DIABETIC RETINOPATHY DETECTION WITH DEEP LEARNING

LIST OF FIGURES

ABSTRACT :

Diabetic Retinopathy (DR) is a common type of diabetes mellitus, which causes ulcers in the retina that affects vision. If not seen early, it can lead to blindness. Unfortunately, DR is not a reversible problem, and treatment only supports the vision temporarily, not curing the disease entirely. Early detection of DR and treatment can significantly reduce the risk of vision loss. Approaching ophthalmologists for the treatment of Diabetic Retinopathy is often time-consuming, time-consuming, and costly. More often than not, the process is prone to misdiagnosis. The computer-aided diagnosis minimizes the chances of error and incorrect diagnosis. In recent times, using deep learning for complex problems has become the go-to approach owing to its ability to achieve better performance in many areas, especially in image processing and classification, and medical image analysis. The most widely used deep learning method in medical image analysis is Convolutional Neural Networks(CNN). CNNs have proven to be very effective. The main goal of this project is to try and reduce the time taken in the detection of Diabetic Retinopathy while achieving the maximum accuracy in the results.

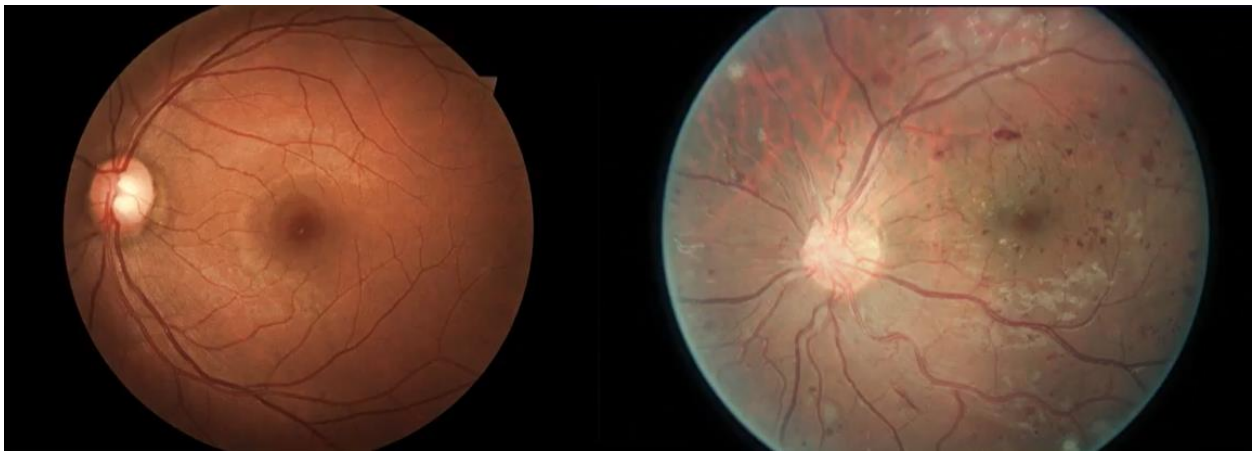

Fig 1: The left eye is the image of a healthy eye while the right eye is the image of an eye infected with DR.

TABLE OF CONTENTS

1. INTRODUCTION :

Millions of people suffer from diabetic retinopathy, the leading cause of blindness in elderly adults. Aravind Eye Hospital in India hopes to diagnose and prevent the disease in people living in rural areas where accurate and effective medical diagnosis is difficult to do. Currently, eye specialists go to these rural areas to take pictures of the retina and then rely on highly trained doctors to review the pictures and give a diagnosis. This is a very time taking process and leaves a lot of scope for human error.

In the field of health care, the treatment of diseases is most effective if they are diagnosed early. Diabetes is a disease that increases the level of glucose in the blood due to insulin deficiency. It affects 425 million adults worldwide. Diabetes affects the retina, heart, nerves, and kidneys.
Diabetic Retinopathy (DR) is a type of diabetes mellitus that causes retinal blood vessels to swell and leak fluid and blood. DR can lead to loss of vision if it is in the advanced stage. Globally, DR causes 2.6% blindness The chances of having DR increase in patients with chronic diabetes. Regular retina screening is important so that diabetic patients can diagnose and treat DR early to avoid the risk of blindness. DR is diagnosed by the appearance of different types of lesions in the retina. These lesions are microaneurysms (MA), hemorrhages (HM), soft and strong exudates (EX)

Diabetic Retinopathy (DR) is a common type of diabetes mellitus, which causes ulcers in the retina that affects vision. If not seen early, it can lead to blindness. Unfortunately, DR is not a reversible problem, and treatment only supports the vision temporarily, not curing the disease entirely. Early detection of DR and treatment can significantly reduce the risk of vision loss. Approaching ophthalmologists for the treatment of Diabetic Retinopathy is often time-consuming, time-consuming, and costly. More often than not, the process is prone to misdiagnosis. The computer-aided diagnosis minimizes the chances of error and incorrect diagnosis. In recent times, using deep learning for complex problems has become the go-to approach owing to its ability to achieve better performance in many areas, especially in image processing and classification, and medical image analysis. The most widely used deep learning method in medical image analysis is Convolutional Neural Networks(CNN). CNNs have proven to be very effective. The main goal of this project is to try and reduce the time taken in the detection of Diabetic Retinopathy while achieving the maximum accuracy in the results.

The goal here is to measure their efforts through technology; gain the ability to automatically diagnose disease images and provide information on how serious the condition can be. We will achieve this by creating a CNN that can automatically view a patient's eye image and measure the severity of blindness in a patient. This spontaneous procedure can shorten the time and thus evaluate the process of treating diabetic retinopathy on a large scale.

1.1 What is DR?

Diabetic retinopathy is a classification of eye disease. It is caused by damage to the blood vessels of the retina. Initially, diabetic retinopathy may be asymptomatic or the person may experience visual impairment without knowing the actual cause. In the long run, it can lead to permanent blindness. This condition can develop in anyone with type 1 or type 2 diabetes. If you have diabetes for a long time and your blood sugar is uncontrollable, you are more likely to get the disease.

Around 435 million people on this planet suffer from diabetes and around 94 million of them develop DR. There are two stages of diabetes as mentioned below:

- **1.1.1 NPDR (non-proliferative diabetes retinopathy):**

This is the first stage of diabetes. Many people with diabetes have it. With NPDR, small blood vessels leak, causing the retina to swell. When it is macula edema, it is called macular edema. This is the most common reason why people with diabetes undergo an impairment of vision.

And with NPDR, the blood vessels in the retina can close. This is called macular ischemia. When that happens, the blood cannot reach the macula. Sometimes tiny particles called exudates can form in the retina. This may affect one's vision too. NPDR causes a blurry vision which eventually leads to the fading of the vision.

- **1.1.2. PDR (proliferative diabetic retinopathy):**

PDR is the most advanced stage of diabetic eye disease. It occurs when the retina begins to enlarge new blood vessels. This is called neovascularization. These fragile new vessels often bleed with vitreous. As soon as a person bleeds a little, they may see a few black floats. If it bleeds too much, it may block the whole vision.

These new blood vessels can form red tissue. Red tissue can cause problems with the macula or lead to a detached retina. PDR is very sensitive and can cause major damage to vision.

1.2. Symptoms :

A person may not have symptoms in the early stages of diabetic retinopathy but when the situation worsens, they can experience the following symptoms:

- Dots or black strings floating in your vision (floating)
- Blurred vision
- Changing the view
- Dark or blank spots in your view
- Loss of vision

1.3 Suitable time to ask for medical help:

Careful management of your diabetes is the best way to prevent vision loss. If you have diabetes, consult your optometrist to have your eye checked every year - even if your vision seems right.

Having diabetes during pregnancy (gestational diabetes) or having prediabetes can increase the risk of diabetic retinopathy. If you are pregnant, your doctor may recommend further eye examinations during pregnancy.

Contact your eye doctor immediately if your vision suddenly changes or becomes blurred or fuzzy.

1.4. Diagnosis of Diabetic Retinopathy:

Eyedrops are used to enhance (enlarge) the subject's eye. This allows the eye specialist to look through a special lens to see the inside of the eye. The doctor may perform optical coherence tomography (OCT) to examine the retina. The machine examines the retina and provides detailed images of its durability. This helps the doctor to diagnose and evaluate your macular degeneration.

Fluorescein angiography or OCT angiography helps the doctor determine what is happening to the blood vessels in the retina. Fluorescein angiography uses a yellow dye called fluorescein, which is injected into a vein (usually on the arm). The dye travels through your bloodstream. A special camera captures images of the retina as the dye travels through all the blood vessels. This indicates that any blood vessels that are blocked or leaky are fluid. It also indicates when there are abnormal blood vessels growing. OCT angiography is a new technique and does not require dyeing to direct blood vessels.

1.5. Already existing treatment options:

- **1.5.1 Medical control :**
  Controlling your blood sugar and blood pressure can stop vision loss. Carefully follow the diet recommended by your dietitian. Take the medicine prescribed by your diabetes doctor. Sometimes, good sugar control can restore your vision. Controlling your blood pressure keeps your eye blood vessels healthy.

- **1.5.2 The tree :**
  Another type of medication is called anti-VEGF medication. These include Avastin, Eylea, and Lucentis. Anti-VEGF medications help reduce macula inflammation, reduce vision loss and improve vision. This medicine is given by injection (gun) to the eye. Steroid therapy is another way to reduce macular inflammation. These are also given as eye injections. Your doctor will recommend how many injections you will need over time.

- **1.5.3 Laser surgery :**
  Laser surgery may be used to help close leaky blood vessels. This can reduce inflammation of the retina. Laser surgery can also help to narrow the blood vessels and prevent them from growing again. Sometimes more than one treatment is needed.

