



**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (Autonomous)**

Approved by AICTE, New Delhi and Affiliated to JNTU-Kakinada

Accredited by NBA & NAAC with "A" Grade with a CGPA of 3.47/4.00

## **DIABETIC RETINOPATHY DETECTION USING DEEP LEARNING**

A Main Project Thesis submitted in partial fulfilment of  
requirements for the award of degree of VIII semester of

**BACHELOR OF TECHNOLOGY (B. Tech)**

In

**COMPUTER SCIENCE AND ENGINEERING (CSE)**

By

R Nitin Sai Srinivas	(Reg No. : 20131A05L6)
S. R. N. Sai Rahul	(Reg No. : 20131A05M5)
S. Sri Sai Aditya	(Reg No. : 20131A05N7)
Shaik Abdul Salam	(Reg No. : 21131A0521)

Under the esteemed guidance of

**Dr. K. Narasimha Raju**

**(Associate Professor)**

**Department of Computer Science and Engineering (CSE)**



**COLLEGE OF ENGINEERING  
(AUTONOMOUS)**

**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)**

**(Affiliated to JNTU-K, Kakinada)**

**VISAKHAPATNAM**

**2023 – 2024**



## **GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (Autonomous)**

Approved by AICTE, New Delhi and Affiliated to JNTU-Kakinada

Accredited by NBA & NAAC with "A" Grade with a CGPA of 3.47/4.00



COLLEGE OF ENGINEERING  
(AUTONOMOUS)

### **CERTIFICATE**

This is to certify that the **Main Project** work entitled **"DIABETIC RETINOPATHY DETECTION USING DEEP LEARNING"** being submitted by

**R Nitin Sai Srinivas**

**(Reg No. : 20131A05L6)**

**S. R. N. Sai Rahul**

**(Reg No. : 20131A05M5)**

**S. Sri Sai Aditya**

**(Reg No. : 20131A05N7)**

**Shaik Abdul Salam**

**(Reg No. : 21131A0521)**

in partial fulfilment of the requirements for the award of the degree **"Bachelor of Technology"** in the Department of **Computer Science and Engineering (CSE)** to the **Jawaharlal Nehru Technological University, Kakinada (JNTU-K)** is a record of bonafide work done under my guidance and supervision during **VIII semester** in the academic year **2023-2024**.

The results embodied in this record have not been submitted to any other University or Institution for the award of any Degree or Diploma.

**Signature of Project Guide**

**Dr. K. Narasimha Raju**

Associate Professor,  
Department of CSE ,  
GVPCE(A)

**I/C Head of the Department**

**Dr. D.UMA DEVI**

Associate Professor and  
Associate Head of CSE with AI&ML  
Department of CSE,  
GVPCE(A)

## **DECLARATION**

We hereby declare that this main project entitled “**DIABETIC RETINOPATHY DETECTION USING DEEP LEARNING**” is a bonafide main project work done by us and submitted to the **Department of Computer Science and Engineering (CSE) , Gayatri Vidya Parishad College of Engineering (Autonomous), Madhurawada, Visakhapatnam - 530048**, in partial fulfilment for the award of the degree of **Bachelor of Technology (B. Tech)** is of own and it is not submitted to any other university or has been published any time before.

**Place:** Visakhapatnam

**Date:**

**By**

R Nitin Sai Srinivas

(Reg No. : 20131A05L6)

S. R. N. Sai Rahul

(Reg No. : 20131A05M5)

S. Sri Sai Aditya

(Reg No. : 20131A05N7)

Shaik Abdul Salam

(Reg No. : 21131A0521)

## **ACKNOWLEDGEMENT**

We take this opportunity to thank one and all who have helped in making the project possible. We are grateful to **Gayatri Vidya Parishad College of Engineering (Autonomous)**, for giving us the opportunity to work on the **Main Project** as a part of the curriculum.

We thank our **Prof. Dr. A.B.KOTESWARA RAO, Principal, Gayatri Vidya Parishad College of Engineering (Autonomous)** for extending his utmost support and cooperation in providing all the provisions for the successful completion of the main project.

We consider it as our privilege to express our deepest gratitude to **Dr. D.UMA DEVI, Associate Professor and Associate Head of CSE with AL & ML and I/C Head of the Department**, for her valuable suggestions and constant motivation that greatly helped the main project work to get successfully completed as per the schedule.

Our sincere gratitude to our **guide Dr. K . NARASIMHA RAJU, Department of Computer Science and Engineering**, for his continuous assessment and valuable guidance. Throughout the project period, he provided immense encouragement and sound advice for every member in the team for efficient team performance.

We also sincerely thank our coordinator, **Dr. CH.SITA KUMARI, Associate Professor and Main Project Coordinator, Department of Computer Science and Engineering**, for her kind suggestions in successful completion of this main project work.

Finally, we take this opportunity to thank all the people who helped us in completion of main project work, directly or indirectly, and for their timely encouragement and faithful services.

**By**

R Nitin Sai Srinivas	(Reg No. : 20131A05L6)
S. R. N. Sai Rahul	(Reg No. : 20131A05M5)
S. Sri Sai Aditya	(Reg No. : 20131A05N7)
Shaik Abdul Salam	(Reg No. : 21131A0521)

## **ABSTRACT**

In the Present era, every one is prone to get affected by diseases. The most frequently happening is loss of sight among people who were diabetic. Diagnosing diabetic retinopathy manually with the assistance of an ophthalmologist has been a time-consuming and labor-intensive technique. The existing Machine learning models are not efficient when it comes to real time usage. The proposed model not only detects diabetic retinopathy but also analyses distinct severe phases, which is accomplished using Deep Learning (DL) and transfer learning methods. This approach enhances the model's ability to accurately classify and diagnose diabetic retinopathy by learning relevant hierarchical features from a diverse set of retinal images(fundus images). Images are being trained to automatically recognize which stage of DR has progressed.

### **Keywords :**

- Diabetic Retinopathy, Opthamologist , Deep Learning Model, Retina, Diagnostic process , Hierarchial features, Fundus images

## **INDEX**

<b>COVER PAGE</b>	<b>I</b>
<b>CERTIFICATE</b>	<b>II</b>
<b>DECLARARTION</b>	<b>III</b>
<b>ACKNOWLEDGEMENT</b>	<b>IV</b>
<b>ABSTRACT</b>	<b>V</b>
<b>INDEX</b>	<b>VI - VIII</b>
<b>CHAPTER 01. INTRODUCTION</b>	<b>01</b>
1.1    Objective	01
1.2    About the Algorithm	02
1.2.1    Algorithm steps for CNN	02
1.2.2    DenseNet201 Architecture	04
1.2.3    InceptionV3 & MobileNetV2 Architecture	07
1.3    Purpose	06
1.4    Scope	07
<b>CHAPTER 02. SRS DOCUMENT</b>	<b>08</b>
2.1    Functional Requirements	08
2.2    Non-functional Requirements	08
2.3    Minimum Hardware Requirements	09
2.4    Minimum Software Requirements	09
<b>CHAPTER 03. SYSTEM ANALYSIS</b>	<b>10</b>
3.1    Existing System	10
3.1.1    Drawbacks of Existing Systems	10
3.2    Proposed System	10
3.2.1    Advantages of Proposed System	11
3.3    Feasibility Study	11
3.3.1    Economic Feasibility	12
3.3.2    Technical Feasibility	12
3.3.3    Operational Feasibility	13

3.4	Cost-Benefit Analysis	13
<b>CHAPTER 04. SOFTWARE DESCRIPTION</b>		<b>14</b>
4.1	Google Colab	14
4.2	Stream lit	14
4.3	Keras	15
4.4	TensorFlow	15
4.5	Pillow	15
4.6	Matplotlib	16
4.7	NumPy	16
4.8	Pandas	16
4.9	Python IDLE	16
4.10	Visual Studio (VS) Code	16
4.11	Scikit-Learn	17
4.12	PyQt5	17
<b>CHAPTER 05. PROJECT DESCRIPTION</b>		<b>18</b>
5.1	Problem Definition	18
5.2	Project Overview	18
5.3	Module Description	19
5.3.1	Stream lit Framework	19
5.3.2	PyQt5 Framework	19
5.3.3	Model	19
5.3.3.1	Deep Learning	19
5.3.3.2	CNN	20
5.3.3.3	DenseNet201 Connections	20
5.3.3.3	InceptionV3 Connections	20
5.3.3.3	MobileNetV2 Connections	20

<b>CHAPTER 06. SYSTEM DESIGN</b>	<b>22</b>
<b>6.1</b> Introduction To UML	22
<b>6.1.1</b> Why we need UML	22
<b>6.1.2</b> Characteristics of UML	22
<b>6.2</b> Building Blocks Of the UML	23
<b>6.2.1</b> Things in the UML	24
<b>6.2.2</b> Relationships in the UML	26
<b>6.3</b> UML Diagrams	28
<b>6.3.1</b> Class Diagram	29
<b>6.3.2</b> Flowchart	30
<b>6.3.3</b> Use Case Diagram	31
<b>6.3.4</b> Activity Diagram	32
<b>CHAPTER 07. DEVELOPMENT</b>	<b>33</b>
<b>7.1</b> Data Set Used	33
7.1.1    Data Set Images	34
<b>7.2</b> Sample Code	36
<b>7.3</b> Results and Discussions	47
<b>CHAPTER 08. TESTING</b>	<b>50</b>
<b>8.1</b> Introduction to Testing	50
<b>8.1.1</b> Importance of Testing	50
<b>8.1.2</b> Benefits of Testing	50
<b>8.1.3</b> Different Types of Testing	51
<b>8.2</b> Test Code	53
<b>8.3</b> Test Case and Outputs	55
<b>CHAPTER 09. CONCLUSION</b>	<b>58</b>
<b>CHAPTER 10. FUTURE SCOPE</b>	<b>59</b>
<b>CHAPTER 11. BIBLIOGRAPHY</b>	<b>60</b>



## **1. INTRODUCTION**

Diabetic retinopathy, a complication arising from diabetes, presents a significant threat to vision health globally. Its pathophysiology involves damage to the delicate blood vessels of the retina due to prolonged high blood sugar levels. This damage initiates a cascade of events, including leakage of fluid or blood and the development of abnormal blood vessels. In its early stages, diabetic retinopathy may manifest with subtle symptoms or none at all, making detection challenging. However, as the condition progresses, symptoms such as blurred vision, floaters, and impaired color vision may become pronounced, signaling the need for prompt intervention. The progression to proliferative diabetic retinopathy (PDR) can lead to severe complications such as retinal detachment and ultimately blindness if left untreated.

Detection of diabetic retinopathy has been revolutionized by the integration of deep learning techniques with imaging modalities such as fundus photography and optical coherence tomography (OCT). These technologies enable the automated analysis of retinal images, allowing for the detection of characteristic abnormalities associated with diabetic retinopathy, including microaneurysms, haemorrhages, and fluid accumulation. Additionally, AI-based screening systems facilitate population-wide screening efforts, triaging patients based on the severity of their condition and ensuring timely intervention for those at risk of vision loss. Furthermore, the fusion of information from multiple imaging modalities enhances the accuracy of diagnosis and staging, providing clinicians with valuable insights into disease progression and treatment planning.

Despite the promise of deep learning in diabetic retinopathy detection, several challenges persist, including the need for large, annotated datasets for model training and the lack of interpretability in AI-based predictions. Efforts are underway to address these challenges through the development of explainable AI techniques and strategies for improving the generalizability of models across diverse patient populations. By overcoming these obstacles, deep learning-based approaches have the potential to revolutionize the early detection and management of diabetic retinopathy, ultimately preserving vision and improving outcomes for patients worldwide.

## **1.1 OBJECTIVE**

The project's primary objective is to precisely develop a robust deep learning model using convolutional neural networks (CNNs) to accurately detect and classify diabetic retinopathy (DR) from retinal images. The model should effectively identify various stages of DR, including mild, moderate, severe, and No DR stages, with high precision and sensitivity.

The secondary objective of the project is to integrate the deep learning model into a user-friendly web application interface, allowing healthcare professionals to upload retinal images for rapid and accurate DR diagnosis. All these requirements of the user are to be satisfied in an efficient and simple way so as to provide maximum user satisfaction.

## **1.2 ABOUT THE ALGORITHM**

Convolution Neural Network (CNN) is the algorithm used in this project for detecting diabetic retinopathy. We have used DenseNet201, InceptionV3 and MobileNetV2 architecture. In Deep learning CNN is a class of artificial neural networks, Most commonly applied to analyze visual imagery. CNN works by extracting features from images. A Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assigns importance (learnable weights and biases) to various aspects/objects in the image and then makes them to differentiate from one other. Given a training dataset then CNN, unlike traditional machine learning techniques, optimizes the weights and filter parameters in the hidden layers to generate features suitable to solve the classification problem.

### **1.2.1 Algorithm steps for CNN :**

1. a) Convolution Operation
  - b) Re LU Layer
2. Pooling
3. Flattening
4. Full Connection

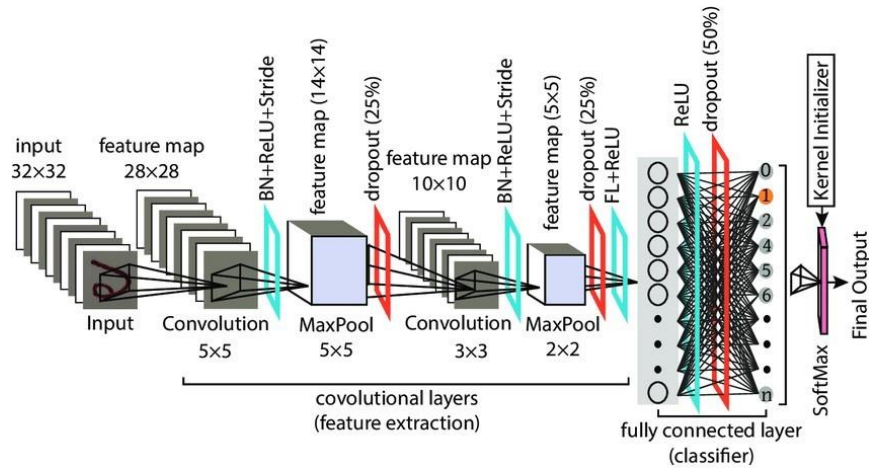


Fig 1.1 Internal Block of CNN

1. a) Convolution Operation :

- The first building block in our plan of attack is convolution operation. In this step, we will touch on feature detectors, which basically serve as the neural network's filters. We will discuss feature maps, learning the parameters of such maps, how patterns are detected, the layers of detection and how the findings are mapped out. [6]
- Extract the unique features from the input image. Convolution is a mathematical operation on two functions ( $f$  and  $g$ ) that produces a third function ( $f*g$ ) expressing how the shape of one is modified by the other. [6]

b) ReLU Layer :

- ReLU (Rectified Linear Unit) Layer.
- ReLU refers to the Rectifier Unit, the most deployed activation function for the outputs of the CNN neurons.

2. Pooling :

- Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer to generate a pooled feature map.
- The pooling layer uses various filters to identify different parts of the image like edges, corners, body, feathers, eyes, and beak. [4]

### 3. Flattening :

- Flattening is converting the data into a 1-D array for inputting it to next layer.
- We flatten the output of the convolution layers to create a single long feature vector and it is connected to the final classification model.
- The flattened matrix is fed as input to connected layer to classify the image. [4]

### 4. Full Connection :

- Fully Connected Layers from the last few layers in the network.
- The input to the fully connected layer is the output from final pooling or Convolution layer. [6]

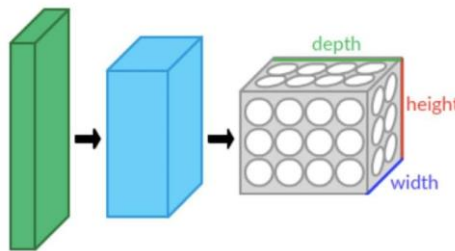


Fig 1.2 CNN layers arranged in 3 dimensions

#### 1.2.2 DenseNet 201 Architecture

DenseNet, short for Densely Connected Convolutional Networks, is a type of convolutional neural network (CNN) architecture proposed by Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger in their paper titled "Densely Connected Convolutional Networks" in 2017.

DenseNet introduces the concept of dense connectivity between layers, where each layer is connected to every other layer in a feed-forward fashion. Unlike traditional CNN architectures where each layer is connected only to its subsequent layer, DenseNet allows for direct connections from any layer to all subsequent layers. This dense connectivity pattern facilitates feature reuse, strengthens feature propagation, and encourages gradient flow during training.

DenseNet-201 stands out as a prominent variant of the DenseNet architecture, recognized for its remarkable depth and extensive layer count. In the realm of convolutional neural networks (CNNs), its introduction marked a significant advancement, offering a solution that prioritizes feature reuse and information flow throughout the network. This architectural innovation stems from the fundamental concept of dense connectivity, where each layer establishes direct connections with every other layer in a feed-forward manner. This unique characteristic fosters a robust framework for learning intricate representations of input data.

