DETECTION OF DIABETIC RETINOPATHY USING MACHINE LEARNING

An Industrial Oriented Mini Project Report

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

In partial fulfillment of the requirements for the

Award of the degree of

BACHELOR OF TECHNOLOGY

in

CSE (DATA SCIENCE)

by

D.V.S. MALLIKARJUN	**(22VE1A6713)**
M. ARAVIND KUMAR	**(22VE1A6736)**
P. PRANITH REDDY	**(22VE1A6748)**
R. AKHIL SAI	**(22VE1A6749)**

Under the Guidance

of

Mrs. M. SHRAVANI

ASSISTANT PROFESSOR

SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF CSE (DATA SCIENCE)

(Affiliated to JNTUH, Approved by A.I.C.T.E and Accredited by NAAC, New Delhi)

Bandlaguda, Beside Indu Aranya, Nagole, Hyderabad-500068,Ranga Reddy Dist.

2022-2026

SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the Industrial Oriented Mini Project Report on *"DETECTION OF DIABETIC RETINOPATHY USING MACHINE LEARNING"* submitted by **D.V.S. Mallikarjun, M. Aravind Kumar, P. Pranith Reddy, R. Akhil Sai** bearing Hall ticket Numbers. **22VE1A6713, 22VE1A6736, 22VE1A6748, 22VE1A6749** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **CSE (DATA SCIENCE)** from Jawaharlal Nehru Technological University, Kukatpally, Hyderabad for the academic year 2024-2025 is a record of bonafide work carried out by them under our guidance and Supervision.

Internal Guide
Mrs. M. Shravani
Assistant Professor

Head of the Department
Dr. K. Rohit Kumar
Associate Professor

Project Co-Ordinator
Mrs. D. Chaithanya
Assistant Professor

External Examiner

SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF CSE (DATA SCIENCE)

DECLARATION

We **D.V.S. Mallikarjun, M. Aravind Kumar, P. Pranith Reddy, R. Akhil Sai** bearing Hall ticket Nos. **22VE1A6713, 22VE1A6736, 22VE1A6748, 22VE1A6749** hereby declare that the Industry Oriented Mini Project title **DETECTION OF DIABETIC RETINOPATHY USING MACHINE LEARNING** done by us under the guidance of **Mrs. M. Shravani, Assistant Professor** which is submitted in the partial fulfillment of the requirement for the award of the B. Tech degree in **CSE(Data Science)** at **Sreyas Institute of Engineering & Technology** for Jawaharlal Nehru Technological University, Hyderabad is our original work.

D.V.S. MALLIKARJUN	**22VE1A6713**
M. ARAVIND KUMAR	**22VE1A6736**
P. PRANITH REDDY	**22VE1A6748**
R. AKHIL SAI	**22VE1A6749**

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

We take this opportunity to acknowledge with thanks and deep sense of gratitude to **Mrs. M. Shravani, Assistant Professor, Department of CSE (Data Science)** for her constant encouragement and valuable guidance during the Project work.

A Special vote of Thanks to **Mrs. D. Chaithanya, Project Co-Ordinator and Dr. K. Rohit Kumar, Head of the Department**. who has been a source of Continuous motivation and support. They had taken time and effort to guide and correct me all through the span of this work.

We owe very much to the **Department Faculty, Principal** and **Management** who made my term at Sreyas a stepping stone for my career. We treasure every moment we had spent in the college.

Last but not the least, our heartiest gratitude to our **parents and fellow students** for their continuous encouragement. Without their support this work would not have been possible.

D.V.S. MALLIKARJUN 22VE1A6713
M. ARAVIND KUMAR 22VE1A6736
P. PRANITH REDDY 22VE1A6748
R. AKHIL SAI 22VE1A6749

ABSTRACT

Diabetic Retinopathy (DR) is a progressive eye disease and one of the primary causes of vision impairment and blindness among individuals with diabetes globally. Timely detection and accurate classification of DR are essential for initiating effective treatment and preventing irreversible vision loss. This research presents a deep learning-based solution for the automated detection of DR using retinal fundus images. The approach employs transfer learning with a pre-trained VGG16 convolutional neural network, which is fine-tuned on a publicly available Kaggle dataset comprising retinal images labeled according to five levels of DR severity.

The implementation is carried out using TensorFlow to facilitate efficient model training and inference. To enhance accessibility and usability, the trained model is integrated into a web-based application developed with the Flask framework, allowing users to upload retinal images and obtain diagnostic predictions in real time. Model evaluation is performed using precision, recall, and F1-score metrics to address the challenge of class imbalance and ensure clinically relevant performance.

This work underscores the potential of transfer learning in the domain of medical image classification and contributes a practical, deployable tool that can support ophthalmologists and healthcare providers in the early diagnosis and management of diabetic retinopathy.

KEYWORDS:

Diabetic Retinopathy (DR), Vision Impairment, Deep Learning, Retinal Fundus Images, VGG16, Convolutional Neural Network (CNN), Transfer Learning, Kaggle Dataset, Image Classification, TensorFlow, Model Training, Inference, Flask Web Framework, Real-time Prediction, Precision, Recall, F1-score, Class Imbalance, Medical Image Analysis, Early Diagnosis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF OUTPUT SCREENSHOTS

CHAPTER 1
INTRODUCTION

An innovative convergence between healthcare and artificial intelligence is the detection of diabetic retinopathy by machine learning. The early diagnosis of a diabetes-related problem that poses a serious threat to vision is revolutionized by this method, which harnesses the power of cutting-edge algorithms and enormous databases of retinal pictures. The diligent collection of large datasets, frequently in partnership with healthcare institutions, that include high-resolution retinal images produced using specialist imaging technology such as fundus cameras, is the first step in the process. These photos go through extensive preprocessing, such as resizing, cropping, and contrast modifications, to improve their quality while removing noise that can impede precise analysis. The feature extraction process is at the core of the machine learning methodology. Complex aspects of the retinal pictures, including texture, colour, and form, are painstakingly retrieved, and employed as crucial markers for recognizing diabetic retinopathy's symptoms. Then, other machine learning techniques are used, including but not limited to convolutional neural networks (CNNs).

On labelled data, these algorithms receive considerable training to identify patterns and abnormalities suggestive of diabetic retinopathy. Iterative training involves fine-tuning to maximize the model's precision and propensity for prediction. Once trained, these models are skilled at separating retinal images into various degrees of diabetic retinopathy, from minor symptoms to severe cases that necessitate emergency care. Some devices can even spot related eye diseases, which helps ophthalmologists deliver all-encompassing therapy. Cross-validation techniques and separate test datasets are frequently used to rigorously validate and test the models to ensure their generalizability and dependability. This machine learning application has a significant effect on the actual world. When used in clinical settings, these models support healthcare professionals and are invaluable resources for diabetic retinopathy screening. They serve a crucial role in enabling prompt interventions, potentially reducing vision loss, and enhancing the general quality of life for those with diabetes by helping with early identification. This strategy also has the potential to boost healthcare effectiveness, cut costs, and increase access to screening services, especially in underprivileged communities where the prevalence of diabetic retinopathy is still high. However, this transformative strategy is not without its share of difficulties.

Model interpretability is still a challenging topic because deep learning algorithms' internal workings are frequently difficult to understand. A significant challenge is addressing problems like class imbalance when mild occurrences of diabetic retinopathy outnumber severe cases. Despite these difficulties, machine learning-based diabetic retinopathy detection has a very bright future. Existing models are being improved, their interpretability is being improved, and creative techniques to combine machine learning and telemedicine are being explored to reduce healthcare access gaps. In conclusion, the combination of machine learning with diabetic retinopathy detection has the potential to revolutionize healthcare by protecting people's priceless vision and enhancing their general well-being.

1.1 Motivation:

Diabetic Retinopathy (DR) is a leading cause of vision loss globally, especially among working-age adults. As the prevalence of diabetes continues to rise, early and accurate detection of DR becomes increasingly critical to prevent irreversible blindness. Traditionally, diagnosis relies on manual examination of retinal images by ophthalmologists — a time-consuming, subjective, and expertise-intensive process.

The motivation behind this project is to leverage the power of machine learning and deep learning to assist in the automated detection of DR from retinal images. Automating this process offers several benefits:

- **Scalability**: ML models can screen large populations quickly.
- **Consistency**: Reduces human error and diagnostic variability.
- **Accessibility**: Enables screening in remote or underserved regions lacking specialized care.

By developing a model that can identify early signs of DR, this project aims to support clinical workflows, enable timely treatment, and ultimately contribute to reducing vision impairment among diabetic patients. It represents a step toward more intelligent, accessible, and efficient healthcare systems.

1.2 Objective:

The primary objective of this project is to design and develop a machine learning-based system capable of automatically detecting and classifying the severity of Diabetic Retinopathy (DR) from retinal fundus images. Diabetic Retinopathy is a progressive eye disease associated with diabetes that can lead to permanent vision loss if not identified and treated early. Traditional diagnosis methods are highly dependent on expert ophthalmologists and manual image grading, which can be time-consuming, expensive, and prone to human error.

This project aims to address these challenges by building a deep learning model using Convolutional Neural Networks (CNNs) that can learn complex visual patterns directly from images. The model will be trained to classify retinal images into different stages of DR, enabling early detection and prioritization of cases that require medical attention.

Key objectives include:

- Collecting and preprocessing a dataset of retinal images filtered with Gaussian noise reduction techniques.
- Designing a robust CNN architecture that generalizes well across varying image conditions.
- Handling class imbalance using data augmentation and class weighting strategies.
- Evaluating model performance using appropriate metrics such as accuracy and the Quadratic Weighted Kappa score.
- Developing a user-friendly web application using Flask that allows users to upload images and receive predictions in real-time.

