Deep Learning Fungus Image Analysis For Early Detection of Diabetic Retinopathy

Prepared For

Smart-interz

Artificial Intelligence Guided project

By

Aarati Rajaram Patil

D.Y.Patil Agriculture And Technical University Talsande

On

25-Jully 2025

Final Report Project

1. **INTRODUCTION**
   - Project Overview
   - Purpose
2. **LITERATURE SURVEY**
   - Existing problem
   - References

Problem Statement Definition

3. **IDEATION & PROPOSED SOLUTION**

    Empathy Map Canvas

    Ideation & Brainstorming

4. **REQUIREMENT ANALYSIS**

    Functional requirement

    Non-Functional requirements

5. **PROJECT DESIGN**

    Data Flow Diagrams & User Stories
    Solution Architecture

6. **PROJECT PLANNING & SCHEDULING**
    Technical Architecture
    Sprint Planning & Estimation
    Sprint Delivery Schedule

1. **CODING & SOLUTIONING (Explain the features added in the project along with**
    code)

Feature1
Feature2

7. **PERFORMANCE TESTING**

    Performance Metrics

8. **RESULTS**
    Output Screenshots

9. **ADVANTAGES & DISADVANTAGES**

10. **CONCLUSION**

11. **FUTURE SCOPE**

12. **APPENDIX Source Code GitHub & Project Demo Link**

1. INTRODUCTION:

Diabetic Retinopathy (DR) stands as a formidable complication of diabetes, ranking among the primary causes of vision impairment in the working-age populace. This debilitating ailment targets the retina, inflicting harm on its blood vessels due to sustained periods of elevated blood sugar levels. Timely diagnosis and prompt treatment serve as critical factors in averting vision loss or blindness. However, the manual evaluation of retinal images by ophthalmologists for DR diagnosis is laborious and subject to interpretive variations. To confront this challenge, this project centres on fabricating an automated system harnessing deep learning, aiming to assist in the accurate and efficient detection of diabetic retinopathy.

Project Overview

The project aims to pioneer an advanced deep learning model tailored for the early detection and prognosis of Diabetic Retinopathy (DR), a critical ocular complication prevalent among individuals with diabetes. Diabetic Retinopathy manifests as a progressive condition causing damage to the blood vessels in the retina due to prolonged exposure to high blood sugar levels. The timely identification and prognosis of this ailment play a pivotal role in preventing vision impairment and reducing its impact on patients' ocular health.

The primary focus revolves around leveraging cutting-edge machine learning techniques, specifically delving into deep learning architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). These models will be adeptly trained and optimized to scrutinize retinal images, extract pertinent features, and classify the various stages of Diabetic Retinopathy with a high degree of accuracy.

Key Objectives:

Model Development: Construct a robust deep learning model using CNNs and RNNs capable of efficiently analyzing retinal images to detect and classify different severity stages of Diabetic Retinopathy.

Dataset Curation: Collect and curate a diverse and comprehensive dataset of retinal images encompassing various stages and instances of Diabetic Retinopathy for training and validation purposes.

Feature Extraction: Explore and extra crucial features from retinal images, clinical data, and patient histories, emphasizing patterns and characteristics indicative of different stages of Diabetic Retinopathy.

Model Validation: Rigorously validate the developed deep learning model against established standards and benchmarks to ensure its accuracy, sensitivity, and specificity in DR diagnosis.

Deployment and Integration: Integrate the validated model into a user-friendly interface or healthcare system, empowering healthcare professionals with an automated tool for early DR detection.

Expected Impact:

The successful execution of this project will result in a sophisticated deep learning-driven diagnostic tool that assists healthcare practitioners in swiftly identifying and classifying Diabetic Retinopathy stages from retinal images. This tool's accuracy and efficiency will not only aid in timely intervention but also all aviate the work load on ophthalmologists, enabling broader accessibility to quality eye care services, particularly in screening programs for diabetic patients. Ultimately, this initiative endeavors to significantly reduce the incidence of vision loss caused by Diabetic Retinopathy and enhance patient care outcomes within the realm of diabetic ocular health.

Purpose:

The primary objective of this project is to conceive an advanced deep learning model adept at scrutinizing retinal images to detect and classify the severity stages of diabetic retinopathy. By employing machine learning algorithms, the system endeavors to support healthcare professionals in diagnosing DR in its incipient stages, allowing for timely intervention and tailored treatment strategies. This automated approach not only diminishes reliance on manual assessment but also heightens the precision and expediency of diagnosis, ultimately culminating in superior patient outcomes.

The significance of this project transcends conventional diagnostic methodologies. An accurate and automated detection system possesses the potential to revolutionize the diagnosis and management of diabetic retinopathy. By enabling early intervention, healthcare practitioners can deliver timely treatments, thereby mitigating the risk of vision loss among diabetic patients. Additionally, this initiative optimizes healthcare resources by alleviating the burden on ophthalmologists and refining the efficiency of DR screening programs, ensuring broader accessibility to quality eye care services.

2. LITERATURE SURVEY:

This Literature Survey outlines the exploration and review of existing research, methodologies, and technological landscapes pertinent to the project's objectives, encompassing both Diabetic Retinopathy diagnosis using deep learning and the development of authentication systems utilizing React.js. Feel free to expand or customize this survey with specific references or additional areas of research relevant to your project scope

Existing Problem:

The existing diagnosis of Diabetic Retinopathy faces several challenges including variability in grading due to subjectivity, dependence on experienced ophthalmologists, limited access to specialized care in remote areas, and delays in diagnosis leading to irreversible vision impairment. Additionally, the complexity and volume of retinal imaging data, along with the need for highly accurate predictions, pose significant hurdles in developing an efficient and accessible diagnostic system.

References

1 Gulshan, V., Peng, L., Coram, M., et al. (2016). Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. JAMA, 316(22), 2402–2410.

LINK: https://pubmed.ncbi.nlm.nih.gov/27898976/

overview:

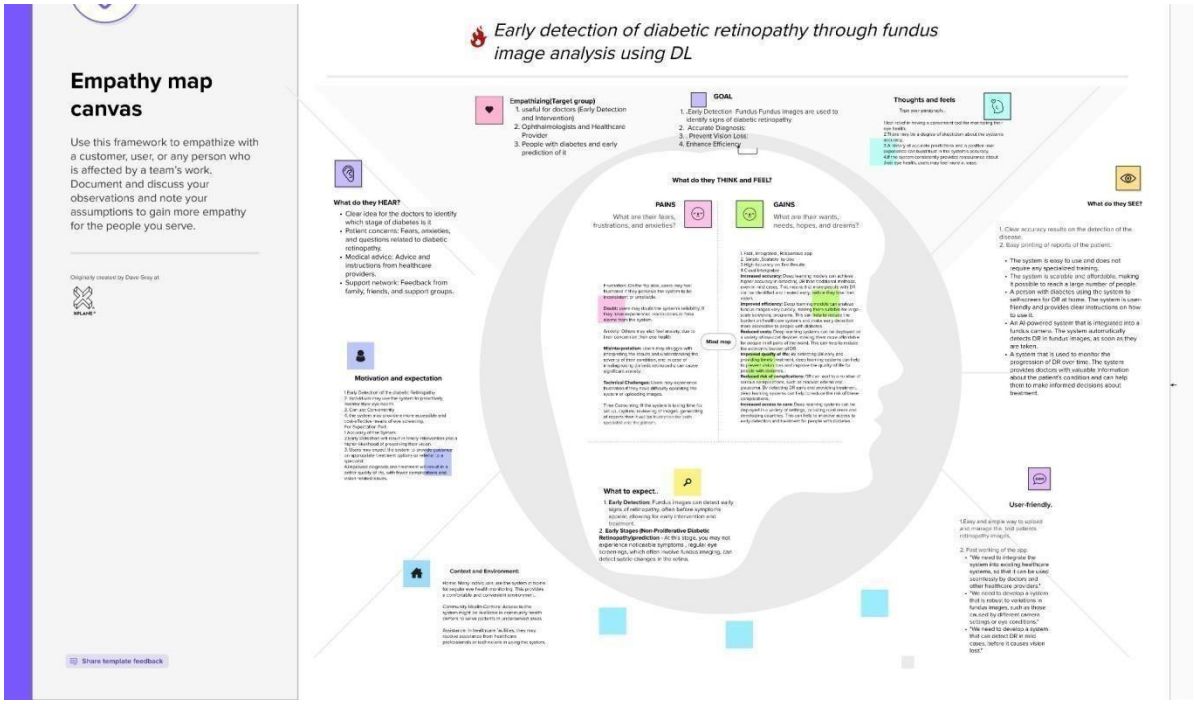
The study presented the remarkable potential of deep learning in revolutionizing the detection of diabeticretinopathyanddiabeticmacularedemathroughtheanalysisofretinal fungus photographs. By employing a specialized deep convolutional neural network, the researchers trained and validated an algorithm using an extensive dataset of retinal images, graded meticulously by ophthalmologists. This algorithm exhibited high sensitivity and specificity in detecting referable diabetic retinopathy, marking a significant stride toward automating the diagnosis of diabetic eye diseases. However, the study emphasizes the need for further research and clinical validation to ascertain the practical application and assess the real-world impact of this technology on patient care and outcomes. The research aligns with a broader landscape of studies exploring the integration of artificial intelligence and deep learning techniques in ophthalmology, underlining the collective endeavor to enhance diagnostic capabilities, particularly in diabetic retinopathy detection and management.

