

# **AI Framework for Diagnosing Stages of Diabetic Retinopathy**

**B.Tech**

**Project Work in  
Department of CSE**



**Submitted By :**

<b>Student Name</b>	<b>Registration Number</b>	<b>Course Code:</b>
<b>Nisarga Patil K</b>	<b>21ETCS002101</b>	<b>CSP402A</b>
<b>Junaidahmed Khazi</b>	<b>21ETCS002089</b>	<b>CSP402A</b>
<b>Krishna shivkant Betgeri</b>	<b>21ETCS002092</b>	<b>CSP402A</b>
<b>Pruthviraj Prakash Revatagoan</b>	<b>21ETCS002107</b>	<b>CSP402A</b>

**Supervisors : Dr. Nayana B R**

**June – 2025**

**FACULTY OF ENGINEERING AND TECHNOLOGY  
M. S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES  
Bengaluru -560 054**

## FACULTY OF ENGINEERING AND TECHNOLOGY



### **Certificate**

*This is to certify that the Project Work titled “**AI Framework for Diagnosing Stages of Diabetic Retinopathy**” is a bonafide record of the work carried out by **Ms. Nisarga Patil K**, Reg. No. **21ETCS002101**, **Mr. Junaidahmed Khazi**, Reg. No. **21ETCS002089**, **Mr. Krishna Shivkant Betgeri**, Reg. No. **21ETCS002092**, **Mr. Pruthviraj Prakash Revatagoan**, Reg. No. **21ETCS002107** in partial fulfilment of requirements for the award of **B.Tech** Degree of M. S. Ramaiah University of Applied Sciences in the Department of Computer Science & Engineering*

**June 2025**

**Dr. Nayana B R**  
**Associate Professor**

Dr Rinki Sharma  
HoD  
Department of CSE,FET

Dr. Sarat Kumar Maharana  
Dean, FET

## Declaration

### ***AI Framework for Diagnosing Stages of Diabetic Retinopathy.***

The project work is submitted in partial fulfilment of academic requirements for the B.Tech . Degree of M. S. Ramaiah University of Applied Sciences in the Department of CSE. This project work is a result of our own work and is in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results, which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of University regulations, hence this project report has been passed through plagiarism check and the repouniversityn submitted to the supervisor

Sl. No.	Reg. No.	Student Name	Signature
1.	21ETCS002101	Nisarga Patil K	
2.	21ETCS002089	Junaidahmed Khazi	
3.	21ETCS002092	Krishna Shivkant Betgeri	
4.	21ETCS002107	Pruthviraj Prakash Ravitagoan	

Date : **20 June 2025**

## Acknowledgements

---

We take this opportunity to express our sincere gratitude and appreciation to **Ramaiah University of Applied Sciences**, Bengaluru for providing us an opportunity to carry out our group project work.

It brings a sense of privilege to associate this work with Dr. Nayana B R, Associate Professor. Working under them have been a precious opportunity. We are immensely grateful for their vigilant supervision, constant encouragement, inspiring ideas, constructive criticism, and timely expertise, which reflects in the final images of this work. We are grateful to the University and thank Dr Rinki Sharma, HOD of Computer Science and Engineering Department for providing us with this opportunity, guidance, and encouragement throughout.

We would like to earnestly thank Dr. Sarat Kumar Maharana, the Dean, FET, for facilitating academic excellence in the college that helped in completing this project. We would like to earnestly thank Prof. Kuldeep Kumar Raina, the Vice Chancellor, for facilitating academic excellence in the college that helped in completing this project. We are thankful to the management of Ramaiah University of Applied Sciences for providing all the facilities and resources for the successful completion of the course.

Finally, we thank our parents, friends and all our well-wishers who have supported us in this endeavor. The values of dedication, discipline, work ethics and patience that we have imbibed here will always remain the guiding force throughout our lives.

## **Abstract**

---

Being one of the leading causes of vision impairment throughout the world, early diagnosis of DR is very important in prevention. The project is about building an AI-based framework for DR diagnosis and classification of DR stages from retinal fundus images. The motivation is the dire need for a rapid, reliable, and interpretable diagnostic tool that could assist clinicians in rural/island setups or wherever resources enabling an ophthalmological expert are scarce.

The framework incorporates a range of advanced methods to ensure performance and explainability. YOLOv8 detects lesions swiftly and accurately-careful attention is given to Microaneurysms, Hemorrhages, and Exudates. DenseNet-121 is used for classification of DR in five clinical stages. Texture-based features extracted with GLCM are used for interpretation. Optuna optimizes hyperparameters, and GradCAM will give insight into the decisions made by the models. A PyQt5-GUI supports interaction with the software where image/video upload and real-time alerts for severe DR cases are possible.

The entire application has been deployed using Inno Setup with batch scripts for easy installation. The results are improved diagnostic performance and interpretability, thanks to the combination

## Table of Contents

---

Certificate .....	(ii)
Declaration.....	(iii)
Acknowledgements.....	(iv)
Abstract .....	(v)
Table of Contents.....	(vi)
List of Tables.....	(x)
List of Figures.....	(xi)
Nomenclature.....	(xii)
Abbreviations and Acronyms.....	(xiii)
Chapter-1: Introduction.....	01
1.1 Introduction.....	01
1.1.1 Motivation and Research Issues.....	06
Chapter-2: Literature Review and Problem Formulation.....	07
2.1 Background Theory.....	07
2.1.1 Literature Review.....	07
2.1.2 Principles and Assumptions.....	07
2.1.3 Key definitions and Implications.....	07
2.1.4 Standard Formulae and Units.....	09
2.1.5 Merits, Demerits and Applicability.....	10
2.1.6 Peripheral Topics Comparison.....	10
2.1.7 Related Work.....	11
2.2 Critical Review of Literature.....	11
2.3 Problem Formulation.....	08
2.3.1 Research Gaps Identified.....	13
2.3.2 Research Questions.....	14

Chapter-3: Problem Statement.....	15
Chapter-4: Design, Modelling and Simulation.....	19
4.1 Design .....	20
4.1.1 Low-Level System Design .....	20
4.1.2 System Architecture .....	21
4.1.2 Block Diagram .....	22
4.1.2 CNN Architecture .....	24
4.2 Implementation.....	27
4.2.1 DR Segmentation.....	27
4.2.2 DR Class.....	30
4.2.3 App.py.....	37
Chapter-5: Results and Discussions.....	42
Chapter-6: Conclusions and Future Directions.....	55
6.1 Conclusions.....	55
6.2 Suggestion for future work.....	55
References.....	57

## List of Tables

---

Table 2.1	Merits, Demerits and Appleciability.....	04
Table 2.2	Summary of significant research articles supporting formulation.....	06
Table 3.1	Methods and Methodology.....	11



## List of Figures

---

Figure 1 Difference between Normal and Retinopathy eye.....	2
Figure 2 Low-Level System Design.....	19
Figure 3 System Architecture .....	20
Figure 4 Block Diagram .....	22
Figure 5 CNN architecture .....	23
Figure 6: DR Grading System .....	24
Figure 7 DR Segmentation code.....	26
Figure 8 DR Segmentation code.....	27
Figure 9 DR Class code.....	29
Figure 10 DR Class code.....	29
Figure 11 DR Class code.....	31
Figure 12 DR Class code.....	32
Figure 13 DR Class code.....	33
Figure 14 App.py Code.....	35
Figure 15 App.py Code.....	36
Figure 16 App.py Code.....	37
Figure 17 App.py Code.....	38
Figure 18 DR Segmentation.....	40
Figure 19 DR Segmentation(training).....	41
Figure 20 DR Segmentation(original and segmentation) .....	42
Figure 21 Calculating the best Parameter for Densenet 121.....	42
Figure 22 Calculating the best Parameter for Resnet 50.....	42
Figure 23 Selecting the best Parameter for Resnet 50 and Densenet 121.....	42
Figure 24 Classification Report for Densenet 121+GLCM.....	43
Figure 25 Confusion Matrix of Densenet 121+GLCM.....	43
Figure 26 Loss, Accuracy and Validation Curve of Densenet 121+GLCM.....	44

---

Figure 27 Classification Report for Resnet 50+GLCM.....	44
Figure 28Confusion Matrix of ResNet50+GLCM.....	45
Figure 29Loss, Accuracy and Validation Curve of ResNet50+GLCM.....	45
Figure 30 Confusion Matrix of Efficent+GLCM.....	46
Figure 31 Confusion Matrix of EfficientNet-B3+GLCM.....	46
Figure 32Loss, Accuracy and Validation Curve of EfficientNet-B3+GLCM.....	47
Figure 33: UserInterface1.....	48
Figure 34 UserInterface2.....	49
Figure 35: UserInterface3.....	50
Figure 36 UserInterface.....	51
Figure 37 UserInterface.....	52

## Abbreviation and Acronyms

---

DR	Diabetic Retinopathy
NPDR	Non-Proliferative Diabetic Retinopathy
PDR	Proliferative Diabetic Retinopathy
DME	Diabetic Macular Edema
OCT	Optical Coherence Tomography
FFA	Fundus Fluorescein Angiography
AI	Artificial Intelligence
CNN	Convolutional Neural Network
GLCM	Gray-Level Co-occurrence Matrix
DL	Deep Learning
ML	Machine Learning
RGB	Red-Green-Blue (Color Model)
ROI	Region of Interest
WHO	World Health Organization
GPU	Graphics Processing Unit
SVM	Support Vector Machine
ReLU	Rectified Linear Unit
DRL	Diabetic Retinopathy Lesions
CAM	Class Activation Map
Grad-CAM	Gradient-weighted Class Activation Mapping
BCVA	Best-Corrected Visual Acuity

## 1. Introduction

---

### 1.1 Introduction

Diabetic Retinopathy (DR) is a serious and progressive eye disease that arises as a complication of long-term diabetes, affecting the retinal blood vessels and potentially leading to partial or complete vision loss. The disease advances through multiple clinical stages—ranging from No DR to Proliferative Diabetic Retinopathy (PDR)—and early diagnosis is crucial to preventing irreversible damage. However, traditional diagnostic workflows rely on manual evaluation of retinal fundus images by trained ophthalmologists, which is often time-consuming, subjective, and resource-intensive. In many parts of the world, especially rural and underdeveloped areas, there is a shortage of qualified eye care professionals, making timely and accurate diagnosis difficult.

This

dissertation presents an artificial intelligence (AI)-driven framework for the automated classification and interpretation of DR from fundus images using deep learning and computer vision techniques. The proposed system integrates two key components: a DenseNet-121 based classification model enhanced with GLCM texture features to classify DR into five clinical stages, and a YOLOv8 model for lesion detection and segmentation. This hybrid model improves both accuracy and clinical interpretability. The classification is further supported by GradCAM visualizations that highlight image regions contributing to the model's decision-making, enhancing transparency for clinicians.

The

significance of this research lies in its holistic approach—combining classification, lesion localization, explainability, and user interface design—to create a deployable, scalable, and user-friendly desktop application. Built using PyTorch and PyQt5, and optimized using Optuna for automated hyperparameter tuning, the system is designed to operate in offline mode as well, ensuring it remains accessible in resource-limited settings. By bridging the gap between medical expertise and AI, this dissertation aims to support early DR detection, reduce diagnostic workloads, and ultimately help prevent avoidable blindness in diabetic patients.

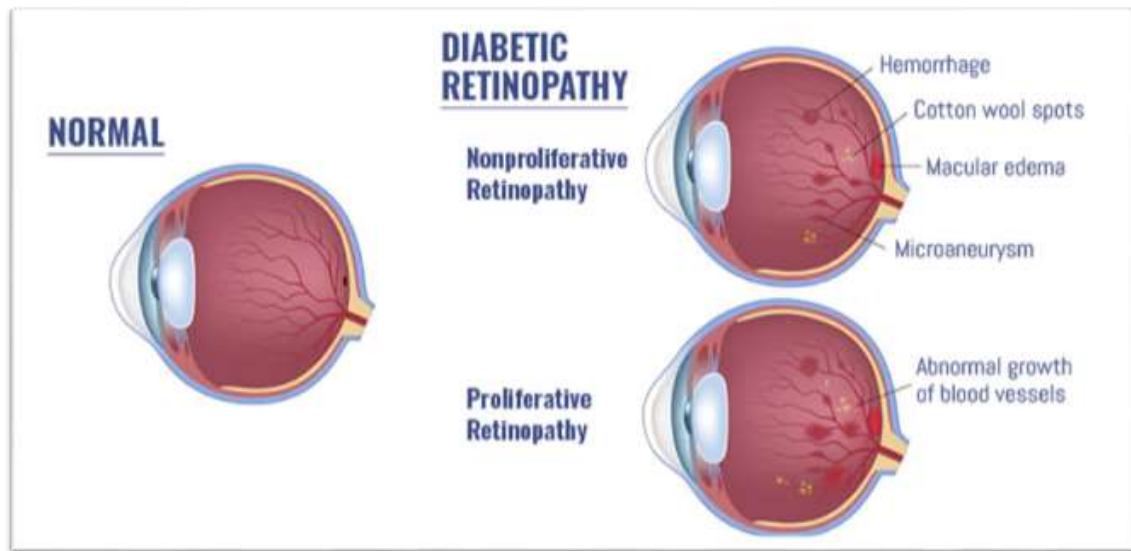


Figure 1 Difference between Normal and Retinopathy eye

Figure 1 shows how the eye of a healthy person differs from the eye of someone with diabetic retinopathy. In the healthy eye, the retina is clear and the blood vessels are clear and not damaged or leaking, which allows a visually-clear and accurate view. A retina that has not been damaged by the complications of diabetes looks bad with leaking macular blood vessels, swollen retina, bright white exudates, and microaneurysms. These visual representations are caused by a prolonged and stagnant period of high blood sugar levels causing damage to the retinal blood vessels, which are capable of leaking/have leaked and caused the retina to swell. The longer there are prolonged high blood sugar levels towards the individual, the more likely it is the eye will continue to develop abnormal weak blood vessels that will leak and cause abnormalities of vision. Left unchecked by a healthcare professional, those changes can disrupt normal vision resulting in partial or complete blindness due to the effects of diabetic retinopathy. The visuals in Figure 1.1 illustrate that the three most important factors in preventing eye damage are having your diabetes properly diagnosed, checking your blood sugar levels frequently, and managing it with medications when prescribed to minimize eye damage and preserve vision.

Early detection and treatment are of utmost importance to stop the progression of DR. However, for screening to take place, retinae must be monitored through the review of fundus images by trained professionals, that requires significant time and health care resources. In several areas with limited health resources, the scarcity of qualified emergency eye care professionals slows down the screening as a whole and puts clinic patients at a substantially higher risk. For these reasons, projects with automated, accurate, and scalable DR detection are urgently needed.

Artificial intelligence, and even more recently deep learning, has been exploring solutions to medical images with a bright future. Specifically, Convolutional Neural Network (CNN) architectures, and advanced models such as ResNet, have also shown their ability to perform at a more than human level. By training these models on previously labelled datasets of retinal images, a computer can learn to perceive and identify differences in patterns as small as  $1\mu\text{m}$  indicating potential DR development which may pass undetected by the human eye.

When being able to distinguish small error changes, computers can drastically reduce the likelihood and error of humans evaluating DR disease and speed up the diagnostic process. This project aims to build a DR detection system based on a deep learning approach. The system is comprised of a backend leveraging Python and PyTorch, and uses CNNs and ResNet architectures to classify images of retina, while offering user interfaces via Tkinter for desktop and Flask for web applications.

the advantages of CNNs for DR detection could result in improved and immediate access to diagnosis by non-specialists, addressing the need for significant human resources for scale-up and delivery.

The strong correlation between diabetic adenoma, which threatens the vision of the diabetic patient, and DR was established in this pilot investigation. The pilot project

also presented an opportunity to educate students in the computer science field on developing deep learning methodologies for the healthcare setting. As previously mentioned, these applications are promising for this area of research. However, the applications will need to address and consider the limitations of droplet activity, which are very real and challenging with respect to margins around retinal data in prior human tests (note the findings directly above). With appropriate resources, students involved in the introduction phase may be able to develop frameworks that could address and integrate droplet activity concerning other medical imaging outputs over the long haul.