- **1.5.4 Vitrectomy :**
  If you have an advanced PDR, your eye doctor may recommend a procedure called vitrectomy. Your eye doctor removes vitreous gel and blood from the leaky arteries behind your eye. This allows the light rays to focus properly on the retina as well. Red tissue may also be removed from the retina.

1.6 Motivation for problem :

According to WHO (world health organization) 412 million people were living with diabetes mellitus in 2014. In 2010, 33 percent of the people who are suffering from

diabetes are detected with diabetic retinopathy and among them one third of the people were affected with loss of vision. It is expected that number of people detected with DR may triple in 2050 particularly in America.

Diagnosis of DR requires expert knowledge and we can detect using deep learning techniques but it requires huge dataset which it lacks in healthcare and takes so much time which we can eliminate in transfer learning.

1.7 Problem statement :

Detecting the stage of Diabetic retinopathy using Fundus photograph images with help of transfer learned approach of EfficientNet-B5 model. The main objective of the project is to detect diabetic retinopathy to stop blindness before it is too late. We detect by classifying the images of retina of patient into five labels numbered from 0 to 4 where each label named as Normal, Mild DR, Moderate DR, Severe DR, Prolific DR respectively represents the Complication of the disease using Deep transfer learning and classification techniques. From these 5 stages one stages is observed as an output label for the given input fundus image

2. PROJECT OVERVIEW

2.1 Literature Review

There has been an increase in the use of digital image processing techniques DR testing after recommendation as one of the diagnostic methods DR at a DR conference in Liverpool UK in 2005. With this increase, much work has been done to improve one of the existing methods of timely testing new methods have also been introduced to really increase sensitivity as well the specification of this method. Sensitivity refers to the percentage of extraordinary fundus an image that is classified as unusual in a way while the specification can be defined as. The percentage of a typical fundus image classifies as normal. The height of these two factors improves the way. Most of the existing activities that can be done can usually be divided into categories in BDR and PDR tests while SDR diagnoses were omitted by an eye doctor.

Vallabha in their work entitled Automatic Discovery and Division of Blood vessels uncommon in diabetic retinopathy using the scale and position of the Gabor filter selected to locate and split retina images into soft or hard shapes. Scale and angle analysis was used because of its ability to distinguish images with them the beauty of its diversity in all its scales and shapes. The inserted image is filtered first through Gabor filter banks. Banks contain a number of filtered filters Scales and standing and work were done on the Fourier base. The output is then analyzed. The detection of NPDR (PDR) was performed by scoping analysis of blood vessels. The presence of a single local dimension in the power system against. more than 100 test images show the presence of mild NPDR to NPDR

(PDR) while the presence of more than one local level indicates a strong PDR. This method only indicates the presence of BDR and PDR but does not specify the links or actual locations or the exact type of disease. Clarity and sensitivity to this method have not been discussed in the work done and they do not use the full picture, instead, a portion of images of 256 x 256 pixels was used. The work done by soumya and Aditi as a way to improve tracking by Discovery and volume of retinopathy using digital angiograms proposes a four-step algorithm Compared to filtering, local entropy thresholding, length filtering, and vascular intersection detection for detection and removal of blood vessels from the retina. The blood vessel was first developed with the use of Comparative filtering, based on the assumption that blood vessels are normal to have a lower display compared to the background. length filter was then used to remove irregular pixels before the use of 3 X 3 and 11 X 11 neighboring windows to search for branch joints and road or crossovers.

The purpose of this work was to identify and separate the light lesion on its own. Fundus the image first goes through a local brightness enhancement as previously processed on stage, and then advanced C-Means (IFCM) is applied to Luminance / Chrominance (LUV) is a color space that separates all light-colored candidates. This is part of all the possible light lesions and false effects due to clusters' overlap contrast in color distribution and noise. third

2.2 Literature Survey :

- **2.2.1 The use of high-quality spectra to diagnose diabetes stages of retinopathy**.
  Separation based on feature extraction and DL were used to differentiate DR. Ku Acharya et al. The high-spectra method was used to extract the features 300 fundus images and supplied with a supporting vector separator; separated photos into 5 classes with 82% sensitivity and 88% clarity. Different algorithms are developed to extract DR lesions such as blood vessels, exudates, and Microaneurysms. Exudates were extracted for inclusion in the DR category.

- **2.2.2 Rethinking the original structure of computer vision**
  Distinguishing methods based on feature removal require expert knowledge in order to find the necessary features, and they involve a time-consuming process of feature selection, identification, and extraction. In addition, DL-based programs such as CNN's have been seen to go beyond methods based on feature release. DL DR isolation training has been done in two main phases: learning from scratch and transfer learning.

- **2.2.3 Development and validation of a comprehensive learning algorithm for acquisition**
  Diabetic retinopathy in retinal fundus images. The convolutional neural network (CNN) was trained to isolate databases 128,175 fundus photographs in 2 classes, of which the first class contains four photographs difficulty levels 0 and 1, and the second phase consists of levels 2, 3, and 4 . In a cutting-edge operation selected for high sensitivity, had a sensitivity of 97.5% as well 93.4% specification in the EyePACS-1 database containing 9963 images;

- **2.2.4 Convolutional neural networks of diabetic retinopathy**
  Using a training database of more than 70,000 fundus images, Pratt et al. trained CNN using a stochastic gradient descent algorithm to divide DR into 5 classes, and it obtained 95% specificity, 75% accuracy, and 30% sensitivity. The DL model was trained from the beginning of the MESSIDOR-2 database to automatically detect DR at, and 96.8% sensitivity and 87% specificity.

- **2.2.5 Automatic diagnosis of diabetic retinopathy using in-depth study**
  CNN was trained from the beginning to separate the fundus images from Kaggle The data collection became usable and manageable classes and received 96.2% sensitivity and 66.6% specifications. An image set of 71896 fundus images was used to train a CNN DR class also caused 90.5% sensitivity and 91.6% specificity [31]. The DL model was designed and trained on a database of 75137 fundus images once resulted in sensitivity and clear statistics of 94% and 98%, respectively.

2.3 Methodology :

The procedure follows a simpler method than existing once. The user interface makes one aware

give a fundus image of the eye. This post to model. The model initially uses gaussian blurring. This removes the presence of noise in images to be processed. The image is later filtered twice and contouring is done. After this a change in color space took place where we create a gray image of fundus. We are now applying k-means to combine to identify the region you are interested in. Then the separation took place. ROI obtained and configured for GLCM's outstanding removal. It includes a seven vector collection element consisting of seven
highlights were obtained by differentiation of retina structures and external research into entropy, segregation, homogeneity. The image below is a preview of the preparation process.
This selects the grayscale strength for each available pixel as structure. These features are then used to determine ratings. This is additionally completed with images of an accessible data set. Release section of this photos have been made and your estimates are both set of edited data
and customer input is displayed differently in correlation and selection of image validity component has. Below the reality is some kind of the proposal will be provided to the customer.
Currently, we are trying to diagnose retina syndrome through these lines a focused image is used techniques for retinal images. Basic changes which can grow in fundus by looking at illness
Exudates, Haemorrhages and Micro have aneurysms are not widely discussed in the paper. The proximity of something similar to the significant number of wounds above Fundus shows closeness to DR. Consent methods include the following two, managed and unregulated, unregulated structures The strategy is used here to diagnose hard exudate. Pre-Image Processing includes changing the images into a necessary or functional framework for collaboration
with. That is why it is called pre-processing. Directly without bat, images are resized and scanned required size, say 512 X 512 at any time required. At that point the space changes like a green scale change hop out at devotee image on dim scale. Subsequent proofreading procedures are then used to edit the image correctly. Center cleaning is used reducing the confusion that exists in the image and fragmentation is used to clear veins clearly. Feature domain, It it basically covers the vector of the part used The merger includes seven elements obtained from the division of retina structures and spatial analysis. Classifier, after Complete care is performed, convolutional neural the frame is used to edit the image by setting lift a picture with a few layers to check that the display is made seamlessly by the actual model the best way to set photos separately. This stage eventually returns no matter what the illness to be found in a poor person. Image user interface, This is obviously the front end, where the user you have requested to upload a fundus image at the click of a button provided by the GUI respectively.

3. SYSTEM REQUIREMENTS AND DATASET

3.1 Software and Hardware Requirements :

- **Software Requirements**: Operating system - Windows 11, Windows 10 or Linux including Ubuntu, Kali etc
- **System architecture:** Windows- 64-bit x86, 32-bit x86; macOS- 64-bit x86 & Apple M1 (ARM64); Linux- 64-bit x86, 64-bit aarch64 etc.

- **Programming Language** - Python 3.6.
- **Library & modules used** - NumPy ,Pandas ,Tensorflow ,PIL ,Keras,Cuda.
- **Platform used** - Jupyter , Tensorboard .
- **Hardware Requirements**. Intel Core i7 Octa Core Edition. 96 GB DDR4 RAM. Nvidia 1080 Ti

3.2 Dataset Used :

There are many publicly available datasets containing retina data which can be used to detect DR and to detect the vessels. These datasets are often used to train, validate and test the systems and also to compare a system's performance against other systems. Fundus color images and optical coherence tomography (OCT) are types of retinal imaging. OCT images are 2-D and 3-D images of the retina taken using low-coherence light and they provide considerable information about the retina's structure and thickness, while fundus images are 2-D images of the retina taken using reflected light . OCT retinal images have been introduced in past few years. There is a diversity of publicly available fundus image datasets that are commonly used.

