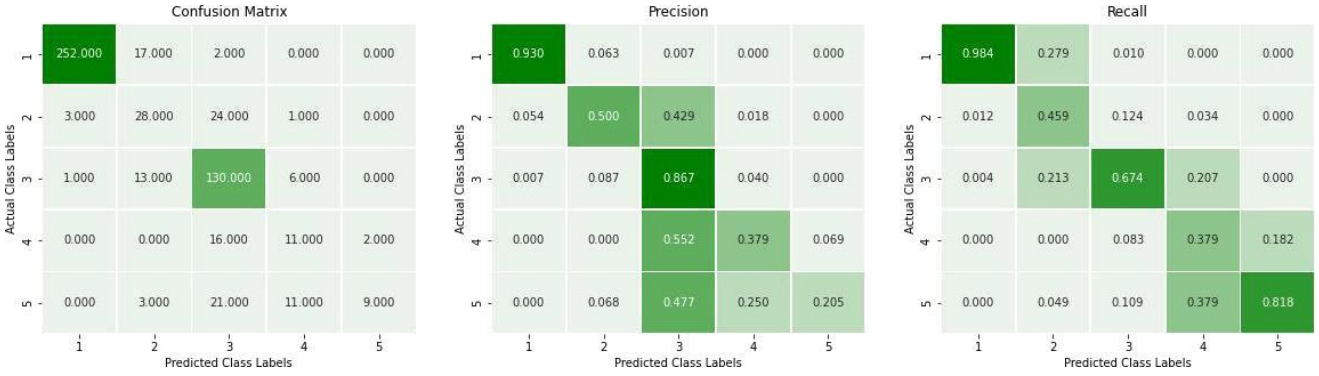

Figure 5.10: *Classification report of ResNet50*

The above figures represents the confusion matrix, precision and recall plotted for actual class labels and predicted class labels.

5.5 COMPARING THE MODELS:

Table 5.1 : Comparison of performance metrics for various transfer learning approaches

Model	Accuracy	Kappa Score
Vgg16	0.9164	0.8532
DenseNet121	0.8164	0.8724
ResNet50	0.9018	0.8623

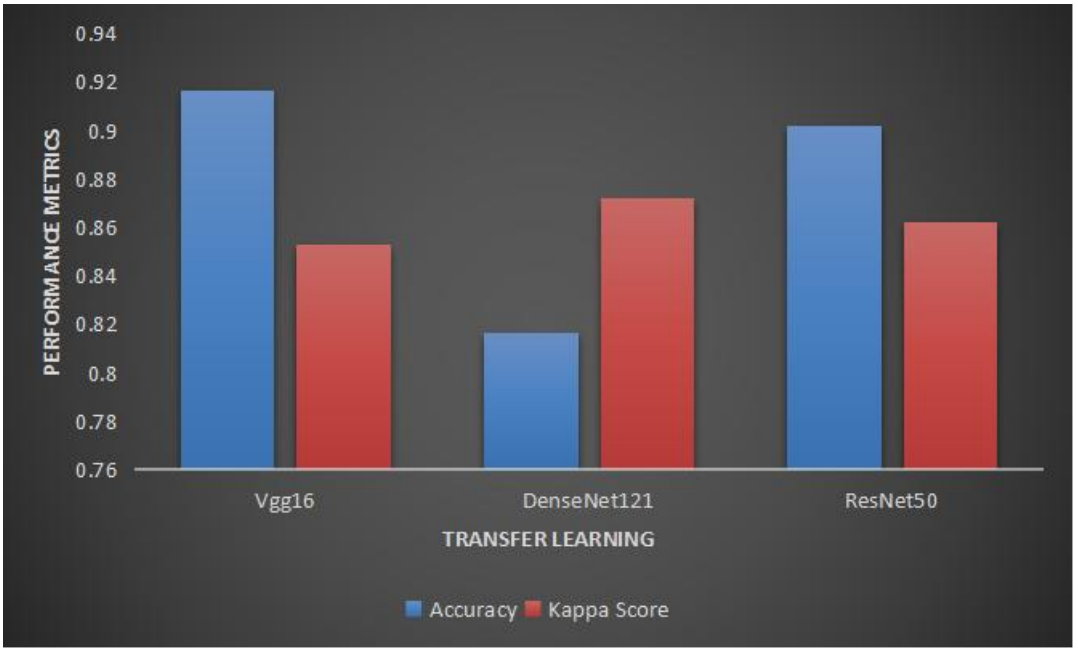

Figure 5.11 : Comparison of performance metrics for various transfer learning approaches

A comparative study is made on various techniques. After evaluation of well-known technique, it is clearly shown the various methods which can detect the DR efficiently and provide accurate result. In the above table 5.1 and figure 5.11 we can see that DenseNet121 gives us the best result

5.6 SUMMARY AND DISCUSSION:

This chapter describes about the data, image by using the different preprocessing technique and performance metrics used to evaluate. All the Comparison is summarized in a table and we have also plotted for all the techniques.

CHAPTER 6

CONCLUSION

In this paper, pre-processing of the diabetic retinal fundus image is done for the detection of diabetic retinopathy using machine learning techniques. The pre-processing techniques such as green channel extraction, histogram equalization and resizing were performed using google colab. Transfer learning(Vgg16,ResNet50 and DenseNet121) is implemented to classify DR into 5 classes. DenseNet121 performed better than the other transfer learning models with quadratic weighted kappa function. Deep learning techniques that can learn from small dataset to categorize medical images should be utilized to classify DR, as this can be transferred to other medical image classification problems facing the challenge of insufficient training data.