We recommend increasing partnerships in knowledge translation across DM, students, and other institutional restrictions associated with volatile activities, standardizing an approach for assessing - student research outlining knowledge transfer learning methodologies to consider how basic instrumental long-term impacts (technology) leads to new use, coherence, and diffusion of research processes packages in the knowledge gap (students), with the aim of eventual incorporation and uptake of knowledge gaps - iterations related to DM.

The use of a standardized process for assessment with international and provincial counterparts and local university resource centres adopting an educational resource document is a priority among partners in DM.

The advantages of manual machine learning for DR detection were made evident by two quarterback teams of students with gradual support resources based on an educational document and frequent pulsed changes to reflect the involvement of teams, their business and respective closure - during activities for scaling up DR outputs.

We suggest expanding local partnerships in knowledge transition across DM, students, and other institutional limitations concerning volatile affairs, standardizing an approach for assessing - students' research establishing knowledge transfer learning methods for how simple instrumental long-term consequences (technology) produces new use, coherence and diffusion of research procedures packaged in information gap (students), eventually leading to acceptance and uptake of information gap - iterations linked to DM.

Improving system support capacity within existing operating systems to use machine learning as a standard framework for estimating detection validity, particularly in challenges concerning DR detection is a priority among stakeholders in DM.

Despite the presence of screening programs and specialized imaging technology, patients remain undiagnosed for (DR) due to inadequate access, especially those patients who are in low resource environments. Even values from supervised human evaluate are not great to allow for country level implementation with even an increased level of access to medical care. In the event of patients being diagnosed, they often find themselves at a stage of treatment too late - requiring more invasive and resource heavy treatment options as they waited for their regularly scheduled eye exam only to find they now have irreversible vision loss that could have been avoided with early screening.

This proposal is clear about the barriers in review to DR screening and detection and is focused on creating a scalable solution through an AI based automated detection for diabetic retinopathy. More specifically, we will be implementing a deep learning capabilities through Convolutional Neural Networks (CNNs) using a large-scale dataset of retinal images. To clarify, a system can very likely be trained to classify DR at various stages with a high degree of accuracy without specialist involvement. By providing a more accessible, less expensive and scalable model where our diagnostic quality will



improve and efficiency using the same time to help reduce preventable blindness to diabetic patients' retina.

### **1.1.1 Motivation and Research Issues**

The primary motivation for this project arises from the growing global burden of diabetes and the associated rise in DR cases, especially in regions with limited access to ophthalmic care. Early-stage DR is often asymptomatic, which means that many patients remain undiagnosed until significant retinal damage has already occurred. The lack of scalable screening programs and specialized tools has exacerbated the problem, leaving millions vulnerable to preventable vision loss. A low-cost, AI-powered solution can fill this critical gap by enabling mass screening and timely intervention.

Key research issues addressed in this project include building a robust deep learning classifier capable of handling the variability and complexity of fundus images. The use of DenseNet-121 is specifically chosen for its superior feature reuse capability and performance in medical image tasks. Additionally, GLCM-based texture features are incorporated to enrich the model's understanding of subtle retinal changes, especially in early DR stages. Hyperparameter optimization using Optuna further ensures that the model performs efficiently without the need for extensive manual tuning.

Another major challenge lies in ensuring that the AI system is not a “black box” but one that clinicians can understand and trust. To address this, the YOLOv8 model segments key retinal lesions—such as microaneurysms, hemorrhages, and exudates—offering tangible visual markers for disease progression. GradCAM heatmaps enhance explainability by showing which image regions influenced the model's decision. Finally, the entire system is wrapped in an intuitive PyQt5 GUI with alert notifications and visualization tools, making it accessible even to non-technical healthcare workers. Together, these components address both the technical and real-world deployment issues in DR diagnosis.

## 2. Literature Review and Problem Formulation

---

### 2.1 Background Theory

#### 2.1.1 Literature Review

Several recent works have dealt with the automated detection diabetic retinopathy (DR) with convolutional neural networks (CNN). Researchers have applied networks such as ResNet, DenseNet, and EfficientNet, to accomplish better classification accuracy of DR stages. Some researchers have implemented visualization methods such as Grad-CAM to increase the explainability of the detection. YOLOv8, a real-time object detection model has been used to perform segmentation tasks in the retinal regions. Finally, optimization libraries such as Optuna have been implemented for adverse detections of the model, while regular GUI libraries such as PyQt5 have been used for user-friendly clinical use by eye care professionals.

#### 2.1.2 Principles and Assumptions

- Deep Learning Principle: CNNs automatically extract hierarchical features from retinal fundus images, removing the need for manual feature engineering.
- Transfer Learning: Pre-trained models on large datasets like ImageNet can be fine-tuned on medical images, improving accuracy even with smaller datasets.
- Assumption: Retinal abnormalities (like microaneurysms and exudates) are visually distinct and can be detected through high-resolution image analysis.

#### 2.1.3 Key Definitions and Implications

- Diabetic Retinopathy (DR): A diabetes-induced damage to retinal blood vessels; early detection prevents vision loss.
- CNN (Convolutional Neural Network): Deep learning architecture suited for image recognition, used here to classify DR severity.
- Grad-CAM: Provides visual explanations of predictions, essential in medical AI for trust and accountability.

- Optuna: A computational intelligence library that optimizes hyperparameters automatically for improved performance.
- PyTorch is an open-source deep learning framework developed at Facebook's AI Research Lab (FAIR). Because of its flexibility, ease of use, and strong community backing, PyTorch finds wide application in developing and training neural networks.
- TorchVision is a library complementary to PyTorch, designed for computer vision tasks. It provides utilities that ease the deep learning workflow for image-based problems such as diabetic retinopathy classification
- OpenCV (Open Source Computer Vision Library) :A very powerful library for real-time image processing (Image resizing, Image filters, Color space conversions (e.g. RGB to Grayscale), and Image enhancements (e.g. Contrast adjustment
- PIL (Python Imaging Library) / Pillow A lightweight image processing library that is best for simple image actions (image loading, saving image to a file, and converting image formats), and is typically used for GUI components, and the first stages of our data pipelines.
- Skimage (Scikit-image) A scientific image processing library that is provides more advanced processing, such as histogram equalization, morphological processing, and some very useful texture feature extracts from images (e.g. GLCM), to highlight the signs of diabetic retinopathy, such as microaneurysms and exudates.
- PyQt5 To ensure the system's practical usage by nontechnical end-users, a GUI was developed using PyQt5. This interface facilitates the uploading of images by the user, displays predictions, and also allows for visualization of explanation maps, thus making it useable in a clinical setting
- ResNet-50 is a deep convolutional neural network with a residual link in the connection structure allowing it to propagate gradients with relative ease during backpropagation. This allows for a much deeper depth of the neural network design without the degradation of the performance. In the case of diabetic retinopathy classification, ResNet-18 provided a good trade-off of performance accuracy and speed of speed fitting which makes it a good candidate for clinical real-time solutions or applications (e.g., at the point of care (POC) or with limited computational resource scenarios).
- DenseNet-121 The DenseNet-121 neural network connects each layer with every other layer in a feed-forward way, which encourages feature reuse (implicitly) and, as a result, reduces the number of parameters in dense layers. Simultaneously,

this model has been designed to improve the flow of the gradient (low impact overfitting), forcing the model to generalise better. The DenseNet makes the best use of the complicated retinal patterns presented with diabetic retinopathy (DR) and was successful at identifying subtle changes across the stages of DR.

- **EfficientNet** The EfficientNet neural network model proposed a lot with their compound scaling capabilities (i.e., to scale depth, width and resolution proportions of a neural network uniformly, but with improved performance and less computational resources used on image classification tasks). EfficientNet on sophisticated image classification tasks achieved state-of-the-art ImageNet accuracy. In DR classification, EfficientNet achieved increased accuracy with very low computational costs making it an attractive option for deployment on external medical imaging platforms.
- **YOLOv8**: An advanced object detection model capable of real-time retinal segmentation, aiding in focused diagnosis.

#### 2.1.4 Standard Formulae, Units, and Relation Between Parameters

- Accuracy =  $(TP + TN) / (TP + TN + FP + FN)$
- Loss Function (e.g., Cross-Entropy) is minimized during training:

$$\mathcal{L}_{CE} = - \sum_{c=1}^C y_c \log(\hat{y}_c)$$

- GradCAM Formulation:

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left( \sum_k \alpha_k^c A^k \right)$$

### 2.1.5 Merits, Demerits, and Applicability

Technology	Merits	Demerits	Applicability
<b>PyTorch</b>	Easy model building and debugging	Steeper learning curve for deployment	Research, model prototyping
<b>ResNet-18</b>	Balanced depth and performance	May underperform on complex patterns	Real-time diagnosis
<b>DenseNet-121</b>	Efficient feature reuse, fewer parameters	Higher training time	Detailed pattern analysis
<b>EfficientNet</b>	Scalable and high-performing	Slightly complex to tune	Low-resource environments
<b>OpenCV, PIL</b>	Simple, fast image preprocessing	Limited for advanced medical image features	Pre-processing pipeline
<b>Skimage</b>	Rich feature extraction (e.g., GLCM)	Slightly slower on large-scale images	Detecting texture-based retinal lesions
<b>Grad-CAM</b>	Enhances model transparency	Interpretability may not be clinically conclusive	Clinical decision support
<b>YOLOv8</b>	Fast and accurate segmentation	May require careful training on medical datasets	Retinal region segmentation
<b>Optuna</b>	Automated optimization	Computationally expensive	Hyperparameter tuning
<b>PyQt5</b>	GUI interface for non-technical users	Limited to desktop-based solutions	Front-end deployment in clinics

Table 2.1 Merits, Demerits and Applicability

### 2.1.6 Peripheral Topics Comparison

Traditional machine learning applications (e.g. SVMs, Random Forests) often rely upon manual feature extraction and have limited potential for scaling-up. Deep learning techniques are capable of higher scalability, and they are also more accurate and objective than these traditional applications.

Previous versions of YOLO (YOLOv3/v4) were useful, but they did not implement any form of architectural revision or specific new focus on segmentation that YOLOv8 does. The

YOLOv8 introduces Generalized ELAN , and EfficientViT, along with the ability to complete segmentation tasks.

Using transfer learning through ImageNet-trained neural networks (e.g., EfficientNets), I have achieved better performance than training from scratch particularly when it comes to locating data in the medical field where leather training data exists.

### 2.1.7 Related Work

This project clearly demonstrates the difference from past work since most have focused on classification or segmentation methods. In this project, I have integrated multiple new methods in deep learning (EfficientNet, YOLOv8), the consideration of computational intelligence (Optuna), and techniques for interpretability (Grad-CAM), all together in a single package with a GUI that is clinician-friendly. This novel system balances accuracy, interpretability, and useability and is a better and well-thought solution for real-life application to diabetic retinopathy screening.

### 2.2 Critical Review of Literature

**Table 2.1: Summary of Significant Research Articles Supporting Problem Formulation**

S.NO	AUTHORS	YEAR OF PUBLICATION	RESEARCH FOCUS	METHODS AND METHODOLOGIES USED	CONCLUSION DRAWN BY AUTHORS	LIMITATION OF STUDY
1	Joseph et al.	2024	How well AI finds eye problems in real doctors' offices.	Checked many past studies to see if AI works.	AI is good at screening; can help doctors work faster	Pictures weren't clear; different AI systems; not much info on problem levels.
2	Quelleg et al.	2021	Lesion detection in retinal images	Ensemble of CNNs with region	Accurate lesion localization classificati	Computationally intensive; high

				proposal networks	on improves diagnostic accuracy	annotation cost for lesions
3	Bilal et al..	2022	Building a ne AI system to find and sort out eye problems .	Built a two-part AI system: outlines eye parts, then classifies the disease.	Their new AI system worked very well and was very accurate	Uneven data for training; needs perfectly labeled examples; might not work for everyone.
4	Quelleg et al.	2021	Lesion detection in retinal images	Ensemble of CNNs with region proposal networks	Accurate lesion localization and classification improves diagnostic accuracy	Computationally intensive; high annotation cost for lesions
5	Gargeya and Leng	2019	DR diagnosis using deep learning	Deep CNN with end-to-end learning	Robust detection system with high accuracy	Limited interpretability and need for explainability
6	Lam et al.	2020	Explaina ble AI for DR diagnosis	Attention-based CNN with visualization modules	Improved model transparency and clinician trust	More complex architecture increases training time

Table 2.2: Summary of Significant Research Articles Supporting Problem Formulation

## Discussion on Table

The studies analyzed represent a wide variety of approaches to diabetic retinopathy classification from handcrafted feature-based systems to fully end-to-end deep learning architectures. While previous methods that are based on texture descriptors (e.g., GLCM or LBP) offer interpretability, they typically do not generalize from large-scale clinical data due to innate limitations in representation capacity. Purely deep learning solutions like CNNs and their derivatives in a larger sense ( e.g., DenseNet) tend to perform better in terms of accuracy but are "black-box" systems that generally lack explainability in the medical context whatsoever.

The novelty of this work is the hybrid approach: we propose a combination of DenseNet-121 and GLCM-derived texture features that uses both high-level deep features and low-level clinical data from fundus images. This improves classification accuracy and interpretable accuracy. Additionally, using Grad-CAM visualizations is useful to show which retinal areas were influential in the model prediction, which provides clarity and increases the clinically acceptable and understandable nature of the system. In contrast to earlier works that tend to focus exclusively on classification or segmentation, we take a comprehensive approach towards the classification task while being mindful of accuracy, clinical explainability, and practical use.

## 2.3 Problem Formulation

### 2.3.1 Research Gaps Identified

- Though progress has been made in DR diagnosis using AI, the current literature suffers from numerous and important deficiencies.
- The existing CNN-only models generally have high accuracy when trained on a theatre of dataset but generally lack clinical interpretability.
- Texture-based models (e.g. GLCM or LBP) tend to have interpretation at a clinical level but are generally limited to being performant in complicated datasets.



- Most of the frameworks do not combine lesion segmentation and classification, resulting in limited prospective uses in clinical diagnostic support.
- Most of the framework hyperparameter tuning is predominantly manual, rendering schemas that depend on hyperparameters limited in reproducibility and versatility across datasets.
- Very little attention has been given to constructing user-ready systems that can operate in offline or low-resource environments.

### **2.3.2 Research Questions**

1. In what ways can deep learning (DenseNet) and image texture (GLCM) be effectively combined to enhance DR classification accuracy?
2. What methods of preprocessing and feature fusion can preserve retinal clinical features from the training data?
3. How does the use of explainability tools such as Grad-CAM, affect clinician trust to use and implement the model?
4. Can the model reliably perform offline, in low resource settings, or on hardware with limited capabilities?
5. How does the hybrid model compare to any stand-alone CNN or GLCM-based system classification of varied DR stages?

### 3. Problem Statement

---

**This chapter should contain the following:**

Preamble to the Chapter

In this section, the actual dissection of project is done and each module is built piece by piece in order to complete the project. In design section, the application has been built in accordance with functional requirements based on which diagrams like Use Case and Low-level Sequence Diagram is drawn. In implementation section, snips of important code are displayed with their explanation given below. In testing section, all functional requirements are tested and the result is analyzed resulting in status of test condition.

- **Title**
  - ❖ AI Framework for Diagnosing Stages of Diabetic Retinopathy
- **Aim**
  - ❖ To develop an automated system for diagnosing the stages of Diabetic Retinopathy (DR) using retinal fundus images. By leveraging deep learning techniques, particularly Convolutional Neural Networks (CNNs), the system will accurately classify the severity of DR, enabling early detection and timely intervention to prevent vision loss.
- **Objectives**
  - ❖ Image Pre-Processing & Segmentation: Refine images and accurately define regions of interest for precise analysis.
  - ❖ Feature Selection and Extraction: Extract and select the most relevant image features using computational intelligence for efficient diagnosis.
  - ❖ Deep Learning for Diagnosis: Employ deep learning models for automated and accurate disease classification.
  - ❖ Explainable AI (XAI): Implement XAI to ensure transparency and trust in AI-driven diagnostic decisions.