Ultimately, the goal is to build a reliable, scalable, and accessible tool that supports early DR detection, reduces the workload on medical professionals, and improves patient outcomes through timely diagnosis and intervention.

1.3 Scope:

This project focuses on developing an automated system for the detection and classification of Diabetic Retinopathy (DR) using machine learning techniques, specifically Convolutional Neural Networks (CNNs). The system is intended to assist in the early diagnosis of DR through the analysis of retinal fundus images, providing a scalable and efficient solution that can be used in both clinical and remote settings.

The scope includes several key components:

- **Data Handling:** Preprocessing of retinal images, including resizing, noise reduction (using Gaussian filters), normalization, and augmentation techniques to improve model robustness and accuracy.
- **Model Development:** Building and training a custom CNN model to classify images into different DR severity levels. The project initially supports classification into three categories — No_DR, Mild, and Moderate — based on available data. This can be extended to include more advanced stages (Severe and Proliferative) as additional labeled data becomes available.
- **Model Evaluation**: Employing evaluation metrics such as accuracy and Quadratic Weighted Kappa to assess the performance and reliability of the model, especially in handling imbalanced datasets.
- **Deployment:** Integrating the trained model into a Flask-based web application that allows users to upload retinal images and receive real-time predictions on the presence and severity of DR.
- **Real-world Application:** While not replacing ophthalmologists, the system can act as a screening tool to prioritize cases, especially in regions with limited access to healthcare professionals.

1.4 Outline:

This project, titled *"Detection of Diabetic Retinopathy Using Machine Learning,"* aims to automate the detection and classification of Diabetic Retinopathy (DR) stages using deep learning techniques. The project is structured into several well-defined phases, each addressing a critical component of the system.

- **Problem Understanding and Literature Review:** The project begins with studying the medical background of DR, existing diagnostic methods, and recent advancements in AI-based medical image classification.
- **Dataset Acquisition and Preprocessing:** Retinal fundus images are collected and filtered using Gaussian blurring techniques. Images are resized to 224×224 pixels and normalized to ensure uniform input to the model. Data augmentation is applied to address class imbalance and improve generalization.
- **Model Design and Training:** A Convolutional Neural Network (CNN) is developed using TensorFlow and Keras. The model is trained to classify images into different stages of DR (initially No_DR, Mild, and Moderate). Class weights and regularization techniques are used to handle data imbalance and prevent overfitting.
- **Model Evaluation:** The trained model is evaluated using metrics such as accuracy and the Quadratic Weighted Kappa score, which accounts for the severity of misclassifications in multi-class settings.
- **Web Application Development:** A user-friendly web interface is developed using Flask, enabling users to upload images and view classification results instantly.
- **Result Analysis and Conclusion:** The results are analyzed to assess the system's reliability and usability in real-world screening environments.

CHAPTER 2

LITERATURE SURVEY

1. **Diabetic Retinopathy Detection**

   Authors: V. Sudha, K. Priyanka, T. Suvathi Kannathal, S. Monisha

   Proposed Algorithm:

   The authors proposed a CNN-based classification method that automatically distinguishes between **normal, beginning, mild, and severe** stages of diabetic retinopathy. The CNN removes the need for handcrafted feature extraction and relies on learned features from fundus images.

   Results:

   Achieved **above 95% accuracy** in testing with **minimal overfitting**. The use of deep learning allows more consistent results compared to manual methods.

   Additional Info: Likely used data augmentation to improve generalization; real-time screening potential for rural healthcare.

2. **Detection of Diabetic Retinopathy Using Deep Learning Methodology**

   Authors: Gazala Mushtaq, Farheen Siddiqui

   Proposed Algorithm:

   Introduced a **DenseNet-169** architecture for classifying fundus images into **five severity stages** (No DR to Proliferative DR). Dense Net's skip connections help mitigate vanishing gradient problems and improve feature reuse.

   Results:

   DenseNet-169 achieved **90% accuracy**, outperforming classical models like SVM, Decision Tree, and KNN.

   Additional Info: Model performance suggests high potential for deployment in early DR screening tools; trained on publicly available datasets such as Eye PACS or DIARETDB.

3. **Diagnosis of Diabetic Retinopathy Using Machine Learning**

   Authors: Swati Gupta, Karandikar A.M.

   Proposed Algorithm:

   Employed traditional **machine learning classifiers** (SVM and KNN) with **image processing techniques** to identify DR in retinal images. Feature extraction possibly included blood vessel segmentation or texture analysis.

Results:

SVM delivered **86% classification accuracy**, outperforming KNN in sensitivity and specificity.

Additional Info: Demonstrates the potential of lightweight models for edge devices; useful in resource-limited screening setups.

4. **Identification of Diabetic Retinopathy Through Machine Learning**

   Authors: Malik Bader Alazzam, Fawaz Alassery, Ahmed Almulihi

   Proposed Algorithm:

   Used **Restricted Boltzmann Machines (RBM)** and **Optimum-Path Forest (OPF)** on labeled retinal scan data for classification. These models are suitable for unsupervised and supervised learning respectively.

   Results:

   RBM-1000 showed the best performance with **89.47% accuracy**.

   Additional Info: RBM captures complex visual features; OPF uses graph-based representation for classification. This hybrid technique is suitable for datasets with varying quality.

5. **Early Detection of Diabetic Retinopathy Based on Deep Learning and Ultra-Wide-Field Fundus Images**

   Authors: Kangrok Oh, Hae Min Kang, Dawoon Leem, Hyungyu Lee, KyoungYul Seo, Sangchul Yoon

   Proposed Algorithm:

   Focused on **Ultra-Wide-Field (UWF) fundus imaging**, covering **82% of the retina**.

   Combined this with deep learning models for enhanced early detection.

   Results:

   - **83.38% ± 0.47%** accuracy (ETDRS 7SF)
   - **80.60% ± 0.54%** accuracy (ETDRS F1-F2)

     Additional Info: UWF imaging enables detection of peripheral lesions often missed in standard imaging. Provides more comprehensive retinal assessment.

6. **Detection of Diabetic Retinopathy Using Machine Learning Algorithm**

   Authors: Sonali Chaudhary, Ramya H.R

   Proposed Algorithm:

Combined **image processing techniques** (possibly histogram equalization, edge detection) with **Fuzzy Logic** and **CNNs** to improve detection and classification.

Results:

- **Fuzzy Classifier**: 85% accuracy
- **CNN Classifier**: **90% accuracy**

  Additional Info: Fuzzy logic helps handle image ambiguity, while CNN captures visual patterns. Hybrid approach enhances robustness against poor image quality.

7. **Automated Detection of Diabetic Retinopathy Using Deep Learning**

   Authors: Carson Lam, MD, Darvin Yi, Margaret Guo, Tony Lindsey

   Proposed Algorithm:

   Applied **Google Net**, a deep CNN architecture, for classifying DR severity levels from fundus images. Google Net uses inception modules for multi-scale feature detection.

   Results:

   Achieved an accuracy of **96%**, indicating strong performance in classification.

   Additional Info: Emphasizes medical usability of transfer learning and pretrained models for rapid deployment in clinical settings.

8. **Diabetic Retinopathy Detection Based on Modified CNN Using Fundus Images**

   Authors: Awais Bajwa, Neelam Nosheen, Khalid Iqbal Talpur, Sheeraz Akram

   Proposed Algorithm:

   Evaluated **VGG16 and VGG19** for image feature extraction and classification. These models are known for their depth and simplicity, with stacked convolution layers.

   Results:

   VGG16 achieved best results with **97.14% accuracy** and **0.972 AUC**.

   Additional Info: Model was possibly fine-tuned on a pre-trained ImageNet version and adapted for fundus image input. High performance implies strong generalization and low false negatives.

9. **Diabetic Retinopathy Detection Through Deep Learning Techniques**

   Authors: Wejdan L. Alyoubi, Wafaa M. Shalash, Maysoon F. Abulkhair

   Proposed Algorithm:

   The paper reviewed various **deep learning techniques** for medical image analysis,

including segmentation, retrieval, and classification.

Results:

- **73%** of studies classified images into DR/Non-DR
- **27%** categorized them into multiple DR stages

Additional Info: Emphasized the role of **convolutional architectures** and datasets like Messidor, EyePACS. Highlights the research trend toward multi-class classification and improved diagnosis granularity.

10. A Hybrid Technique for Diabetic Retinopathy Detection Based on Ensemble-Optimized CNN and Texture Features

Authors: Uzair Ishtiaq, Erma Rahayu, Zubair Ishtiaque

Proposed Algorithm:

Introduced a hybrid model combining **ensemble CNN models** with **texture-based features** to improve classification reliability. Feature-level fusion was applied before classification.

Results:

Achieved **95.20% accuracy** using **10-fold cross-validation**, ensuring consistency across data splits.

Additional Info: Texture features capture microaneurysms and hemorrhages; ensemble method reduces overfitting and variance. Effective in real-world DR detection.

CHAPTER 3

PROPOSED SYSTEM

The proposed system which is deep CNN approach, a classificational tool helpful in identifying the disorder from digital fundus images with maximum accuracy. We progress a neural network which is Vgg-16 model, which can identify the intricate features involved in the classification task and consequently provide a diagnosis automatically. We'll be training this network using the Kaggle dataset and validate results, for high-level classification. Several machine learning methods, including KNN, SVM, Decision Tree, Naïve Bayes, Random Forests are utilized to classify the images using the retrieved features. According to the severity of diabetic retinopathy, the output is an integral value on a scale of 0 to 4.

