**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

1. **LITERATURE SURVEY**

Existing problem

References

Problem Statement Definition

1. **IDEATION & PROPOSED SOLUTION**

Empathy Map Canvas

Ideation & Brainstorming

1. **REQUIREMENT ANALYSIS**

Functional requirement

Non-Functional requirements

1. **PROJECT DESIGN**

Data Flow Diagrams & User Stories

Solution Architecture

1. **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

1. **PERFORMANCE TESTING**

Performance Metrics

1. **RESULTS**

Output Screenshots

1. **ADVANTAGES & DISADVANTAGES**
2. **CONCLUSION**
3. **FUTURE SCOPE**
4. **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

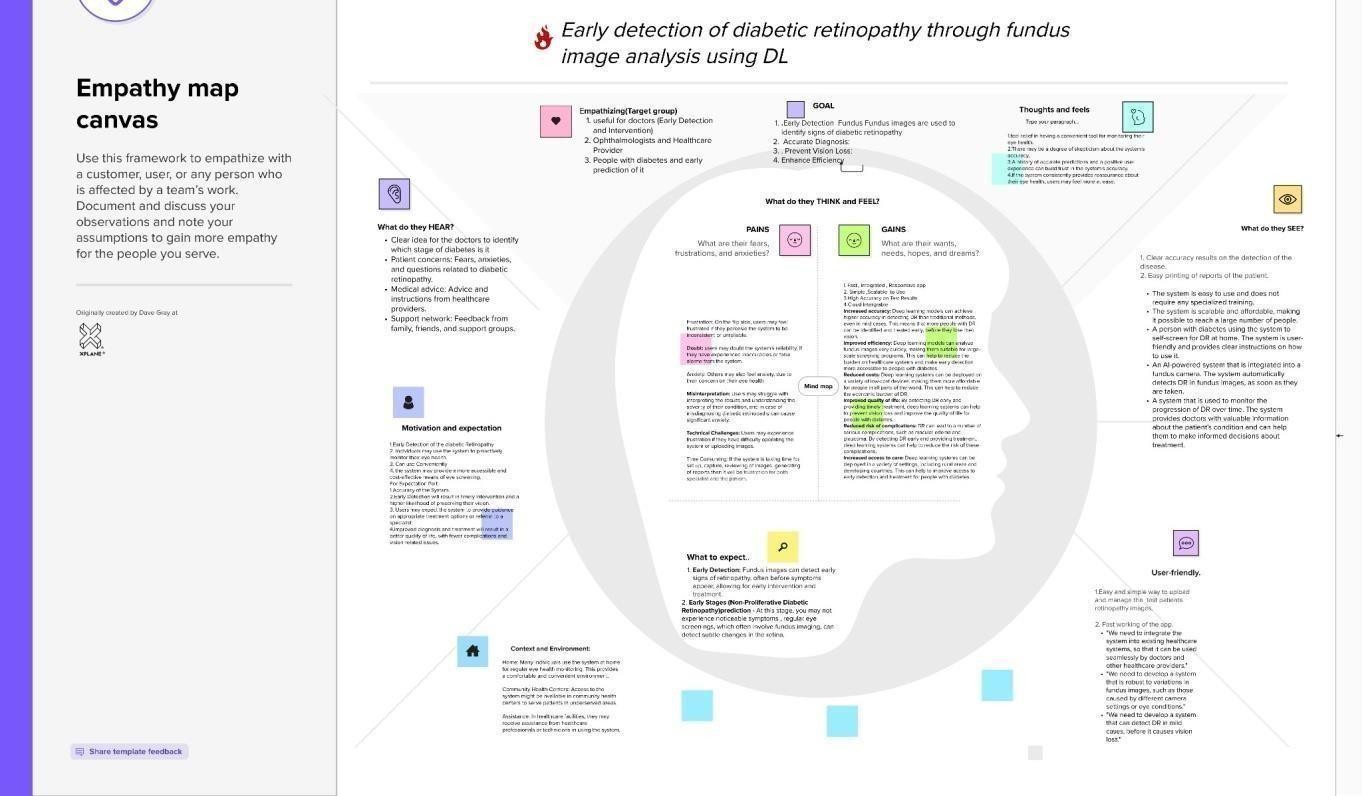
research and clinical validation to ascertain the practical application and assess the real-world impact of this technology on patient care and outcomes. The research aligns with a broader landscape of studies exploring the integration of artificial intelligence and deep learning techniques in ophthalmology, underlining the collective endeavor to enhance diagnostic capabilities, particularly in diabetic retinopathy detection and management.