CHAPTER 7

FUTURE ENHANCEMENT

Apply the feature extraction part from pre-trained model and apply to algorithms such as support vector machines and changing the performance measures such as specificity and sensitivity as it gives healthcare more trust to model usage in real time. Apply the different image pre-processing techniques to the dataset for improve features strength to classify and compare the performances and will compare the different transfer learning techniques and apply the pre-trained models to complex image classification challenges in real world problems.Using fundus images, will diagnosis other diseases related to diabetic retinopathy. Create automated web applications to detect the stages of diabetic retinopathy of that person in a minute.

CHAPTER 8

REFERENCES

1. Sarki, R., Ahmed, K., Wang, H. *et al.* Image Preprocessing in Classification and Identification of Diabetic Eye Diseases. *Data Sci. Eng.* **6,** 455–471 (2021). https://doi.org/10.1007/s41019-021-00167-z
2. Q. Chen, X. Sun, N. Zhang, Y. Cao and B. Liu, "Mini Lesions Detection on Diabetic Retinopathy Images via Large Scale CNN Features," 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), 2019, pp. 348-352, doi: 10.1109/ICTAI.2019.00056.
3. Shaban, M., Ogur, Z., Mahmoud, A., Switala, A., Shalaby, A., Abu Khalifeh, H., Ghazal, M., Fraiwan, L., Giridharan, G., Sandhu, H., & El-Baz, A. S. (2020). A convolutional neural network for the screening and staging of diabetic retinopathy. *PloS one*, *15*(6), e0233514. https://doi.org/10.1371/journal.pone.0233514
4. M. Chetoui, M. A. Akhloufi and M. Kardouchi, "Diabetic Retinopathy Detection Using Machine Learning and Texture Features," 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), 2018, pp. 1-4, doi: 10.1109/CCECE.2018.8447809.
5. Lam, C., Yi, D., Guo, M., & Lindsey, T. (2018). Automated Detection of Diabetic Retinopathy using Deep Learning. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, *2017*, 147–155.
6. Gangwar, A.K., Ravi, V. (2021). Diabetic Retinopathy Detection Using Transfer Learning and Deep Learning. In: Bhateja, V., Peng, SL., Satapathy, S.C., Zhang, YD. (eds) Evolution in Computational Intelligence. Advances in Intelligent Systems and Computing, vol 1176. Springer, Singapore. https://doi.org/10.1007/978-981-15-5788-0_64
7. Nikos Tsiknakis, Dimitris Theodoropoulos, Georgios Manikis, Emmanouil Ktistakis, Ourania Boutsora, Alexa Berto, Fabio Scarpa, Alberto Scarpa, Dimitrios I. Fotiadis, Kostas Marias,Deep learning for diabetic retinopathy detection and classification based on fundus images: A review,Computers in Biology and Medicine,Volume 135,2021,104599,ISSN 0010-4825, https://doi.org/10.1016/j.compbiomed.2021.104599.

8. Chandrakumar T, R Kathirvel, 2016, Classifying Diabetic Retinopathy using Deep Learning Architecture, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 05, Issue 06 (June 2016), http://dx.doi.org/10.17577/IJERTV5IS060055
9. M. Chetoui, M. A. Akhloufi and M. Kardouchi, "Diabetic Retinopathy Detection Using Machine Learning and Texture Features," 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), 2018, pp. 1-4, doi: 10.1109/CCECE.2018.8447809.
10. Lam C, Yi D, Guo M, Lindsey T. Automated Detection of Diabetic Retinopathy using Deep Learning. AMIA Jt Summits Transl Sci Proc. 2018 May 18;2017:147-155. PMID: 29888061; PMCID: PMC5961805.
11. Zheng, Yingfeng et al. “The worldwide epidemic of diabetic retinopathy.” *Indian journal of ophthalmology* vol. 60,5 (2012): 428-31. doi:10.4103/0301-4738.100542.
12. Khalifa, N., Loey, M., Taha, M., & Mohamed, H. (2019). Deep Transfer Learning Models for Medical Diabetic Retinopathy Detection. *Acta informatica medica : AIM : journal of the Society for Medical Informatics of Bosnia & Herzegovina : casopis Drustva za medicinsku informatiku BiH*, *27*(5), 327–332. https://doi.org/10.5455/aim.2019.27.327-332