- **Scope of Present Investigation**

This project is focused on the creation of an AI-based diagnostic tool using retinal fundus images to detect and classify the various levels of Diabetic Retinopathy. The project is limited to classification and lesion segmentation using deep learning models like DenseNet-121 and YOLOv8 together with GLCM-based texture analysis and GradCAM explainability. The system is implemented in Python and evaluated from numerous dimensions (i.e., many scenarios) using public datasets, with performance assessed using standard metrics such as accuracy, sensitivity, and AUC. The investigation does not include video-based analysis, deploying on mobile devices, or implementing real-time integration with clouds or other tools, but does lay the groundwork for scalable, interpretable, and affordable tools for screening DR in future healthcare systems.

- **Functional Requirements**

1. **FR1:** The system shall enable users to upload retinal fundus photographs in common image formats (e.g., JPG, PNG).
2. **FR2:** The system shall preprocess images by resizing, normalizing, and color space conversion using OpenCV, PIL, and Skimage prior to analysis.
3. **FR3:** The system shall segment important regions of the retina (or lesions) using YOLOv9, so that the segmentation / classification model only assesses areas of relevance.
4. **FR4:** The system shall classify the inputted image in stages of severity for overall diabetic retinopathy (e.g. No DR, Mild, Moderate, Severe, Proliferative), using CNN models.
5. **FR5:** The system shall be able to integrate multiple CNN architectures (ResNet-18, DenseNet-121, EfficientNet), for classification, on PyTorch

6. **FR6:** The system shall leverage Optuna for automated hyperparameter tuning for its models, to try to optimize model accuracy.
  7. **FR7:** The system shall produce Grad-CAM heatmaps to visually interpret its predicted class for each inputted image.
  8. **FR8:** The system shall provide a visual in the GUI of the predicted DR stage, the models confidence score, and the explanation heatmap.
  9. **FR9:** The system should contain a user-friendly GUI, developed with PyQt5, for a clinician and / or user to conveniently interact with the model outputs.
  10. **FR10:** The system shall biologically classify multiple images in a single session of use, allowing a degree of scalability in clinical implementation.
- **Non-Functional Requirements:**
    1. **Scalability:** The system will be able to handle and analyze multiple retinal images or video streams simultaneously, to ensure that the performance of the system remains consistent even in high volume or clinically high throughput settings.
    2. **Response Time:** The application must provide classification outcomes and visualizations within 5 seconds per image, to provide timely evidence based assistive decision support to the health care professionals.
    3. **Security & Privacy:** All retinal images and patient data must be processed locally & securely, and no data should be transacted over a network, while processing retinal images and patient data to comply with medical data privacy guidelines.
    4. **Explainability & Interpretability:** The system is required to generate transparent GradCAM heat maps along with every prediction; where heat map indicates clinically relevant aspects of the retina, to help build trust and ultimately the clinically validation of the application

- **Methods and Methodology/Approach to attain each objective**

Objective No.	Statement of the Objective	Method/ Methodology	Resources Utilised
1	Pre-Processing and Segmentation	Resize and normalize images; detect lesions using YOLOv8 (MA, HE, EX, SE, OD)	Retinal image dataset, Python (OpenCV, PIL, Skimage), YOLOv8, PyTorch, GPU
2	Feature Extraction & Hyperparameter Optimization by Computational Intelligence Algorithms	Extract texture features using GLCM; optimize model parameters with Optuna	Skimage (for GLCM), Optuna, extracted lesion regions, Python, Jupyter Notebook
3	Diagnosing with Deep Learning Models	Train CNN models (DenseNet-121, ResNet-18, EfficientNet) to classify 5 DR stages	PyTorch, TorchVision, DR-labeled
4	Explainable AI (XAI)	Slot-based packet scheduling and adaptive transmission energy model.	MATLAB, custom scheduler, adjustable transmission power logic.
5	To evaluate the optimized schemes against the standard approaches.	Generate GradCAM heatmaps to visualize model decision regions	GradCAM (PyTorch), trained CNN models, PyQt5, test images

Table 3.1 Methods and Methodology

## 4. Design, Modelling and Simulation

---

### Preamble

In this section, the actual dissection of project is done and each module is built piece by piece in order to complete the project. In design section, the web application has been built in accordance with functional requirements based on which diagrams like Use Case and Low - level Sequence Diagram is drawn. In implementation section, snips of important code are displayed with their explanation given below. In testing section, all functional requirements are tested and the result is analyzed resulting in status of test condition.

**Model Development and Training:** Deep learning models, such as convolutional neural networks (CNNs), are designed and trained using the preprocessed data. Multiple architectures are experimented with to identify the most suitable model for the task. Hyperparameters are tuned to optimize performance, and techniques like transfer learning may be employed to leverage pre-trained models for improved accuracy.

**Validation and Evaluation:** The trained models are evaluated using cross-validation techniques to assess their generalization capabilities and robustness. Performance metrics such as accuracy, precision, recall, and F1-score are calculated to quantify the models effectiveness in detecting and classifying diabetic retinopathy[8]

**Interpretability Analysis:** Explainable AI techniques are applied to interpret the decisions made by the deep learning models. Methods like saliency maps, gradient-weighted class activation mapping (Grad-CAM), and attention mechanisms are employed to identify which regions of the retinal images are crucial for classification. This ensures transparency and helps clinicians understand the reasoning behind the model's predictions

## 4.1 Design

Design is necessary when it comes to development since it acts as a blueprint for entire process from requirement making to finished (final product). Hence, in this section designs specific to this project like use case and sequence diagram are attached.

### 4.1.1 Low-Level System Design

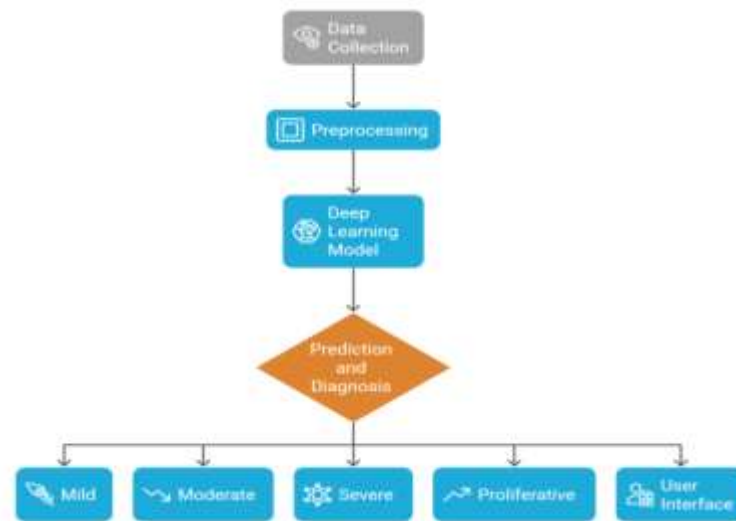


Figure 2 Low-Level System Design

The diagram 2 depicts the entire procedure for an AI-based system that will assist in diagnosing Diabetic Retinopathy (DR). The process begins with the data collection with retinal images from patients. After the images are collected, they will go through preprocessing to ensure the data is consistent and usable by going through processes such as normalizing and resizing. The preprocessed images then go through a deployment of a deep learning model that will extract significant features and patterns. After the feature extraction, the system performs its prediction and diagnosis of the images assigned to different diabetic retinopathy severity levels that include, Mild, Moderate, Severe, and Proliferative. Lastly, the output results are

presented via an interactive user interface allowing clinicians or patients to easily understand the diagnosis and take timely action.

#### 4.1.2 System Architecture

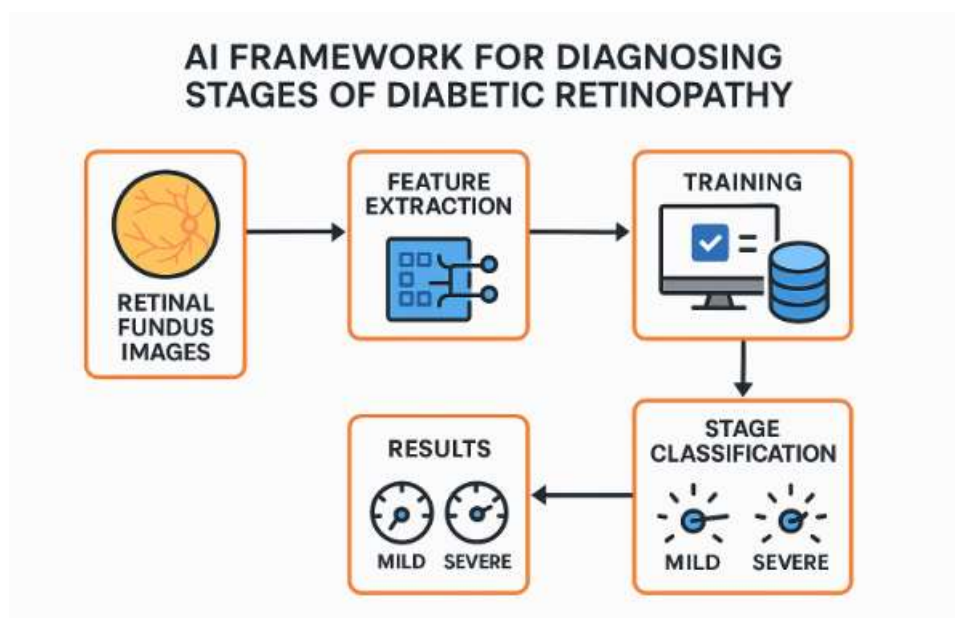


Figure 3 System Architecture

The system architecture Fig 3 to diagnose Diabetic Retinopathy follows a modular pipeline-style of approach with accuracy, efficiency, and explainability. To kick-off the process of diagnosis, images of the retina's fundus are obtained through data selection by obtaining images either from medical datasets or user uploads.

Before images are passed on to a deep learning model, they will be preprocessed including resizing, normalization, and augmenting to ensure that the input is consistent across all images.

After the input images have been preprocessed, they are passed into the deep learning model, which will normally be based on DenseNet or EfficientNet architectures trained for the purpose of recognising retinal lesions in addition to the characteristics of the



different stages of DR.

The prediction and diagnosis module will classify the severity of DR into one of the following four stages: Mild, Moderate, Severe, and Proliferative. For interpretability, we can use XAI techniques such as GradCAM to clearly show users the regions of the image that influenced prediction.

The system will finally present the results through a user interface, which includes the diagnosis, but also alerts where there has been a classification of severe. The use of layered architecture ensures the system is robust, interpretable, and could eventually serve as clinical support.

## Block Diagram

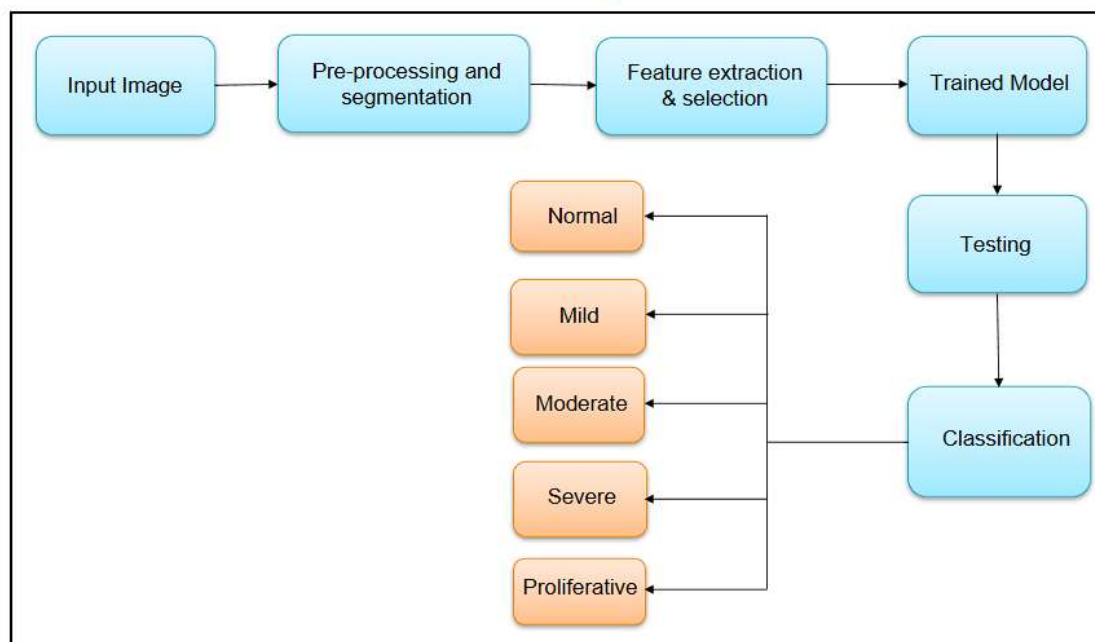


Figure 4 Block Diagram

The block diagram 4 for Diabetic Retinopathy (DR) classification illustrates the flow of processes with retinal fundus images from detection to classification of disease stages. The first block, Image Acquisition, represents the images that were taken from fundus cameras, accommodated across different settings of resolution. The next block was Preprocessing, where the image quality was improved by removal of noise, contrast enhancement, and normalization of images. After the preprocessing block was completed, the next processing block was the Segmentation block, where the most important structural features of the retina were extract: blood vessels, optic disc, and macula. After the Segmentation phase was complete, the system performs the Feature Extraction component, where different features are used to differentiate properties of the image features, e.g., texture (GLCM), shape, intensity, etc. Following features are forwarded into the Classification block of the system, there are many architectures for Deep Learning classification, the main ones are: DenseNet121, ResNet, EfficientNet, etc. The networks are primarily tasked with determining differences of DR severity in their classification casement: No DR – Proliferative DR. Finally, there was an Explainability module that was used, with either sevExam or Grad-CAM, which is utilized to produce heatmaps that represented areas of the retina that a model was exerting its cognizance, thus assisting in transparency and clinical verifiability.

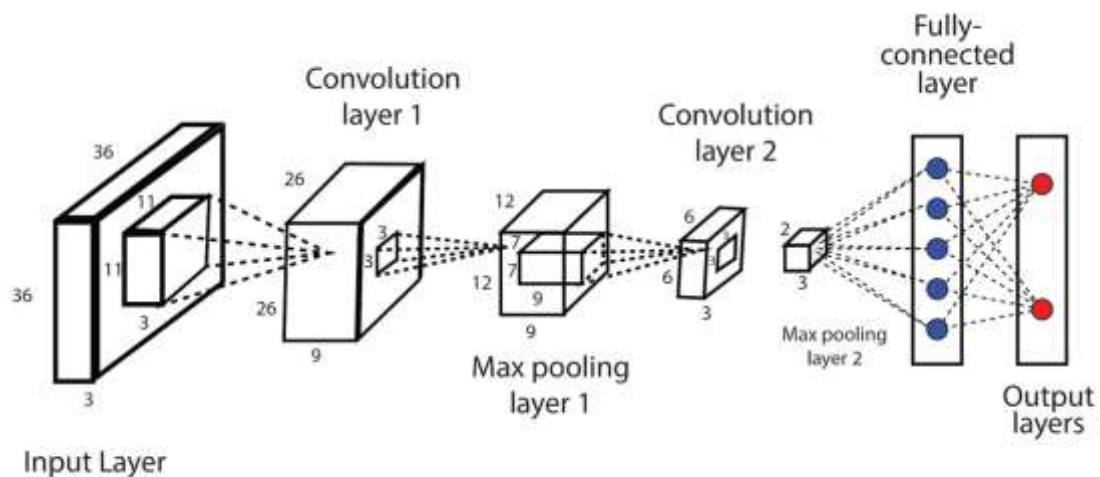


Figure 5 CNN architecture

The shown in Fig 5 system architecture is that of a Convolutional Neural Network (CNN) which is well suited towards image-based classification problems like Diabetic Retinopathy (DR) detection.