Machine learning is essential for detecting diabetic retinopathy since it makes it possible to analyze retinal pictures automatically, aiding in the early detection and screening of the condition. Machine learning algorithms need pertinent features to provide precise predictions, which is known as feature extraction. Features that can be used to identify diabetic retinopathy include blood vessel anomalies, microaneurysms, exudates, hemorrhages, and characteristics of the optic disc. These properties may be automatically extracted from retinal pictures using machine learning approaches, such as convolutional neural networks (CNNs) or image processing algorithms. Convolutional Neural Networks (CNNs) are a kind of deep learning architecture that has excelled in classifying pictures, making them a good choice for studying retinal images. CNNs have proven to be extremely effective in detecting diabetic retinopathy, frequently outperforming or matching human specialists. As strong tools for the early diagnosis and screening of diabetic retinopathy, CNNs' capacity to automatically learn pertinent characteristics from raw pictures might improve patient outcomes and lighten the workload for medical practitioners.

The Convolutional Neural Network (CNN) architecture VGG16, which is often employed, has demonstrated promising performance in several computer vision applications, including the identification of diabetic retinopathy. Convolutional layers, pooling layers, and fully linked layers are among the 16 layers that make up the deep architecture of the VGG16. It can successfully learn complex characteristics from retinal pictures by utilizing its deep architecture and transfer learning capabilities. This helps to provide accurate and reliable diabetic retinopathy diagnosis and screening. Following the extraction of features, a variety of machine learning techniques may be used for categorization tasks. Decision trees, KNN, Voting Classifier, random forests, support vector

machines (SVMs), logistic regression, and Naïve Bayes are examples of common methods utilized in the identification of diabetic retinopathy. These algorithms can distinguish between retinal pictures with normal function and those with various stages or severity of diabetic retinopathy using labelled datasets.Using assessment measures including accuracy, precision, recall, and F1 score, a machine learning model's performance in detecting diabetic retinopathy is evaluated. These metrics give information on how well the algorithm can identify diabetic retinopathy and accurately classify retinal pictures. To get more reliable performance estimates, cross-validation techniques like k-fold cross-validation can be used.

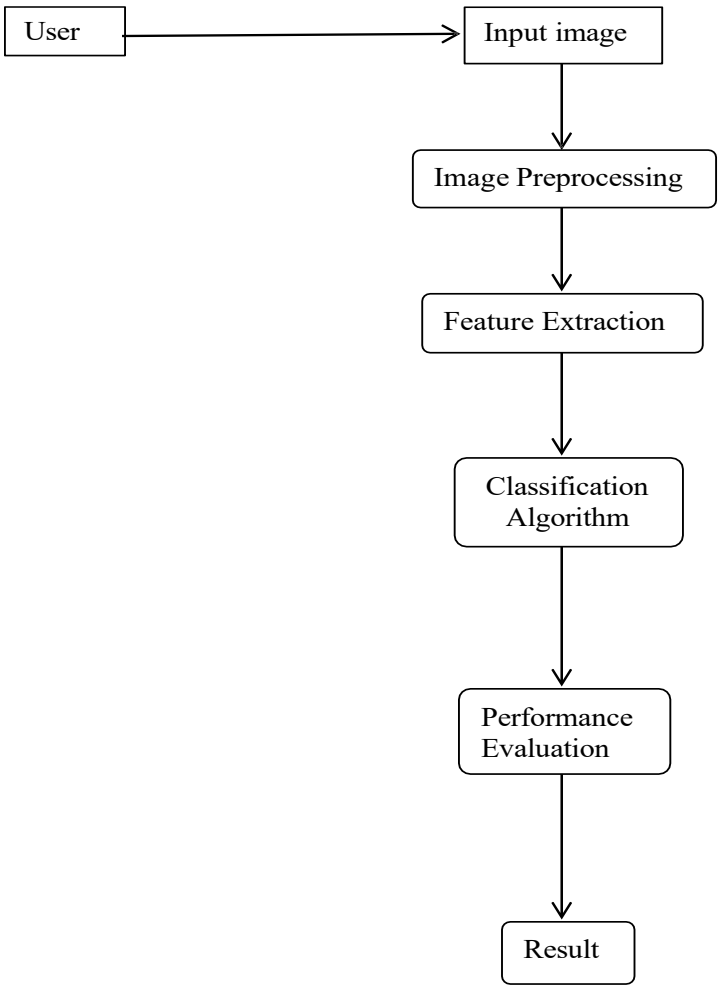

Fig-3.1 Flow Chart

1) **User:** The user uploads the retinal image to check the severity level of the disease.
2) **ImagePreprocessing:** Raw retinal images are being preprocessed to enhance their quality and prepare them for further analysis. Preprocessing techniques may include noise reduction, contrast adjustment, and image resizing.
3) **Feature Extraction:** Relevant features are extracted from the pre-processed images and are given to the 18 models to classify into their classes based on their severity level. It may involve extracting blood vessels, lesions, or other characteristic patterns from the retinal images.
4) **Classification Algorithm:** This process performs the actual classification of the extracted features to determine the presence and severity of diabetic retinopathy. It may use machine learning algorithms to classify the images into different categories based on the extracted features.
5) **Performance Evaluation:** Performance evaluation includes a number of crucial indicators that rate the algorithm's efficiency and precision. Some of the commonly used evaluation metrics are accuracy, sensitivity, specificity, F1 score and precision.

3.1 Existing System:

Diabetic Retinopathy (DR) detection has seen significant advancements with the integration of automated or computer-aided methods, particularly leveraging deep learning techniques. Manual diagnosis methods, such as pupil dilation, optical coherence tomography, biomicroscope, visual acuity test, fundoscopy, and retinography, have traditionally been employed. However, these methods are time-consuming, prone to misdiagnosis, and demand specialized training for accurate interpretation. The advent of Convolutional Neural Networks (CNNs) and, specifically, a refined ResNet50 model has revolutionized DR prediction. This deep learning approach addresses key challenges, including overfitting, vanishing gradients, loss value reduction, and fluctuation problems. The revised ResNet50 not only enhances accuracy but also streamlines the diagnostic process. The current drawbacks of manual diagnosis, namely its slow and labor-intensive nature, highlight the need for more efficient and automated approaches. Computer-aided methods, such as those employing CNNs, present a promising alternative by offering faster and potentially more accurate detection of diabetic retinopathy. These advancements signify a transformative shift towards leveraging technology to improve the speed, accuracy, and accessibility of diabetic retinopathy diagnosis, ultimately enhancing patient care and outcomes. The incorporation of deep learning models, such as the revised ResNet50, addresses these limitations by providing a more efficient and consistent diagnostic tool. By leveraging advanced

algorithms, these models can analyze large datasets of retinal images quickly and accurately, offering a potential solution to the time-consuming nature of manual diagnosis. Furthermore, the automated approach minimizes the risk of misdiagnosis associated with human interpretation.

3.2 Proposed System:

The proposed system which is deep CNN approach, a classificational tool helpful in identifying the disorder from digital fundus images with maximum accuracy. We progress a neural network which is Vgg-16 model, which can identify the intricate features involved in the classification task and consequently provide a diagnosis automatically. We'll be training this network using the Kaggle dataset and validate results, for high-level classification. Several machine learning methods, including KNN, SVM, Decision Tree, Naïve Bayes, Random Forests are utilized to classify the images using the retrieved features. According to the severity of diabetic retinopathy, the output is an integral value on a scale of 0 to 4.

Machine learning is essential for detecting diabetic retinopathy since it makes it possible to analyze retinal pictures automatically, aiding in the early detection and screening of the condition. Machine learning algorithms need pertinent features to provide precise predictions, which is known as feature extraction. Features that can be used to identify diabetic retinopathy include blood vessel anomalies, microaneurysms, exudates, hemorrhages, and characteristics of the optic disc. These properties may be automatically extracted from retinal pictures using machine learning approaches, such as convolutional neural networks (CNNs) or image processing algorithms. Convolutional Neural Networks (CNNs) are a kind of deep learning architecture that has excelled in classifying pictures, making them a good choice for studying retinal images. CNNs have proven to be extremely effective in detecting diabetic retinopathy, frequently outperforming or matching human specialists. As strong tools for the early diagnosis and screening of diabetic retinopathy, CNNs' capacity to automatically learn pertinent characteristics from raw pictures might improve patient outcomes and lighten the workload for medical practitioners.

The Convolutional Neural Network (CNN) architecture VGG16, which is often employed, has demonstrated promising performance in several computer vision applications, including the identification of diabetic retinopathy. Convolutional layers, pooling layers, and fully linked layers are among the 16 layers that make up the deep architecture of the VGG16. It can successfully learn complex characteristics from retinal pictures by utilizing its deep architecture and transfer learning

capabilities. This helps to provide accurate and reliable diabetic retinopathy diagnosis and screening. Following the extraction of features, a variety of machine learning techniques may be used for categorization tasks. Decision trees, KNN, Voting Classifier, random forests, support vector machines (SVMs), logistic regression, and Naïve Bayes are examples of common methods utilized in the identification of diabetic retinopathy. These algorithms can distinguish between retinal pictures with normal function and those with various stages or severity of diabetic retinopathy using labelled datasets.Using assessment measures including accuracy, precision, recall, and F1 score, a machine learning model's performance in detecting diabetic retinopathy is evaluated. These metrics give information on how well the algorithm can identify diabetic retinopathy and accurately classify retinal pictures. To get more reliable performance estimates, cross-validation techniques like k-fold cross-validation can be used.

3.3 Software Requirement Specification

This project aims to develop a web-based application for the automated detection of Diabetic Retinopathy (DR) using a deep learning model trained on retinal images. The system will allow users to upload retinal fundus images and receive a classification of the DR severity.