Problem Statement Definition:

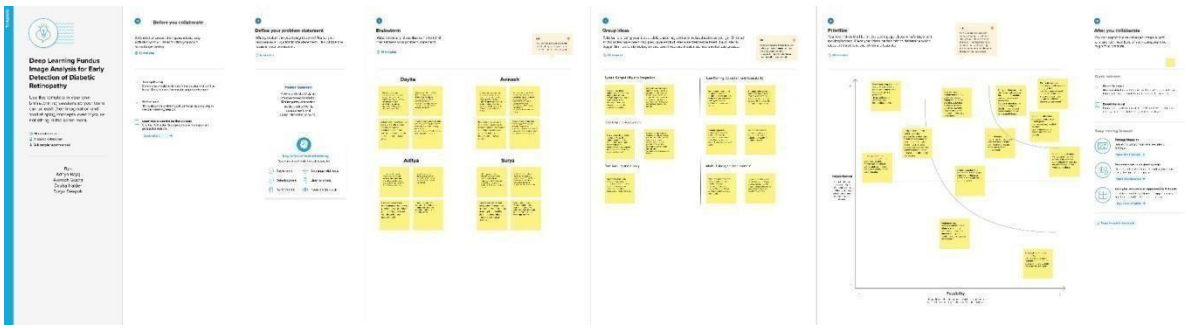
The primary goal of this project is to devise an accurate, interpretable, and scalable machine learning framework leveraging deep learning methodologies for the early detection and classification of Diabetic Retinopathy utilizing retinal images and patient data. This model aims to surmount the limitations of current diagnostic methodologies by furnishing automated and precise grading, thereby facilitating timely intervention and refining patient outcomes.

3. IDEATION & PROPOSED SOLUTION:

Empathy Map Canvas:



Ideation & Brainstorming



3. REQUIREMENT ANALYSIS

Functional Requirements:

Image Acquisition: The system must possess the capability to acquire retinal fundus images from various sources, ensuring image quality and standardization.

Data Preprocessing: It should preprocess acquired images by performing operations such as resizing, normalization, and noise reduction to optimize them for analysis.

Feature Extraction: The system needs to extract relevant features from retinal images, including blood vessel patterns, microaneurysms, exudates, and lesions.

Model Development: It should support the development and integration of deep learning models, specifically Convolutional Neural Networks (CNNs), for accurate diabetic retinopathy detection and grading.

Prediction and Classification: The system should predict and classify diabetic retinopathy severity levels based on extracted features, providing diagnoses for different stages of the disease.

Diagnostic Reports: It should generate comprehensive reports detailing the detected conditions, along with the corresponding severity levels for each patient.

Integration with Health care Systems: The system must integrate seamlessly with hospital or clinic databases to store diagnostic reports and patient information securely.

Non-Functional Requirements:

Performance: The system should deliver quick an efficient responses during image analysis and diagnosis, minimizing processing time.

Scalability: It must handle a growing number of images and patient data without compromising performance or accuracy.

Reliability: The system needs to maintain a high level of reliability, ensuring minimal errors in detection and diagnosis.

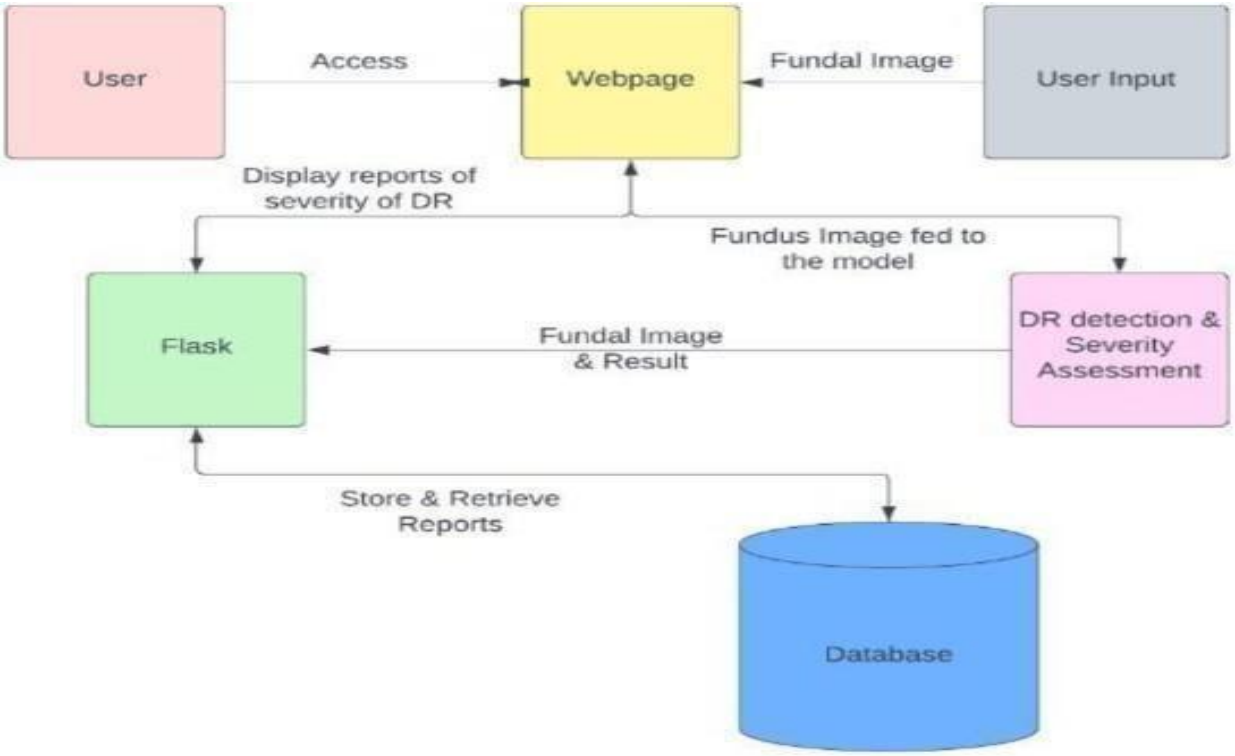
Security: Robust measures should be implemented to secure patient data, employing encryption protocols, access controls, and compliance with healthcare data regulations.

Usability: The system should have an intuitive user interface, allowing healthcare professionals to navigate and interpret results easily.

Interoperability: It should be inter operable with existing hospital information systems, enabling seamless data exchange and integration.

4.PROJECT DESIGN :

DataFlow Diagrams & User Stories



1.Users access a webpage to input their fundal image for diabetic retinopathy(DR) detection and severity assessment.

2.The input is processed, and the results along with the fundal image are sent to a Flask application.

3.The Flask app displays reports indicating the severity of DR to the users.

4.The systems tires these reports in a database for future retrieval.

5.Users can later access and retrieve their DR reports from the data base through the webpage .

User Stories:

UserType	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Release
Customer (Web user)	Dashboard	USN-1	As a user, I can access the web paget o upload my fundal image for diabetic retinopathy detection	The webpage should have an intuitive interface for uploading	Sprint-1

			and severity assessment	and submitting fundal images	
		USN-2	As a user ,I can view the severity	The Flask application should be display security.	Sprint-2
		USN-3	As user, I should be retrieve my Dr report to database.	The webpage Should provide a user-friendly interface	Sprint-3

Solution Architecture

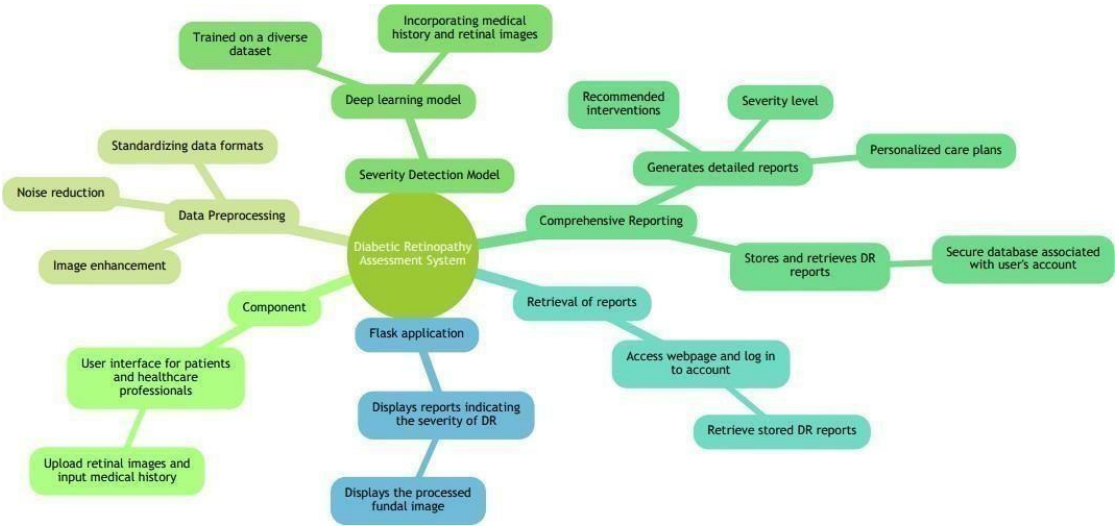
1. **User Inputs:** This component provides an intuitive user interface for patients and health care professionals to upload retinal images and input relevant medical history.