In our DR project, this CNN architecture is central to automatically learning hierarchical feature extraction from retinal fundus images. The model starts with an input layer of size  $36 \times 36 \times 3$  which represents the three RGB channels of the input images.

Then, it subsequently goes through two convolutional layers and each of the two convolutional layers is followed by a max pooling layer that reduces the spatial dimensions of the feature maps while preserving crucial features. These layers help to learn the identified features like microaneurysms, hemorrhages, or exudates.

The resulting feature map is then flattened and sent directly to a fully connected layer, which learns the complex interactions of numerous features and finally outputs to the output layer for classification into the individual class labels to represent each of the different stages of DR (e.g. No DR, Mild NPDR, Moderate NPDR, Severe NPDR, PDR). This architecture enables early detection and grading of DR and the relevant features will help aid any concluding diagnosis and treatment plan made by an ophthalmologist.

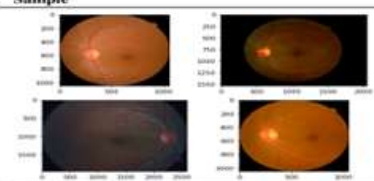
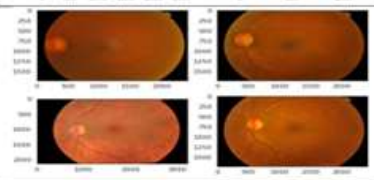
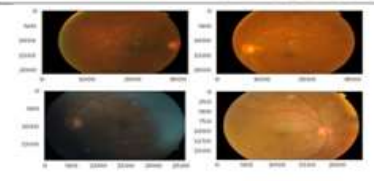
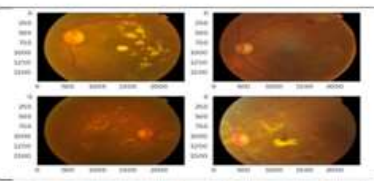
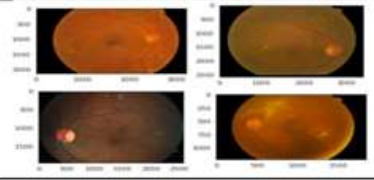
Sample	Characteristics
	<b>Label 0: No diabetic retinopathy (NoDR)</b>
	<b>Label 1: Mild nonproliferative retinopathy:</b> In this early stage of the disease, small patches of balloon-like swelling in the small blood cells in the retina, known as microaneurysms. The fluid will leak into the retinas through these microaneurysms as shown in the left images.
	<b>Label 2: Moderate nonproliferative retinopathy:</b> As the disease progresses, blood vessels feeding the retina may swell and distort and also lose blood transportation capacity. These conditions cause significant changes to the appearance of the retina and can contribute to diabetic macular edema (DME) as shown in the left images.
	<b>Label 3: Severe nonproliferative retinopathy:</b> Many further blood vessels are blocked, which deprive the retinal region of blood supply. These regions secrete growth factors that suggest that the retina is forming new blood vessels as shown in the left images.
	<b>Label 4: Proliferative diabetic retinopathy (PDR):</b> This more serious form called proliferative diabetic retinopathy. Damaged blood vessels are blocked in the retina in this case, causing the development of irregular new blood vessels, and can flow into the clear, jelly-like substance that fills the center of the eye (vitreous). Scar tissue stimulated through new blood vessel growth can gradually separate the retina from the posterior of the eye. Therefore, retinal detachment could lead to permanent eyesight loss as shown in the left images.

Figure 6: DR Grading System

Diabetic Retinopathy in fig 6 (DR) grading is a standard classification system for gradings of retinopathy from diabetes used by ophthalmic surgeons. The system consists of five classifications that rely on the signs of diabetic retinal damage (DR). Because the DR grading is standardized, it consists of these four DR classifications and one no DR classification with the goal of seeing improvement or accessibility to commence treatment. The severity levels include: No DR - retina is healthy and shows no signs of diabetic damage; Mild Non-Proliferative DR (NPDR) - characterized by the presence of microaneurysms; Moderate NPDR- retina has microaneurysms, retinal hemorrhages, and some leakage of blood vessels; Severe NPDR - retina has extensive areas of retinal hemorrhaging, venous beading, and more widespread areas of capillary non-perfusion; Proliferative DR (PDR) - the most severe, with signs of neovascularization (more blood vessels), vitreous hemorrhage, retinal detachment and can lead to blindness if not diagnosed early enough. Early grading and diagnoses are vital for treating and preventing future vision loss.

The system utilizes two widely accepted, high-quality datasets for training and evaluation: APTOS 2019 Blindness Detection. The datasets consist of labeled retinal fundus photographs acquired under various conditions and from diverse populations, making them ideal for constructing a robust deep learning model. Each image is annotated by ophthalmological experts with a diabetic retinopathy grade ranging from 0 to 4, following the international DR grading system:

- Grade 0: No diabetic retinopathy
- Grade 1: Mild non-proliferative DR
- Grade 2: Moderate non-proliferative DR
- Grade 3: Severe non-proliferative DR
- Grade 4: Proliferative DR

## 4.2 Implementation

### 4.2.1 DR Segmentation

```

CLASS_NAMES = ["MA", "HE", "EX", "SE", "OD"]
CLASS_COLORS = {
    "MA": (0, 0, 255), # Red
    "HE": (0, 255, 0), # Green
    "EX": (255, 0, 0), # Blue
    "SE": (255, 255, 0), # Cyan
    "OD": (255, 0, 255) # Orange
}

def plot_sample_image(image_path, label_path):
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Get image filename
    image_name = os.path.basename(image_path)

    # Read YOLO annotation file
    with open(label_path, "r") as f:
        boxes = f.readlines()

    h, w, _ = image.shape

    # Draw bounding boxes with class labels
    for box in boxes:
        class_id, x, y, bw, bh = map(float, box.strip().split())
        class_id = int(class_id)
        class_name = CLASS_NAMES[class_id]
        color = CLASS_COLORS[class_name]

        x1 = int((x - bw / 2) * w)
        y1 = int((y - bh / 2) * h)
        x2 = int((x + bw / 2) * w)
        y2 = int((y + bh / 2) * h)

        # Draw bounding box
        cv2.rectangle(image, (x1, y1), (x2, y2), color, 2)

        # Put class label
        cv2.putText(image, class_name, (x1, y1 - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    # Display image with title
    plt.figure(figsize=(6, 6))
    plt.imshow(image)
    plt.title(f"Image: {image_name}", fontsize=10, fontweight="bold")
    plt.axis("off")
    plt.show()

# Show one sample image
sample_image_path = glob.glob(os.path.join(train_images_path, "*.jpg"))[15]
sample_label_path = os.path.join(train_labels_path, os.path.basename(sample_image_path).replace(".jpg", ".txt"))
plot_sample_image(sample_image_path, sample_label_path)

```

Figure 7 DR Segmentation code

This visualization in the Figure 7 script plots YOLO-based bounding box annotations over the original fundus images to provide interpretability, and confirm that the bounding box

annotations are correctly implementing the lesion detection results. Each type of lesion (Microaneurysms (MA), Hemorrhages (HE), Exudates (EX), Soft Exudates (SE), and Optic Disc (OD)) is represented by a different color, for ease of understanding. The function `plot_sample_image()` calls in the image path and the associated YOLO label file; it reads in the x—y coordinates of the bounding boxes, and converts the normalized YOLO form into pixel values based on the image size. It uses OpenCV to overlay these bounding boxes on top of the image, and it annotates each lesion with a label from the model. It uses Matplotlib to display the modified image. This tool not only assists in verifying that the localization of the lesions is correct, but enhances the interpretability of the detection module, which is particularly useful for clinician assisted reviews of the model or system debugging efforts.

```
# Load YOLOv8 model
model = YOLO("/kaggle/working/runs/detect/train/weights/best.pt")

# Image path
image_path = "/kaggle/input/idrid-yolo/images/train/IDRiD_01.jpg"

# Run inference (prediction)
results = model(image_path, save=True)

# Load the original image
original_image = Image.open(image_path)

# Get the predicted image as a NumPy array (with bounding boxes)
predicted_img = results[0].plot()

# Convert BGR to RGB (Matplotlib expects RGB format)
predicted_img = cv2.cvtColor(predicted_img, cv2.COLOR_BGR2RGB)

# Display the original and predicted images side by side
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(original_image)
plt.axis("off")
plt.title("Original Image")

plt.subplot(1, 2, 2)
plt.imshow(predicted_img) # Now correctly formatted in RGB
plt.axis("off")
plt.title("Predicted Output")

plt.show()
```

### Figure 8 DR Segmentation code

DR Segmentation code This visualisation in the Figure 8 pipeline illustrates the output of the YOLOv8 model on a diabetic retinopathy (DR) fundus image. The script first loads a YOLOv8 model that has been pre-trained and fine-tuned on lesion detection and performs inference on the selected fundus image. The original is loaded using PIL to facilitate clear display, while the prediction report on the original image with the bounding boxes around the identified lesions, is captured and formatted. In addition, OpenCV handles images in BGR format whereas Matplotlib requires RGB format; we ensure that the colour is rendered correctly as such. Finally, we display both the untouched original image and the image alongside the predicted lesion locations using Matplotlib. This visualisation is invaluable as a means to understand model performance and provides a visual means to localise the features of pathological findings for more clarity and clinical relevance.



### 4.2.2 DR Class

```
class GLCMFeatureExtractor:
    """GLCM Feature Extractor optimized for DR classification"""

    def __init__(self, distances=[1, 2], angles=[0, 45, 90, 135], levels=32):
        self.distances = distances
        self.angles = [np.deg2rad(angle) for angle in angles]
        self.levels = levels
        print(f"GLCM Extractor: {len(distances)} distances, {len(angles)} angles, {levels} levels")

    def extract_glcml_features(self, image):
        """Extract compact GLCM features from PIL image"""
        try:
            # Convert PIL to numpy and preprocess
            if isinstance(image, Image.Image):
                img_array = np.array(image)
            else:
                img_array = image

            # Use green channel for better retinal contrast
            if len(img_array.shape) == 3:
                gray = img_array[:, :, 1] # Green channel
            else:
                gray = img_array

            # Resize and enhance
            gray = cv2.resize(gray, (128, 128)) # Smaller for speed
            gray = cv2.equalizeHist(gray)

            # Reduce gray levels
            gray = (gray // (256 // self.levels)).astype(np.uint8)

            # Extract GLCM features
            features = []

            for distance in self.distances:
                glcm = graycomatrix(gray, distances=[distance], angles=self.angles,
                                    levels=self.levels, symmetric=True, normed=True)

                # Core texture properties
                contrast = graycoprops(glcm, 'contrast').flatten()
                homogeneity = graycoprops(glcm, 'homogeneity').flatten()
                energy = graycoprops(glcm, 'energy').flatten()
```

Figure 9 DR Class code

This module in a Figure 9 is a hybrid deep learning system created to automatically classify diabetic retinopathy (DR) from fundus images. The design uses the feature learning capability of DenseNet-121 and the hand extracted texture features acquired using GLCM (Gray Level Co-occurrence Matrix). The GLCMFeatureExtractor class takes care of the

GLCM statistics: contrast, energy, and homogeneity - sampled at multiple spatial distances and angles, using the metadata from green channel of retinal image (which has better contrast for DR lesions). Normalized GLCM features are flattened into a fixed size vector.

```
class CombinedDenseNetClassifier(nn.Module):
    """DenseNet121 + GLCM Combined Classifier"""

    def __init__(self, n_classes, glcm_size=12):
        super(CombinedDenseNetClassifier, self).__init__()
        print(f"🚧 Building DenseNet121 + GLCM model...")

        # DenseNet121 backbone
        self.base_model = models.densenet121(pretrained=True)
        self.base_model.classifier = nn.Identity()
        self.feature_size = 1024
        self.glcm_size = glcm_size

        # Combined classifier
        self.classifier = nn.Sequential(
            nn.BatchNorm1d(self.feature_size + self.glcm_size, momentum=0.009, eps=0.001),
            nn.Linear(self.feature_size + self.glcm_size, 1024),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(1024, 512),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Dropout(0.4),
            nn.Linear(256, n_classes)
        )

        print(f"✅ Model built: {self.feature_size} CNN + {self.glcm_size} GLCM → {n_classes} classes")

    def forward(self, x, glcm_feats):
        # DenseNet features
        x = self.base_model(x)

        # Combine CNN and GLCM features
        combined = torch.cat((x, glcm_feats), dim=1)

        # Final classification
        return self.classifier(combined)

def create_data_loaders(data_dir, batch_size=16, val_split=0.2, test_split=0.1):
    """Create train/val/test data loaders from folder structure"""

    print("🚧 Setting up data transformations...")
```

Figure 10 DR Class code

In the above Figure 10 Optuna is embedded in the framework discussed in this chapter for tuning hyperparameters to improve the DenseNet121 + GLCM hybrid classifier. The `objective_densenet_glcm` function defines the search space of a range of important parameters, including learning rate, momentum, batch size, and dropout rates in different

layers of the last fully connected classifier. For each trial, these parameters are sampled and a version of the CombinedDenseNetClassifier is instantiated with the dropout values assigned dynamically.

After the construction of the model, the function will then go through the train\_loader for a few epochs and value the model with the val\_loader. The optimizer (SGD or Adam) is initialized based on the trial-specific learning rate and momentum and will be optimally guided based on early stopping or validation accuracy combinations to inform the optimization direction. With this in mind, the best combined training parameters for good classification of the relevant classes across all levels of diabetic retinopathy can be identified.

This pipeline allows for the power of deep feature learning from a DenseNet approach to be coupled with hand-crafted GLCM textures descriptors ( from pix/image analysis), while once again allowing Optuna to automate model tuning, resulting in a more fluid and optimized diagnostic model, we hope, to be utilized in real-world clinics within ophthalmology, or (diabetic retinopathy) screening.

```
def plot_training_results(train_accs, val_accs, train_losses, val_losses, test_accuracy, class_names, all_labels, all_preds):
    """Plot training curves and confusion matrix"""

    plt.figure(figsize=(15, 5))

    # Loss curves
    plt.subplot(1, 3, 1)
    plt.plot(train_losses, label='Train Loss', color='blue')
    plt.plot(val_losses, label='Val Loss', color='red')
    plt.title('Loss Curves')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.grid(True)

    # Accuracy curves
    plt.subplot(1, 3, 2)
    plt.plot(train_accs, label='Train Acc', color='blue')
    plt.plot(val_accs, label='Val Acc', color='red')
    plt.title('Accuracy Curves')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.grid(True)

    # Validation accuracy progress
    plt.subplot(1, 3, 3)
    plt.plot([acc * 100 for acc in val_accs], 'g-', linewidth=2)
    plt.axhline(y=test_accuracy*100, color='r', linestyle='--', label=f'Test Acc: {test_accuracy:.3f}')
    plt.title('Validation Accuracy Progress')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy (%)')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    plt.show()
```

Figure 11 DR Class code

In the Figure 11 diabetic retinopathy classification pipeline that employs a hybrid approach that uses DenseNet121 for deep feature extraction and GLCM to capture texture-based information. The classification pipeline consists of a series of steps, some pre-processing and some model-training, that can be summarized in the following workflow: first the stratified train, valid, and test datasets are prepared from a folder-based image dataset to ensure that the representation among DR severity classes are balanced; then data augmentation is performed through various transformations including random flips, rotation, and color jittering to improve training generalization; next, the most appropriate hyperparameters for training the deep learning model, including learning rate, dropout rates, and batch size, are optimized using Optuna, a hyperparameter optimization framework; a number of different trials are performed to evaluate multiple different configurations until the best fitting approach is identified, and then the deterministic model is trained fully to evaluate the model according to the optimized hyperparameters; losses and accuracies are recorded for both training and validation sets throughout the training procedure; the final model is then evaluated on the test set and a confusion matrix is generated to evaluate performance based on each class; finally, model training trends and outcomes of the classification process are visualized for purposes of interpretability, and explainability, and the key performance indicators such as best validation accuracy and the final test accuracy measures are displayed. This pipeline has been constructed to produce a robust, interpretable, and explainable approach to diabetic retinopathy detection using both deep features, and texture-based features, in a structured and understandable process.

```

def objective_resnet50_glm(trial, train_loader, val_loader, n_classes):
    """Optuna objective function for ResNet50 + GLCM hyperparameter optimization"""

    # Suggest hyperparameters
    lr = trial.suggest_loguniform("lr", 1e-5, 1e-2)
    momentum = trial.suggest_uniform("momentum", 0.5, 0.99)
    dropout_1 = trial.suggest_uniform("dropout_1", 0.1, 0.4)
    dropout_2 = trial.suggest_uniform("dropout_2", 0.2, 0.5)
    dropout_3 = trial.suggest_uniform("dropout_3", 0.3, 0.6)

    print(f"\n Trial {trial.number}: lr={lr:.6f}, momentum={momentum:.3f}")

    # Create model with trial-specific dropout
    class TrialResNet50Classifier(nn.Module):
        def __init__(self, n_classes, glm_size=12):
            super(TrialResNet50Classifier, self).__init__()
            self.base_model = models.resnet50(pretrained=True)
            self.base_model.fc = nn.Identity()
            self.feature_size = 2048
            self.glm_size = glm_size

            self.classifier = nn.Sequential(
                nn.BatchNorm1d(self.feature_size + self.glm_size, momentum=0.999, eps=0.001),
                nn.Linear(self.feature_size + self.glm_size, 1024),
                nn.ReLU(),
                nn.Dropout(dropout_1),
                nn.Linear(1024, 512),
                nn.ReLU(),
                nn.Dropout(dropout_2),
                nn.Linear(512, 256),
            )

```

Figure 12 DR Class Code

In the above Figure 12 Optuna is embedded in the framework discussed in this chapter for tuning hyperparameters to improve the Resnet + GLCM hybrid classifier. The `objective_resnet_glm` function defines the search space of a range of important parameters, including learning rate, momentum, batch size, and dropout rates in different layers of the last fully connected classifier. For each trial, these parameters are sampled and a version of the CombinedResNetClassifier is instantiated with the dropout values assigned dynamically.

