DEEP LEARNING BASED RECOGNITION OF OCULAR DISEASE IN FUNDUS IMAGES.

A PROJECT REPORT

Submitted by

VETTRI CHEZHIAN P	**211419205175**
NITHISH KUMAR S	**211419205119**
RAJ KUMAR K	**211419205136**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DEEP LEARNING BASED RECOGNITION OF OCULAR DISEASE IN FUNDUS IMAGES."** is the bonafide work of **"VETTRI CHEZHIAN P (211419205175), NITHISH KUMAR S (211419205119), RAJ KUMAR K (211419205136)"** who carried out the project under my supervision.

SIGNATURE

Dr. M. Helda Mercy M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Department of Information Technology
Panimalar Engineering College
Poonamallee, Chennai - 600 123

SIGNATURE

Mr. Dr. K. Sridharan M.E., Ph.D.,

SUPERVISOR

Department of Information Technology
Panimalar Engineering College
Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on _____

SIGNATURE

INTERNAL EXAMINER

SIGNATURE

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project report entitled "**DEEP LEARNING BASED RECOGNITION OF OCULAR DISEASE IN FUNDUS IMAGES. DETECTION OF GLAUCOMA CATARACT AND DIABETIC RETINOPATHY**" which is being submitted in partial fulfilment of the requirement of the course leading to the award of the 'Bachelor Technology in Information Technology' in **Panimalar Engineering College, Autonomous institution Affiliated to Anna university-Chennai** is the result of the project carried out by me under the guidance and supervision of **Dr. K. Sridharan M.E., Ph.D., Professor in the Department of Information Technology**. We further declared that we or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

VETTRI CHEZHIAN P **NITHISH KUMAR S**

RAJ KUMAR K

Date**:**

Place: Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date: **Dr. K. Sridharan, M.E., Ph.D.,**

Place: Chennai (Professor / IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to Our **Honorable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to Our **Dynamic Directors, Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E., M.B.A., Ph.D.,** and **Dr.saranya sree sakthikumar., B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, Dr. **M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project co-ordinator Mr. **M. DILLI BABU**, **M.E., (Ph.D.,)** Associate Professor, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our supervisor **Dr. K. Sridharan M.E., Ph.D., Professor, Department of Information Technology** for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

TABLE OF CONTENTS

ABSTRACT

The aim of this system is to develop a deep learning model for the recognition of ocular diseases in fundus images, specifically glaucoma, cataract, and diabetic retinopathy. The system utilizes a dataset of fundus images that are labeled with disease categories to train a convolutional neural network (CNN) model. The CNN model consists of four convolutional layers, four max-pooling layers, and two fully connected layers, with rectified linear unit (ReLU) activation used in the convolutional layers. Image augmentation is used to increase the size of the dataset and reduce overfitting. The model is trained and validated using the training and testing datasets, respectively, with early stopping used to prevent overfitting. The performance of the model is evaluated using metrics such as accuracy, precision, recall, and F1 score. The results show that the model achieves a high level of accuracy in recognizing the three ocular diseases in the fundus images, demonstrating the potential for deep learning models to aid in the diagnosis of ocular diseases.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ReLU	-	Rectified Linear Unit
IOP	-	Intraocular Pressure
CNN	-	Convolutional Neural Network
AI	-	Artificial Intelligence
ML	-	Machine Language
URL	-	Uniform Resource Locator
AMD	-	Age-Related Macular Degeneration
DR	-	Diabetic Retinopathy
HR	-	Hypertensive Retinopathy
SVM	-	Support Vector Machines
KNN	-	K-nearest Neighbors
VEGF	-	Vascular Endothelial Growth Factor
DBN	-	Deep Belief Networks
RNN	-	Recurrent Neural Networks
GON	-	Glaucomatous Optic Neuropathy
FCN	-	Fully Convolutional Network
ANN	-	Artificial Neural Networks
OCT	-	Optical Coherence Tomography
IDE	-	Integrated Development Environment
RAM	-	Random Access Memory
OS	-	Operating System
OpenCV	-	Open-Source Computer Vision Library
GAN	-	Generative Adversarial Networks

CHAPTER-1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT:

The Deep learning based recognition of ocular disease in Fundus images is a replacement way for the traditional way of detecting ocular diseases. The proposed system aims to develop a web-based application for diagnosing common eye diseases using deep learning models. The application can detect three common eye diseases: glaucoma, cataract, and diabetic retinopathy. The deep learning models are trained using fundus images of the eye, which are taken by an ophthalmologist. The system uses the Flask web framework to develop a user interface for the application. Users can upload fundus images of the eye and select the type of disease they suspect. The deep learning model corresponding to the selected disease is used to analyze the uploaded image and make a prediction about the presence or absence of the disease. The model is trained to predict the likelihood of disease occurrence in the image, which is presented to the user as a percentage.

The models have been trained using the Keras deep learning library and have been saved in the H5 format for easy loading and use in the application. The user interface of the application has been designed using HTML and Bootstrap, while the back-end logic has been implemented in Python using the Flask web framework. In this system, CNNs are used to predict three common eye diseases (glaucoma, cataract, and diabetic retinopathy) based on fundus images of the eye. The models are trained on a dataset of thousands of labelled fundus images using a transfer learning approach, where a pre-trained model is fine-tuned on the task of classifying eye diseases. The resulting models are then deployed in a Flask web application to allow users to upload their own fundus images and receive predictions of whether they are likely to have one of the three eye diseases. This system can be useful for individuals who are at risk of developing these eye diseases or for individuals who suspect that they may be experiencing symptoms of the diseases. By quickly identifying the presence or absence of these diseases, individuals can take the necessary steps to address their eye health and prevent the diseases from progressing. The proposed system is an Ocular Disease Recognition system that aims to recognize three types of ocular diseases, namely glaucoma, cataract,

and diabetic retinopathy. The system utilizes a Convolutional Neural Network (CNN) to make predictions on fundus images of the eye. The system consists of five modules, including Data Collection, Data Pre-Processing, Image Augmentation, CNN Model Development and Evaluation, and Web App Development.

1.1.1 Title Explanation

Deep Learning: Deep learning-based system that can automatically recognize and classify these three ocular diseases from Fundus images, thereby facilitating early detection and timely treatment.

Ocular Diseases: Glaucoma, Cataract, and Diabetic Retinopathy are three of the most common ocular diseases, and their early detection and treatment are crucial for preventing vision loss and blindness.

Fundus Images: Images of the interior surface of the eye shown in fig 1.1, captured by a special camera called a Fundus camera. These images can reveal information about various ocular diseases, and are widely used for diagnosis and treatment planning.

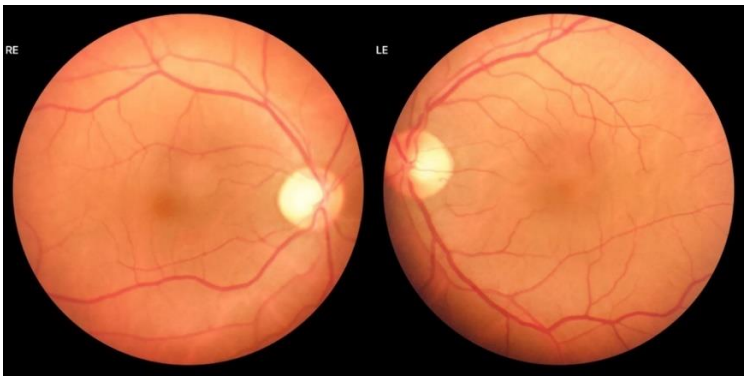

Fig 1.1 Normal Eye Fundus images

Glaucoma: It is a group of eye diseases that damage the optic nerve, which connects the eye to the brain shown in fig1.2. It is usually caused by high pressure inside the eye, called intraocular pressure (IOP), which can damage the optic nerve and lead to vision loss or blindness.

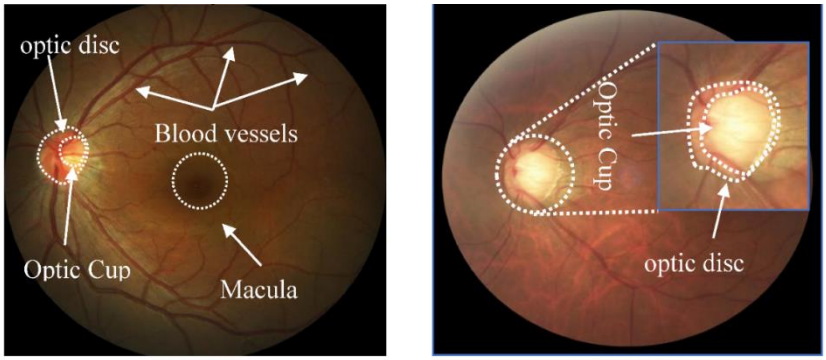

Fig 1.2 Normal Eye vs Glaucoma Fundus images

Cataract: It is a clouding of the eye's natural lens, which lies behind the iris and the pupil. The lens works much like a camera lens, focusing light onto the retina at the back of the eye shown in fig 1.3. When the lens becomes cloudy, it can cause vision problems such as blurred vision, sensitivity to light and glare, and difficulty seeing at night.

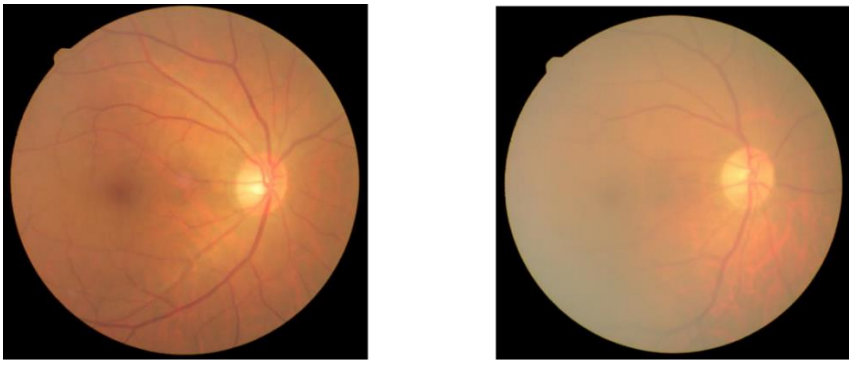

Fig 1.3 Normal Eye vs Cataract Fundus images

Diabetic Retinopathy: It is a serious complication of diabetes that affects the eyes. It occurs when high blood sugar levels damage the blood vessels in the retina, leading to vision loss and blindness if left untreated shown in fig 1.4.

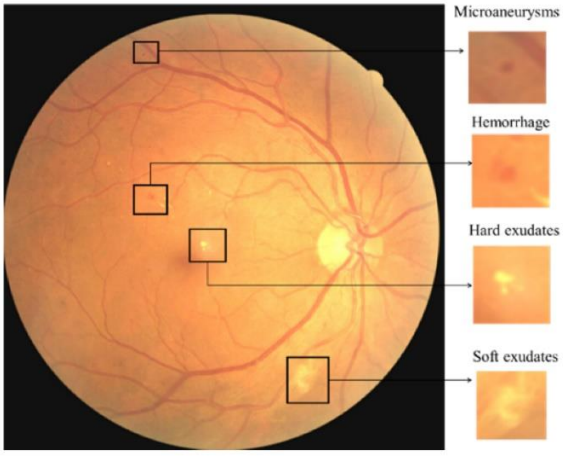

Fig 1.4 Diabetic retinopathy Fundus images

1.2 NEED FOR THE PROJECT:

The need for this proposed system is to develop a computer-aided diagnostic system for detecting three common ocular diseases: glaucoma, cataract, and diabetic retinopathy. These diseases are prevalent and can lead to vision loss or blindness if not diagnosed and treated early. However, the diagnosis of these diseases requires specialized skills and equipment, making it challenging for patients in rural or remote areas to access reliable diagnoses.

The proposed system aims to provide a solution to this problem by using a deep learning model to accurately detect these diseases from fundus images, which can be taken using a simple and inexpensive camera. Additionally, a web application is developed to provide easy access to the model's predictions, making it accessible to anyone with an internet connection.

Therefore, the need for this proposed system is to develop a reliable, accurate, and accessible system for detecting ocular diseases, particularly for patients in rural or remote areas who have limited access to specialized medical facilities.

