# CNN-based Classification of Diabetic Retinopathy Stages in Fundus Images

**A Project Report submitted in partial fulfillment of the requirements for the award of the degree of**

**BACHELOR OF TECHNOLOGY**
**IN**
**COMPUTER SCIENCE AND ENGINEERING**
**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**Submitted by**

**Krithi Kethan Pujari -222010322009**
**Aryan Sahu - 222010322013**
**Ajay Esakki - 222010322039**
**Sai Tarun Maryala- 222010323009**

**Under the esteemed guidance of**

**Dr. Nandita Bhanja Chaudhuri**

**Assistant Professor**


**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**
**GITAM**
**(Deemed to be University)**

**HYDERABAD**

**APRIL - 2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GITAM SCHOOL OF TECHNOLOGY**

**GITAM**

**(Deemed to be University)**

# DECLARATION

I/We, hereby declare that the project report entitled "**CNN-based Classification of Diabetic Retinopathy Stages in Fundus Images**" is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:02-04-2024

| Registration No(s). | Name(s) | Signature(s) |
|---|---|---|
| **222010322009** | **Pujari Krithik Kethan** | |
| **222010322013** | **Aryan Sahu** | |
| **222010322039** | **Ajay Esakki** | |
| **222010323009** | **Sai Tarun Maryala** | |

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GITAM SCHOOL OF TECHNOLOGY**
**GITAM**
**(Deemed to be University)**


# CERTIFICATE

This is to certify that the project report entitled "**CNN-based Classification of Diabetic Retinopathy Stages in Fundus Images**" is a bonafide record of work carried out by **p. Krithik (222010322009), Aryan Sahu (222010322013), E. Ajay (222010322039), M. Sai Tarun (222010323009)** students submitted in partial fulfillment of requirements for the award of degree of Bachelors of Technology in Computer Science and Engineering.

**Project Guide**

Dr. Nandita Bhanja Chaudhuri

Assistant Professor

**Head of the Department**

Dr.Mahaboob Basha

Professor & HOD

# ACKNOWLEDGEMENT

Our project report would not have been successful without the help of several people. We would like to thank the personalities who were part of our seminar in numerous ways, those who gave us outstanding support from the birth of the seminar.

We are extremely thankful to our honorable Pro-Vice-Chancellor, **Prof. D. Sambasiva Rao**, for providing the necessary infrastructure and resources for the accomplishment of our seminar. We are highly indebted to **Prof. N. Seetharamaiah**, Associate Director, School of Technology, for his support during the tenure of the seminar.

We are very much obliged to our beloved **Prof. Dr. Mahaboob Basha**, Head of the Department of Computer Science & Engineering, for providing the opportunity to undertake this seminar and encouragement in the completion of this seminar.

We hereby wish to express our deep sense of gratitude to **Dr. S Aparna**, Project Coordinator,Department of Computer Science and Engineering, School of Technology and to our guide, **Dr. Nandita Bhanja Chaudhuri**, Assistant Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by them for the success of the project report.

We are also thankful to all the Computer Science and Engineering department staff members who have cooperated in making our seminar a success. We would like to thank all our parents and friends who extended their help, encouragement, and moral support directly or indirectly in our seminar work.

Sincerely,

**Krithik Kethan Pujari -222010322009**
**Aryan Sahu - 222010322013**
**Ajay Esakki - 222010322039**
**Sai Tarun Maryala- 222010323009**

# ABSTRACT

Diabetic retinopathy and diabetic macular edema result from chronic damage to the neurovascular structures of the retina. The pathophysiology of retinal damage remains uncertain but includes metabolic and neuroinflammatory insults. These mechanisms are addressed by intensive metabolic control of the systemic disease and by the use of ocular anti-inflammatory agents, including vascular endothelial growth factor inhibitors and corticosteroids. Improved understanding of the ocular and systemic mechanisms that underlie diabetic retinopathy will lead to improved means to diagnose and treat retinopathy and better maintain vision. Diabetic retinopathy and diabetic macular edema result from chronic damage to the neurovascular structures of the retina. Our project uses a deep learning classification technique using CNN pre-trained model resnet152 to classify severity levels of DR ranging from 0 which means no presence of diabetic retinopathy to 4 which means the presence of proliferative diabetic retinopathy. To provide access to ocular care to the low-income population with diabetes and to offer early assessment and referral for timely treatment. To establish a health-care network for the referral of patients with diabetic retinopathy for screening, evaluation, ocular diagnosis, and treatment if required. To strengthen the technical capacity of medical teams, providing adequate training to improve the efficiency and quality of diabetic retinopathy services.

# TABLE OF CONTENT


**Declaration** i

**Certificate** ii

**Acknowledgement** iii

**Abstract** iv

**Table of Content** v

**List of Figures** viii

**1.Introduction** **1**

1.1 Problem Statement 1

1.2 Objective 1

1.3 Limitations 2

1.4 Outcome 3

1.5 Applications 3

**2.Literature Review** **5**

**3.Problem Analysis and Requirements** **7**

3.1 Problem statement 1 7

3.2 Disadvantages 7

3.3 Scope of Project 7

3.4 Functional Requirements 8

3.5 Non-Functional Requirements 9

3.5.1 Reliability 9

3.5.2 Maintainability 9

3.5.3 Performance 9

3.5.4 Portability 9

3.5.5 Scalability 10

3.5.6 Flexibility 10

**4.System Architecture** **11**

4.1 Data Preparation 13

4.2 Data Processing 13

4.3 Modeling 13

4.3.1 Model Training 13

4.3.2 Model Evaluation 13

4.3.3 Saving Progress 14

4.3.4 Streamlit App Setup 14

4.3.5 Streamlit App Development 14

4.3.6 Perform Inference 14

4.3.7 Display Results 14

4.4 Uml Diagram 15
4.4.1 Use Case Diagram 15
4.4.2 Activity Diagram 15
4.4.3 Class Diagram 16
4.4.4 Sequence Diagram 16
4.4.5 Interaction Diagrams 18
18

**5. System Design** **19**

5.1 Proposed System Architecture 19
5.1.1 Architecture Diagram Of Diabetic Retinopathy Detection CNN 19
20
5.2 Model Training 20
5.2.1 Data Preprocessing 20
5.2.2 Data Splitting 20
5.2.3 Model Compilation 20
5.2.4 Model Training 20
20
5.3 Model Evaluation 21
5.3.1 Test Data 21
5.3.2 Metrics 21
5.4 Model Deployment
5.4.1 Model Saving 21
21
5.4.2 Web Application 21
5.4.3 Image Upload 21
22
5.4.4 Inference 22
5.4.5 Result Display 22
5.5 Conclusion

**6. Tools and Technologies used** 23

6.1 Development 23
6.1.1 TensorFlow 23
6.1.2 Pandas 23
6.1.3 NumPy 23
6.1.4 OpenCV 23
6.2 Machine Learning: 23
6.2.1 TensorFlow 23
6.3 Data Processing and Analysis: 23
6.3.1Pandas 24
6.4 Web Development: 24
6.4.1 Streamlit 24
6.4.2 HTML/CSS/JavaScript 24
6.5 Version Control: 24
6.5.1 Git 24
6.6 Data Visualization 24
6.6.1 Matplotlib 24
6.7 Other Libraries: 24
6.7.1 tqdm 24
6.7.2 scikit-learn

**7. Code** 25-40
7.1 Obtaining Kaggle Dataset 25
7.2 Importing the necessary libraries 25
7.3 Processing of the dataset 26
7.4 Training and Testing of our model 31
7.5 Hosting the model on Streamlit 38

**8. Result** 41-46

**9. Conclusion** 47

**10. BIBLIOGRAPHY** 48

**11. REFERENCES** 49

# LIST OF FIGURES


| Fig. No. | Figure Name | Pg. No. |
|---|---|---|
| Fig. 1.2 | Retinal images showing blood vessels | 2 |
| Fig. 4 | System Architecture | 12 |
| Fig. 4.1 | Flow chart of the model | 13 |
| Fig. 4.4 | UML diagrams | 15-18 |
| Fig 7.3.1 | Diagnosis (i) | 26 |
| Fig 7.3.2 | Diagnosis (ii) | 27 |
| Fig 7.3.3 | 25 samples of retinal images after applying the Gaussian filter | 30 |
| Fig 7.4.1 | Accuracy graph | 35 |
| Fig 7.4.2 | Loss graph | 36 |
| Fig 8.1 | Frontend UI | 42 |
| Fig 8.2 | Upload retinal left | 43 |
| Fig 8.3 | Upload retinal right | 43 |
| Fig 8.4 | Result retinal images left | 44 |
| Fig 8.5 | Result retinal images right | 45 |
| Fig 8.6 | Final result | 45 |

# 1. INTRODUCTION

## 1.1 Problem Statement

Despite the significant advancements in medical technology and healthcare, Diabetic Retinopathy (DR) remains a major cause of vision impairment and blindness, particularly among individuals with diabetes mellitus. The primary challenge lies in the timely detection and management of DR, as early intervention is crucial for preventing irreversible vision loss. Traditional methods of DR screening, such as manual grading of retinal images by trained ophthalmologists, are resource-intensive, time-consuming, and often inaccessible in underserved regions. Moreover, the increasing prevalence of diabetes further exacerbates the need for efficient and scalable screening solutions to accommodate the growing demand for diabetic eye care.

## 1.2 Objective

The objective of this project is to develop and evaluate a deep learning-based approach for the automated detection of Diabetic Retinopathy. By leveraging state-of-the-art machine learning algorithms, we aim to create a scalable and efficient screening tool capable of accurately identifying retinal abnormalities associated with DR. Specifically, our objectives include:

- Designing and training a convolutional neural network (CNN) model using retinal images from the APTOS 2019 Blindness Detection Dataset.
- Evaluating the performance of the model in terms of sensitivity, specificity, and overall accuracy in detecting different stages of DR.
- Assessing the potential clinical utility of the automated DR detection system in improving access to timely diagnosis and management, particularly in resource-constrained settings.

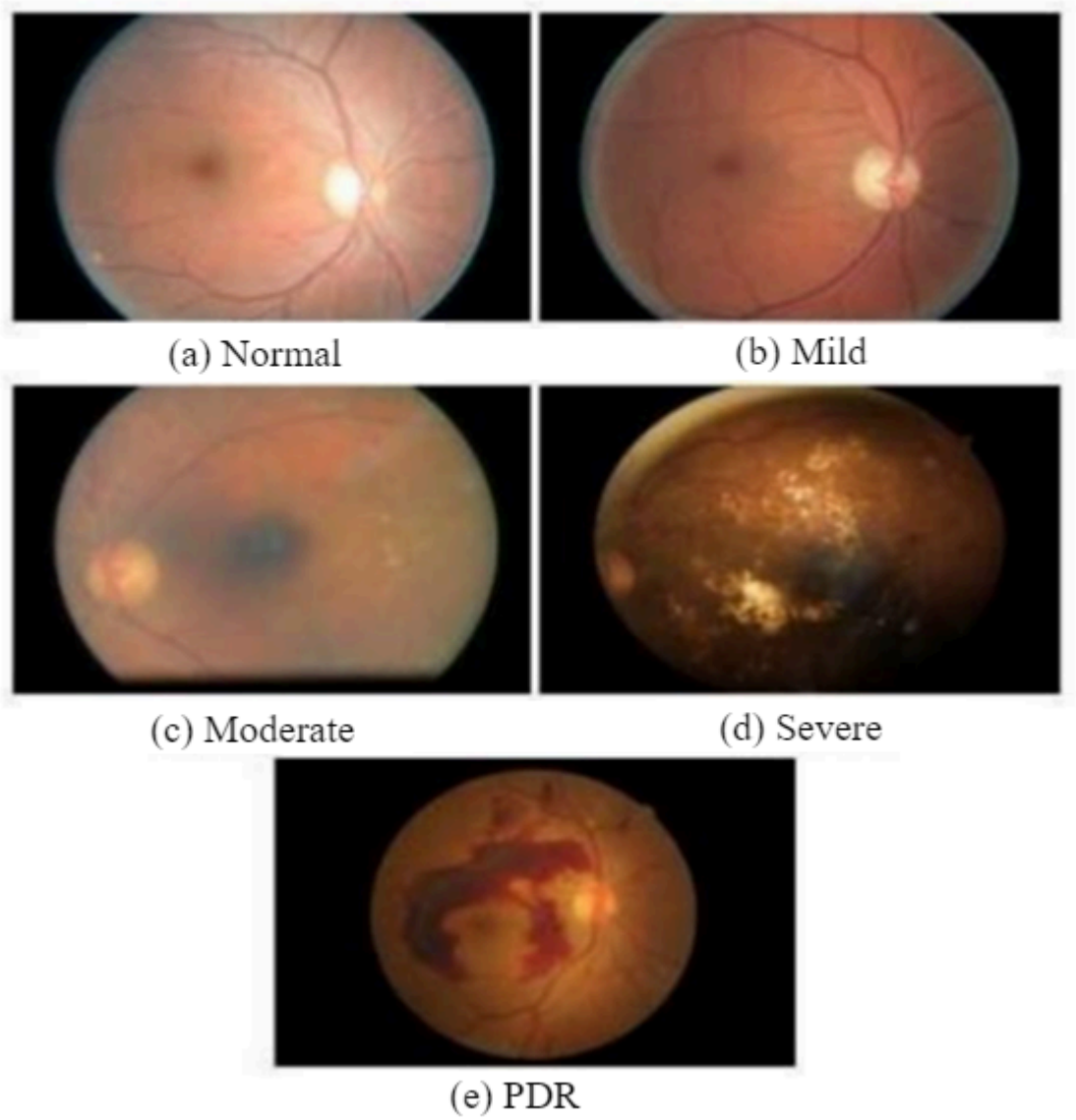


**fig 1.2: retinal images showing blood vessels**

## 1.3 Limitations

While the retinopathy detection project holds promise for improving healthcare outcomes, it grapples with several limitations. One significant challenge lies in the quality and quantity of data available for training the detection models, as biased or insufficient datasets can impact the system's accuracy and reliability. Additionally, the interpretability of deep learning models used in the project, like CNNs, poses a hurdle, as understanding how these models arrive at their diagnoses remains complex.