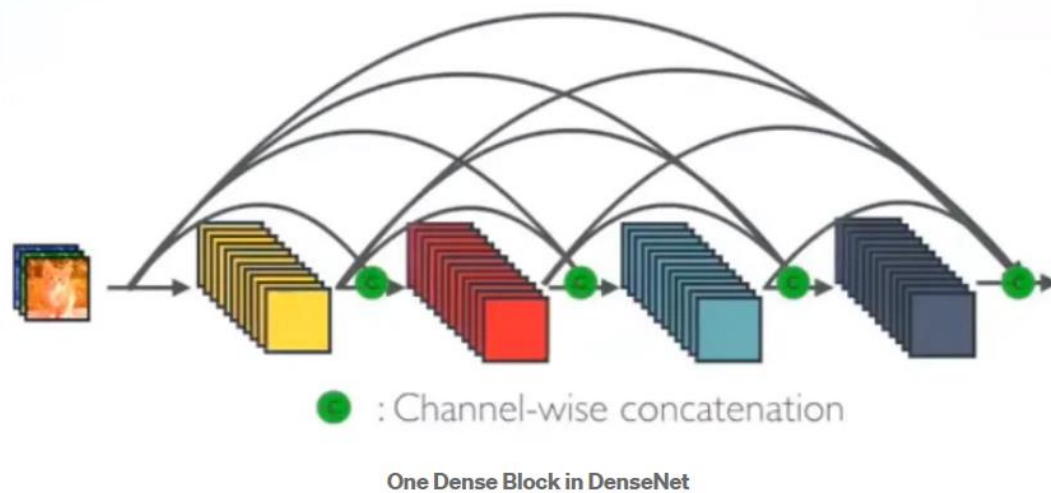


Fig 1.3 Connections of Block in DenseNet Architecture

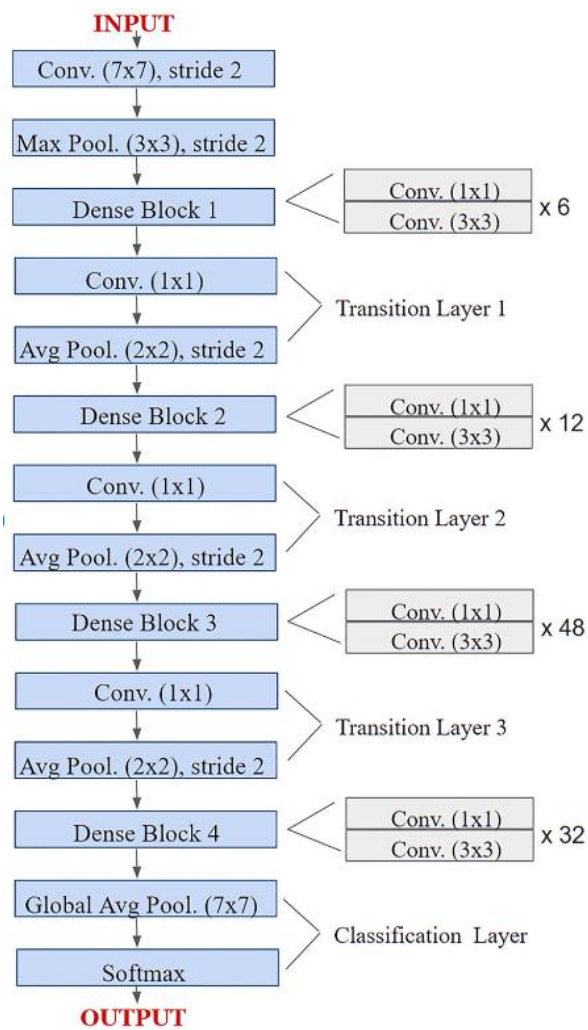


Fig 1.4 DenseNet-201 Architecture with respective layers

DenseNet-201 is a deep convolutional neural network architecture designed for computer vision tasks. It employs dense connectivity, facilitating strong feature propagation and reuse. The network starts with a 7x7 convolutional block with 64 kernels, followed by max pooling. It then consists of three dense blocks, each with nine layers, interspersed with transition layers containing 1x1 convolutions and average pooling. The complexity increases with denser blocks, featuring 12 repetitions of 1x1 and 3x3 convolutions. Transition layers persist throughout, reducing spatial dimensions. This cycle repeats twelve times, resulting in 48 dense blocks and transition layers. The network ends with global average pooling, a fully connected layer with 1000 nodes, and a SoftMax activation for output. DenseNet-201 comprises 201 layers, offering robust capabilities for tasks like image classification and object localization, with versatility for broader applications beyond computer vision.

Totalling the layers: 1 (initial convolution block) + 9 \* 3 (dense blocks) + 9 \* 3 (transition layers after dense blocks) + 12 \* 3 (dense blocks with increased kernel size) + 12 \* 3 (transition layers after dense blocks with increased kernel size) + 48 \* 3 (dense blocks with further increased kernel size) + 48 \* 3 (transition layers after dense blocks with further increased kernel size) + 3 (final layers) = 201 layers in DenseNet-201 architecture.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Fig 1.5 DenseNet Architecture with different variations

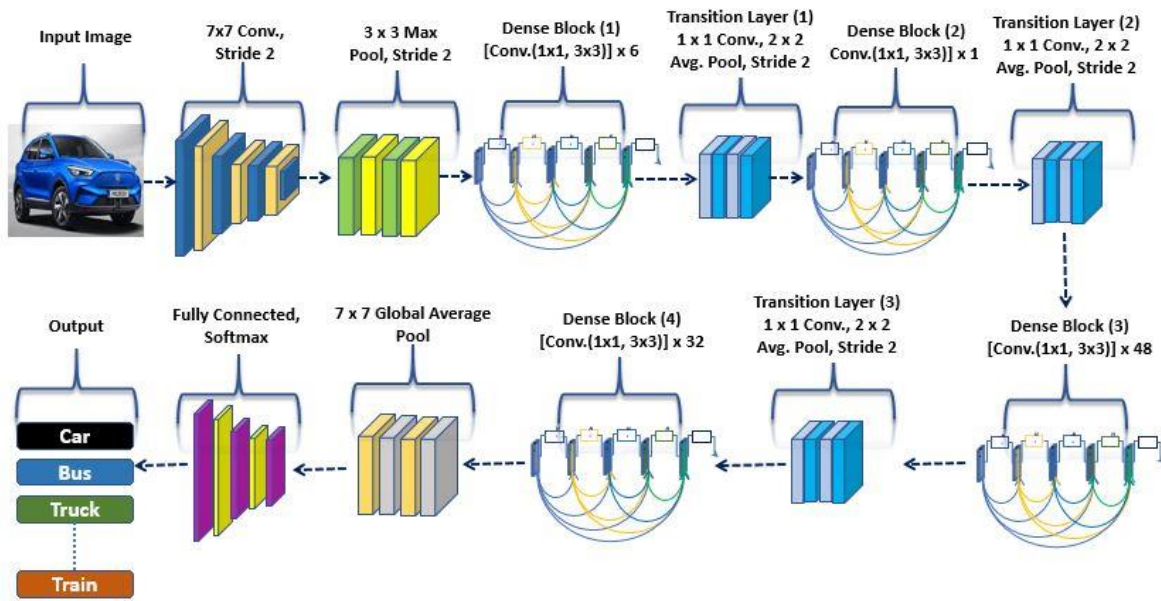


Fig 1.6 Depicting the concept of DenseNet CNN

A distinguishing feature of DenseNet-201 lies in its substantial depth, comprising a total of 201 layers. Such depth enables the network to encapsulate and distill increasingly complex features from the input data, thereby enhancing its capacity to discern subtle patterns and nuances. Within this depth, DenseNet-201 leverages a combination of bottleneck layers strategically integrated throughout the architecture. These bottleneck layers serve to mitigate computational complexity by employing a mix of 1x1 convolutional layers for dimensionality reduction and 3x3 convolutional layers for capturing spatial information.

### 1.2.2 Inception v3 & MobileNetV2

Inception v3 represents a significant milestone in the evolution of the Inception convolutional neural network (CNN) architecture, pioneered by Google in 2015. Building upon the successes of its predecessors, this iteration aims to push the boundaries of efficiency and performance in the realm of computer vision tasks.

Factorization-Based Inception Module is One of the standout features of Inception v3 is its adoption of Factorization-Based Inception Modules. These modules employ sophisticated factorization techniques to streamline the network's parameter structure while preserving its expressive capacity. By reducing redundancy and promoting feature reuse, this innovation facilitates the creation of deeper networks with fewer parameters, resulting in enhanced computational efficiency and scalability.

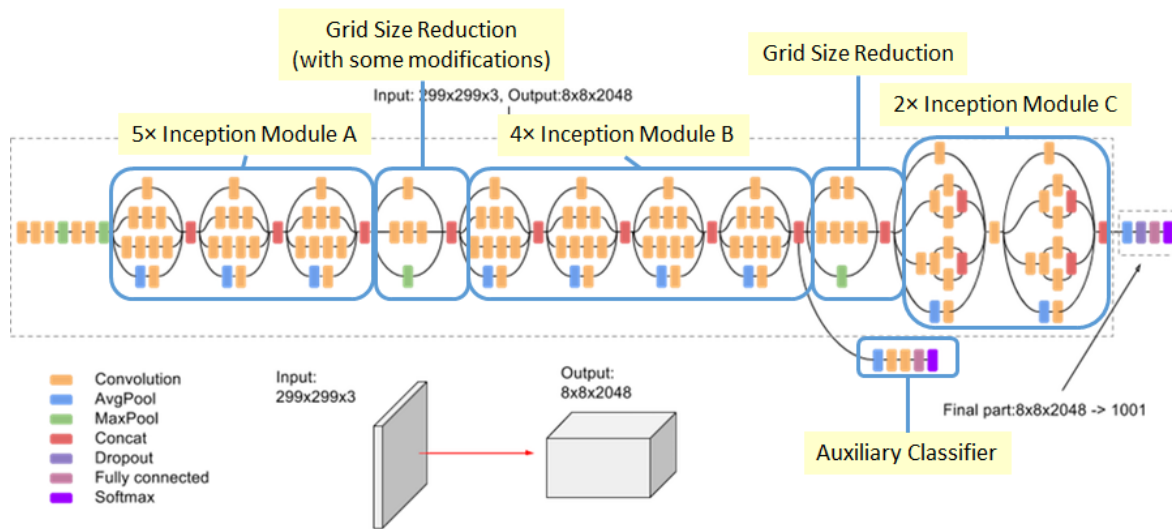


Fig 1.7 Depicting the concept of InceptionV3

Integration of Batch Normalization Continues the trend set by earlier versions, Inception v3 incorporates batch normalization layers strategically positioned after each convolutional operation. This inclusion plays a pivotal role in stabilizing the network's training process by normalizing activations within mini-batches. The resultant improvement in gradient flow and convergence dynamics contributes to superior training efficiency and model performance.

Achievement of State-of-the-Art Performance: Inception v3 has garnered acclaim for its exceptional performance on benchmark datasets such as ImageNet. Through meticulous design and rigorous experimentation, the architecture achieves state-of-the-art results, surpassing previous benchmarks in terms of accuracy, speed, and computational efficiency. This achievement underscores the efficacy of its architectural enhancements and underscores its relevance in contemporary research and applications.

Versatility Across Applications: Inception v3's versatility extends beyond its prowess in image classification, encompassing a broad spectrum of computer vision tasks. From object detection and localization to semantic segmentation and scene understanding, the architecture demonstrates robust performance across diverse domains. Its adaptability and generalization capabilities make it a sought-after choice for real-world applications spanning autonomous driving, medical imaging, and beyond.



MobileNetV2 is a convolutional neural network architecture tailored for efficient deployment on mobile and embedded devices, introduced by Google researchers in 2018. Its design focuses on enhancing the original MobileNet model's efficiency while maintaining high accuracy. The architecture incorporates depthwise separable convolutions, linear bottlenecks, and inverted residuals with linear bottlenecks. Depthwise separable convolutions split standard convolutions into depthwise and pointwise convolutions, reducing parameters and computational cost. Inverted residuals involve expanding features with a lightweight  $1 \times 1$  convolution, applying depthwise convolution, and projecting features back with another  $1 \times 1$  convolution, enabling efficient representation learning. MobileNetV2 introduces hyperparameters like width multiplier and resolution multiplier, allowing flexibility in balancing accuracy and model size. Width multiplier scales the number of channels in each layer, while resolution multiplier scales down the input resolution. Despite its efficiency, MobileNetV2 achieves state-of-the-art accuracy in image classification tasks, making it ideal for resource-constrained devices such as mobile phones, embedded systems, and IoT devices. It finds applications in various domains, including image classification, object detection, and semantic segmentation. MobileNetV2 implementations are available in popular deep learning frameworks like TensorFlow, PyTorch, and Keras, facilitating its adoption by researchers and developers in their projects.

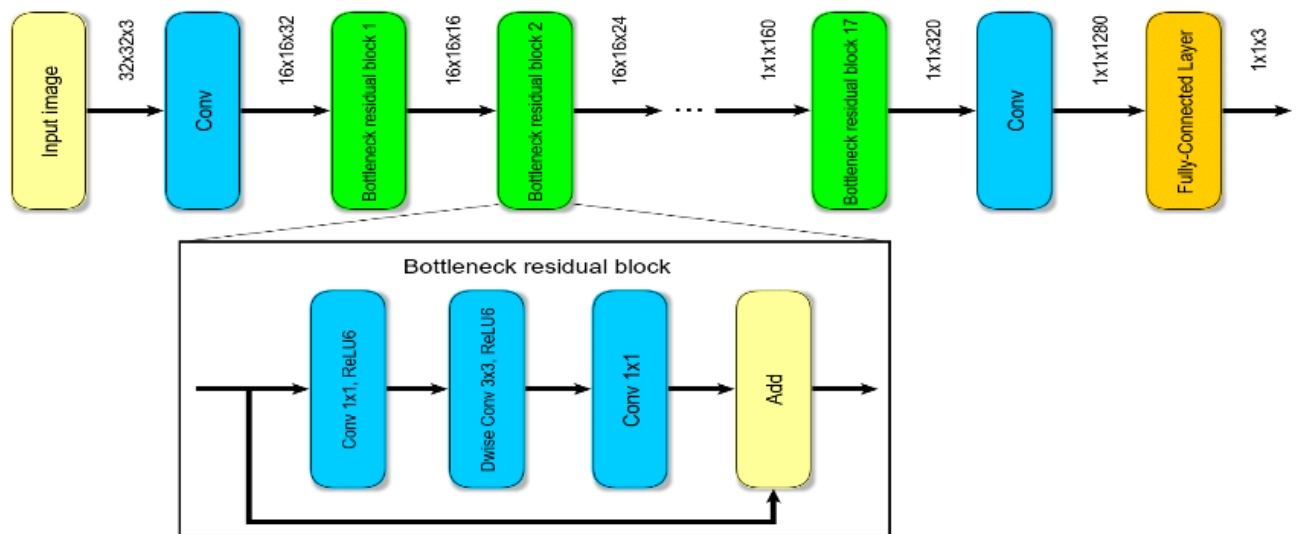


Fig 1.8 Depicting the concept of MobileNetV2

### **1.3 PURPOSE**

In today's generation, many individuals lack sufficient knowledge about diabetic retinopathy and its early signs, making timely detection a challenge. Manual diagnosis by specialists is time-consuming and prone to errors, exacerbating the issue. While traditional methods such as fundus photography or optical coherence tomography (OCT) are available for detecting diabetic retinopathy, they may not always be easily accessible or affordable for all individuals. There's a pressing need for a user-friendly application that can accurately detect diabetic retinopathy from retinal images and offer actionable insights for further medical intervention. Our goal is to develop an intuitive, efficient, and portable application powered by Deep Learning (DL) techniques, capable of automatically identifying diabetic retinopathy from retinal images. This solution aims to democratize access to early detection and expert consultation, ensuring equitable distribution of healthcare resources..

### **1.4 SCOPE**

Even with the plethora of technological advancements in medical imaging, diagnosing diabetic retinopathy remains challenging due to various constraints. Many individuals have limited understanding of retinal imaging techniques and may struggle to interpret the severity of diabetic retinopathy. Moreover, even experts can make errors during manual analysis, leading to delays in diagnosis and treatment. Hence, there is an urgent need for a web application that leverages deep learning to automatically detect and categorize diabetic retinopathy from retinal images. This application should not only identify the presence of retinal abnormalities.

In this project, the DenseNet201, InceptionV3 architecture and MobileNetV2, based on CNN was explored to automatically detect the presence of diabetic retinopathy in the retina in fundus image uploaded by the user. This application will be a great substitute for detecting abnormalities in eyes like retinopathy within no time and also ensure faster access as well as efficient and useful suggestions to the user for further diagnosis through expert consultation.

The primary goal for future work will be developing a complete web application that encompasses all possibilities for detecting diabetic retinopathy abnormalities from any fundus image of any quality with clear accuracy.

## **2. SRS DOCUMENT**

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill the needs of all stakeholders (business, users). These include the high-level business requirements dictating the goal of the project, end-user requirements and needs, and the product's functionality in technical terms. To put it simply, an SRS provides a detailed description of how a software product should work and how your development team should make it work.

### **2.1 FUNCTIONAL REQUIREMENTS**

A FUNCTIONAL REQUIREMENT (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements are also called Functional Specification.

- Inputting the image from the user by uploading it in the web application.
- Upon image upload, the deep learning algorithm should process the retinal images to detect any abnormalities indicative of diabetic retinopathy, such as microaneurysms, hemorrhages, and exudates.
- The algorithm should classify the detected abnormalities into different stages of diabetic retinopathy based on learned patterns from the dataset.
- If abnormalities indicative of diabetic retinopathy are detected, the application should provide a recommendation for expert consultation with an ophthalmologist.
- The outputs that are generated are displayed on the web application.

### **2.2 NON-FUNCTIONAL REQUIREMENTS**

NON-FUNCTIONAL REQUIREMENTS (NFR) specify the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability. Nonfunctional requirements are called qualities of a system, there are as follows:

Performance - The average response time of the system is less in most of the cases but in a few cases like if the input data is too high then more generation of solutions takes place eventually it takes time to generate the solution.

Reliability - The system is highly reliable as it outputs the accurate landmark name and details along with the location the map (if necessary).

Operability - The web application is safe to use, and data does not get deleted unless the user explicitly deletes the data.

Efficiency - Once the user has learned how to use the web application, the user can efficiently perform the required task.

Understandability - As the web application interface is very simple and user-friendly. The users find it easy to use, and it can be understandable.

### **2.3 MINIMUM HARDWARE REQUIREMENTS**

- Processor – 1) Intel Core i5 or more, OR  
2) AMD Ryzen 4000 series or more
- Storage – 1) Hard Disk – 512GB or more, OR  
2) SSD – 256 GB or more
- RAM - 4GB (minimum)
- Mobile Phone with Internet Access

### **2.4 MINIMUM SOFTWARE REQUIREMENTS**

Python-based Stream lit Framework (Open Source) for the development of the Web Application with a clean UI and Easy Access for the Knee Osteoarthritis detection.