1.2.1 Problem Statement

The problem statement of this proposed system is to develop a web-based application for diagnosing common eye diseases using deep learning models. The goal is to create a system that can analyze fundus images of the eye and accurately detect ocular diseases such as glaucoma, cataract, and diabetic retinopathy. The system should be user-friendly, easy to use, and accessible to anyone with an internet connection. The proposed system involves several modules, including data collection, data pre-processing, image augmentation, CNN model development and evaluation, and web app development. By implementing these modules, the proposed system aims to address the problem of diagnosing eye diseases in a timely and accurate manner, improving patient outcomes and reducing the burden on healthcare professionals.

1.3 OBJECTIVE OF THE PROJECT:

The objectives of this proposed system are to develop convolutional neural network models that can accurately classify fundus images of healthy and diseased eyes. The specific diseases targeted are diabetic retinopathy, glaucoma, and cataracts. The ultimate goal is to provide a tool for early detection and intervention to prevent or reduce vision loss.

Our primary goals of this proposed system are to develop a convolutional neural network model that can accurately classify healthy and non-healthy eyes based on fundus images, and to create separate binary classification models for each disease. Additionally, the proposed system aims to analyze and document the effectiveness of various techniques, such as image augmentation and CNN architecture, used to develop these models. The ultimate goal is to develop a reliable tool for early detection and diagnosis of eye diseases. Additionally, we seek to:

1. Provide an early detection model with high accuracy.
2. By Automated Diagnosis we are reducing the burden on medical professionals and increasing the speed and accuracy of diagnosis.
3. This proposed system has the potential to make eye disease detection more accessible, especially in areas where access to specialized medical professionals is limited.
4. Cost-Effective and Scalability, Once trained, the machine learning models can be easily scaled to handle large volumes of images, making it suitable for use in large-scale screening programs.

1.4 SCOPE OF THE PROJECT:

The scope of this proposed system is to develop a computer-aided diagnosis system for ocular disease recognition using a deep learning algorithm. The system can recognize three major ocular diseases, including glaucoma, cataract, and diabetic retinopathy. It involves multiple modules, including the image dataset preparation, image augmentation, CNN model development and evaluation, and web application

development. The system's main objective is to provide accurate and reliable disease recognition for early diagnosis and treatment, reducing the risk of blindness and other complications caused by the ocular diseases. The proposed system's scope also includes the deployment of the web application to make it accessible to healthcare professionals and patients worldwide.

The use of artificial intelligence (AI) and machine learning (ML) techniques in medical diagnosis and treatment has been gaining popularity in recent years. One area of interest is the use of ML to analyze medical images for the diagnosis and prognosis of various diseases. In this proposed system, we focused on using ML techniques to analyze fundus images of eyes to classify healthy and non-healthy eyes. Fundus images of eyes are widely used in ophthalmology for the diagnosis of various eye diseases, including glaucoma, diabetic retinopathy, and cataracts. With the help of ML, we aimed to develop accurate and efficient models for the early detection of these diseases.

1.4.1 Ocular diseases detection Usage

The primary goal of this proposed system is to develop a system for automated detection of ocular diseases, such as Glaucoma, Diabetic Retinopathy, and Cataracts, from fundus images using convolutional neural network (CNN) models. The proposed system uses a dataset of fundus images of healthy eyes and eyes with each of the three diseases to train and test the CNN models. Image pre-processing techniques are used to standardize the images and make them suitable for training the CNN models. The CNN models are trained using the Keras deep learning library with the TensorFlow backend.

In addition to CNN models, the proposed system uses various techniques, such as data augmentation, to improve the accuracy of the models. Data augmentation is used to increase the size of the dataset by applying various transformations to the images, such as rotating, shifting, and flipping. This helps to reduce overfitting and improve the generalization of the models.

The proposed system uses a separate binary classification model for each disease to address the challenge of eyes with multiple diseases. The binary classification models are trained to detect the presence or absence of each disease in the fundus images.

The advantages of this proposed system include the ability to provide an automated and accurate method for detecting ocular diseases from fundus images, which can assist ophthalmologists in making diagnoses and treatment decisions. The use of CNN models and data augmentation techniques improves the accuracy and generalization of the models, and the separate binary classification models for each disease address the challenge of eyes with multiple diseases.

1.4.2 Ocular diseases detection Techniques and Methods

This proposed system employs several techniques and methods to accomplish its goal of accurately classifying healthy and non-healthy eyes based on fundus images. First, the proposed system uses convolutional neural network (CNN) models, a type of deep learning model commonly used for image recognition tasks. These models are designed to automatically learn and extract relevant features from images, making them well-suited for this proposed system's classification task.

The proposed system also employs several image processing techniques to pre-process the fundus images before they are fed into the CNN models. These techniques include resizing the images to a standardized size, converting them to grayscale, and normalizing the pixel values to improve model performance.

To further improve model performance, the proposed system also uses data augmentation techniques to artificially increase the size of the training dataset. These techniques include random rotations, flips, and zooms of the original images, which create additional variations of the images that the models can learn from.

The proposed system also utilizes transfer learning, a technique where pre-trained CNN models are used as a starting point for training the models on new datasets. This approach can significantly reduce the training time and improve model performance, particularly when working with smaller datasets.

CHAPTER-2

LITERATURE SURVEY

2.1 Ocular diseases detection using recent deep learning techniques.

Author: T. Guergueb and M. A. Akhloufi.

Year: 2021

This article presents a study on the use of recent deep learning techniques for the detection of ocular diseases. The authors focus on three common eye diseases: diabetic retinopathy, age-related macular degeneration, and glaucoma. These diseases are major causes of blindness and visual impairment worldwide, and early detection is critical for effective treatment.

The proposed architecture is trained and evaluated using a publicly available dataset of retinal images, achieving high classification accuracy for all tested diseases. The authors also compare their results to those of other state-of-the-art deep learning models and traditional machine learning methods, demonstrating the superiority of their proposed approach.

The study uses a dataset of retinal fundus images, which are images of the back of the eye that can reveal the presence of these diseases. The authors compare the performance of several deep learning models, including VGG16, Inception-v3, and ResNet50, for the detection of the three ocular diseases. The models were trained on a subset of the dataset and tested on a separate subset.

The results of the study show that all three models achieved high accuracy in the detection of the ocular diseases, with ResNet50 achieving the highest accuracy. The authors also compare the performance of the deep learning models to that of human experts and find that the models perform comparably to or better than the experts in detecting the diseases.

The study highlights the potential of deep learning techniques for the detection and diagnosis of ocular diseases and emphasizes the need for further research in this area to develop more accurate and efficient methods for early detection and intervention. The authors suggest that future work could focus on incorporating additional clinical and demographic data into the deep learning models to improve their performance and applicability in real-world clinical settings.

2.2 Application of Ocular Fundus Photography and Angiography.

Author: Caroline Ka Lin Chee, Patrick A. Santiago, Gopal Lingam, Mandeep Singh.
Year: 2014

This article provides an overview of ocular fundus photography and angiography and their applications in ophthalmology. The ocular fundus is the back part of the eye, which includes the retina, optic disc, macula, and blood vessels. Fundus photography and angiography are non-invasive imaging techniques that allow for the visualization and evaluation of these structures.

The article discusses the technical aspects of fundus photography and angiography, including the different types of cameras and imaging systems used. It also covers the various clinical applications of these techniques, including the diagnosis and monitoring of various retinal and choroidal diseases, such as diabetic retinopathy, age-related macular degeneration, and retinal vascular occlusions.

The article also describes the role of fundus photography and angiography in the management of ocular diseases, including the planning and monitoring of treatment strategies such as laser therapy and anti-vascular endothelial growth factor (VEGF) injections. The authors highlight the importance of accurate interpretation and diagnosis of fundus images and angiograms by trained ophthalmologists and the potential for computer-aided diagnosis and automated analysis.

The chapter highlights the importance of ocular fundus photography and angiography in the diagnosis and management of ocular diseases, as they allow for the early detection of pathological changes in the eye and enable the monitoring of disease progression over time. The authors suggest that future research could focus on the development of new imaging techniques and the refinement of existing techniques to improve their accuracy and applicability in clinical settings.

Overall, the article emphasizes the clinical significance of ocular fundus photography and angiography in ophthalmology and the importance of ongoing research and technological advancements in the field.

2.3 Ocular diseases diagnosis in fundus images using a deep learning: approaches, tools and performance evaluation.

Author: Y. Elloumi, M. Akil, and H. Boudegga.

Year: 2019

Describes a deep learning-based approach for the diagnosis of ocular diseases in fundus images. The authors present a comprehensive review of existing deep learning techniques for fundus image analysis, and propose their own approach using a convolutional neural network (CNN) with transfer learning.

The proposed approach consists of three main steps: pre-processing, feature extraction, and classification. In the pre-processing step, the fundus image is enhanced and normalized to reduce noise and artifacts. In the feature extraction step, a pre-trained CNN is used to extract high-level features from the pre-processed image. Finally, the extracted features are fed into a SoftMax classifier for disease classification.

The authors evaluate their approach on a publicly available dataset of fundus images and compare the performance with other state-of-the-art techniques. The results show that their approach achieves high accuracy and outperforms other methods in terms of sensitivity, specificity, and F1 score.

The authors evaluate the performance of their models using metrics such as accuracy, precision, recall, and F1 score. They report that the DenseNet-121 model achieves the best performance, with an accuracy of 92.5% for AMD detection, 94.7% for DR detection, and 93.8% for glaucoma detection.

The authors conclude that deep learning models can effectively diagnose ocular diseases in fundus images, and that the performance of the models can be improved by using more advanced architectures and larger datasets. They suggest that their approach could be used in clinical settings to aid ophthalmologists in the early detection and diagnosis of ocular diseases.

Overall, this paper provides a valuable contribution to the field of ocular disease diagnosis using deep learning techniques, and demonstrates the potential of these approaches for improving the accuracy and efficiency of disease diagnosis in clinical settings.

2.4 Simulating the Influences of Aging and Ocular Disease on Biometric Recognition Performance.

Author: Borgen, P. Bours, and S. D. Wolthusen.

Year: 2009

This article presents a study on the effects of aging and ocular diseases on the performance of biometric recognition systems. Biometric recognition refers to the use of physical or behavioural characteristics, such as fingerprints or facial features, to identify individuals.

The paper discusses the effects of aging and ocular disease on biometric recognition performance, specifically in iris recognition. The authors use a simulation approach to investigate the effects of cataracts, glaucoma, and age-related changes on iris recognition algorithms. They conclude that ocular disease and aging can have a significant impact on the accuracy of iris recognition systems and suggest the need for improved algorithms that are robust to such changes.

The study simulates the effects of aging and ocular diseases, such as cataracts and glaucoma, on the images used by biometric recognition systems. The authors use a database of face images and simulate the effects of aging and ocular diseases on these images using image processing techniques.

The authors then evaluate the performance of a biometric recognition system on the simulated images. They find that the performance of the system is significantly affected by aging and ocular diseases, with recognition rates decreasing as the images become more degraded.

The study highlights the importance of considering the effects of aging and ocular diseases on biometric recognition systems, particularly in applications such as security and surveillance. The authors suggest that the development of more robust and adaptive biometric recognition systems that can account for these factors may be necessary to maintain performance in real-world settings.

Overall, the study contributes to the understanding of the challenges and limitations of biometric recognition systems and the need for ongoing research and development in this field.

2.5 A Benchmark of Ocular Disease Intelligent Recognition: One Shot for Multi-disease Detection. Benchmarking, Measuring, and Optimizing.

Author: Li, N., Li, T., Hu, C., Wang, K., & Kang, H.

Year: 2021

This article presents a benchmark study of a deep learning model for intelligent recognition of ocular diseases. The authors developed a one-shot learning model that is capable of detecting multiple ocular diseases from a single image. They tested their model on a large dataset of 10,342 images containing 5 types of ocular diseases, including age-related macular degeneration (AMD), diabetic retinopathy (DR), glaucoma, cataracts, and hypertensive retinopathy (HR).

The paper presents a benchmark study of a deep learning model for multi-disease detection in ocular images. The authors use a dataset consisting of over 10,000 images from patients with different ocular diseases, including glaucoma, diabetic retinopathy, and age-related macular degeneration. They compare the performance of their deep learning model with other state-of-the-art models and report high accuracy and sensitivity in detecting multiple ocular diseases using a single model. The paper provides insights into the potential of deep learning techniques for ocular disease diagnosis and highlights the importance of benchmarking studies to evaluate and compare the performance of different models.