Reputational resources and infrastructure, which could be financially prohibitive for certain healthcare settings. Regulatory approval processes, such as those mandated by the FDA, present further hurdles, requiring adherence to rigorous standards and potentially delaying deployment. Finally, ethical and legal considerations, including patient privacy, consent, and liability, add layers of complexity to the project's implementation and acceptance within healthcare systems. Addressing these limitations will be crucial for realizing the full potential of retinopathy detection technology in clinical practice.

## 1.4 Outcomes

Our project yielded several significant outcomes in the development, evaluation, and interpretation of a deep learning model for automated Diabetic Retinopathy (DR) detection. Leveraging convolutional neural network (CNN) architecture, we successfully trained and validated a model using the APTOS 2019 Blindness Detection Dataset, achieving commendable performance metrics in terms of sensitivity, specificity, and overall accuracy for detecting different stages of DR. Additionally, we explored techniques to enhance the interpretability of the model, such as generating saliency maps and attention mechanisms, to elucidate the features influencing its predictions. Through thorough analysis and iterative refinement, we identified areas for improvement and optimization, laying the groundwork for future advancements in automated DR detection technology.

## 1.5 Application

Early Intervention: Retinopathy detection systems enable early identification of diabetic retinopathy, facilitating timely medical intervention and treatment to prevent vision loss.

Accessible Healthcare: Automated screening tools support telemedicine initiatives, making retinopathy screening more accessible to individuals in underserved areas or remote locations, thus bridging gaps in healthcare access.

Population Health Management: Integration of retinopathy detection technology into population screening programs allows for efficient screening of a large number of individuals, aiding in the early detection and management of diabetic retinopathy at scale.

Research Advancements: Data collected through retinopathy detection systems contribute to research efforts aimed at better understanding the progression of the disease, informing the development of new treatment modalities and enhancing patient outcomes.

Preventive Care: By facilitating early diagnosis and intervention, retinopathy detection technologies play a crucial role in preventive healthcare, empowering healthcare providers to implement strategies that mitigate the risk of vision loss among diabetic patients.

# 2. LITERATURE REVIEW

[1] Telemedicine for Diabetic Retinopathy Screening Ling Dai, Liang Wu With advances in digital image processing and communications, telemedicine can become a viable screening tool for patients at risk for developing diabetic retinopathy (DR). A deep-learning telemedicine platform achieved statistically significant sensitivities, specificities, and positive predictive values for both referrable and vision-threatening DR. A follow-up study is planned to further assess the system's ability to automatically detect hard exudates and hemorrhages compared with traditional examination techniques.

[2] EyeDeep-Net: Multi-class Diagnosis of Retinal Diseases Authors: Neha Sengar, Rakesh Chandra Joshi This paper provides a comprehensive overview of recent advancements in automated diagnosis of eye diseases using deep learning techniques applied to retinal fundus images. It encompasses studies on various eye conditions such as diabetic retinopathy, macular degeneration, glaucoma, and cataracts, showcasing the efficacy of deep learning models in accurate disease detection. Additionally, it discusses the availability of publicly accessible datasets like RFMiD for model training and evaluation, along with references to key works highlighting the significance of this research domain in addressing global eye health challenges.

[3] Detection of Diabetic Retinopathy using Convolutional Neural Networks for Feature Extraction and Classification (DRFEC) Authors: Dolly Das, Saroj Kumar Biswas This paper underscores the significance of automated systems for diabetic retinopathy (DR) detection due to its prevalence and potential for blindness. It highlights the necessity for efficient and reliable diagnostic tools given the limitations of manual assessment methods. The study evaluates various deep learning architectures for DR classification, emphasizing the importance of accurate feature extraction and image classification from fundus images. It identifies EfficientNetB4 as the most optimal model, showcasing its high accuracy and robust performance in DR detection compared to other state-of-the-art networks. Additionally, the review points out the ongoing research efforts to enhance diagnostic accuracy through advancements in convolutional neural network (CNN) technology for DR assessment.

[4] Early Detection of Diabetic Retinopathy based on Deep Learning and Ultra-wide-field Fundus Images Authors: Kangrok Oh, Hae Min Kang etc . People with vision-threatening diabetic retinopathy are likely to experience enhanced social and emotional strain. Those with both vision-threatening diabetic retinopathy and psychosocial problems may have significantly reduced levels of functioning compared with psychologically healthy counterparts. However, the emotional and social health consequences of diabetic retinopathy have not yet been systematically explored. Adverse emotional responses include fear, anxiety, vulnerability, guilt, loss of confidence, anger, and stress.

# 3. PROBLEM ANALYSIS

## 3.1 Problem Statement

Diabetic Retinopathy (DR) poses a significant threat to vision, particularly among working-age individuals, and timely diagnosis is crucial to prevent severe vision impairment. Traditional methods of diagnosis, relying on manual grading of retinal images, are labor-intensive, time-consuming, and prone to inter-observer variability. The increasing prevalence of DR necessitates scalable and accessible screening solutions. Automated analysis of retinal images using deep learning models offers a promising approach for early detection, reducing manual workload and diagnosis costs. Our proposal aims to leverage pretrained convolutional neural network (CNN) models, such as CNN, ResNet, and VGG-16, to classify the severity levels of DR, addressing the need for efficient and accurate screening tools in healthcare systems.

## 3.2 Disadvantages

The manual grading system for DR diagnosis suffers from several flaws. Firstly, its reliance on subjective human assessment introduces variability in diagnoses, potentially leading to misinterpretation and delayed treatment. Moreover, the labor-intensive nature of manual grading results in significant time and resource expenditure, limiting its scalability and accessibility, especially in resource-constrained settings. Additionally, the expertise required for accurate manual grading may not be readily available in all healthcare settings, exacerbating disparities in access to timely diagnosis and treatment. Furthermore, the manual grading process is prone to errors and fatigue, particularly with large volumes of retinal images, compromising the accuracy of diagnoses. Overall, these limitations highlight the need for automated and scalable solutions for DR diagnosis to improve efficiency, accuracy, and accessibility in healthcare delivery.

## 3.3 Scope of Project

(i) To provide access to ocular care to the low-income population with diabetes and to offer early assessment and referral for timely treatment.

(ii) To establish a health-care network for the referral of patients with diabetic retinopathy for screening, evaluation and ocular diagnosis, and treatment if required. Automated Detection Of Diabetic Retinopathy Using Deep Learning Department Of Computer Science & Engineering, SOE, DSU 4

(iii) To strengthen the technical capacity of medical teams, providing adequate training to improve the efficiency and quality of diabetic retinopathy services.

Diabetes was once thought of as a disease of the affluent but it has now reached epidemic proportion in both developed and developing countries. Currently, at least 366 million people worldwide have diabetes, and this number is likely to increase as a result of an aging global population Globally, the number of people with DR will grow from 126.6 million in 2010 to 191.0 million by 2030, and we estimate that the number with vision-threatening diabetic retinopathy (VTDR) will increase from 37.3 million to 56.3 million, if prompt action is not taken.

## 3.4 Functional Requirements

**GUI Development:** Develop a user-friendly GUI using Python GUI Toolkit.

**Data Collection**: Gather a diverse dataset of diabetic retinopathy images.

**Image Classification:** Implement deep learning-based image classification for diabetic retinopathy detection. Use pre-trained CNN models for classification.

**Backend Implementation:** Develop backend using a deep learning framework.

**Database Connectivity:** Implement MySQL database connectivity for metadata management.

**Integration:** Integrate GUI with the backend for seamless communication.

## 3.5 Non-Functional Requirements

### 3.5.1 Reliability

The structure must be reliable and strong in giving the functionalities. The movements must be made unmistakable by the structure when a customer has revealed a couple of enhancements. The progressions made by the Programmer must be the Project pioneer and in addition the Test designer.

### 3.5.2 Maintainability

The system watching and upkeep should be fundamental and focus in its approach. There should not be an excess of occupations running on diverse machines such that it gets hard to screen whether the employments are running without lapses.

### 3.5.3 Performance

The framework will be utilized by numerous representatives all the while. Since the system will be encouraged on a single web server with a lone database server outside of anyone's ability to see, execution transforms into a significant concern. The structure should not capitulate when various customers would use everything the while. It should allow brisk accessibility to each and every piece of its customers. For instance, if two test specialists are all the while attempting to report the vicinity of a bug, then there ought not to be any irregularity at the same time.

### 3.5.4 Portability

The framework should to be effectively versatile to another framework. This is obliged when the web server, which is facilitating the framework gets adhered to because of a few issues, which requires the framework to be taken to another framework.

### 3.5.5 Scalability

The framework should be sufficiently adaptable to include new functionalities at a later stage. There should be a run-of-the-mill channel, which can oblige the new functionalities.

### 3.5.6 Flexibility

Flexibility is the capacity of a framework to adjust to changing situations and circumstances and to adapt to changes in business approaches and rules. An adaptable framework is anything but difficult to reconfigure or adjust because of diverse client and framework prerequisites. The deliberate division of concerns between the trough and motor parts helps adaptability as just a little bit of the framework is influenced when strategies or principles change

# 4. THE SYSTEM ARCHITECTURE

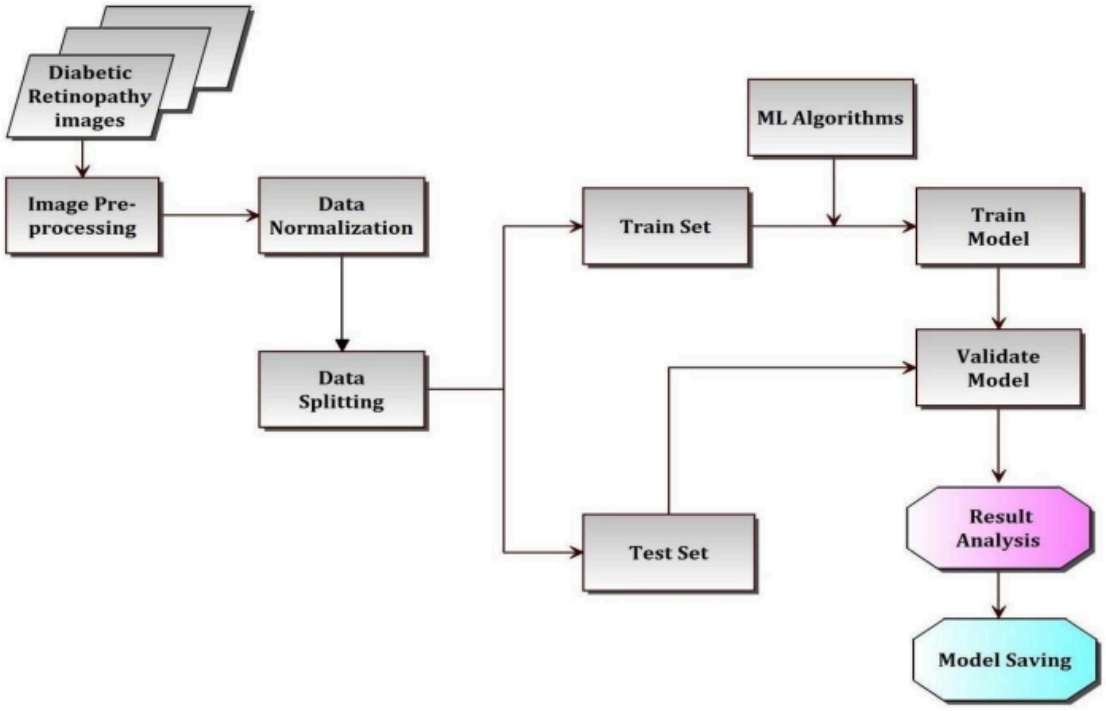


**fig 4: System Architecture**

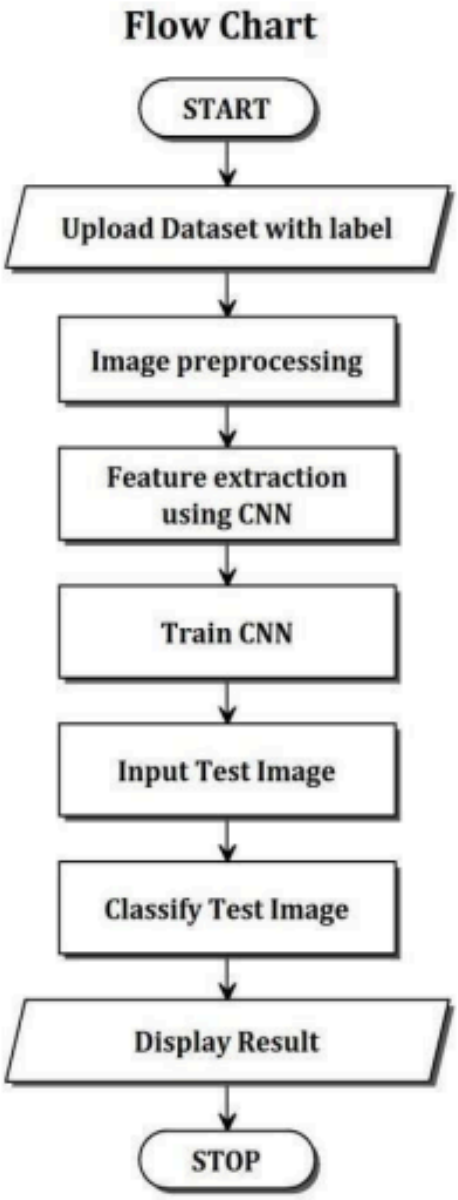


**fig 4.1: flowchart of the model**

## 4.1 Data Preparation

- Download the dataset from Kaggle and extract the files.