- Operating System : Windows, MacOS, Linux
- Programming Language : Python
- Python Version : 3 or higher
- IDE : Visual Studio (VS) Code, Google Colab
- Supported Web Browser : Edge , Chrome , Firefox , Brave
- Packages used (with Version) :

- ☐ TensorFlow >= 2.5.0
- ☐ Keras >= 2.4.3
- ☐ NumPy >= 1.19.5
- ☐ Matplotlib >= 3.3.4
- ☐ Stream lit >= 1.20.0

### **3. SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEMS**

Diabetic retinopathy detection utilize conventional Convolutional Neural Networks (CNN) as a single base model for the classification of retinal images. These systems typically involve preprocessing steps such as image resizing and normalization to optimize the input data for model training and evaluation. Once trained, the CNN model is deployed in healthcare settings, enabling automated detection and early intervention of diabetic retinopathy. These systems leverage the power of deep learning to analyze retinal images accurately, assisting healthcare professionals in diagnosing and managing diabetic retinopathy efficiently. Additionally, the deployment of such models facilitates timely intervention, ultimately improving patient outcomes and reducing the risk of vision loss associated with diabetic retinopathy.

##### **3.1.1 DRAWBACKS OF EXISTING ALGORITHMS**

- Existing system lacks accuracy.
- This is due to usage of poor preprocessing techniques and cnn model architecture.
- The dataset available on various platform may contain unbalanced data i.e the frequency of one class is majority over the others.
- This also leads to high chances of increase in error rates while generalizing to new data..
- If the data distribution changes significantly after the model is trained,the model might become less effective because it has been optimized for the characteristics of the initial distribution.

#### **3.2 PROPOSED SYSTEM**

Diabetic retinopathy detection system aims to achieve an acceptable level of accuracy by leveraging model ensembling techniques. The idea involves combining the predictions of multiple pretrained models using ensemble learning. Initially, we will identify two or three best-performing models as a base and further fine-tune them to enhance their performance for diabetic retinopathy detection. Subsequently, a strategy will be devised to combine the predictions from these models using common methods such as average voting, where each model's prediction is weighted according to its performance. By employing model ensembling,

the proposed system seeks to enhance accuracy, robustness, and reliability in diabetic retinopathy detection, ultimately improving patient outcomes and facilitating timely intervention and treatment.

### **3.2.1 ADVANTAGES OF PROPOSED SYSTEM**

- Model ensembling improves performance by combining multiple models' strengths.
- It enhances robustness to noisy data and outliers.
- It helps mitigate overfitting by capturing diverse aspects of the data.
- It is highly accurate and very much reliable.

### **3.3 FEASIBILITY STUDY**

A feasibility study is an analysis that takes all a project's relevant factors into account including economic, technical, legal, and scheduling considerations to ascertain the likelihood of completing the project successfully. A feasibility study is important and essential to evaluate any proposed project is feasible or not. A feasibility study is simply an assessment of practicality of proposed project.

The main objectives of feasibility are mentioned below:

To determine if the product is technically and financially feasible to develop, is the main aim of the feasibility study. A feasibility study should provide management with enough information to decide:

- Whether the project can be done.
- To determine how successful your proposed action will be.
- Whether the final product will benefit its intended users.
- To describe the nature and complexity of the project.
- What are the alternatives among which a solution will be chosen.
- To analyze if the software meets organizational requirements.

There are various types of feasibility that can be determined. They are:

Operational – Define the urgency of the problem and the acceptability of any solution, includes people-oriented and social issues: internal issues, such as manpower problems, labor objections, manager resistance, organizational conflicts, and policies; also, external issues, including social acceptability, legal aspects, and government regulations.

Technical - Is the feasibility within the limits of current technology? Does the technology exist at all? Is it available within a given resource?

Economic - Is the project possible, given resource constraints? Are the benefits that will accrue from the new system worth the costs? What will be the savings that will result from the system, including tangible and intangible ones? What are the development and operational costs?

Schedule - Constraints on the project schedule and whether they could be reasonably met.

### **3.3.1 Economic Feasibility**

Economic analysis could also be referred to as cost/benefit analysis. It is the most frequently used method for evaluating the effectiveness of a new system. In economic analysis the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. Economic feasibility study related to price, and all kinds of expenditure related to the scheme before the project starts. This study also improves project reliability. It is also helpful for the decisionmakers to decide the planned scheme processed latter or now, depending on the financial condition of the organization. This evaluation process also studies the price benefits of the proposed scheme.

Economic Feasibility also performs the following tasks.

- Cost of packaged software/ software development.
- Cost of doing full system study.
- Is the system cost Effective?

### **3.3.2 Technical Feasibility**

A large part of determining resources has to do with assessing technical feasibility. It considers the technical requirements of the proposed project. The technical requirements are



then compared to the technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirements. The analyst must find out whether current technical resources can be where the expertise of system analysts is beneficial, since using their own experience and their contact with vendors they will be able to answer the question of technical feasibility.

Technical Feasibility also performs the following tasks.

- Is the technology available within the given resource constraints?
- Is the technology have the capacity to handle the solution
- Determines whether the relevant technology is stable and established.
- Is the technology chosen for software development has a large number of users so that they can be consulted when problems arise, or improvements are required?

### **3.3.3 Operational Feasibility**

Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility refers to the availability of the operational resources needed to extend research results beyond on which they were developed and for which all the operational requirements are minimal and easily accommodated. In addition, the operational feasibility would include any rational compromises farmers make in adjusting the technology to the limited operational resources available to them.

The Operational Feasibility also perform the tasks like :

- Does the current mode of operation provide adequate response time?
- Does the current of operation make maximum use of resources.
- Determines whether the solution suggested by the software development team is acceptable.
- Does the operation offer an effective way to control the data?

Our project operates with a processor and packages installed are supported by the system.

### **3.4 COST BENEFIT ANALYSIS**

The financial and the economic questions during the preliminary investigation are verified to estimate the following:

- The cost of the hardware and software for the class of application being considered.
- The benefits in the form of reduced cost.
- The proposed system will give the minute information, as a result.
- Performance is improved which in turn may be expected to provide increased profits.
- This feasibility checks whether the system can be developed with the available funds.
- This can be done economically if planned judiciously, so it is economically feasible.
- The cost of the project depends upon the number of man-hours required.

## **4. SOFTWARE DESCRIPTION**

### **4.1 Google Colab**

Google Colab, short for Google Colaboratory, is a cloud-based platform provided by Google that offers free access to computing resources, including CPU, GPU, and TPU. It provides an interactive environment similar to Jupyter Notebooks, facilitating Python code writing and execution collaboratively in real-time. Colab seamlessly integrates with Google Drive, allowing users to import datasets, save notebooks, and share access for collaboration. Pre-installed with popular Python libraries such as TensorFlow and PyTorch, Colab eliminates the need for manual installations. It supports version control using Git, enabling tracking changes and collaboration. Colab is highly customizable, allowing users to configure runtime settings, including CPU, GPU, or TPU selection, memory, disk resources, and Python runtime environment specifications. This platform accelerates machine learning tasks like deep neural network training through free GPU and TPU access. Overall, Google Colab is invaluable for researchers, students, and developers, offering a versatile and efficient environment for prototyping, experimentation, data analysis, and collaborative work.

### **4.2 Stream lit**

Stream lit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Stream lit is a promising open-source Python library, which enables developers to build attractive user interfaces in no time. Data scientists or machine learning engineers are not web developers and they are not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can display data and collect needed parameters for modeling. The best thing about Stream lit is that you do not even need to know the basics of web development to get started.

Stream lit is the easiest way especially for people with no front-end knowledge to put their code into a web application:

- No front-end (HTML, JS, CSS) experience or knowledge is required.
- You do not need to spend days or months to create a web app, you can create a really beautiful machine learning or data science app in only a few hours or even minutes.
- It is compatible with the majority of Python libraries (e.g., pandas, matplotlib, seaborn, plotly, Keras, PyTorch, SymPy(latex)).
- Less code is needed to create amazing web apps.
- Data caching simplifies and speeds up computation pipelines.

### **4.3 Keras**

Keras is an open-source neural-network library written in Python. It can run on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or Plaid ML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open- ended Neuro- Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the Xception deep neural network model. In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. It offers a higher- level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the backend used.

### **4.4 TensorFlow**

TensorFlow is an open-source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research fascinating ideas on artificial intelligence, the Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy- to-understand framework. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.

### **4.5 Pillow**

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and powerful image processing capabilities. The core image library is designed for fast access to data stored

in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

To read an image with Python Pillow library, should follow these steps.

- Import Image from PIL library.
- Use Image. open () method and pass the path to the image file as argument. Image. open ()
- returns an Image object. You can store this image object and apply image operations on it.

#### **4.6 Matplotlib**

Matplotlib is a plotting library for python. Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. Matplotlib comes with a wide variety of plots. Plots help to understand trends, patterns, and to make correlations. It consists of various plots like line, bar, scatter etc. pyplot () is the most important function in matplotlib library, which is used to plot 2D data.

#### **4.7 NumPy**

NumPy stands for Numerical Python. NumPy is a python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open- source project, and you can use it Freely. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. It is optimized to work with the latest CPU architectures. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

#### **4.8 Python IDLE**

An IDE or Integrated Development Environment is software that allows users to write, run, and test codes in a simple manner. There are many IDEs for running python like PyCharm, Spyder, Visual Studio, etc. The Python installer for Windows contains the IDLE module by default.

IDLE is not available by default in Python distributions for Linux. It needs to be installed using the respective package managers.

#### **4.9 Visual Studio (VS) Code**

Visual Studio Code is a source code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, python, and C++. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink Layout Engine. Visual Studio Code employs the same editor component used in Azure DevOps.

- Visual studios are available for Windows, Linux, and Mac OS.
- Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON,
- It can detect if any snippet of code is left incomplete. Also, common variable syntaxes and variable declarations are made automatically.

#### **4.10 Scikit-Learn**

An open-source Python package to implement machine learning models in Python is called Scikitlearn. This library supports modern algorithms like KNN, random forest, XG Boost, and SVC. It is constructed over NumPy. Both well-known software companies and the Kaggle competition frequently employ Scikit-learn. Sc-learn aids in various processes of model building, like model selection, regression, classification, clustering, and dimensionality reduction (parameter selection).

#### **4.11 Seaborn**

Seaborn is a Python data visualization library used for making statistical graphs. While the library can make any number of graphs, it specializes in making complex statistical graphs beautiful and simple. The library is meant to help you explore and understand your data. Because data in Python often comes in the form of a Pandas Data Frame, Seaborn integrates nicely with Pandas. However, it provides very much high-level functions to help you easily produce consistently attractive visualizations.

## **5. PROJECT DESCRIPTION**

### **5.1 PROBLEM DEFINITION**

In the modern landscape of healthcare, diabetic retinopathy stands as a prevalent concern, posing a significant risk of vision loss among diabetic individuals. However, the conventional method of diagnosing this condition with the aid of ophthalmologists is characterized by its time-consuming nature, demanding extensive labor and resources. Moreover, existing machine learning models, although available, exhibit inefficiencies when applied to real-time scenarios, creating a pressing need for more effective diagnostic tools.

To address these challenges, a proposed model harnesses the power of Deep Learning (DL) and transfer learning techniques. This approach is underpinned by its ability to glean relevant hierarchical features from a diverse array of retinal images, particularly fundus images, thereby enhancing its diagnostic accuracy and efficacy.

Crucially, the proposed model's capacity to automatically discern the progression stages of diabetic retinopathy signifies a significant stride forward in early detection and intervention.

## **5.2 PROJECT OVERVIEW**

The Diabetic Retinopathy Detection project harnesses the power of deep learning techniques to revolutionize the early diagnosis and management of diabetic retinopathy. Leveraging Convolutional Neural Networks (CNNs) with a focus on the ResNet architecture, this project aims to provide accurate and timely detection of diabetic retinopathy through analysis of retinal images. By curating a comprehensive dataset annotated for various stages of the disease, the CNN model undergoes meticulous training and validation to ensure optimal performance and generalizability across diverse patient populations.

In addition to the development of the CNN model, the project prioritizes user accessibility and convenience by implementing a user-friendly web application interface. Utilizing the Streamlit framework, the interface allows for seamless uploading of retinal images and provides prompt, detailed results. This empowers healthcare professionals with valuable insights for timely intervention and management, ultimately improving patient outcomes and reducing the risk of vision loss associated with diabetic retinopathy. Overall, the project represents a significant advancement in the field of diabetic retinopathy detection, offering a scalable and efficient solution for early diagnosis and intervention.

The steps involved in our project are:

1. Gather a dataset consisting of retinal images of patients with and without diabetic retinopathy.
2. Prepare the collected data by preprocessing the images and visualizing them to ensure data quality.
3. Divide the dataset into training, validation, and testing sets, typically with a split of 70% for training, 10% for validation, and 20% for testing.
4. Design and train a deep learning model, such as a convolutional neural network (CNN), using the training dataset to detect signs of diabetic retinopathy in retinal images.
5. Validate the trained model using the validation dataset to assess its performance and fine-tune hyperparameters if necessary.
6. Iterate over steps 2-5 with different configurations, such as varying the architecture of the deep learning model or adjusting preprocessing techniques, to maximize the accuracy of diabetic retinopathy detection.



7. Implement a GUI application using PyQt5 framework, providing users with an intuitive interface for uploading retinal images, initiating predictions, and viewing results.
8. Optionally, develop a web application using frameworks like Streamlit with a clean and user-friendly interface for users to upload retinal images and receive predictions for diabetic retinopathy.

## **5.3 MODULE DESCRIPTION**

### **5.3.1 Stream lit Frame Work**

Stream lit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Data scientists or machine learning engineers are not web developers and they are not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can display data and collect needed parameters for modeling. Stream lit allows you to create a stunning-looking application with only a few lines of code. Stream lit allows you to write an app the same way you write a python code. The stream lit has a distinctive data flow, any time something changes in your code or anything needs to be updated on the screen, Stream lit reruns your python script entirely from top to bottom.

### **5.3.2 PyQt5 Framework**

PyQt5 is a Python framework built upon the Qt library, enabling developers to create cross-platform graphical user interfaces (GUIs) effortlessly. With a comprehensive range of tools and widgets, PyQt5 simplifies the development of interactive applications for desktop, mobile, and embedded platforms. Its seamless integration with Python allows for efficient GUI design and development, leveraging Python's simplicity and Qt's extensive features. Supporting advanced functionalities like event handling and custom widget creation, PyQt5 empowers developers to build responsive and visually appealing applications. Furthermore, its cross-platform compatibility ensures smooth execution across different operating systems, making it a preferred choice for creating versatile and user-friendly GUIs in Python.

### **5.3.3 Model**

#### **5.3.3.1 Deep Learning**

Deep learning also called as deep structured learning is part of a broader family of Machine based on Artificial neural networks learning methods with Representation learning. Learning can be Supervised, Unsupervised and Semi-supervised. Deep learning models are built using neural networks. Deep learning models can achieve state-of-the-art accuracy, learnings can be supervised, semi supervised, unsupervised sometimes exceeding human- level performance. Models are trained by using a large set of labelled data.

Deep learning has attracted a lot of attention because it is particularly good at a type of learning that has the potential to be very useful for real-world applications. Deep learning networks can be successfully applied to big data for knowledge discovery, knowledge application, and knowledge-based prediction. Deep neural networks have recently been successfully applied in many diverse domains as examples of end-to-end learning networks provide a mapping between an input—such as an image containing many details - to an output - such as DR severity terms. It is useful for image recognition. It provides better solutions to get optimized images.

#### **5.3.3.2 CNN (Convolutional Neural Network)**

In neural networks, Convolutional neural networks (ConvNets or CNNs) are one of the main categories to do image recognition, images classifications. Objects detections, recognition face etc., are some of the areas where CNNs are widely used. CNN image classifications take an input image, process it, and classify it under certain categories. Computers see an input image as an array of pixels and it depends on the image resolution. Based on the image resolution, it will see  $h \times w \times d$  ( $h$  = Height,  $w$  = Width,  $d$  = Dimension). A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product, such as pooling layers, fully connected layers, and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

#### **5.3.3.3 DenseNet201 Connections**

DenseNet201 is a convolutional neural network architecture renowned for its dense connectivity pattern, where each layer receives feature maps from all preceding layers. This connectivity scheme fosters rich information flow throughout the network, facilitating effective feature reuse and promoting gradient flow during training, leading to more efficient and accurate representations of complex visual data. across various computer vision tasks, making it a valuable asset for image classification, object detection, and semantic segmentation tasks.

#### **5.3.3.4 InceptionV3 Connections**

InceptionV3 is a convolutional neural network architecture notable for its sophisticated inception modules, which consist of multiple convolutional layers of different kernel sizes concatenated in parallel. These parallel pathways allow the network to capture both local and global features across different scales simultaneously, enhancing its ability to extract meaningful information from input images. Additionally, the inclusion of auxiliary classifiers during training aids in combating the vanishing gradient problem and facilitates more stable training. With its intricate network design and multi-scale feature extraction capabilities, InceptionV3 achieves superior performance in various computer vision tasks such as image classification and object recognition.

#### **5.3.3.5 MobileNetV2 Connections**

MobileNetV2 is a lightweight convolutional neural network architecture designed for efficient mobile and embedded applications. Its key innovation lies in the use of depthwise separable convolutions, which split the standard convolutional operation into separate depthwise and pointwise convolutions. This separation significantly reduces computational complexity while preserving representational capacity, making MobileNetV2 well-suited. Additionally, MobileNetV2 incorporates inverted residual blocks with linear bottleneck layers, allowing for efficient information flow and feature reuse across different network depths. With its compact design and efficient architecture, MobileNetV2 achieves excellent performance in tasks such as image classification and object detection on mobile devices.

## **6. SYSTEM DESIGN**

### **6.1 INTRODUCTION TO UML**

Unified Modeling Language (UML) is a general-purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite like blueprints used in other fields of engineering. UML is not a programming language; it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system. UML helps software engineers, businessmen and system architects with modeling, design, and analysis. The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. It has been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

#### **6.1.1 Why we need UML**

- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- Businessmen cannot understand code. So, UML becomes essential to communicate with nonprogrammers' requirements, functionalities, and processes of the system.
- A lot of time is saved down the line when teams can visualize processes, user interactions and static structure of the system.