2. **Data Preprocessing:** In this phase, the system processes the uploaded data, conducting tasks such as noise reduction, image enhancement, and standardizing data formats for compatibility with the assessment algorithm.

3. **Severity Detection Model:** This core component encompasses the deep learning model specifically designed to detect the severity of diabetic retinopathy. It's trained on a diverse dataset, incorporating medical history and retinal images.

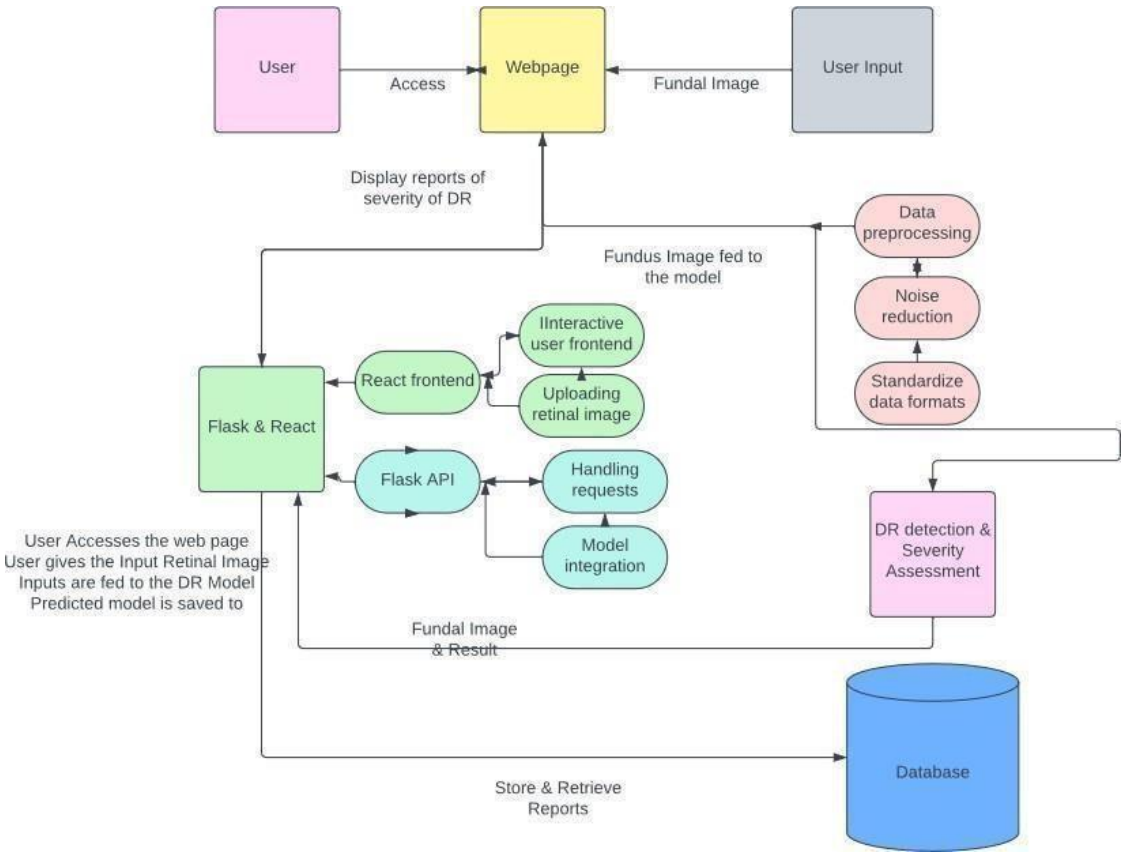
4. **Comprehensive Reporting:** Based on the severity assessment, this component generates detailed reports that include the detected severity level, recommended interventions, and personalized care plans.

User Feedback Loop: -Incorporating user feedback allows for continuous improvement of the model and system performance, ensuring it remains at the cutting edge of diabetic retinopathy assessment.



6.PROJECT PLANNING & SCHEDULING

Technical Architecture



1. Users access a webpage to input their fundal image for diabetic retinopathy (DR) detection and severity assessment.

2. The input is processed, and the results along with the fundal image are sent to a Flask application.

3. The Flask app displays reports indicating the severity of DR to the users.

4. The system stores these reports in a database for future retrieval.

5. Users can later access and retrieve their DR reports from the database through the webpage

After the user provides the input retinal image, the image is fed into the DR Model. The DR Model then analyzes the image and makes predictions based on the data it has been trained on. The predicted model is then saved for further use or reference. The Flask application displays reports indicating these verity of diabetic retinopathy (DR) to the users.

These reports are generated after processing the uploaded fundal image. Additionally, the Flask application also displays the processed fundal image along with the severity assessment report. The system stores and retrieves DR reports by utilizing a database. After the assessment of a user's fundal image, the system stores the report details in a secure database associated with the user's account. This ensures that the reports are securely stored for future reference. To retrieve the DR reports, users can access the webpage and log in to their account. The webpage provides a user-friendly interface that allows users to access and retrieve their stored DR reports from the database. This functionality enables users to conveniently review their previous reports and track the progression of their diabetic retinopathy .

Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	UserStory/Task	Story Points	Priority	TeamMembers
Sprint-1	Registration	USN-1	Asauser,Icanregisterfor the application by entering my email, password,and confirming my password.	2	High	DevelopmentTeam
Sprint-1	Registration	USN-2	Asauser,Iwillreceivea confirmationemailonceI have registered for the application.	1	High	DevelopmentTeam
Sprint-1	Registration	USN-3	Asauser,Ican register for the application through Gmail.	2	Medium	DevelopmentTeam
Sprint-1	Login	USN-4	As a user, I can log into the application by enteringemail&password.	1	High	DevelopmentTeam
Sprint-2	Data Collection and Preprocessing	USN-5	Asauser,Ican upload retinal images for analysis.	3	High	DevelopmentTeam
Sprint-3	Machine Learning Model Development	USN-6	Asauser,Ican view the progress of the Model training process.	2	High	DevelopmentTeam
Sprint-4	UserInterface Design	USN-7	As a user, I can easily navigateandinteractwith the application.	3	High	DevelopmentTeam
Sprint-5	Testing and Quality Assurance	USN-8	As a user, I can be assured that the applicationisreliable and accurate.	2	High	Quality Assurance Team

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint EndDate (Planned)	Story Points Completed (as on Planned EndDate)	Sprint Release Date (Actual)
Sprint-1	6	1Days	25-07-25	25-07-25	6	25-07-25
Sprint-2	3	1Days	25-07-25	25-07-25	3	25-07-25
Sprint-3	2	1Days	25-07-25	25-07-25	2	25-07-25
Sprint-4	3	1Days	25-07-25	25-07-25	3	25-07-25
Sprint-5	2	1Day	25-07-25	25-07-25	2	25-07-25

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day.

7. CODING & SOLUTIONING

Importing Libraries: Import the necessary libraries

```
 import numpy as np # linear algebra

importpandasaspd#dataprocessing,CSVfileI/O(e.g.

pd.read_csv) import tensorflow as tf from matplotlib import
pyplot as plt from sklearn.metrics import cohen_kappa_score
fromkeras.preprocessing.imageimportImageDataGenerator from
keras.applications.densenet import DenseNet121 import keras
import

cv2

# Input data files are available in the
"../input/"directory.
#Forexample,runningthis(byclickingrunorpressingShift+Enter)
will list the files in the input directory importcv2 import
os  from keras.callbacks import Callback
fromsklearn.model_selectionimporttrain_test_split from
sklearn.metrics import confusion_matrix
fromsklearn.utils.multiclassimportunique_labels from
sklearn.utils import class_weight
print(os.listdir("../input"))
```

Quadratic Weighted Kappa(QWK)Callback forResNet 50 Training :

This code defines a custom Keras callback (QWK Callback) to compute and monitor the Quadratic Weighted Kappa score on validation data during ResNet 50 model training, saving the model when QWK improves.

```
# borrowed from

https://www.kaggle.com/mathormad/aptosresnet50baselineclass
QWKCallback(Callback):

def init(self,validation_data): super(Callback,self).init() self.X =
validation_data[0] self.Y = validation_data[1] self.history =

[]

defon_epoch_end(self,epoch,logs={}): pred

= self.model.predict(self.X) score =

cohen_kappa_score(np.argmax(self.Y,axis=1),np.argmax(pred,axis=1),lab
els=[0,1,2,3,4],weights='quadratic')
print("Epoch{}:QWK:{}".format(epoch,score))
self.history.append(score) if score >= max(self.history):
print('savingcheckpoint:',score)
self.model.save('../working/Resnet50_bestqwk.h5')
```

We are using the APTOS - blindness dataset
https://www.kaggle.com/c/aptos2019-blindness-detection/data

We are provided with a large set of retina images taken using fundus photography under a variety of imaging conditions.