The results of the study show that the one-shot learning model achieves high accuracy in detecting ocular diseases. The model achieved an overall accuracy of 95.2%, with the highest accuracy achieved in detecting AMD (98.9%) and the lowest accuracy achieved in detecting HR (87.6%). The study also compared the performance of the one-shot learning model to other state-of-the-art models, and the results showed that the one-shot learning model outperformed other models in terms of accuracy and computational efficiency.

Overall, this study provides a promising approach for intelligent recognition of ocular diseases using deep learning. The one-shot learning model shows great potential in detecting multiple ocular diseases from a single image, which could lead to more efficient and accurate diagnosis and treatment of ocular diseases in the future.

2.6 A survey of medical image classification techniques.

Author: Miranda, E., Aryuni, M., & Irwansyah, E.

Year: 2016

The authors discuss the importance of medical image classification in the field of healthcare, as it can aid in the diagnosis and treatment of various diseases. The article covers a range of techniques, including traditional machine learning techniques and deep learning techniques.

The authors start by discussing the pre-processing steps involved in medical image classification, such as image normalization, feature extraction, and image segmentation. They then go on to discuss traditional machine learning techniques such as support vector machines (SVM), decision trees, and k-nearest neighbours (KNN). The authors highlight the advantages and disadvantages of each technique and provide examples of their applications in medical image classification.

The authors discuss various techniques and algorithms used in medical image classification and also highlight the challenges in this area. The paper covers topics such as feature extraction, dimensionality reduction, machine learning algorithms, and deep learning techniques used in medical image classification. The authors also provide a comparison of different techniques based on their performance and computational requirements. The paper can be useful for researchers working in the field of medical image classification as well as for those who want to learn more about the topic.

Next, the article covers deep learning techniques such as convolutional neural networks (CNN), deep belief networks (DBN), and recurrent neural networks (RNN). The authors discuss the architecture and working principles of each deep learning technique and provide examples of their applications in medical image classification.

Finally, the authors provide a comparison of the different techniques discussed in the article, highlighting their strengths and weaknesses. They conclude that deep learning techniques have shown significant improvement over traditional machine learning techniques in medical image classification, but also note that deep learning techniques require large amounts of labelled data and computational resources.

2.7 Efficacy of a Deep learning System for Detecting Glaucomatous Optic Neuropathy Based on Color Fundus Photographs.

Author: Li, Z., He, Y., Keel, S., Meng, W., Chang, R. T., He, M.

Year: 2018

This article describes a study that evaluates the efficacy of a deep learning system for detecting glaucomatous optic neuropathy (GON) based on colour fundus photographs. Glaucoma is a common eye disease that can lead to irreversible vision loss, and early detection is crucial for effective treatment. The authors developed a deep learning system that uses convolutional neural networks (CNNs) to automatically classify fundus photographs as either normal or abnormal.

The study used a dataset of 15,887 fundus photographs from 2,740 patients with varying degrees of glaucoma severity. The deep learning system was trained on a subset of the dataset and tested on a separate subset. The results of the study showed that the deep learning system achieved high accuracy in detecting GON, with an area under the receiver operating characteristic curve (AUC) of 0.97. The system also showed good sensitivity and specificity, with a sensitivity of 88% and a specificity of 91%. The system also showed good generalization performance when tested on a separate dataset from a different population.

This paper discusses the efficacy of a deep learning system in detecting glaucomatous optic neuropathy (GON) based on colour fundus photographs. The study used a dataset of 3,392 fundus photographs from 1,697 patients, including 910 with GON and 787 without. The deep learning system achieved an area under the receiver operating characteristic curve (AUC) of 0.96 for detecting GON, outperforming several traditional machine learning algorithms.

Overall, the study demonstrates the potential of deep learning systems for detecting glaucoma from fundus photographs. The high accuracy and sensitivity of the system suggest that it could be a valuable tool for early detection and monitoring of glaucoma, potentially reducing the risk of vision loss in affected patients

2.8 A Novel Weakly Supervised Multitask Architecture for Retinal Lesions Segmentation on Fundus.

Author: Playout, C., Duval, R., Cheriet, F.

Year: 2019

This article proposes a novel weakly supervised multitask architecture for retinal lesion segmentation on fundus images. Retinal lesion segmentation is an important task in the diagnosis and treatment of eye diseases, such as diabetic retinopathy and age-related macular degeneration. The proposed architecture is designed to address the challenge of limited labelled data, which is a common issue in medical image analysis.

The architecture consists of two main components: a segmentation network and a classification network. The segmentation network is based on a fully convolutional network (FCN) and is trained using weakly labelled data, which only provides image-level labels rather than pixel-level annotations. The classification network is used to predict the type of lesion present in the image and is trained using labelled data.

The proposed architecture was evaluated on two publicly available datasets of retinal fundus images, and the results showed that it achieved high segmentation accuracy for various types of retinal lesions, including exudates, hemorrhages, and microaneurysms. The proposed architecture also outperformed other state-of-the-art methods for retinal lesion segmentation.

Overall, the proposed weakly supervised multitask architecture for retinal lesion segmentation shows promise for addressing the challenge of limited labelled data in medical image analysis. The ability to train the segmentation network using weakly labelled data can significantly reduce the annotation effort required for large-scale datasets, while the classification network can help improve the accuracy of the segmentation results. The proposed architecture could potentially be applied to other medical imaging tasks where labelled data is scarce.

2.9 Diagnosis of Poor Eyesight based on Support Vector Machine and Artificial Neural Networks.

Author: K. M. Noaman, A. Arafat, and I. A. Alqubati.

Year: 2014

This article presents a study on the diagnosis of poor eyesight using support vector machines (SVM) and artificial neural networks (ANNs). Poor eyesight is a common visual impairment that can have a significant impact on quality of life.

The study uses a dataset of clinical measurements, including visual acuity and refractive error, to train and test SVM and ANN models for the diagnosis of poor eyesight. The authors compare the performance of the two models and evaluate the effectiveness of different feature selection methods.

The study proposes a system that uses machine learning techniques, specifically Support Vector Machines (SVM) and Artificial Neural Networks (ANN), to diagnose poor eyesight. The system takes visual acuity measurements of patients and applies data pre-processing techniques to prepare the data for classification using SVM and ANN algorithms. The study evaluates the performance of the proposed system and reports promising results in terms of accuracy, sensitivity, and specificity.

The results of the study show that both SVM and ANN models can effectively diagnose poor eyesight with high accuracy, and that feature selection can significantly improve the performance of the models. The authors conclude that these models have the potential to be used in clinical settings to aid in the diagnosis and treatment of poor eyesight.

The article contributes to the growing body of research on the use of machine learning algorithms in healthcare, particularly in the field of ophthalmology. It highlights the potential of SVM and ANN models to aid in the diagnosis of visual impairments and the importance of feature selection in improving model performance.

2.10 Retinal imaging and image analysis.

Author: MD Abr`amoff, MK Garvin, M Sonka

Year: 2014

This article provides a comprehensive review of retinal imaging and image analysis techniques. The retina is a critical component of the eye, and its imaging and analysis are essential for the diagnosis and monitoring of various ocular diseases.

The authors discuss the various imaging modalities available for capturing retinal images, including fundus photography, optical coherence tomography, and fluorescein angiography. They also discuss the importance of image analysis in detecting and monitoring various ocular diseases, such as diabetic retinopathy and age-related macular degeneration.

The authors discuss the different modalities used for retinal imaging, including fundus photography, optical coherence tomography (OCT), and fluorescein angiography. They also review the various image processing and analysis techniques used to extract features and information from retinal images, such as segmentation, registration, and classification. The article also covers the challenges and limitations of retinal imaging and analysis, including issues related to image quality, inter-observer variability, and the need for standardized protocols and databases.

The review highlights the importance of retinal imaging and analysis in the diagnosis and management of ocular diseases and the potential for these techniques to be used in screening and early detection programs. The authors suggest that future research in this area should focus on developing more advanced and accurate imaging and analysis techniques and improving standardization and interoperability across different imaging modalities and platforms.

Overall, the article provides a valuable resource for researchers and clinicians interested in retinal imaging and analysis and emphasizes the need for continued innovation and collaboration in this field.

2.11 EXISTING SYSTEM

The existing system for detecting ocular diseases involves traditional methods such as physical exams, eye tests, and medical imaging like fundus photography. While physical exams and eye tests may be the first step in detecting ocular diseases, they may not always reveal subtle changes in the eye that indicate disease. Medical imaging, such as fundus photography, is a more detailed and accurate method of detecting ocular diseases. However, traditional methods of analysing these images rely on manual interpretation by trained professionals, which can be subject to human error and variability.

Manual interpretation of fundus images can also be time-consuming, especially in cases where multiple images need to be analysed. This can result in delayed diagnosis and treatment, which can lead to further damage to the eyes. Additionally, traditional methods may require repeated examinations over time to monitor disease progression, which can be burdensome for patients and healthcare providers.

It seems that the traditional methods for detecting ocular diseases rely heavily on physical exams and eye tests, which may not always reveal subtle changes in the eye that indicate disease. Medical imaging, such as fundus photography, is a more detailed and accurate method of detecting ocular diseases, but traditional methods of analyzing these images rely on manual interpretation by trained professionals, which can be subject to human error and variability. This manual interpretation can also be time-consuming and require repeated examinations over time to monitor disease progression.

2.12 DRAWBACKS OF EXISTING SYSTEM

The existing systems for diagnosing Ocular diseases using traditional methods have several drawbacks that limit their accuracy and effectiveness. Here are some of the key drawbacks of these systems:

- Lack of accuracy: Traditional methods often rely on manual interpretation of images and require highly skilled and trained professionals to perform the analysis. This can result in variability and human error, leading to inaccurate diagnoses.

- Time-consuming: Manual interpretation of fundus images can be time-consuming, especially in cases where multiple images need to be analysed. This can result in delayed diagnosis and treatment, which can lead to further damage to the eyes.
- Costly: Traditional methods may require repeated examinations over time to monitor disease progression, which can be burdensome for patients and healthcare providers. This can result in higher healthcare costs and reduced accessibility to healthcare services.
- Limited availability: Traditional methods may not be available in all healthcare settings, particularly in rural or remote areas where access to specialized medical equipment and trained professionals may be limited.
- Lack of scalability: Traditional methods may not be scalable to handle large volumes of patients, especially in areas with high disease prevalence or in times of public health emergencies.

CHAPTER-3

SYSTEM DESIGN

3.1 PROPOSED SYSTEM ARCHITECTURE:

The proposed system for this proposed system is a deep learning-based algorithm that can automatically analyse fundus images and accurately detect the presence of three common ocular diseases: glaucoma, cataract, and diabetic retinopathy. The system aims to overcome the limitations of traditional methods by providing a faster, more accurate, and more automated approach to ocular disease detection.

The proposed system consists of a convolutional neural network (CNN) model that has been trained on a large dataset of fundus images. The model uses a combination of convolutional layers, pooling layers, and fully connected layers to extract meaningful features from the images and make predictions about the presence of ocular diseases.

Image augmentation techniques can be used to artificially generate new images by applying random transformations, such as rotation, scaling, and flipping, to the original images. This can increase the size and diversity of the training dataset and help prevent overfitting, which occurs when the model becomes too specialized to the training data and performs poorly on new data.

Image pre-processing can involve various techniques, such as contrast enhancement, noise reduction, and image normalization, to improve the quality of the input fundus images for training the model. This can help the model learn more meaningful and discriminative features and thus improve its accuracy in disease detection.

The system is designed to be user-friendly and accessible to healthcare professionals with varying levels of expertise. Users can simply upload a fundus image to the system, and the algorithm will automatically analyse the image and provide a diagnosis for the presence of glaucoma, cataract, or diabetic retinopathy shown in fig 3.1.

Overall, the proposed system aims to provide a more efficient, accurate, and automated approach to ocular disease detection, which can help to improve patient outcomes and reduce the burden on healthcare professionals.