1. Functional Requirements:

- **Image Upload Interface:** Users should be able to upload fundus images through the web interface.
- **Image Preprocessing:** The uploaded image is resized, normalized, and passed through a Gaussian blur before prediction.
- **Model Inference:** The system should classify images into DR severity stages (e.g., No_DR, Mild, Moderate).
- **Result Display:** The predicted class and confidence score should be displayed clearly to the user.
- **Error Handling:** The system must handle incorrect file types or corrupted images gracefully.

2. Non-Functional Requirements:

- **Usability:** The interface must be simple and user-friendly for medical professionals and non-technical users.
- **Performance:** Model inference should be performed within a few seconds of image upload.

- **Scalability:** The architecture should support future expansion to include more DR classes or additional eye diseases.
- **Maintainability:** The codebase should be modular and well-documented to facilitate updates or integration with other systems.

3. Software and Tools:

- **Frontend:** HTML, CSS (within Flask templates)
- **Backend:** Python with Flask
- **ML Framework:** TensorFlow, Keras
- **Libraries:** NumPy, OpenCV, Matplotlib, scikit-learn
- **Environment:** Python 3.7+, Web browser, Localhost or cloud hosting platform

3.4 SDLC Methodologies:

For the development of the project *"Detection of Diabetic Retinopathy Using Machine Learning,"* the Iterative SDLC model is the most suitable methodology. This approach allows for building the system incrementally, improving functionality with each iteration based on testing, feedback, and performance evaluation.

1. Requirement Analysis:

The initial phase involved identifying the functional and non-functional requirements of the system. The core objective was defined — to classify retinal images into different stages of Diabetic Retinopathy using machine learning.

2. System Design:

Based on the requirements, the system architecture was designed. This included selecting appropriate tools and frameworks such as TensorFlow, Keras, Flask, and OpenCV. A custom CNN architecture was chosen for image classification tasks.

3. Implementation:

In this phase, the model was developed and trained on preprocessed retinal images. Techniques like image augmentation, Gaussian filtering, and class weighting were used to improve model robustness and handle data imbalance.

4. Testing:

The trained model was tested using unseen validation data. Performance was evaluated using metrics like accuracy and Quadratic Weighted Kappa. The web interface was also tested for

usability and reliability.

5. Deployment and Maintenance:

The final model was integrated into a Flask-based web application. The system is designed to be maintainable and scalable, allowing for future improvements such as the inclusion of more DR classes or cloud-based deployment.

3.5 Functional Requirements:

Functional requirements define the specific behaviors, features, and functionalities that the system must perform. For the project *"Detection of Diabetic Retinopathy Using Machine Learning,"* the following core functional requirements are essential:

1. Image Upload Functionality:

The system must provide users with the ability to upload retinal fundus images via a web interface. It should support standard image formats such as JPEG, PNG, and JPG.

2. Image Preprocessing:

Once an image is uploaded, the system must automatically preprocess it. This includes resizing the image to a fixed dimension (e.g., 224x224 pixels), applying Gaussian filters to remove noise, and normalizing pixel values for consistent model input.

3. Model Prediction:

The preprocessed image is passed through a trained Convolutional Neural Network (CNN) model. The system must classify the image into one of the predefined DR severity categories — such as No_DR, Mild, or Moderate and return the result.

4. Displaying Results:

The system must display the classification result along with the confidence score. This output should be clear, informative, and easy to interpret by users without a technical background.

5. Error Handling:

The system must detect and report errors such as unsupported file formats, corrupted images, or missing inputs. It should also provide meaningful error messages to guide the user.

CHAPTER 4
SYSTEM DESIGN

Designing a system for the detection of diabetic retinopathy involves several components, including data acquisition, pre-processing, feature extraction, model training, and deployment. Here's a high-level overview of a system design for diabetic retinopathy detection using a Convolutional Neural Network (CNN):

1. Data Acquisition: Collect a diverse and representative dataset of retinal images with varying degrees of diabetic retinopathy.

2. Pre-processing: Resize images to a standard format suitable for the CNN architecture. Apply normalization and standardization techniques to ensure consistent input to the model.

3. Feature Extraction: Fine-tune the pre-trained model on the diabetic retinopathy dataset to adapt it to the specific characteristics of retinal images. Extract relevant features from the last few layers of the CNN architecture.

4. Model Training: Split the dataset into training, validation, and test sets.Train the model using the training set, validating its performance on the validation set. Optimize hyperparameters, such as learning rate and dropout rate, to improve the model's generalization capability.

5. Model Evaluation: Assess the model's performance on the test set using evaluation metrics like accuracy, precision, recall, and F1 score.Consider using a confusion matrix to understand the model's ability to classify different stages of diabetic retinopathy.

6. Deployment: Integrate the trained model into a user-friendly interface for deployment. Develop a secure and scalable system that allows for easy interaction with healthcare professionals.

4.1 Importance of Design:

The importance of system design in the development of a diabetic retinopathy detection system cannot be overstated, as it directly influences the accuracy, efficiency, and reliability of diagnostic outcomes. A well-crafted system design serves as the foundation for seamless integration of advanced technologies, such as Convolutional Neural Networks (CNNs), into the healthcare domain. The careful selection of components, from data acquisition and pre-processing to model training and deployment, ensures that the system not only effectively addresses the

complexities of diabetic retinopathy but also aligns with clinical requirements and standards. Properly designed systems optimize the utilization of diverse and labeled datasets, leveraging pre-trained models for feature extraction, and allowing for fine-tuning to adapt to specific medical conditions. The design also incorporates critical aspects such as interpretability and explainability, providing healthcare professionals with insights into the model's decision-making process.