A clinician has rated each image for these verity of diabetic retinopathy on a scale of 0 to 4:

0 - No DR

1 - Mild

2 - Moderate

3 - Severe

4 -Proliferative Dr

Preprocess the Data:

The code implements a Mixup Generator class, facilitating mixup data augmentation during model training. It blends pairs of images and labels to generate augmented training batches, enhancing the model's generalization.

```
#borrowedfromhttps://github.com/yu4u/mixup-generator class
MixupGenerator():  definit(self,X_train,y_train,batch_size=32,alpha=0.2,
shuffle=True,  datagen=None):
        self.X_train = X_train self.y_train
        = y_train self.batch_size =
        batch_size self.alpha = alpha
        self.shuffle
        = shuffle self.sample_num=len(X_train)
        self.datagen = datagen

 defcall(self): while
        True:
            indexes = self.get_exploration_order()
```

```
            itr_num = int(len(indexes) // (self.batch_size * 2))


            for i in range(itr_num):
                batch_ids=indexes[i*self.batch_size*2:(i+1)*
self.batch_size * 2]
                X, y = self.data_generation(batch_ids)


                yield X, y


    def get_exploration_order(self):
        indexes=np.arange(self.sample_num)


        if self.shuffle: np.random.shuffle(indexes)


        return indexes


     def data_generation(self, batch_ids): _, h, w, c =
        self.X_train.shape
        l=np.random.beta(self.alpha,self.alpha,self.batch_size) X_l =
        l.reshape(self.batch_size, 1, 1, 1) y_l
        = l.reshape(self.batch_size, 1)


        X1=self.X_train[batch_ids[:self.batch_size]]
        X2= self.X_train[batch_ids[self.batch_size:]] X
        = X1 * X_l + X2 * (1 - X_l)


        if self.datagen:
            for i in range(self.batch_size):
```

```
                    X[i]=self.datagen.random_transform(X[i])
                    X[i] = self.datagen.standardize(X[i])

 ifisinstance(self.y_train,list): y
            = []

 for y_train_ in self.y_train:
y1=y_train_[batch_ids[:self.batch_size]]
                y2=y_train_[batch_ids[self.batch_size:]]
                y.append(y1 * y_l + y2 * (1 - y_l))
        else:
 y1=self.y_train[batch_ids[:self.batch_size]]
            y2=self.y_train[batch_ids[self.batch_size:]] y = y1 * y_l
            + y2 * (1 - y_l)

        return X, y
```

Confusion Matrix Plotting Functions

plot_confusion_matrixborrowedfromscikit-learnforvisualizingconfusionmatrices.

```
# borrowed from scikit learn

def plot_confusion_matrix(y_true, y_pred, classes,
normalize=False, title=None, cmap=plt.cm.Blues):
    """

    This function prints and plots the confusion matrix.
    Normalizationcanbeappliedbysetting`normalize=True`. """
    if not title:
```

```
    if normalize:
        title='Normalizedconfusionmatrix'else: title =
        'Confusion matrix, without normalization'

# Compute confusion matrix cm = confusion_matrix(y_true, y_pred)
# Only use the labels that appear in the data
classes=classes[unique_labels(y_true,y_pred)] if normalize:
cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis] print("Normalized
confusion matrix") else: print('Confusion matrix, without
normalization')

print(cm)

fig, ax = plt.subplots()
im=ax.imshow(cm,interpolation='nearest',cmap=cmap)
ax.figure.colorbar(im, ax=ax) # We want to show all ticks...
ax.set(xticks=np.arange(cm.shape[1]),
       yticks=np.arange(cm.shape[0]),
       #...andlabelthemwiththerespectivelistentries
       xticklabels=classes, yticklabels=classes, title=title,
       ylabel='True label', xlabel='Predictedlabel')
```

```
    # Rotate the tick labels and set their alignment.

    plt.setp(ax.get_xticklabels(),rotation=45,ha="right",

             rotation_mode="anchor")




    #Loopoverdatadimensionsandcreatetextannotations. fmt =

    '.2f'if normalize else 'd'

    thresh = cm.max() / 2.   for i in
range(cm.shape[0]):   for j in
range(cm.shape[1]):
ax.text(j,i,format(cm[i,j],fmt), ha="center",

                    va="center",

                    color="white"ifcm[i,j]>threshelse"black")

    fig.tight_layout() return ax
```

Raw Image Loading and Processing

The load _raw_ images_ defunction loads raw images from a Data Frame, resizes them, performs one-hot encoding on labels, and returns processed image data (X) and corresponding labels (Y).

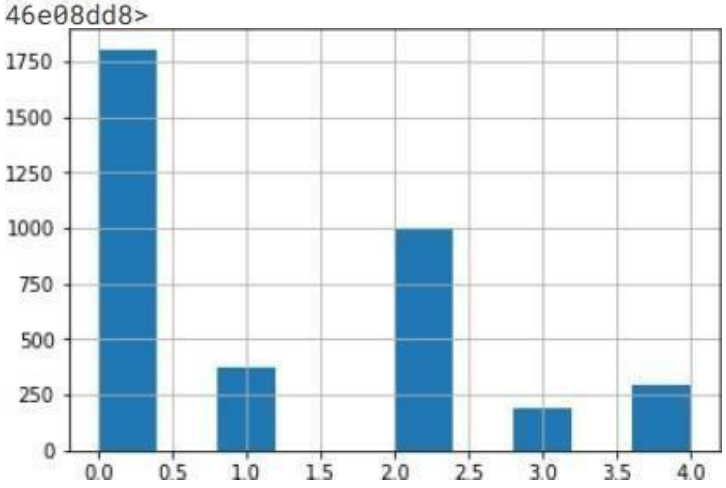
```
def
load_raw_images_df(data_frame,filenamecol,labelcol,img_size,n_classes):

    n_images = len(data_frame)

    X=np.empty((n_images,img_size,img_size,3)) Y  =

    np.zeros((n_images,n_classes)) for index,entry

    in data_frame.iterrows():

        Y[index,entry[labelcol]]=1#onehotencodingofthelabel # Load the
        image and resize img = cv2.imread(entry[filenamecol])




        X[index,:] = cv2.resize(img, (img_size, img_size))

        X[index,:]=X[index,:]/255.0 return

    X,Y
```

EDA

The code reads a CSV file containing information about the APTOS Blindness Detection dataset, appends file paths to image filenames, and visualizes the distribution of diagnosis classes using a histogram.

```
train_raw_data                                                    =
pd.read_csv("../input/aptos2019blindnessdetection/train.csv")
train_raw_data["filename"]    =    train_raw_data["id_code"].map(lambda
x:os.path.join("../input/aptos2019blindnessdetection/train_images",x+
".png")) train_raw_data.diagnosis.hist()#Seethedistributionoftheclasses

# train_raw_data.dtypes



##train_data["diagnosis"]=train_data["diagnosis"].astype(str) # #

print(train_data.head())

##print(train_data.diagnosis.unique())#Lookatdifferenttypesof classes

##labels=list(map(str,range(5))) # # print(labels)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f59
46e08dd8>
```

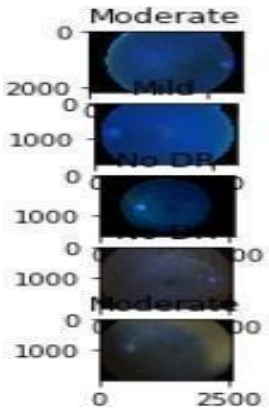


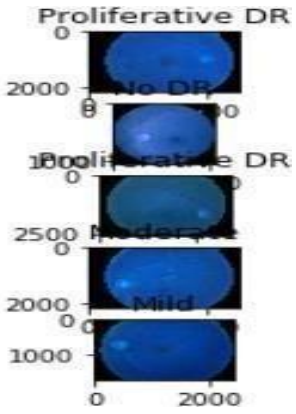
Creates a mapping between numerical labels and descriptive titles for APTOS Blindness Detection classes, providing both a dictionary (label_title) and a list (class_labels) for convenient reference .

```
label_title= {"0": "No DR","1": "Mild","2": "Moderate","3"
:"Severe","4": "Proliferative DR"}

class_labels=["No
DR","Mild","Moderate","Severe","Proliferative DR"]
```

Code creates a 5x2 grid of subplots to display images from the dataset. It iterates through the first 10 rows of the dataset, loads and displays images with their corresponding diagnosis labels as titles.

```
# Display some images

figure,ax=plt.subplots(5,2) ax

= ax.flatten()

for i,row in train_raw_data.iloc[0:10,:].iterrows():

 ax[i].imshow(cv2.imread(os.path.join("../input/aptos2019-blindness-dete
ction/train_images",row["id_code"]+".png")))
ax[i].set_title(label_title[str(row["diagnosis"])])
```