**Problem Statement Definition:**

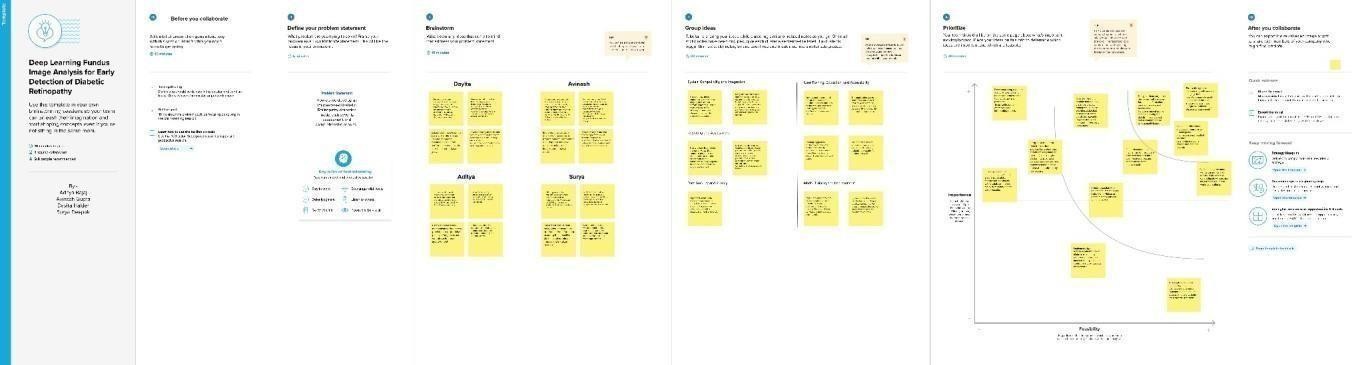
The primary goal of this project is to devise an accurate, interpretable, and scalable machine learning framework leveraging deep learning methodologies for the early detection and classification of Diabetic Retinopathy utilizing retinal images and patient data. This model aims to surmount the limitations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

1. **IDEATION & PROPOSED SOLUTION:**

**Empathy Map Canvas:**



**Ideation & Brainstorming**



1. **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.

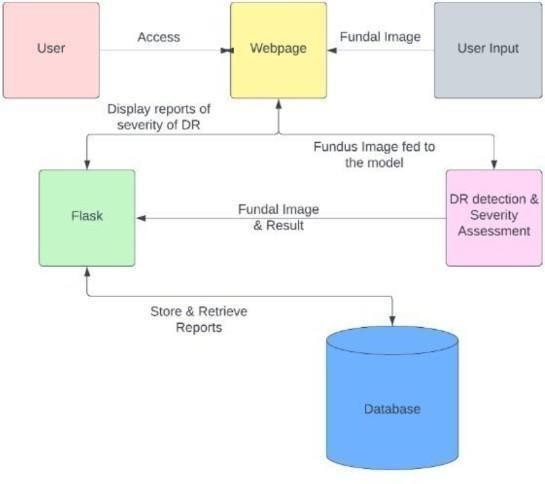
**Security:** Robust measures should be implemented to secure patient data, employing encryption protocols, access controls, and compliance with healthcare data regulations.

**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 and severity assessment | The webpage should have an intuitive interface for uploading and submitting fundal images | Sprint-1 |
|  |  | 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**

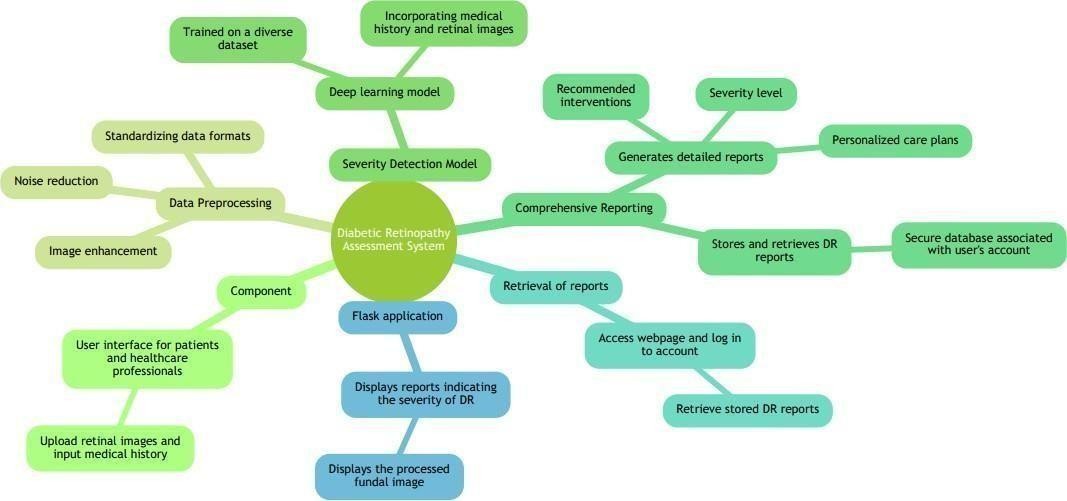
* 1. **User Inputs:** This component provides an intuitive user interface for patients and health care professionals to upload retinal images and input relevant medical history.