#### **6.1.2 Characteristics of UML**

The Unified Modelling Language exhibits the following features:

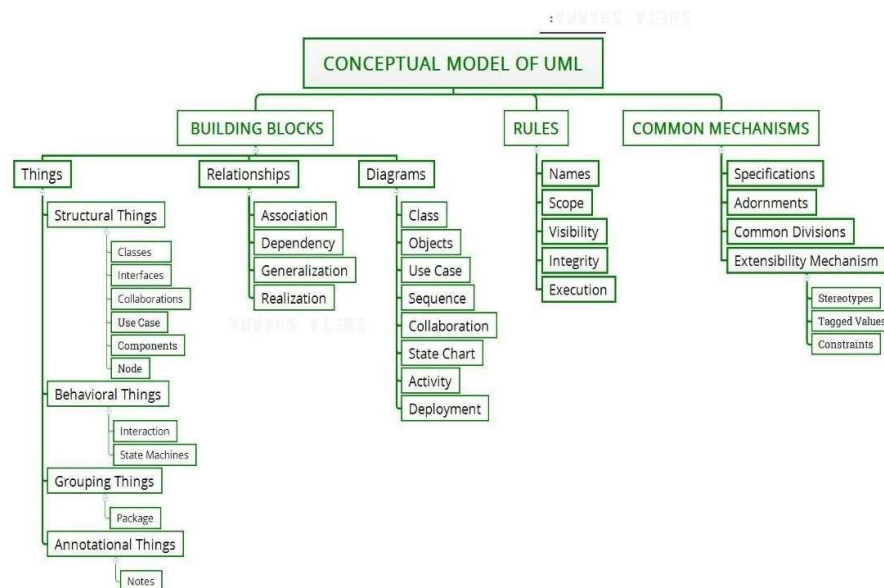
- It is a generalized modeling language.

- It is distinct from other programming languages like C++, Python.
- It is interrelated to object-oriented analysis and design.
- It is used to visualize the workflow of the system.
- It is a pictorial language, used to generate powerful modeling artifacts.

UML is linked with object-oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

- Structural Diagrams – Capture static aspects or structure of a system. Structural Diagrams include Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.
- Behavior Diagrams – Capture dynamic aspects or behavior of the system. Behavior diagrams include Use Case Diagrams, State Diagrams, Activity Diagrams, and Interaction Diagrams.

### Conceptual Model of the UML



## **6.2 BUILDING BLOCKS OF THE UML**

UML is composed of three main building blocks, i.e., things, relationships, and diagrams. Building blocks generate one complete UML model diagram by rotating around several different blocks. It plays an essential role in developing UML diagrams.

The vocabulary of the UML encompasses three kinds of building blocks:

- o Things
- o Relationships
- o Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

### **6.2.1 Things in the UML**

There are four kinds of things in the UML:

- Structural things
- Behavioral things
- Grouping things
- Annotational things

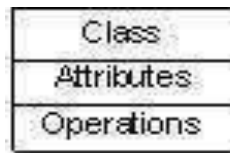
These things are the basic object-oriented building blocks of the UML. You use them to write wellformed models.

#### **Structural Things**

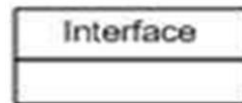
Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. Collectively, the structural things are called classifiers. Structural things define the static part of the model. They represent the physical and conceptual elements.

**Class** - A Class is a set of identical things that outlines the functionality and properties of an object. It also represents the abstract class whose functionalities are not defined. A class is a description of a set of objects that share the same attributes, operations, relationships, and

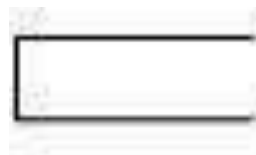
semantics. Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations. Its notation is as follows.



**Interface** - A collection of functions that specify a service of a class or component, i.e., Externally visible behavior of that class.



**Collaboration** - A larger pattern of behaviors and actions. Example: All classes and behaviors that create the modeling of a moving tank in a simulation.

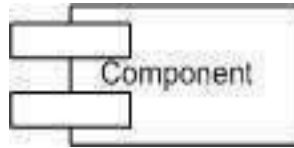


**Use Case** - A sequence of actions that a system performs that yields an observable result. Used to structure behavior in a model. Is realized by collaboration.

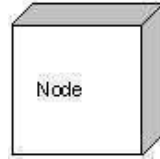
A use case can be used to identify the tasks that are performed by the actors involved in the system or in any OOP scenario. A use case is a description of how a person who actually uses that process or system will accomplish a goal. It's typically associated with software systems, but can be used in reference to any process.



**Component** - A physical and replaceable part of a system that implements a number of interfaces. Example: a set of classes, interfaces, and collaborations. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function.



**Node** - A physical element existing at run time and represents are source.



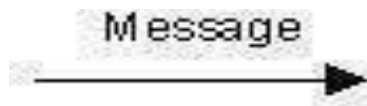
### **Behavioral Things**

Behavioral things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space. In all, there are three primary kinds of behavioral things. They are :

- Interaction
- State machine

### **Interaction**

Interaction is a behavior that comprises a set of messages exchanged among a set of objects or roles within a particular context to accomplish a specific purpose. The behavior of a society of objects or

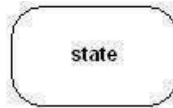


of an individual operation may be specified with an interaction. An interaction involves a number of other elements, including messages, actions, and connectors (the connection between objects). Graphically, a message is rendered as a directed line, almost always including the name of operation.

### **State machine**

State machine is a behavior that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its responses to those events. A state machine involves a number of other elements, including states, transitions (the flow from state to state), events (things that trigger a transition), and activities (the response to a transition).





### **Grouping Things**

Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available.

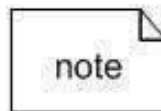
**Package** – Package is the grouping thing available for gathering structural and behavioral things.



### **Annotational Things**

Annotational things are the explanatory parts of UML models. These are the comments you may apply to describe, illuminate, and remark about any element in a model. There is one primary kind of annotational thing, called a note.

**Note** - A note is simply a symbol for rendering constraints and comments attached to an element or a collection of elements.



## **6.2.2 Relationships in the UML**

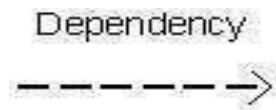
Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships in the UML:

- Dependency
- Association
- Generalization
- Realization

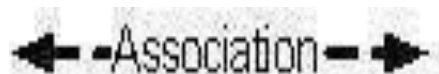
### **Dependency**

It is an element (the independent one) that may affect the semantics of the other element (the dependent one). Graphically, a dependency is rendered as a dashed line, possibly directed, and occasionally including a label. You can use dependency relationships in class diagrams, component diagrams, deployment diagrams, and use-case diagrams to indicate that a change to the supplier might require a change to the client. You can also use a dependency relationship to represent precedence, where one model element must precede another.



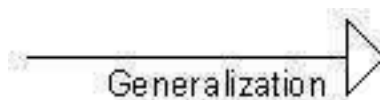
### **Association**

Association is basically a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship. In UML models, an association is a relationship between two classifiers, such as classes or use cases, that describes the reasons for the relationship and the rules that govern the relationship. An association represents a structural relationship that connects two classifiers. Like attributes, associations record the properties of classifiers.



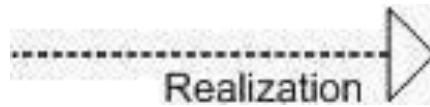
### **Generalization**

It is a specialization/generalization relationship in which the specialized element (the child) builds on the specification of the generalized element (the parent). The child shares the structure and the behavior of the parent. Graphically, a generalization relationship is rendered as a solid line with a hollow arrowhead pointing to the parent.



## **Realization**

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.



So object-oriented systems are generally modeled using the pictorial language. UML diagrams are drawn from different perspectives like design, implementation, deployment, etc. UML can be defined as a modeling language to capture the architectural, behavioral, and structural aspects of a system.

## **6.3 UML DIAGRAMS :**

UML is a modern approach to modeling and documenting software. It is based on diagrammatic representations of software components. It is the final output, and the diagram represents the system. The acronym UML refers to the Unified Modeling Language. It is a standard that is mostly used to develop object-oriented documentation models that are useful for any software system that is utilized in the real world. It gives us a tool to create detailed models that explain how any software or hardware system functions.

UML includes the following

- Class diagram
- Object diagram
- Component diagram
- Use case diagram
- Sequence diagram

➤ State diagram

➤ Activity diagram

### 6.3.1 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system.

Following are the purpose of class diagrams given below:

- It analyses and designs a static view of an application.
- It describes the major responsibilities of a system.
- It is a base for component and deployment diagrams.
- It incorporates forward and reverse engineering.

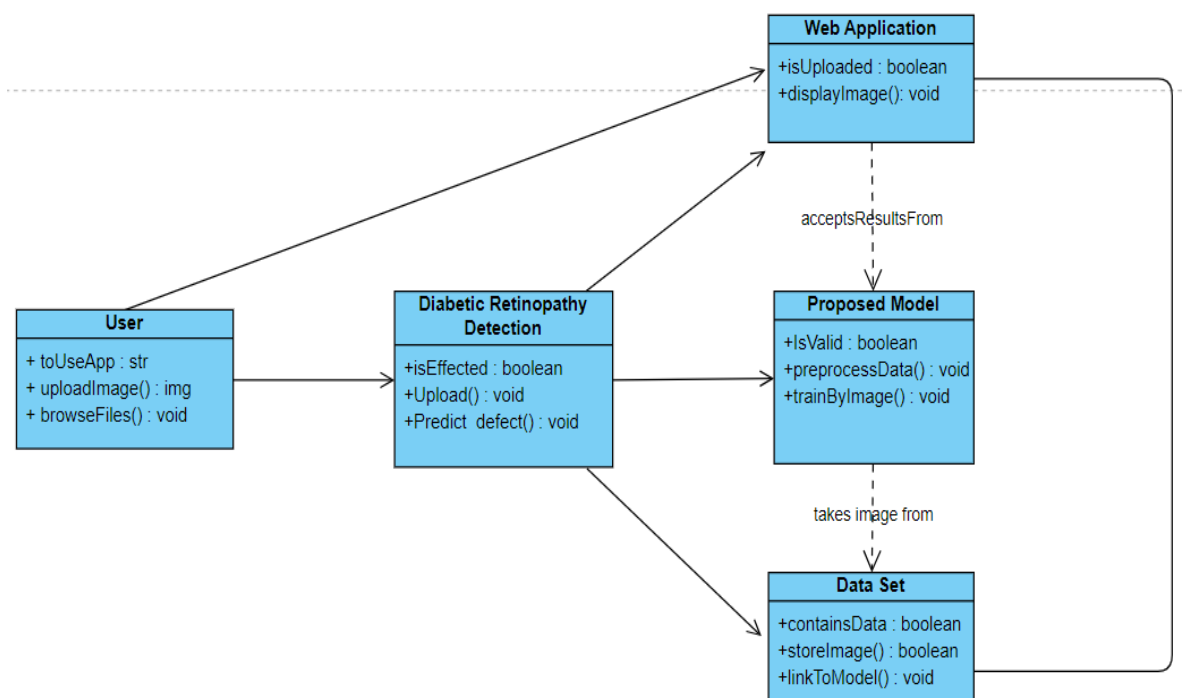


Fig 6.2 CLASS DIAGRAM

### 6.3.2 FLOWCHART

A flowchart is a visual representation of the sequence of steps and decisions needed to perform a process. Each step in the sequence is noted within a diagram shape. Steps are linked by connecting lines and directional arrows. When designing and planning a process, flowcharts can help you identify its essential steps and simultaneously offer the bigger picture of the process. Each step is independent of implementation as the flowchart only describes what should happen at that step, what input is needed and what the output of the step is but it says nothing about how

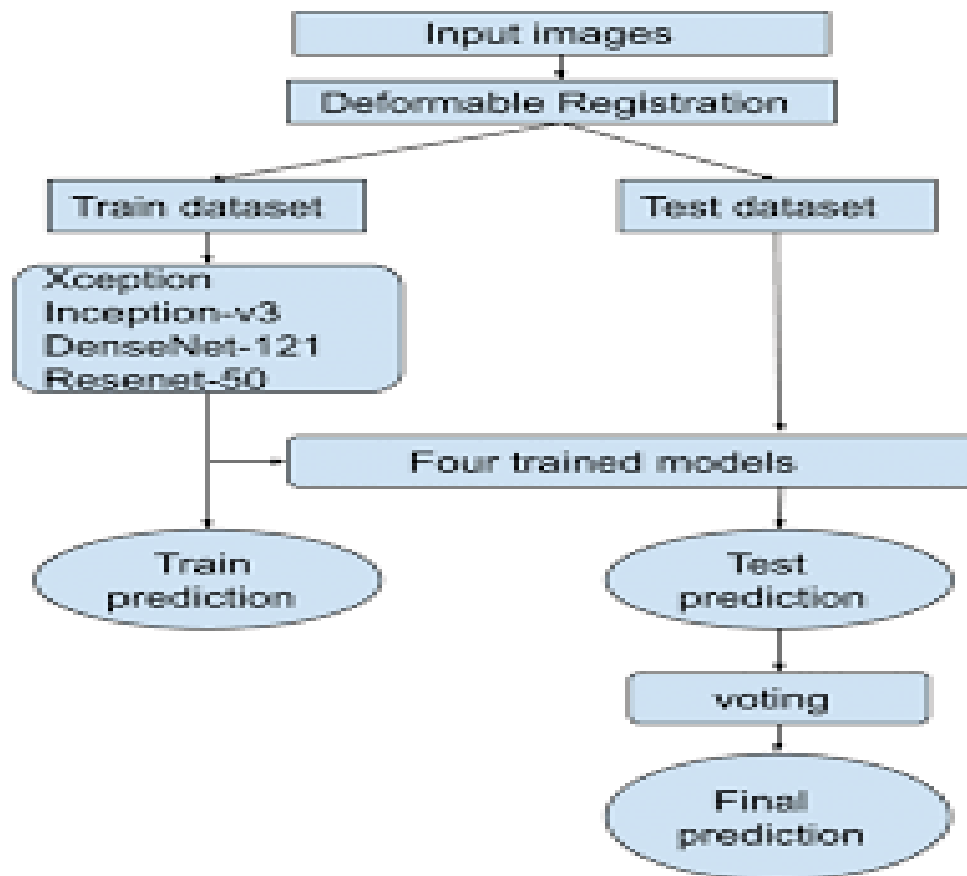


Fig 6.3 FLOWCHART

### 6.3.3 USE CASE DIAGRAM

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the highlevel functionality of a system and also tells how the user handles a system.

In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

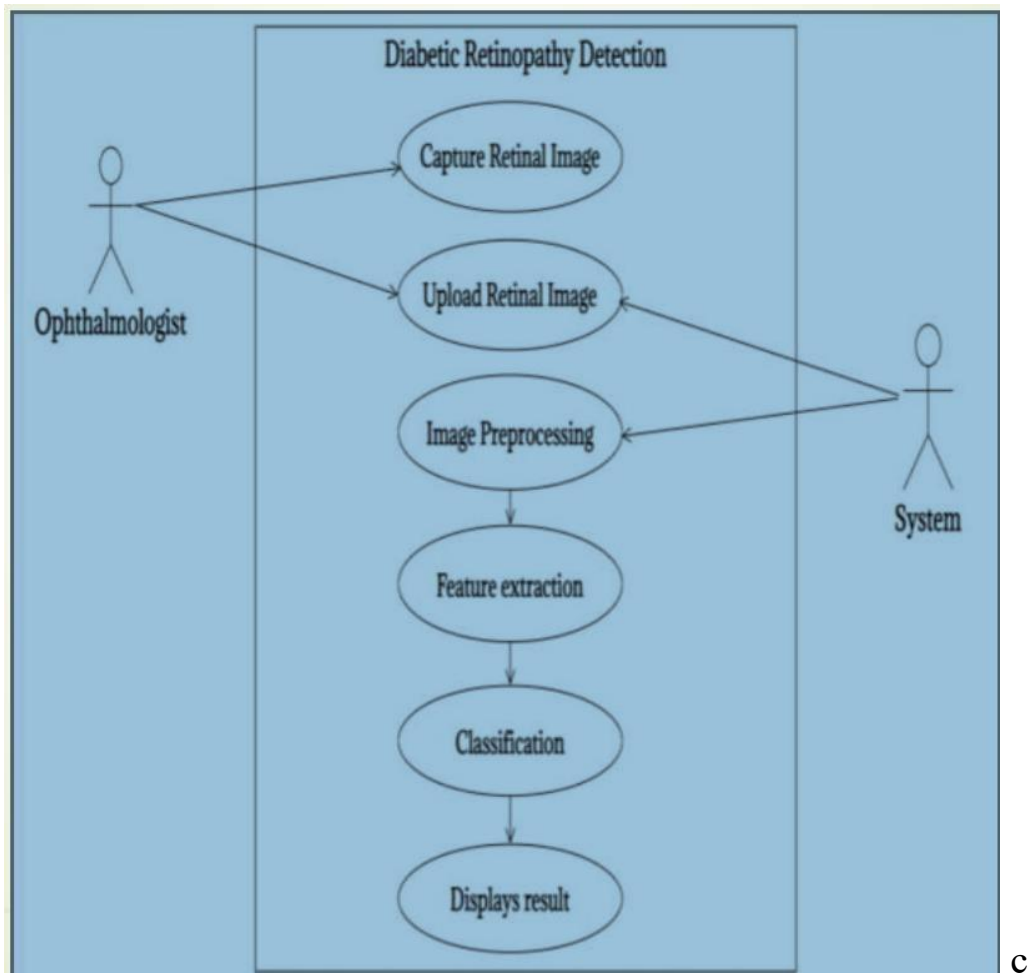


Fig 6.4 USE CASE DIAGRAM

#### **6.3.4 ACTIVITY DIAGRAM**

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be

sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).

### Purpose of Activity Diagrams

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another. Activity is a particular operation of the system.