4.2 System Architecture

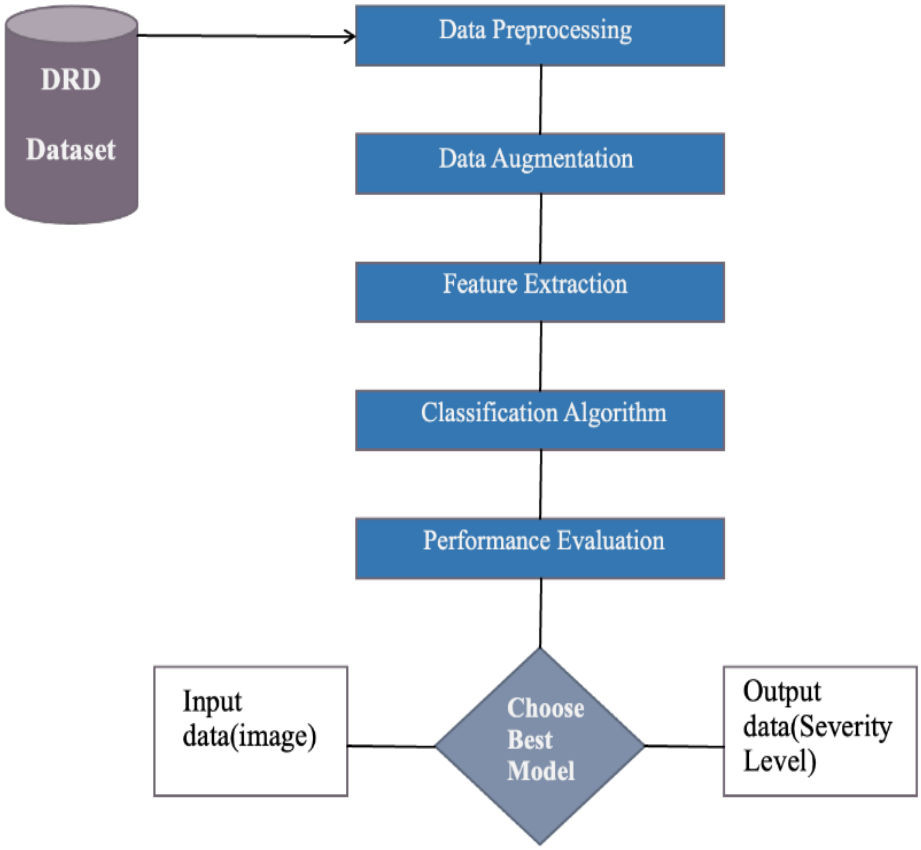

Fig-4.1 System Architecture

4.3 UML Diagrams:

4.3.1 Use Case Diagram:

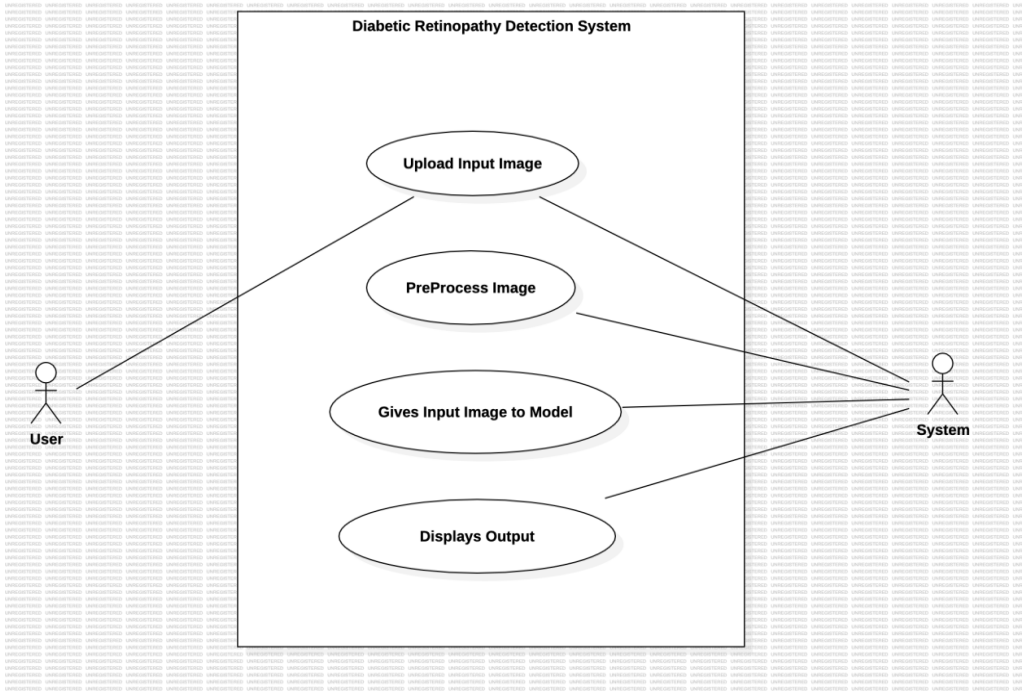

Fig-4.2 Use Case Diagram

4.3.2 Sequence Diagram:

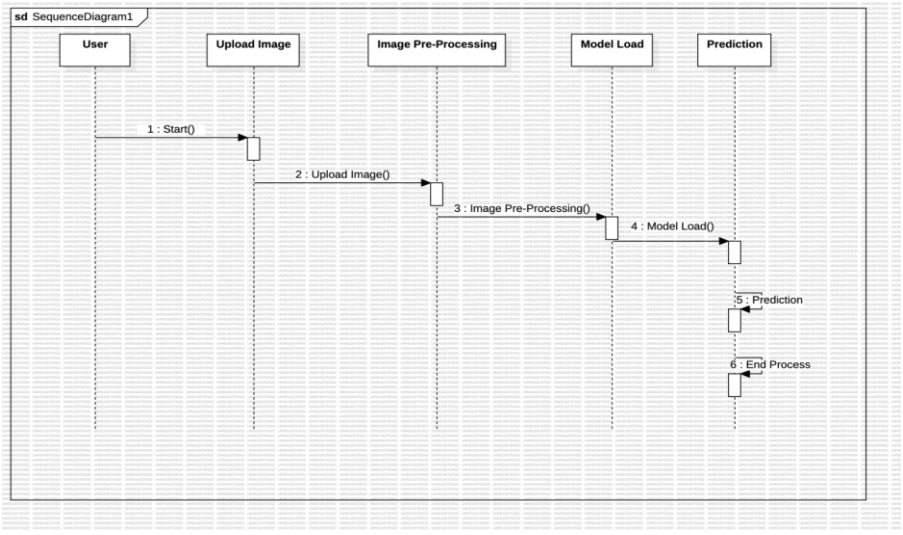

Fig-4.3 Sequence Diagram

4.3.3 Activity Diagram:

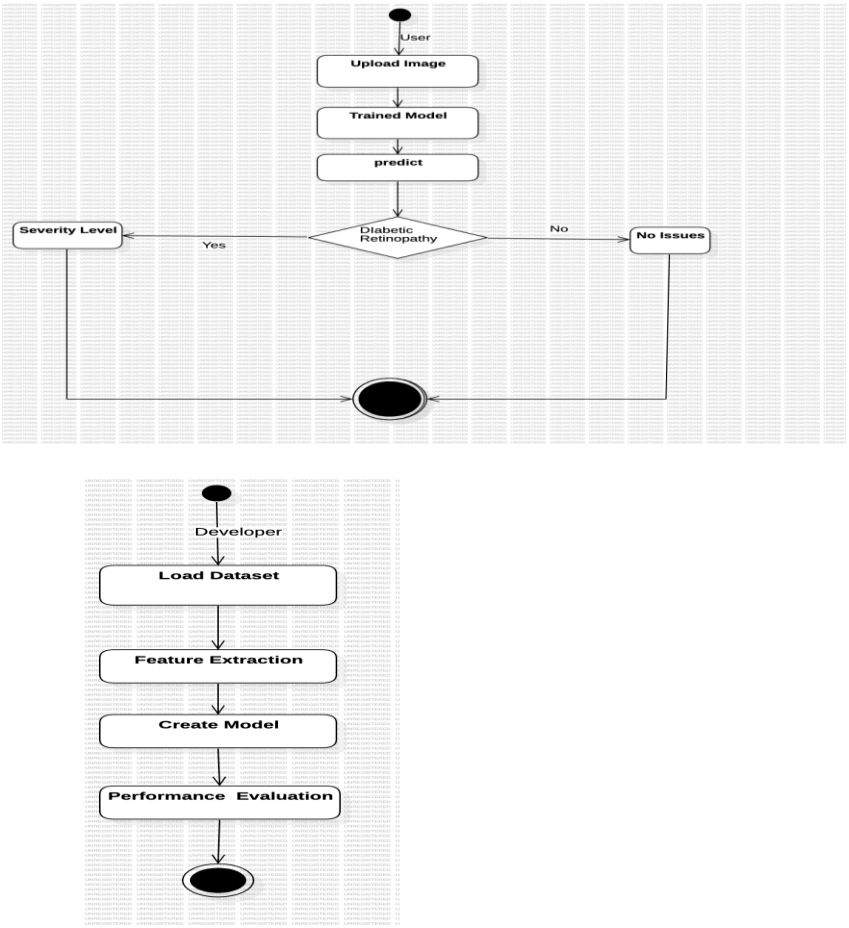

Fig-4.4 Activity Diagram

4.3.4 Class Diagram:

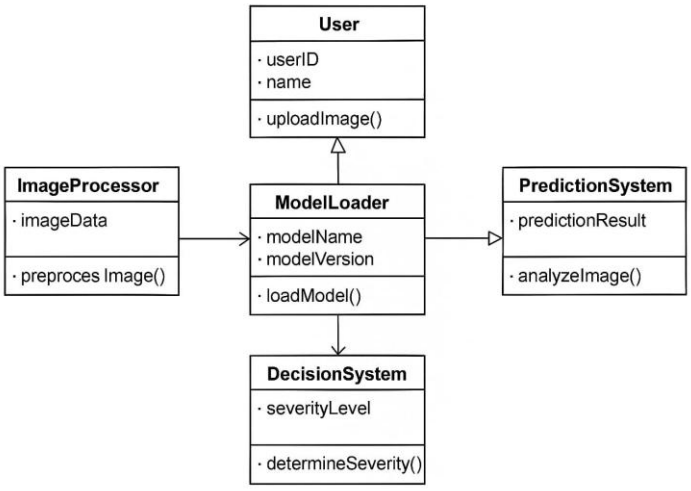

Fig-4.5 Class Diagram

CHAPTER 5

IMPLEMENTATION

5.1 Module Description:

Module 1: Importing Libraries

In this module, necessary libraries and modules are imported to facilitate various tasks such as data manipulation, image processing, machine learning model building, and performance evaluation. Notable libraries include Pandas for data handling, NumPy for numerical operations, OpenCV for image processing, Keras for building deep learning models, and scikit-learn for machine learning algorithms.

Module 2: Importing Dataset and Data Preprocessing

This section is dedicated to importing a dataset related to diabetic retinopathy and performing data preprocessing. The dataset is split into training and testing sets using the `splitfolders` library. Class labels are mapped to numerical values for model training. The images are loaded, resized to a consistent format, and then saved for further use.

Module 3: Model Building - CNN Feature Extraction (VGG16)

The code in this module focuses on building a Convolutional Neural Network (CNN) using the VGG16 architecture for feature extraction. The model is pretrained on the ImageNet dataset and then modified to suit the specific task of diabetic retinopathy classification. The CNN model's last few layers are removed, leaving only the convolutional layers responsible for feature extraction. This reduced model is then used to transform the input images into feature vectors.

Module 4: Machine Learning Model Training and Evaluation

Several machine learning models are trained and evaluated using the extracted features. The models include k-Nearest Neighbors (KNN), Random Forest, Support Vector Machine (SVM), Logistic Regression, Naive Bayes, XGBoost, AdaBoost, Decision Tree, and a Voting Classifier that combines multiple models. Each model's performance is assessed using metrics such as accuracy, precision, recall, and the confusion matrix.

Module 5: Model Comparison and Visualization

The performance scores of each model are compiled, and a bar chart is generated to visually compare their effectiveness.