After the construction of the model, the function will then go through the `train_loader` for a few epochs and value the model with the `val_loader`. The optimizer (SGD or Adam) is initialized based on the trial-specific learning rate and momentum and will be optimally guided based on early stopping or validation accuracy combinations to inform the optimization direction. With this in mind, the best combined training parameters for good classification of the relevant classes across all levels of diabetic retinopathy can be identified.

This pipeline allows for the power of deep feature learning from a ReseNet approach to be coupled with hand-crafted GLCM textures descriptors ( from pix/image analysis), while once again allowing Optuna to automate model tuning, resulting in a more fluid and optimized diagnostic model, we hope, to be utilized in real-world clinics within ophthalmology, or (diabetic retinopathy) screening.

```
class CombinedEfficientNetB3Classifier(nn.Module):
    """EfficientNet-B3 + GLCM Combined Classifier"""

    def __init__(self, n_classes, glcm_size=12):
        super(CombinedEfficientNetB3Classifier, self).__init__()
        print(f"🚧 Building EfficientNet-B3 + GLCM model...")

        if EFFICIENTNET_AVAILABLE:
            # Use efficientnet_pytorch library
            self.base_model = EfficientNet.from_pretrained('efficientnet-b3')
            self.base_model.fc = nn.Identity() # Remove final FC layer
            self.feature_size = 1536 # EfficientNet-B3 feature size
            self.use_pytorch_efficientnet = True
            print("✅ Using efficientnet_pytorch library")

        else:
            # Use torchvision version
            try:
                self.base_model = models.efficientnet_b3(pretrained=True)
                self.base_model.classifier = nn.Identity()
                self.feature_size = 1536
                self.use_pytorch_efficientnet = False
                print("✅ Using torchvision EfficientNet-B3")
            except:
                # Fallback to ResNet34 if EfficientNet not available
                print("⚠️ EfficientNet-B3 not available, using ResNet34 as fallback...")
                self.base_model = models.resnet34(pretrained=True)
                self.base_model.fc = nn.Identity()
                self.feature_size = 512
                self.use_pytorch_efficientnet = False

        self.glcm_size = glcm_size

        # Combined classifier
        self.classifier = nn.Sequential(
            nn.BatchNorm1d(self.feature_size + self.glcm_size, momentum=0.000, eps=0.001),
            nn.Linear(self.feature_size + self.glcm_size, 1024),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(1024, 512),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Dropout(0.4),
            nn.Linear(256, n_classes)
        )
```

Figure 13 DR Class Code

In the above Figure 13 Optuna is embedded in the framework discussed in this chapter for tuning hyperparameters to improve the efficient net+ GLCM hybrid classifier. The

objective\_ efficient net \_glcm function defines the search space of a range of important parameters, including learning rate, momentum, batch size, and dropout rates in different layers of the last fully connected classifier. For each trial, these parameters are sampled and a version of the efficient net Classifier is instantiated with the dropout values assigned dynamically.

After the construction of the model, the function will then go through the train\_loader for a few epochs and value the model with the val\_loader. The optimizer (SGD or Adam) is initialized based on the trial-specific learning rate and momentum and will be optimally guided based on early stopping or validation accuracy combinations to inform the optimization direction. With this in mind, the best combined training parameters for good classification of the relevant classes across all levels of diabetic retinopathy can be identified.

This pipeline allows for the power of deep feature learning from a efficientnet approach to be coupled with hand-crafted GLCM textures descriptors ( from pix/image analysis), while once again allowing Optuna to automate model tuning, resulting in a more fluid and optimized diagnostic model, we hope, to be utilized in real-world clinics within ophthalmology, or (diabetic retinopathy) screening.



### 4.2.3 App.py

```

1  import sys
2  import os
3  import torch
4  from PyQt5.QtWidgets import (
5      QApplication, QMainWindow, QPushButton, QLabel, QVBoxLayout,
6      QWidget, QFileDialog, QTabWidget, QMessageBox, QTextEdit
7  )
8  from PyQt5.QtGui import QPixmap, QImage
9  from PyQt5.QtCore import QThread, pyqtSignal, Qt
10 from PIL import Image
11 import torch.nn as nn
12 import torchvision.transforms as transforms
13 from torchvision import models
14 from ultralytics import YOLO
15 import numpy as np
16 import cv2
17 from skimage.feature import graycomatrix, graycoprops
18 import matplotlib.pyplot as plt
19 from io import BytesIO
20
21 device = 'cpu'
22
23 class GradCAM:
24     def __init__(self, model, target_layer, glcm_features=None):
25         self.model = model
26         self.target_layer = target_layer
27         self.glcm_features = glcm_features
28         self.gradients = None
29         self.activations = None
30         self.register_hooks()
31
32     def register_hooks(self):
33         def forward_hook(module, input, output):
34             self.activations = output
35
36         def backward_hook(module, grad_input, grad_output):
37             self.gradients = grad_output[0]
38
39         self.target_layer.register_forward_hook(forward_hook)
40         self.target_layer.register_backward_hook(backward_hook)
41
42     def generate_cam(self, input_tensor, glcm_tensor, target_class=None):
43         self.model.zero_grad()
44 
```

Figure 14 App.py Code

This code in the above Figure 14 presents an explainability pipeline using Grad-CAM for a hybrid DenseNet121 + GLCM-based diabetic retinopathy classifier. The GradCAM class hooks into the specified CNN layer to capture activations and gradients during backpropagation and uses these to compute a Class Activation Map (CAM) which indicates which regions were most important to the prediction. The method overlays this heatmap on the original image for visual inspection.

The GLCMFeatureExtractor class extracts texture features in conjunction with training by calculating Gray-Level Co-occurrence Matrices (GLCM) from the green channel of the input retinal image. There are different distances and angles evaluated, and the procedure also allows for histogram equalization and gray level reduction as a way to improve contrast and reduce dimensionality. This combination of spatial-texture features and



visual saliency gives rise to an example of interpretable, reliable deep learning models in medical imaging.

[illegible]

Figure 15 App.py code

This section in Figure 15 of the PyQt5 GUI layout introduces the Classifier tab of your diabetic retinopathy diagnosis app using DenseNet121 + GLCM (12 features). Here is a summary in paragraph form:

The interface has a classifier tab that focuses on inference and explainability. The tab contains an image label that includes instructions for the user (i.e., your device... the model type DenseNet121 + GLCM... and a Bordered grey background). This section contains a "Load Image" button that enables the user to upload the fundus image for analysis. Once an image is uploaded, and the user presses the "Perform Inference" button, the inference will be performed through the model pipeline. If the user also chooses to

include an explanation of the prediction probabilities, pressing the "Explain with GradCAM" button will apply an interpretability heatmap on top of the input image to visualize the areas of attention regarding the decision. Prediction labels are displayed in a label below the image. Additionally, there is a legend that defines the anatomical or pathological markers, such as MA (Microaneurysms), HE (Hemorrhages), EX (Exudates), SE (Soft Exudates), and OD (Optic Disc). Alignment is centred, and the spacing throughout the interface is consistent to provide a clean diagnostic workflow. This UI is designed to look clean and medical-grade.

```
def show_severity_popup(self, predicted_class, confidence):
    if predicted_class in ['PDR', 'Severe_NPDR']:
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Critical)
        msg.setWindowTitle("Severity Classification")
        msg.setText(f"({predicted_class}) detected.\nSevere stage of DR. Immediate attention needed!\nConfidence: {confidence:.2%}")
        msg.exec_()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = ImageClassifierApp()
    window.show()
    sys.exit(app.exec_())
```

Figure 16 App.py code

In the Figure 16 When the classifier predicts a severe stage of diabetic retinopathy, specifically Proliferative DR (PDR) or Severe Non-Proliferative DR (Severe\_NPDR), the show\_severity\_popup function triggers a high alert popup. A red danger alert dialog is generated using QMessageBox that alerts the user of the condition's severity and the classification confidence score asserted by the model. This facilitates immediate user acknowledgement to high risk predicted classifications. The application continues by initiating the standard PyQt5 QApplication loop, which instantiates the GUI and keeps it running while allowing for user interactions.

```

def init_ui(self):
    self.tabs = QTabWidget()
    self.setCentralWidget(self.tabs)
    self.setStyleSheet("""
        QMainWindow {
            background-color: #a5a5a5;
        }
        QLabel {
            font-size: 16px;
            color: #333;
        }
        QPushButton {
            font-size: 15px;
            padding: 10px;
            background-color: #4a90e2;
            color: white;
            border-radius: 6px;
        }
        QPushButton:hover {
            background-color: #357ABD;
        }
        QTextEdit {
            font-size: 14px;
            background-color: #ffffff;
            border: 1px solid #ccc;
            padding: 10px;
        }
    """)

    QTabWidget::pane {
        border: none;
    }
    QTabBar::tab {
        padding: 10px 20px;
        background: #ddd;
        font-size: 14px;
        border-top-left-radius: 6px;
        border-top-right-radius: 6px;
        min-width: 140px;
    }
    QTabBar::tab:selected {
        background: #ffffff;
    }
    """
)

self.info_tab = QWidget()

```

Figure 17 App.py code

The `init_ui` method in the Figure 17 initializes the general layout of the main GUI using PyQt5. The method also applies a modern stylesheet, which is clean and sharp with consistent theming. Below is a brief overview of how it is structured:

The method instantiates a `QTabWidget` as the central widget in the application, allowing multiple tabbed views consisting of image classification, information to be displayed, or segmentation on images, etc. The embedded stylesheet makes the appearance pleasing with a light gray background, styled button with hover effects, rounded tabs, and text with readable formatting. This user interface design helps create a clear and usable

experience, which is very important for medical imaging applications like diagnosing diabetic retinopathy.

---

## 5. Results and Discussions

---

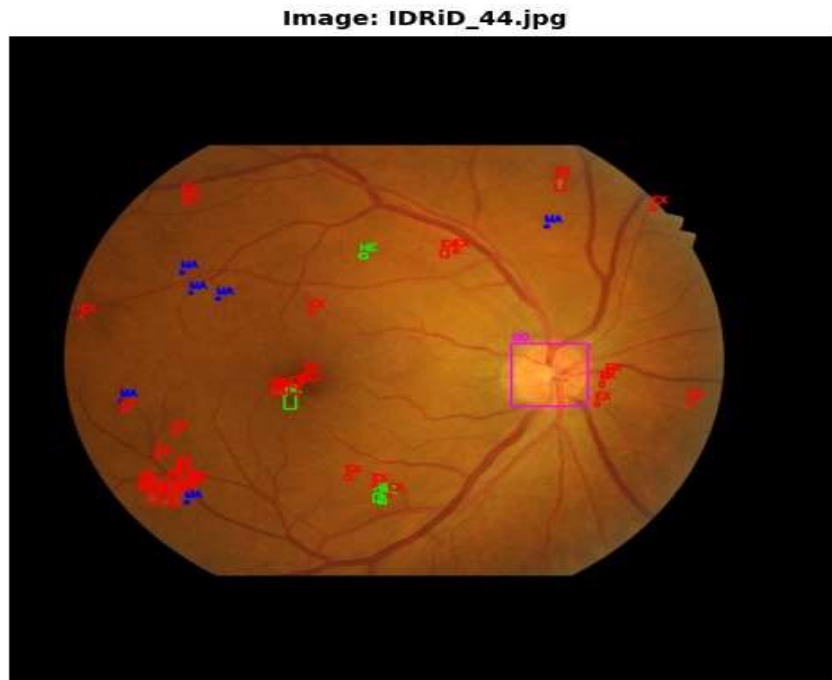


Figure 18 DR Segmentation

```

Epoch      GPU_mem  box_loss  cls_loss  dfl_loss  Instances  Size
97/100      6.64G    2.05     1.452     1.026     634        1024: 100%|██████████| 7/7 [00:01:00:00, 3.861t/s]
              Class    Images  Instances  Box(P      R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00:00:00, 3.121t/s]
              all      27      3194     0.559     0.47     0.471   0.293

Epoch      GPU_mem  box_loss  cls_loss  dfl_loss  Instances  Size
98/100      6.64G    2.034    1.445     1.01      388        1024: 100%|██████████| 7/7 [00:01:00:00, 3.951t/s]
              Class    Images  Instances  Box(P      R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00:00:00, 3.061t/s]
              all      27      3194     0.585     0.465   0.471   0.294

Epoch      GPU_mem  box_loss  cls_loss  dfl_loss  Instances  Size
99/100      6.64G    2.053    1.407     1.013     790        1024: 100%|██████████| 7/7 [00:01:00:00, 3.881t/s]
              Class    Images  Instances  Box(P      R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00:00:00, 3.021t/s]
              all      27      3194     0.565     0.40   0.469   0.293

Epoch      GPU_mem  box_loss  cls_loss  dfl_loss  Instances  Size
100/100     6.64G    2.076    1.442     1.009     740        1024: 100%|██████████| 7/7 [00:01:00:00, 3.851t/s]
              Class    Images  Instances  Box(P      R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00:00:00, 2.961t/s]
              all      27      3194     0.506     0.459   0.465   0.293

100 epochs completed in 0.084 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.39M
Optimizer stripped from runs/detect/train/weights/best.pt, 6.39M

Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.124 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (Tesla P100-PCIE-16GB, 16289MiB)
Model summary (fused): 72 layers, 3,086,623 parameters, 0 gradients, 8.1 GFLOPs

```

Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
all	27	3194	0.561	0.466	0.47	0.293
MA	27	662	0.483	0.9888	0.165	0.9562
HE	27	526	0.409	0.388	0.339	0.125
EX	27	1941	0.45	0.327	0.308	0.12
SE	14	38	0.495	0.525	0.544	0.271
OD	27	27	0.969	1	0.995	0.992

```

/usr/local/lib/python3.11/dist-packages/matplotlib/colormaps.py:721: RuntimeWarning: Invalid value encountered in less
sa[sa < 0] = -1