- Import necessary libraries for data processing and modeling.

## 4.2. Data Processing

- Read the CSV file containing image paths and labels.

- Explore and visualize the distribution of labels.

- Create a data frame for image paths and labels.

- Filter images by applying a Gaussian filter to reduce noise.

- Split the data into training and testing sets.

## 4.3. Modeling

- Build a convolutional neural network (CNN) model using TensorFlow Keras.

- The model consists of several convolutional layers with pooling, followed by fully connected layers with dropout regularization.

- The last layer uses softmax activation for multi-class classification.

### 4.3.1 Model Training

- Split the training set into K folds for cross-validation using StratifiedKFold from scikit-learn library.

- Perform cross-validation training on each fold, using image data generators to preprocess images during training.

- Use early stopping to prevent overfitting and save the best-performing model.

### 4.3.2 Model Evaluation

Evaluate the trained model on the testing set using ImageDataGenerator to preprocess images.

### 4.3.3 Saving Progress

Save the trained model in H5 format for future use along with information about loss, accuracy, validation loss, and validation accuracy in a JSON file

### 4.3.4 Streamlit App Setup

Mount Google Drive to Colab to save uploaded images or any other required files in Google Drive.

### 4.3.5 Streamlit App Development

Develop a streamlit app that allows users to upload left and right-eye images through a sidebar interface.

### 4.3.6 Perform Inference

Perform inference on uploaded left eye image and right eye image using the loaded model by expanding dimensions of input image tensors

### 4.3.7 Display Results

Display predicted results (class probabilities) for left eye result and right eye.

## 4.4 UML DIAGRAM

### 4.4.1 Use Case Diagrams

A use case diagram is a visual representation of the interactions between users (actors) and a system in a specific environment to achieve a particular goal. It depicts the functionality of a system from the user's perspective. Use case diagrams are crucial in software development as they provide a visual representation of how users interact with a system, aiding in requirements understanding, communication, scope definition, and validation. They serve as a basis for testing, ensuring the system behaves as intended, making them an integral part of the development process.

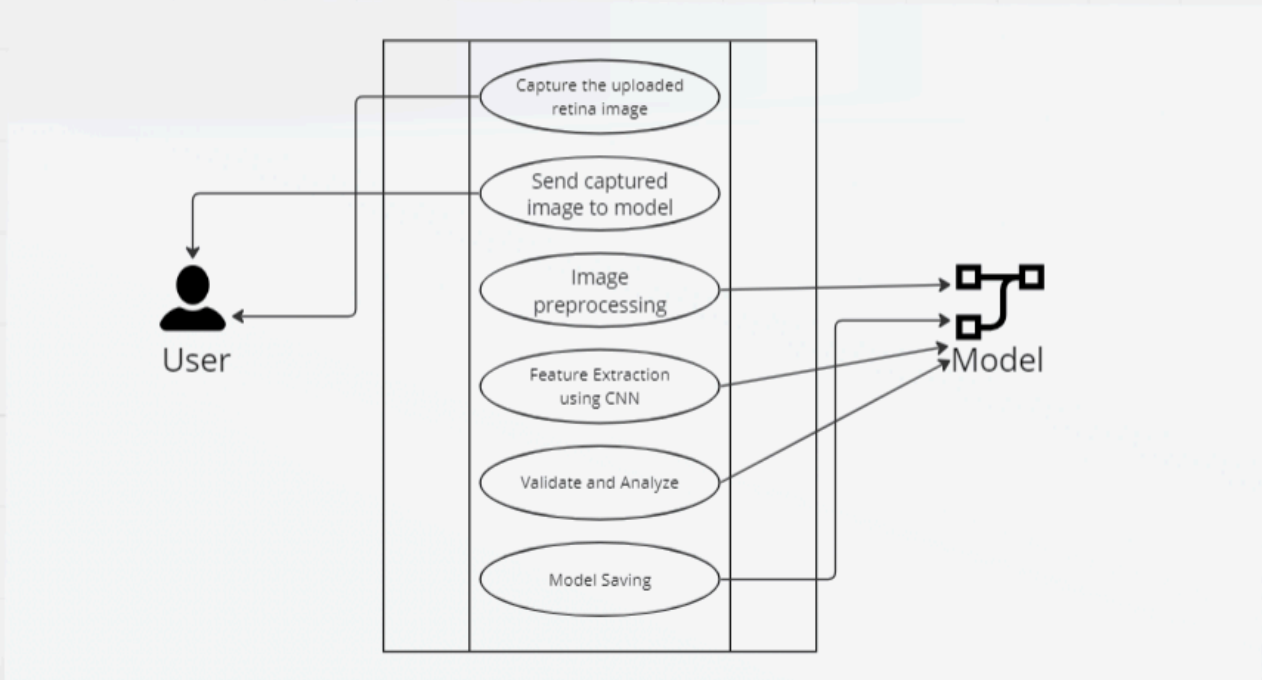


**fig 4.4.2: Activity Diagram**

### 4.4.2 Activity Diagram

The activity diagram visually represents the flow of activities or actions in a system or process. It includes activities, transitions, decision points, forks, and joins. Activities are the actions or steps in the process, while transitions show the flow of control between activities. Decision points represent points where decisions must be made, and forks and joins indicate parallel activities that can be performed simultaneously or converging parallel activities. Activity diagrams are

important for visualizing process flows, modeling system behavior, identifying bottlenecks, documenting processes, and communicating complex workflows to stakeholders. Overall, both diagrams play crucial roles in understanding, designing, and communicating the functionality and behavior of systems and processes.

### 4.4.3 Class Diagram

Class diagrams are a fundamental part of object-oriented modeling that represent the static structure of a system. They illustrate the classes, attributes, operations, and relationships among objects. Classes represent the blueprint for creating objects, while attributes define the properties of objects. Operations, also known as methods, represent the behaviors or actions that objects can perform. Relationships, such as association, aggregation, and inheritance, depict how classes are connected. Class diagrams are essential for visualizing the structure of a system, defining the relationships between classes, and ensuring that the system's design is cohesive and scalable.

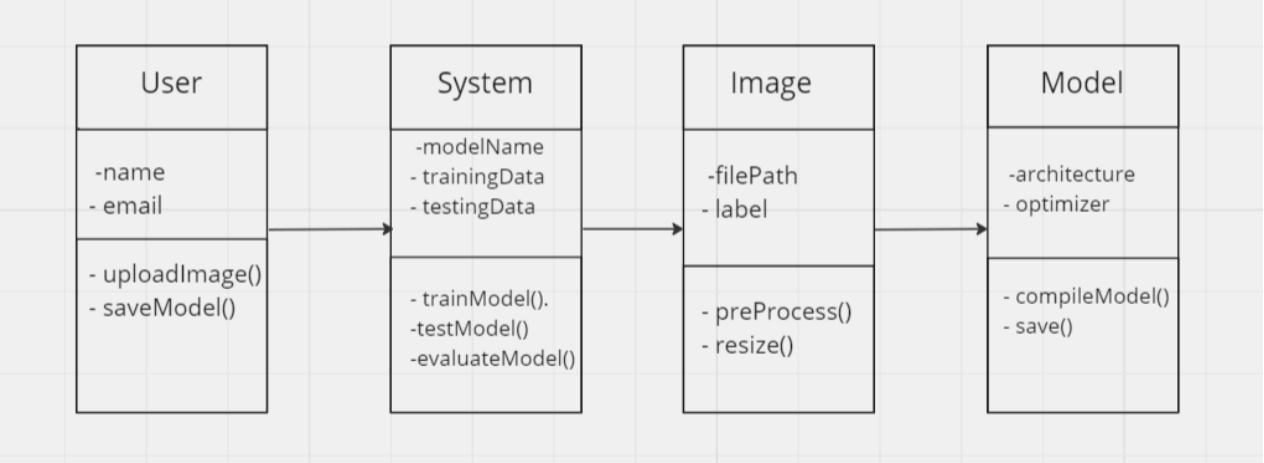


**fig 4.4.3: Class Diagram**

### 4.4.4 Sequence Diagram

Sequence diagrams are interaction diagrams that depict the interactions between objects in a particular scenario of a use case or operation. They show the sequence of messages exchanged between objects to accomplish a specific task. Each object is represented by a vertical lifeline, and

messages are represented by arrows between lifelines. Sequence diagrams help in understanding the flow of control and communication between objects in a system. They are useful for visualizing the dynamic behavior of a system and identifying potential issues in the interaction between objects.

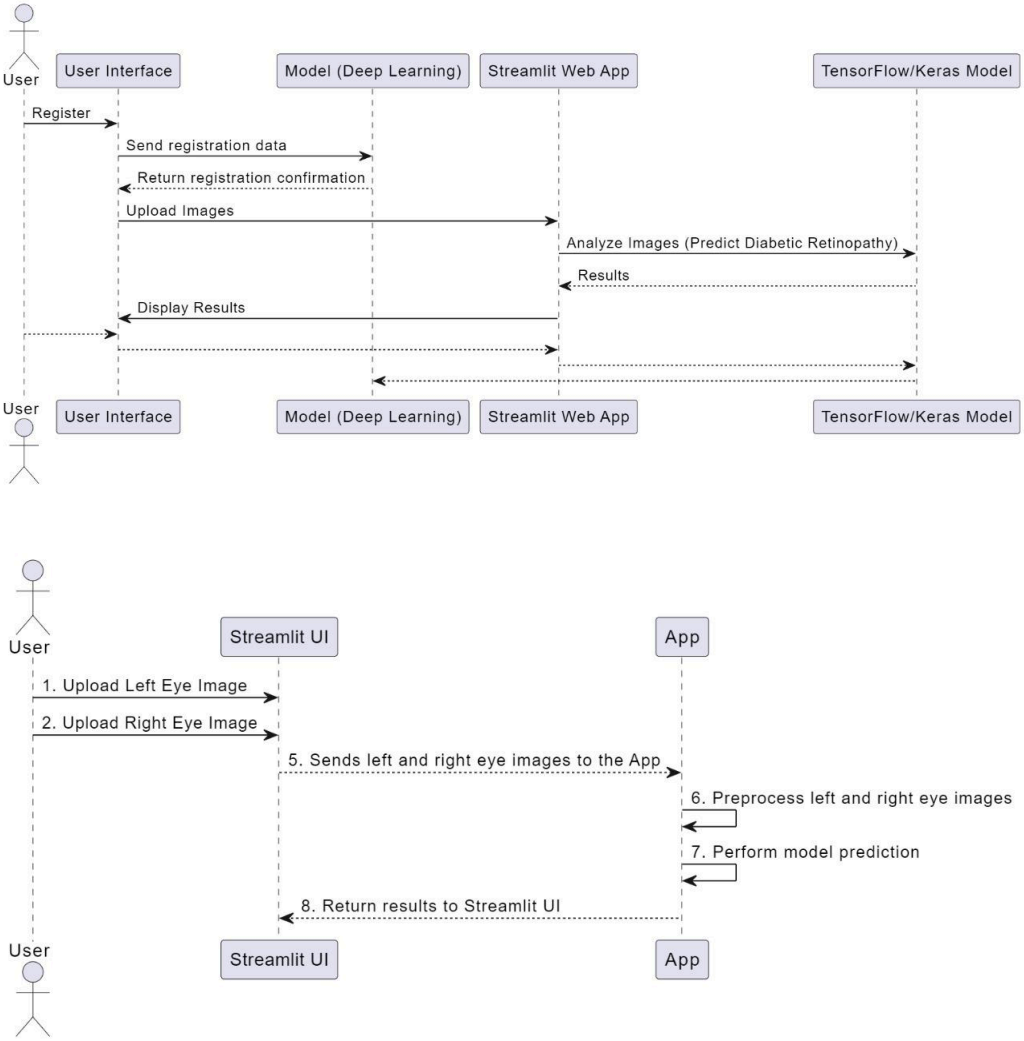


**fig 4.4.4: Sequence Diagram**

### 4.4.5 Interaction Diagrams

Interaction diagrams, including sequence diagrams and communication diagrams, are used to visualize the interactions between objects in a system. Communication diagrams focus on the structural organization of objects and the messages exchanged between them, while sequence diagrams emphasize the sequence of messages and the temporal ordering of interactions. Interaction diagrams are valuable for understanding how objects collaborate to achieve specific tasks and for identifying the dynamic behavior of a system. They help in designing and analyzing system interactions, ensuring that the system functions correctly and efficiently.