5.2 Sample Code:

App.py

```
import os
from flask import Flask, render_template, request, redirect, url_for
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import random
import logging

app = Flask(__name__)

# Set up logging
logging.basicConfig(level=logging.DEBUG)

# Label mapping
verbose_name = {
    0: "No_DR",
    1: "Mild",
    2: "Moderate",
    3: "ERRORDATA"
}

# Load model
model_path = 'C:\\Users\\pc\\OneDrive\\Desktop\\diabetic detection using retinopathy\\model\\dr_model.h5'
model = load_model(model_path)

# State
last_prediction = None
last_confidence = None
last_metrics = {
```

```
    "accuracy": 0.0,
    "precision": 0.0,
    "recall": 0.0,
    "f_measure": 0.0
}
y_true_list = []
y_pred_list = []

def predict_label(img_path):
    test_image = image.load_img(img_path, target_size=(224, 224))
    test_image = image.img_to_array(test_image) / 255.0
    test_image = test_image.reshape(1, 224, 224, 3)
    predict_x = model.predict(test_image)
    classes_x = np.argmax(predict_x, axis=1)
    confidence = float(np.max(predict_x)) * 100
    label = verbose_name.get(classes_x[0], "Unknown")
    return label, confidence, classes_x[0]

@app.route("/")
@app.route("/first")
def first():
    return render_template('first.html')

@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return redirect(url_for('preview'))
    return render_template('login.html')

@app.route("/preview", methods=['GET', 'POST'])
def preview():
    global last_prediction, last_confidence, last_metrics, y_true_list, y_pred_list
```

```
    if request.method == 'POST':
        img = request.files.get('file')
        if not img or img.filename == '':
            return render_template("index.html", error="No file selected")

        img_path = os.path.join("static", "tests", img.filename)
        os.makedirs(os.path.dirname(img_path), exist_ok=True)
        img.save(img_path)

        predict_result, confidence, pred_class = predict_label(img_path)

        # Simulate varied true labels for confusion matrix
        true_class = pred_class if random.random() > 0.4 else random.randint(0, 3)
        y_true_list.append(true_class)
        y_pred_list.append(pred_class)

        # Log the lists to debug
        app.logger.debug(f"y_true_list: {y_true_list}")
        app.logger.debug(f"y_pred_list: {y_pred_list}")

        # Update metrics only if there are predictions
        if len(y_true_list) > 0 and len(y_pred_list) > 0:
            last_prediction = predict_result
            last_confidence = confidence
            last_metrics["accuracy"] = round(accuracy_score(y_true_list, y_pred_list), 2)
            last_metrics["precision"] = round(precision_score(y_true_list, y_pred_list, average='macro',
zero_division=0), 2)
            last_metrics["recall"] = round(recall_score(y_true_list, y_pred_list, average='macro',
zero_division=0), 2)
            last_metrics["f_measure"] = round(f1_score(y_true_list, y_pred_list, average='macro',
zero_division=0), 2)
```

```
        else:
            app.logger.warning("No predictions available to calculate metrics.")

        app.logger.debug(f"Updated last_metrics: {last_metrics}")

        return render_template("prediction.html", prediction=last_prediction,
                               confidence=last_confidence, img_path=img_path)

    return render_template("index.html")

@app.route("/performance")
def performance():
    conf_matrix = confusion_matrix(y_true_list, y_pred_list, labels=list(verbose_name.keys()))
    metrics = {
        'accuracy': last_metrics["accuracy"],
        'precision': last_metrics["precision"],
        'recall': last_metrics["recall"],
        'f_measure': last_metrics["f_measure"],
        'confusion_matrix': conf_matrix.tolist()
    }
    app.logger.debug(f"Performance metrics: {metrics}")
    return render_template('performance.html', metrics=metrics, severity_labels=verbose_name)

@app.route("/chart")
def chart():
    labels = ['Accuracy', 'Precision', 'Recall', 'F-Measure']
    # Scale the metrics to percentages
    values = [
        last_metrics["accuracy"] * 100,
        last_metrics["precision"] * 100,
        last_metrics["recall"] * 100,
        last_metrics["f_measure"] * 100
```

```
    ]
    app.logger.debug(f"Chart labels: {labels}, values: {values}")
    return render_template('chart.html', labels=labels, values=values)

@app.route("/logout")
def logout():
    return redirect(url_for('login'))

if __name__ == '__main__':
    app.run(debug=True)
```

train_model.py:

```
import os
import numpy as np
from sklearn.metrics import cohen_kappa_score
from sklearn.utils.class_weight import compute_class_weight
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D,
BatchNormalization, Input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

# === Paths ===
data_dir = "diabetic detection using retinopathy//gaussian_filtered_images"
model_dir = 'model'
os.makedirs(model_dir, exist_ok=True)
model_path = os.path.join(model_dir, 'dr_model.h5')

# === Classes ===
class_names = ['No_DR', 'Mild', 'Moderate', 'ERRORDATA']
```

```
# === Data Generators ===
train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    zoom_range=0.2,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(224, 224),
    batch_size=16,
    class_mode='categorical',
    subset='training',
    shuffle=True,
    classes=class_names
)

val_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(224, 224),
    batch_size=16,
    class_mode='categorical',
    subset='validation',
    shuffle=False,
    classes=class_names
)
```

```
# === Class Weights ===
if train_generator.samples == 0:
    raise ValueError("No training data found.")
if val_generator.samples == 0:
    raise ValueError("No validation data found.")

print("Class indices:", train_generator.class_indices)
print("Train class distribution:", np.bincount(train_generator.classes))

classes = np.array(list(train_generator.class_indices.values()))
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=classes,
    y=train_generator.classes
)
class_weight_dict = dict(zip(classes, class_weights))
print("Class weights:", class_weight_dict)

# === Load Pretrained VGG16 (without top) ===
base_model = VGG16(include_top=False, weights='imagenet', input_tensor=Input(shape=(224, 224, 3)))

# === Freeze base_model layers ===
for layer in base_model.layers:
    layer.trainable = False

# === Add Custom Head ===
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu')(x)
```

```
x = Dropout(0.5)(x)
output = Dense(len(class_names), activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)

# === Compile ===
model.compile(
    optimizer=tf.keras.optimizers.Adam(1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# === Callbacks ===
callbacks = [
    EarlyStopping(patience=10, restore_best_weights=True),
    ReduceLROnPlateau(factor=0.5, patience=5)
]

# === Train ===
history = model.fit(
    train_generator,
    epochs=100,
    validation_data=val_generator,
    class_weight=class_weight_dict,
    callbacks=callbacks
)

# === Evaluate QWK ===
val_generator.reset()
y_pred = model.predict(val_generator)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = val_generator.classes
```

```
kappa = cohen_kappa_score(y_true, y_pred_classes, weights='quadratic')
print(f"Quadratic Weighted Kappa: {kappa:.4f}")

# === Save Model ===
model.save(model_path)
print(f"Model saved to {model_path}")
```

CHAPTER 6
TESTING

6.1 Importance of Testing:

Testing is a crucial aspect of the diabetic retinopathy classification project described in the provided code. The importance of testing lies in various aspects of machine learning model development and deployment. Here are key reasons why testing is essential for this project:

1. **Model Evaluation and Performance Assessment:**

   Purpose: Testing allows the evaluation of the machine learning models' performance on unseen data.

   Significance: It provides insights into how well the trained models generalize to new, previously unseen retinal images. This assessment is critical for determining the effectiveness of the models in real-world scenarios.

2. **Generalization Capability Check:**

   Purpose: Testing assesses the generalization capability of the models.
   Significance: Generalization refers to how well a model performs on data it has not seen during training. Testing helps identify if the models can make accurate predictions on new retinal images, which is essential for real-world applications

3. **Preventing Overfitting:**

   Purpose: Testing helps identify and mitigate overfitting.
   Significance: Overfitting occurs when a model learns the training data too well but fails to generalize to new data. Testing allows detection of overfitting, enabling model adjustments to improve robustness.

4. **Hyperparameter Tuning:**

   Purpose: Testing aids in optimizing model hyperparameters.
   Significance: By evaluating model performance on a separate test set, hyperparameters can be fine-tuned to enhance model accuracy and reliability without compromising generalization.

5. Model Selection:

Purpose: Testing facilitates the comparison of multiple models.
Significance: Different machine learning models may be trained and tested, and their performances compared to select the most suitable model for diabetic retinopathy classification. Testing helps identify which model performs best on unseen data.

6. Decision Making for Development:

Purpose: Testing informs decisions regarding model deployment.
Significance: Before deploying a model for real-world use, it must undergo rigorous testing. Reliable and accurate models are essential for providing valuable insights into diabetic retinopathy severity and guiding appropriate medical interventions.

7. Quality Assurance:

Purpose: Testing ensures the quality and reliability of the entire model pipeline.
Significance: Rigorous testing helps identify potential issues, errors, or inconsistencies in the code, data preprocessing, and model implementation, contributing to the overall quality assurance of the project.

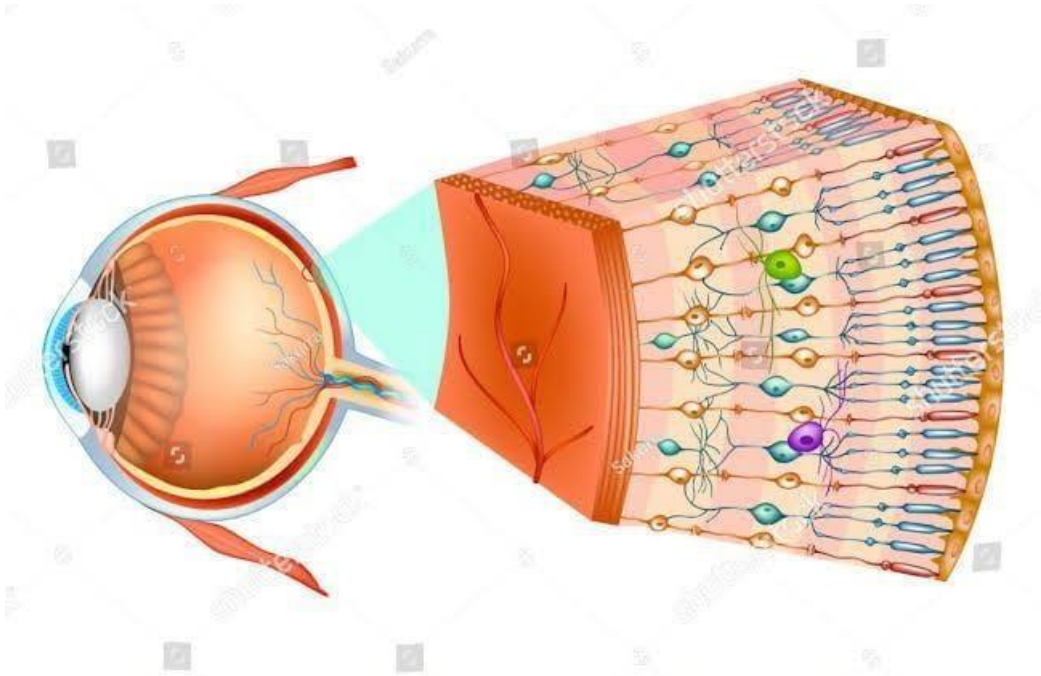

Fig-6.1 Internal Structure

6.2 Types of Testing:

I. White Box Testing:

In white-box testing, the developer will inspect every line of code before handing it over to the testing team or the concerned test engineers.