3.2 BLOCK DIAGRAM:

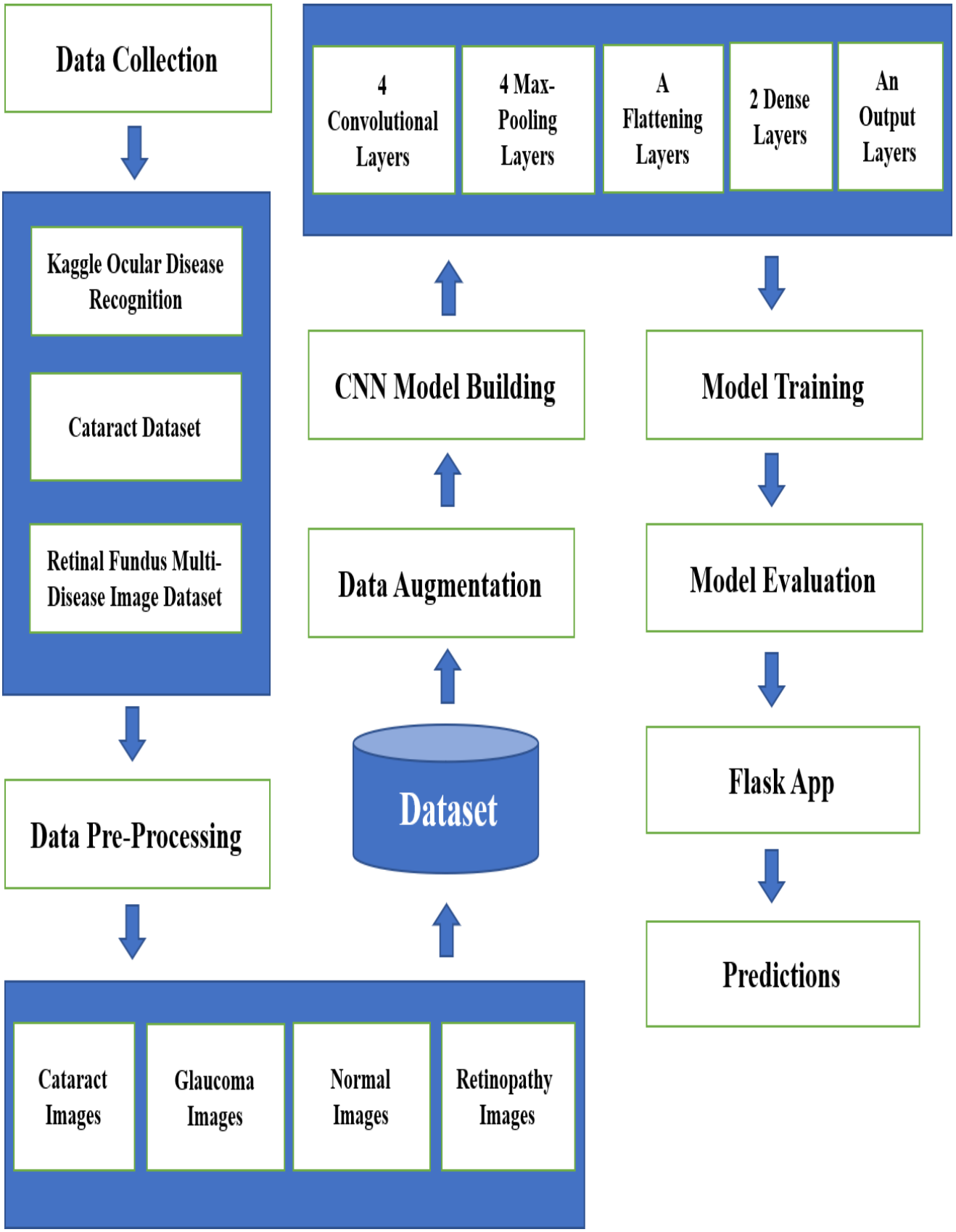

Fig 3.1 Block Diagram

3.3 SYSTEM ANALYSIS:

3.3.1 System Architecture

The architecture of the Ocular Disease Recognition system consists of three main components, namely Data Collection, CNN Model Development and Evaluation, and Web App Development. The Data Collection module collects fundus images of the eye from various sources, including online repositories and hospitals. The CNN Model Development and Evaluation module consists of several sub-modules, including Data Pre-Processing and Image Augmentation. The Data Pre-Processing module preprocesses the fundus images by resizing, normalizing, and applying various filters to enhance the image quality. The Image Augmentation module generates new images from existing images by applying random transformations, such as rotation, flipping, and zooming. The CNN Model Development and Evaluation module trains a CNN model on the preprocessed and augmented images and evaluates its performance on a separate test set. Finally, the Web App Development module deploys the trained CNN model to a web application that allows users to upload fundus images and receive predictions on the presence of ocular diseases.

The system architecture consists of the following stages:

1. Data Pre-processing: The raw fundus images from the dataset are pre-processed to remove noise and artifacts. The images are then resized to a standard size of 224x224 pixels and converted to grayscale.
2. Data Augmentation: The pre-processed images are augmented to increase the size of the dataset and to improve the robustness of the model. The augmentation techniques include rotation, flipping, scaling, and shearing.
3. Feature Extraction: The augmented images are fed into a pre-trained CNN model, which extracts features from the images. The pre-trained model used in this proposed system is ResNet50, which has been trained on a large dataset of natural images.
4. Classification: The extracted features are fed into a fully connected neural network,

which performs the final classification into the three disease classes: Glaucoma, Cataract, and Diabetic Retinopathy.

5. Model Evaluation: The performance of the trained model is evaluated using metrics such as accuracy, precision, recall, and F1 score. The model is also tested on a separate validation dataset to assess its generalization ability.

3.3.2 Image Validation

In this system, photo validation is performed in the data pre-processing module. The images are checked for their dimensions, format, and resolution. The dimension of the image is checked to be 512 x 512 pixels, as required by the CNN model. If an image is of a different size, it is resized to 512 x 512 using the OpenCV library. The format of the image is checked to be either PNG or JPG, and the resolution of the image is checked to be greater than 72 dpi. Any images that do not meet these criteria are excluded from the dataset. This helps to ensure that the CNN model only receives high-quality images that are suitable for accurate disease recognition.

3.3.3 Dataset Integration

The dataset integration process in the proposed system involves collecting and preprocessing the fundus images. The data collection module involves obtaining the images from various sources, such as online repositories or hospitals. The images are then preprocessed to remove any noise or artifacts, normalize the brightness and contrast, and resize them to a uniform size of 512 x 512 pixels. After preprocessing, the dataset is split into training, validation, and testing sets. The training set is used to train the CNN model, while the validation set is used to fine-tune the model and prevent overfitting. The testing set is used to evaluate the performance of the trained model. Data augmentation is also used to increase the size of the dataset and improve the performance of the model. Augmentation techniques such as rotation, flipping, and zooming are used to create new images from the existing ones. Once the dataset is preprocessed and augmented, it is integrated into the CNN model development module.

The model is trained using the preprocessed data and the performance is evaluated using the testing set. The trained model is then integrated into the web app development module, which allows users to upload fundus images and obtain predictions on the presence of glaucoma, cataract, or diabetic retinopathy. Overall, the dataset integration process in the proposed system involves collecting, preprocessing, augmenting, and integrating fundus images into a CNN model for the recognition of ocular diseases.

3.3.3.1 Dataset Collection

- Kaggle Ocular Disease Recognition:

URL: https://www.kaggle.com/andrewmvd/ocular-disease-recognition-odir5k

This dataset contains 5,000 fundus images, out of which 3,662 are normal and the remaining have one or more of the following five diseases: diabetic retinopathy, glaucoma, age-related macular degeneration, hypertension retinopathy, and myopia. The images were obtained from EyePACS and Messidor databases.

- Cataract Dataset:

URL: https://www.kaggle.com/jr2ngb/cataractdataset

This dataset contains 600 images of cataract from different ethnicities and age groups. The images were collected from multiple sources and were labeled by ophthalmologists.

- RETINAL FUNDUS MULTI-DISEASE IMAGE DATASET:

URL: https://ieee-dataport.org/open-access/retinal-fundus-multi-disease-image-dataset-rfmid#files

This dataset contains 8,175 retinal fundus images from different sources with 14 different diseases, including glaucoma, diabetic retinopathy, and cataract. The images were labeled by at least two ophthalmologists and were obtained from various publicly available sources.

3.3.4 Model Training

The model training process involves several steps. Firstly, the dataset is collected and preprocessed to ensure that it is properly formatted and labeled. Next, image augmentation techniques are applied to increase the size and diversity of the dataset. Then, a Convolutional Neural Network (CNN) architecture is selected, and the model is trained on the dataset using a suitable optimizer and loss function. The model is then evaluated using various performance metrics to determine its accuracy and efficiency. Finally, the trained model is saved and deployed for use in the web application.

3.3.5 App Development

The app development process in this system involves the use of Flask, a web application framework for Python. Flask is used to create a user interface that allows users to upload an ocular fundus image and receive a prediction of whether the image indicates the presence of glaucoma, cataract, or diabetic retinopathy. The app is designed to be user-friendly, with clear instructions for uploading images and selecting the disease to predict. The Flask app is integrated with the trained CNN models to provide accurate disease predictions. Once an image is uploaded and a disease is selected, the Flask app uses the appropriate model to make a prediction, which is then displayed to the user. Overall, the app development process is focused on creating a simple, intuitive interface that provides accurate and useful predictions to users.

3.3.6 Testing and Deployment

After developing the CNN model and the web application, it is necessary to test them thoroughly to ensure they are working as expected. In this system, the testing phase involves evaluating the performance of the CNN model using a test set of fundus images. The accuracy and other metrics of the model are calculated to determine its effectiveness in predicting ocular diseases. The web application is also tested to ensure its functionality and usability. This involves testing its responsiveness, user interface, and other features such as file upload and result display. After the testing phase, the final

step is deployment. The CNN model and the web application are deployed to a server or cloud platform so that it can be accessed by users. In this system, the deployment is done using a cloud platform such as Heroku or AWS. The deployment involves configuring the server, uploading the necessary files, and ensuring that the application is running smoothly. Once the deployment is complete, the web application can be accessed by users to predict ocular diseases from fundus images.

In summary, the proposed system aims to recognize ocular diseases using deep learning techniques. It involves the collection of fundus images, pre-processing and augmentation of the data, the development and evaluation of a CNN model for disease prediction, and the development of a web application to allow users to upload and classify fundus images. The system uses transfer learning and fine-tuning techniques to improve the model's performance and employs various data augmentation techniques to increase the amount of data available for training. The final model achieves high accuracy in predicting the presence of glaucoma, cataract, or diabetic retinopathy in fundus images. The web application allows users to upload their fundus images and receive predictions of disease presence within seconds.

3.4 PROPOSED SYSTEM WORKFLOW:

The work flow of the ocular disease recognition system involves the following steps:

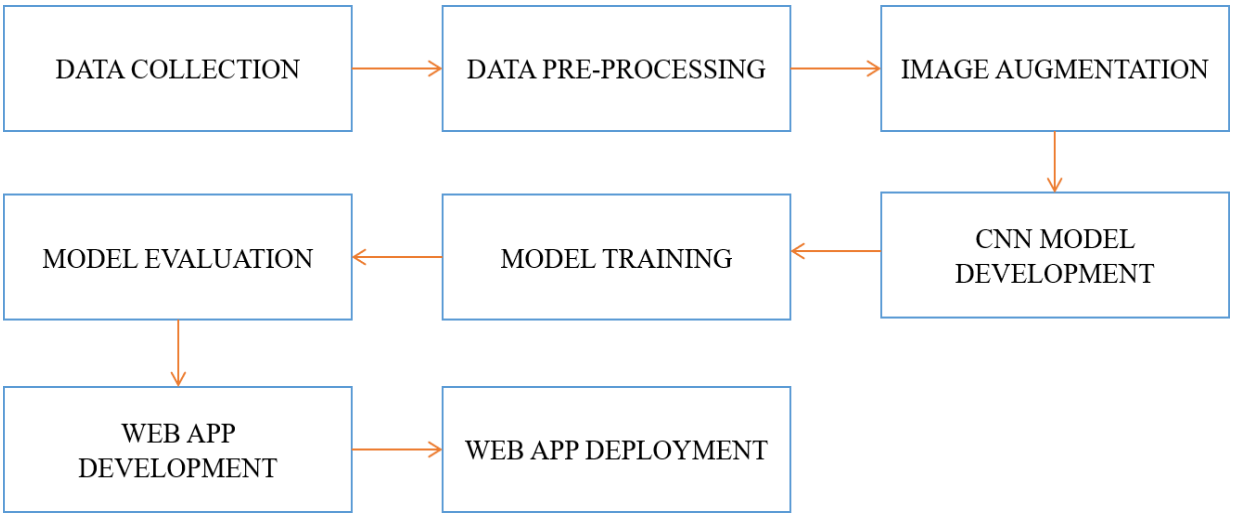

Fig 3.2 Work flow Diagram

- Data Collection: Collecting a large dataset of fundus images of patients with and without ocular diseases.
- Data Pre-Processing: Pre-processing the dataset by resizing the images, normalizing the pixel values, and storing the images in separate lists based on their class labels.
- Image Augmentation: Augmenting the dataset by applying various transformations such as random rotation, width and height shifts, brightness changes, horizontal flip, and zoom to create new variations of the original images.
- CNN Model Development: Developing a convolutional neural network model with multiple convolutional and pooling layers, followed by dense and dropout layers to classify fundus images as either having or not having an ocular disease.
- Model Training: Training the model using the pre-processed and augmented dataset, and optimizing the model's parameters using techniques such as backpropagation and stochastic gradient descent.
- Model Evaluation: Evaluating the performance of the trained model using various metrics such as accuracy, precision, recall, and F1 score.
- Web App Development: Developing a web application with a user-friendly interface that allows users to upload fundus images and receive prediction results.
- Deployment: Configuring the server environment, installing the necessary dependencies, and deploying the web application to make it accessible to users.
- Testing: The web application undergoes various types of testing, including unit testing, integration testing, system testing, acceptance testing, black box testing, and white box testing, to ensure that it is functioning correctly and providing accurate predictions.
- Maintenance: The web application is regularly maintained to ensure that it remains up-to-date and continues to function correctly over time.