Training and Validation

```
train_df,val_df =
train_test_split(train_raw_data,random_state=42,shuffle=True,test_size=
0.333) train_df.reset_index(drop=True,inplace=True)
val_df.reset_index(drop=True,inplace=True)
```

train-validation split on then dataset, then calculates class weight using the compute_class_weight function from scikit-learn.

```
_train,Y_train =
load_raw_images_df(train_df,"filename","diagnosis",img_size,
5)

X_val,Y_val =
load_raw_images_df(val_df,"filename","diagnosis",img_size,
5)
```

```
Y_train_labels = np.argmax(Y_train,axis=1)
```

```
class_weights =
class_weight.compute_class_weight('balanced',np.unique(Y_tra
in_labels)
, Y_train_labels) cls_wt_dict=dict(enumerate(class_weights))
print(cls_wt_dict)
```

```
{0: 0.40397022332506205, 1: 1.9304347826086956,
2: 0.7333333333333333, 3: 3.7282442748091604, 4:
2.6688524590163936}
```

```
datagen = ImageDataGenerator(
zoom_range=0.15, # set range for random zoom
#setmodeforfillingpointsoutsidetheinputboundaries fill_mode='constant',
cval=0.,
          #valueusedforfill_mode="constant"horizo  ntal_flip=True,     #
        randomly flip images vertical_flip=True, # randomly flip images
)
training_generator=MixupGenerator(X_train,Y_train,
```

Build the Model:

Dense Net121-based Classification Model

The build Model function constructs a classification model based on Dense Net 121 architecture. It uses transfer learning, loading pre-trained weights and appending additional layers for classification .

```
def buildModel():

    DenseNet121_model =
DenseNet121(include_top=False,weights=None,input_tensor=keras.layers.In
put(shape=(img_size,img_size,3)))

DenseNet121_model.load_weights('../input/densenet-keras/DenseNet-BC-121
-32-no-top.h5')

#  model = keras.Sequential()



#  model.add(keras.layers.Conv2D(filters = 32, kernel_size =
(5,5),padding = 'same',activation ='relu',

#                       input_shape = (img_size,img_size,3)))

#  model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
```

```
#  model.add(keras.layers.Conv2D(filters=64,kernel_size=
(3,3),padding = 'Same',activation ='relu'))
#  model.add(keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2)))


#  model.add(keras.layers.Conv2D(filters=96,kernel_size=
(3,3),padding = 'Same',activation ='relu'))
#  model.add(keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2)))


#  model.add(keras.layers.Conv2D(filters=96,kernel_size=
(3,3),padding = 'Same',activation ='relu'))
#  model.add(keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2)))


#  model.add(keras.layers.Flatten())
#  model.add(keras.layers.Dense(units = 512, activation = 'relu'))
#  model.add(keras.layers.Dense(units=5,activation='softmax'))


    p =
keras.layers.GlobalAveragePooling2D()(DenseNet121_model.output)
#  fl = keras.layers.Flatten()(p)
#  d2 = keras.layers.Dense(units = 1024, activation =
'relu',kernel_regularizer=keras.regularizers.l2(0.001))(p)
#  d1 = keras.layers.Dense(units = 512, activation =
'relu',kernel_regularizer=keras.regularizers.l2(0.001))(d2)
    d11 = keras.layers.Dense(units = 256, activation =
'relu',kernel_regularizer=keras.regularizers.l2(0.0001))(p) o1 =
    keras.layers.Dense(units = 5, activation = 'softmax')(d11)  model =
    keras.models.Model(inputs = DenseNet121_model.input,outputs
= o1)  sgd=keras.optimizers.SGD(lr=0.01,decay=1e-6,momentum=0.9,
```

```
nesterov=True)



model.compile(optimizer=sgd,loss='categorical_crossen
tropy',
metrics     =     ['accuracy'])
   print(model.summary()) return model
```

conv2_block3_0_relu (Activation	(None, 56, 56, 128)	0	conv2_block3_0_bn[0][0]
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 128)	16384	conv2_block3_0_relu[0][0]
conv2_block3_1_bn (BatchNormali	(None, 56, 56, 128)	512	conv2_block3_1_conv[0][0]
conv2_block3_1_relu (Activation	(None, 56, 56, 128)	0	conv2_block3_1_bn[0][0]
conv2_block3_2_conv (Conv2D)	(None, 56, 56, 32)	36864	conv2_block3_1_relu[0][0]
conv2_block3_concat (Concatenat	(None, 56, 56, 160)	0	conv2_block2_concat[0][0] conv2_block3_2_conv[0][0]
conv2_block4_0_bn (BatchNormali	(None, 56, 56, 160)	640	conv2_block3_concat[0][0]
conv2_block4_0_relu (Activation	(None, 56, 56, 160)	0	conv2_block4_0_bn[0][0]
conv2_block4_1_conv (Conv2D)	(None, 56, 56, 128)	20480	conv2_block4_0_relu[0][0]
conv2_block4_1_bn (BatchNormali	(None, 56, 56, 128)	512	conv2_block4_1_conv[0][0]
conv2_block4_1_relu (Activation	(None, 56, 56, 128)	0	conv2_block4_1_bn[0][0]
conv2_block4_2_conv (Conv2D)	(None, 56, 56, 32)	36864	conv2_block4_1_relu[0][0]
conv2_block4_concat (Concatenat	(None, 56, 56, 192)	0	conv2_block3_concat[0][0] conv2_block4_2_conv[0][0]
conv2_block5_0_bn (BatchNormali	(None, 56, 56, 192)	768	conv2_block4_concat[0][0]
conv2_block5_0_relu (Activation	(None, 56, 56, 192)	0	conv2_block5_0_bn[0][0]
conv2_block5_1_conv (Conv2D)	(None, 56, 56, 128)	24576	conv2_block5_0_relu[0][0]
conv2_block5_1_bn (BatchNormali	(None, 56, 56, 128)	512	conv2_block5_1_conv[0][0]
conv2_block5_1_relu (Activation	(None, 56, 56, 128)	0	conv2_block5_1_bn[0][0]
conv2_block5_2_conv (Conv2D)	(None, 56, 56, 32)	36864	conv2_block5_1_relu[0][0]
conv2_block5_concat (Concatenat	(None, 56, 56, 224)	0	conv2_block4_concat[0][0] conv2_block5_2_conv[0][0]
conv2_block6_0_bn (BatchNormali	(None, 56, 56, 224)	896	conv2_block5_concat[0][0]
conv2_block6_0_relu (Activation	(None, 56, 56, 224)	0	conv2_block6_0_bn[0][0]
conv2_block6_1_conv (Conv2D)	(None, 56, 56, 128)	28672	conv2_block6_0_relu[0][0]
conv2_block6_1_bn (BatchNormali	(None, 56, 56, 128)	512	conv2_block6_1_conv[0][0]
conv2_block6_1_relu (Activation	(None, 56, 56, 128)	0	conv2_block6_1_bn[0][0]
conv2_block6_2_conv (Conv2D)	(None, 56, 56, 32)	36864	conv2_block6_1_relu[0][0]

conv3_block3_2_conv (Conv2D)	(None, 28, 28, 32)	36864	conv3_block3_1_relu[0][0]
conv3_block3_concat (Concatenat	(None, 28, 28, 224)	0	conv3_block2_concat[0][0] conv3_block3_2_conv[0][0]
conv3_block4_0_bn (BatchNormali	(None, 28, 28, 224)	896	conv3_block3_concat[0][0]
conv3_block4_0_relu (Activation	(None, 28, 28, 224)	0	conv3_block4_0_bn[0][0]
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	28672	conv3_block4_0_relu[0][0]
conv3_block4_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block4_1_conv[0][0]
conv3_block4_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block4_1_bn[0][0]
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 32)	36864	conv3_block4_1_relu[0][0]
conv3_block4_concat (Concatenat	(None, 28, 28, 256)	0	conv3_block3_concat[0][0] conv3_block4_2_conv[0][0]
conv3_block5_0_bn (BatchNormali	(None, 28, 28, 256)	1024	conv3_block4_concat[0][0]
conv3_block5_0_relu (Activation	(None, 28, 28, 256)	0	conv3_block5_0_bn[0][0]
conv3_block5_1_conv (Conv2D)	(None, 28, 28, 128)	32768	conv3_block5_0_relu[0][0]
conv3_block5_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block5_1_conv[0][0]
conv3_block5_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block5_1_bn[0][0]
conv3_block5_2_conv (Conv2D)	(None, 28, 28, 32)	36864	conv3_block5_1_relu[0][0]
conv3_block5_concat (Concatenat	(None, 28, 28, 288)	0	conv3_block4_concat[0][0] conv3_block5_2_conv[0][0]
conv3_block6_0_bn (BatchNormali	(None, 28, 28, 288)	1152	conv3_block5_concat[0][0]
conv3_block6_0_relu (Activation	(None, 28, 28, 288)	0	conv3_block6_0_bn[0][0]
conv3_block6_1_conv (Conv2D)	(None, 28, 28, 128)	36864	conv3_block6_0_relu[0][0]
conv3_block6_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block6_1_conv[0][0]
conv3_block6_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block6_1_bn[0][0]
conv3_block6_2_conv (Conv2D)	(None, 28, 28, 32)	36864	conv3_block6_1_relu[0][0]
conv3_block6_concat (Concatenat	(None, 28, 28, 320)	0	conv3_block5_concat[0][0] conv3_block6_2_conv[0][0]
conv3_block7_0_bn (BatchNormali	(None, 28, 28, 320)	1280	conv3_block6_concat[0][0]
conv3_block7_0_relu (Activation	(None, 28, 28, 320)	0	conv3_block7_0_bn[0][0]