The one we have used is: Kaggle : It contains 88,702 high-resolution images with various resolutions, ranging from 433 × 289 pixels to 5184 × 3456 pixels, collected from different cameras. All images are classified into five DR stages. Only training images ground truths are publicly available. Kaggle contains many images with poor quality and incorrect labeling

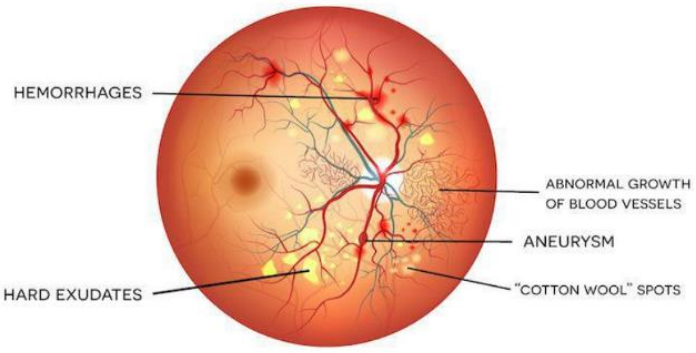

Fig 2 : An image of a retina indicating the major vessels of the eye

3.3 System Architecture :

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. Transfer learning is an optimization that allows rapid progress or improved performance when modeling the second task.

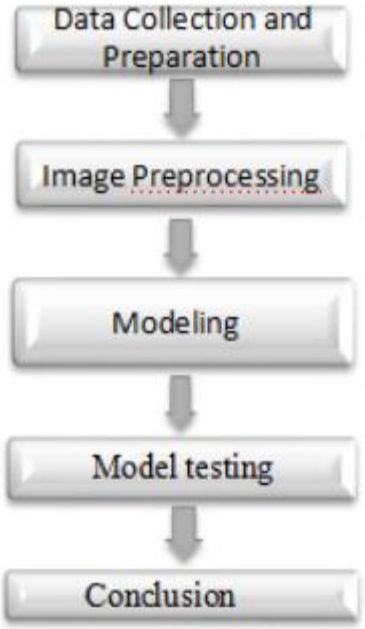

Fig 3: System Architecture

4. TRANSFER LEARNING, AUTO ML AND EFFICIENTNET ARCHITECTURE :

4.1 Introduction to Transfer Learning:

Humans have an inherent ability to transfer knowledge across tasks. What we acquire as knowledge while learning about one task, we utilize in the same way to solve related tasks. The more related the tasks, the easier it is for us to transfer, or cross-utilize our

knowledge. Some simple examples would be,

- Know how to ride a motorbike -> Learn how to ride a car
- Know how to play classic piano -> Learn how to play jazz piano
- Know math and statistics -> Learn machine learning

The first thing to remember here is that, transfer learning, is not a new concept which is very specific to deep learning. There is a stark difference between the traditional approach of building and training machine learning models, and using a methodology following transfer learning principles.

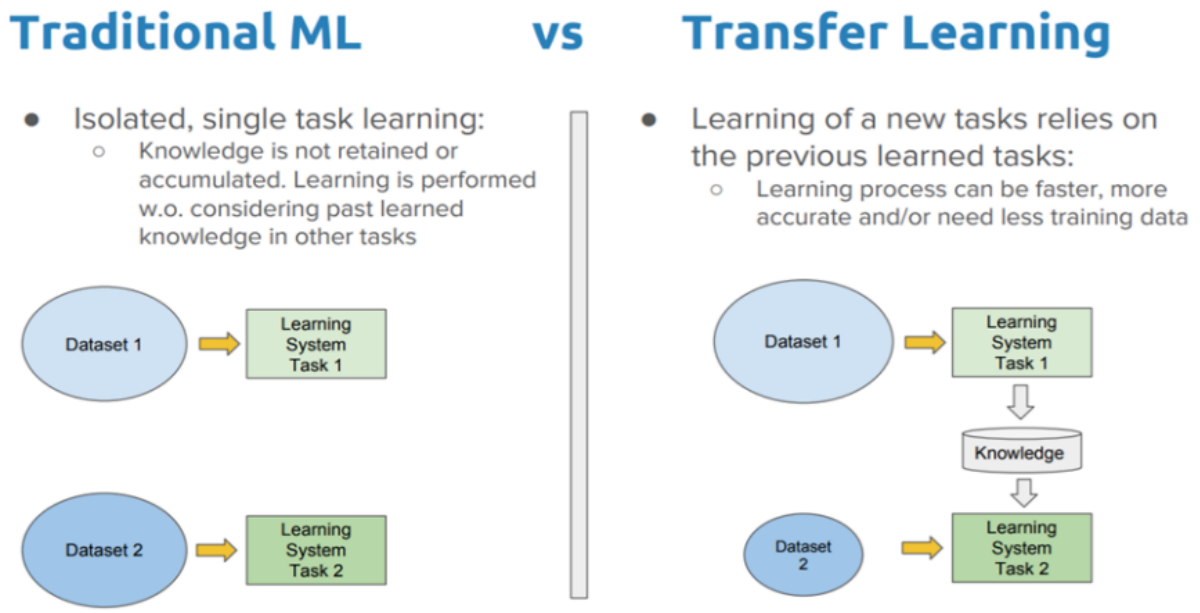

Fig 4 : Differences between traditional Machine Learning and Transfer Learning

Traditional learning is isolated and occurs purely based on specific tasks, datasets and training separate isolated models on them. No knowledge is retained which can be transferred from one model to another. In transfer learning, you can leverage knowledge (features, weights etc.) from previously trained models for training newer models and even tackle problems like having less data for the newer task.

Let's understand the preceding explanation with the help of an example. Let's assume our task is to identify objects in images within a restricted domain of a restaurant. Let's mark this task in its defined scope as T1. Given the dataset for this task, we train a model and tune it to perform well (generalize) on unseen data points from the same domain (restaurant). Traditional supervised ML algorithms break down when we do not have sufficient training examples for the required tasks in given domains. Suppose, we now must detect objects from images in a park or a cafe (say, task T2). Ideally, we should be able to apply the model trained for T1, but in reality, we face performance degradation and models that do

not generalize well. This happens for a variety of reasons, which we can liberally and collectively term as the model's bias towards training data and domain.

Transfer learning should enable us to utilize knowledge from previously learned tasks and apply them to newer, related ones. If we have significantly more data for task T1, we may utilize its learning, and generalize this knowledge (features, weights) for task T2 (which has significantly less data). In the case of problems in the computer vision domain, certain low-level features, such as edges, shapes, corners and intensity, can be shared across tasks, and thus enable knowledge transfer among tasks! Also, as we have depicted in the earlier figure, knowledge from an existing task acts as an additional input when learning a new target task.

Transfer learning is an optimization, a shortcut to saving time or getting better performance. In general, it is not obvious that there will be a benefit to using transfer learning in the domain until after the model has been developed and evaluated.

4.1.1 How to Use Transfer Learning?

You can use transfer learning on your own predictive modeling problems.
Two common approaches are as follows:

- Develop Model Approach
- Pre-trained Model Approach

- #### 4.1.1.1 Develop Model Approach :

  Select Source Task : You must select a related predictive modeling problem with an abundance of data where there is some relationship in the input data, output data, and/or concepts learned during the mapping from input to output data.

  Develop Source Model. : Next, you must develop a skillful model for this first task. The model must be better than a naive model to ensure that some feature learning has been performed.

  Reuse Model : The model fit on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.

  Tune Model.: Optionally, the model may need to be adapted or refined on the inputoutput pair data available for the task of interest.

- #### 4.1.1.2 Pre-trained Model Approach :

  Select Source Model.: A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may

be included in the pool of candidate models from which to choose from.

Reuse Model. : The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.

Tune Model. : Optionally, the model may need to be adapted or refined on the inputoutput pair data available for the task of interest.

4.2 AutoML :

CNNs have been widely used in image classification, face recognition, object detection and many other domains. Unfortunately, designing CNNs for mobile devices is challenging because mobile models need to be small and fast, yet still accurate. Although significant effort has been made to design and improve mobile models, such as MobileNet and MobileNetV2, manually creating efficient models remains challenging when there are so many possibilities to consider. Inspired by recent progress in AutoML neural architecture

search, we wondered if the design of mobile CNN models could also benefit from an AutoML approach.

The overall flow of our approach consists mainly of three components: a RNN-based controller for learning and sampling model architectures, a trainer that builds and trains models to obtain the accuracy, and an inference engine for measuring the model speed on real mobile phones using TensorFlow Lite. We formulate a multi-objective optimization problem that aims to achieve both high accuracy and high speed and utilize a reinforcement learning algorithm with a customized reward function to find Pareto optimal solutions (e.g., models that have the highest accuracy without worsening speed).

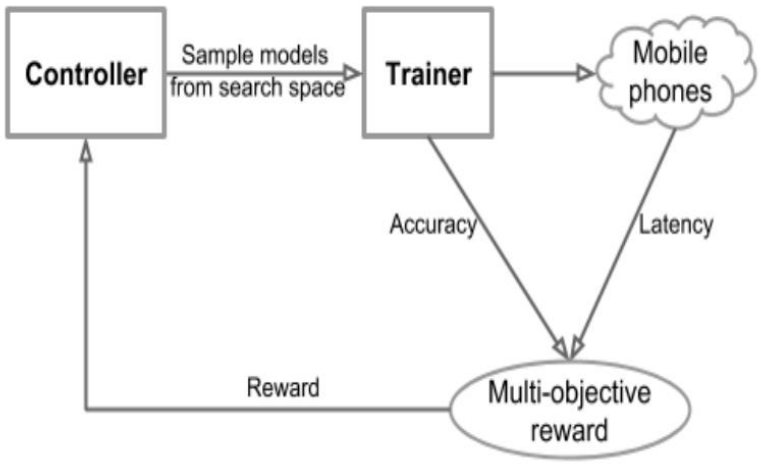

Fig 5: Overall flow of our automated neural architecture

4.3 EfficientNet Architecture :

The effectiveness of model scaling also relies heavily on the baseline network. So, to further improve performance, we have also developed a new baseline network by performing a neural architecture search using the AutoML MNAS framework, which optimizes both accuracy and efficiency (FLOPS). The resulting architecture uses mobile inverted bottleneck convolution (MBConv), similar to MobileNetV2 and MnasNet, but is slightly larger due to an increased FLOP budget.