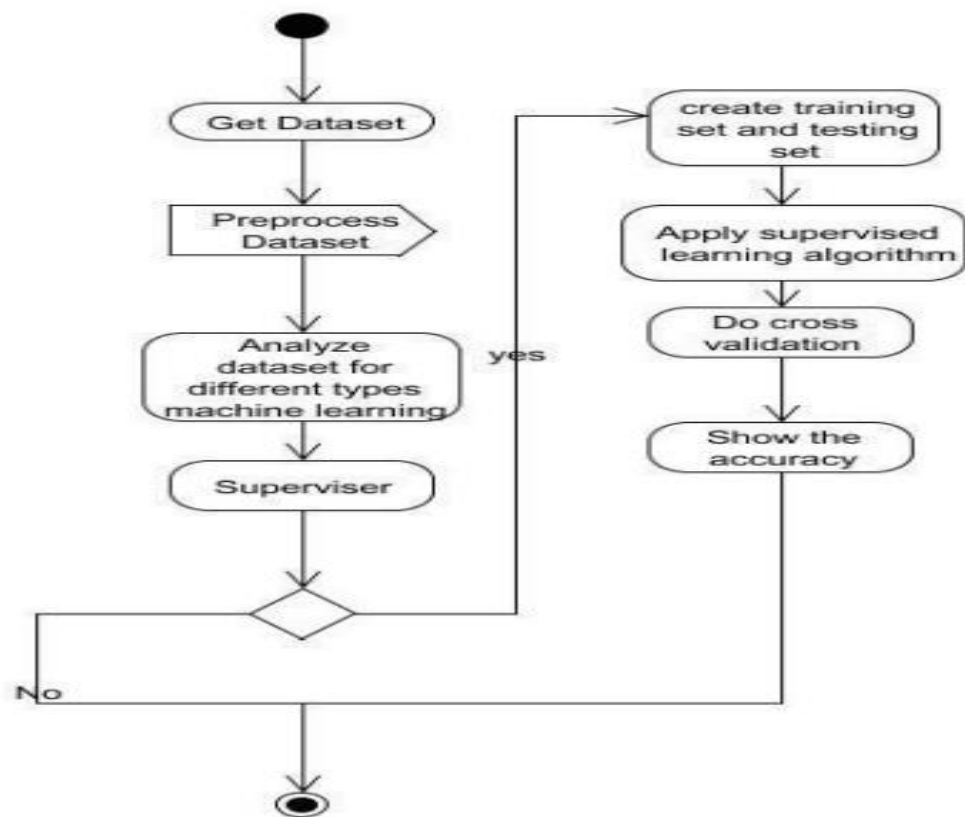


Fig 6.5 ACTIVITY DIAGRAM

## **7. DEVELOPMENT**

A DATASET is a collection of data. This set is normally presented in a tabular pattern. It contains data that is obtained from various sources. Every column describes a particular variable. Each row corresponds to a given member of the data set, as per the given question. This is a part of DATA MANAGEMENT in the data analytics domain. Some types of Datasets are: Numerical, Bivariate, Multivariate, Correlational, Categorical.

### **7.1 DATASET USED – DIABETIC RETINOPATHY DETECTION**

For the fracture detection purpose, we have acquired a Bone X-ray dataset from the link i.e. <https://www.kaggle.com/c/diabetic-retinopathy-detection/data>. The dataset encompasses a diverse range of fundus images showcasing various images of the human eye.

The dataset contains a comprehensive collection of microscopic fundus images, offering insights into retina structure, density, and potential abnormalities across different body parts. These serve as valuable resources for tasks such as retinopathy detection and anatomical analysis. The dataset comprises images captured using various imaging modalities, providing a broad spectrum of visual information for research, analysis, and algorithm development in the field of medical imaging and diagnostics.

The dataset is divided into three different sets of training, validation and testing data for accurate prediction of bone related abnormalities like retinopathy . All the sets of 5508 images in the dataset are divided in the ratio 70% training, 20% validation and 10% testing parts of the dataset.

Train Dataset (70%) : There are 4480 images from the total of the 6400 images in the dataset that fall into the training dataset. These images are mainly used during the training of the model for accurate prediction of the severity.

Valid Dataset (20%) : Another 1280 images constitute the validation set to underline the effectiveness of the model under each and every category and in turn ensure for the maximum accuracy score for accurate results.



Testing Dataset (10%) : The final set of 448 images constitute the testing part of the dataset for the optimal testing of the accuracy of the proposed model. These testing images are further enhanced with high level of data processing and image recognition in order to provide optimal results to the user.

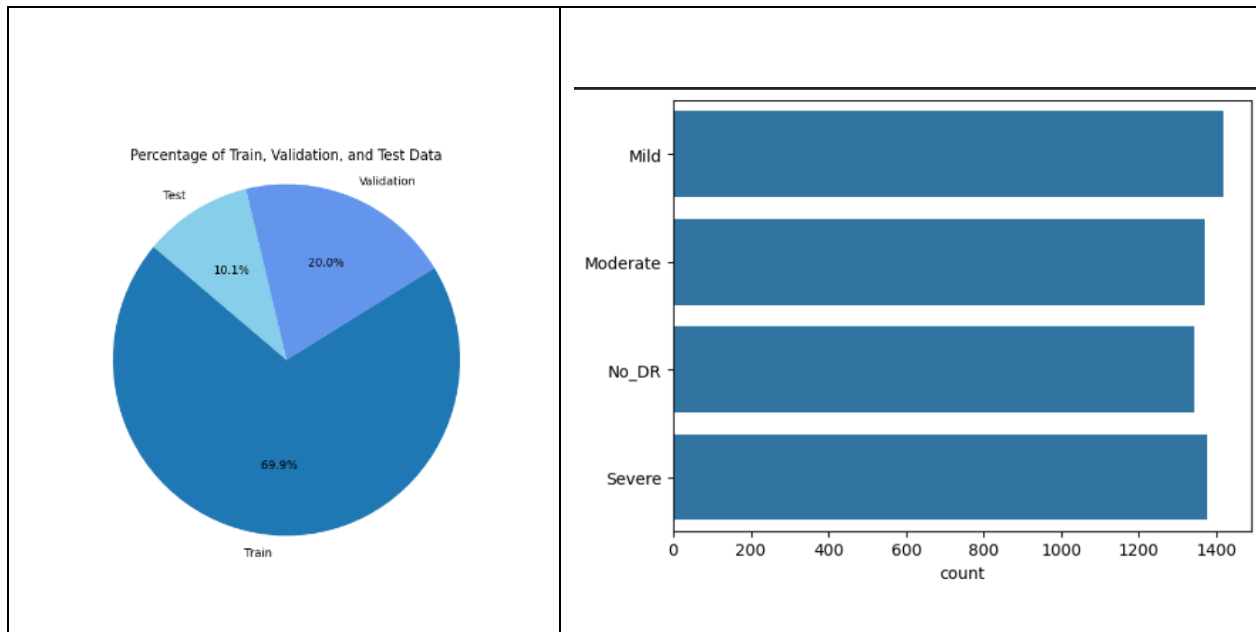


Fig 7.1 Graphical depiction of the division of data in the dataset

### 7.1.1 DATA SET IMAGES

The below set of all the images constitute the X-Ray samples of bone fracture of various patients as per the Bone X-ray dataset. These samples are further highly verified for micro details and necessary magnification and fitting of the image samples is carried out.

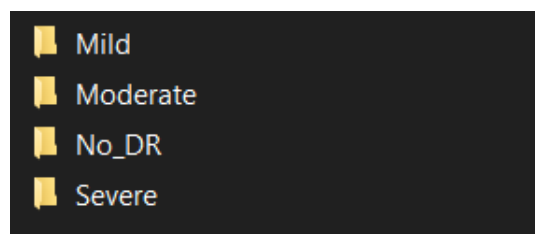


Fig 7.2 Total Classes in the Selected Dataset

Fig 7.3 Images in the Training Division of the Dataset

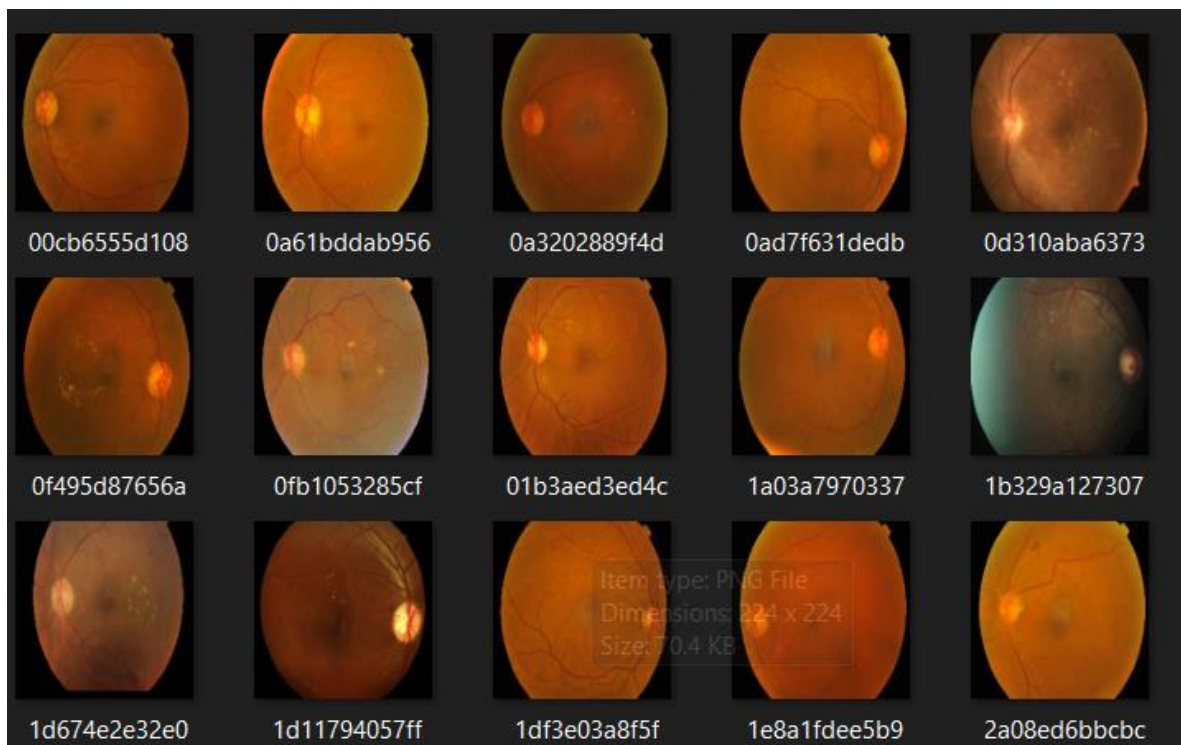


Fig 7.4 Images in the Validation Division of the Dataset

The set of images above refer to the validation division of the dataset which ensure the further better filtering of the details of the specific position of the nerves alignment in the fundus image in the dataset chosen for the model.

The below set of images constitute the testing division of the dataset which are mainly used for the testing purpose with the uploaded patient's images in order to ensure optimal result by the model with maximum accuracy.

### **7.3 SAMPLE CODE**

#### 1. Importing Libraries

```
import numpy as np
import os
import shutil
np.random.seed(42)
from sklearn.preprocessing import LabelEncoder
import cv2
import tensorflow as tf
import keras
import shutil
import random
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.utils import class_weight
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

```
# Generate batches of augmented data
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=360,
    horizontal_flip=True,
    vertical_flip=True)
```

```
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
```

## 2. Model Train - DENSENET201

```
# Load pre-trained DenseNet50 model without the top classification layer
```

```
IMG_SHAPE = (224, 224, 3)
```

```
base_model = tf.keras.applications.DenseNet201(weights='imagenet',
    include_top=False,
    input_shape=IMG_SHAPE)
```

```
x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
```

```
x = tf.keras.layers.Dropout(0.25)(x)
```

```
x = tf.keras.layers.Dense(2048, activation='relu')(x)
```

```
x = tf.keras.layers.Dropout(0.25)(x)
```

```
final_output = tf.keras.layers.Dense(N_CLASSES, activation='softmax', name='final_output')(x)
```

```
model = tf.keras.models.Model(inputs=base_model.inputs, outputs=final_output)
```

```
# Final model
```

```
model = Model(inputs=base_model.input, outputs=predictions)
```

```
# Train top layers of the pre-trained model
```

```
for layer in model.layers:
```

```
    layer.trainable = False
```

```
for i in range(-5, 0):
```

```
    model.layers[i].trainable = True
```

```
metric_list = ["accuracy"]
```

```
optimizer = tf.keras.optimizers.Adam(lr=WARMUP_LEARNING_RATE)
```

```
model.compile(optimizer=optimizer, loss="categorical_crossentropy", metrics=metric_list)
```

```
model.summary()
```

```
# Train the model
```

```
history_finetuning = model.fit_generator(generator=train_generator,  
                                       steps_per_epoch=STEP_SIZE_TRAIN,  
                                       validation_data=val_generator,  
                                       validation_steps=STEP_SIZE_VALID,  
                                       epochs=EPOCHS,  
                                       callbacks=callback_list,  
                                       verbose=1).history
```

### 3. Evaluate Model

```
loss_Val, acc_Val = model.evaluate(X_val, y_val_ohe, batch_size=1, verbose=1)  
print("Validation: accuracy = %f ; loss_v = %f" % (acc_Val, loss_Val))
```

```
# Classification report
```

```
Validation_pred = DensNet201_model.predict(X_val)
```

```
Prdict_label = np.argmax(Validation_pred, -1)
```

```
Actual_label = y_val
```

```
print('Accuracy on Validation Data: %2.2f%%' % (100*accuracy_score(Actual_label,  
Prdict_label)))
```

```
print(classification_report(Actual_label, Prdict_label))
```

### 2. Model Train - INCEPTIONV3

```
# Load pre-trained inceptionV3 model without the top classification layer
```

```
IMG_SHAPE = (224, 224, 3)
```

```
IMG_SHAPE = (224, 224, 3)
```

```
base_model = tf.keras.applications.InceptionV3(weights='imagenet',  
                                              include_top=False,  
                                              input_shape=IMG_SHAPE)
```

```
x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
```

```
x = tf.keras.layers.Dropout(0.25)(x)
```

```
x = tf.keras.layers.Dense(2048, activation='relu')(x)
```

```
x = tf.keras.layers.Dropout(0.25)(x)
```

```

final_output =tf.keras.layers.Dense(N_CLASSES, activation='softmax', name='final_output')(x)
model =tf.keras.models.Model(inputs=base_model.inputs,outputs=final_output)
for layer in model.layers:
    layer.trainable = False

for i in range(-5, 0):
    model.layers[i].trainable = True

metric_list = ["accuracy"]
optimizer =tf.keras.optimizers.Adam(lr=WARMUP_LEARNING_RATE)
model.compile(optimizer=optimizer, loss="categorical_crossentropy", metrics=metric_list)
model.summary()
# Train the model
history_finertuning = model.fit_generator(generator=train_generator,
                                         steps_per_epoch=STEP_SIZE_TRAIN,
                                         validation_data=val_generator,
                                         validation_steps=STEP_SIZE_VALID,
                                         epochs=EPOCHS,
                                         callbacks=callback_list,
                                         verbose=1).history

```

### 3. Evaluate Model

```

loss_Val, acc_Val = model.evaluate(X_val, y_val_ohe, batch_size=1, verbose=1)
print("Validation: accuracy = %f ; loss_v = %f" % (acc_Val, loss_Val))

```

#### # Classification report

```

Test_predict = DensNet201_model.predict(X_test)

```

```

Predict_label = np.argmax(Test_predict, -1)

```

```

Actual_label = y_test

```

```

print('Accuracy on Test Data: %2.2f%%' % (100*accuracy_score(Actual_label, Predict_label)))

```

```

print(classification_report(Actual_label, Predict_label))

```

## 2. Model Train - MobileNetV2

```
# Load pre-trained MobileNetV2 model without the top classification layer
IMG_SHAPE = (224, 224, 3)
base_model = tf.keras.applications.MobileNetV2(weights='imagenet',
                                                include_top=False,
                                                input_shape=IMG_SHAPE)
x = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
x = tf.keras.layers.Dropout(0.25)(x)
x = tf.keras.layers.Dense(2048, activation='relu')(x)
x = tf.keras.layers.Dropout(0.25)(x)
final_output = tf.keras.layers.Dense(N_CLASSES, activation='softmax', name='final_output')(x)
model = tf.keras.models.Model(inputs=base_model.inputs, outputs=final_output)
for layer in model.layers:
    layer.trainable = False

for i in range(-5, 0):
    model.layers[i].trainable = True

metric_list = ["accuracy"]
optimizer = tf.keras.optimizers.Adam(lr=WARMUP_LEARNING_RATE)
model.compile(optimizer=optimizer, loss="categorical_crossentropy", metrics=metric_list)
model.summary()
# Train the model
history_finetuning = model.fit_generator(generator=train_generator,
                                        steps_per_epoch=STEP_SIZE_TRAIN,
                                        validation_data=val_generator,
                                        validation_steps=STEP_SIZE_VALID,
                                        epochs=EPOCHS,
                                        callbacks=callback_list,
                                        verbose=1).history
```

## 3. Evaluate Model

```
loss_Val, acc_Val = model.evaluate(X_val, y_val_ohe, batch_size=1, verbose=1)
```

```

print("Validation: accuracy = %f ; loss_v = %f" % (acc_Val, loss_Val))

# Classification report
Test_predict = MobileNetV2_model.predict(X_test)

Prdict_label = np.argmax(Test_predict, -1)
Actual_label = y_test

print('Accuracy on Test Data: %2.2f%%' % (100*accuracy_score(Actual_label, Prdict_label)))
print(classification_report(Actual_label, Prdict_label))

```

#### PyQt5 GUI Code:

```

import Arsi
from PyQt5 import QtCore, QtGui, QtWidgets
import tensorflow as tf
import cv2
import numpy as np
from tqdm import tqdm, trange
from time import sleep
from Preprocess_Gaussian_Blur import load_ben_color
from Size_Normalize import del_black_or_white
from ContratsNormalize_CLAHE import CLAHE
import cv2
import numpy as np
import os
import tempfile

```

```

input_shape = (224,224,3)
model_input = tf.keras.Input(shape=input_shape)

```

```

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")

```



```

MainWindow.resize(1400, 750)
self.centralwidget = QtWidgets.QWidget(MainWindow)
self.centralwidget.setObjectName("centralwidget")
self.label = QtWidgets.QLabel(self.centralwidget)
# self.label.setGeometry(QtCore.QRect(250, 10, 771, 21))
self.label.setGeometry(QtCore.QRect(350, 20, 971, 31))

font = QtGui.QFont()
font.setFamily("Rockwell Extra Bold")
font.setPointSize(26)
#font.setBold(True)
font.setWeight(75)
self.label.setFont(font)
self.label.setObjectName("label")

font = QtGui.QFont()
font.setPointSize(22)
font.setBold(True)
font.setWeight(75)
self.label_4.setFont(font)
self.label_4.setObjectName("label_4")
self.label_5 = QtWidgets.QLabel(self.centralwidget)
self.label_5.setGeometry(QtCore.QRect(320, 30, 871, 751))
self.label_5.setStyleSheet("image: url(:/ars/Arshi.png);")
self.label_5.setText("")
self.label_5.setObjectName("label_5")
self.label_6 = QtWidgets.QLabel(self.centralwidget)
self.label_6.setGeometry(QtCore.QRect(480, 70, 381, 41))
font = QtGui.QFont()
font.setFamily("Swis721 Hv BT")
font.setPointSize(14)
self.label_6.setFont(font)

```