**fig 4.4.5: Interaction Diagram**

# 5 . System Design

## 5.1 Proposed System Architecture

### 5.1.1 Architecture Diagram Of Diabetic Retinopathy Detection CNN

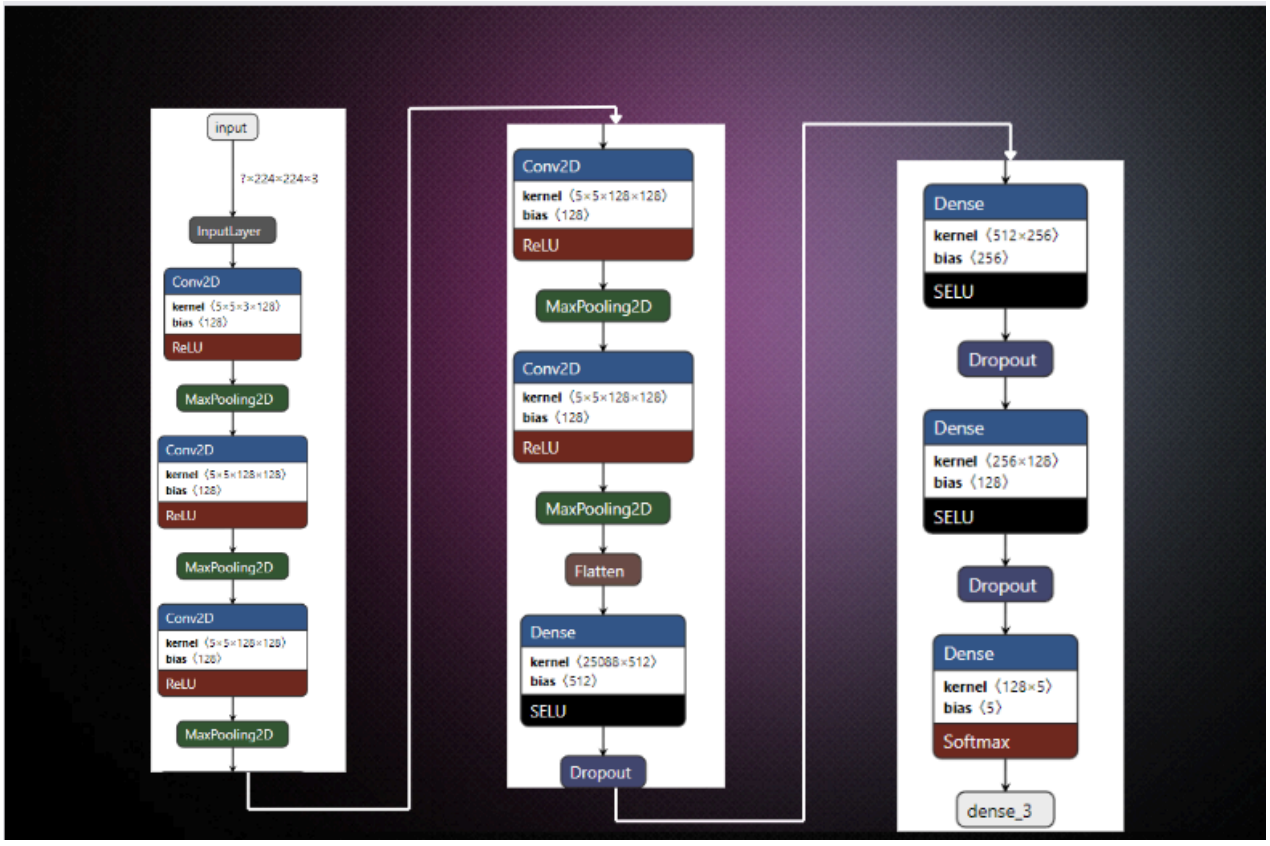


**fig 5.1.1: Architecture diagram of diabetic retinopathy detection using CNN**

The proposed system architecture leverages a Convolutional Neural Network (CNN) for Diabetic Retinopathy Detection. The input layer processes RGB images of retinas with a resolution of 224x224 pixels. The architecture consists of four sets of convolutional layers, each followed by max-pooling, to capture hierarchical features. The flattened layer is then connected to densely connected layers with dropout for feature extraction and classification. The output layer, utilizing softmax activation, facilitates multi-class classification into No DR, Mild, Moderate, Severe, and Proliferative DR.

## 5.2 Model Training

### 5.2.1 Data Preprocessing

Prior to training, images undergo preprocessing steps. A Gaussian filter is applied to enhance features, contributing to the robustness of the model. Images are resized to a uniform 224x224 resolution, ensuring consistency in input dimensions.

### 5.2.2 Data Splitting

The dataset is split into training and testing sets with an 80-20 ratio. This division ensures that the model is trained on a substantial portion of the data while retaining an independent subset for evaluation.

### 5.2.3 Model Compilation

The CNN model is compiled using the Adam optimizer with a learning rate of 1e-4. Sparse Categorical Crossentropy is chosen as the loss function for multi-class classification, and the model is configured to monitor accuracy as a metric.

### 5.2.4 Model Training

Training involves K-fold cross-validation with early stopping to prevent overfitting. The model is trained on the training set, and validation is performed on a separate validation set, ensuring the generalization of learned features.

## 5.3 Model Evaluation

### 5.3.1 Test Data

Model evaluation is conducted on a dedicated test dataset, distinct from the training and validation sets. This ensures unbiased assessment of the model's performance on unseen data.

### 5.3.2 Metrics

Evaluation metrics such as accuracy and loss are employed to quantify the performance of the trained model. These metrics provide insights into the model's ability to correctly classify retinopathy severity levels.

## 5.4 Model Deployment

### 5.4.1 Model Saving

Upon successful training, the model is saved in an H5 file format, preserving the learned weights and architecture. This facilitates easy retrieval and reuse of the trained model for future predictions.

### 5.4.2 Web Application

To provide an interactive interface for end-users, a Streamlit web application is developed. This application allows users to interact with the trained model and submit retinal images for prediction.

### 5.4.3 Image Upload

The web application includes functionality for users to upload left and right eye images for prediction. This user-friendly feature enhances accessibility and encourages user engagement.

### 5.4.4 Inference

Upon image upload, the model performs inference using the pre-trained weights. The inference process involves passing the uploaded images through the CNN architecture to generate predictions for each eye.

### 5.4.5 Result Display

The results of the Diabetic Retinopathy prediction for the left and right eyes are dynamically displayed on the web interface. This ensures transparency and allows users to interpret the model's predictions effectively.

## 5.5 Conclusion

The system design encompasses the architecture, training, evaluation, and deployment of a CNN for Diabetic Retinopathy Detection. The use of Streamlit enhances user interaction, making the model accessible to a broader audience for real-world applications.

# 6. TOOLS AND TECHNOLOGIES USED

## 6.1 Development

**TensorFlow (6.1.1)** Employed for building and training the Convolutional Neural Network (CNN), offering a comprehensive framework for machine learning tasks. Its versatility and scalability enhance the efficiency of model development.

**Pandas (6.1.2)** Played a pivotal role in data manipulation and analysis. Its DataFrame structure facilitated efficient handling of datasets, simplifying tasks such as filtering, sorting, and summarizing data.

**NumPy (6.1.3**) Utilized for numerical operations, providing support for mathematical functions and operations on arrays. Its high-performance capabilities enhance the computational efficiency of the back end.

**OpenCV (6.1.4)** Applied for image processing tasks, including reading, transforming, and analyzing images. Its robust set of functionalities contributed to the enhancement of image-related operations.

## 6.2 Machine Learning

**TensorFlow (6.2.1)** Served as the primary machine learning framework, facilitating the construction and training of the Convolutional Neural Network (CNN) model. Its deep learning capabilities empowered the back-end for advanced machine learning tasks.

## 6.3 Data Processing and Analysis

**Pandas (6.3.1)** Acted as a fundamental tool for data processing and analysis. Its capabilities in handling missing data, merging datasets, and performing statistical operations contributed to effective data management.

## 6.4 Web Development

**Streamlit (6.4.1)** Employed as a key tool for web development, enabling the creation of interactive and user-friendly web interfaces for the project. Its simplicity and rapid prototyping capabilities facilitated quick development.

**HTML/CSS/JavaScript (6.4.2)** Used collectively for web page development, providing the foundational structure, styling, and dynamic behavior needed for an engaging and responsive user experience.

### 6.5 Version Control

**Git (6.5.1)** Utilized for version control, enabling the tracking of changes in the source code. Git's branching and merging features supported collaborative development and maintained a history of project modifications.

### 6.6 Data Visualization

**Matplotlib (6.6.1)** Played a crucial role in data visualization, offering a comprehensive library for creating various plots and charts. Its customization options enhanced the presentation of data insights.

### 6.7 Other Libraries

**tqdm (6.7.1)** Applied for creating progress bars, providing visual feedback on the progress of lengthy tasks. This enhanced the user experience by conveying real-time updates during data processing.

### scikit-learn (6.7.2)

Utilized for machine learning tasks such as preprocessing, model selection, and metrics. Its comprehensive set of tools contributed to the overall effectiveness of the machine learning pipeline.

# 7. Code

## 7.1: Obtaining Kaggle Dataset

1. Download Dataset from Kaggle

```
[ ]  1 !mkdir ~/.kaggle
     2 !cp kaggle.json ~/.kaggle
     3 !chmod 600 ~/.kaggle/kaggle.json
```

```
[ ]  1 !kaggle competitions download -c aptos2019-blindness-detection
```

```
Downloading aptos2019-blindness-detection.zip to /content
100% 9.51G/9.51G [01:35<00:00, 38.0MB/s]
100% 9.51G/9.51G [01:35<00:00, 107MB/s]
```

```
     1 !unzip '/content/aptos2019-blindness-detection.zip'
```

```
Streaming output truncated to the last 5000 lines.
  inflating: test_images/512beaada8f0.png
  inflating: test_images/51323e9d2070.png
  inflating: test_images/515675001b9e.png
  inflating: test_images/515877da2def.png
  inflating: test_images/5168ddccea88.png
  inflating: test_images/5172c056a687.png
  inflating: test_images/5177461f5339.png
  inflating: test_images/51910bcb980d.png
  inflating: test_images/51a3e9b6a1c0.png
  inflating: test_images/51bbd82feec8.png
  inflating: test_images/51de5fb8eccd.png
  inflating: test_images/51e88694c81f.png
```

## 7.2: Importing the necessary libraries

2. Import Libraries

```
[ ]  1 import os
     2 import shutil
     3 import json
     4 import cv2
     5 import tqdm
     6 import numpy as np
     7 import pandas as pd
     8 import matplotlib.pyplot as plt
     9
    10 import tensorflow as tf
    11
    12 from sklearn import (
    13     preprocessing,
    14     model_selection,
    15     metrics,
    16 )
```

```
     1 SEED = 42
     2 np.random.seed = SEED
     3 tf.random.set_seed = SEED
```

## 7.3: Processing of the dataset

### 3. Data Processing

```
train_dataset = pd.read_csv('/content/train.csv')
train_dataset.head()
```

| | id_code | diagnosis |
|---|---|---|
| 0 | 000c1434d8d7 | 2 |
| 1 | 001639a390f0 | 4 |
| 2 | 0024cdab0c1e | 1 |
| 3 | 002c21358ce6 | 0 |
| 4 | 005b95c28852 | 0 |

diagnosis

```
# @title diagnosis

from matplotlib import pyplot as plt
train_dataset['diagnosis'].plot(kind='line', figsize=(8, 4), title='diagnosis')
plt.gca().spines[['top', 'right']].set_visible(False)
```

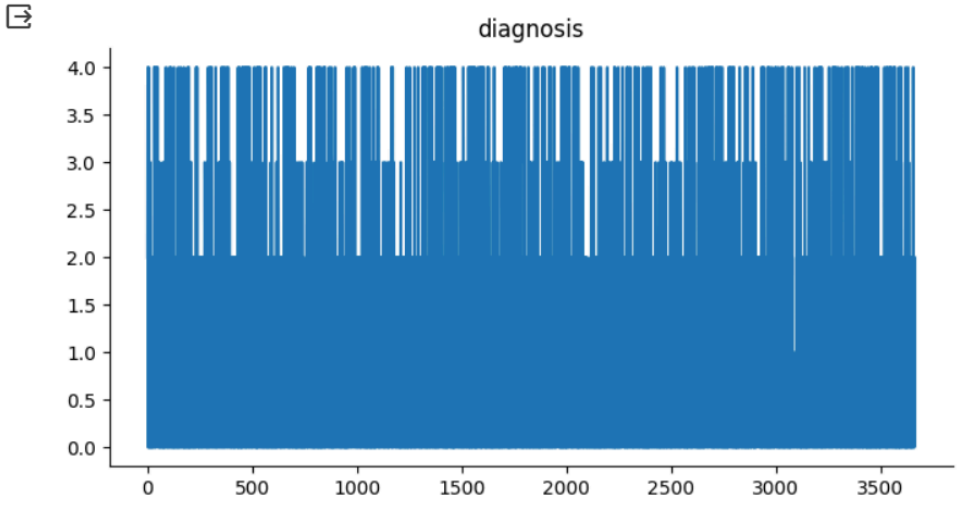


**fig 7.3.1: Diagnosis (i)**

diagnosis

```
# @title diagnosis

from matplotlib import pyplot as plt
train_dataset['diagnosis'].plot(kind='hist', bins=20, title='diagnosis')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

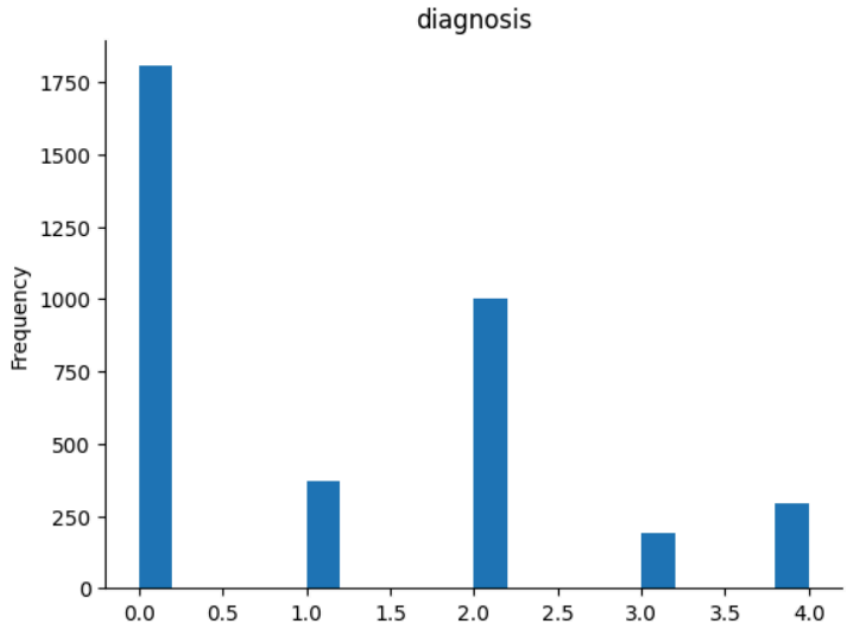


**fig 7.3.2: Diagnosis (ii)**

```
[ ]  1 classes = ['No DR','Mild','Moderate','Severe','Proliferative DR']
```

## 3.2 make dataframe for image paths & labels