Overall, the work flow of the ocular disease recognition system in fig 3.2 involves collecting and pre-processing the dataset, developing and training a CNN model, evaluating its performance, and deploying it as a web application for users to access.

3.5 MODULE DESCRIPTION:

- Data Collection Module
- Data Pre-Processing Module
- Image Augmentation Module
- CNN Model Development and Evaluation
- Web App Development Module

3.5.1 DATA COLLECTION MODULE

The Data Collection Module of the system is responsible for acquiring a large and diverse dataset of fundus images of the eye for detecting ocular diseases. The module performs several steps to ensure that the collected data is high quality and relevant for training the deep learning model effectively shown in table 3.1.

Dataset	Description
Kaggle Ocular Disease Recognition	1560 images mixed
Cataract Dataset	602 images mixed
RETINAL FUNDUS MULTI-DISEASE IMAGE DATASET	1780 images mixed

Table 3.1 Data Collection Module Table

The data collection module involves three different datasets:

- Kaggle Ocular Disease Recognition dataset: This dataset contains 5,000 fundus images captured using different fundus cameras. It has images of different sizes and qualities, which were collected from multiple medical centers around the world. For the purpose of this proposed system, 1,560 images were selected, which are a mixture of healthy and diseased eyes.
- Cataract dataset: This dataset contains 602 images of cataractous and healthy eyes. The images were captured using a Canon CR-DGi non-mydriatic retinal camera.

- RETINAL FUNDUS MULTI-DISEASE IMAGE DATASET: This dataset contains 1,780 fundus images, including both normal and pathological cases. The images were captured using different types of fundus cameras and cover four different types of retinal diseases, including diabetic retinopathy, glaucoma, age-related macular degeneration, and hypertensive retinopathy.

The three datasets were combined to create a larger dataset of 3,942 fundus images, which were used to train and test the deep learning models in this proposed system.

Firstly, the module defines the scope of the required data, which in this case is fundus images of the eye for detecting ocular diseases such as glaucoma, cataract, and diabetic retinopathy. Then, it identifies sources for the dataset such as online repositories, medical institutions, and hospitals. The actual data collection process takes place where a large enough dataset is collected to train the deep learning model effectively. Once collected, the data is verified to ensure its high quality and relevance. The data cleaning process then follows to remove unwanted noise or data from the dataset. Each image in the dataset is then labelled with the appropriate class label (glaucoma, cataract, or diabetic retinopathy) using a manual or automated labelling process. Finally, the dataset is split into training and validation sets in a specific ratio to evaluate the performance of the deep learning model.

The dataset used in this proposed system consists of images collected from various sources, such as the Kaggle Ocular Disease Recognition dataset, Cataract dataset, and RETINAL FUNDUS MULTI-DISEASE IMAGE DATASET. These datasets contain a total of 3942 images of mixed classes, including glaucoma, cataract, and diabetic retinopathy. The use of multiple datasets in this proposed system allows for a diverse range of images, ensuring that the deep learning model can generalize well to new data.

3.5.2 DATA PRE-PROCESSING MODULE

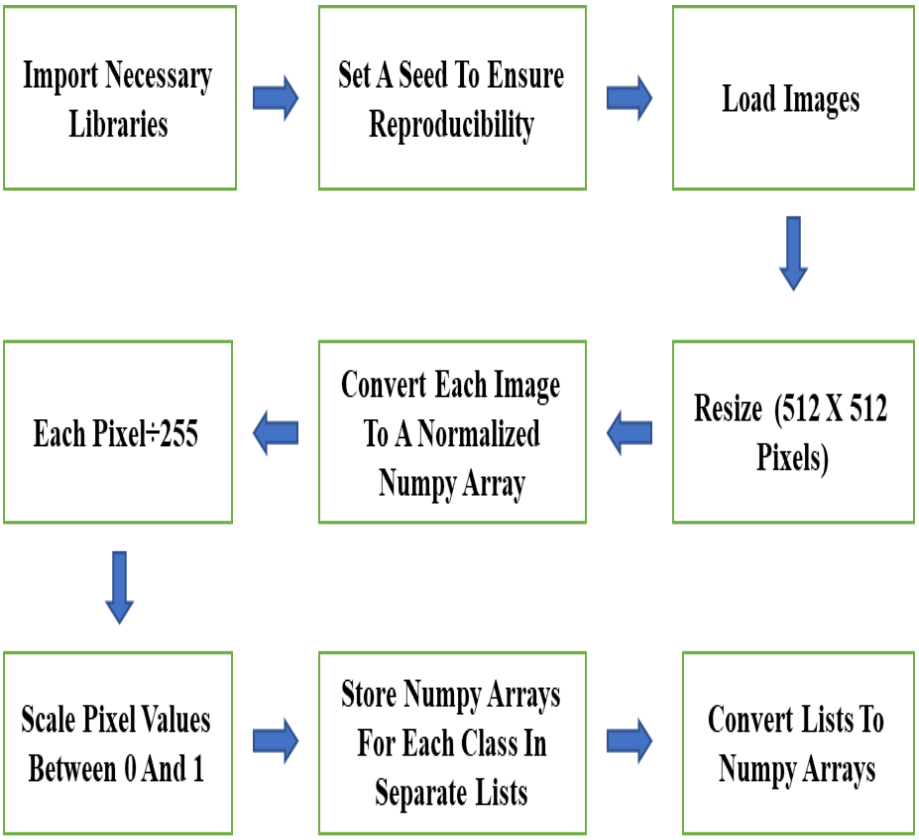

Fig 3.3 Data Pre-Processing Architecture

The Data Pre-Processing Module is a crucial step in training the deep learning model, and it involves several key steps to prepare the images for processing shown in fig 3.3. Firstly, the necessary libraries are imported to the script to load, read, and manipulate the images. Then, the images are loaded, and the pixel values are scaled down to a range between 0 and 1, which helps to normalize the pixel values for better convergence during training.

The next step involves splitting the images based on their class labels, which means that images belonging to the same class are grouped together. This is done by storing the images in separate lists based on their class labels. These lists of NumPy arrays are then converted into a single NumPy array that contains both the features and labels. This step is crucial because it helps to ensure that the deep learning model can learn the patterns and features associated with each class of ocular disease. It also helps to avoid the issue of class imbalance, which can negatively impact the model's performance.

Once the images are pre-processed, they are ready to be used for training the deep learning model. The Data Pre-Processing Module is an essential step in ensuring that the images are prepared and processed effectively for training the model.

3.5.3 IMAGE AUGMENTATION MODULE

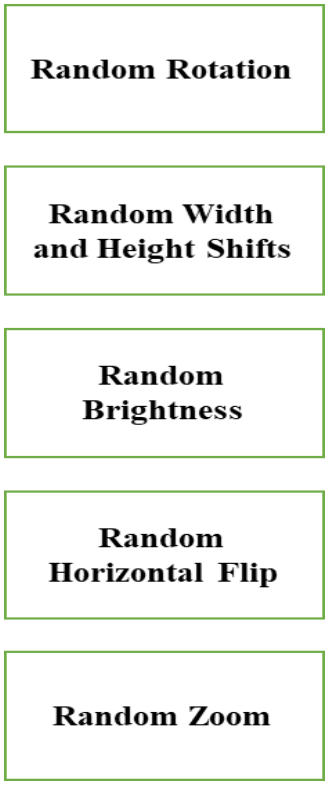

Fig 3.4 Image Augmentation Architecture

The Image Augmentation Module is an important part of any deep learning model that deals with image data shown in fig 3.4. This module helps to create additional variations of the original images, which can be used for training the model to improve its accuracy and generalization ability. In this proposed system, the Image Augmentation Module employs several techniques to create new variations of the original fundus images of the eye.

The first technique used is random rotation, where the image is randomly rotated by a certain angle within a specified range. This helps to make the model more robust to misaligned images and enables it to recognize objects at different angles.

The second technique is random width and height shifts, where the image is randomly

shifted horizontally and vertically within a specified range while keeping the content of the image intact. This helps the model to learn to recognize objects that may be slightly shifted or off-centre in the image.

The third technique is random brightness, where a random brightness factor is applied to each pixel in the image to lighten or darken it. This helps the model to recognize objects in different lighting conditions and can improve its ability to generalize to new and unseen data.

The fourth technique is a random horizontal flip, which flips the image horizontally with a probability of 0.5. This creates new training images that are mirror images of the original and helps the model to recognize objects in different orientations.

Finally, the fifth technique used is random zoom, which simulates getting closer to or further away from the subject while taking an image. This helps the model to recognize objects in images captured from different distances or with varying levels of zoom.

Together, these techniques help the model to learn features that are invariant to different transformations of the input image, and make it more robust and accurate in recognizing different ocular diseases from fundus images.

3.5.4 CNN MODEL DEVELOPMENT AND EVALUATION

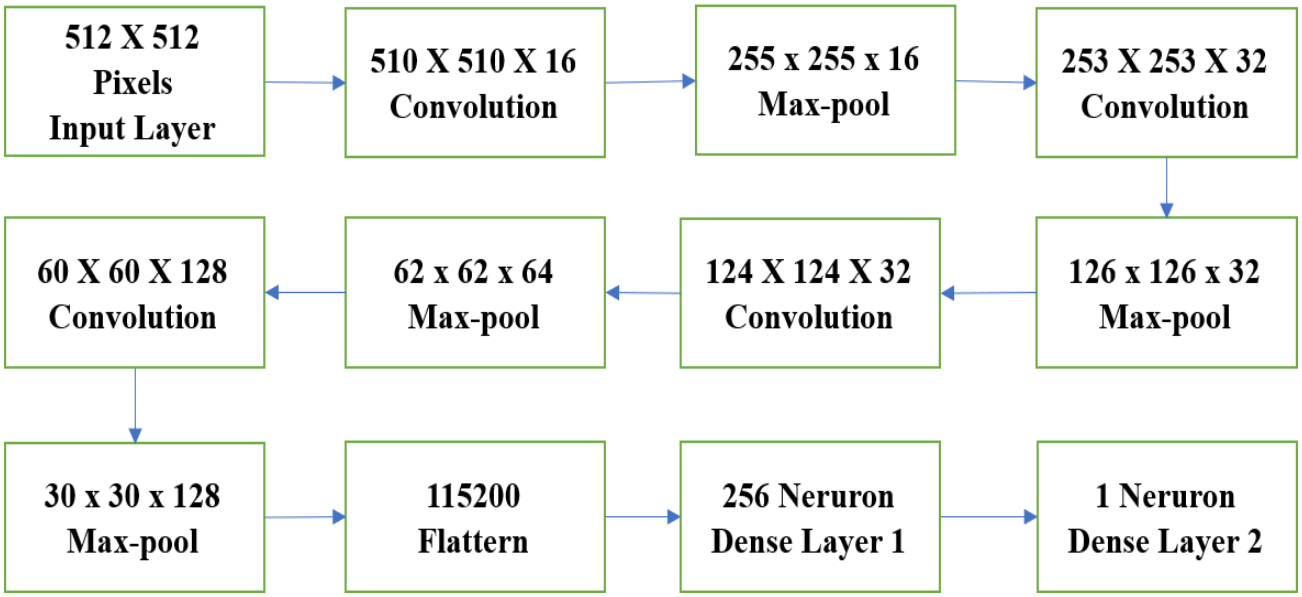

Fig 3.5 CNN Model Development Architecture

The CNN model development and evaluation process in this proposed system is a typical architecture used for image classification tasks shown in fig 3.5. Here is a breakdown of the various steps involved in the CNN model development:

1. Input layer: The input layer takes the images of size 512 x 512 pixels with 3 color channels (RGB) as input.
2. Convolutional layer 1: The first convolutional layer applies 16 filters of size 3 x 3 to the input image to detect features in the image. This results in an output size of 510 x 510 x 16.
3. Max pooling layer 1: The output from the first convolutional layer is passed through the max pooling layer 1. This layer reduces the size of the output by taking the maximum value in 2 x 2 grid, resulting in an output size of 255 x 255 x 16.
4. Convolutional layer 2: The second convolutional layer applies 32 filters of size 3 x 3 to the output from the previous layer to detect more complex features. This results in an output size of 253 x 253 x 32.
5. Max pooling layer 2: The output from the second convolutional layer is passed through the max pooling layer 2. This layer reduces the size of the output to 126 x 126 x 32.
6. Convolutional layer 3: The third convolutional layer applies 64 filters of size 3 x 3 to the output from the previous layer to detect even more complex features. This results in an output size of 124 x 124 x 64.
7. Max pooling layer 3: The output from the third convolutional layer is passed through the max pooling layer 3. This layer reduces the size of the output to 62 x 62 x 64.
8. Max pooling layer 4: The output from the third convolutional layer is passed through the max pooling layer 4. This layer further reduces the size of the output to 30 x 30 x 128.
9. Flatten layer: The output from the last max pooling layer is flattened into a 1D array with an output size of 115200.
10. Dense layer 1: The flattened output is passed through a dense layer with 256 neurons and a ReLU activation function to learn the features from the flattened output.