```

self.label_6.setObjectName("label_6")
self.txt_V3_0 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_V3_0.setGeometry(QtCore.QRect(830, 360, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_V3_0.setFont(font)
self.txt_V3_0.setAlignment(QtCore.Qt.AlignCenter)
self.txt_V3_0.setObjectName("txt_V3_0")
self.txt_V3_1 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_V3_1.setGeometry(QtCore.QRect(830, 390, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_V3_1.setFont(font)
self.txt_V3_1.setAlignment(QtCore.Qt.AlignCenter)
self.txt_V3_1.setObjectName("txt_V3_1")
self.txt_V3_3 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_V3_3.setGeometry(QtCore.QRect(830, 420, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_V3_3.setFont(font)
self.txt_V3_3.setAlignment(QtCore.Qt.AlignCenter)
self.txt_V3_3.setObjectName("txt_V3_3")
self.txt_V3_4 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_V3_4.setGeometry(QtCore.QRect(830, 450, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)

```

```

font.setWeight(75)
self.txt_V3_4.setFont(font)
self.txt_V3_4.setAlignment(QtCore.Qt.AlignCenter)
self.txt_V3_4.setObjectName("txt_V3_4")
self.txt_V2_0 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_V2_0.setGeometry(QtCore.QRect(830, 540, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_V2_0.setFont(font)
self.txt_V2_0.setAlignment(QtCore.Qt.AlignCenter)
self.txt_V2_0.setObjectName("txt_V2_0")
self.txt_V2_1 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_V2_1.setGeometry(QtCore.QRect(830, 570, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_V2_1.setFont(font)
self.txt_V2_1.setAlignment(QtCore.Qt.AlignCenter)
self.txt_V2_1.setObjectName("txt_V2_1")
self.txt_V2_2 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_V2_2.setGeometry(QtCore.QRect(830, 600, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_V2_2.setFont(font)
self.txt_V2_2.setAlignment(QtCore.Qt.AlignCenter)
self.txt_V2_2.setObjectName("txt_V2_2")
self.txt_V2_3 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_V2_3.setGeometry(QtCore.QRect(830, 630, 131, 20))

```

```

____font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_V2_3.setFont(font)
self.txt_V2_3.setAlignment(QtCore.Qt.AlignCenter)
self.txt_V2_3.setObjectName("txt_V2_3")
self.txt_201_1 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_201_1.setGeometry(QtCore.QRect(830, 210, 141, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_201_1.setFont(font)
self.txt_201_1.setText("")
self.txt_201_1.setAlignment(QtCore.Qt.AlignCenter)
self.txt_201_1.setObjectName("txt_201_1")
self.txt_201_2 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_201_2.setGeometry(QtCore.QRect(830, 240, 141, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_201_2.setFont(font)
self.txt_201_2.setAlignment(QtCore.Qt.AlignCenter)
self.txt_201_2.setObjectName("txt_201_2")
self.txt_201_3 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_201_3.setGeometry(QtCore.QRect(830, 270, 141, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_201_3.setFont(font)

```

```

self.txt_201_3.setAlignment(QtCore.Qt.AlignCenter)
self.txt_201_3.setObjectName("txt_201_3")
self.txt_ESL_0 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_ESL_0.setGeometry(QtCore.QRect(1200, 350, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_ESL_0.setFont(font)
self.txt_ESL_0.setAlignment(QtCore.Qt.AlignCenter)
self.txt_ESL_0.setObjectName("txt_ESL_0")
self.txt_ESL_1 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_ESL_1.setGeometry(QtCore.QRect(1200, 380, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_ESL_1.setFont(font)
self.txt_ESL_1.setAlignment(QtCore.Qt.AlignCenter)
self.txt_ESL_1.setObjectName("txt_ESL_1")
self.txt_ESL_2 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_ESL_2.setGeometry(QtCore.QRect(1200, 410, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_ESL_2.setFont(font)
self.txt_ESL_2.setAlignment(QtCore.Qt.AlignCenter)
self.txt_ESL_2.setObjectName("txt_ESL_2")
self.txt_ESL_4 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_ESL_4.setGeometry(QtCore.QRect(1180, 510, 161, 21))
font = QtGui.QFont()
font.setPointSize(10)

```

```

self.txt_ESL_4.setFont(font)
self.txt_ESL_4.setAlignment(QtCore.Qt.AlignCenter)
self.txt_ESL_4.setObjectName("txt_ESL_4")
self.label_7 = QtWidgets.QLabel(self.centralwidget)
self.label_7.setGeometry(QtCore.QRect(1240, 490, 71, 21))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.label_7.setFont(font)
self.label_7.setObjectName("label_7")
self.predict_img_lbl = QtWidgets.QLabel(self.centralwidget)
self.predict_img_lbl.setGeometry(QtCore.QRect(240, 290, 201, 201))
self.predict_img_lbl.setAutoFillBackground(True)
self.predict_img_lbl.setFrameShape(QtWidgets.QFrame.WinPanel)
self.predict_img_lbl.setText("")
self.predict_img_lbl.setObjectName("predict_img_lbl")
self.Load_img_btn = QtWidgets.QPushButton(self.centralwidget)
self.Load_img_btn.setGeometry(QtCore.QRect(0, 610, 171, 51))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.Load_img_btn.setFont(font)
self.Load_img_btn.setObjectName("Load_img_btn")
self.Predict_btn = QtWidgets.QPushButton(self.centralwidget)
self.Predict_btn.setGeometry(QtCore.QRect(290, 610, 171, 51))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.Predict_btn.setFont(font)
self.Predict_btn.setObjectName("Predict_btn")

```

```

self.txt_ESL_3 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_ESL_3.setGeometry(QtCore.QRect(1200, 440, 131, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_ESL_3.setFont(font)
self.txt_ESL_3.setAlignment(QtCore.Qt.AlignCenter)
self.txt_ESL_3.setObjectName("txt_ESL_3")
self.txt_201_0 = QtWidgets.QLineEdit(self.centralwidget)
self.txt_201_0.setGeometry(QtCore.QRect(830, 180, 141, 20))
font = QtGui.QFont()
font.setPointSize(9)
font.setBold(True)
font.setWeight(75)
self.txt_201_0.setFont(font)
self.txt_201_0.setText("")
self.txt_201_0.setAlignment(QtCore.Qt.AlignCenter)
self.txt_201_0.setObjectName("txt_201_0")
self.Clr_btn = QtWidgets.QPushButton(self.centralwidget)
self.Clr_btn.setGeometry(QtCore.QRect(200, 630, 75, 23))
self.Clr_btn.setObjectName("Clr_btn")
self.Load_image_lbl_2 = QtWidgets.QLabel(self.centralwidget)
self.Load_image_lbl_2.setGeometry(QtCore.QRect(10, 290, 211, 201))
self.Load_image_lbl_2.setAutoFillBackground(True)
self.Load_image_lbl_2.setFrameShape(QtWidgets.QFrame.WinPanel)
self.Load_image_lbl_2.setText("")
self.Load_image_lbl_2.setObjectName("Load_image_lbl_2")
self.Preprocess_btn = QtWidgets.QPushButton(self.centralwidget)
self.Preprocess_btn.setGeometry(QtCore.QRect(140, 540, 151, 41))
font = QtGui.QFont()
font.setPointSize(12)
font.setBold(True)

```

```

font.setWeight(75)
self.Preprocess_btn.setFont(font)
self.Preprocess_btn.setObjectName("Preprocess_btn")
self.label_8 = QtWidgets.QLabel(self.centralwidget)
self.label_8.setGeometry(QtCore.QRect(30, 250, 161, 31))
font = QtGui.QFont()
font.setFamily("Arial Black")
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.label_8.setFont(font)
self.label_8.setObjectName("label_8")
self.label_9 = QtWidgets.QLabel(self.centralwidget)
self.label_9.setGeometry(QtCore.QRect(280, 250, 121, 31))
font = QtGui.QFont()
font.setFamily("Arial Black")
font.setPointSize(12)
font.setBold(True)
font.setWeight(75)
self.label_9.setFont(font)
self.label_9.setObjectName("label_9")
self.label_5.raise_()
self.label.raise_()
# self.label_2.raise_()
# self.label_3.raise_()
self.label_4.raise_()
self.label_6.raise_()
self.txt_V3_0.raise_()
self.txt_V3_1.raise_()
self.txt_V3_3.raise_()
self.txt_V3_4.raise_()
self.txt_V2_0.raise_()
self.txt_V2_1.raise_()

```



```

self.txt_V2_2.raise_()
self.txt_V2_3.raise_()
self.txt_201_1.raise_()
self.txt_201_2.raise_()
self.txt_201_3.raise_()
self.txt_ESL_0.raise_()
self.txt_ESL_1.raise_()
self.txt_ESL_2.raise_()
self.txt_ESL_4.raise_()
self.label_7.raise_()
self.predict_img_lbl.raise_()
self.Load_img_btn.raise_()
self.Predict_btn.raise_()
self.txt_ESL_3.raise_()
self.txt_201_0.raise_()
self.Clr_btn.raise_()
self.Load_image_lbl_2.raise_()
self.Preprocess_btn.raise_()
self.label_8.raise_()
self.label_9.raise_()
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 1345, 21))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)
self.Load_img_btn.clicked.connect(self.setImage)
self.Preprocess_btn.clicked.connect(self.PraProcess)

```

```

self.Predict_btn.clicked.connect(self.Deteksi)
self.Clr_btn.clicked.connect(self.Clear)

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "DR-DETECTION"))
    self.label.setText(_translate("MainWindow", "DIABETIC RETINOPATHY
CLASSIFIER"))
    self.label_4.setText(_translate("MainWindow", "    Ensemble Architecture"))
    # self.label_6.setText(_translate("MainWindow", "(FAKULTAS TEKNIK-UNIB)")
    self.label_7.setText(_translate("MainWindow", "Output"))
    self.Load_img_btn.setText(_translate("MainWindow", "LOAD"))
    self.Predict_btn.setText(_translate("MainWindow", "PREDICT"))
    self.Clr_btn.setText(_translate("MainWindow", "Clear"))
    self.Preprocess_btn.setText(_translate("MainWindow", "Preprocess"))
    self.label_8.setText(_translate("MainWindow", "    ORIGINAL"))
    self.label_9.setText(_translate("MainWindow", "    INPUT"))

def Clear (self):
    self.txt_ESL_0.clear()
    self.txt_ESL_1.clear()
    self.txt_ESL_2.clear()
    self.txt_ESL_3.clear()
    self.txt_ESL_4.clear()
    self.txt_201_0.clear()
    self.txt_201_1.clear()
    self.txt_201_2.clear()
    self.txt_201_3.clear()
    self.txt_V3_0.clear()
    self.txt_V3_1.clear()
    self.txt_V3_3.clear()
    self.txt_V3_4.clear()
    self.txt_V2_0.clear()

```

```

self.txt_V2_1.clear()
self.txt_V2_2.clear()
self.txt_V2_3.clear()
self.predict_img_lbl.clear()
self.Load_image_lbl_2.clear()

def setImage(self):
    global fileName
    fileName,_=QtWidgets.QFileDialog.getOpenFileName(None, "Select Image", "", "Image
Files(*.png *.jpg *.jpeg *.bmp *.tif)")
    if fileName:
        pixmap=QtGui.QPixmap(fileName)
        pixmap=pixmap.scaled(self.Load_image_lbl_2.width(),self.Load_image_lbl_2.height(),Q
tCore.Qt.KeepAspectRatio)
        self.Load_image_lbl_2.setPixmap(pixmap)
        self.Load_image_lbl_2.setAlignment(QtCore.Qt.AlignCenter)
        print(fileName)

# def PraProcess(self):
#     global PreprocessFile
#     img= cv2.imread(fileName)
#     crop_size=1000
#     image1=del_black_or_white(img)
#     min_width_height=min(image1.shape[0],image1.shape[1])
#     image_size_before_hough=crop_size*2
#     if min_width_height<100:
#         crop_ratio=image_size_before_hough/min_width_height
#         image1=cv2.resize(image1,None, fx=crop_ratio, fy=crop_ratio)

#     dim=(224,224)
#     fundus1 = cv2.resize(image1, dim, interpolation = cv2.INTER_AREA)
#     imag=cv2.imwrite('Lokasi/Size_normalize.jpg',fundus1)
#     gambar='Lokasi/Size_normalize.jpg'
#     Clahe=CLAHE(gambar)

```

```

#   imag=cv2.imwrite('Lokasi/CLAHE.jpg',Clahe)
#   img='Lokasi/CLAHE.jpg'
#   img2= load_ben_color(img,sigmaX=10)
#   cv2.imwrite('Lokasi/Gaussian_BLUR.jpg',img2)

#   PreprocessFile='Lokasi/Gaussian_BLUR.jpg'

#   pixmap=QtGui.QPixmap(PreprocessFile)
#   pixmap=pixmap.scaled(self.predict_img_lbl.width(),self.predict_img_lbl.height(),QtCore.
Qt.KeepAspectRatio)
#   self.predict_img_lbl.setPixmap(pixmap)
#   self.predict_img_lbl.setAlignment(QtCore.Qt.AlignCenter)

def PraProcess(self):
    global PreprocessFile
    img = cv2.imread(fileName)
    crop_size = 1000
    image1 = del_black_or_white(img)
    min_width_height = min(image1.shape[0], image1.shape[1])
    image_size_before_hough = crop_size * 2
    if min_width_height < 100:
        crop_ratio = image_size_before_hough / min_width_height
        image1 = cv2.resize(image1, None, fx=crop_ratio, fy=crop_ratio)

    dim = (224, 224)
    fundus1 = cv2.resize(image1, dim, interpolation=cv2.INTER_AREA)

    # Temporary file paths
    temp_file1 = tempfile.mktemp('.jpg')
    temp_file2 = tempfile.mktemp('.jpg')

    imag = cv2.imwrite(temp_file1, fundus1)
    Clahe = CLAHE(temp_file1)

```

```

imag = cv2.imwrite(temp_file2, Clahe)
img2 = load_ben_color(temp_file2, sigmaX=10)

# Temporary output file path
temp_output_file = tempfile.mktemp('.jpg')
cv2.imwrite(temp_output_file, img2)

PreprocessFile = temp_output_file

pixmap = QtGui.QPixmap(PreprocessFile)
pixmap = pixmap.scaled(self.predict_img_lbl.width(), self.predict_img_lbl.height(),
QtCore.Qt.KeepAspectRatio)
self.predict_img_lbl.setPixmap(pixmap)
self.predict_img_lbl.setAlignment(QtCore.Qt.AlignCenter)

def Deteksi(self):
    IMG_SIZE=(224,224)
    img = tf.keras.preprocessing.image.load_img(PreprocessFile, target_size=IMG_SIZE)
    x = tf.keras.preprocessing.image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    Base_model1 =tf.keras.applications.DenseNet201(input_shape=input_shape,
input_tensor=model_input, include_top=False, weights=None)
    for layer in Base_model1.layers:
        layer.trainable = True
    Base_model1_last_layer = Base_model1.get_layer('relu')
    Base_model1_last_output = Base_model1_last_layer.output
    x1 =tf.keras.layers.GlobalAveragePooling2D()(Base_model1_last_output)
    x1 =tf.keras.layers.Dropout(0.25)(x1)
    x1 =tf.keras.layers.Dense(512, activation='relu')(x1)
    x1 =tf.keras.layers.Dropout(0.25)(x1)
    final_output1 =tf.keras.layers.Dense(4, activation='softmax', name='final_output')(x1)
    DensNet201_model =tf.keras.models.Model(model_input, final_output1)

```

```

metric_list = ["accuracy"]
optimizer =tf.keras.optimizers.Adam()
DensNet201_model.compile(optimizer=optimizer,
loss="categorical_crossentropy", metrics=metric_list)
if getattr(sys, 'frozen', False):
    base_path = sys._MEIPASS
else:
    base_path = os.path.abspath(".")
weights_path = os.path.join(base_path, "WEIGHT",
"Weight_DensNet201_Optimal_Ori.h5")

DensNet201_model.load_weights(weights_path)
print(DensNet201_model.summary)
print ("DensNet201 Start predict.....")

for i in trange(10):
    sleep(0.1)
    Predict1=DensNet201_model.predict(x)
    pass

print ("Normal Probabality:",Predict1[0][0])
print ("Mild Probabality:",Predict1[0][1])
print ("Moderate Probabality:",Predict1[0][2])
print ("Severe Probabality:",Predict1[0][3])

self.txt_201_0.setText(str(Predict1[0][0]))
self.txt_201_1.setText(str(Predict1[0][1]))
self.txt_201_2.setText(str(Predict1[0][2]))
self.txt_201_3.setText(str(Predict1[0][3]))

Base_model2 =tf.keras.applications.InceptionV3(input_shape=input_shape,
input_tensor=model_input, include_top=False, weights=None)
for layer in Base_model2.layers:
    layer.trainable = True

```

```

Base_model2_last_layer = Base_model2.get_layer('mixed10')
Base_model2_last_output = Base_model2_last_layer.output
x2 =tf.keras.layers.GlobalAveragePooling2D()(Base_model2_last_output)
x2 =tf.keras.layers.Dropout(0.25)(x2)
x2 =tf.keras.layers.Dense(1024, activation='relu')(x2)
x2 =tf.keras.layers.Dropout(0.25)(x2)
final_output2 =tf.keras.layers.Dense(4, activation='softmax', name='final_output2')(x2)
InceptionV3_model =tf.keras.models.Model(model_input, final_output2)
metric_list = ["accuracy"]
optimizer = tf.keras.optimizers.Adam(1.0000e-06)
InceptionV3_model.compile(optimizer=optimizer,
loss="categorical_crossentropy", metrics=metric_list)
if getattr(sys, 'frozen', False): # Running as a bundled executable
    base_path = sys._MEIPASS
else:
    base_path = os.path.abspath(".")