```
 1 PATH = '/content/aptos2019-blindness-detection/train_images/'
 2
 3 file_paths = []
 4 labels = []
 5
 6 for i in range(0,len(train_dataset)):
 7     file_paths.append(PATH + train_dataset.id_code[i] + '.png')
 8     labels.append(train_dataset.diagnosis[i])
 9
10 path_series = pd.Series(file_paths, name='filepath')
11 label_series = pd.Series(labels, name='label')
12 training_data = pd.concat([path_series, label_series], axis=1)
13 training_data.head()
```

| | filepath | label |
|---|---|---|
| 0 | /content/aptos2019-blindness-detection/train_i... | 2 |
| 1 | /content/aptos2019-blindness-detection/train_i... | 4 |
| 2 | /content/aptos2019-blindness-detection/train_i... | 1 |
| 3 | /content/aptos2019-blindness-detection/train_i... | 0 |
| 4 | /content/aptos2019-blindness-detection/train_i... | 0 |

## 3.3 make folder to hold the filtered images

```
dest_path = '/content/'

os.mkdir(dest_path + 'filtered_train_images')
```

```
train_source_path = '/content/train_images/'
train_des_path = '/content/filtered_train_images/'

for img in tqdm.tqdm(os.listdir(train_source_path)):
    shutil.copy(train_source_path + img,train_des_path)
```

```
100%|██████████| 3662/3662 [01:19<00:00, 46.18it/s]
```

### 3.4 apply filter for images

```
def gaussian_filter(path):
    sigmaX = 15
    img = cv2.imread(path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    gaussian = cv2.addWeighted(img, 4, cv2.GaussianBlur(img, (0,0), sigmaX), -4, 128)
    gaussian = cv2.resize(gaussian, (224, 224))
    cv2.imwrite(path,gaussian)
```

```
for img in tqdm.tqdm(os.listdir(train_des_path)):
    gaussian_filter(train_des_path + img)
```

```
100%|██████████| 3662/3662 [15:50<00:00,  3.85it/s]
```

```
PATH = '/content/filtered_train_images/'

file_paths = []
labels = []

for i in range(0,len(train_dataset)):
    file_paths.append(PATH + train_dataset.id_code[i] + '.png')
    labels.append(train_dataset.diagnosis[i])

path_series = pd.Series(file_paths, name='filepath')
label_series = pd.Series(labels, name='label')
training_data = pd.concat([path_series, label_series], axis=1)
training_data.head()
```

| | filepath | label |
|---|---|---|
| 0 | /content/filtered_train_images/000c1434d8d7.png | 2 |
| 1 | /content/filtered_train_images/001639a390f0.png | 4 |
| 2 | /content/filtered_train_images/0024cdab0c1e.png | 1 |
| 3 | /content/filtered_train_images/002c21358ce6.png | 0 |
| 4 | /content/filtered_train_images/005b95c28852.png | 0 |

### 3.5 Display 25 sample of filtered images

```
plt.figure(figsize=(20, 20))
for idx in range(0,25):
    image = cv2.imread(training_data.filepath[idx])
    plt.subplot(5,5,idx+1)
    plt.imshow(image)
plt.show()
```

**Output:** 25 samples of retinal images after applying the gaussian filter.

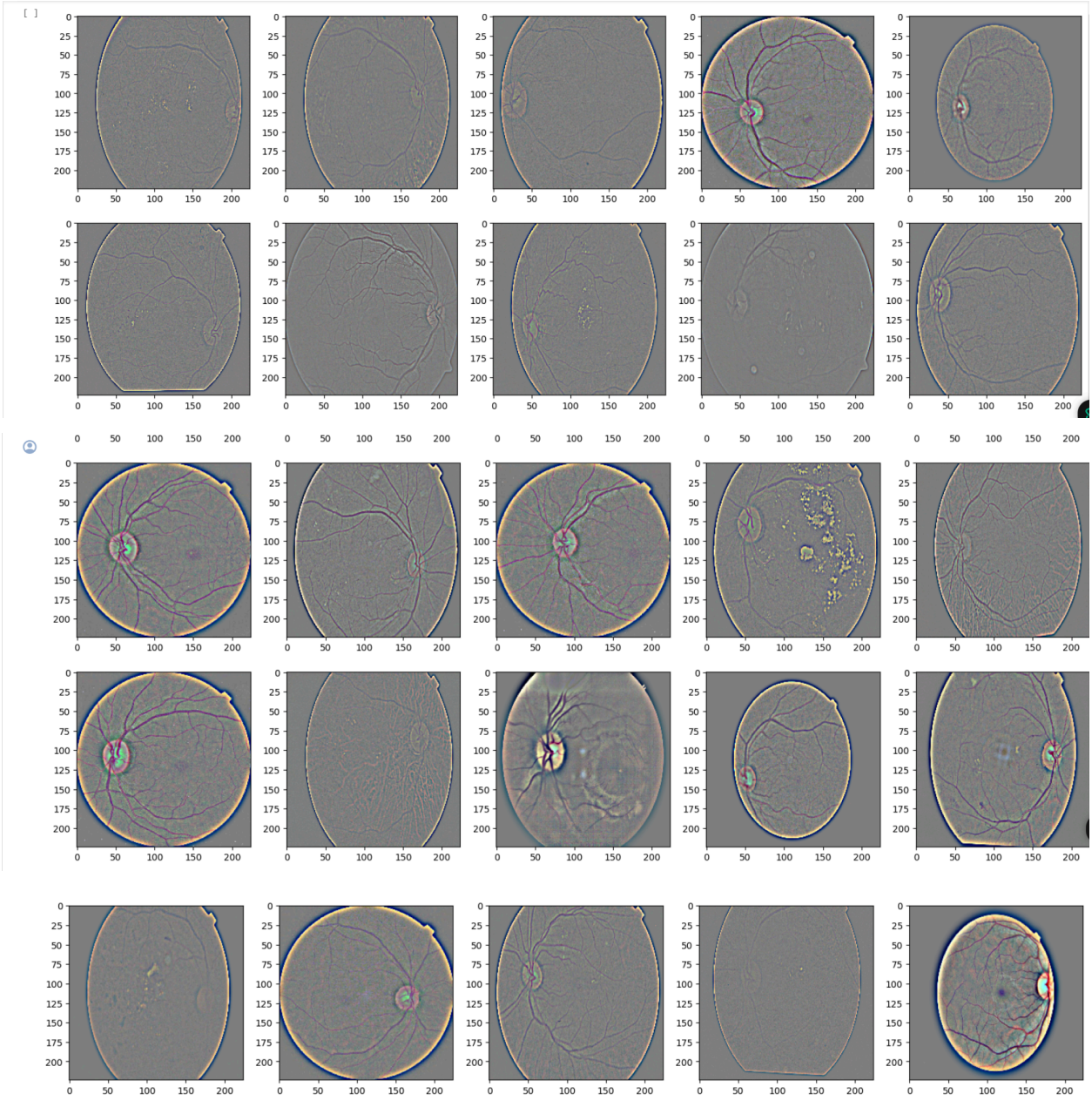

**fig 7.3.3:** 25 samples of retinal images after applying the gaussian filter.

### 3.6 Splitting Data

```
[ ]  1 mask = np.random.rand(len(training_data)) < 0.8
     2 train = training_data[mask]
     3 test = training_data[~mask]
     4
     5 print('Training Data Size :- ',train.shape)
     6 print('Testing Data Size :- ',test.shape)
```

```
Training Data Size :-  (2953, 2)
Testing Data Size :-  (709, 2)
```

```
[ ]  1 x = train.filepath
     2 y = train.label
```

## 7.4: Training and Testing of our Model

### 4. Modeling

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(128, (5,5), padding="same", input_shape=(224,224,3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

    tf.keras.layers.Conv2D(128, (5,5), padding="same", activation = 'relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

    tf.keras.layers.Conv2D(128, (5,5), padding="same", activation = 'relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

    tf.keras.layers.Conv2D(128, (5,5), padding="same", activation = 'relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(512, activation="selu", kernel_initializer="lecun_normal"),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(256, activation="selu", kernel_initializer="lecun_normal"),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(128,activation="selu", kernel_initializer="lecun_normal"),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(len(classes), activation = 'softmax')
])

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4, beta_1=0.9, beta_2=0.999),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 128)     9728

 max_pooling2d (MaxPooling2  (None, 112, 112, 128)     0
 D)

 conv2d_1 (Conv2D)           (None, 112, 112, 128)     409728

 max_pooling2d_1 (MaxPoolin  (None, 56, 56, 128)       0
 g2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 128)       409728

 max_pooling2d_2 (MaxPoolin  (None, 28, 28, 128)       0
 g2D)

 conv2d_3 (Conv2D)           (None, 28, 28, 128)       409728

 max_pooling2d_3 (MaxPoolin  (None, 14, 14, 128)       0
 g2D)

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 512)               12845568

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dropout_2 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 5)                 645

=================================================================
Total params: 14249349 (54.36 MB)
Trainable params: 14249349 (54.36 MB)
Non-trainable params: 0 (0.00 Byte)
```

```
path = '/content/train.csv'

train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

validation_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale = 1./255)


kfold = model_selection.StratifiedKFold(n_splits=5,shuffle=True,random_state=42)

# Variable for keeping count of split we are executing
j = 0

model_history = []

# K-fold Train and test for each split
for train_idx, val_idx in tqdm.tqdm(list(kfold.split(x,y))):
    x_train_df = training_data.iloc[train_idx]
    x_valid_df = training_data.iloc[val_idx]

    j+=1

    training_set = train_datagen.flow_from_dataframe(dataframe=x_train_df, directory=path,
                                                     x_col="filepath", y_col="label",
                                                     class_mode="raw",
                                                     target_size=(224,224), batch_size=32)

    validation_set = validation_datagen.flow_from_dataframe(dataframe=x_valid_df, directory=path,
                                                     x_col="filepath", y_col="label",
                                                     class_mode="raw",
                                                     target_size=(224,224), batch_size=32)

    early_stopping = tf.keras.callbacks.EarlyStopping(patience=5, monitor='val_accuracy', mode='max', restore_best_weights=True)

    history = model.fit(training_set,validation_data=validation_set,epochs = 10,callbacks=[early_stopping])

    model_history.append(history)
```

```
  0%|          | 0/5 [00:00<?, ?it/s]Found 2360 validated image filenames.
Found 591 validated image filenames.
Epoch 1/10
74/74 [==============================] - 61s 612ms/step - loss: 1.1657 - accuracy: 0.5436 - val_loss: 0.9831 - val_accuracy: 0.6345
Epoch 2/10
74/74 [==============================] - 36s 489ms/step - loss: 0.9887 - accuracy: 0.6640 - val_loss: 0.9901 - val_accuracy: 0.6548
Epoch 3/10
74/74 [==============================] - 39s 521ms/step - loss: 0.9247 - accuracy: 0.6767 - val_loss: 0.9261 - val_accuracy: 0.6667
Epoch 4/10
74/74 [==============================] - 36s 490ms/step - loss: 0.8948 - accuracy: 0.6907 - val_loss: 0.9360 - val_accuracy: 0.6785
Epoch 5/10
74/74 [==============================] - 36s 491ms/step - loss: 0.8808 - accuracy: 0.6928 - val_loss: 0.9278 - val_accuracy: 0.6582
Epoch 6/10
74/74 [==============================] - 37s 503ms/step - loss: 0.8563 - accuracy: 0.7068 - val_loss: 0.9191 - val_accuracy: 0.6802
Epoch 7/10
74/74 [==============================] - 36s 491ms/step - loss: 0.8458 - accuracy: 0.7000 - val_loss: 0.8853 - val_accuracy: 0.6701
Epoch 8/10
74/74 [==============================] - 36s 484ms/step - loss: 0.8258 - accuracy: 0.7013 - val_loss: 0.8630 - val_accuracy: 0.6667
Epoch 9/10
74/74 [==============================] - 36s 490ms/step - loss: 0.8163 - accuracy: 0.7186 - val_loss: 0.8965 - val_accuracy: 0.6836
Epoch 10/10
74/74 [==============================] - 36s 489ms/step - loss: 0.8077 - accuracy: 0.7068 - val_loss: 0.8435 - val_accuracy: 0.6751

 20%|██        | 1/5 [07:07<28:31, 427.80s/it]Found 2361 validated image filenames.
Found 590 validated image filenames.
Epoch 1/10
74/74 [==============================] - 46s 626ms/step - loss: 0.8142 - accuracy: 0.7031 - val_loss: 0.7313 - val_accuracy: 0.7390
Epoch 2/10
74/74 [==============================] - 37s 496ms/step - loss: 0.8225 - accuracy: 0.7145 - val_loss: 0.7256 - val_accuracy: 0.7271
Epoch 3/10
74/74 [==============================] - 37s 501ms/step - loss: 0.8226 - accuracy: 0.6997 - val_loss: 0.7443 - val_accuracy: 0.7305
Epoch 4/10
74/74 [==============================] - 36s 486ms/step - loss: 0.7867 - accuracy: 0.7179 - val_loss: 0.7376 - val_accuracy: 0.7254
Epoch 5/10
74/74 [==============================] - 36s 486ms/step - loss: 0.7855 - accuracy: 0.7154 - val_loss: 0.7299 - val_accuracy: 0.7356
Epoch 6/10
74/74 [==============================] - 38s 516ms/step - loss: 0.7721 - accuracy: 0.7264 - val_loss: 0.7533 - val_accuracy: 0.7203
```

```
 40%|████      | 2/5 [11:34<16:39, 333.20s/it]Found 2361 validated image filenames.
Found 590 validated image filenames.
Epoch 1/10
74/74 [==============================] - 36s 490ms/step - loss: 0.8088 - accuracy: 0.6997 - val_loss: 0.7079 - val_accuracy: 0.7559
Epoch 2/10
74/74 [==============================] - 37s 500ms/step - loss: 0.8051 - accuracy: 0.7107 - val_loss: 0.7167 - val_accuracy: 0.7339
Epoch 3/10
74/74 [==============================] - 36s 485ms/step - loss: 0.8022 - accuracy: 0.7099 - val_loss: 0.7129 - val_accuracy: 0.7542
Epoch 4/10
74/74 [==============================] - 36s 489ms/step - loss: 0.7921 - accuracy: 0.7137 - val_loss: 0.7393 - val_accuracy: 0.7288
Epoch 5/10
74/74 [==============================] - 38s 513ms/step - loss: 0.7748 - accuracy: 0.7205 - val_loss: 0.7277 - val_accuracy: 0.7542
Epoch 6/10
74/74 [==============================] - 36s 488ms/step - loss: 0.7780 - accuracy: 0.7145 - val_loss: 0.7047 - val_accuracy: 0.7644
Epoch 7/10
74/74 [==============================] - 38s 505ms/step - loss: 0.7588 - accuracy: 0.7196 - val_loss: 0.7605 - val_accuracy: 0.7576
Epoch 8/10
74/74 [==============================] - 36s 487ms/step - loss: 0.7352 - accuracy: 0.7327 - val_loss: 0.7281 - val_accuracy: 0.7661
Epoch 9/10
74/74 [==============================] - 38s 515ms/step - loss: 0.7404 - accuracy: 0.7238 - val_loss: 0.6950 - val_accuracy: 0.7593
Epoch 10/10
74/74 [==============================] - 37s 495ms/step - loss: 0.7313 - accuracy: 0.7243 - val_loss: 0.7021 - val_accuracy: 0.7390
 60%|██████    | 3/5 [18:05<11:58, 359.35s/it]Found 2361 validated image filenames.
Found 590 validated image filenames.
Epoch 1/10
74/74 [==============================] - 38s 511ms/step - loss: 0.7160 - accuracy: 0.7310 - val_loss: 0.6920 - val_accuracy: 0.7508
Epoch 2/10
74/74 [==============================] - 37s 496ms/step - loss: 0.7211 - accuracy: 0.7323 - val_loss: 0.6902 - val_accuracy: 0.7475
Epoch 3/10
74/74 [==============================] - 36s 491ms/step - loss: 0.7079 - accuracy: 0.7298 - val_loss: 0.7013 - val_accuracy: 0.7576
Epoch 4/10
74/74 [==============================] - 37s 496ms/step - loss: 0.7032 - accuracy: 0.7357 - val_loss: 0.6885 - val_accuracy: 0.7424
Epoch 5/10
74/74 [==============================] - 37s 498ms/step - loss: 0.6873 - accuracy: 0.7471 - val_loss: 0.7278 - val_accuracy: 0.7475
Epoch 6/10
74/74 [==============================] - 37s 499ms/step - loss: 0.6799 - accuracy: 0.7501 - val_loss: 0.7541 - val_accuracy: 0.7458
Epoch 7/10
74/74 [==============================] - 37s 496ms/step - loss: 0.6667 - accuracy: 0.7484 - val_loss: 0.6900 - val_accuracy: 0.7508
Epoch 8/10
74/74 [==============================] - 38s 512ms/step - loss: 0.6809 - accuracy: 0.7446 - val_loss: 0.7358 - val_accuracy: 0.7525
 80%|████████  | 4/5 [23:13<05:39, 339.37s/it]Found 2361 validated image filenames.
Found 590 validated image filenames.
Epoch 1/10
74/74 [==============================] - 37s 501ms/step - loss: 0.7295 - accuracy: 0.7277 - val_loss: 0.5754 - val_accuracy: 0.7932
Epoch 2/10
74/74 [==============================] - 37s 500ms/step - loss: 0.7119 - accuracy: 0.7382 - val_loss: 0.5782 - val_accuracy: 0.8017
Epoch 3/10
74/74 [==============================] - 38s 512ms/step - loss: 0.7091 - accuracy: 0.7421 - val_loss: 0.5584 - val_accuracy: 0.7983
Epoch 4/10
74/74 [==============================] - 37s 494ms/step - loss: 0.6910 - accuracy: 0.7438 - val_loss: 0.5819 - val_accuracy: 0.7831
Epoch 5/10
74/74 [==============================] - 37s 494ms/step - loss: 0.6783 - accuracy: 0.7501 - val_loss: 0.5543 - val_accuracy: 0.8000
Epoch 6/10
74/74 [==============================] - 37s 495ms/step - loss: 0.6765 - accuracy: 0.7476 - val_loss: 0.5824 - val_accuracy: 0.7847
Epoch 7/10
74/74 [==============================] - 36s 492ms/step - loss: 0.6648 - accuracy: 0.7586 - val_loss: 0.5578 - val_accuracy: 0.7932
100%|██████████| 5/5 [27:40<00:00, 332.04s/it]
```