* 1. **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.

* 1. **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.

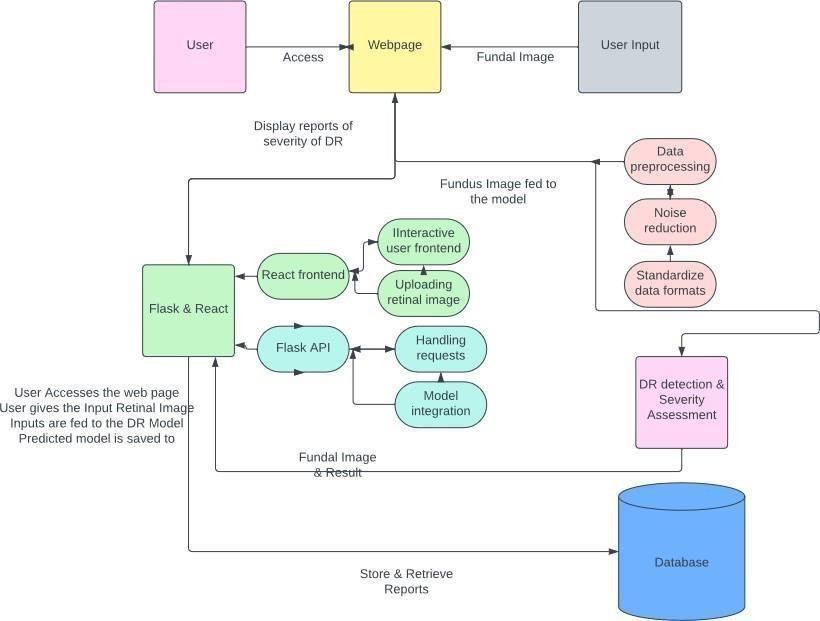
* 1. **Comprehensive Reporting:** Based on the severity assessment, this component generates detailed reports that include the detected severity level, recommended interventions, and personalized care plans.