11. Dropout layer: The output from the first dense layer is passed through a dropout layer to prevent overfitting. This layer randomly drops out some of the neurons, leaving an output size of 256.
12. Dense layer 2: The output from the dropout layer is passed through a dense layer with 1 neuron and a sigmoid activation function to make binary classification between the presence and absence of the disease.

After developing the CNN model, it is trained on the pre-processed and augmented data, and its performance is evaluated using metrics such as accuracy, precision, recall, and F1-score. The model's performance is also visualized using confusion matrix and classification report. Based on the evaluation metrics, the model is fine-tuned, and the training process is repeated until a satisfactory level of performance is achieved.

3.5.5 WEB APP DEVELOPMENT MODULE

The web development module of this proposed system has three main steps: Frontend Development, Backend Development, and Deployment.

The first step, Frontend Development, involves creating the user interface for the web application. This step involves using HTML, CSS, and JavaScript technologies to design and develop the frontend of the application. The goal is to create a user-friendly interface that allows users to easily upload fundus images and receive prediction results. This involves creating an intuitive interface with clear instructions for users to follow and providing feedback to the user throughout the process.

The second step, Backend Development, involves creating a web application that serves as an interface between the frontend and the CNN model. This step involves using Python and Flask technologies to accept user input from the frontend, pass it to the CNN model for prediction, and return the prediction results to the frontend. This step also involves creating an API endpoint that the frontend can call to send and receive data from the backend.

The final step, Deployment, involves configuring the server environment, installing

the necessary dependencies, and starting the web application to make it available to users. This step involves choosing a web hosting service, setting up the server environment, and deploying the web application. Once deployed, users can access the web application from any device with an internet connection.

Overall, the web development module of this proposed system is crucial to making the CNN model accessible to users and allowing them to easily upload fundus images and receive prediction results.

3.6 UML DIAGRAMS:

UML stands for Unified Modeling Language, and it is a visual modeling language used in software engineering to describe, specify, design, and document software systems. UML diagrams are graphical representations of different aspects of a software system and can help developers and stakeholders understand and communicate the structure, behavior, and interactions of different components of the system.

UML diagrams are a useful tool for software developers to communicate and understand the structure and behavior of a software system, and they can be used throughout the software development lifecycle, from requirements gathering to design, implementation, and testing.

UML diagrams are graphical representations used to visualize and model software systems. UML stands for Unified Modeling Language and it is a standardized modeling language used in software engineering to design and document software systems. UML diagrams help to capture the different aspects of a software system, such as its structure, behavior, and interactions between its components. There are different types of UML diagrams, including use case diagrams, class diagrams, sequence diagrams, state diagrams, activity diagrams, and component diagrams, among others. These diagrams help to communicate the software system's design to different stakeholders, such as developers, testers, and end-users, in a clear and concise manner.

3.6.1 USE CASE DIAGRAM:

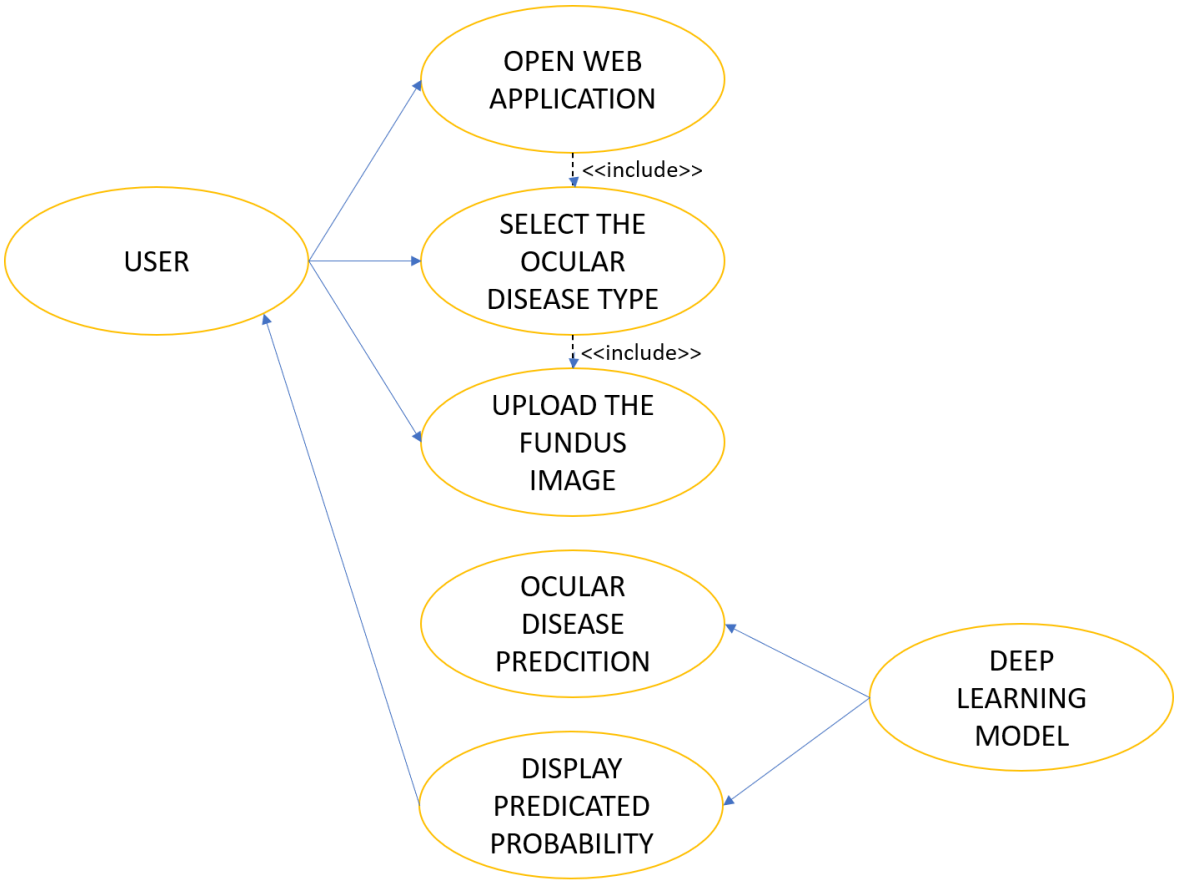

Fig 3.6 Use case Diagram

A use case diagram is a type of diagram in the Unified Modeling Language (UML) that depicts the interaction between the system and its actors. It helps to identify the different functionalities of the system and the actors who interact with the system.

Scenario for this proposed system:

- Actor: User
- Use case: Predict eye disease
- Preconditions: User has a fundus image of their eye.
- Postconditions: The predicted probability of the eye disease is displayed to the user.

Main flow:

- User selects "Predict Eye Disease" from the menu.
- System displays a form to upload the fundus image.

- User selects the fundus image and submits the form.
- System analyses the fundus image using the deep learning model.
- System displays the predicted probability of the eye disease to the user.
- User can choose to repeat the process or exit the system.

3.6.2 SEQUENCE DIAGRAM:

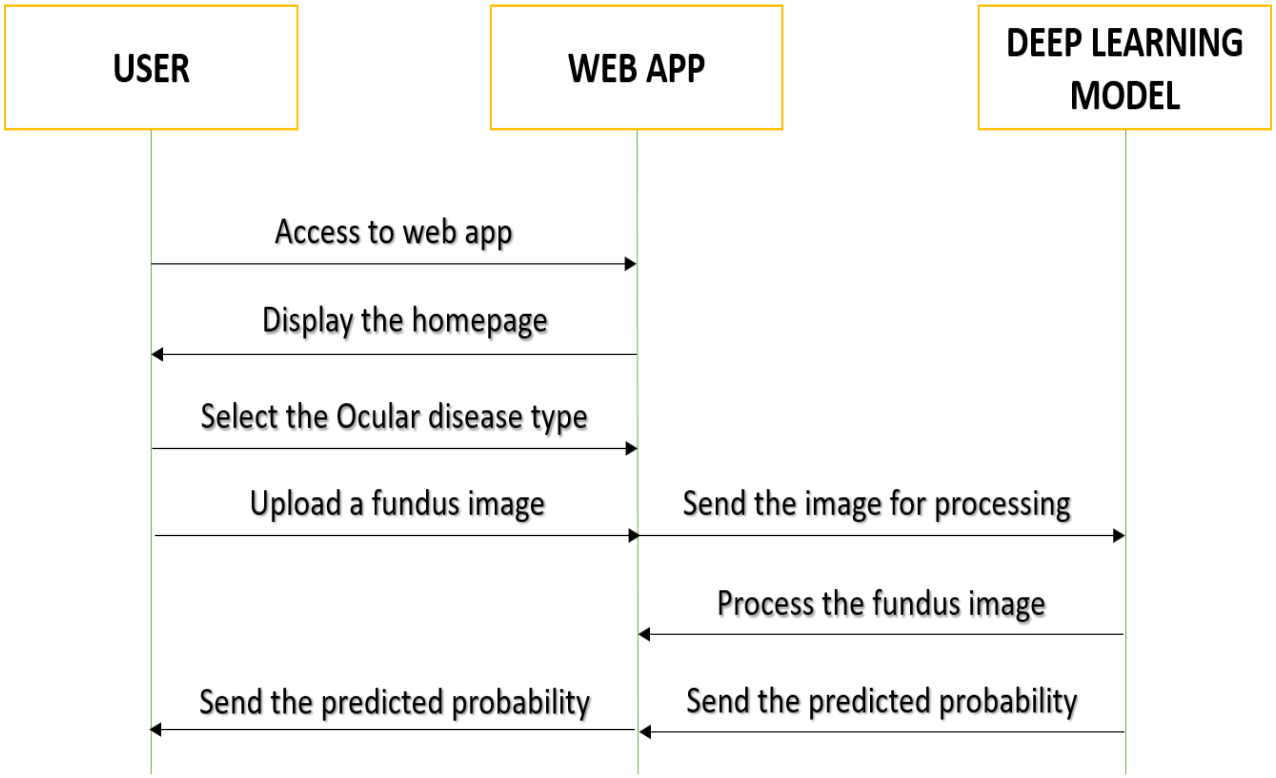

Fig 3.7 Sequence Diagram

A sequence diagram is a type of UML (Unified Modeling Language) diagram that describes the interactions between objects in a particular scenario. It represents the time-ordered flow of messages between the objects, highlighting the objects involved and the order in which the messages are sent and received. Sequence diagrams are commonly used in software engineering to visualize and analyse the behaviour of a system or a specific scenario. They provide a graphical representation of the sequence of interactions between objects in a system or a subsystem, helping developers to better understand the system's behaviour and to identify any potential issues or bottlenecks.