APPENDIX

- **DR_EDA.ipynb** #Import

```
Dataset from Kaggel: !pip
install kaggle
! pip install kaggle torchvision torch pandas matplotlib numpy scipy scikit-
learn !mkdir ~/.kaggle1
import json
token = {"username":"annjovi","key":"0f76f8---------"}
with open('/content/kaggle.json', 'w') as file:
   json.dump(token, file)
file.close()
!ls -a
!cp /content/kaggle.json
~/.kaggle/kaggle.json !kaggle config set -n
path -v{/content} !chmod 600
/root/.kaggle/kaggle.json !kaggle datasets list
import os
os.chdir('/content/drive/My Drive/DR-Data-
Blindness_Detection') !cp /content/drive/MyDrive/kaggle.json
~/.kaggle/kaggle.json
!kaggle competitions download -c aptos2019-blindness-detection -p
/content !unzip aptos2019-blindness-detection.zip
#Load Datasets:
# for system settings import os
os.chdir('/content/drive/My Drive/')
import warnings

from tqdm import tqdm
warnings.filterwarnings('ignore')
import time
# for excessing or creating tabular data
import pandas as pd
import pandas.util.testing as tm
# for matrix manipulation
```

```
import numpy as np
# for visualization
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
# for splitting the data
from sklearn.model_selection import train_test_split
train_data = pd.read_csv("/content/drive/MyDrive/DR-Data-Blindness_Detection/train.csv")
print("train.csv:")
print("Number of Training images: {}\n".format(train_data.shape[0]))
print(train_data.head(2),"\n")
print("-"*100)
test_data = pd.read_csv("/content/drive/MyDrive/DR-Data-Blindness_Detection/test.csv")
print("test.csv: ")
print("Number of Testing images: {}\n".format(test_data.shape[0]))
print(test_data.head(2))
print("\n","-"*100)
print("sample_submission.csv:")
sample_submission = pd.read_csv("/content/drive/MyDrive/DR-Data-
Blindness_Detection/sample_submission.csv")
print("The format of submitting the final predictions on testing images: ")
print(sample_submission.head(5))
# Intialization of variables which are useful for the later tasks.
img_width = 256
img_height = 256
no_channels = 3
split_size = 0.15
class_labels = {0: 'No DR[0]',1: 'Mild[1]', 2: 'Moderate[2]', 3: 'Severe[3]', 4: 'Proliferative DR[4]'}


#Exploratory Data Analysis:
data = [len(train_data), len(test_data)] print("Number
of Images in train dataset: ", data[0]) print("Number
of Images in test dataset: ", data[1])
```

```
labels = ['train_data','test_data']
plt.pie(data,explode = [0,0.1], labels= labels, shadow = True, colors =
['yellowgreen','gold'],autopct=' %1.1f%%', startangle = 120)
plt.title('Pie Chart Analysis of size of train and test datasets')
plt.axis('equal')
plt.show()
class_labels_ = list(set(train_data['diagnosis']))
print("Number of target classes: {}".format(class_labels_))
class_sizes = []
for i in range(0,5):
    class_sizes.append(list(train_data['diagnosis']).count(i))
labels = class_labels.values()
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue','darkgreen']
plt.pie(class_sizes,explode = [0.1,0,0,0,0], labels= labels, shadow = True,autopct='%1.1f%%', startang
le = 35)
plt.title('Pie Chart Analysis of Number of Images on each target label:')
plt.show()
#Splitting the Dataset:
def splitting_data(train_data, size, is_split = True):
    try:
        if is_split:
            data = train_data['id_code']
            labels = train_data['diagnosis']
            train_x, validation_x, train_labels, validation_labels = train_test_split(data, labels, stratify=lab
els,     shuffle=True, test_size=size)
            print("Training data: {} {}".format(train_x.shape, train_labels.shape))
            print("Validation data: {} {}".format(validation_x.shape,validation_labels.shape))
            return train_x, train_labels, validation_x, validation_labels
        else:
            return train_data['id_code'], train_data['diagnosis'], [], []
    except:
        print("Error: Invalid file format, Function argument requires .csv file!!!")
train_x, train_labels, validation_x, validation_labels = splitting_data(train_data, split_size) # function
calling
```

```
train = pd.read_csv("/content/drive/MyDrive/DR-Data-Blindness_Detection/training.csv")
validation = pd.read_csv("/content/drive/MyDrive/DR-Data-Blindness_Detection/validation.csv")
train_x = train['id_code']
train_labels = train['diagnosis']
validation_x = validation['id_code']
validation_labels = validation['diagnosis']
#Target Labels Analysis on Train and Validation sets:
def class_analysis(labels, d_set):
    if d_set == 'training': print("-"*100,'\n')
    counter = labels.value_counts().sort_index()
    counter.plot(kind = 'bar')
    plt.title('Number of images for each Class label in {} set'.format(d_set))
    plt.xlabel('Classes')
    plt.ylabel('Number of images')
    plt.grid()
    plt.show()
    iter=0

    for i in list(set(labels)):
        percentage = list(labels).count(i)/len(list(labels))
        print("Number of images in class -