**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**



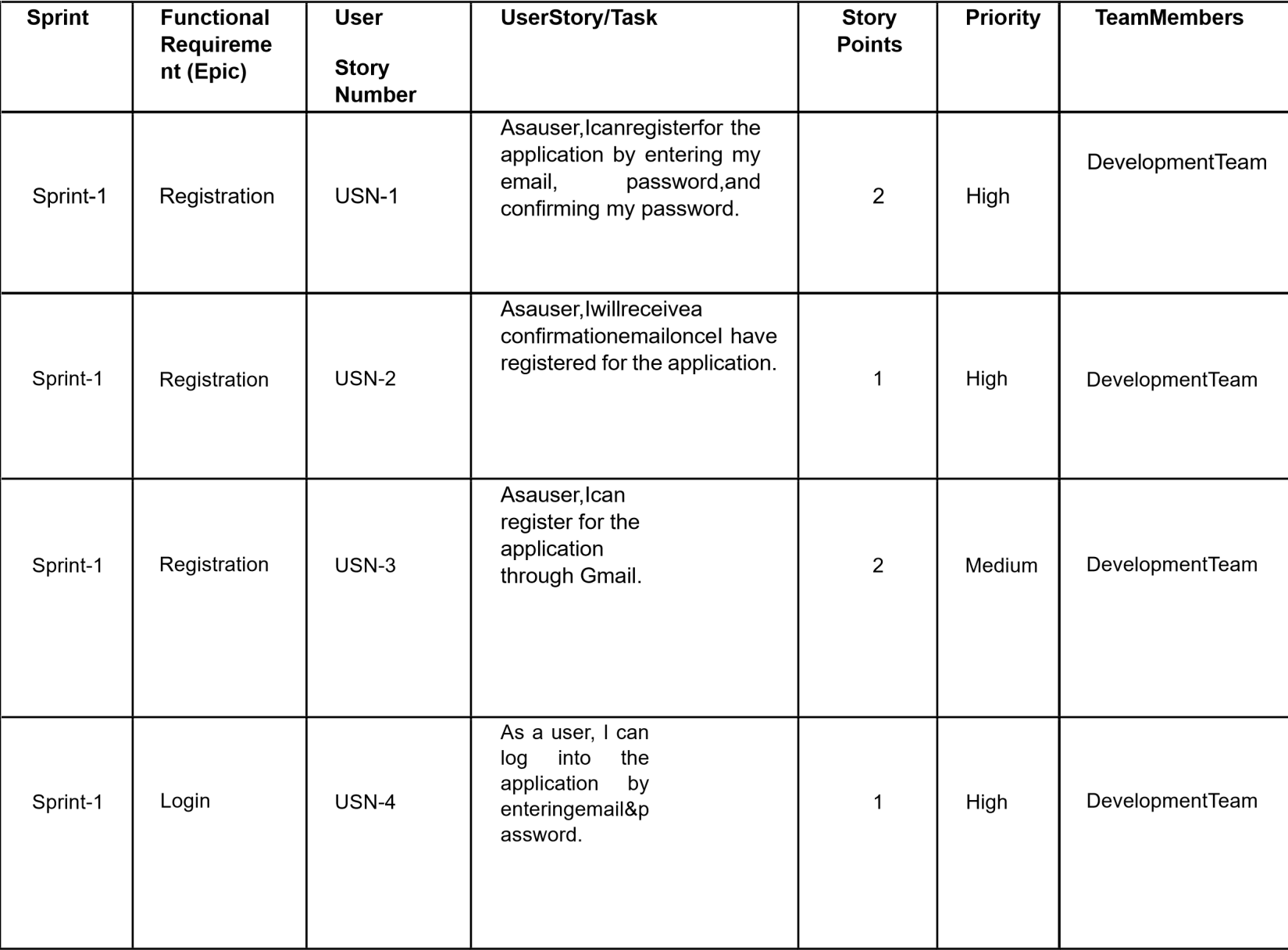
1. Users access a webpage to input their fundal image for diabetic retinopathy (DR) detection and severity assessment.