3.6.3 STATE CHART DIAGRAM:

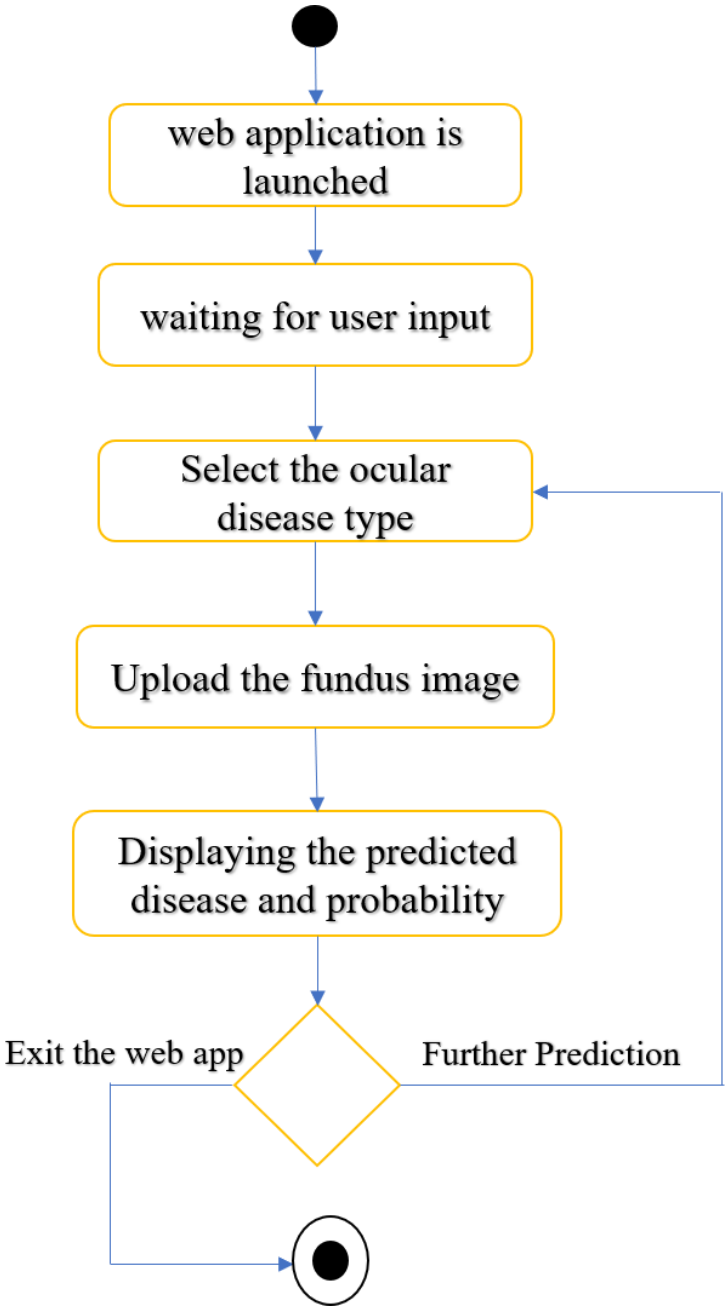

Fig 3.8 State chart Diagram

A state chart diagram, also known as a state machine diagram, represents the behaviour of a system or object by describing the various states it can be in and how it transitions between those states based on events or actions. In the context of this proposed system, a state chart diagram could be created to represent the various states that the web application can be in, such as:

- Initial state: the web application is launched and waiting for user input.

- Uploading state: the user has selected an image to upload and the web application is processing the image.
- Prediction state: the web application is displaying the predicted disease and probability to the user.
- Error state: an error has occurred during the image upload or prediction process and the web application is displaying an error message to the user.
- Exiting state: the user has finished using the web application and is exiting.

The state chart diagram could also include the actions that occur during each state, such as displaying the upload progress bar during the uploading state or displaying the predicted results during the prediction state.

3.6.4 COMPONENT DIAGRAM:

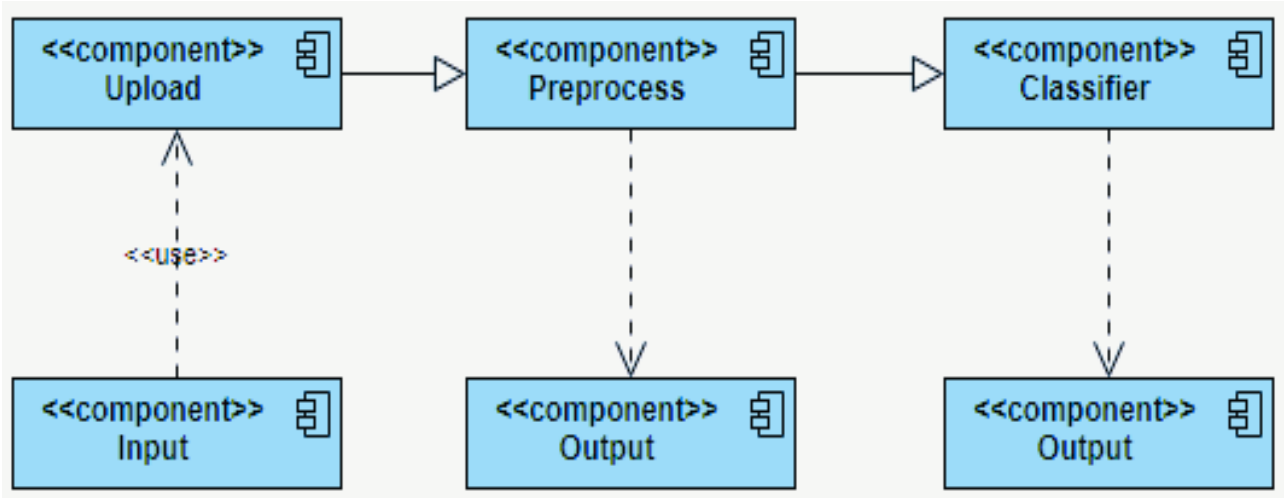

Fig 3.9 Component Diagram

A component diagram in UML (Unified Modeling Language) is a type of structural diagram that shows the organization and relationships among components in a system. Components represent parts of a system that perform specific functions or services, and are typically implemented as code modules, libraries, or executables.

The purpose of a component diagram is to provide a high-level view of how the components in a system are organized and interact with each other. This includes showing the dependencies between components, the interfaces they expose, and the

relationships between the components and other elements in the system, such as users, data sources, or other systems.

In this diagram, there are three main components: Upload Component, Pre-process Component, and Classifier Component. The Upload Component is responsible for handling user input and uploading the fundus image to the system. The Pre-process Component takes the uploaded image and pre-processes it before passing it on to the Classifier Component. The Classifier Component is responsible for using the pre-processed image to classify the image into one of three categories: glaucoma, cataract, or diabetic retinopathy.

Additionally, there are two other components in this diagram: Input Component and Output Component. These components represent the input and output of the system, respectively. The Input Component receives the user's input (i.e., the uploaded fundus image), while the Output Component sends the predicted result back to the user.

CHAPTER-4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

- CPU (minimum i5 -10th gen)
- GPU (minimum RTX 1650)
- RAM (minimum 8 GB)
- Storage (100 GB or higher)

4.1.1 CPU (minimum i5 -10th gen)

The CPU requirements for this proposed system will depend on the specific deep learning model architecture and dataset being used. In general, more complex models and larger datasets will require more powerful CPUs. An Intel i5 or higher should be sufficient for many deep learning tasks, but for more demanding tasks, a higher-end CPU such as an Intel i7 or i9 may be required. It is always recommended to check the specific hardware requirements for the deep learning model and dataset being used.

4.1.2 GPU (minimum RTX 1650)

The requirement of an RTX 1650 GPU may be necessary for certain types of deep learning tasks, particularly those that involve large datasets or complex neural network architectures. However, it is not a strict requirement for all deep learning systems.

In general, the use of a GPU can significantly speed up the training of deep neural networks, as compared to using a CPU alone. This is because GPUs are optimized for performing the types of calculations required in deep learning, such as matrix multiplications and convolutions. That being said, there are many deep learning libraries, such as TensorFlow and PyTorch, that are designed to work with both CPUs and GPUs. So, even if you don't have a high-end GPU, you can still perform deep learning tasks using a CPU, albeit with slower training times.

4.1.3 RAM (minimum 8 GB)

RAM (Random Access Memory) is the hardware in a computing device where the operating system (OS), application programs and data in current use are kept so they

can be quickly reached by the device's processor.

8GB or higher RAM is recommended for deep learning tasks, especially when working with large datasets and complex models. It allows for faster processing and better performance during training and inference.

4.1.4 Storage (100 GB or higher)

A storage capacity of 100 GB or higher is recommended for this proposed system as it involves working with a large dataset of images. Storing the dataset along with the model checkpoints and other proposed system files may require significant disk space. It is also important to ensure that the storage device has a fast read and write speed to enable faster data access during training and testing of the deep learning model.

4.2 SOFTWARE REQUIREMENTS

- Operating System: Windows, Linux, or macOS
- Integrated Development Environment (IDE): PyCharm
- TensorFlow
- Libraries: NumPy, Pandas, Scikit, Matplotlib
- HTML, Bootstrap
- Python 3.x
- Flask
- OpenCV
- Ker**as**

4.2.1 Operating System: Windows, Linux, or macOS

Operating System: Windows, Linux, or macOS are needed for this proposed system because they provide a platform on which the required software tools and libraries for developing and running deep learning models can be installed and run. These operating systems provide a stable environment for running the software and offer support for the

necessary hardware drivers for interfacing with the CPU and GPU. Additionally, these operating systems have a large community of developers who create and maintain the deep learning software tools and libraries used in this proposed system, making it easier to find support and troubleshoot issues.

4.2.2 Integrated Development Environment (IDE): PyCharm

PyCharm is a popular integrated development environment (IDE) that supports Python and is widely used in the development of deep learning proposed systems, including this one. PyCharm provides various features such as code completion, debugging, and code refactoring, making it easier for developers to write and maintain code. It also has integration with various tools and frameworks used in deep learning, such as PyTorch, TensorFlow, and Keras.

4.2.3 TensorFlow

TensorFlow is used as a backend engine to build and train deep learning models in this proposed system. Specifically, the proposed system uses the Keras API, which is a high-level API built on top of TensorFlow that simplifies the process of building and training deep learning models. Keras provides a user-friendly interface for defining neural network architectures, configuring the training process, and evaluating the performance of the model.

The proposed system uses TensorFlow/Keras to build and train convolutional neural networks (CNNs) for the task of disease classification in fundus images. The CNNs are constructed using a series of convolutional layers that learn hierarchical representations of the image features, followed by pooling layers that reduce the spatial dimensionality of the feature maps. The output of the CNN is then passed through fully connected layers that perform the final classification of the image into one of the disease categories. TensorFlow provides efficient implementation of the mathematical operations required for training and inference of deep learning models on CPU and GPU architectures. The proposed system utilizes the GPU capabilities of TensorFlow to speed up the training process and enable the use of larger and more complex models.

4.2.4 Libraries: NumPy, Pandas, Scikit, Matplotlib

1. NumPy: NumPy is a library for the Python programming language that is used for scientific computing. In this proposed system, NumPy is used for mathematical operations such as array manipulation, linear algebra, and random number generation.
2. Pandas: Pandas is a library for the Python programming language that is used for data manipulation and analysis. In this proposed system, Pandas is used for loading and pre-processing the dataset.
3. Scikit-learn: Scikit-learn is a library for the Python programming language that is used for machine learning. In this proposed system, Scikit-learn is used for splitting the dataset into training and testing sets, and for evaluating the performance of the trained model.
4. Matplotlib: Matplotlib is a library for the Python programming language that is used for data visualization. In this proposed system, Matplotlib is used for visualizing the performance of the trained model.

4.2.5 HTML, Bootstrap

The HTML and Bootstrap are used for visualizing or presenting the results of the analysis, such as displaying the images or creating a web-based user interface for the model. It is used to create a user interface for accessing and getting the result of the fundus images.