II. Black Box Testing:

Another type of manual testing is black-box testing. In this testing, the test engineer will analyse the software against requirements, identify the defects or bug, and sends it back to the development team.

III. Functional Testing:

The test engineer will check all the components systematically against requirement specification is known as functional testing. It is also known as Component testing.

IV. Non-functional Testing:

The next part of black-box testing is non-functional testing. It provides detailed information on software product performance and used technologies.

V. Unit Testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

The unit test unit testing framework was originally inspired by JUnit and has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing

of setup and shut down code for tests, aggregation of tests into collections, and independence of the tests from there porting framework.

To achieve this, unit test supports some important concepts in an object-oriented way:

- **Test Fixture –** A test fixture represents the preparation needed to perform one or more tests, and any associated cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.
- **Test Case –** A test case is the individual unit of testing. It checks for a specific response to a particular set of inputs. Unit test provides a base class, Test Case, which may be used to create new test cases.
- **Test Suite -** A test suite is a collection of test cases, test suites, or both. It is used to aggregate tests that should be executed together.
- **Test Runner -** A test runner is a component which orchestrates the execution of tests and provides the outcome to the user. The runner may use a graphical interface, a textual interface, or return a special value to indicate the results of executing the tests.

VI. Outcomes Possible:

There are three types of possible test outcomes:

1.OK–This means that all the tests are passed.

2.FAIL–This means that the test did not pass and an Assertion Error exception is raised.

3.ERROR–This means that the test raises an exception other than Assertion Error

6.3 Test Cases:

S.No	Test Scenario	Test Case	Predictions	Test Steps	Test Data	Expected Results	Post Conditions	Actual Result	Status
1.	Detection of Severity Level of Diabetic Retinopath y	Retinal Image captured through Fundus Idea	Trained CNN model	Retinal image capturing and sending it to build model	Dataset containing all the classes of Diabetic Retinopathy	Classifying the image into No_DR	Results are stored for further analyses	Image is classified into No_DR	Pass
2.	Detection of Severity Level of Diabetic Retinopath y	Retinal Image captured through Fundus Idea	Trained CNN model	Retinal image capturing and sending it to build model	Dataset containing all the classes of Diabetic Retinopathy	Classifying the image into Mild	Results are stored for further analyses	Image is classified into Mild	Pass
3.	Detection of Severity Level of Diabetic Retinopath y	Retinal Image captured through Fundus Idea	Trained CNN model	Retinal image capturing and sending it to build model	Dataset containing all the classes of Diabetic Retinopathy	Classifying the image into Moderate	Results are stored for further analyses	Image is classified into Moderate	Pass

4.	Detection of Severity Level of Diabetic Retinopath y	Retinal Image captured through Fundus Idea	Trained CNN model	Retinal image capturing and sending it to build model	Dataset containing all the classes of Diabetic Retinopathy	Classifying the image into Proliferate	Results are stored for further analyses	Image is classified into Prolifer ate	Pass
5.	Detection of Severity Level of Diabetic Retinopath y	Retinal Image captured through Fundus Idea	Trained CNN model	Retinal image capturing and sending it to build model	Dataset containing all the classes of Diabetic Retinopathy	Classifying the image into Severe	Results are stored for further analyses	Image is classif i ed into Sever e	Pass

CHAPTER 7

OUTPUT SCREENSHOTS

Home page:

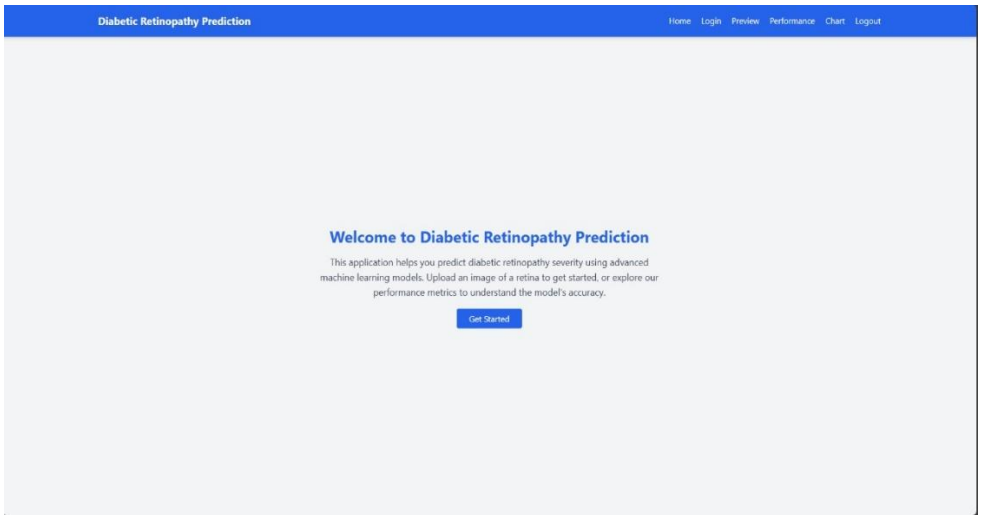

Fig-7.1 Home Page

Login page:

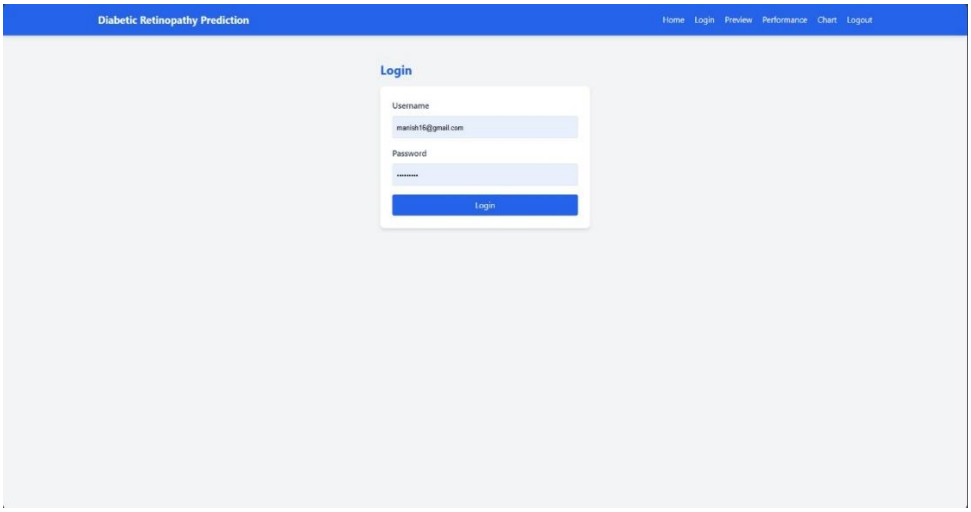

Fig-7.2 Login Page

Preview Page:

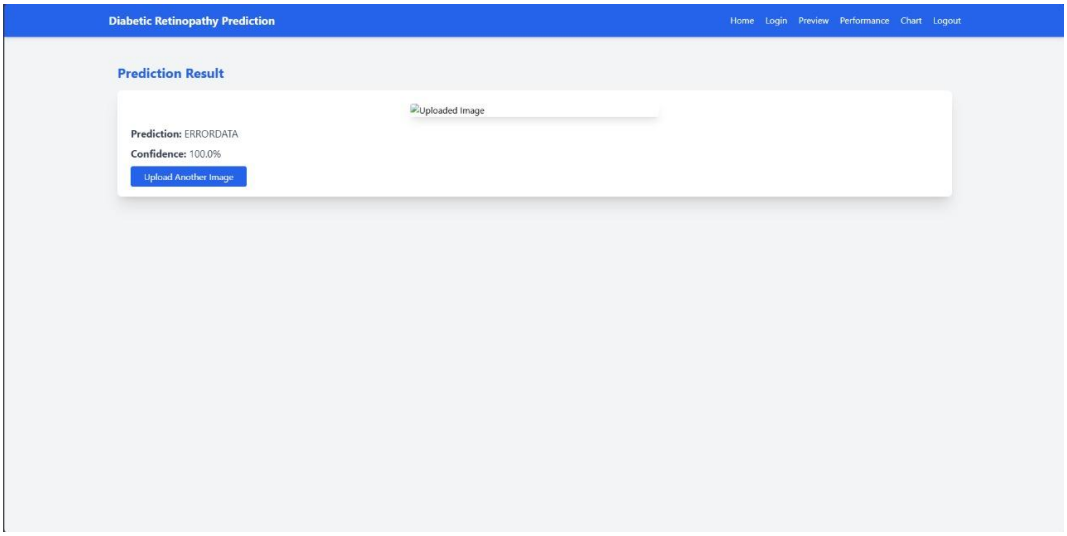

Fig-7.3 Preview Page

Performance Page:

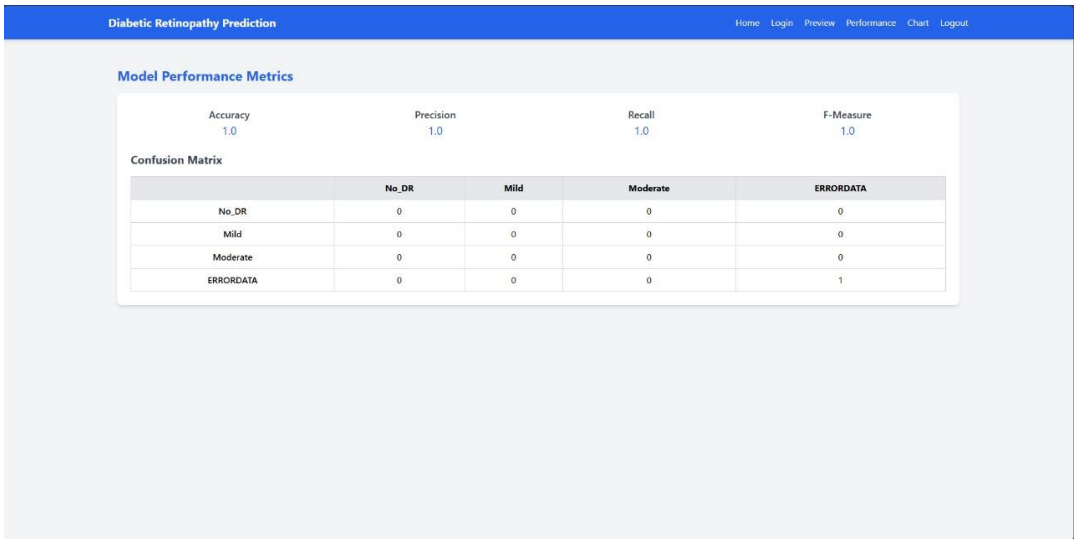

Fig-7.4 Performance Page

Chart Page:

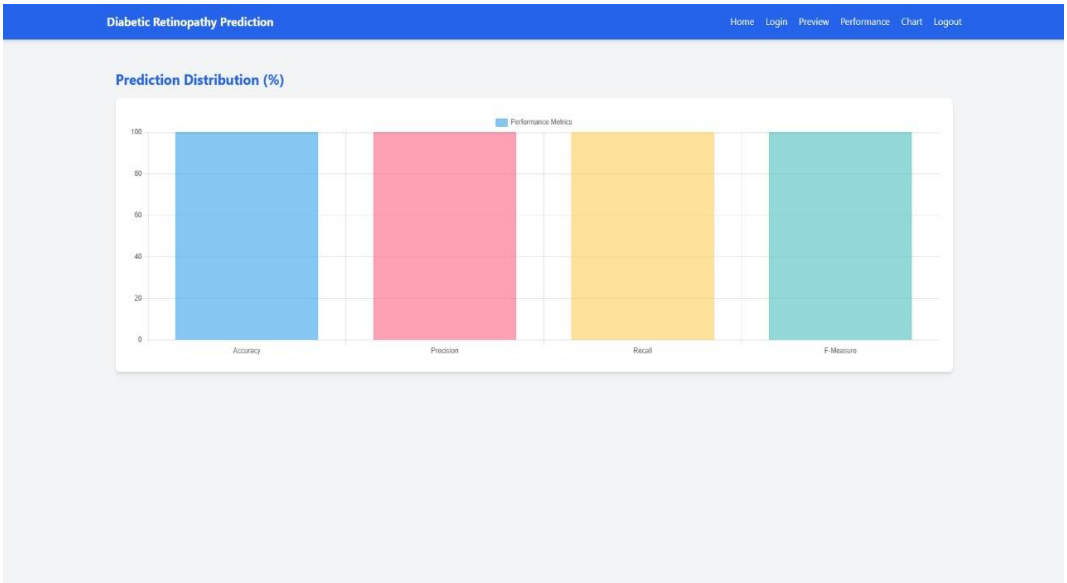

Fig-7.5 Chart Page

CHAPTER 8

CONCLUSION

In conclusion, the use of Convolutional Neural Networks (CNN) in conjunction with machine learning has proven to be a highly successful method for the early diagnosis of diabetic retinopathy. This method has demonstrated great effectiveness in precisely recognizing and categorizing retinal pictures, enabling early diagnosis and intervention. It does this by utilizing deep learning and machine learning algorithms. We used Convolutional Neural Network each input image will be classified with the greatest degree of precision possible. There will be a hope that there won't be any misclassification because of the accuracy of our model. We started with the basics. CNN was taught to recognize lines, edges, corners, and other features in images. In the process of analyzing retinal images, the system identifies small components of a picture, starting with individual features and progressively recognizing entire features. It has learned to recognize specific features such as microaneurysms and hemorrhages in the retina. The proposed approach is not only beneficial to those with Diabetic Retinopathy, but it may also be employed by persons with Melanoma and Myeloid Leukemia. The use of CNN in the identification of diabetic retinopathy offers a number of significant benefits. CNN models are highly suited for analyzing the subtle structures and patterns inherent in retinal pictures since they were created particularly to analyze images and extract fine characteristics. These models can automatically pick up on and adjust to the features of diabetic retinopathy, leading to increased detection accuracy and resilience. We have used various machine learning algorithms like KNN, Xgboost, Voting Classifier, Decision Tree, Logistic Regression, Adaboost, Support Vector Machine, Random Forest to classify the retinal images according to their severity level. We integrated CNN and machine learning algorithms.

CHAPTER 9

FUTURE SCOPE

The identification of diabetic retinopathy using CNN and machine learning has the potential to transform clinical practice. It can aid medical practitioners in making an accurate diagnosis, encourage early action, and stop permanent eyesight loss. However, despite the progress made, there are still challenges to address. The development of robust algorithms requires large and diverse datasets, including data from different ethnicities and age groups. Additionally, the integration of AI systems into existing healthcare workflows and the acceptance of these technologies by healthcare providers are crucial for successful implementation. Ensuring the privacy and security of patient data is another important consideration that must be carefully addressed. In conclusion, the development of AI-based systems for diabetic retinopathy detection holds great promise in revolutionizing DR screening and improving patient outcomes. With continued research, technological advancements, and collaborations between healthcare providers, researchers, and technology developers, the future of DR detection looks bright.

CHAPTER 10

REFERENCES

[1]. Kumar, P.N.S.; Deepak, R.U.; Sathar, A.; Sahasranamam, V.; Kumar, R.R. Automated Detection System for Diabetic Retinopathy Using Two Field Fundus Photography. Procedia Comput. Sci. 2016, 93, 486–494.

[2]. Zhu, C.Z.; Hu, R.; Zou, B.J.; Zhao, R.C.; Chen, C.L.; Xiao, Y.L. Automatic Diabetic Retinopathy Screening via Cascaded Framework Based on Image- and Lesion-Level Features Fusion. J. Comput. Sci. Technol. 2019 346 2019, 34, 1307–1318.

[3]. Sambyal, N.; Saini, P.; Syal, R.; Gupta, V. Modified Residual Networks for Severity Stage Classification of Diabetic Retinopathy. Evol. Syst. 2022, 1–19.

[4]. Khan, A.I.; Kshirsagar, P.R.; Manoharan, H.; Alsolami, F.; Almalawi, A.; Abushark, Y.B.; Alam, M.; Chamato, F.A. Computational Approach for Detection of Diabetes from Ocular Scans.

Comput. Intell. Neurosci. **2022**, 2022, 1–8

[5]. Ali, R.; Hardie, R.C.; Narayanan, B.N.; Kebede, T.M. IMNets: Deep Learning Using an Incremental Modular Network Synthesis Approach for Medical Imaging Applications. Appl. Sci. **2022**, 12, 5500.

[6]. Menaouer, B.; Dermane, Z.; El HoudaKebir, N.; Matta, N. Diabetic Retinopathy Classification Using Hybrid Deep Learning Approach. SN Comput. Sci. **2022**, 3, 357.

[7]. Gunasekaran, K.; Pitchai, R.; Chaitanya, G.K.; Selvaraj, D.; Annie Sheryl, S.; Almoallim, H.S.; Alharbi, S.A.; Raghavan, S.S.; Tesemma, B.G. A Deep Learning Framework for Earlier Prediction of Diabetic Retinopathy from Fundus Photographs. Biomed Res. Int. **2022**, 2022, 1–15.

[8]. Khan, A.; Kulkarni, N.; Kumar, A.; Kamat, A. D-CNN and Image Processing Based Approach for Diabetic Retinopathy Classification. Appl. Inf. Process. Syst. **2022**, 1354, 283–291.

[9]. Fang, L.; Qiao, H. Diabetic Retinopathy Classification Using a Novel DAG Network Based on Multi-Feature of Fundus Images. Biomed. Signal Process. Control 2022, 77, 103810

[10]. Luo, X.; Pu, Z.; Xu, Y.; Wong, W.K.; Su, J.; Dou, X.; Ye, B.; Hu, J.; Mou, L. MVDRNet: Multi-View Diabetic Retinopathy Detection by Combining DCNNs and Attention

Mechanisms. Pattern Recognition. 2021, 120, 108104.
[11]. Adriman, R.; Muchtar, K.; Maulina, N. Performance Evaluation of Binary Classification of Diabetic Retinopathy through Deep Learning Techniques Using Texture Feature. Procedia Comput. Sci. 2021, 179, 88–94.
[12]. Fatima; Imran, M.; Ullah, A.; Arif, M.; Noor, R. A Unified Technique for Entropy Enhancement Based Diabetic Retinopathy Detection Using Hybrid Neural Network. Comput. Biol. Med. 2022, 145, 105424.
[13]. Ragab, M.; Aljedaibi, W.H.; Nahhas, A.F.; Alzahrani, I.R. Computer Aided Diagnosis of Diabetic Retinopathy Grading Using Spiking Neural Network. Comput. Electr. Eng. 2022, 101, 108014.
[14]. Qureshi, I.; Ma, J.; Abbas, Q. Diabetic retinopathy detection and stage classification in eye fundus images using active deep learning. Multimed Tools Appl 2021, 80, 11691–11721.
[15]. Gayathri, S.; Gopi, V.P.; Palanisamy, P. Diabetic Retinopathy Classification Based on Multipath CNN and Machine Learning Classifiers. Phys. Eng. Sci. Med. 2021, 44, 639–653.