We then scale up the baseline network to obtain a family of models, called EfficientNets. The architecture for our baseline network EfficientNet-B0 is simple and clean, making it easier to scale and generalize. Below is the architecture of EfficientNet-b5 architecture

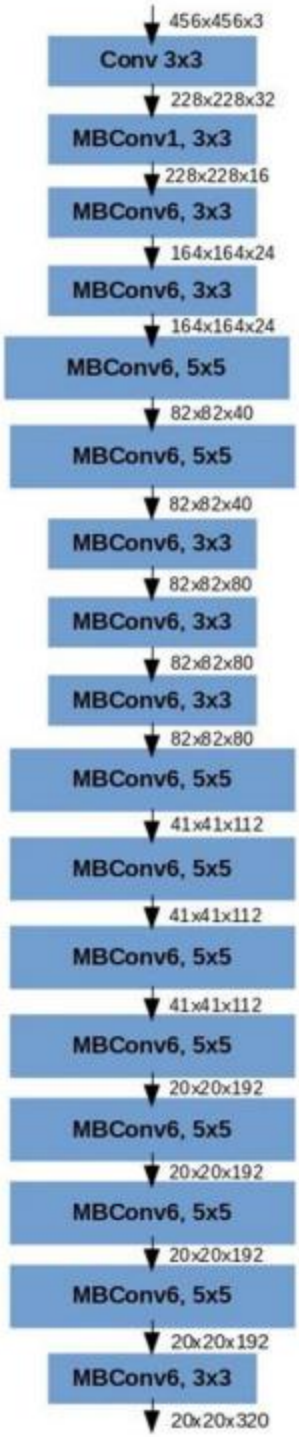

Fig 6: EfficientNet-b5 architecture

4.3.1 EfficientNet Performance :

We have compared our EfficientNets with other existing CNNs on ImageNet. In general, the EfficientNet models achieve both higher accuracy and better efficiency over existing CNNs, reducing parameter size and FLOPS by an order of magnitude.

For example, in the high-accuracy regime, our EfficientNet-B7 reaches state-of-the-art 84.4% top-1 / 97.1% top-5 accuracy on ImageNet, while being 8.4x smaller and 6.1x faster on CPU inference than the previous Gpipe. Compared with the widely used ResNet-50, our EfficientNet-B4 uses similar FLOPS, while improving the top-1 accuracy from 76.3% of ResNet-50 to 82.6% (+6.3%).

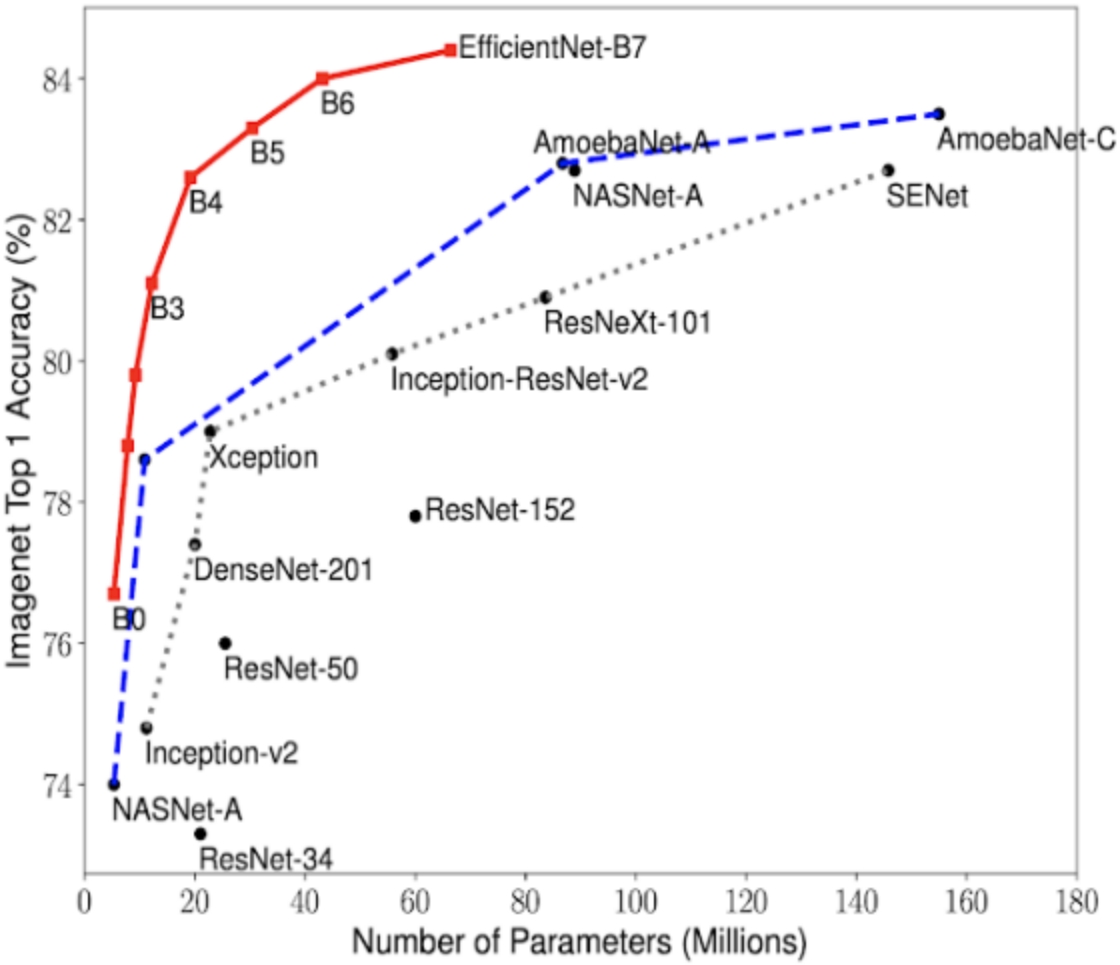
EfficientNet-B7
B6
B5
B4
B3
B0
AmoebaNet-A
AmoebaNet-C
NASNet-A
SENet
ResNeXt-101
Inception-ResNet-v2
Xception
ResNet-152
DenseNet-201
ResNet-50
Inception-v2
NASNet-A
ResNet-34
Imagenet Top 1 Accuracy (%)
Number of Parameters (Millions)
84
82
80
78
76
74
0
20
40
60
80
100
120
140
160
180

Fig 7: Graph Representation of the EfficientNet performance

5. IMAGE PREPROCESSING :

One intuitive way to improve the performance of our model is to simply improve the quality of input images. In this kernel, we will share two ideas which we hope may be useful to some of you:

- **Reducing lighting-condition effects:** images come with many different lighting conditions, some images are very dark and difficult to visualize. We can try to convert the image to gray scale, and visualize better.
- **Cropping uninformative area:** images often have areas which are out of focus and do not contribute towards are input information. These are the areas that can be cropped out.

To avoid this color distraction, we convert this original image (BGR format) to RGB format. So, we can crop the images for uninformative area. After cropping the images, they are converted into gray scale to resize the images and detect in which stage it is.

5.1 Image Processing and Feature Extraction

This is a very important step of the project as the texture obtained will be regarded as neural networking material dividing images into categories.

- **5.1.1 Image compression** :
  As one sees there are different types of images in the database with different layouts, different camera quality, and different sizes My job is to classify them into different categories. So the first problem I faced was related to database variability. With this, I have compressed all my training and testing images in 256 * 256 format.

- **5.1.2 Separating layers :**
  In the next sections, we will use 6 features such as inclusion in the category that includes the red parameter layer, the blue parameter layer, the green parameter layer, the red area layer, the green layer, the green layer in this step all 3 layers. Green Shows Separate Photos.

- **5.1.3 Measurement** :
  After the last step, there is a big difference in the intensity of the image and one can see that the veins and other features of the eyes are not clearly visible there. To make the variation of intensity I used histogram measurements in the image. A histogram Equation is a way of identifying the various variations of strength in a given image and increasing its global brightness.

- **5.1.4 Morphological functions** :
  In this section, various morphological functions are used to improve blood vessels and remove noise in the back. I have used the proposed method (use quote here) to improve the required features. Vascular fractures are a major cause of DR disease. It is therefore important to remove them and separate them from the background and remove the background noise as much as possible.

- **5.1.5 Feature Release :**
  This is the final step in processing a project image. In this step I will first remove the perimeter on all three layers and then remove the three layer layers. Discovery of Canny's Edge In this step we proceed to find the perimeters of all 3 layers. This is done with the discovery of a canny edge. At the edges of the canny gaussian filters are applied and a double limit of part of the intensity change is obtained.Thresholding This step is used for motifs that provide space for 3 layers. This is done by adaptive thresholding.