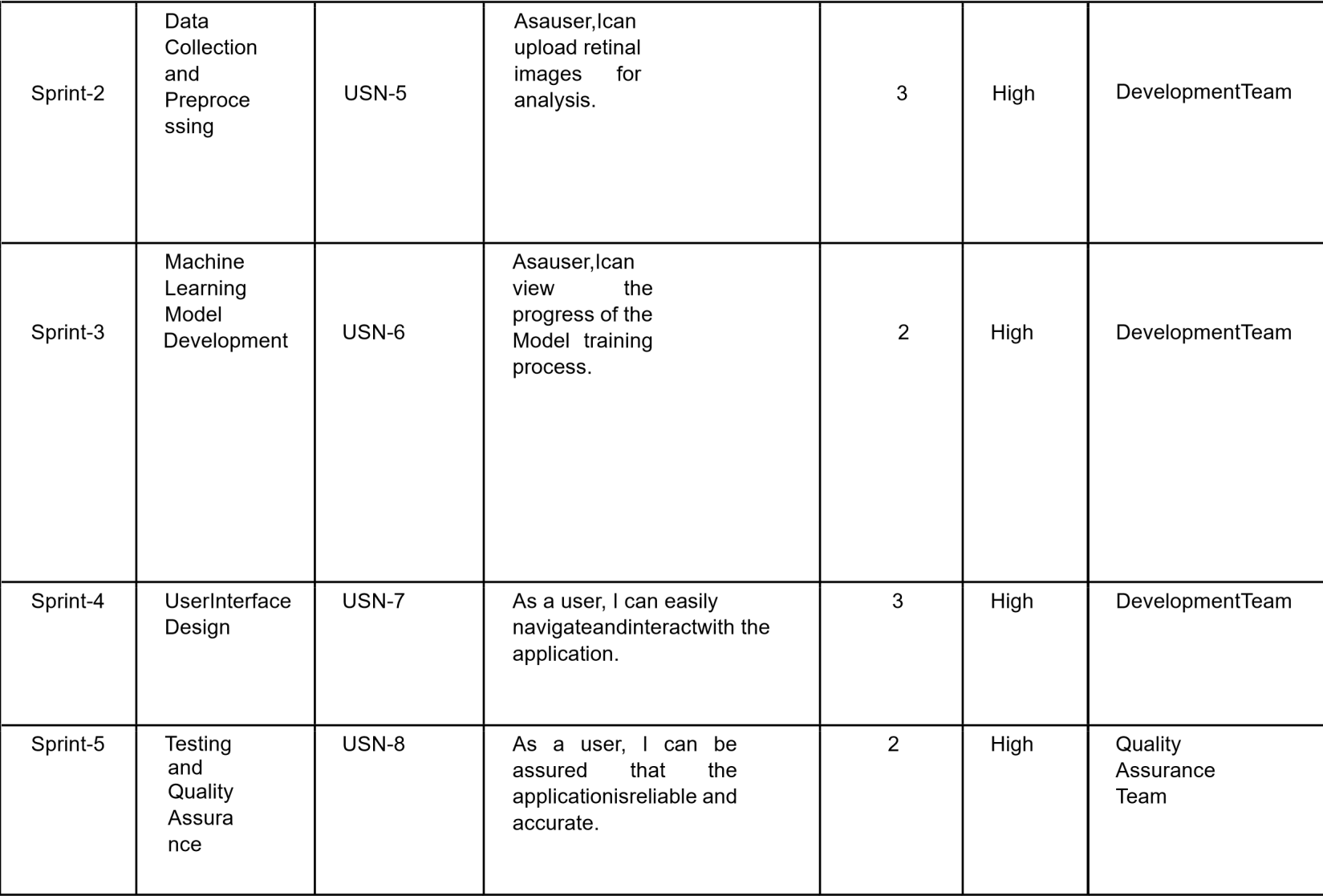
1. The input is processed, and the results along with the fundal image are sent to a Flask application.
2. The Flask app displays reports indicating the severity of DR to the users.
3. The system stores these reports in a database for future retrieval.
4. Users can later access and retrieve their DR 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** | **Total**  **Story Point**  **s** | **Duratio n** | **Sprint**  **Start**  **Date** | **Sprint**  **EndDate**  **(Planned**  **)** | **Story**  **Points Complete d (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 algebraimportpandasaspd#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 importcv2# 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/aptosresnet50baselinecla](http://www.kaggle.com/mathormad/aptos-resnet50-baseline)ss 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>

[W](https://www.kaggle.com/c/aptos2019-blindness-detection/data)e are provided with a large set of retina images taken using [fundus photography un](https://en.wikipedia.org/wiki/Fundus_photography)der a variety of imaging conditions.

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

1. - No DR
2. - Mild
3. - Moderate
4. - Severe
5. -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**

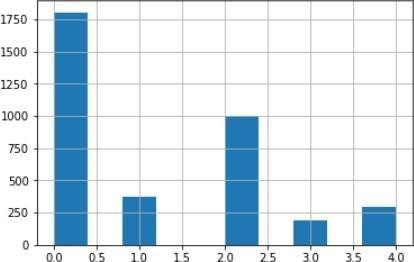
The load \_raw\_ images\_ defunction loads raw images from a Data Frame, resizes them, performs one-hot encoding on labels, and returns processed image data (X) and corresponding labels (Y).

|  |
| --- |
| 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.



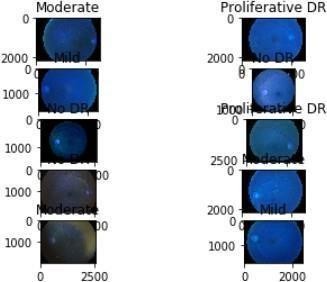


Creates a mapping between numerical labels and descriptive titles for APTOS Blindness Detection classes, providing both a dictionary (label\_title) and a list (class\_labels) for convenient reference .



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.





**Training and Validation**



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)

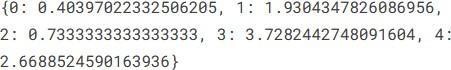
Y\_train\_labels = np.argmax(Y\_train,axis=1)

class\_weights =

class\_weight.compute\_class\_weight('balanced',np.unique(Y\_train\_labels)