```

Figure 19 DR Segmentation(training)

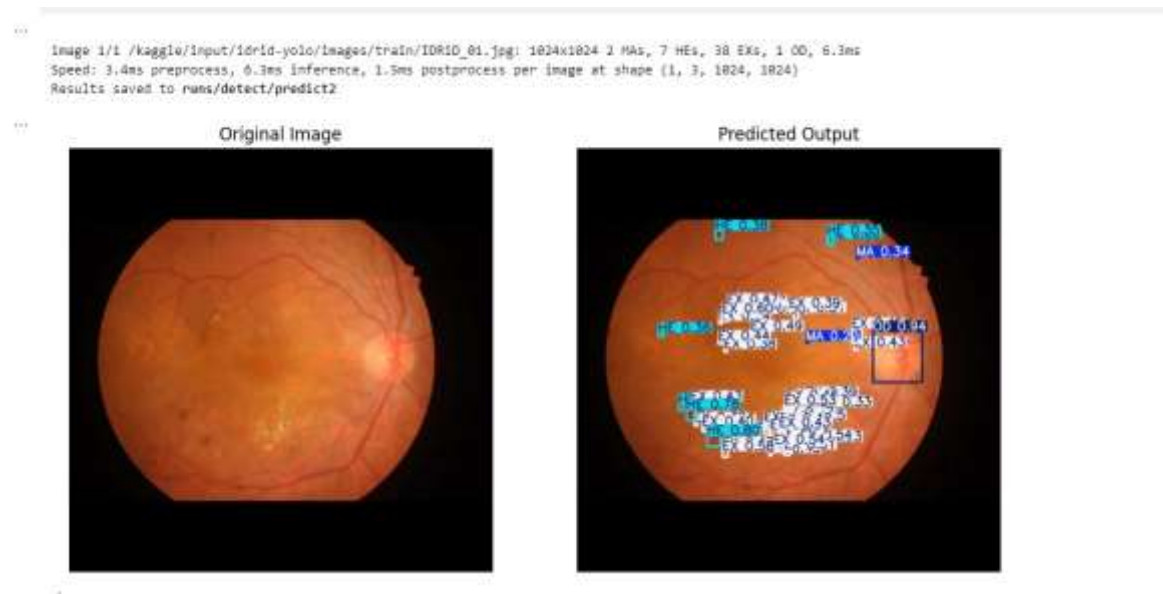


Figure 20 DR Segmentation(original and segmentation)

```

/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in a future release.
warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet18-f37072f9.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f37072f9.pth
100%|#####| 44.7M/44.7M [00:00:00.00, 266MB/s]
[I 2025-05-04 21:07:26.154] Trial 0 finished with value: 0.5025 and parameters: {'lr': 0.0032927133365774825, 'momentum': 0.8221399090658457}. Best is trial 0 with value: 0.5025.
[I 2025-05-04 21:09:52.237] Trial 1 finished with value: 0.5625 and parameters: {'lr': 0.00824576815958914763, 'momentum': 0.8134772156883706}. Best is trial 0 with value: 0.5025.
[I 2025-05-04 21:12:21.940] Trial 2 finished with value: 0.5025 and parameters: {'lr': 0.009766224889857e-05, 'momentum': 0.548790849063175}. Best is trial 0 with value: 0.5025.
[I 2025-05-04 21:14:52.557] Trial 3 finished with value: 0.71875 and parameters: {'lr': 1.3141084466514035e-05, 'momentum': 0.9386546482589577}. Best is trial 0 with value: 0.5025.
[I 2025-05-04 21:17:23.276] Trial 4 finished with value: 0.05025 and parameters: {'lr': 1.1286397493762622e-05, 'momentum': 0.5853888832072595}. Best is trial 0 with value: 0.5025.
[I 2025-05-04 21:19:47.413] Trial 5 finished with value: 0.5625 and parameters: {'lr': 9.043166348304138e-05, 'momentum': 0.7850067581040375}. Best is trial 0 with value: 0.5025.
[I 2025-05-04 21:22:17.500] Trial 6 finished with value: 0.875 and parameters: {'lr': 1.2758628985897783e-05, 'momentum': 0.6382059559670804}. Best is trial 0 with value: 0.5025.
[I 2025-05-04 21:25:02.100] Trial 7 finished with value: 0.50375 and parameters: {'lr': 0.000567110683248503, 'momentum': 0.6324433482507372}. Best is trial 0 with value: 0.5025.
[I 2025-05-04 21:27:26.772] Trial 8 finished with value: 0.6875 and parameters: {'lr': 0.0001404140049050225, 'momentum': 0.0900585335629432}. Best is trial 0 with value: 0.5025.
[I 2025-05-04 21:30:01.765] Trial 9 finished with value: 0.6875 and parameters: {'lr': 1.870487534366809e-05, 'momentum': 0.6197874432573502}. Best is trial 0 with value: 0.5025.

study_densenet = optuna.create_study()
study_densenet.optimize(objective_densenet, n_trials=10)

```

Figure 21 Calculating the best Parameter for Densenet 121

```

/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in a future release.
warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/densenet121-a63ec97.pth" to /root/.cache/torch/hub/checkpoints/densenet121-a63ec97.pth
100%|#####| 30.8M/30.8M [00:00:00.00, 137MB/s]
[I 2025-05-04 21:13:12.819] Trial 0 finished with value: 0.40025 and parameters: {'lr': 0.004555429552554882, 'momentum': 0.8284606615032872}. Best is trial 0 with value: 0.40025.
[I 2025-05-04 21:16:22.974] Trial 1 finished with value: 0.375 and parameters: {'lr': 0.0031206994447859464, 'momentum': 0.5488380599551422}. Best is trial 1 with value: 0.375.
[I 2025-05-04 21:19:34.697] Trial 2 finished with value: 0.4375 and parameters: {'lr': 0.004267413690692877, 'momentum': 0.7602288348358663}. Best is trial 1 with value: 0.375.
[I 2025-05-04 21:42:46.777] Trial 3 finished with value: 0.5 and parameters: {'lr': 0.0038133610082374385, 'momentum': 0.8870605541472170}. Best is trial 1 with value: 0.375.
[I 2025-05-04 21:45:58.881] Trial 4 finished with value: 0.05025 and parameters: {'lr': 0.0009441480200074323, 'momentum': 0.8913777944257345}. Best is trial 1 with value: 0.375.
[I 2025-05-04 21:49:07.835] Trial 5 finished with value: 0.53125 and parameters: {'lr': 0.0001040095852635593, 'momentum': 0.5824423743330858}. Best is trial 1 with value: 0.375.
[I 2025-05-04 21:52:19.770] Trial 6 finished with value: 0.40625 and parameters: {'lr': 0.0036041271687067508, 'momentum': 0.9728187662131223}. Best is trial 1 with value: 0.375.
[I 2025-05-04 21:55:30.883] Trial 7 finished with value: 0.46875 and parameters: {'lr': 0.003258071735004863, 'momentum': 0.8899112115077075}. Best is trial 1 with value: 0.375.
[I 2025-05-04 21:58:39.481] Trial 8 finished with value: 0.53125 and parameters: {'lr': 0.000140408409312795574, 'momentum': 0.7350068280102025}. Best is trial 1 with value: 0.375.
[I 2025-05-04 22:01:52.420] Trial 9 finished with value: 0.40025 and parameters: {'lr': 0.0007473219223611253, 'momentum': 0.958550499093386}. Best is trial 1 with value: 0.375.

```

Figure 22 Calculating the best Parameter for Resnet 50

```


best_resnet_params = study_resnet.best_params
best_densenet_params = study_densenet.best_params

print("Best parameters for ResNet:", best_resnet_params)
print("Best parameters for DenseNet:", best_densenet_params)

```

Best parameters for ResNet: {'lr': 0.0032927133365774825, 'momentum': 0.8221399090658457}  
Best parameters for DenseNet: {'lr': 0.0032386994447859464, 'momentum': 0.5488380599551422}

Figure 23 Selecting the best Parameter for Resnet 50 and Densenet 121

 Classification Report:

	precision	recall	f1-score	support
Mild	0.64	0.57	0.60	37
Moderate	0.77	0.86	0.81	99
No_DR	0.98	0.98	0.98	180
Proliferate_DR	0.71	0.59	0.64	29
Severe	0.53	0.42	0.47	19
accuracy			0.85	364
macro avg	0.72	0.68	0.70	364
weighted avg	0.84	0.85	0.84	364

Figure 24 Classification Report for Densnet 121+GLCM



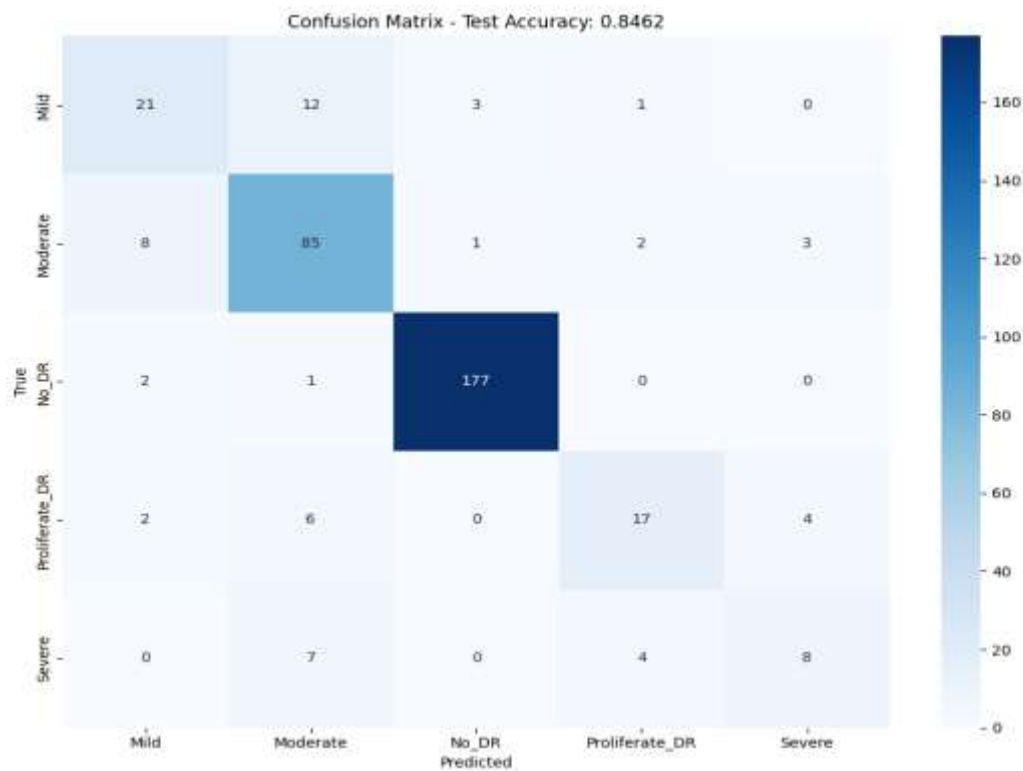


Figure 25 Confusion Matrix of Densenet121+GLCM

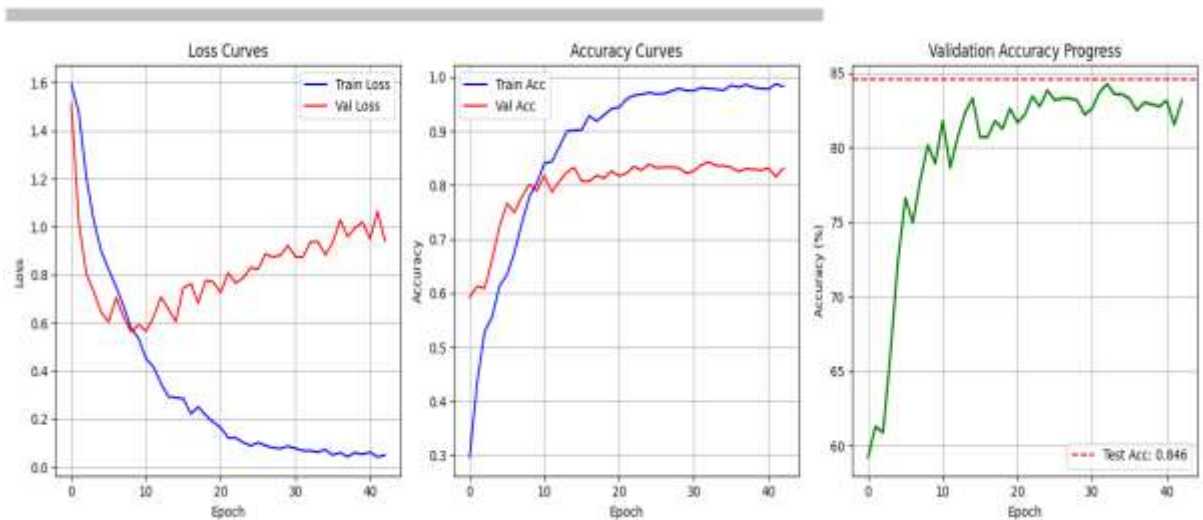



Figure 26 Loss, Accuracy and Validation Curve of Densenet121+GLCM

 Classification Report:

	precision	recall	f1-score	support
Mild	0.54	0.68	0.60	37
Moderate	0.75	0.81	0.78	99
No_DR	0.98	0.98	0.98	180
Proliferate_DR	0.74	0.69	0.71	29
Severe	0.67	0.21	0.32	19
accuracy			0.84	364
macro avg	0.74	0.67	0.68	364
weighted avg	0.84	0.84	0.83	364

Figure 27 Classification Report for Resnet 50+GLCM

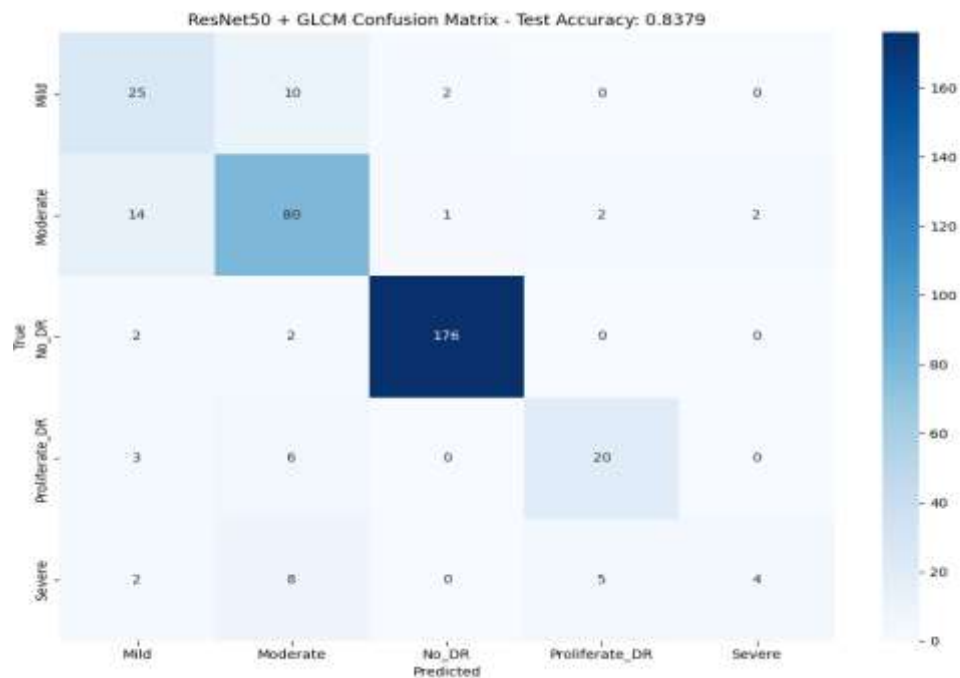


Figure 28 Confusion Matrix of ResNet50+GLCM

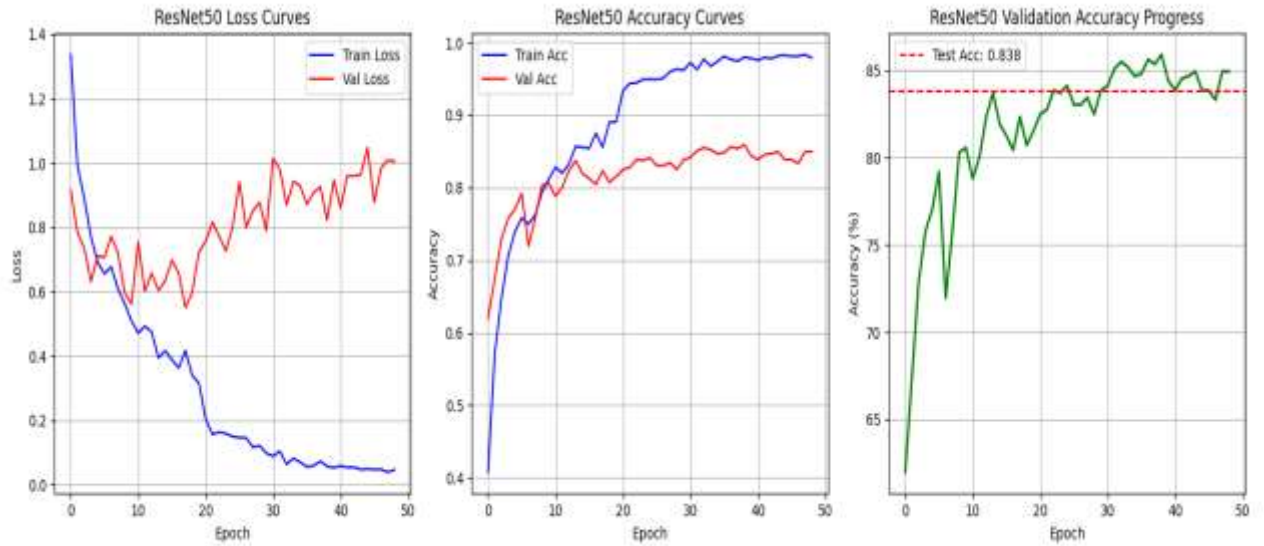



Figure 29 Loss, Accuracy and Validation Curve of ResNet50+GLCM

 Classification Report:

	precision	recall	f1-score	support
Mild	0.53	0.62	0.57	37
Moderate	0.77	0.76	0.77	99
No_DR	0.96	0.99	0.98	180
Proliferate_DR	0.62	0.45	0.52	29
Severe	0.29	0.26	0.28	19
accuracy			0.81	364
macro avg	0.64	0.62	0.62	364
weighted avg	0.81	0.81	0.81	364

Figure 30 Confusion Matrix of Effcient+GLCM

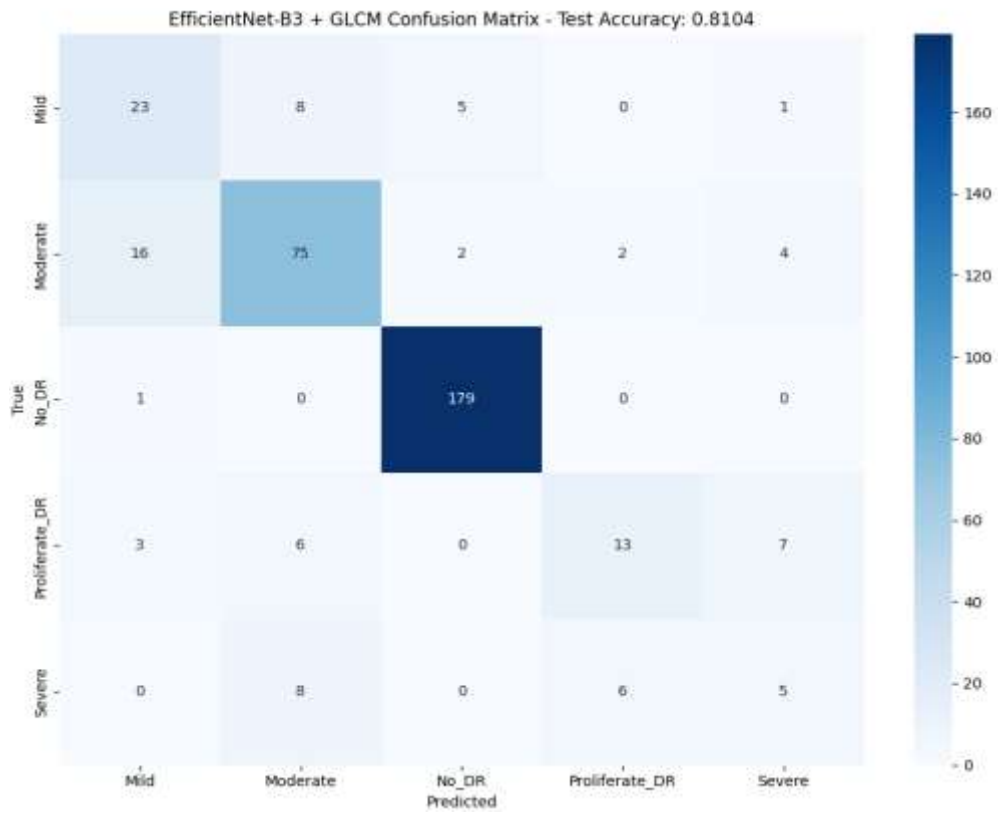


Figure 31 Confusion Matrix of EfficientNet-B3+GLCM

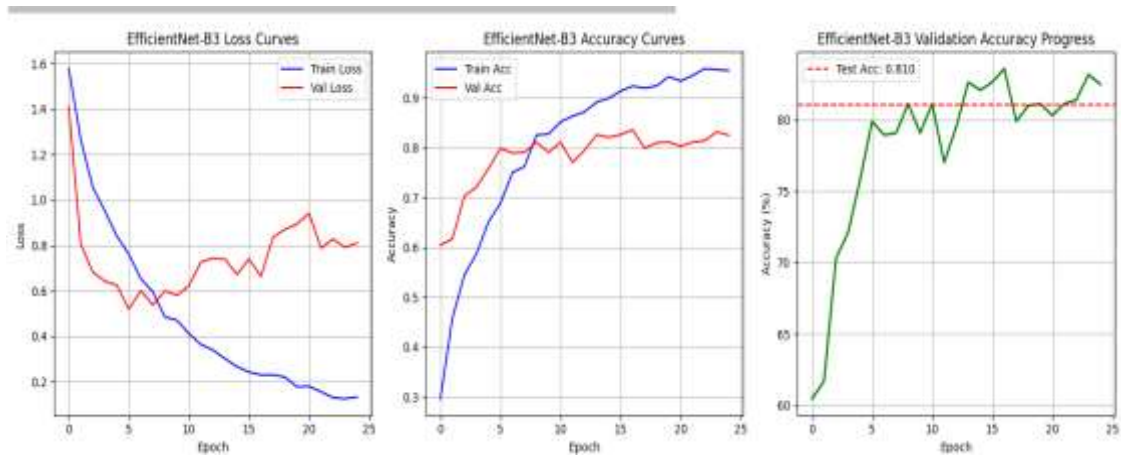


Figure 32 Loss, Accuracy and Validation Curve of EfficientNet-B3+GLCM

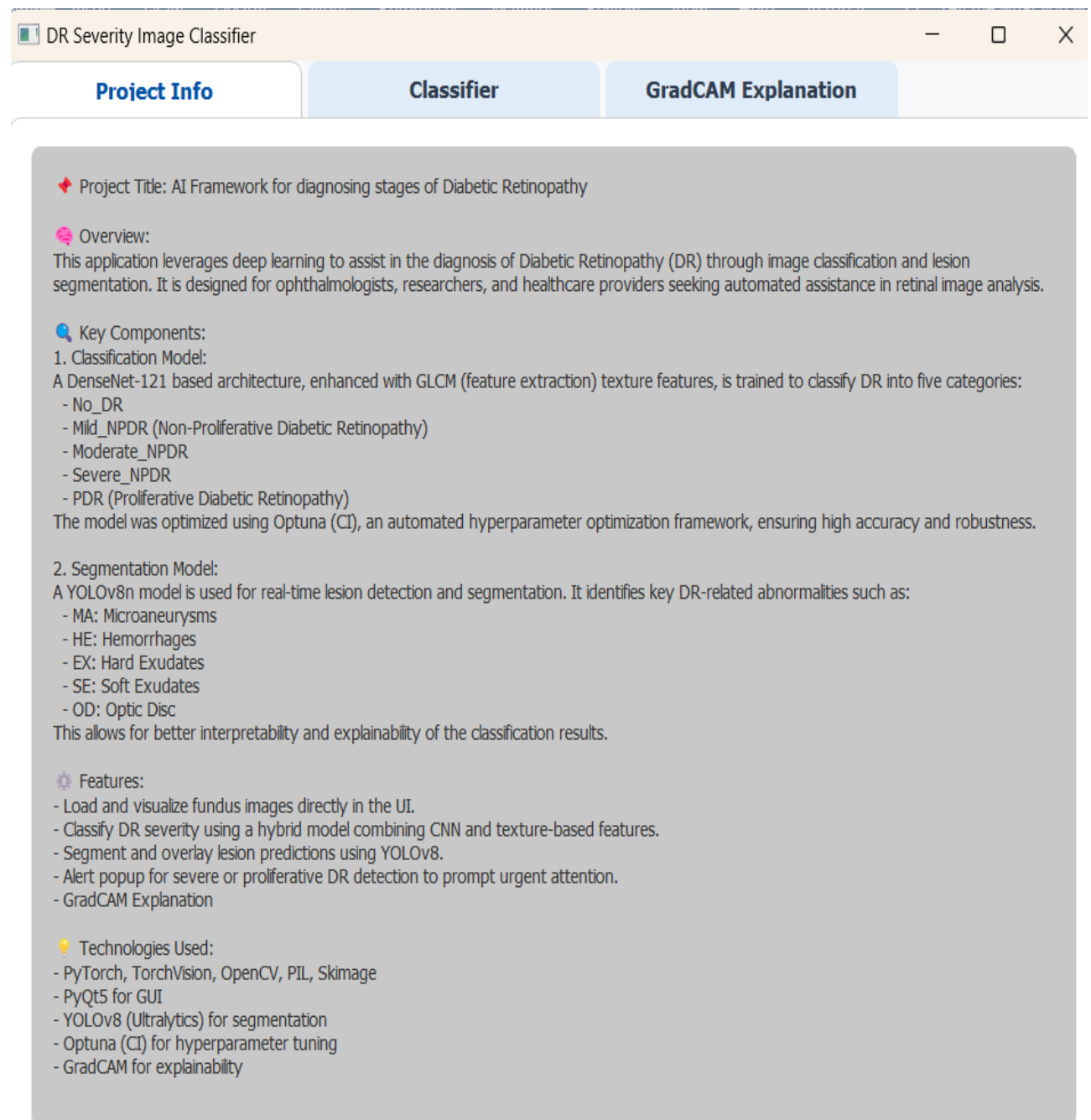


Figure 33: UserInterface1

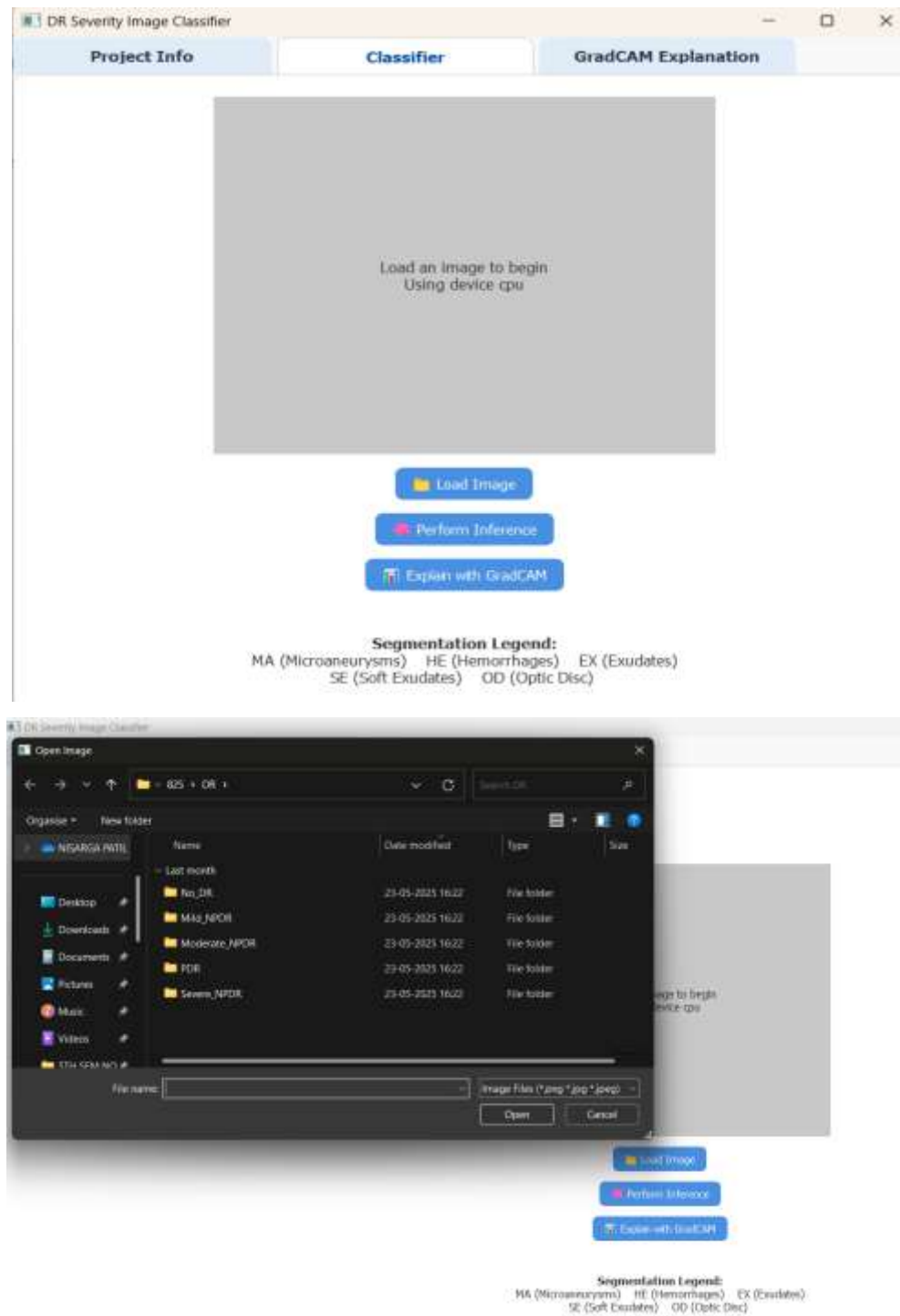


Figure 34: UserInterface2

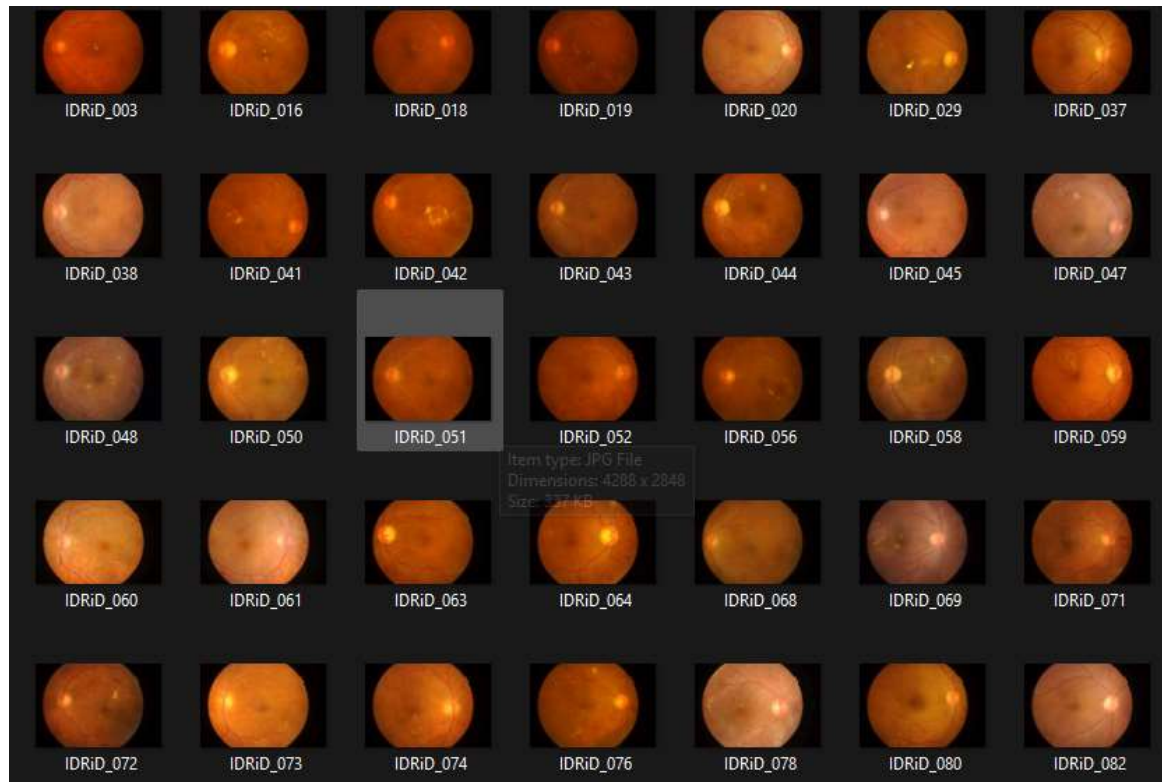


Figure 35: UserInterface3

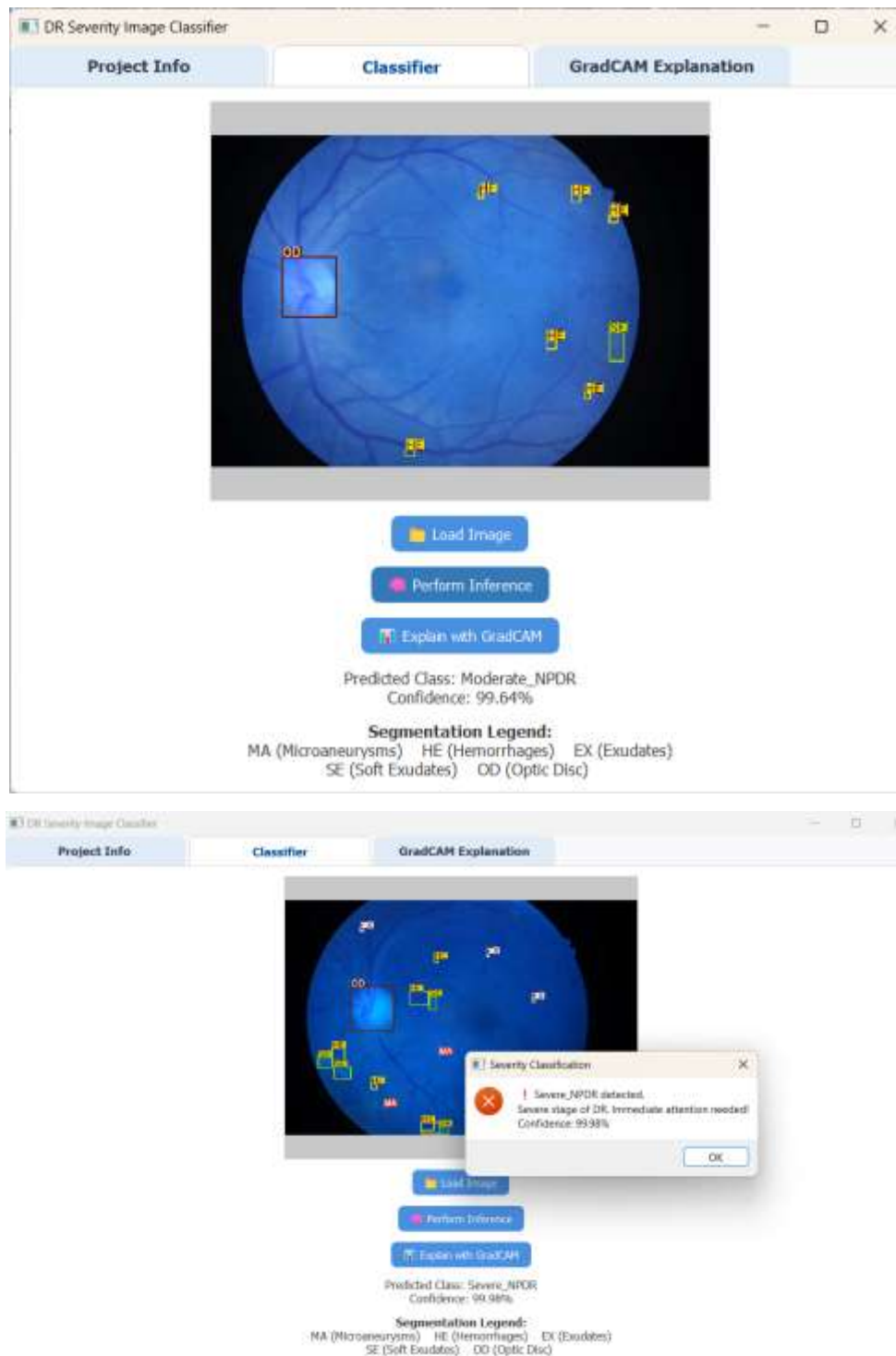


Figure 36: UserInterface4



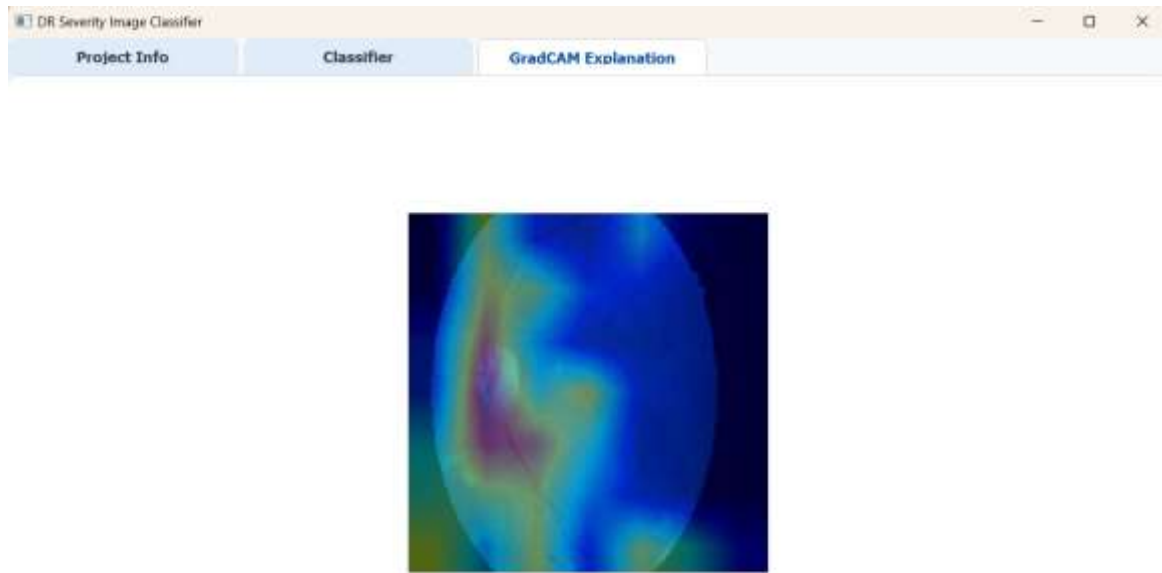


Figure 37: UserInterface4

## 6. Conclusions and Future Directions

---

### 6.1 Conclusions

The project was created to demonstrate the utility of a deep learning-based diagnostic tool for the early detection and classification of Diabetic Retinopathy (DR), a classical retinal disease that can be vision threatening in the worst case. The system is capable of providing reliable, real-time analysis of retinal image for DR by utilizing advanced lesion segmentation models such as YOLOv8, and CNN based predictive classifiers for predicting DR stage. The modelling uses Explainable AI approaches some of which examples are Grad-CAM, which provide an interpretable and transparent rationale for the outputs supplied by the system, which in a medical context, is of extreme importance. The application is desktop based, developed in PyQt5 and Python, and provides an easy to use graphical interface, which means a user can upload fundus images in many different formats or resolutions, and receive thorough visual rationalizations, with a diagnostic output. Based on the deployment and performance of project defined above, it can be concluded that it is a valid clinical decision-support system to assist the ophthalmologist and healthcare sectors in the early detection and ongoing monitoring of diabetic retinopathy.

### 6.2 Suggestions for Future Work

- Integration with EHR systems: Linking the app to electronic health record systems would help facilitate a patient's diagnostic history and monitor overtime.
- Fine-tuning on larger, diverse datasets: Fine-tuning the models on other larger, more diverse datasets (i.e., Messidor or EyePACS) will help diversify the generalizability of the models and increase accuracy across a more diverse demographic base.
- Web and mobile application extensions: Extending the current desktop application into web and mobile applications would help better reach users and practitioners working in locations that are remote or resource-limited.
- Automated DR progression tracking: Developing a temporal model to help track the progression of disease based on all patient images taken on multi visits.

- Multilingual and voice-based instructions: Adding support for more regional languages as well as voice-based instructions would increase applicability for the user interacting with the mobile application, especially for older adults and/or visually impaired end-users.
- Integration with real-time cloud-based imaging: As a future integration of real-time image analysis APIs run on cloud-based computing (e.g. Microsoft Azure or AWS) to develop scalable cloud-based workflows.

## 7. References

---

- [1] **Bansode, A., et al.** (2023) Efficient Deep Learning Architectures for Medical Image Analysis: A Review. *Journal of Medical Imaging and Health Informatics*
- [2] **Bhardwaj, S., et al.** (2020) Detection of diabetic retinopathy using image processing techniques. *Procedia Computer Science*.
- [3] **Chen, Z., et al.** (2023) Vision Transformers for Retinal Disease Detection: A Survey. *Artificial Intelligence in Medicine*.
- [4] **Gulshan, V., et al.** (2022) Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22), pp. 2402–2410.
- [5] **Khan, S.H., et al.** (2023) DenseNet-based classification of diabetic retinopathy from retinal images. *Biomedical Signal Processing and Control*..
- [6] **Kumar, A., et al.** (2024) Hybrid deep learning architecture for diabetic retinopathy detection. *Health Information Science and Systems*.
- [7] **Mohanty, C., et al.** (2023) AI-driven DR detection using ResNet and MobileNet. *Computers in Biology and Medicine*.