conv3_block3_2_conv (Conv2D)	(None, 28, 28, 32)	36864	conv3_block3_1_relu[0][0]
conv3_block3_concat (Concatenat	(None, 28, 28, 224)	0	conv3_block2_concat[0][0] conv3_block3_2_conv[0][0]
conv3_block4_0_bn (BatchNormali	(None, 28, 28, 224)	896	conv3_block3_concat[0][0]
conv3_block4_0_relu (Activation	(None, 28, 28, 224)	0	conv3_block4_0_bn[0][0]
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	28672	conv3_block4_0_relu[0][0]
conv3_block4_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block4_1_conv[0][0]
conv3_block4_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block4_1_bn[0][0]
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 32)	36864	conv3_block4_1_relu[0][0]
conv3_block4_concat (Concatenat	(None, 28, 28, 256)	0	conv3_block3_concat[0][0] conv3_block4_2_conv[0][0]
conv3_block5_0_bn (BatchNormali	(None, 28, 28, 256)	1024	conv3_block4_concat[0][0]
conv3_block5_0_relu (Activation	(None, 28, 28, 256)	0	conv3_block5_0_bn[0][0]
conv3_block5_1_conv (Conv2D)	(None, 28, 28, 128)	32768	conv3_block5_0_relu[0][0]
conv3_block5_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block5_1_conv[0][0]
conv3_block5_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block5_1_bn[0][0]
conv3_block5_2_conv (Conv2D)	(None, 28, 28, 32)	36864	conv3_block5_1_relu[0][0]
conv3_block5_concat (Concatenat	(None, 28, 28, 288)	0	conv3_block4_concat[0][0] conv3_block5_2_conv[0][0]
conv3_block6_0_bn (BatchNormali	(None, 28, 28, 288)	1152	conv3_block5_concat[0][0]
conv3_block6_0_relu (Activation	(None, 28, 28, 288)	0	conv3_block6_0_bn[0][0]
conv3_block6_1_conv (Conv2D)	(None, 28, 28, 128)	36864	conv3_block6_0_relu[0][0]
conv3_block6_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block6_1_conv[0][0]
conv3_block6_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block6_1_bn[0][0]
conv3_block6_2_conv (Conv2D)	(None, 28, 28, 32)	36864	conv3_block6_1_relu[0][0]
conv3_block6_concat (Concatenat	(None, 28, 28, 320)	0	conv3_block5_concat[0][0] conv3_block6_2_conv[0][0]
conv3_block7_0_bn (BatchNormali	(None, 28, 28, 320)	1280	conv3_block6_concat[0][0]
conv3_block7_0_relu (Activation	(None, 28, 28, 320)	0	conv3_block7_0_bn[0][0]

```
conv4_block5_0_bn (BatchNormali (None, 14, 14, 384)  1536        conv4_block4_concat[0][0]
conv4_block5_0_relu (Activation (None, 14, 14, 384)  0           conv4_block5_0_bn[0][0]
conv4_block5_1_conv (Conv2D)    (None, 14, 14, 128)  49152       conv4_block5_0_relu[0][0]
conv4_block5_1_bn (BatchNormali (None, 14, 14, 128)  512         conv4_block5_1_conv[0][0]
conv4_block5_1_relu (Activation (None, 14, 14, 128)  0           conv4_block5_1_bn[0][0]
conv4_block5_2_conv (Conv2D)    (None, 14, 14, 32)   36864       conv4_block5_1_relu[0][0]
conv4_block5_concat (Concatenat (None, 14, 14, 416)  0           conv4_block4_concat[0][0]
                                                                 conv4_block5_2_conv[0][0]
conv4_block6_0_bn (BatchNormali (None, 14, 14, 416)  1664        conv4_block5_concat[0][0]
conv4_block6_0_relu (Activation (None, 14, 14, 416)  0           conv4_block6_0_bn[0][0]
conv4_block6_1_conv (Conv2D)    (None, 14, 14, 128)  53248       conv4_block6_0_relu[0][0]
conv4_block6_1_bn (BatchNormali (None, 14, 14, 128)  512         conv4_block6_1_conv[0][0]
conv4_block6_1_relu (Activation (None, 14, 14, 128)  0           conv4_block6_1_bn[0][0]
conv4_block6_2_conv (Conv2D)    (None, 14, 14, 32)   36864       conv4_block6_1_relu[0][0]
conv4_block6_concat (Concatenat (None, 14, 14, 448)  0           conv4_block5_concat[0][0]
                                                                 conv4_block6_2_conv[0][0]
conv4_block7_0_bn (BatchNormali (None, 14, 14, 448)  1792        conv4_block6_concat[0][0]
conv4_block7_0_relu (Activation (None, 14, 14, 448)  0           conv4_block7_0_bn[0][0]
conv4_block7_1_conv (Conv2D)    (None, 14, 14, 128)  57344       conv4_block7_0_relu[0][0]
conv4_block7_1_bn (BatchNormali (None, 14, 14, 128)  512         conv4_block7_1_conv[0][0]
conv4_block7_1_relu (Activation (None, 14, 14, 128)  0           conv4_block7_1_bn[0][0]
conv4_block7_2_conv (Conv2D)    (None, 14, 14, 32)   36864       conv4_block7_1_relu[0][0]
conv4_block7_concat (Concatenat (None, 14, 14, 480)  0           conv4_block6_concat[0][0]
                                                                 conv4_block7_2_conv[0][0]
conv4_block8_0_bn (BatchNormali (None, 14, 14, 480)  1920        conv4_block7_concat[0][0]
conv4_block8_0_relu (Activation (None, 14, 14, 480)  0           conv4_block8_0_bn[0][0]
conv4_block8_1_conv (Conv2D)    (None, 14, 14, 128)  61440       conv4_block8_0_relu[0][0]
conv4_block8_1_bn (BatchNormali (None, 14, 14, 128)  512         conv4_block8_1_conv[0][0]
conv4_block8_1_relu (Activation (None, 14, 14, 128)  0           conv4_block8_1_bn[0][0]
```

```
                                                                 conv5_block14_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block15_0_bn (BatchNormal (None, 7, 7, 960)    3840        conv5_block14_concat[0][0]
__________________________________________________________________________________________________
conv5_block15_0_relu (Activatio (None, 7, 7, 960)    0           conv5_block15_0_bn[0][0]
__________________________________________________________________________________________________
conv5_block15_1_conv (Conv2D)   (None, 7, 7, 128)    122880      conv5_block15_0_relu[0][0]
__________________________________________________________________________________________________
conv5_block15_1_bn (BatchNormal (None, 7, 7, 128)    512         conv5_block15_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block15_1_relu (Activatio (None, 7, 7, 128)    0           conv5_block15_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block15_2_conv (Conv2D)   (None, 7, 7, 32)     36864       conv5_block15_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block15_concat (Concatena (None, 7, 7, 992)    0           conv5_block14_concat[0][0]
                                                                 conv5_block15_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block16_0_bn (BatchNormal (None, 7, 7, 992)    3968        conv5_block15_concat[0][0]
__________________________________________________________________________________________________
conv5_block16_0_relu (Activatio (None, 7, 7, 992)    0           conv5_block16_0_bn[0][0]
__________________________________________________________________________________________________
conv5_block16_1_conv (Conv2D)   (None, 7, 7, 128)    126976      conv5_block16_0_relu[0][0]
__________________________________________________________________________________________________
conv5_block16_1_bn (BatchNormal (None, 7, 7, 128)    512         conv5_block16_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block16_1_relu (Activatio (None, 7, 7, 128)    0           conv5_block16_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block16_2_conv (Conv2D)   (None, 7, 7, 32)     36864       conv5_block16_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block16_concat (Concatena (None, 7, 7, 1024)   0           conv5_block15_concat[0][0]
                                                                 conv5_block16_2_conv[0][0]
__________________________________________________________________________________________________
bn (BatchNormalization)         (None, 7, 7, 1024)   4096        conv5_block16_concat[0][0]
__________________________________________________________________________________________________
relu (Activation)               (None, 7, 7, 1024)   0           bn[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_1 (Glo (None, 1024)         0           relu[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          262400      global_average_pooling2d_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 5)            1285        dense_1[0][0]
==================================================================================================
Total params: 7,301,189
Trainable params: 7,217,541
Non-trainable params: 83,648
__________________________________________________________________________________________________
None
```