, Y\_train\_labels) cls\_wt\_dict=dict(enumerate(class\_weights))

print(cls\_wt\_dict)



|  |  |
| --- | --- |
| 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:**

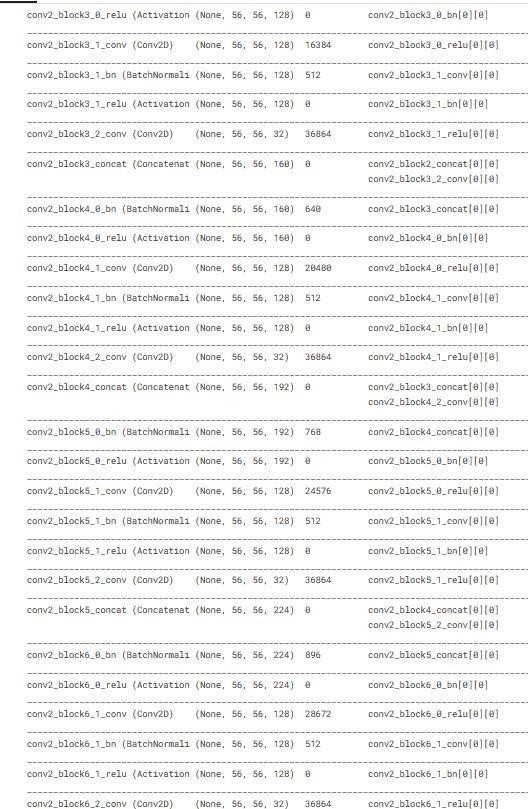
**Dense Net121-based Classification Model**

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, |

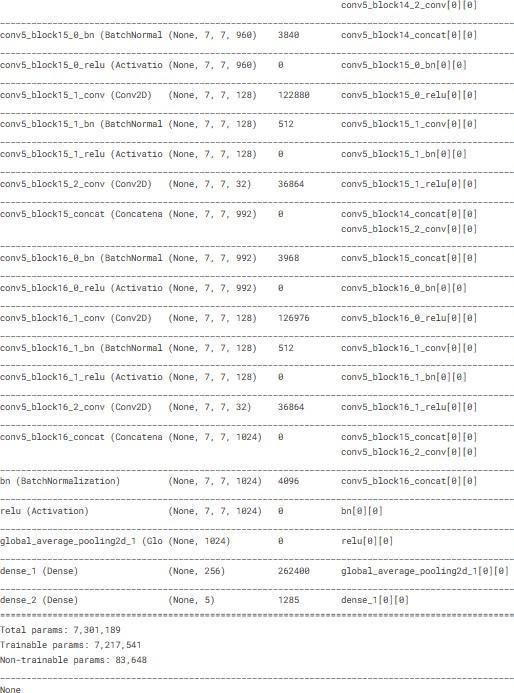
|  |
| --- |
| nesterov=True) |
| model.compile(optimizer=sgd,loss='categorical\_crossentropy',  metrics = ['accuracy'])  print(model.summary()) return model |











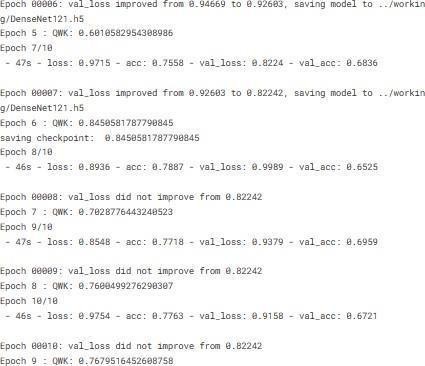
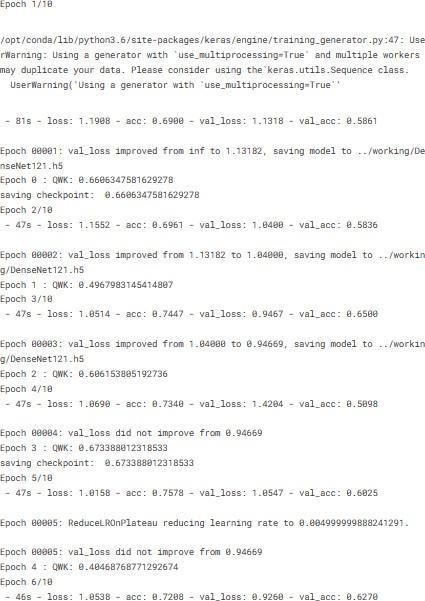
The code configures and trains a DenseNet121-based model (my model) for a specified number of epochs with early stopping, learning rate reduction on plateau, model check pointing, and a custom Quadratic Weighted Kappa (QWK) callback for monitoring .