- [8] **Mostafa, F., et al.** (2025) Limitations of AI Models in Early DR Detection. *International Journal of Computer Assisted Radiology and Surgery*.
- [9] **Ram, K., et al.** (2021) Machine learning-based methods for diabetic retinopathy detection: A review. *Computer Methods and Programs in Biomedicine*.



Page 1 of 39 - Cover Page

Submission ID trn:aid::1:3280220451

## Student 2

### plagrism (2).docx

8th Sem 2021 Batch

Mihir Paul

M. S. Ramaiah University Of Applied Sciences

#### Document Details

Submission ID

trn:aid::1:3280220451

Submission Date

Jun 19, 2025, 12:10 PM GMT+5:30

Download Date

Jun 19, 2025, 12:11 PM GMT+5:30

File Name

plagrism\_2\_.docx

File Size

1.7 MB

34 Pages


7,674 Words

45,579 Characters



Page 1 of 39 - Cover Page

Submission ID trn:aid::1:3280220451

Page 2 of 39 - Integrity OverviewSubmission ID trmcd::1.3280220451





## 9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




### Filtered from the Report

► Bibliography

#### Match Groups


-  **46 Not Cited or Quoted 9%**  
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**  
Matches that are still very similar to source material
-  **0 Missing Citation 0%**  
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

#### Top Sources

- 3%  Internet sources
- 3%  Publications
- 5%  Submitted works (Student Papers)


#### Integrity Flags

1 Integrity Flag for Review





-  **Hidden Text**  
70 suspect characters on 1 page  
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.




A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.


Page 3 of 39 - Integrity Overview
Submission ID trrcoid::1-3280220451

### Match Groups

- 
**46 Not Cited or Quoted 9%**  
Matches with neither in-text citation nor quotation marks
- 
**0 Missing Quotations 0%**  
Matches that are still very similar to source material
- 
**0 Missing Citation 0%**  
Matches that have quotation marks, but no in-text citation
- 
**0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

### Top Sources


- 3%  Internet sources
- 3%  Publications
- 5%  Submitted works (Student Papers)

---

### Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	M S Ramaiah University of Applied Sciences	3%
2	Internet	link.springer.com	<1%
3	Internet	www.rapidinnovation.io	<1%
4	Student papers	University of Wales Institute, Cardiff	<1%
5	Student papers	Queensland University of Technology	<1%
6	Internet	elitca.org	<1%
7	Internet	web.realinfo.tv	<1%
8	Publication	Shashi Kant Dargar, Shilpi Birla, Abha Dargar, Avtar Singh, D. Ganeshaperumal. "...	<1%
9	Publication	Corceiro, Ana Catarina Antunes. "Artificial Intelligence-Powered Classification of ...	<1%
10	Student papers	Gitam University	<1%


Page 3 of 39 - Integrity Overview
Submission ID trrcoid::1-3280220451

turnitin

Page 4 of 33 - Integrity Overview

Submission ID (mset): 1:3280220451


11	Student papers	IFIM Business School	<1%
12	Student papers	University of Alabama at Birmingham	<1%
13	Internet	www.arxiv-vanity.com	<1%
14	Publication	"Image Analysis and Recognition", Springer Science and Business Media LLC, 2020	<1%
15	Publication	Anil Kumar Bondala, Kranthi Kumar Lella. "Revolutionizing diabetic retinopathy d...	<1%
16	Student papers	British University in Egypt	<1%
17	Student papers	Universiti Kebangsaan Malaysia	<1%
18	Student papers	University of Leeds	<1%
19	Internet	sietjournals.com	<1%
20	Internet	www.geeksforgeeks.org	<1%
21	Internet	www.jetir.org	<1%
22	Internet	theacademic.in	<1%
23	Internet	www.coursehero.com	<1%
24	Internet	www.ir.juit.ac.in:8080	<1%

turnitin


Page 4 of 33 - Integrity Overview

Submission ID (mset): 1:3280220451




Page 5 of 39 - Integrity Overview
Submission ID: 0m3d1:13285220451

25	Internet	www.medicaltourism.com	<1%
26	Publication	Carlos Santos, Marilton Aguiar, Daniel Welfer, Marcelo Dias, Alejandro Pereira, M...	<1%
27	Publication	Peddapullaiahgari Hariobulesu, Fahimuddin Shaik. "Enhanced multi-grade diabet...	<1%
28	Publication	Fatema Tujjohra, Md. Mazharul Islam, Taslim Ur Rashid, Mohammed Mizanur Rah...	<1%
29	Publication	Krishna Kant Singh, Vibhav Kumar Sachan, Akansha Singh, Sanjeevikumar Padma...	<1%
30	Internet	dialnet.unirioja.es	<1%
31	Internet	ierjournal.org	<1%
32	Internet	sos-vo.org	<1%
33	Internet	www.ncbi.nlm.nih.gov	<1%
34	Publication	Federico Rissotto, Emanuele Fusi, Luca Bonagura, Stefano Lingardo et al. "Inflam...	<1%
35	Publication	Uzair Aslam Bhatti, Jingbing Li, Mengxing Huang, Sibghat Ullah Bazai, Muhamma...	<1%


Page 5 of 39 - Integrity Overview
Submission ID: 0m3d1:13285220451