The code configures and trains a DenseNet121-based model (my model) for a specified number of epochs with early stopping, learning rate reduction on plateau, model check pointing, and a custom Quadratic Weighted Kappa (QWK) callback for monitoring .

```
EPOCHS = 50

earlystop= keras.callbacks.EarlyStopping(patience=10)

learning_rate_reduction =
keras.callbacks.ReduceLROnPlateau(monitor='val_acc',
                                  patience=2,
                                  verbose=1, factor=0.5,
                                  min_lr=0.00001)

checkpoint =
keras.callbacks.ModelCheckpoint('../working/DenseNet121.h5',
monitor='val_loss', verbose=1,
                             save_best_only=True, mode='min',
save_weights_only = True)

qwk = QWKCallback((X_val,Y_val))

mycallbacks= [earlystop, learning_rate_reduction,checkpoint,qwk]
```

```
print(qwk)
```

```
<__main__.QWKCallback object at 0x7f540ab3fd68>
```

```
#Warmupthemodelwithclassweights EPOCHS =

10

history=mymodel.fit_generator(training_generator,steps_per_epoch=
X_train.shape[0] // batch_size,epochs = EPOCHS,

validation_data=(X_val,Y_val), validation_steps = 10,
workers=2,use_multiprocessing=True, verbose=2,
callbacks=mycallbacks, class_weight=cls_wt_dict)
```

```
Epoch 1/10

/opt/conda/lib/python3.6/site-packages/keras/engine/training_generator.py:47: Use
rWarning: Using a generator with `use_multiprocessing=True` and multiple workers
may duplicate your data. Please consider using the`keras.utils.Sequence class.
  UserWarning('Using a generator with `use_multiprocessing=True`'

 - 81s - loss: 1.1908 - acc: 0.6900 - val_loss: 1.1318 - val_acc: 0.5861

Epoch 00001: val_loss improved from inf to 1.13182, saving model to ../working/De
nseNet121.h5
Epoch 0 : QWK: 0.6606347581629278
saving checkpoint:  0.6606347581629278
Epoch 2/10
 - 47s - loss: 1.1552 - acc: 0.6961 - val_loss: 1.0400 - val_acc: 0.5836

Epoch 00002: val_loss improved from 1.13182 to 1.04000, saving model to ../workin
g/DenseNet121.h5
Epoch 1 : QWK: 0.4967983145414807
Epoch 3/10
 - 47s - loss: 1.0514 - acc: 0.7447 - val_loss: 0.9467 - val_acc: 0.6500

Epoch 00003: val_loss improved from 1.04000 to 0.94669, saving model to ../workin
g/DenseNet121.h5
Epoch 2 : QWK: 0.6061538805192736
Epoch 4/10
 - 47s - loss: 1.0690 - acc: 0.7340 - val_loss: 1.4204 - val_acc: 0.5098

Epoch 00004: val_loss did not improve from 0.94669
Epoch 3 : QWK: 0.6733880123185333
saving checkpoint:  0.6733880123185333
Epoch 5/10
 - 47s - loss: 1.0158 - acc: 0.7578 - val_loss: 1.0547 - val_acc: 0.6025

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.0049999999888241291.

Epoch 00005: val_loss did not improve from 0.94669
Epoch 4 : QWK: 0.40468768771292674
Epoch 6/10
 - 46s - loss: 1.0538 - acc: 0.7208 - val_loss: 0.9260 - val_acc: 0.6270
 Epoch 00006: val_loss improved from 0.94669 to 0.92603, saving model to ../workin
 g/DenseNet121.h5
 Epoch 5 : QWK: 0.6010582954308986
 Epoch 7/10
  - 47s - loss: 0.9715 - acc: 0.7558 - val_loss: 0.8224 - val_acc: 0.6836

 Epoch 00007: val_loss improved from 0.92603 to 0.82242, saving model to ../workin
 g/DenseNet121.h5
 Epoch 6 : QWK: 0.8450581787790845
 saving checkpoint:  0.8450581787790845
 Epoch 8/10
  - 46s - loss: 0.8936 - acc: 0.7887 - val_loss: 0.9989 - val_acc: 0.6525

 Epoch 00008: val_loss did not improve from 0.82242
 Epoch 7 : QWK: 0.7028776443240523
 Epoch 9/10
  - 47s - loss: 0.8548 - acc: 0.7718 - val_loss: 0.9379 - val_acc: 0.6959

 Epoch 00009: val_loss did not improve from 0.82242
 Epoch 8 : QWK: 0.7600499276290307
 Epoch 10/10
  - 46s - loss: 0.9754 - acc: 0.7763 - val_loss: 0.9158 - val_acc: 0.6721

 Epoch 00010: val_loss did not improve from 0.82242
 Epoch 9 : QWK: 0.7679516452608758
```

```
EPOCHS = 50

history=mymodel.fit_generator(training_generator,steps_per_epoch=
X_train.shape[0] // batch_size,epochs = EPOCHS,
validation_data=(X_val,Y_val), validation_steps = 10,
workers=2,use_multiprocessing=True, verbose=2, callbacks=mycallbacks)
```

```
Epoch 1/50
 - 53s - loss: 0.7034 - acc: 0.8076 - val_loss: 0.6413 - val_acc: 0.7943

Epoch 00001: val_loss improved from 0.82242 to 0.64127, saving model to ../workin
g/DenseNet121.h5
Epoch 0 : QWK: 0.8430588862243539
Epoch 2/50
 - 47s - loss: 0.6261 - acc: 0.8326 - val_loss: 0.6528 - val_acc: 0.7828

Epoch 00002: val_loss did not improve from 0.64127
Epoch 1 : QWK: 0.8516125400571757
saving checkpoint:  0.8516125400571757
Epoch 3/50
 - 47s - loss: 0.6396 - acc: 0.8043 - val_loss: 0.7131 - val_acc: 0.7623

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.0024999999441206455.

Epoch 00003: val_loss did not improve from 0.64127
Epoch 2 : QWK: 0.8414919398212692
Epoch 4/50
 - 46s - loss: 0.6370 - acc: 0.8306 - val_loss: 0.5923 - val_acc: 0.8189

Epoch 00004: val_loss improved from 0.64127 to 0.59228, saving model to ../workin
g/DenseNet121.h5
Epoch 3 : QWK: 0.880622590197721
saving checkpoint:  0.880622590197721
Epoch 5/50
 - 47s - loss: 0.5706 - acc: 0.8483 - val_loss: 0.5980 - val_acc: 0.8057

Epoch 00005: val_loss did not improve from 0.59228
Epoch 4 : QWK: 0.8866740469412809
saving checkpoint:  0.8866740469412809
Epoch 6/50
 - 47s - loss: 0.5910 - acc: 0.8368 - val_loss: 0.5928 - val_acc: 0.8139

Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.0012499999720603228.

Epoch 00006: val_loss did not improve from 0.59228
Epoch 5 : QWK: 0.8744554679721348
Epoch 7/50
 - 47s - loss: 0.5729 - acc: 0.8516 - val_loss: 0.5930 - val_acc: 0.8082

Epoch 00007: val_loss did not improve from 0.59228
```

```
mymodel.save_weights("model.h5")
```

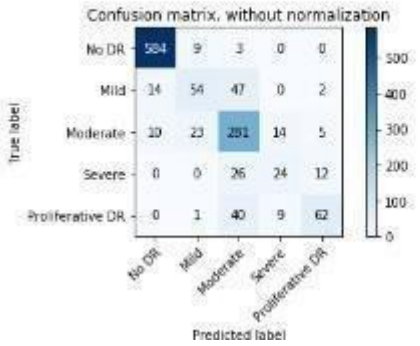
```
Y_val_pred= mymodel.predict_on_batch(X_val)
```

```
Y_val_pred_hot=np.argmax(Y_val_pred,axis=1)

Y_val_actual_hot = np.argmax(Y_val,axis=1)
```

```
plot_confusion_matrix(Y_val_actual_hot,Y_val_pred_hot,
np.array(class_labels))
```

```
Confusion matrix, without normalization
[[584   9   3   0   0]
 [ 14  54  47   0   2]
 [ 10  23 281  14   5]
 [  0   0  26  24  12]
 [  0   1  40   9  62]]
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5403b3c978>
```