## 4.1. Model Activity Chart

```
acc,val_acc,loss,val_loss = [],[],[],[]
for idx in range(0,len(model_history)):
    acc.extend(model_history[idx].history['accuracy'])
    val_acc.extend(model_history[idx].history['val_accuracy'])

    loss.extend(model_history[idx].history['loss'])
    val_loss.extend(model_history[idx].history['val_loss'])
```

```
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, color='red', label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

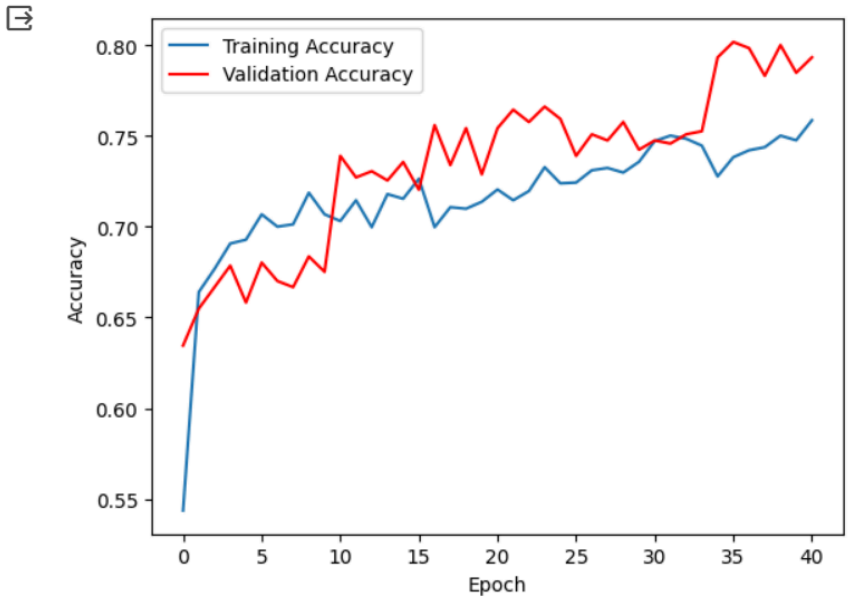


**fig 7.4.1: Accuracy Graph**

```
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, color='red', label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

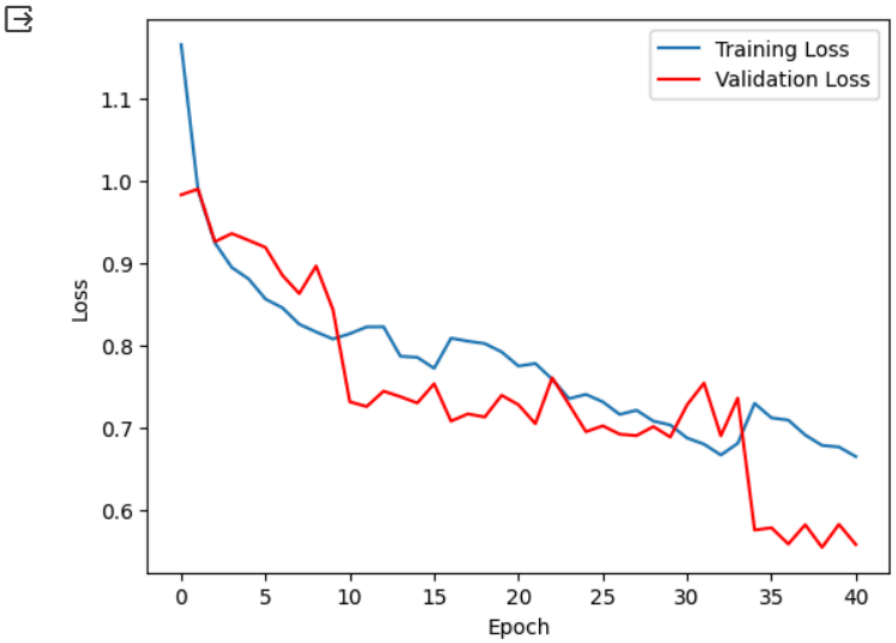


**fig 7.4.2: Loss Graph**

## 4.2.Save Progress

```
model.save("/content/diabetic_retina_model.h5")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered leg
  saving_api.save_model(
```

```
model_infos = {
  "loss": loss,
  "accuracy": acc,
  "val_loss": val_loss,
  "val_acc" : val_acc
}

with open("/content/model_history.json", "w") as outfile:
    json.dump(model_infos, outfile)
```

## 4.3.Model Evaluation

```
test_path = "/content/train.csv"

test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale = 1./255)

test_set = test_datagen.flow_from_dataframe(dataframe=test, directory=path,
                                            x_col="filepath", y_col="label",
                                            class_mode="raw",
                                            target_size=(224,224), batch_size=32)
```

```
Found 717 validated image filenames.
```

```
model.evaluate(test_set)
```

```
23/23 [==============================] - 229s 10s/step - loss: 1.2262 - accuracy: 0.5509
[1.226181983947754, 0.5509065389633179]
```

```
[ ]
# After processing the files and before the end of your script
import shutil

# Mount Google Drive to Colab
from google.colab import drive
drive.mount('/content/drive')

# Example destination folder in Google Drive
dest_folder = '/content/drive/My Drive/dataset/'
```

```
Mounted at /content/drive
```