 {} ({}) , nearly {} % of total data".format(i,class_labels[i],np.round(percentage*100,4)))
        iter+=1
    if d_set == 'training':
        print("\n","="*100,"\n")
    if d_set == 'validation': print("-"*100,'\n')
class_analysis(train_labels,'training')
class_analysis(validation_labels,'validation')
#Image Processing:
class ImageProcessing:
    def __init__(self, img_height, img_width, no_channels, tol=7, sigmaX=8):
        self.img_height = img_height
        self.img_width = img_width
        self.no_channels = no_channels
```

```
    self.tol = tol
    self.sigmaX = sigmaX

def cropping_2D(self, img, is_cropping = False):

    '''This function is used for Cropping the extra dark part of the GRAY images'''

    mask = img>self.tol
    return img[np.ix_(mask.any(1),mask.any(0))]

def cropping_3D(self, img, is_cropping = False):

    '''This function is used for Cropping the extra dark part of the RGB images'''

    gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    mask = gray_img>self.tol

    check_shape = img[:,:,0][np.ix_(mask.any(1),mask.any(0))].shape[0] if
    (check_shape == 0): # if image is too dark we return the image
        return img
    else:
        img1 = img[:,:,0][np.ix_(mask.any(1),mask.any(0))]  #for channel_1 (R)
        img2 = img[:,:,1][np.ix_(mask.any(1),mask.any(0))]  #for channel_2 (G)
        img3 = img[:,:,2][np.ix_(mask.any(1),mask.any(0))]  #for channel_3 (B)
        img = np.stack([img1,img2,img3],axis=-1)
    return img

def Gaussian_blur(self, img, is_gaussianblur = True):
    img = cv2.addWeighted(img,4,cv2.GaussianBlur(img,(0,0),self.sigmaX),-4,128)
    return img

def draw_circle(self,img, is_drawcircle = True):
    x = int(self.img_width/2)
    y = int(self.img_height/2)
```

```
        r = np.amin((x,y)) # finding radius to draw a circle from the center of the image
        circle_img = np.zeros((img_height, img_width), np.uint8) cv2.circle(circle_img,
        (x,y), int(r), 1, thickness=-1) img = cv2.bitwise_and(img, img, mask=circle_img)

        return img


    def CLAHEgreen(self, img, is_clahe = True):
        green = img[:,:,1]
        lab = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        lab_planes = cv2.split(lab)
        gridsize=8
        clahe = cv2.createCLAHE(clipLimit=5.0,tileGridSize=(gridsize,gridsize))
        cla = clahe.apply(green)
        cla=clahe.apply(cla)
        img = cv2.merge((lab,cla,cla))
#       bgr = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
        return img

    def image_preprocessing(self, img,is_gaussianblur=True, is_cropping = True, is_clahe = True):
        if img.ndim == 2:
            img = self.cropping_2D(img, is_cropping) #calling cropping_2D for a GRAY image
        else:
            img = self.cropping_3D(img, is_cropping) #calling cropping_3D for a RGB image
        img = cv2.resize(img, (self.img_height, self.img_width)) # resizing the image img =
        self.draw_circle(img) #calling draw_circle
        img = self.Gaussian_blur(img, is_gaussianblur) #calling Gaussian_blur
        img = self.CLAHEgreen(img, is_clahe) #ClAHE return img
def images_per_class(class_labels,num, data_x , is_preprocess = False):
    # class_labels num data_x data_y
    labels = list(set(class_labels))
    classes = ['No DR','Mild','Moderate','Severe','Proliferative DR']
    iter=0
```

```
    for i in labels:
        j=1
        plt.figure(figsize=(20,5))
        for row in range(len(data_x)):
            if class_labels.iloc[row] == i:
                if is_preprocess == False:plt.subplot(1,num,j)
                else: plt.subplot(1,num*2,j)
                img = cv2.imread('/content/drive/MyDrive/DR-Data-
Blindness_Detection/train_images/'+data_x.iloc[row]+'.png')
                img1 = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
                plt.imshow(img1)
                plt.axis('off')
                plt.title("Class = {} ({})".format(class_labels.iloc[row],classes[iter]))
                j+=1
                if is_preprocess == True:
                    obj = ImageProcessing(img_width,img_height,no_channels,sigmaX=14)
                    image = obj.image_preprocessing(img)
                    plt.subplot(1,num*2,j)
                    plt.imshow(image)
                    plt.axis('off')
                    plt.title('==> After Image Processing')
                    j+=1
            if is_preprocess == False and j>num: break
            elif is_preprocess == True and j>num*2: break
        iter+=1
        plt.show()
images_per_class(train_labels,5,train_x,False) #printing 5 random images per each class.
#Exploring the Dataset
def get_histograms(df,columns=4, rows=3):
    ax, fig=plt.subplots(columns*rows,figsize=(5*columns, 4*rows))

    for i in range(columns*rows):
        image_path = df.loc[i,'id_code']
        image_id = df.loc[i,'diagnosis']
```

```
        plt.subplot(columns, rows, i+1)