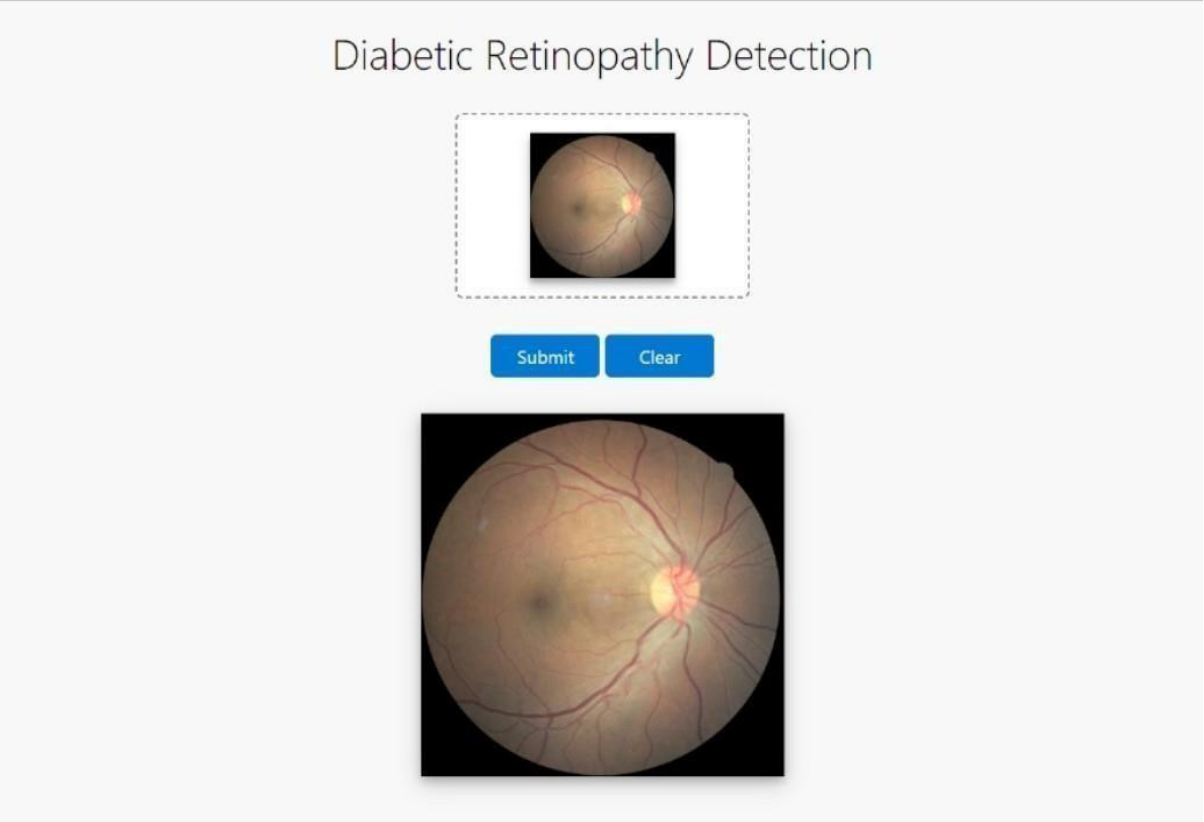


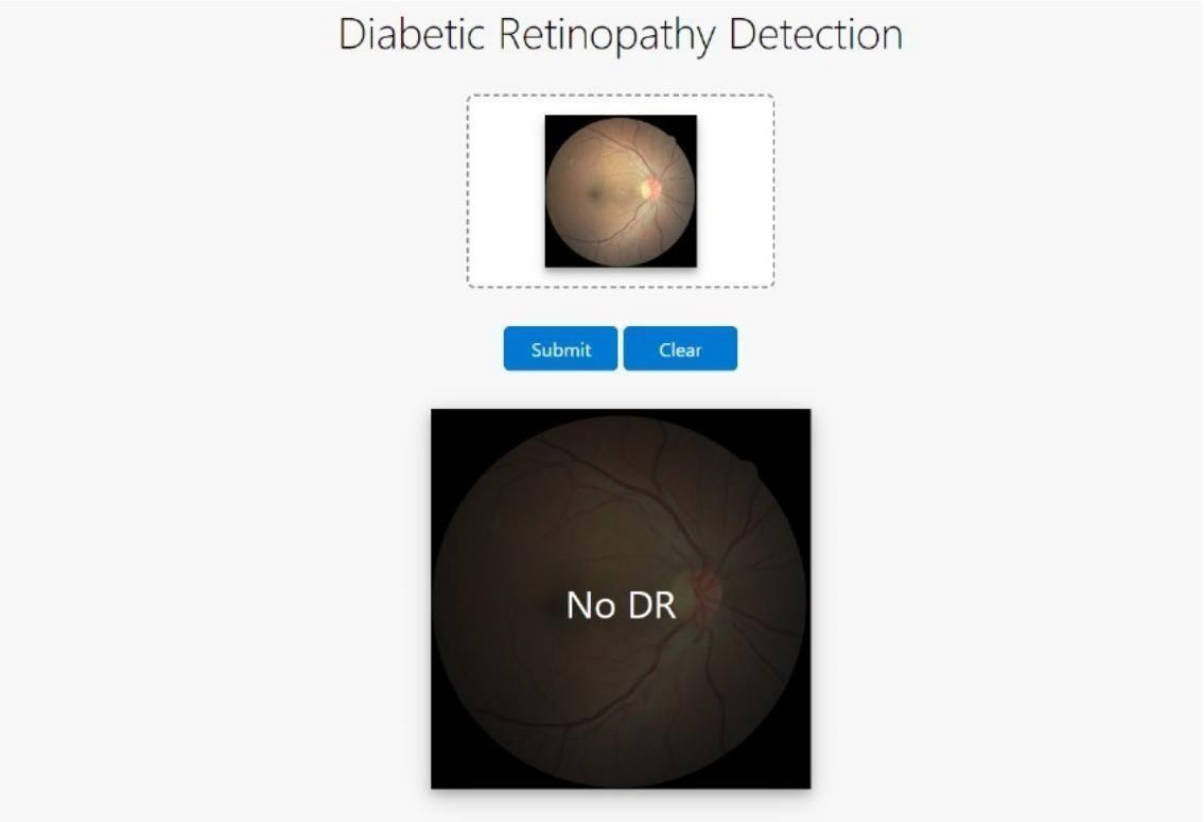
WEBPAGE:

After the user provides the input retinal image, the image is fed into the DR Model. The DR Model then analyzes the image and makes predictions based on the data it has been trained on. The predicted model is then saved for further use or reference. The Flask application displays reports indicating the severity of diabetic retinopathy (DR) to the users.

These reports are generated after processing the uploaded fundal image.

Additionally, the Flask application also displays the processed fundal image along with the severity assessment report.





8. PERFORMANCE TESTING :

```
from sklearn.metrics import classification_report
classification_rep = classification_report(Y_val_actual_hot, Y_val_pred_hot, target_names=class_labels)
print(classification_rep)
```

```
                precision    recall  f1-score   support

         No DR       0.96      0.98      0.97       596
          Mild       0.61      0.47      0.53       117
      Moderate       0.71      0.83      0.77       333
        Severe       0.46      0.39      0.42        62
Proliferative DR       0.73      0.52      0.61       112

      accuracy                           0.82      1220
     macro avg       0.70      0.64      0.66      1220
  weighted avg       0.81      0.82      0.81      1220
```

8. RESULTS

Output Screenshots:

With an overall accuracy of 0.89, the model demonstrates effective classification performance. The model's strong accuracy implies its capability to predict diabetic retinopathy accurately on the test dataset. The recall of the classification report in the test is 0.97

```
Epoch 28 : QWK: 0.8984336700467013
```

8. ADVANTAGES & DISADVANTAGES

Advantages:

Early Detection: Enables early identification of diabetic retinopathy, facilitating timely intervention and treatment to prevent vision loss.

Accuracy: Utilizes deep learning algorithms, providing highly accurate diagnoses by analyzing retinal images with precision.

Efficiency: Automates the screening process, reducing the workload on ophthalmologists and enabling faster diagnoses for a larger number of patients.

Improved Patient Care: Enhances patient care by offering rapid and reliable diagnostic reports, aiding in personalized treatment planning.

Cost-Effectiveness: Reduces healthcare costs associated with manual screening and latestage interventions by detecting retinopathy at its earlier stages.

Accessibility: Allows for remote screening and diagnosis, particularly beneficial in regions with limited access to healthcare facilities or specialists.

Disadvantages:

Dependency on Imaging Quality: Accuracy heavily relies on the quality of acquired retinal images; poor-quality images might affect the system's performance.

Algorithm Bias: Algorithms might demonstrate biases based on the dataset used for training, leading to inaccuracies or misdiagnoses in certain cases.

Regulatory
Challenges: Compliance with health care regulations and ethical considerations regarding patient data privacy and usage can pose challenges.

Technical Complexity: Developing and maintaining a deep learning-based system requires specialized skills and ongoing technical expertise.

Initial Investment: Implementation and setup costs, including infrastructure, software, and training, might be substantial initially.

Limited Generalization: The system's accuracy might vary based on patient diversity, as certain populations or demographic factors could influence its effectiveness.

11. CONCLUSION:

In conclusion, the development and implementation of a Diabetic Retinopathy Detection System leveraging deep learning techniques present a significant leap forward in early diagnosis and intervention for diabetic retinopathy .This automated system offer promising prospects in revolutionizing the way retinal images are analysed, providing accurate and timely identification of retinopathy stages. The amalgamation of machine learning algorithms with medical imaging not only enhances the efficiency of diagnoses but also opens avenues for personalized patient care. Despite challenges in image quality and algorithm biases, the system's potential in preventing vision loss and improving patient outcomes cannot be understated. Collaborative efforts between technologists, healthcare practitioners, and regulatory bodies are crucial to address challenges, ensure ethical compliance, and further enhance the system's accuracy and accessibility.

11.Future Scope:

- The future scope for the Diabetic Retinopathy Detection System is vast and multifaceted. Some potential avenues for further development and improvement include:
- Enhanced Algorithm Refinement: Continuous refinement of deep learning algorithms to reduce biases, improve accuracy, and handle a more diverse range of retinal images.
- Integration with Telemedicine: Integrating the system with telemedicine platforms for remote patient screening, especially in under serve dare as with limited access to specialists.
- Extended Diagnostic Capabilities: Expanding the system's capabilities to detect and classify other ocular diseases or abnormalities from retinal images.
- Longitudinal Data Analysis: Implementing tools for longitudinal analysis of patient data to monitor disease progression, treatment response, and long-term outcomes.

- Real-time Decision Support: Advancing the system to offer real-time diagnostic support to healthcare professionals during patient consultations.
- Ethical and Regulatory Compliance: Striving for adherence to evolving healthcare regulations, privacy laws, and ethical considerations while handling patient data.

Collaborative Research: Encouraging collaboration among medical researchers, data scientists, and healthcare providers for collective efforts in refining the system's accuracy and efficacy.

11.APPENDIX Source Code GitHub & Project Demo Link

GitHub files **https://github.com/aratipatil2227/ai-project**

Demo Link:

https://drive.google.com/file/d/1fyED7kvIYQ2Q_faGWhr3-GHfJhZkZzn1/view?usp=drive_link