## 7.5: Hosting the website on Streamlit

5.Host the website

```
import tensorflow as tf
import cv2
import numpy as np

# Load the trained model
model = tf.keras.models.load_model('/content/diabetic_retina_model.h5')

# Function to preprocess the uploaded image
def preprocess_image(image):
    image = cv2.resize(image, (224, 224))
    # Add any other preprocessing steps here
    return image
```

```
%%writefile app.py
import streamlit as st

st.title('Deep Diabetic Retinopathy Detection')

st.sidebar.title('Upload Images')
left_image = st.sidebar.file_uploader('Upload Left Eye Image', type=['jpg', 'png'])
right_image = st.sidebar.file_uploader('Upload Right Eye Image', type=['jpg', 'png'])

if left_image and right_image:
    left_image_data = preprocess_image(cv2.imread(left_image))
    right_image_data = preprocess_image(cv2.imread(right_image))

    # Perform inference using the model
    left_eye_result = model.predict(np.expand_dims(left_image_data, axis=0))
    right_eye_result = model.predict(np.expand_dims(right_image_data, axis=0))

    # Display results
    st.write('### Left Eye Result:', left_eye_result)
    st.write('### Right Eye Result:', right_eye_result)
```

```
Writing app.py
```

```
!pip install pyngrok --upgrade
```

```
Collecting pyngrok
  Downloading pyngrok-7.1.6-py3-none-any.whl (22 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.10/dist-packages (from pyngrok) (6.0.1)
Installing collected packages: pyngrok
Successfully installed pyngrok-7.1.6
```

```
!pip install streamlit
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.1/8.1 MB 9.2 MB/s eta 0:00:00
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.2.2)
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/lib/python3/dist-packages (from streamlit) (1.4)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (5.3.3)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (8.1.7)
Requirement already satisfied: numpy<2,>=1.19.3 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.25.2)
Collecting packaging<24,>=16.8 (from streamlit)
  Downloading packaging-23.2-py3-none-any.whl (53 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.0/53.0 kB 5.2 MB/s eta 0:00:00
Requirement already satisfied: pandas<3,>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.5.3)
Requirement already satisfied: pillow<11,>=7.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (9.4.0)
Requirement already satisfied: protobuf<5,>=3.20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (3.20.3)
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (14.0.2)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.10/dist-packages (from streamlit) (2.31.0)
Requirement already satisfied: rich<14,>=10.14.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (13.7.1)
Requirement already satisfied: tenacity<9,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (8.2.3)
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.10.0)
Collecting gitpython!=3.1.19,<4,>=3.0.7 (from streamlit)
  Downloading GitPython-3.1.43-py3-none-any.whl (207 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.3/207.3 kB 8.5 MB/s eta 0:00:00
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.8.1b0-py2.py3-none-any.whl (4.8 MB)
```

```
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->altair<6,>=4.0->streamlit) (2.1.5)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (23.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6,>=4.0-
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.34.
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich<14,>=10.14.0->streamlit) (0.1.2
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas<3,>=1.3.0->streamlit) (1.16.0)
Installing collected packages: watchdog, smmap, packaging, pydeck, gitdb, gitpython, streamlit
  Attempting uninstall: packaging
    Found existing installation: packaging 24.0
    Uninstalling packaging-24.0:
      Successfully uninstalled packaging-24.0
Successfully installed gitdb-4.0.11 gitpython-3.1.43 packaging-23.2 pydeck-0.8.1b0 smmap-5.0.1 streamlit-1.32.2 watchdog-4.0.0
```

```
!pip install streamlit pillow tensorflow
```

```
Requirement already satisfied: streamlit in /usr/local/lib/python3.10/dist-packages (1.32.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (9.4.0)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.2.2)
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/lib/python3/dist-packages (from streamlit) (1.4)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (5.3.3)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (8.1.7)
Requirement already satisfied: numpy<2,>=1.19.3 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.25.2)
Requirement already satisfied: packaging<24,>=16.8 in /usr/local/lib/python3.10/dist-packages (from streamlit) (23.2)
Requirement already satisfied: pandas<3,>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.5.3)
Requirement already satisfied: protobuf<5,>=3.20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (3.20.3)
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (14.0.2)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.10/dist-packages (from streamlit) (2.31.0)
Requirement already satisfied: rich<14,>=10.14.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (13.7.1)
Requirement already satisfied: tenacity<9,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (8.2.3)
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.10.0)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3.10/dist-packages (from streamlit) (3.1.43)
```

```
#Overwrite writefile app.py

import streamlit as st
import cv2
import numpy as np
import tensorflow as tf
import pandas as pd

# Load the trained model
model = tf.keras.models.load_model('/content/diabetic_retina_model.h5')

# Function to preprocess the uploaded image
def preprocess_image(image):
    image = cv2.resize(image, (224, 224))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Ensure correct color format
    image = image / 255.0  # Normalize pixel values
    return image

st.title('Deep Diabetic Retinopathy Detection')

# Adding slogans
st.write('''
Your eyes, your best tool, take care of them

Your eyes are in your hands

Love your eyes, love your future
''')

# Path to your image
image_path = '/content/retina.jpg'
```

```
# Display the image
st.image(image_path, caption='The Retina', use_column_width=True)

st.sidebar.title('Upload Images')
left_image = st.sidebar.file_uploader('Upload Left Eye Image', type=['jpg', 'png'])
right_image = st.sidebar.file_uploader('Upload Right Eye Image', type=['jpg', 'png'])

if left_image and right_image:
    left_image_data = preprocess_image(cv2.imdecode(np.fromstring(left_image.read(), np.uint8), 1))
    right_image_data = preprocess_image(cv2.imdecode(np.fromstring(right_image.read(), np.uint8), 1))

    # Perform inference using the model
    left_eye_result = model.predict(np.expand_dims(left_image_data, axis=0))
    right_eye_result = model.predict(np.expand_dims(right_image_data, axis=0))

    # Convert numeric labels to class names
    class_mapping = {0: 'No DR', 1: 'Mild', 2: 'Moderate', 3: 'Severe', 4: 'Proliferative DR'}
    left_eye_class = class_mapping[np.argmax(left_eye_result)]
    right_eye_class = class_mapping[np.argmax(right_eye_result)]

    # Display results
    st.write('### Left Eye Result:', left_eye_class)
    st.write('#### Probabilities:')
    left_prob_df = pd.DataFrame({ 'Class': class_mapping.values(), 'Probability': left_eye_result[0] })
    st.write(left_prob_df)

    st.write('### Right Eye Result:', right_eye_class)
    st.write('#### Probabilities:')
    right_prob_df = pd.DataFrame({ 'Class': class_mapping.values(), 'Probability': right_eye_result[0] })
    st.write(right_prob_df)
```

```
# Show Result button
if st.button('Get Results'):
    left_eye_disease = 'No Disease' if left_eye_class == 'No DR' else 'Diabetic Retinopathy'
    right_eye_disease = 'No Disease' if right_eye_class == 'No DR' else 'Diabetic Retinopathy'

    result_text = ""
    result_text += f"Left eye is prone to {left_eye_disease}.\n"
    result_text += f"Highest Probability for Left Eye: {left_prob_df['Probability'].max()}\n\n"

    result_text += f"Right eye is prone to {right_eye_disease}.\n"
    result_text += f"Highest Probability for Right Eye: {right_prob_df['Probability'].max()}\n\n"

    if left_eye_disease == 'No Disease' and right_eye_disease == 'No Disease':
        result_text += "Both eyes are not prone to any disease.\n"
    elif left_eye_disease == 'Diabetic Retinopathy' and right_eye_disease == 'Diabetic Retinopathy':
        result_text += "Both eyes are prone to Diabetic Retinopathy.\n"
    elif left_eye_disease == 'No Disease' and right_eye_disease == 'Diabetic Retinopathy':
        result_text += "Left eye is not prone to any disease, but right eye is prone to Diabetic Retinopathy.\n"
    elif left_eye_disease == 'Diabetic Retinopathy' and right_eye_disease == 'No Disease':
        result_text += "Right eye is not prone to any disease, but left eye is prone to Diabetic Retinopathy.\n"

    st.markdown(result_text)
```

```
[107] !streamlit run app.py &>/content/logs.txt &
```

```
[108] !curl https://loca.lt/mytunnelpassword
```

```
34.123.82.6
```

```
!npx localtunnel --port 8501
```

```
npx: installed 22 in 2.143s
your url is: https://late-tables-create.loca.lt
```

# 8. RESULTS

## Deep Diabetic Retinopathy Detection

Welcome to our Deep Diabetic Retinopathy Detection application. We utilize advanced deep learning technology to detect diabetic retinopathy from retinal images. Diabetic retinopathy is a diabetes-related eye disease that affects the retina and can lead to vision loss if left untreated. Early detection is crucial for timely intervention and prevention of vision impairment.

**STEPS:**

### Upload Images:

Use the sidebar to upload images of the left and right eye.
Click on the "Upload Left Eye Image" button to select and upload an image of the left eye.
Similarly, click on the "Upload Right Eye Image" button to upload an image of the right eye.
Ensure that the uploaded images are clear and focused for accurate analysis.

### Analysis Process:

After uploading, our model will process the images using advanced deep learning algorithms.
The model will analyze each image and provide a diagnosis based on the severity of diabetic retinopathy.
The severity levels include:
i) No DR (No Diabetic Retinopathy)
ii) Mild
iii) Moderate
iv) Severe
v) Proliferative DR

### View Results:

Once the analysis is complete, we get the option to view results. The diagnosis for each eye will be displayed on the screen.
You'll see the diagnosis indicating the severity of diabetic retinopathy detected in each eye.

This information is valuable for understanding the current state of diabetic retinopathy and guiding further treatment decisions.

Next Steps:
Based on the diagnosis provided, you can consult with an eye care professional for further evaluation and treatment, if necessary.
Regular screening and early intervention are essential for managing diabetic retinopathy and preserving vision