        img = cv2.imread(f'/content/drive/MyDrive/DR-Data-
Blindness_Detection/train_images/{image_path}.png')
        plt.hist(img.flatten(),256,[0,256],color='r')
#         fig.add_subplot(rows, columns, i+1)
        plt.title(image_id)
#         plt.axis('off')
#         plt.imshow(img)

    plt.tight_layout()
get_histograms(train_data)
def plotting(img, title,i):
    """
    This function is used for subplots
    Args: img (numpy.ndarray) - image we need to plot
          title(string) - title of the plot
          i (integer) - column number
    output: None - this function doesn't return anything.
    """
    plt.subplot(1,6,i)
    plt.imshow(img)
    plt.axis('off')
    plt.title(title)
obj1 = ImageProcessing(img_width,img_height, no_channels, sigmaX = 14)
img = '/content/drive/MyDrive/DR-Data-
Blindness_Detection/train_images/002c21358ce6.png'  #random train image
img = cv2.imread(img)
img1 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(30,5))
plotting(img1,'Before Image Processing:',1)
img1 = obj1.cropping_3D(img1)
plotting(img1,'Step-1: (Cropping extra dark pixels)',2)
img1 = cv2.resize(img1, (img_height,img_width))
plotting(img1,'Step-2: (Resizing the image)',3)
```

```
img1 = obj1.draw_circle(img1)
plotting(img1,'Step-3: (Drawing circle)',4)
#img = obj1.Gaussian_blur(img1,True)
#plotting(img1,'Step-4: (Adding gaussian blur)',5)
img = obj1.image_preprocessing(img,True)
plotting(img,'Step-5: (CLAHE)',5)

images_per_class(train_labels,3,train_x,True)
obj1 = ImageProcessing(img_width,img_height, no_channels, sigmaX = 14)
def get_histograms_preprocess(df,columns=4, rows=3):
    ax, fig=plt.subplots(columns*rows,figsize=(5*columns, 4*rows))

    for i in range(columns*rows):
        image_path = df.loc[i,'id_code']
        image_id = df.loc[i,'diagnosis']
        plt.subplot(columns, rows, i+1)
        img = cv2.imread(f'/content/drive/MyDrive/DR-Data-
Blindness_Detection/train_images/{image_path}.png')
        img = obj1.image_preprocessing(img)
        plt.hist(img.flatten(),256,[0,256],color='r')
#         fig.add_subplot(rows, columns, i+1)
        plt.title(image_id)
#         plt.axis('off')
#         plt.imshow(img)

    plt.tight_layout()
get_histograms_preprocess(train_data)
#Image Convertion:
def image_2_vector(data, sep):
    """
    This function is used for Converting an images into a vector and storing it in a file (.npy) format.
    Input: data (Series Object) - which contains the path of the images
           sep (String)   - used in file creation
    Output: None - This function doesn't return anything.
```

```
    """
    start_time = time.time() # storing timestamp
    image_vector = np.empty((len(data),img_width, img_height, no_channels), dtype = np.uint8)
    image_processing = ImageProcessing(img_width, img_height, no_channels, sigmaX) # Object cre
ation
    if sep !='test':
        c = '/content/drive/MyDrive/DR-Data-Blindness_Detection/train_images/'
    else:
        c = '/content/drive/MyDrive/DR-Data-Blindness_Detection/test_images/'
    for iter,row in enumerate(tqdm(data)):
        img_path = c+data.iloc[iter]+'.png'
        img = cv2.imread(img_path)
        img = image_processing.image_preprocessing(img) #calling image_preprocessing
        image_vector[iter,:,:,:] = img

    if sep == 'training': print("\nShape of the vector:",image_vector.shape)
    else: print("\n\nShape of the vector:",image_vector.shape)
    print("Time taken to process the {} images: {} seconds".format(sep,np.round(time.time()-
start_time,5)))
    path = '/content/drive/My Drive/DR-Data-Blindness_Detection/processed_images3'
    print("... Saving image_vector to {}".format(path+'/'+sep))

    if sep == 'training':
        print("\n","-"*100,"\n")
    if not os.path.exists(path):
        os.makedirs(path)
    np.save(path+'/'+sep+'.npy', image_vector) #saving file
sigmaX = 14
image_2_vector(train_x, "training") # function calling
image_2_vector(validation_x,"validation") #function calling
test = pd.read_csv("/content/drive/MyDrive/DR-Data-Blindness_Detection/test.csv")
image_2_vector(test['id_code'], 'test') #function calling x_train =
np.load('/content/drive/My Drive/DR-Data-
Blindness_Detection/processed_images3/training.npy')  #training set
```

```
x_validation = np.load('/content/drive/My Drive/DR-Data-
Blindness_Detection/processed_images3/validation.npy')  #validation set
x_test = np.load('/content/drive/My Drive/DR-Data-
Blindness_Detection/processed_images3/test.npy')    #test set
plt.figure(figsize=(15,5))
plt.subplot(131)
plt.imshow(x_train[9])   #random training example
plt.axis('off')
plt.title("Training sample image")
plt.subplot(132)
plt.imshow(x_validation[111]) #random validation example
plt.title("Validation sample image") plt.axis('off')

plt.subplot(133)
plt.imshow(x_test[1200]) #random test example
plt.title("Testing sample image") plt.axis('off')

plt.show()
```