|  |
| --- |
| 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] |

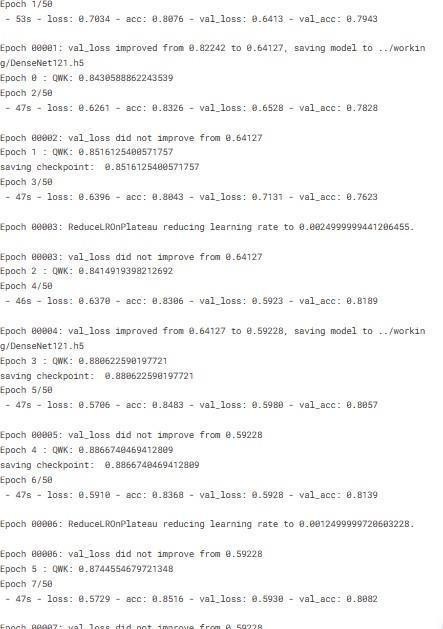
print(qwk)



|  |
| --- |
| #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) |



|  |
| --- |
| 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) |

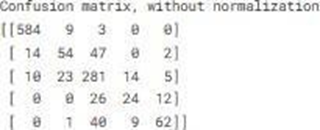


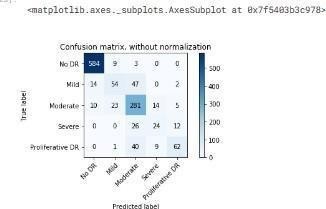










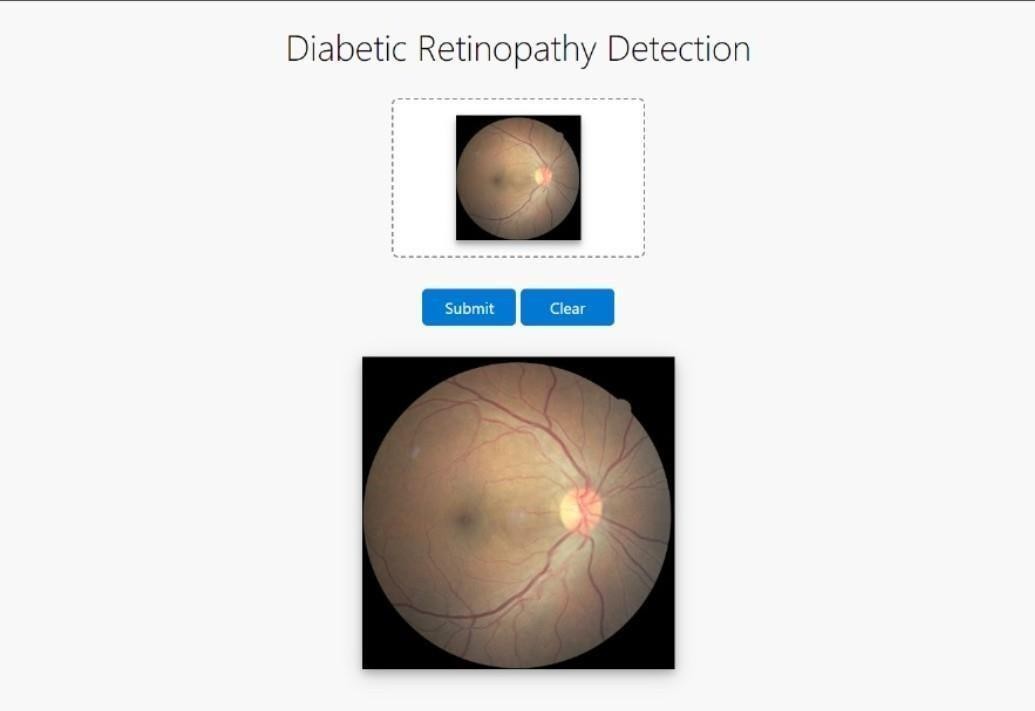
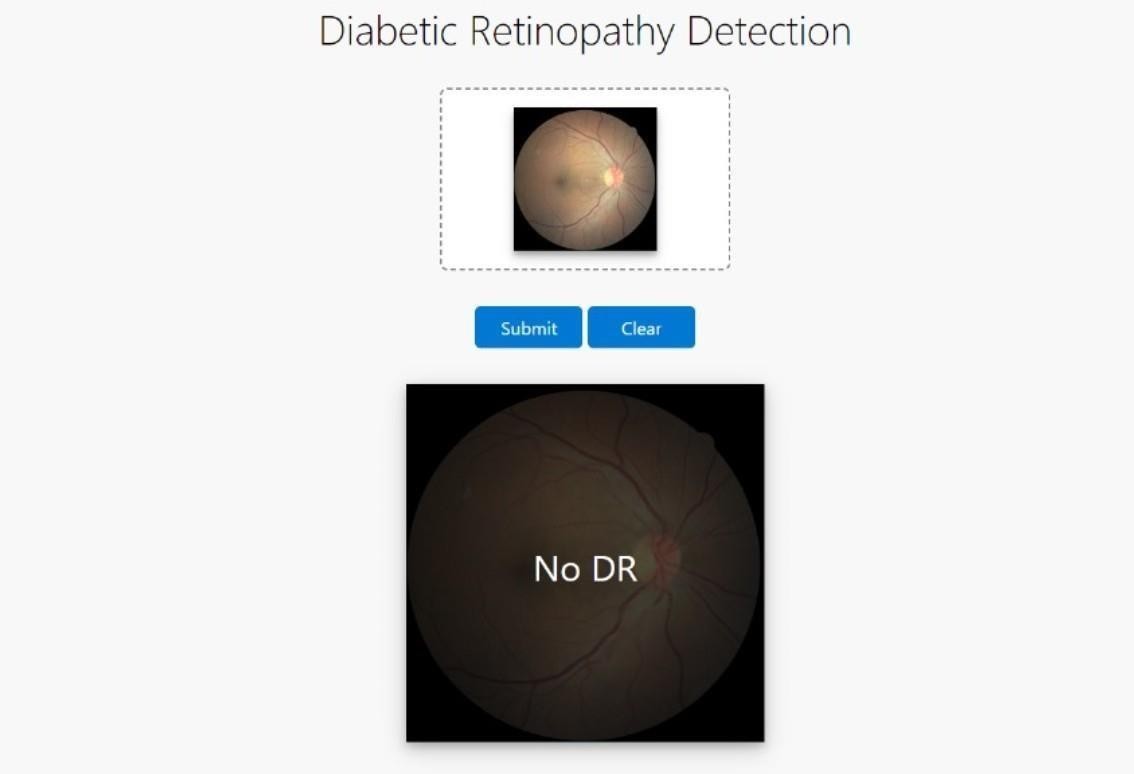