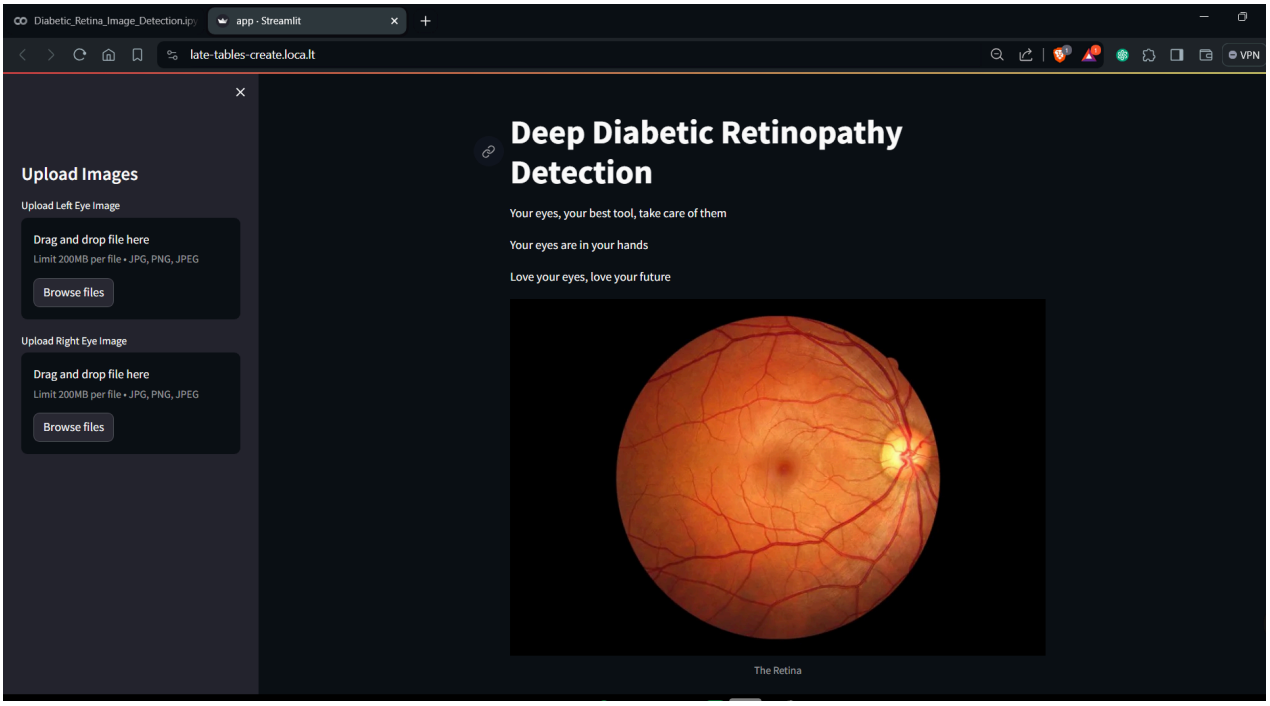


**fig 8.1: Frontend UI**

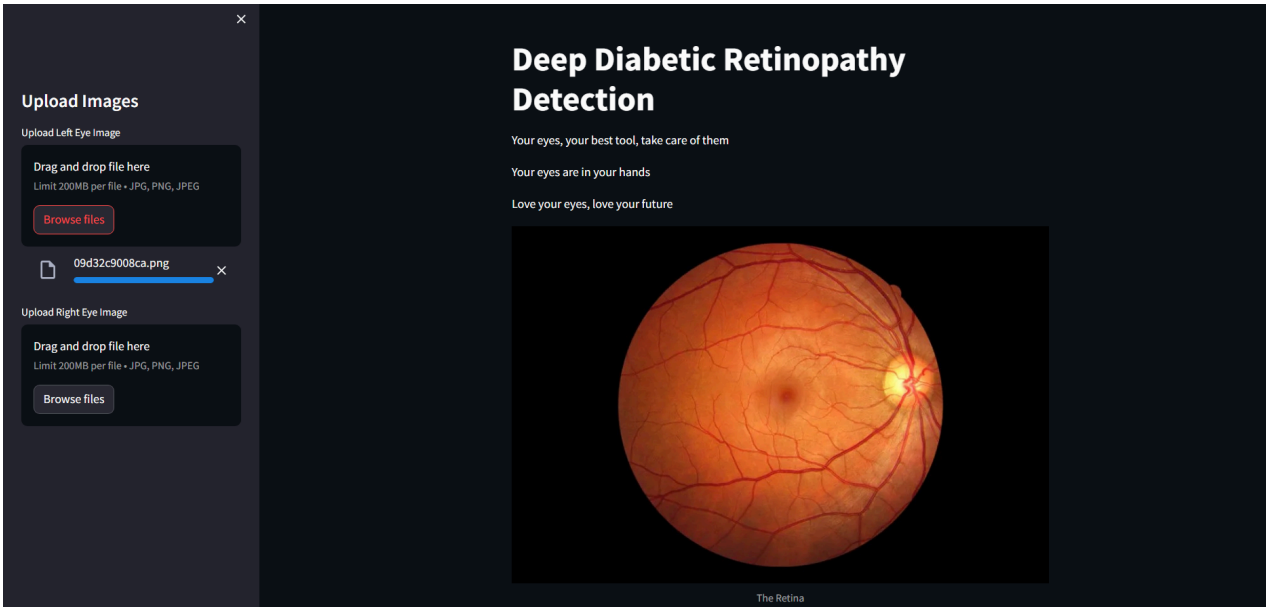


**fig 8.2: upload retinal images Left**

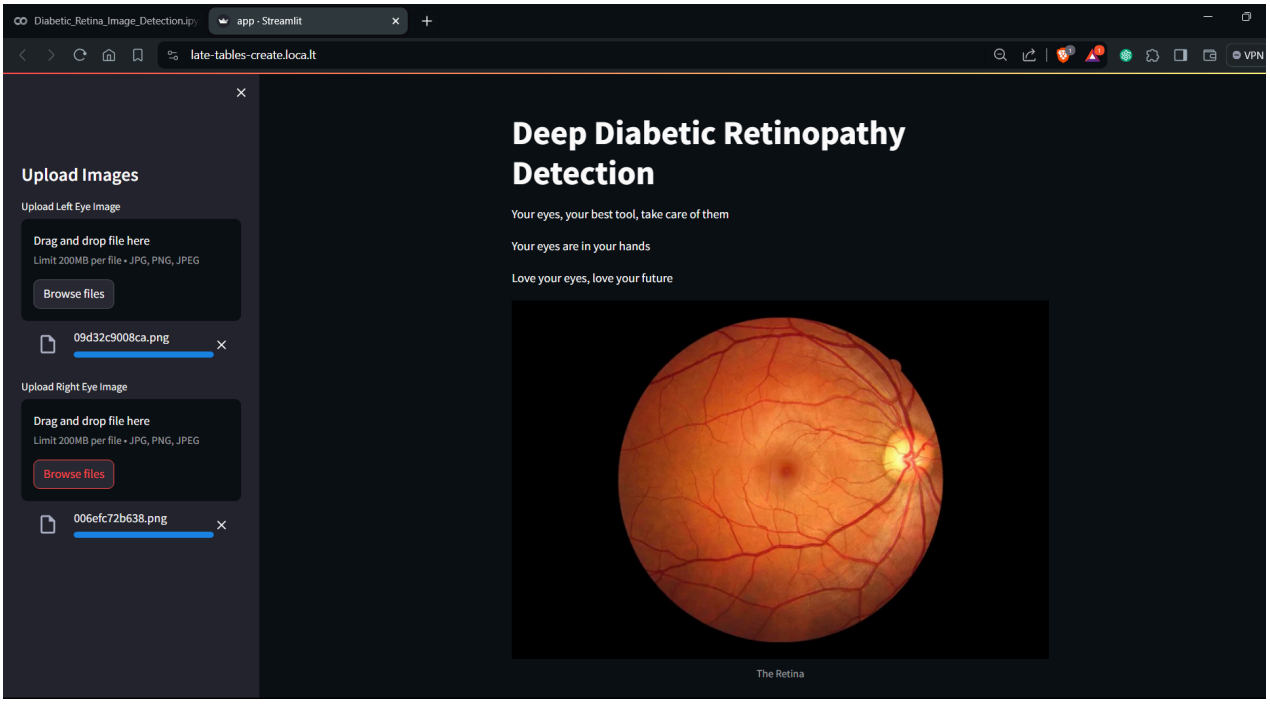


**fig 8.3: upload retinal images right**

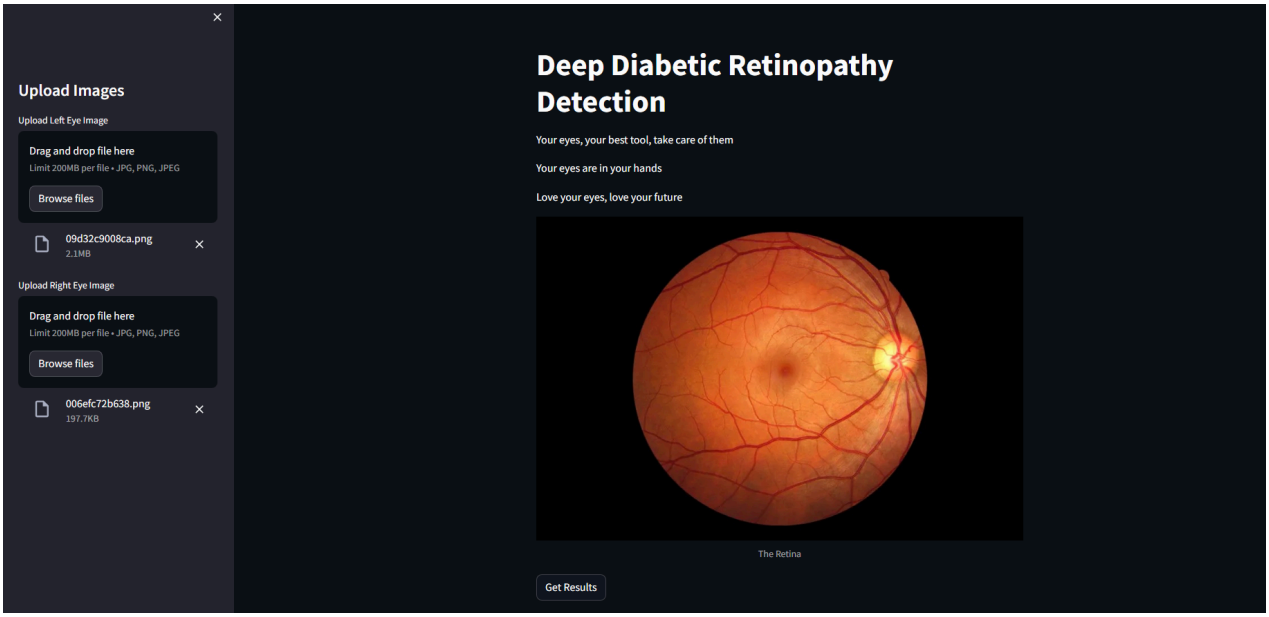


**fig 8.4: click on Get Results**

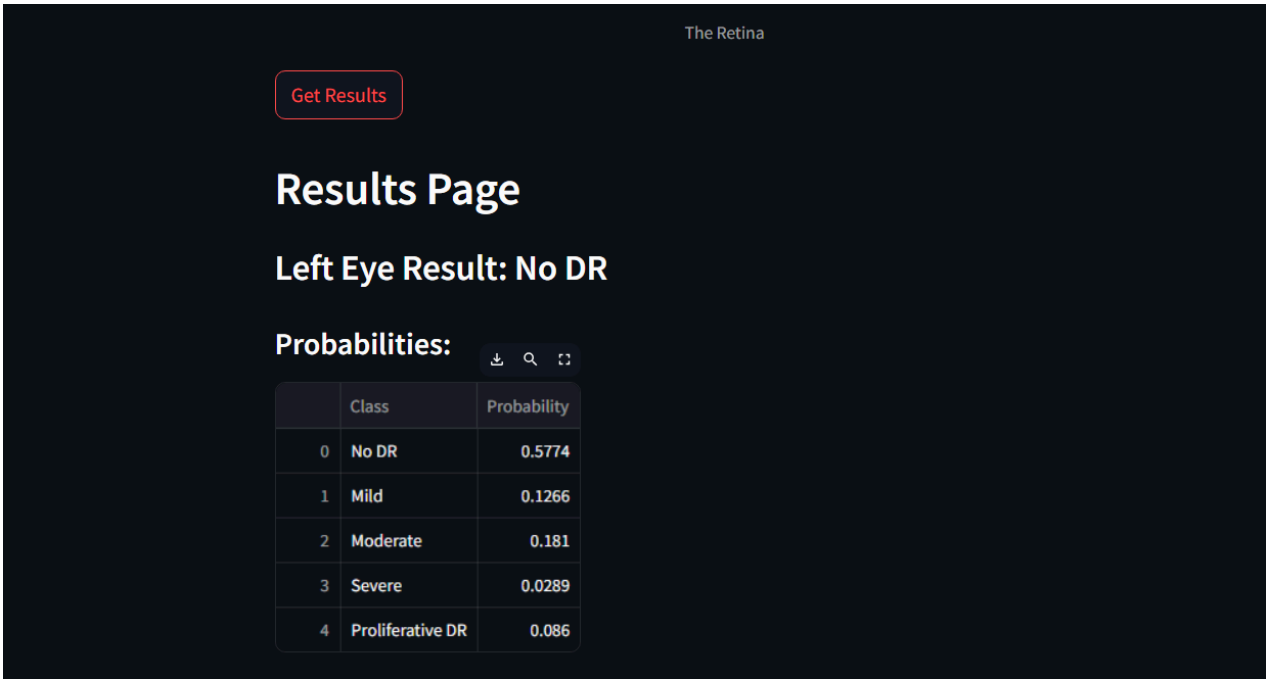


**fig 8.5: Result retinal images Left**

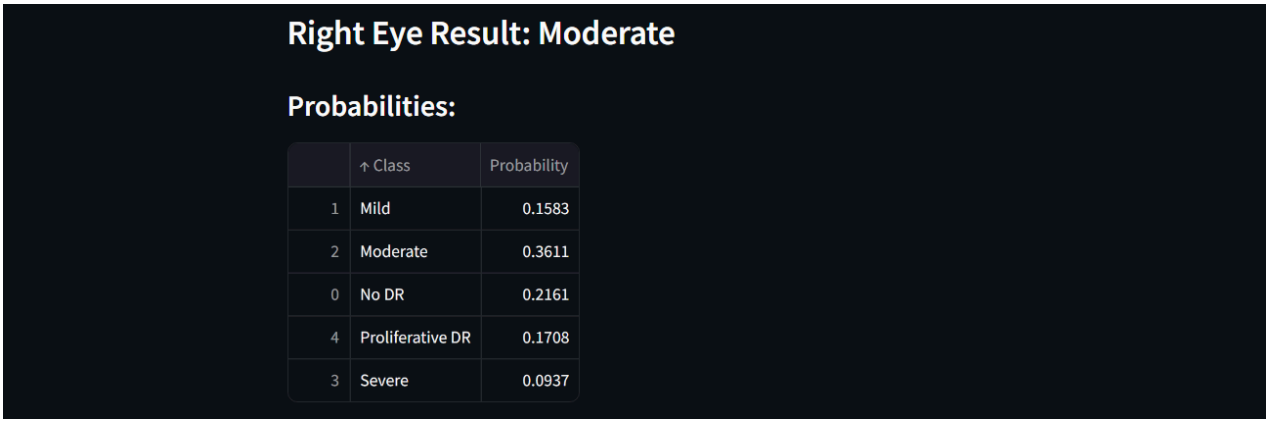


**fig 8.6: Result retinal images right**

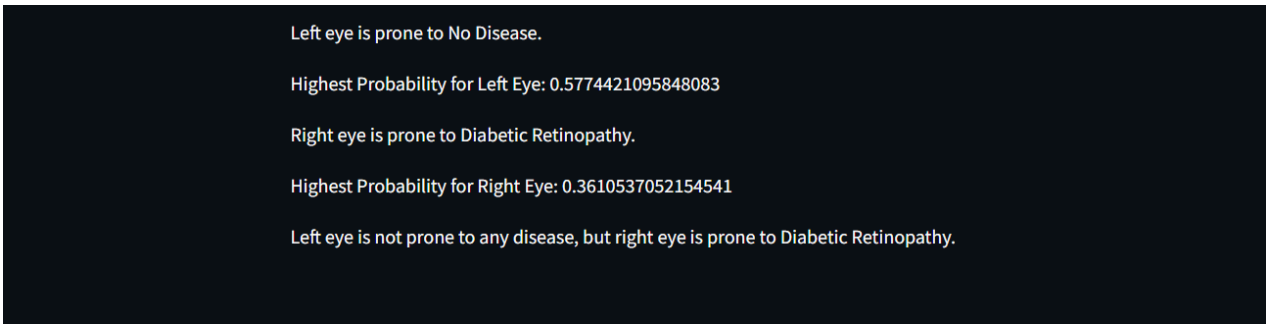


**fig 8.7: Final Result**

# 9. CONCLUSION

In this project, we endeavored to develop an advanced solution for the early detection of diabetic retinopathy using fundus photography images. Leveraging a diverse dataset annotated with retinopathy severity levels, we meticulously preprocessed the images to address real-world challenges such as artifacts and label noise. Our CNN architecture, carefully designed with convolutional and max-pooling layers, was augmented with dropout layers to enhance generalization performance. Through rigorous training and evaluation, employing techniques like stratified K-fold cross-validation and early stopping, our model demonstrated promising results in accurately predicting retinopathy severity. This underscores the potential of deep learning in improving diabetic retinopathy diagnosis, paving the way for enhanced patient care and outcomes.

Moreover, the deployment of our model using Streamlit showcased its practical applicability, enabling healthcare professionals to seamlessly interact with the model and obtain real-time predictions for retinopathy severity. This user-friendly interface facilitates its integration into clinical workflows, empowering clinicians with timely and accurate diagnostic insights. Furthermore, our project highlights the importance of interdisciplinary collaboration between data scientists and healthcare practitioners, underscoring the transformative impact of technology in revolutionizing healthcare diagnostics.

In conclusion, while our model represents a significant step forward in diabetic retinopathy detection, there remains ample room for further refinement and validation in real-world clinical settings. Continued efforts in research and development, coupled with advancements in deep learning techniques and data availability, hold promise for improving the accuracy and accessibility of diabetic retinopathy diagnosis. Ultimately, our endeavor contributes to the broader mission of leveraging technology to enhance healthcare delivery and improve patient outcomes in the fight against diabetic retinopathy.

# 10. BIBLIOGRAPHY

**1. Detection of Diabetic Retinopathy in Retinal Fundus Images Using CNN Classification Models MDPI**:

https://www.mdpi.com/2313-433X/10/1/8

**2. Understanding inherent image features in CNN-based assessment of diabetic retinopathy Nature Journal:**

https://www.nature.com/articles/nrdp201612

**3. CNN-Based Diabetic Retinopathy Status Prediction Using Fundus Images ResearchGate:**

https://iopscience.iop.org/article/10.1088/1757-899X/1022/1/012081/pdf

**4. Convolutional Neural Network (CNN) | TensorFlow Core (Google):**

https://www.datacamp.com/tutorial/cnn-tensorflow-python

This comprehensive tutorial by TensorFlow provides a hands-on introduction to building and training CNNs using TensorFlow. It includes code examples and explanations of key concepts like convolution, pooling, and backpropagation.

**5. Intro to Deep Learning with PyTorch (Udacity Nanodegree Program):**

https://www.udacity.com/course/deep-learning-pytorch--ud188

This Udacity program offers a project-oriented approach to learning deep learning with PyTorch. While it covers various deep learning architectures, it includes modules on CNNs and their training process.

# 11. REFERENCES

[1] Ling Dai, Liang Wu, Huating Li, Chun Cai, Qiang Wu, Hongyu Kong, Ruhan Liu, Xiangning Wang, Xuhong Hou, Yuexing Liu, Xiaoxue Long, Yang Wen, Lina Lu, Yaxin Shen, Yan Chen, Dinggang Shen, Xiaokang Yang, Haidong Zou, Bin Sheng, Weiping Jia. "Telemedicine for Diabetic Retinopathy Screening."
⇒ Link: https://www.nature.com/articles/s41467-021-23458-5

[2] Neha Sengar, Rakesh Chandra Joshi, Malay Kishore Dutta, Radim Burget. "EyeDeep-Net: Multi-class Diagnosis of Retinal Diseases."

[3] Dolly Das, Saroj Kumar Biswas, Sivaji Bandyopadhyay. "Detection of Diabetic Retinopathy using Convolutional Neural Networks for Feature Extraction and Classification (DRFEC)."

[4] Kangrok Oh, Hae Min Kang, Dawoon Leem, Hyungyu Lee, Kyoung Yul Seo, Sangchul Yoon. "Early Detection of Diabetic Retinopathy based on Deep Learning and Ultra-wide-field Fundus Images."

[5] Yashal Shakti Kanungo, Bhargav Srinivasan, Dr. Savita Choudhary. "Detecting Diabetic Retinopathy using Deep Learning." In: 2017 2nd IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT), May 19-20, 2017, India.

[6] Kavakiotis, I.; Tsave, O.; Salifoglou, A.; Maglaveras, N.; Vlahavas, I.; Chouvarda, I. "Machine learning and data mining methods in diabetes research." Compute. Struct. Biotechnol. J. 2017, 15, 104–116.