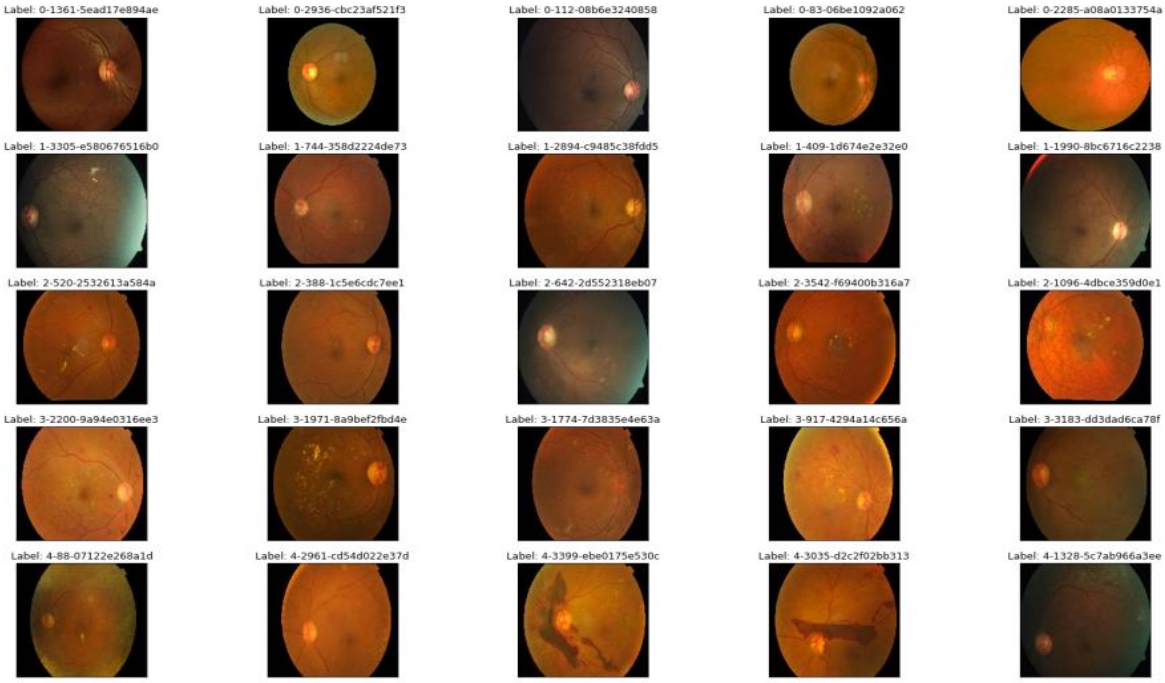

Fig 8: Original image inputs from the dataset

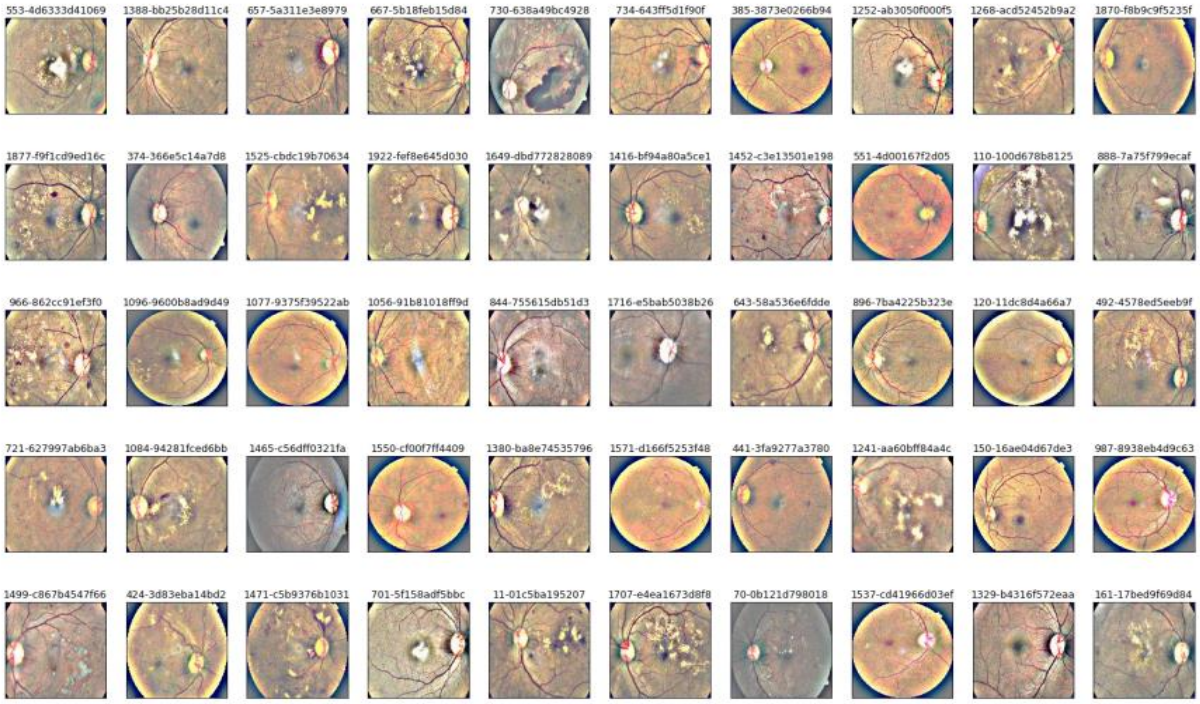

Fig 9: After cropping the images

After preprocessing we have managed to enhance the distinctive features in the images. This will increase performance when we train our Efficient Net model. Jupyter notebook is used for preprocessing.

Fig 9: After preprocessing the images

6. MODELING AND EVALUATION :

6.1 Metric (Quadratic Weighted Kappa) :

The Quadratic Weighted Kappa is used to calculate the similarity between the actuals and predictions. A perfect score of 1.0 is granted when both the predictions and actuals are the same. Whereas, the least possible score is -1 which is given when the predictions are furthest away from actuals. In our case, consider all actuals were 0's and all predictions were 4's.The aim is to get as close to 1 as possible. Generally, a score of 0.6+ is considered to be a really good score. This Metric is used to know when to stop the training of the images by the model. The training model be selected based on the best metric value.

$$\text{Weighted kappa(K)} = 1 - \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}x_{ij}}{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}m_{ij}}$$

Where i= actual values; j= predicted values; k= number of labels i.e., 5

w_{ij}=Weight matrix of actuals and predicted values

x_{ij}= Confusion matrix of actuals and predicted values

m_{ij}=Expected matrix, which is calculated based on outer product of actuals and predicted vectors

This Weighted kappa is calculated as follows:

Step 1: First, an NxN matrix X is constructed, such that X (i, j) corresponds to the actual ratings i and predicted ratings j. This NxN matrix is considered as Confusion matrix.

Step 2: Construct a weighted matrix W which is calculated based on difference between the actual and predicted rating scores.

Step 3: Create two vectors, one for predictions and another for actuals, which tells us that how many values of each rating exists.

Step 4: Now, Construct Expected Matrix E which is the outer product of two vectors (prediction vector and the actual vector) calculated in step 3.

Step 5: Normalize both matrices to have same sum. Since, it is easiest to get sum to be '1', we will simply divide each matrix by its sum to normalize the data.

Step 6: Now calculate the weighted kappa as per formulae by substituting the values.

Since we want to optimize the Quadratic Weighted Kappa score, we can formulate this challenge as a regression problem. In this way we are more flexible in our optimization

and we can yield higher scores than solely optimizing for accuracy. We will optimize a pre-trained EfficientNetB5 with a few added layers. The metric that we try to optimize is the Mean Squared Error. This is the mean of squared differences between our predictions and labels, as showed in the formula below. By optimizing this metric, we are also optimizing for Quadratic Weighted Kappa if we round the predictions afterwards.

$$\text{Mean Squared Error} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2$$

Where n = number of data points

Y_i=represents observed values

$\hat{Y_i}$ = represents Predicted values

Then in the process of training model, we used the 35 epochs and we used the metrics as quadratic weighted kappa (qwk) and accuracy. If the metrics does not change up to 4 epochs, we stop the training and save the best model according to quadratic weighted kappa score. Then we load in the weights that are provided by our dataset. we will use the RAdam Optimizer since it often yields better convergence and to get the optimized output. And we use batch normalization to take input images and process the result. If the batch size increases then this speed of the RAM will be decreased. So that, better to take small batch size.

6.2 Evaluation :

To evaluate our performance, we test model on validation data and predict values from the validation generator and round them off to the corresponding integer to get valid predictions. To detect the severity level of the diabetic retinopathy of the patient from the values of generator we set threshold value for each stage of the disease During training of the model. Based on the threshold values we get accuracy of the model. we can improve the model Accuracy by optimizing the model performance using optimizers like RAdam optimizer we used in this model. We initially set initial values of coefficient values to (0.5,1.5,2.5,3.5). Later we minimize the quadratic weighted kappa Score with respect to coefficient values using neldermead method.

We evaluate the performance of the model depending on validation accuracy and quadratic weighted kappa score. since quadratic weighted kappa score plays vital role in assessing performance of the model. It further enhances using the optimizers on the quadratic weighted kappa score with respect to coefficients of the threshold values. We get 0.8712 quadratic weighted kappa score and 83% accuracy on validation data. Now train the model till the model does not improve further apply it on test data we get severity of diabetic retinopathy using fundus images.