**WEBPAGE:**

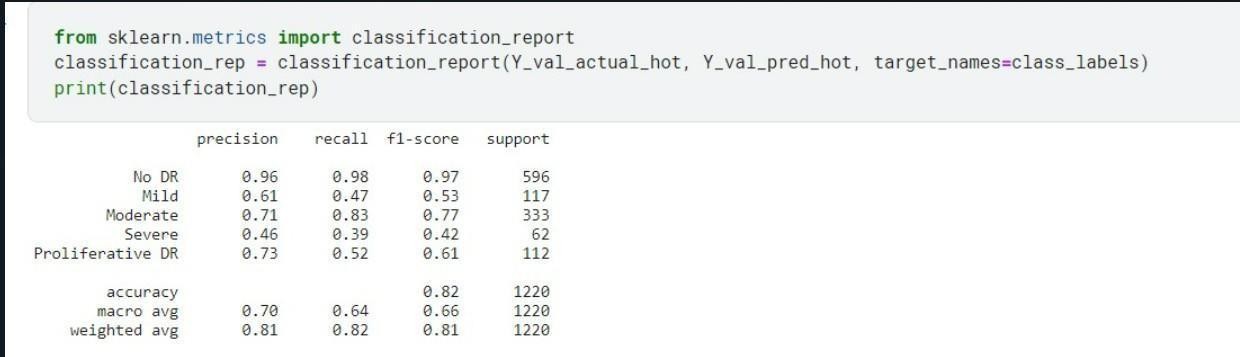
After the user provides the input retinal image, the image is fed into the DR Model. The DR Model then analyzes the image and makes predictions based on the data it has been trained on. The predicted model is then saved for further use or reference. The Flask application displays reports indicating the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



1. **PERFORMANCE TESTING**  :



1. **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



1. **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.

1. **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.

1. **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.

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

**GitHub files** [**https://github.com/aratipatil2227/ai-project**](https://github.com/aratipatil2227/ai-project%20%20%20)  **Demo Link:**

[https://drive.google.com/file/d/1fyED7kvIYQ2Q\_faGWhr3-](https://drive.google.com/file/d/1fyED7kvIYQ2Q_faGWhr3-GHfJhZkZzn1/view?usp=drive_link%20)

[GHfJhZkZzn1/view?usp=drive\_link](https://drive.google.com/file/d/1fyED7kvIYQ2Q_faGWhr3-GHfJhZkZzn1/view?usp=drive_link%20)