# Construct the full path to the weights file
weights_path1 = os.path.join(base_path, "WEIGHT",
"Weight_InceptionV3_Optimal_Ori.h5")

# Load the model weights
InceptionV3_model.load_weights(weights_path1)
print ("InceptionV3 Start predict.....")

for i in trange(10):
    sleep(0.1)
    Predict2=InceptionV3_model.predict(x)
    pass

print ("Normal Probabality:",Predict2[0][0])
print ("Mild Probabality:",Predict2[0][1])
print ("Moderate Probabality:",Predict2[0][2])
print ("Severe Probabality:",Predict2[0][3])

```

```

self.txt_V3_0.setText(str(Predict2[0][0]))
self.txt_V3_1.setText(str(Predict2[0][1]))
self.txt_V3_3.setText(str(Predict2[0][2]))
self.txt_V3_4.setText(str(Predict2[0][3]))

Base_model3 = tf.keras.applications.MobileNetV2(input_shape=input_shape,
input_tensor=model_input, include_top=False, weights=None)
for layer in Base_model3.layers:
    layer.trainable = True
Base_model3_last_layer = Base_model3.get_layer('out_relu')
Base_model3_last_output = Base_model3_last_layer.output
x3 = tf.keras.layers.GlobalAveragePooling2D()(Base_model3_last_output)
x3 = tf.keras.layers.Dropout(0.5)(x3)
x3 = tf.keras.layers.Dense(512, activation='relu')(x3)
x3 = tf.keras.layers.Dropout(0.5)(x3)
final_output3 = tf.keras.layers.Dense(4, activation='softmax', name='final_output3')(x3)
MobileNetV2_model = tf.keras.models.Model(model_input, final_output3)
metric_list = ["accuracy"]
optimizer = tf.keras.optimizers.Adam()
MobileNetV2_model.compile(optimizer=optimizer,
loss="categorical_crossentropy", metrics=metric_list)
if getattr(sys, 'frozen', False): # Running as a bundled executable
    base_path = sys._MEIPASS
else:
    base_path = os.path.abspath(".")

# Construct the full path to the weights file
weights_path2 = os.path.join(base_path, "WEIGHT",
"Weight_MobileNetV2_Optimal_(Ori).h5")

# Load the model weights
MobileNetV2_model.load_weights(weights_path2)
print ("MobileNetV2 Start predict.....")

```



```

for i in trange(10):
    sleep(0.1)
    Predict3=MobileNetV2_model.predict(x)
    pass
print ("Normal Probabality:",Predict3[0][0])
print ("Mild Probabality:",Predict3[0][1])
print ("Moderate Probabality:",Predict3[0][2])
print ("Severe Probabality:",Predict3[0][3])

self.txt_V2_0.setText(str(Predict3[0][0]))
self.txt_V2_1.setText(str(Predict3[0][1]))
self.txt_V2_2.setText(str(Predict3[0][2]))
self.txt_V2_3.setText(str(Predict3[0][3]))

def ensemble(models, model_input):
    outputs = [model.outputs[0] for model in models]
    y=tf.keras.layers.Average()(outputs)
    model =tf.keras.Model(model_input,y,name='ensemble')
    return model

ensemble_model =
ensemble([DensNet201_model,InceptionV3_model,MobileNetV2_model], model_input)
metric_list = ["accuracy"]
optimizer =tf.keras.optimizers.Adam()
ensemble_model.compile(optimizer=optimizer,
loss="categorical_crossentropy", metrics=metric_list)

print ("Ensemble Start predict.....")

for i in trange(10):
    sleep(0.1)
    Predict=ensemble_model.predict(x)
    pass

```

```

print ("Normal Probabality:",Predict[0][0])
print ("Mild Probabality:",Predict[0][1])
print ("Moderate Probabality:",Predict[0][2])
print ("Severe Probabality:",Predict[0][3])

self.txt_ESL_0.setText(str(Predict[0][0]))
self.txt_ESL_1.setText(str(Predict[0][1]))
self.txt_ESL_2.setText(str(Predict[0][2]))
self.txt_ESL_3.setText(str(Predict[0][3]))

if Predict[0][0]>=0.5:
    self.txt_ESL_4.setText("NORMAL (0)")
    print ("Diagnosis: NORMAL")
elif Predict[0][1]>=0.5:
    self.txt_ESL_4.setText("MILD (1)")
    print ("Diagnosis: MILD")
elif Predict[0][2]>=0.5:
    self.txt_ESL_4.setText("MODERATE (2)")
    print ("Diagnosis: MODERATE")
elif Predict[0][3]>=0.5:
    self.txt_ESL_4.setText("SEVERE (3)")
    print ("Diagnosis: SEVERE")

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

```

## 7.4 RESULTS AND DISCUSSIONS

The web application is designed using PyQt5 framework using the python programming language in the VS code IDE and when the code is run using the python run Main RD.py command the following is displayed on the preferred browser using the local host as shown in Fig 7.7.

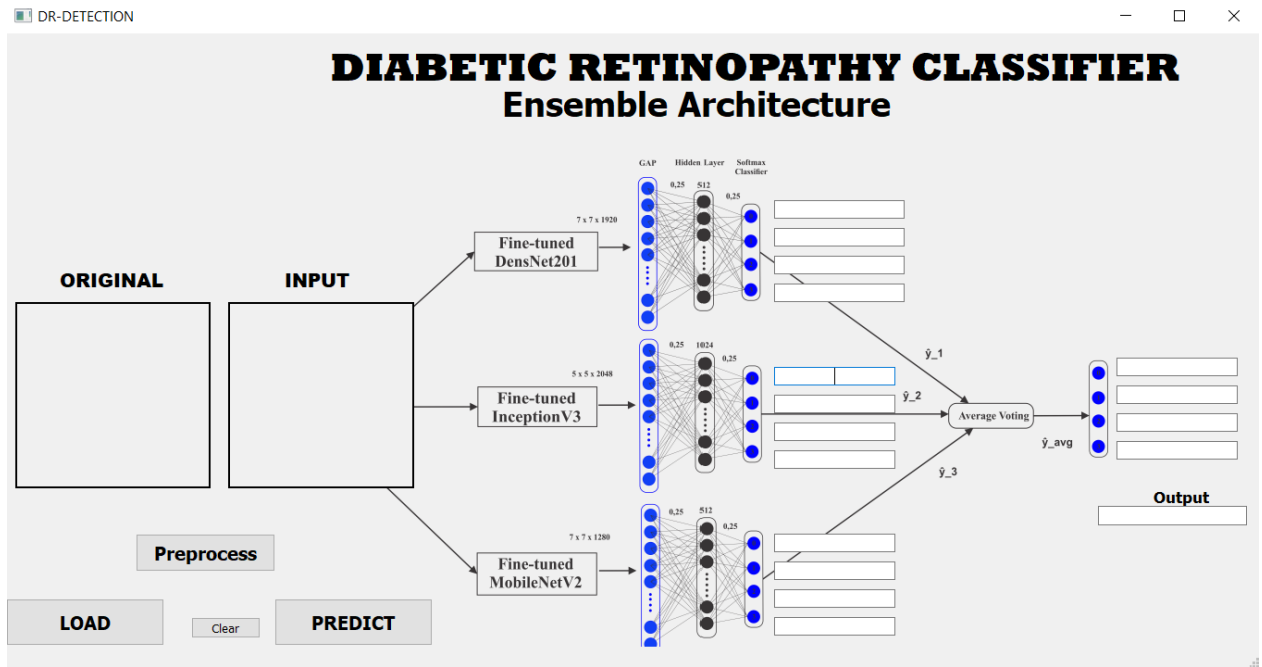


Fig 7.7 Interface of the designed Application

To upload the necessary fundus image of the patient's eye, the user has to click on the Load files option on the left-bottom corner of the website or they can simply drag and drop the image from the preferred location and click on Upload image button to upload fundus as shown in Fig 7.8.

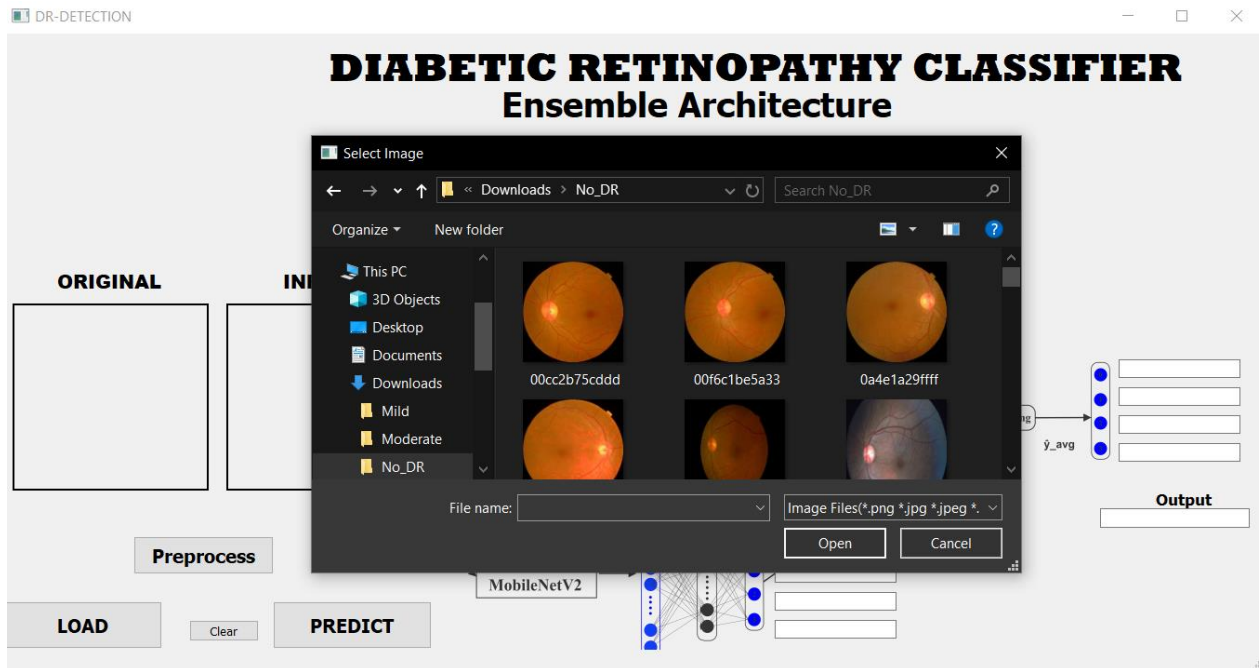


Fig 7.8 Uploading the input image to the Application

The uploaded image is displayed to the user just beside the box to the left portion of the application and to see the preprocessed image, the user needs to click on Preprocess button to display the processed image of the fundus image as shown in Fig 7.9.

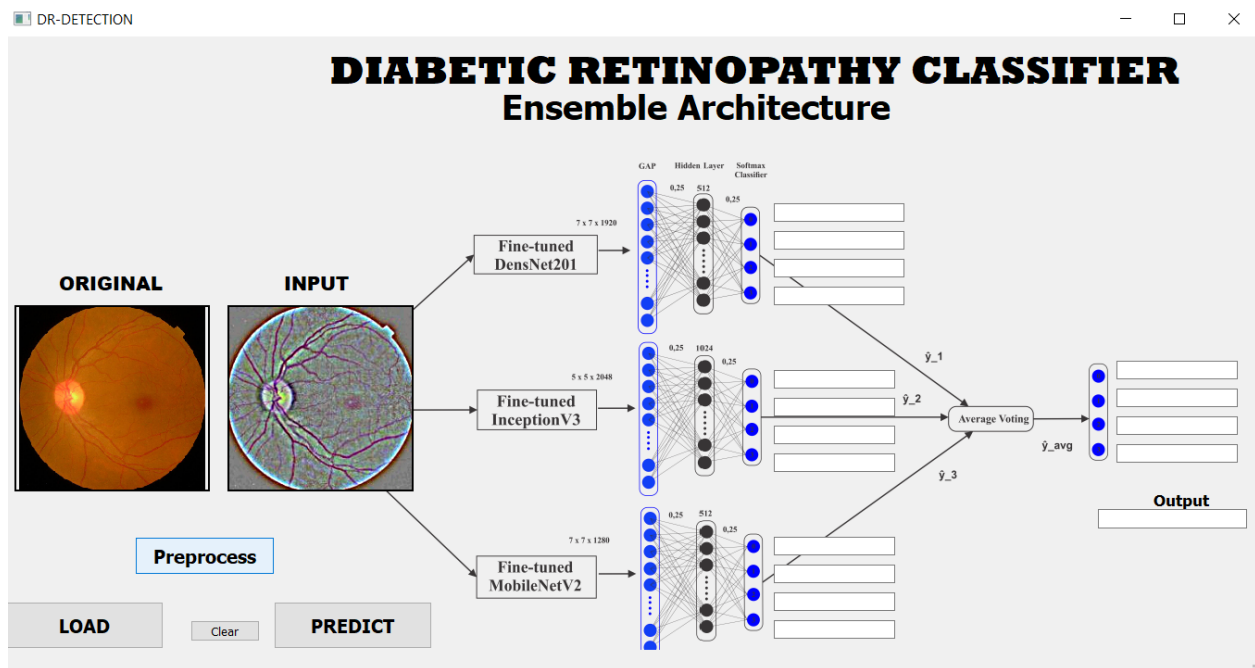


Fig 7.9 Displaying Preprocessed Uploaded Image to The User

After clicking on the Predict button, the website using the model in the backend (“ensemble model.h5” file), the GUI application displays the affected area in image

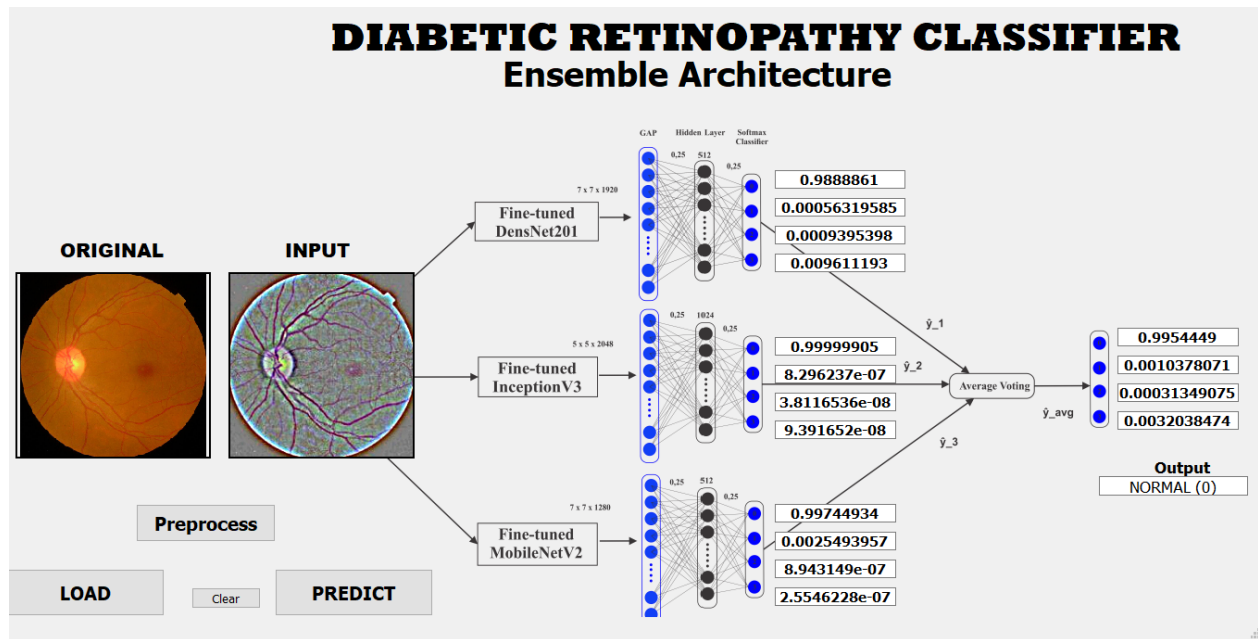


Fig 7.10 Displaying the severity grade of the affected eye

The accuracy of the proposed model is obtained ~94% as shown below in following epoch values. These accuracy scores are subjected to various parameters and various data preprocessing techniques as shown in Fig 7.12.

#### EVALUATE ENSEMBLE ON VALIDATION DATA

```
loss_val, acc_val = ensemble_model.evaluate(X_val, y_val_ohe, batch_size=1, verbose=1)
print("Validation: accuracy = %f ; loss_v = %f" % (acc_val, loss_val))
```