- **DR_Model-3.ipynb:**

```
# for accessing tabular data import
pandas as pd import numpy as np
import os
os.chdir('/content/drive/My Drive/')

# adding classweight
from sklearn.utils import class_weight
# Evaluation Metric
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix, precision_score, recall_score
# for visualization
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
from prettytable import PrettyTable
```

```
# backend
import keras
from keras import backend as K
import tensorflow as tf
from keras.callbacks import Callback
# for transfer learning
from keras.applications.vgg16 import VGG16
from keras.applications.vgg19 import VGG19
from keras.applications.densenet import DenseNet121 from
keras.applications.resnet import ResNet50, ResNet152 from
keras.applications.inception_v3 import InceptionV3
#from efficientnet.keras import EfficientNetB0, EfficientNetB3, EfficientNetB4
from keras.applications.xception import Xception # for model architecture
from keras.models import Sequential
from keras.layers import GlobalAveragePooling2D, Dropout, Dense, Conv2D, MaxPooling2D, Activa
tion, Flatten
# for Tensorboard visualization
from keras.callbacks import TensorBoard
# for Data Augmentation
from keras.preprocessing.image import ImageDataGenerator
import tensorflow
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from keras.applications.densenet import DenseNet121
try:
   x_train = np.load("/content/drive/MyDrive/DR-Data-
Blindness_Detection/processed_images3/training.npy", mmap_mode=None, allow_pickle=False, fix_
imports=True)
   x_validation = np.load("/content/drive/MyDrive/DR-Data-
Blindness_Detection/processed_images3/validation.npy",mmap_mode=None, allow_pickle=False, fix
_imports=True)
   x_test = np.load("/content/drive/MyDrive/DR-Data-
Blindness_Detection/processed_images3/test.npy", mmap_mode = None,allow_pickle = False, fix_im
```

```
ports = True)
    print("Loaded Successfully...\n")
    print(x_train.shape)
    print(x_validation.shape)
    print(x_test.shape)
except:
    print("file not exists")
train_labels = pd.read_csv('/content/drive/MyDrive/DR-Data-Blindness_Detection/training.csv')
train_labels = train_labels['diagnosis']
validation_labels = pd.read_csv('/content/drive/MyDrive/DR-Data-
Blindness_Detection/validation.csv')
validation_labels = validation_labels['diagnosis']
print("Training:",train_labels.shape[0])
print("Validation:",validation_labels.shape[0])
#Transforming Target Labels:
def ordinal_regression(labels):
    y_train = pd.get_dummies(labels).values
    y_train_multi = np.empty(y_train.shape, dtype=y_train.dtype)
    y_train_multi[:, 4] = y_train[:, 4]
    for i in range(3, -1, -1):
        y_train_multi[:, i] = np.logical_or(y_train[:, i], y_train_multi[:, i+1])
    return y_train_multi
labels_train = ordinal_regression(train_labels)
labels_validation = ordinal_regression(validation_labels)
print(labels_train.shape)
print(labels_validation.shape)
print("Ex: Original Category: {}\n After one_hot_encoding: {}".format(train_labels.iloc[1],labels_trai
n[1]))
labels_train
#computing class weight
class_weights = class_weight.compute_class_weight(class_weight='balanced', classes =
                                        [0,1,2,3,4],
                                        y = train_labels)
print(class_weights)
```

```
#Cohen''s Kappa
def kappa_metric(y_true, y_pred):
    y_true = y_true.sum(axis=1) - 1
    y_pred = y_pred > 0.5
    y_pred = y_pred.astype(int).sum(axis=1) - 1
    _kappa_ = cohen_kappa_score( y_true, y_pred,  weights='quadratic' )
    return _kappa_
class Metrics(Callback):
    def __init__(self, path, validation_data=()):
        super(Callback, self).__init__()
        self.path = path
        self.X_val, self.y_val = validation_data
    def ModelCheckPoint(self, path):
        self.model.save(path)
    def on_train_begin(self, logs={}):
        self.val_kappas = []
    def on_epoch_end(self, epoch, logs={}):
        y_val = self.y_val.sum(axis=1) - 1
        y_pred = self.model.predict(self.X_val) > 0.5
        y_pred = y_pred.astype(int).sum(axis=1) - 1
        _val_kappa = cohen_kappa_score(
            y_val,
            y_pred,
            weights='quadratic'
        )
        self.val_kappas.append(_val_kappa)
        print(f"\b - val_kappa:  {_val_kappa:.4f}")
        if _val_kappa == max(self.val_kappas):
            print("\n\t\tValidation Kappa has improved. Saving model to {}...".format(self.path))
            self.ModelCheckPoint(self.path)
        else:
            print("\n\t\tValidation kappa did not improved from {}".format(max(self.val_kappas)))
        return
```

```
#Evaluation
class PerformanceMetric:
    def __init__(self,actual_labels,predicted_labels):
        ''' Initialization of variables '''
        self.actual_labels = actual_labels
        self.predicted_labels = predicted_labels
    def single_value_conversion(self):
        predicted_labels = self.predicted_labels > 0.5
        prediction_ordinal = np.empty(predicted_labels.shape, dtype = int)
        prediction_ordinal[:,4] = predicted_labels[:,4]
        for i in range(3, -1, -
1): prediction_ordinal[:, i] = np.logical_or(predicted_labels[:,i], prediction_ordinal[:,i+1])
        self.predicted_labels = prediction_ordinal.sum(axis = 1)-1
        self.actual_labels = self.actual_labels.sum(axis = 1)-1
    def confusionMatrix(self):
        confusion_matrix_ = confusion_matrix(self.actual_labels, self.predicted_labels)