7. SAMPLE CODE :

```
import os
import sys
sys.path.append(os.path.abspath('input/efficientnet/efficientnetmaster/efficientnet-master/'))
import cv2
import time
import scipy as sp
import numpy as np
import random as rn
import pandas as pd
from tqdm import tqdm
from PIL import Image
from functools import partial
import matplotlib.pyplot as plt
import tensorflow as tf
```

```
import os
import sys
import keras
from keras import initializers
from keras import regularizers
from keras import constraints
from keras import backend as K
from keras.activations import elu
from keras.optimizers import Adam
from keras.models import Sequential
from keras.engine import Layer, InputSpec
from keras.utils.generic_utils import get_custom_objects
from keras.callbacks import Callback, EarlyStopping, ReduceLROnPlateau
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.layers import Dense, Conv2D, Flatten, GlobalAveragePooling2D, Dropout
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import cohen_kappa_score
KAGGLE_DIR = 'APTOS/'
TRAIN_DF_PATH = KAGGLE_DIR + "train.csv"
TEST_DF_PATH = KAGGLE_DIR + 'test.csv'
TRAIN_IMG_PATH = KAGGLE_DIR + "train_images/"
TEST_IMG_PATH = KAGGLE_DIR + 'test_images/'
SAVED_MODEL_NAME = 'effnet_modelB5.h5'
seed = 1234
rn.seed(seed)
np.random.seed(seed)
tf.set_random_seed(seed)
os.environ['PYTHONHASHSEED'] = str(seed)
t_start = time.time()
print("Image IDs and Labels (TRAIN)")
train_df = pd.read_csv(TRAIN_DF_PATH)
train_df['id_code'] = train_df['id_code'] + ".png"
```

```
print(f"Training images: {train_df.shape[0]}")
display(train_df.head())
print("Image IDs (TEST)")
test_df = pd.read_csv(TEST_DF_PATH)
test_df['id_code'] = test_df['id_code'] + ".png"
print(f"Testing Images: {test_df.shape[0]}")
display(test_df.head())
IMG_WIDTH = 456
IMG_HEIGHT = 456
CHANNELS = 3


def get_preds_and_labels(model, generator):
    preds = []
    labels = []
    for _ in range(int(np.ceil(generator.samples / BATCH_SIZE))):
        x, y = next(generator)
        preds.append(model.predict(x))
        labels.append(y)
    return np.concatenate(preds).ravel(), np.concatenate(labels).ravel()
class Metrics(Callback):
        def on_train_begin(self, logs={}):
            self.val_kappas = []
        def on_epoch_end(self, epoch, logs={}):
            y_pred, labels = get_preds_and_labels(model, val_generator)
            y_pred = np.rint(y_pred).astype(np.uint8).clip(0, 4)
            _val_kappa = cohen_kappa_score(labels, y_pred, weights='quadratic')
            self.val_kappas.append(_val_kappa)
            print(f"val_kappa: {round(_val_kappa, 4)}")
            if _val_kappa == max(self.val_kappas):
                print("Validation Kappa has improved. Saving model.")
                self.model.save(SAVED_MODEL_NAME)
            return
    train_df['diagnosis'].value_counts().sort_index().plot(kind="bar",
```

```
                                    figsize=(12,5),
                                    rot=0)
plt.title("Label Distribution (Training Set)",
          weight='bold',
          fontsize=18)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel("Label", fontsize=17)
plt.ylabel("Frequency", fontsize=17);
        train_df['diagnosis'].value_counts().sort_index().plot(kind="bar",
figsize=(12,5), rot=0)

# Function for Cropping the image
def  crop_image_from_gray(img, tol=7):
    if img.ndim == 2:
        mask = img > tol
        return img[np.ix_(mask.any(1),mask.any(0))]
    elif img.ndim == 3:
        gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        mask = gray_img > tol

        check_shape = img[:,:,0][np.ix_(mask.any(1),mask.any(0))].shape[0]
        if (check_shape == 0):
            return img
        else:
            img1=img[:,:,0][np.ix_(mask.any(1),mask.any(0))]
            img2=img[:,:,1][np.ix_(mask.any(1),mask.any(0))]
            img3=img[:,:,2][np.ix_(mask.any(1),mask.any(0))]
            img = np.stack([img1,img2,img3],axis=-1)
        return img

#Preprocessing the image
def  preprocess_image(image, sigmaX=10):
```

```
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image = crop_image_from_gray(image)
        image = cv2.resize(image, (IMG_WIDTH, IMG_HEIGHT))
          image = cv2.addWeighted(image,4, cv2.GaussianBlur(image, (0,0) ,sigmaX), -
    4, 128)
        return image

fig, ax = plt.subplots(1, 5, figsize=(15, 6))
for i in range(5):
    sample = train_df[train_df['diagnosis'] == i].sample(1)
    image_name = sample['id_code'].item()
    X = preprocess_image(cv2.imread(f"{TRAIN_IMG_PATH}{image_name}"))
    ax[i].set_title(f"Image: {image_name}\n Label = {sample['diagnosis'].item()}",
                weight='bold', fontsize=10)
    ax[i].axis('off')
    ax[i].imshow(X);
BATCH_SIZE = 4

train_datagen = ImageDataGenerator(rotation_range=360,
                           horizontal_flip=True,
                           vertical_flip=True,
                           validation_split=0.15,
                           preprocessing_function=preprocess_image,
                           rescale=1 / 128.)

train_generator = train_datagen.flow_from_dataframe(train_df,
                                        x_col='id_code',
                                        y_col='diagnosis',
                                        directory = TRAIN_IMG_PATH,
                                        target_size=(IMG_WIDTH, IMG_HEIGHT),
                                        batch_size=BATCH_SIZE,
                                        class_mode='other',
                                        subset='training')
```

```
val_generator = train_datagen.flow_from_dataframe(train_df,
                                    x_col='id_code',
                                    y_col='diagnosis',
                                    directory = TRAIN_IMG_PATH,
                                    target_size=(IMG_WIDTH, IMG_HEIGHT),
                                    batch_size=BATCH_SIZE,
                                    class_mode='other',
                                    subset='validation')
class GroupNormalization(Layer):
    def __init__(self,
                 groups=32,
                 axis=-1,
                 epsilon=1e-5,
                 center=True,
                 scale=True,
                 beta_initializer='zeros',
                 gamma_initializer='ones',
                 beta_regularizer=None,
                 gamma_regularizer=None,
                 beta_constraint=None,
                 gamma_constraint=None,
                 **kwargs):
        super(GroupNormalization, self).__init__(**kwargs)
        self.supports_masking = True
        self.groups = groups
        self.axis = axis
        self.epsilon = epsilon
        self.center = center
        self.scale = scale
        self.beta_initializer = initializers.get(beta_initializer)
        self.gamma_initializer = initializers.get(gamma_initializer)
        self.beta_regularizer = regularizers.get(beta_regularizer)
```

```
        self.gamma_regularizer = regularizers.get(gamma_regularizer)
        self.beta_constraint = constraints.get(beta_constraint)
        self.gamma_constraint = constraints.get(gamma_constraint)


    def build(self, input_shape):
        dim = input_shape[self.axis]

        if dim is None:
            raise ValueError('Axis ' + str(self.axis) + ' of '
                             'input tensor should have a defined dimension '
                             'but the layer received an input with shape ' +
                             str(input_shape) + '.')

        if dim < self.groups:
            raise ValueError('Number of groups (' + str(self.groups) + ') cannot be '
                             'more than the number of channels (' +
                             str(dim) + ').')

        if dim % self.groups != 0:
            raise ValueError('Number of groups (' + str(self.groups) + ') must be a '
                             'multiple of the number of channels (' +
                             str(dim) + ').')

        self.input_spec = InputSpec(ndim=len(input_shape),
                                    axes={self.axis: dim})
        shape = (dim,)

        if self.scale:
            self.gamma = self.add_weight(shape=shape,
                                         name='gamma',
                                         initializer=self.gamma_initializer,
                                         regularizer=self.gamma_regularizer,
```

```
                                constraint=self.gamma_constraint)
        else:
            self.gamma = None
        if self.center:
            self.beta = self.add_weight(shape=shape,
                                        name='beta',
                                        initializer=self.beta_initializer,
                                        regularizer=self.beta_regularizer,
                                        constraint=self.beta_constraint)
        else:
            self.beta = None
        self.built = True

    def call(self, inputs, **kwargs):
        input_shape = K.int_shape(inputs)
        tensor_input_shape = K.shape(inputs)

    # Prepare broadcasting shape.
        reduction_axes = list(range(len(input_shape)))
        del reduction_axes[self.axis]
        broadcast_shape = [1] * len(input_shape)
        broadcast_shape[self.axis] = input_shape[self.axis] // self.groups
        broadcast_shape.insert(1, self.groups)

        reshape_group_shape = K.shape(inputs)
        group_axes = [reshape_group_shape[i] for i in range(len(input_shape))]
        group_axes[self.axis] = input_shape[self.axis] // self.groups
        group_axes.insert(1, self.groups)
        group_shape = [group_axes[0], self.groups] + group_axes[2:]
        group_shape = K.stack(group_shape)
        inputs = K.reshape(inputs, group_shape)

        group_reduction_axes = list(range(len(group_axes)))
```

```
    group_reduction_axes = group_reduction_axes[2:]

    mean = K.mean(inputs, axis=group_reduction_axes, keepdims=True)
    variance = K.var(inputs, axis=group_reduction_axes, keepdims=True)

    inputs = (inputs - mean) / (K.sqrt(variance + self.epsilon)
    inputs = K.reshape(inputs, group_shape)
    outputs = inputs

    if self.scale:
        broadcast_gamma = K.reshape(self.gamma, broadcast_shape)
        outputs = outputs * broadcast_gamma

    if self.center:
        broadcast_beta = K.reshape(self.beta, broadcast_shape)
        outputs = outputs + broadcast_beta

    outputs = K.reshape(outputs, tensor_input_shape)

    return outputs

def get_config(self):
    config = {
        'groups': self.groups,
        'axis': self.axis,
        'epsilon': self.epsilon,
        'center': self.center,
        'scale': self.scale,
        'beta_initializer': initializers.serialize(self.beta_initializer),
        'gamma_initializer': initializers.serialize(self.gamma_initializer),
        'beta_regularizer': regularizers.serialize(self.beta_regularizer),
        'gamma_regularizer': regularizers.serialize(self.gamma_regularizer),
        'beta_constraint': constraints.serialize(self.beta_constraint),
```

```
            'gamma_constraint': constraints.serialize(self.gamma_constraint)
        }
        base_config = super(GroupNormalization, self).get_config()
        return dict(list(base_config.items()) + list(config.items()))

    def compute_output_shape(self, input_shape):
        return input_shape

effnet = EfficientNetB5(weights=None,
                        include_top=False,
                        input_shape=(IMG_WIDTH, IMG_HEIGHT, CHANNELS))
effnet.load_weights('../input/efficientnet-keras-weights-b0b5/efficientnet-
b5_imagenet_1000_notop.h5')
for i, layer in enumerate(effnet.layers):
    if "batch_normalization" in layer.name:
        effnet.layers[i] = GroupNormalization(groups=32, axis=-1, epsilon=0.00001)


def build_model():

    model = Sequential()
    model.add(effnet)
    model.add(GlobalAveragePooling2D())
    model.add(Dropout(0.5))
    model.add(Dense(5, activation=elu))
    model.add(Dense(1, activation="linear"))
    model.compile(loss='mse',
                  optimizer=RAdam(lr=0.00005),
                  metrics=['mse', 'acc'])
    print(model.summary())
    return model
model = build_model()
kappa_metrics = Metrics()
```

```
# Monitor MSE to avoid overfitting and save best model
es = EarlyStopping(monitor='val_loss', mode='auto', verbose=1, patience=12)
rlr = ReduceLROnPlateau(monitor='val_loss',
                        factor=0.5,
                        patience=4,
                        verbose=1,
                        mode='auto',
                        epsilon=0.0001)

# Begin training
model.fit_generator(train_generator,
                    steps_per_epoch=train_generator.samples // BATCH_SIZE,
                    epochs=35,
                    validation_data=val_generator,
                    validation_steps = val_generator.samples // BATCH_SIZE,
                    callbacks=[kappa_metrics, es, rlr])
model.load_weights('../input/trainmodel2/effnet_original.h5')
history_df = pd.DataFrame(model.history.history)
history_df[['loss', 'val_loss']].plot(figsize=(12,5))
plt.title("Loss (MSE)", fontsize=16, weight='bold')
plt.xlabel("Epoch")
plt.ylabel("Loss (MSE)")
history_df[['acc', 'val_acc']].plot(figsize=(12,5))
plt.title("Accuracy", fontsize=16, weight='bold')
plt.xlabel("Epoch")
plt.ylabel("% Accuracy");
    y_train_preds, train_labels = get_preds_and_labels(model, train_generator)
    y_train_preds = np.rint(y_train_preds).astype(np.uint8).clip(0, 4)

# Calculate score
    train_score = cohen_kappa_score(train_labels, y_train_preds, weights="quadratic")

# Calculate QWK on validation set
```

```
    y_val_preds, val_labels = get_preds_and_labels(model, val_generator)
    y_val_preds = np.rint(y_val_preds).astype(np.uint8).clip(0, 4)

# Calculate score
    val_score = cohen_kappa_score(val_labels, y_val_preds, weights="quadratic")
    print(f"The Training Cohen Kappa Score is: {round(train_score, 5)}")
    print(f"The Validation Cohen Kappa Score is: {round(val_score, 5)}")
    class OptimizedRounder(object):

        def __init__(self):
            self.coef_ = 0

        def _kappa_loss(self, coef, X, y):
            X_p = np.copy(X)
            for i, pred in enumerate(X_p):
                if pred < coef[0]:
                    X_p[i] = 0
                elif pred >= coef[0] and pred < coef[1]:
                    X_p[i] = 1
                elif pred >= coef[1] and pred < coef[2]:
                    X_p[i] = 2
                elif pred >= coef[2] and pred < coef[3]:
                    X_p[i] = 3
                else:
                    X_p[i] = 4

            ll = cohen_kappa_score(y, X_p, weights='quadratic')
            return -ll

        def fit(self, X, y):

            loss_partial = partial(self._kappa_loss, X=X, y=y)
            initial_coef = [0.5, 1.5, 2.5, 3.5]
```

```
        self.coef_ = sp.optimize.minimize(loss_partial, initial_coef, method='nelder-
mead')

    def predict(self, X, coef):
        X_p = np.copy(X)
        for i, pred in enumerate(X_p):
            if pred < coef[0]:
                X_p[i] = 0
            elif pred >= coef[0] and pred < coef[1]:
                X_p[i] = 1
            elif pred >= coef[1] and pred < coef[2]:
                X_p[i] = 2
            elif pred >= coef[2] and pred < coef[3]:
                X_p[i] = 3
            else:
                X_p[i] = 4
        return X_p

    def coefficients(self):
        """
        Return the optimized coefficients
        """
        return self.coef_['x']

# Optimize on validation data and evaluate again
    y_val_preds, val_labels = get_preds_and_labels(model, val_generator)
    optR = OptimizedRounder()
    optR.fit(y_val_preds, val_labels)
    coefficients = optR.coefficients()
    opt_val_predictions = optR.predict(y_val_preds, coefficients)
    new_val_score = cohen_kappa_score(val_labels, opt_val_predictions,
weights="quadratic")
    test_df['diagnosis'] = np.zeros(test_df.shape[0])
```

```
# For preprocessing test images
test_generator = ImageDataGenerator(preprocessing_function=preprocess_image,
                    rescale=1 / 128.).flow_from_dataframe(test_df,
 x_col='id_code', y_col='diagnosis',directory=TEST_IMG_PATH,
 target_size=(IMG_WIDTH, IMG_HEIGHT), batch_size=BATCH_SIZE,
 class_mode='other', shuffle=False)

train_df['diagnosis'].value_counts().sort_index().plot(kind="bar",figsize=(12,5),
rot=0)
plt.title("Label Distribution (Training Set)", weight='bold',  fontsize=18)

plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel("Label", fontsize=17)
plt.ylabel("Frequency", fontsize=17);
t_finish = time.time()
total_time = round((t_finish-t_start) / 3600, 4)
print('Kernel runtime = {} hours ({} minutes)'.format(total_time,
                                    int(total_time*60)))
```