```
1500/1500 [=====] - 65s 43ms/step - loss: 0.1613 - accuracy: 0.9507
Validation: accuracy = 0.950667 ; loss_v = 0.161301
```

Fig 7.12 The Accuracy Scores of the Proposed Mode

## 8. TESTING

### 8.1 INTRODUCTION TO TESTING

Software Testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. It involves the execution of a software component or system component to evaluate one or more properties of interest. It is required for evaluating the system. This phase is the critical phase of software quality assurance and presents the ultimate view of coding and also ensure the improvement in the performance of the system.

#### 8.1.1 Importance of Testing

The importance of software testing is imperative. A lot of times this process is skipped, therefore, the product and business might suffer. To understand the importance of testing, here are some key points to explain

- ☐ Software Testing saves money
- ☐ Provides Security
- ☐ Improves Product Quality
- ☐ Customer satisfaction

The main idea behind the testing is to reduce the errors and do it with a minimum time and effort.

#### 8.1.2 Benefits of Testing

- Cost-Effective : It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix the bugs discovered.
- Security : It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
- Product quality : It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

- Customer Satisfaction : The main aim of any product is to give satisfaction to their customers.

UI/UX Testing ensures the best user experience.

### 8.1.3 Different types of Testing

Unit Testing : Unit tests are very low level, close to the source of your application. They consist in testing individual methods and functions of the classes, components or modules used by your software. Unit tests are in general quite cheap to automate and can be run very quickly by a Continuous integration server.

Integration Testing : Integration tests verify that different modules or services used by your application work well together. For example, it can be testing the interaction with the database or making sure that microservices work together as expected. These types of tests are more expensive to run as they require multiple parts of the application to be up and running.

Functional Tests : Functional tests focus on the business requirements of an application. They only verify the output of an action and do not check the intermediate states of the system when performing that action.

There is sometimes a confusion between integration tests and functional tests as they both require multiple components to interact with each other. The difference is that an integration test may simply verify that you can query the database while a functional test would expect to get a specific value from the database as defined by the product requirements.

Regression Testing : Regression testing is a crucial stage for the product & very useful for the developers to identify the stability of the product with the changing requirements. Regression testing is a testing that is done to verify that a code change in the software does not impact the existing functionality of the product.

System Testing : System testing of software or hardware is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system.

Performance Testing : It checks the speed, response time, reliability, resource usage, scalability of a software program under their expected workload. The purpose of Performance Testing is not to find functional defects but to eliminate performance bottlenecks in the software or device.

Alpha Testing : This is a form of internal acceptance testing performed mainly by the in- house software QA and testing teams. Alpha testing is the last testing done by the test teams at the development site after the acceptance testing and before releasing the software for the beta test. It can

also be done by the potential users or customers of the application. But still, this is a form of in-house acceptance testing.

Beta Testing : This is a testing stage followed by the internal full alpha test cycle. This is the final testing phase where the companies release the software to a few external user groups outside the company test teams or employees. This initial software version is known as the beta version. Most companies gather user feedback in this release.

Black Box Testing : It is also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

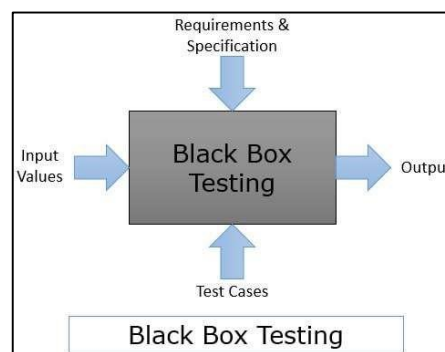


Fig 8.1 Black Box Testing

White Box Testing: White box testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/ implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential.



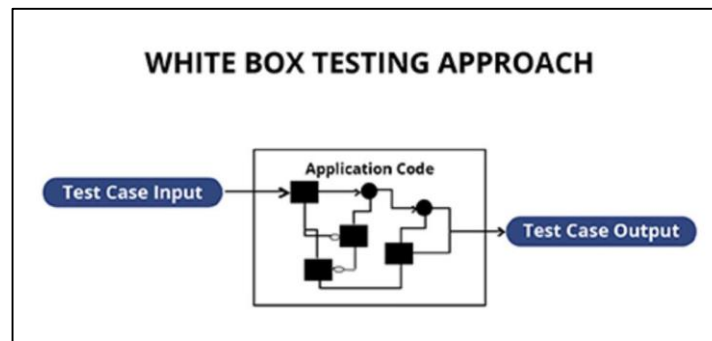


Fig 8.2 White Box Testing

## 8.2 TEST CODE

### EVALUATE DENSNET ON TEST DATA

```

1 loss_test, acc_test = DensNet201_model.evaluate(X_test, y_test_ohe, batch_size=1, verbose=1)
  print("Test: accuracy = %f ; loss_v = %f" % (acc_test, loss_test))

```

500/500 [=====] - 13s 26ms/step - loss: 0.1846 - accuracy: 0.9360

Test: accuracy = 0.936000 ; loss\_v = 0.184588

[+ Code](#)

[+ Markdown](#)

```

1 Test_predict = DensNet201_model.predict(X_test)

  Prdict_label = np.argmax(Test_predict, -1)
  Actual_label = y_test

  print('Accuracy on Test Data: %2.2f%%' % (100*accuracy_score(Actual_label, Prdict_label)))
  print(classification_report(Actual_label, Prdict_label))

```

Accuracy on Test Data: 93.60%

	precision	recall	f1-score	support
0	1.00	0.98	0.99	125
1	0.93	0.97	0.95	125
2	0.90	0.88	0.89	125
3	0.91	0.91	0.91	125
accuracy			0.94	500
macro avg	0.94	0.94	0.94	500
weighted avg	0.94	0.94	0.94	500

## EVALUATE INCEPTIONV3 ON TEST DATA

```
loss_test, acc_test = InceptionV3_model.evaluate(X_test, y_test_one, batch_size=1, verbose=1)
print("Test: accuracy = %f ; loss_v = %f" % (acc_test, loss_test))
```

```
500/500 [=====] - 7s 14ms/step - loss: 0.1954 - accuracy: 0.9420
Test: accuracy = 0.942000 ; loss_v = 0.195404
```

[+ Code](#)[+ Markdown](#)

```
Test_predict = InceptionV3_model.predict(X_test)

Prdict_label = np.argmax(Test_predict, -1)
Actual_label = y_test

print('Accuracy on Test Data: %2.2f%%' % (100*accuracy_score(Actual_label, Prdict_label)))
print(classification_report(Actual_label, Prdict_label))
```

Accuracy on Test Data: 94.20%

	precision	recall	f1-score	support
0	0.99	0.99	0.99	125
1	0.94	0.98	0.96	125
2	0.90	0.89	0.89	125
3	0.94	0.91	0.93	125
accuracy			0.94	500
macro avg	0.94	0.94	0.94	500
weighted avg	0.94	0.94	0.94	500

## EVALUATE MOBILENETV2 ON TEST DATA

```
loss_test, acc_test = MobileNetV2_model.evaluate(X_test,y_test_ohe,batch_size=1, verbose=1)
print("Test: accuracy = %f ; loss_v = %f" % (acc_test, loss_test))
]
```

500/500 [=====] - 2s 4ms/step - loss: 0.2947 - accuracy: 0.8900  
Test: accuracy = 0.890000 ; loss\_v = 0.294664

[+ Code](#)[+ Markdown](#)

```
Test_predict = MobileNetV2_model.predict(X_test)

Prdict_label = np.argmax(Test_predict, -1)
Actual_label = y_test

print('Accuracy on Test Data: %2.2f%%' % (100*accuracy_score(Actual_label, Prdict_label)))
print(classification_report(Actual_label, Prdict_label))
]
```

Accuracy on Test Data: 89.00%

	precision	recall	f1-score	support
0	0.98	1.00	0.99	125
1	0.85	0.92	0.88	125
2	0.80	0.85	0.82	125
3	0.94	0.79	0.86	125
accuracy			0.89	500
macro avg	0.89	0.89	0.89	500
weighted avg	0.89	0.89	0.89	500


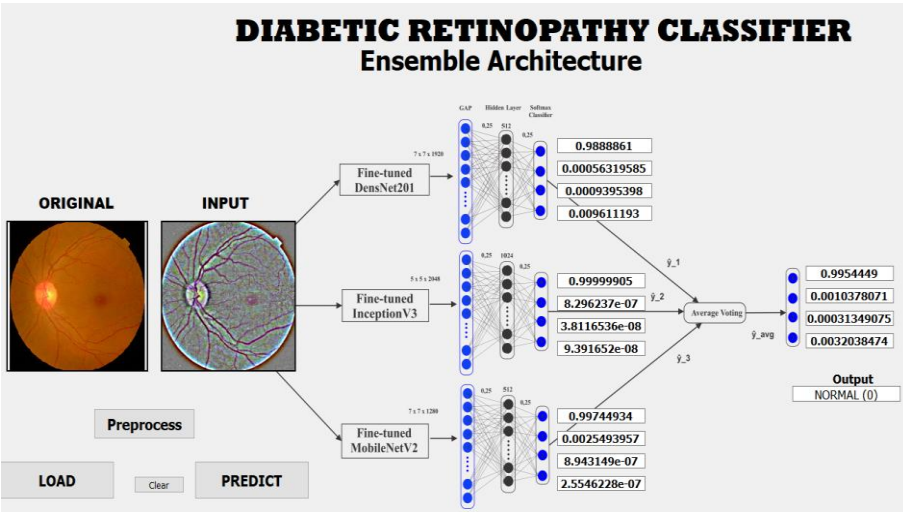
### 8.3 TEST CASES AND OUTPUTS :

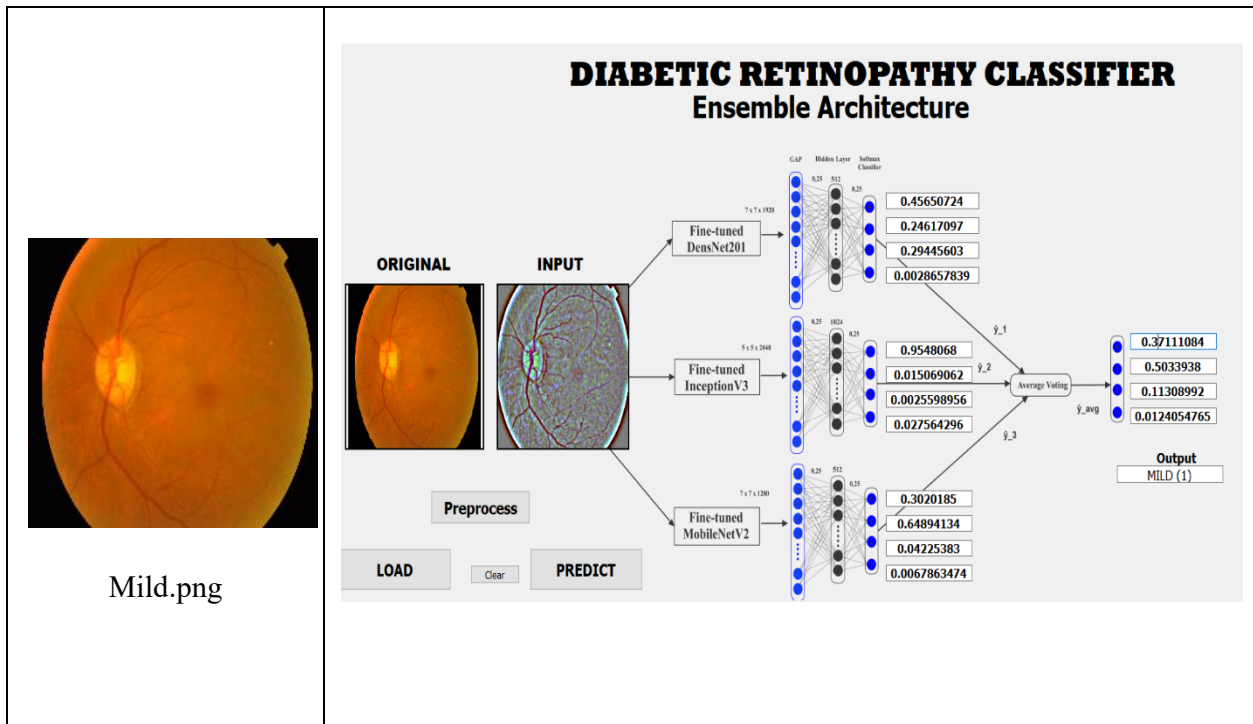
#### Categories of Input Images :



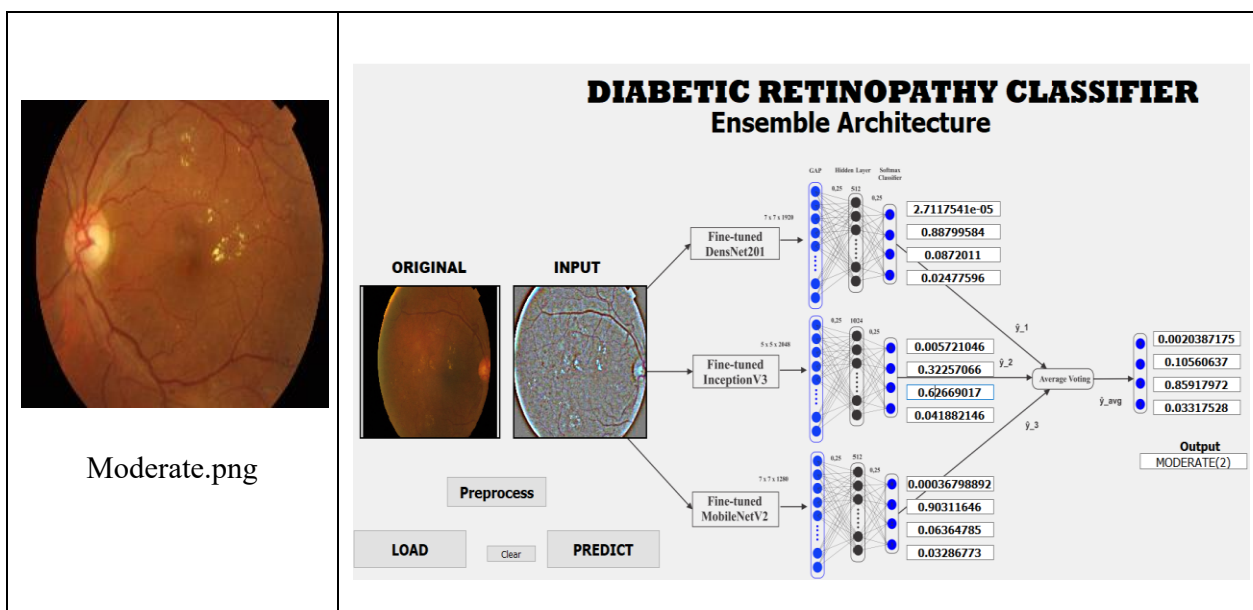
Fig 8.3 Different Grades of Fundus images in the Dataset

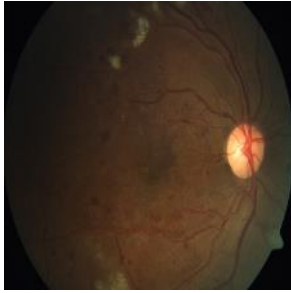
Table 8.4 Depicting the Severity of Patient's eye for different Samples

Input Image	Actual Output
 <p>No_DR.png</p>	<p><b>DIABETIC RETINOPATHY CLASSIFIER</b> Ensemble Architecture</p>  <p>The diagram illustrates the ensemble architecture. It starts with an 'ORIGINAL' fundus image, which is processed into an 'INPUT' image. This input is then fed into three parallel fine-tuned networks: DenseNet201 (7x7x1920), InceptionV3 (5x5x2048), and MobileNetV2 (7x7x1280). Each network produces a set of outputs (y1, y2, y3) which are then averaged to produce the final output (y_avg). The final output is 'NORMAL (0)'.</p>

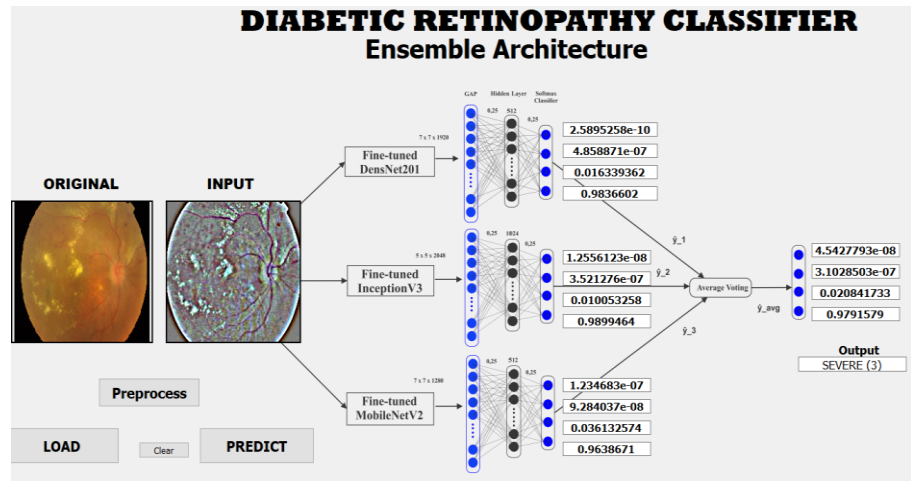


The above output results of the input eye image as displayed in the Table 8.4 constitute the various cases of Diabetic retinopathy in the eye of the infected person and based on the intensity of the disease in the explainability section, the severe the condition of the patient increases and the immediate diagnosis is taken care of. This way, the user gets intimated about the severity of the DR in the eye and therefore, appropriate medication would be taken .





Severe.png



## 9. CONCLUSION

Our diabetic retinopathy detection project represents a significant step forward in leveraging machine learning for medical diagnosis. With an ensemble approach boasting an impressive good accuracy the efficacy of our model in identifying signs of diabetic retinopathy is unmistakable. This amalgamation of sophisticated algorithms and medical expertise holds promise for enhancing early detection and intervention in diabetic retinopathy cases, potentially saving countless individuals from vision loss. Moving forward, our focus will be on streamlining the model's response time and ensuring scalability for seamless integration into real-world clinical settings. By continually refining and advancing our techniques, we aim to make diabetic retinopathy diagnosis not only more accurate but also more accessible to patients worldwide, ushering in a new era of preventive healthcare.

## 10. FUTURE SCOPE

Innovations in diabetic retinopathy detection with a focus on scalability and response time are crucial for enhancing accessibility and effectiveness. Future endeavors could entail the development of portable retinal imaging devices, integrating AI algorithms into screening tools, and creating cloud-based platforms for rapid image analysis. These strategies aim to streamline screening processes, prioritize patient referrals, and ultimately mitigate the burden of diabetic retinopathy.



## 11. BIBLIOGRAPHY

- [1] Dataset used <https://www.kaggle.com/c/diabetic-retinopathy-detection/data> from kaggle.com. D
- [2] Harry Pratta, Frans Coenenb, Deborah M Broadbentc, Simon P Hardinga, Yalin Zhenga  
<https://www.sciencedirect.com/science/article/pii/S1877050916311929>
- [3] <https://www.sciencedirect.com/science/article/pii/S1319157823000228>
- [4] Gayathri, S.; Gopi, V.P.; Palanisamy, P. Diabetic retinopathy classification based on multipath CNN and machine learning classifiers. *Phys. Eng. Sci. Med.* **2021**, *44*, 639–653.
- [5] <https://www.geeksforgeeks.org/python-introduction-to-pyqt5/>
- [6] Adarsh, P., Jeyakumari, D.. Multiclass svm-based automated diagnosis of diabetic retinopathy.  
In: Communications and Signal Processing (ICCSP), 2013 International Conference on. IEEE; 2013, p. 206–210.
- [7] <https://www.mdpi.com/1832596>