        return confusion_matrix_
    def precision(self, matrix):
        precision_matrix =(((matrix.T)/(matrix.sum(axis=1))).T)
        return precision_matrix
    def recall(self, matrix):
        recall_matrix =(matrix/matrix.sum(axis=0))
        return recall_matrix
    def subplot_(self, matrix, i, title):
        plt.subplot(1,3,i)
        labels = [1,2,3,4,5]
        sns.heatmap(matrix, annot=True, cmap=sns.light_palette('green'),linewidths = 0.8,cbar = False, f
mt=".3f", xticklabels=labels, yticklabels=labels)
        plt.title(title)
        plt.xlabel('Predicted Class Labels')
        plt.ylabel('Actual Class Labels')
    def plotting(self):
        self.single_value_conversion()
        confusion_matrix = self.confusionMatrix()
```

```
        np.trace(confusion_matrix))/len(self.actual_labels)*100,"\n")
        precision_matrix = self.precision(confusion_matrix)
        recall_matrix = self.recall(confusion_matrix)
        plt.figure(figsize=(20,5))
        self.subplot_(confusion_matrix, 1, 'Confusion Matrix')
        self.subplot_(precision_matrix, 2, 'Precision')
        self.subplot_(recall_matrix, 3, 'Recall')
        plt.show()
def plotting(iter, val_kappa):
    epoch = [i for i in range(iter)]
    plt.plot(epoch,val_kappa)
    plt.title('validation_kappa on each epoch')
    plt.xlabel('epoch')
    plt.ylabel('val_kappa')
    plt.grid()
    plt.show()
def test_prediction(predicted_labels):
    predicted_labels = predicted_labels > 0.5
    prediction_ordinal = np.empty(predicted_labels.shape, dtype = int)
    prediction_ordinal[:,4] = predicted_labels[:,4]
    for i in range(3, -1, -
1): prediction_ordinal[:, i] = np.logical_or(predicted_labels[:,i], prediction_ordinal[:,i+1])
    predicted_labels = prediction_ordinal.sum(axis = 1)-1
    return predicted_labels
#Data Augmentation with Modeling:
def GAP2D():
    '''Global average pooling layer'''
    global_average_pooling = GlobalAveragePooling2D()
    return global_average_pooling
def dropout(value = 0.5):
    dropout_layer = Dropout(value)
    return dropout_layer
def dense():
    dense_layer = Dense(5, activation='sigmoid')
```

```
    return dense_layer
global_average_pooling_layer = GAP2D()
dropout_layer = dropout()
dense_layer = dense()

#VGG16
def VGG16_(top_6 = False):
    '''This function is used for building a model architecture of pretrained vgg16 on imagenet data set.'''
    vgg = VGG16(weights = 'imagenet', include_top = False, input_shape = (256,256,3)) if not top_6:
        for layer in vgg.layers:
            layer.trainable = False
    else:
         for layer in vgg.layers[:13]:
            layer.trainable=False
    x = global_average_pooling_layer(vgg.layers[-1].output)
    x = dropout_layer(x)
    output = dense_layer(x)
    model = Model(vgg.layers[0].input,output)
    model.compile(loss='binary_crossentropy', optimizer=tensorflow.keras.optimizers.Adam(lr=0.0000
5), metrics=['accuracy'])
    return model
vgg16 = VGG16_(True)
vgg16.summary()
tensorboard = TensorBoard(log_dir = '/content/drive/My Drive/models/vgg16')
kappa_metrics = Metrics('/content/drive/MyDrive/DR-Data-
Blindness_Detection/Models/vgg16.h5',validation_data=(x_validation, labels_validation))
data_generator =
ImageDataGenerator(horizontal_flip=True,vertical_flip=True,rotation_range=180,zo om_range = 0.2)
vgg16 = VGG16_(top_6 = True)
history = vgg16.fit_generator(
                data_generator.flow(x_train, labels_train, batch_size=12),
                 steps_per_epoch=len(x_train) / 12, epochs=5,
```

```
                initial_epoch=0,
                verbose=1,
                validation_data=(x_validation, labels_validation),
                validation_steps=len(x_validation) / 12,
                callbacks=[kappa_metrics,tensorboard],
                class_weight = {0:0.40573664,1: 1.98216561,2: 0.73309776,3: 3.79512195,4: 2.4796812
7})
print(history.history.keys())

vgg16 = VGG16_(top_6 = True)
vgg16.load_weights("/content/drive/MyDrive/DR-Data-Blindness_Detection/Models/vgg16.h5")
result = vgg16.evaluate(x_validation,labels_validation)
y_pred = vgg16.predict(x_validation)
print("After running the model for 5 epochs we got loss = {} Accuracy = {} kappa_score = {} on vali
dation      data".format(np.round(result[0],4),np.round(result[1],4),np.round(kappa_metric(labels_validati
on,y_pred),4)))

%reload_ext tensorboard
%tensorboard --logdir='/content/drive/My Drive/models/vgg16'
metric = PerformanceMetric(labels_validation, y_pred)
metric.plotting()
ytrain_vgg16 = vgg16.predict(x_train)
ytrain_vgg16 = test_prediction(ytrain_vgg16)
print("First five data points predictions in training:",ytrain_vgg16[:5])
print("length of traindata prediction:",ytrain_vgg16.shape,"\n")

yvalidation_vgg16 = vgg16.predict(x_validation)
yvalidation_vgg16 = test_prediction(yvalidation_vgg16)
print("First five data points predictions in validation:",yvalidation_vgg16[:5])
print("length of validation data prediction:",yvalidation_vgg16.shape,"\n")

ytest_vgg16 = vgg16.predict(x_test)
ytest_vgg16 = test_prediction(ytest_vgg16)
print("First five data points predictions in test:",ytest_vgg16[:5])
```

```
print("length of test data prediction:",ytest_vgg16.shape)
#ResNet50
def ResNet50_():
'''This function is used for building a model architecture of pretrained Resnet50 on imagenet data set.'''
    resnet = ResNet50(weights='imagenet',include_top=False,layers=keras.layers,input_shape=(256,256,3))