8. CONCLUSIVE REMARKS :

8.1 Sample Output :

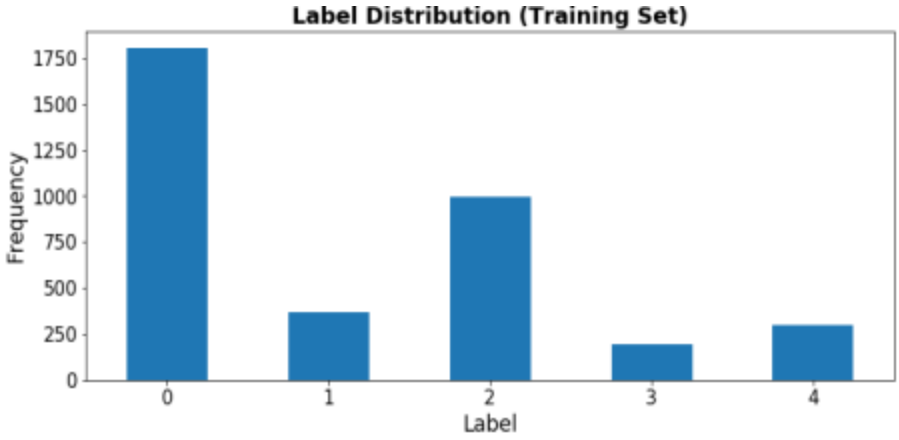

Fig 10: Label Distributing of the training set

The Figure 10 tells about distribution of the images present in the training dataset. The histogram represents the categories of the images present in the training data set. Here, The Vertical axis (y) represents Frequency of the images in training dataset and the Horizontal axis (x) represents the label (No DR-0, Mild DR-1, Moderate DR-2, Severe DR-3, Prolific DR-4) of the image.

8.2 Comparison between images before pre-processing and images after preprocessing:

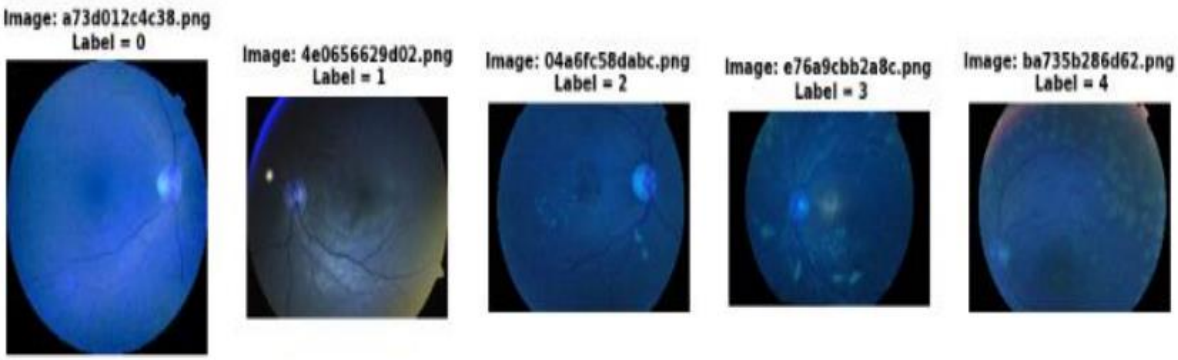

Fig 11: Images before pre-processing

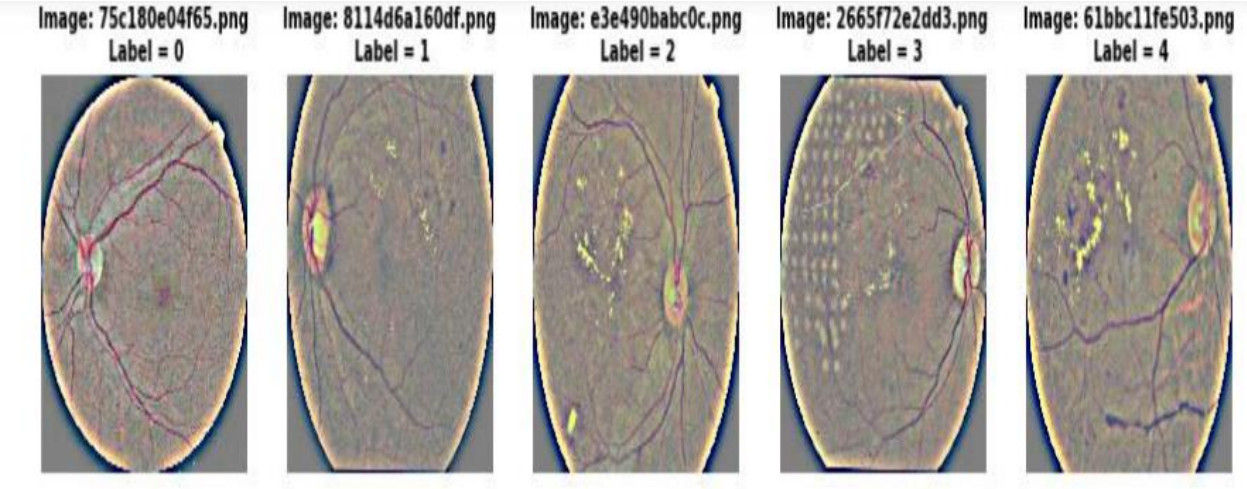

Fig 12: Images after pre-processing

We used the gaussian blurry on the image because smoothen images often suppress high frequency components that we get high frequency components of image by blurring the images. Then we subtract this blurred image from original image and adding back the difference known as mask enhance the high frequency components of image. The resulting image is fig 12