   x = global_average_pooling_layer(resnet.layers[-1].output)
   x = dropout_layer(x)
   output = dense_layer(x)
   model = Model(resnet.layers[0].input,output)
   model.compile(loss='binary_crossentropy', optimizer= tensorflow.keras.optimizers.Adam(lr=0.00005), metrics=['accuracy'])
   return model
resnet = ResNet50_()
resnet.summary()
tensorboard = TensorBoard(log_dir = '/content/drive/My Drive/models/resnet50')
kappa_metrics = Metrics('/content/drive/MyDrive/DR-Data-Blindness_Detection/Models/resnet50.h5',validation_data=(x_validation, labels_validation))
data_generator = ImageDataGenerator(horizontal_flip=True,vertical_flip=True,rotation_range=180,zoom_range = 0.2)
resnet = ResNet50_()
history = resnet.fit_generator(
                data_generator.flow(x_train, labels_train, batch_size = 8),
                 steps_per_epoch=len(x_train) / 8, epochs=5,

                 initial_epoch=0,
                 verbose=1,
                 validation_data=(x_validation, labels_validation),
                 validation_steps=len(x_validation) / 8,
                 callbacks=[kappa_metrics,tensorboard],
              class_weight =  {0:0.40573664,1: 1.98216561,2: 0.73309776,3: 3.79512195,4: 2.47968127})
resnet = ResNet50_()
resnet.load_weights("/content/drive/MyDrive/DR-Data-Blindness_Detection/Models/resnet50.h5")
result = resnet.evaluate(x_validation,labels_validation)
```

```
y_pred = resnet.predict(x_validation)
print("After running the model for 5 epochs we got loss = {} Accuracy = {} kappa_score = {} on vali
dation data".format(np.round(result[0],4),np.round(result[1],4),np.round(kappa_metric(labels_validati
on,y_pred),4)))

metric = PerformanceMetric(labels_validation, y_pred)
metric.plotting()
ytrain_resnet = resnet.predict(x_train)
ytrain_resnet = test_prediction(ytrain_resnet)
print("First five data points predictions in training:",ytrain_resnet[:5])
print("length of traindata prediction:",ytrain_resnet.shape,"\n")

yvalidation_resnet = resnet.predict(x_validation)
yvalidation_resnet = test_prediction(yvalidation_resnet)
print("First five data points predictions in validation:",yvalidation_resnet[:5])
print("length of validation data prediction:",yvalidation_resnet.shape,"\n")

ytest_resnet = resnet.predict(x_test)
ytest_resnet = test_prediction(ytest_resnet)
print("First five data points predictions in test:",ytest_resnet[:5])
print("length of test data
prediction:",ytest_resnet.shape) %load_ext tensorboard
%tensorboard --logdir='/content/drive/My Drive/models/resnet50'
#DenseNet121
def DenseNet():
    '''This function is used for building a model architecture of pretrained Densenet121 on imagenet dat
a set.'''
    densenet = DenseNet121(weights='imagenet', include_top=False, input_shape=(256,256,3))
    x = global_average_pooling_layer(densenet.layers[-1].output)
    x = dropout_layer(x)
    output = dense_layer(x)
    model = Model(densenet.layers[0].input,output)
    model.compile(loss='binary_crossentropy', optimizer=tensorflow.keras.optimizers.Adam(lr=0.0000
5), metrics=['accuracy'])
```

```
    return model
densenet = DenseNet()
densenet.summary()
tensorboard = TensorBoard(log_dir = '/content/drive/My Drive/models/densenet')
kappa_metrics = Metrics('/content/drive/MyDrive/DR-Data-
Blindness_Detection/Models/densenet121.h5',validation_data=(x_validation, labels_validation))
data_generator = ImageDataGenerator(horizontal_flip=True,vertical_flip=True,rotation_range=180,zo
om_range = 0.2)
densenet = DenseNet()
history = densenet.fit_generator(
                data_generator.flow(x_train, labels_train, batch_size=8),
                 steps_per_epoch=len(x_train) / 8,
                 epochs=5,
                 initial_epoch=0,
                 verbose=1,
                 validation_data=(x_validation, labels_validation),
                 validation_steps=len(x_validation) / 8,
                 callbacks=[kappa_metrics,tensorboard],
               class_weight = {0:0.40573664,1: 1.98216561,2: 0.73309776,3: 3.79512195,4: 2.47968127})
densenet = DenseNet()
densenet.load_weights("/content/drive/MyDrive/DR-Data-
Blindness_Detection/Models/densenet121.h5")
result = densenet.evaluate(x_validation,labels_validation)
y_pred = densenet.predict(x_validation)
print("After running the model for 5 epochs we got loss = {} Accuracy = {} kappa_score = {} on vali
dation       data".format(np.round(result[0],4),np.round(result[1],4),np.round(kappa_metric(labels_validati
on,y_pred),4)))

metric = PerformanceMetric(labels_validation, y_pred)
metric.plotting()
%load_ext tensorboard
%tensorboard --logdir='/content/drive/My Drive/models/densenet'
ytrain_densenet = densenet.predict(x_train)
ytrain_densenet = test_prediction(ytrain_densenet)
```

```
print("First five data points predictions in training:",ytrain_densenet[:5])
print("length of traindata prediction:",ytrain_densenet.shape,"\n")

yvalidation_densenet = densenet.predict(x_validation)
yvalidation_densenet = test_prediction(yvalidation_densenet)
print("First five data points predictions in validation:",yvalidation_densenet[:5])
print("length of validation data prediction:",yvalidation_densenet.shape,"\n")

ytest_densenet = densenet.predict(x_test)
ytest_densenet = test_prediction(ytest_densenet)
print("First five data points predictions in test:",ytest_densenet[:5])
print("length of test data prediction:",ytest_densenet.shape)
```