8.3 Prediction Distributions :

Here histogram of the diagram represents the number of images comes under each stage in diabetic retinopathy disease and we can see a greater number of images comes under stage-2 of Diabetic Retinopathy. The different stages of Diabetic Retinopathy are:

- 0-No DR
- 1-Mild DR
- 2-Moderate DR
- 3-Proliferative DR
- 4-Severe DR

```
test_df['diagnosis'].value_counts().sort_index().plot(kind="bar",
                                                      figsize=(12,5),
                                                      rot=0)
plt.title("Label Distribution (Predictions)",
          weight='bold',
          fontsize=18)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel("Label", fontsize=17)
plt.ylabel("Frequency", fontsize=17);
```

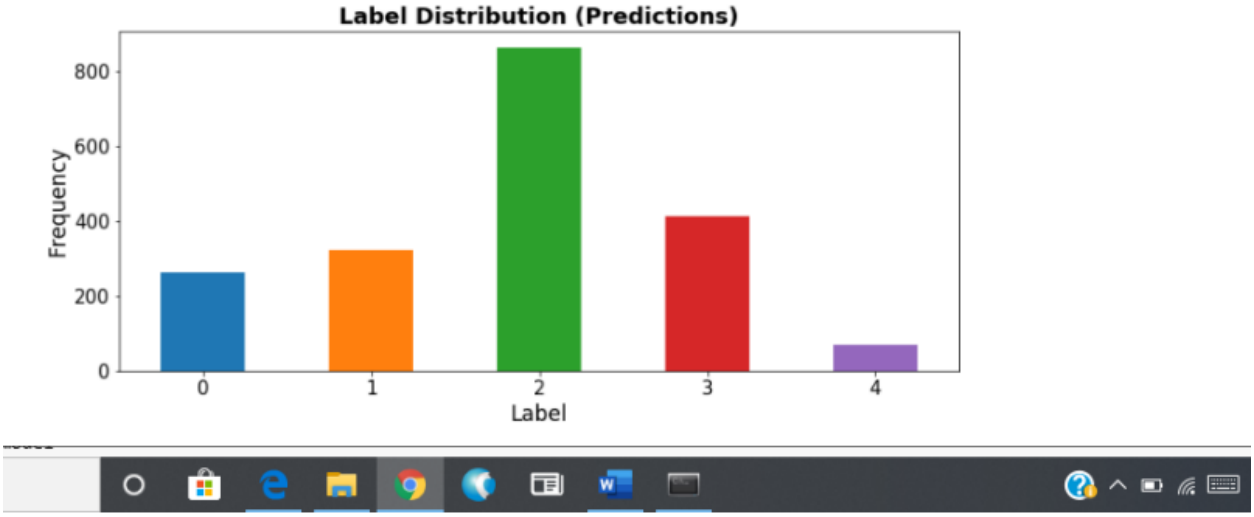

Fig 13: Label Distribution for the predictions

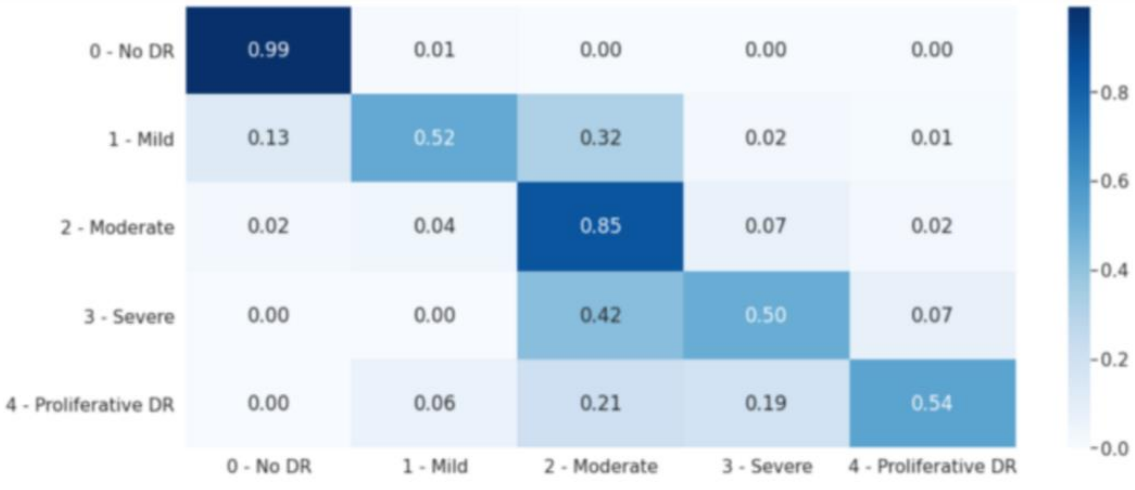

Fig 14: Normalized Matrix for the given test data depicting the accuracy of the results

9. FUTURE ENHANCEMENTS AND CONCLUSION :

9.1 Summary of work done and futue scope

The prevalence of diabetes is increasing worldwide, and the problem of DR is increasing and increased. This disease threatens the perception of patients with diabetes when DR is diagnosed final stages. Therefore, the diagnosis and treatment of DR in its early stages is important reducing the risk of blindness. DR hand diagnostic procedure is increasingly popular suffering from DR has not been effective enough. Therefore, automatic DR diagnosis using computer-assisted assessment systems (CASS) saves effort, time, and expense. Additionally, the most important point is to use CASS to avoid negative impact of eye loss. Recently, the in-depth learning method (DL) has gained high performance in segmentation and segregation. The current function provides full functionality automated diagnostic system to assist in the diagnosis of DR. Quality and balance of the data sets used to create the DR test system are very important. In the future, we aim combine multiple data sets to achieve a database balance.

Scope of the future

- Modified version can assist in detecting and diagnosing any eye-related problem.
- An affordable solution can be provided to the public to the cause of blindness in humans.

9.2 Conclusion :

In this project, transfer learning is implemented to classify DR into 5 classes with a much-reduced training data than other previous DR classification techniques employed. This was done to design a way to train a DL model that performs well on unseen data by efficiently learning from small dataset because training data is limited in healthcare. Our model has reached at an accuracy that is higher than other techniques that have used transfer learning on the whole Kaggle DR challenge dataset for multi-class classification. Our model has reached at a superior performance on account of the selected training algorithm, which is batch gradient descent with ascending learning rate, and the quadratic weighted kappa loss 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. Experiments should be done to compare performances of other pre-trained deep convolutional Networks.

REFERENCES :

[1] R. Taylor, D. Batey, handbook of retinal screening in diabetes:diagnosis and management (second ed.), John Wiley & Sons, Ltd Wiley-Blackwell (2012)

[2] V. Gulshan, L. Peng, M. Coram, Martin C. Stumpe,"Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs", JAMA The Journal of the American Medical Association, November 2016, DOI:10.1001/jama.2016.17216

[3] Rory Sayres, Ankur Taly, Ehsan Rahimy, Katy Blumer: "Using a Deep Learning Algorithm and Integrated Gradients Explanation to Assist Grading for Diabetic Retinopathy",December 2018, Ophthalmology, DOI:10.1016/j.ophtha.2018.11.016

[4] Ling Dai, Liang Wu, Huating Li, Chun Cai, Qiang Wu,"A deep learning system for detecting diabetic retinopathy across the disease spectrum",Nature Communication 2021

[5]. Wahren J, Larsson C. "C-peptide: New findings and therapeutic possibilities". Diabetes Res Clin Pract. 2015;107:309–319.

[6]. Jaiswal M, McKeon K, Comment N, Henderson J, Swanson S, Plunkett C, Nelson P, Pop-Busui R. "Association between impaired cardiovascular autonomic function and hypoglycemia in patients with type 1 diabetes". Diabetes Care. 2014;37:2616–2621.

[7]. Faust, O., Acharya, R., Ng, E.Y.K., Ng, K.H. and Suri, J.S."Algorithms for the automated detection of diabetic retinopathy using digital fundus images: a review" Journal of medical systems, 2012

[8]. Kumar, S. and Kumar, B., "Diabetic Retinopathy Detection by Extracting Area and Number of Microaneurysm from Colour Fundus Image". IEEE at the 5th International Conference on Signal Processing and Integrated Networks (SPIN), 2018, February.

[9]. Asiri, N., Hussain, M. and Abualsamh, H.A., "Deep Learning based Computer-Aided Diagnosis Systems for Diabetic Retinopathy: A Survey" arXiv preprintarXiv:1811.01238.