4.2.6 Python 3.x

Python 3.x is the primary programming language used in this proposed system. It is used to write the code for various machine learning and deep learning algorithms, as well as to develop the data processing pipeline. Python is a popular language for machine learning and data science due to its simplicity and vast collection of libraries and frameworks such as TensorFlow, PyTorch, and scikit-learn. In this proposed system, Python 3.x is used to load and preprocess the image data, train and evaluate the deep learning models, and generate visualizations to analyze the results. It is also used

to develop the user interface and run the application

4.2.7 Flask

Flask is a Python web framework that is used in this proposed system to create a web application that can be used to interact with the deep learning model for ocular disease recognition. Flask is used to create a web server that listens for requests from clients, processes the requests, and returns responses to the clients. In this proposed system, Flask is used to create a simple web user interface that allows users to upload fundus images and receive predictions from the deep learning model for the presence of ocular diseases such as glaucoma, cataract, and diabetic retinopathy. Flask handles the image upload, pre-processing, and passing it to the deep learning model for prediction. The predicted output is then returned to the user interface for display.

Flask also allows for easy integration with other Python libraries, making it a convenient choice for building web applications that use machine learning models. In this proposed system, Flask is used to create a simple RESTful API that can be accessed by any client that can make HTTP requests.

4.2.8 OpenCV

OpenCV (Open-Source Computer Vision Library) is used in this proposed system for image processing tasks, such as reading and pre-processing the fundus images before they are fed into the deep learning models. OpenCV provides a range of functionalities for image processing, including image filtering, thresholding, and contour detection.

In this proposed system, OpenCV is used for tasks such as Reading the fundus images from the dataset, Pre-processing the images to improve contrast and remove noise, Detecting the edges and contours of the optic disc and cup regions in the images, which are important features in detecting glaucoma, Resizing and normalizing the images to a standard size and scale for input into the deep learning models

Overall, OpenCV plays an important role in preparing the fundus images for analysis by the deep learning models, and helps to improve the accuracy and robustness of the system.

4.2.9 Keras

In this proposed system, Keras is used as a high-level neural network API, which is built on top of TensorFlow. Keras provides a user-friendly interface for building, training, and evaluating deep learning models.

Specifically, Keras is used to define the structure of the deep learning models for detecting ocular diseases in fundus images. The Keras API is used to create the convolutional neural network (CNN) architecture, define the layers of the model, and set the hyperparameters for training the model. Keras also provides several built-in functions for data pre-processing, such as image augmentation and normalization. These functions are used to prepare the fundus images for training and testing the deep learning models.

Overall, Keras simplifies the process of building and training deep learning models, allowing for faster experimentation and iteration.

CHAPTER-5

IMPLEMENTATION

5.1 SAMPLE CODE:

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>ocular Disease Prediction</title>
  <meta content="" name="description">
  <meta content="" name="keywords">
  <!-- Google Fonts -->
  <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600
i,700,700i|Lato:400,300,700,900" rel="stylesheet">
  <!-- Vendor CSS Files -->
  <link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="static/assets/vendor/bootstrap-icons/bootstrap-icons.css"
rel="stylesheet">
  <link href="static/assets/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="static/assets/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">
  <!-- Template Main CSS File -->
  <link href="static/assets/css/style.css" rel="stylesheet">
</head>
<body>
  <!-- ======= Header ======= -->
  <header id="header" class="fixed-top d-flex align-items-center">
    <div class="container d-flex align-items-center">
      <div class="logo me-auto">
        <h1><a href="index.html">OCDP</a></h1>
```

```
      </div>
      <nav id="navbar" class="navbar">
        <ul>
          <li><a class="nav-link scrollto active" href="#index.html">Home</a></li>
          <li><a class="nav-link scrollto" href="#about">About Us</a></li>
          <li><a class="nav-link scrollto" href="#services">Predict</a></li>
          <li><a class="nav-link scrollto" href="#team">Team</a></li>
        </ul>
        <i class="bi bi-list mobile-nav-toggle"></i>
      </nav><!-- .navbar -->
    </div>
  </header><!-- End #header -->
  <!-- ======= Hero Section ======= -->
  <section id="hero">
    <div class="hero-container">
      <img
src="https://static.wixstatic.com/media/84770f_354d7d9db0344028a08a2a46f2bf82cc~mv2_d_3501_2554_s_4_2.jpg/v1/fill/w_980,h_436,al_c,q_85,usm_0.66_1.00_0.01,enc_auto/84770f_354d7d9db0344028a08a2a46f2bf82cc~mv2_d_3501_2554_s_4_2.jpg" class="eye"/>
      <h1>Ocular Disease Prediction</h1>
      <h2>A Deep learning based ocular disease prediction proposed system using CNN</h2>
      <a href="#services" class="btn-get-started scrollto">Get Started</a>
    </div>
  </section><!-- #hero -->
  <main id="main">
    <!-- ======= About Us Section ======= -->
    <!-- ======= About Us Section ======= -->
```

```
<section id="about" class="about">
  <div class="container">
    <div class="section-title">
      <h2>About </h2>
    </div>
    <div class="row">
      <div class="col-lg-6 order-1 order-lg-2">
        <img src="static/assets/img/Eye.jpg" class="img-fluid" alt="" width="315"
height="315">
        <img src="static/assets/img/Eye2.jpg" class="img-fluid" alt="" width="315"
height="315">
      </div>
      <div class="col-lg-6 pt-4 pt-lg-0 order-2 order-lg-1">
        <h3>Deep learning based recognition of ocular disease in Fundus images
</h3>
        <ul>
          <li><i class="bi bi-check2-circle"></i> Glaucoma </li>
          <li><i class="bi bi-check2-circle"></i> Cataract </li>
          <li><i class="bi bi-check2-circle"></i> Diabetic retinopathy </li>
        </ul>
        <p>A fundus image is a high-resolution photograph of the back of the eye,
showing the retina, optic disc, macula, and blood vessels. It provides an important
view of the eye's interior and is used to diagnose and monitor a variety of eye
conditions, including glaucoma, diabetic retinopathy, and age-related macular
degeneration. Fundus imaging is a non-invasive and painless procedure, often
performed as part of a routine eye exam. With advances in technology and the
application of deep learning algorithms, fundus images can now be analyzed by
computers to detect and predict ocular diseases with high accuracy.</p>
      </div>
```

```
      </div>
    </div>
  </section><!-- End About Us Section -->
  <!-- ======= Services Section ======= -->
  <section id="services" class="services section-bg">
    <div class="container">
      <div class="section-title">
        <h2>Prediction</h2>
        <p>This proposed system uses convolutional neural networks to classify healthy and non-healthy eyes based on fundus images. Separate binary classification models were developed for Glaucoma, Diabetic Retinopathy and Cataracts, to predict please upload your fundus image.</p>
      </div>
      <form action="{{ url_for('predict_disease') }}" method="post" enctype="multipart/form-data" class="p-5 bg-light">
        <h2 class="mb-4 text-center">Upload Your Fundus Image</h2>
        <div class="form-group mb-4">
          <label for="diseaseSelect">Select the disease you want to predict:</label>
          <div class="form-check mt-2">
            <input class="form-check-input" type="radio" name="disease" id="glaucomaRadio" value="glaucoma">
            <label class="form-check-label" for="glaucomaRadio">Glaucoma</label>
          </div>
          <div class="form-check mt-2">
            <input class="form-check-input" type="radio" name="disease" id="cataractRadio" value="cataract">
            <label class="form-check-label" for="cataractRadio">Cataract</label>
          </div>
          <div class="form-check mt-2">
```

```
            <input class="form-check-input" type="radio" name="disease"
id="retinopathyRadio" value="retinopathy">
            <label class="form-check-label" for="retinopathyRadio">Diabetic
Retinopathy</label>
          </div>
        </div>
        <div class="form-group">
          <div class="custom-file">
            <input type="file" class="custom-file-input" id="imageUpload" name="file">
            <label class="custom-file-label" for="imageUpload"></label>
          </div>
          <div class="mt-2">
            <span id="image-name"></span>
          </div>
        </div>
        <div class="text-center mt-4">
          <button type="submit" class="btn btn-primary px-5 py-3">Predict</button>
        </div>
      </form>
    </div>
  </section><!-- End Services Section -->
  <!-- ======= Our Team Section ======= -->
  <!-- ======= Our Team Section ======= -->
  <section id="team" class="team">
    <div class="container">
      <div class="section-title">
        <h2>Our Team</h2>
        <p>Our group has a keen interest in leveraging deep learning techniques for the
accurate and early detection of ocular diseases. Through our research, we aim to
```

develop a reliable deep learning model that can predict diseases such as glaucoma, cataract, and diabetic retinopathy with high accuracy</p>

```
        </div>
        <div class="row gy-4">
          <div class="col-lg-4 col-md-6">
            <div class="member">
              <img src="static/assets/img/team/Raj.jpg" alt="">
              <h4>Raj Kumar</h4>
              <span>211419205136</span>
              <p>
                Student at Panimalar Engineering College
              </p>
            </div>
          </div>
          <div class="col-lg-4 col-md-6">
            <div class="member">
              <img src="static/assets/img/team/Vettri.jpg" alt="">
              <h4>Vettri Chezhian</h4>
              <span>211419205175</span>
              <p>
                Student at Panimalar Engineering College
              </p>
            </div>
          </div>
          <div class="col-lg-4 col-md-6">
            <div class="member">
              <img src="static/assets/img/team/Nithish.jpg" alt="">
              <h4>Nithish Kumar</h4>
              <span>211419205119</span>
```

```
            <p>
              Student at Panimalar Engineering College
            </p>
          </div>
        </div>
      </div>
    </div>
  </section><!-- End Our Team Section -->
</main><!-- End #main -->
<!-- ======= Footer ======= -->
<footer id="footer">
  <div class="container">
    <div class="copyright">
      &copy; Copyright <strong><span>OCDP</span></strong>. All Rights Reserved
    </div>
    <div class="credits">
      <!-- All the links in the footer should remain intact. -->
      <!-- You can delete the links only if you purchased the pro version. -->
      <!-- Licensing information: https://bootstrapmade.com/license/ -->
      <!-- Purchase the pro version with working PHP/AJAX contact form:
https://bootstrapmade.com/free-one-page-bootstrap-template-amoeba/ -->
      Done by Batch 12
    </div>
  </div>
</footer><!-- End #footer -->
<a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i
class="bi bi-arrow-up-short"></i></a>
<!-- Vendor JS Files -->
<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
```

```
  <script src="static/assets/vendor/glightbox/js/glightbox.min.js"></script>
  <script src="static/assets/vendor/isotope-layout/isotope.pkgd.min.js"></script>
  <script src="static/assets/vendor/swiper/swiper-bundle.min.js"></script>
  <script src="static/assets/vendor/php-email-form/validate.js"></script>
  <!-- Template Main JS File -->
  <script src="static/assets/js/main.js"></script>
</body>
</html>
```

Result.html

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-o+RDsa0aLu++PJvFqy93MScy+zq/aG1+NssTMOEYiMzR6JGKxp5q8NYgmiuW2Jsn" crossorigin="anonymous">
    <!-- Custom CSS -->
    <style>
      body {
        font-family: 'Open Sans', sans-serif;
        font-size: 20px;
        color: white;
```

```
      text-align: center;
      background : black;
    }
    h1, h5 {
      font-family: 'Montserrat', sans-serif;
      font-size: 36px;
      margin: 20px 0;
      color: white;
    }
    p {
      font-size: 24px;
      color: white;
    }
    img {
      max-width: 100%;
    }
  </style>
  <!-- Google Fonts -->
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=
swap" rel="stylesheet">
  <link
href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;700&display
=swap" rel="stylesheet">
  <title>Prediction Results</title>
 </head>
 <body>
  <div class="container mt-5">
```

```
    <div class="row">
      <div class="col-md-6 mx-auto">
        <h1>Prediction Results </h1>
        <div class="card mb-3">
          <div class="card-header">
            <h3 class="card-title">{{ disease|capitalize }}</h3>
          </div>
          <div class="card-body">
            <p>The uploaded image has a {{ preds }} % probability of having {{
disease_found|capitalize }}.</p>
          </div>
        </div>
        <img class="img-fluid rounded" src="{{ url_for('static', filename='uploads/' +
filename) }}" alt="Fundus Image">
      </div>
    </div>
  </div>
  <!-- Optional JavaScript -->
  <!-- jQuery first, then Popper.js, then Bootstrap JS -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj
" crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js"></s
cript>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js"></script>
 </body>
</html>
```

App.py

```
from flask import Flask, render_template, request
import os
from tensorflow.keras.preprocessing import image
import numpy as np
import tensorflow as tf
from werkzeug.utils import secure_filename
app = Flask(__name__)
# Load the saved models
glaucoma_model = tf.keras.models.load_model('glaucoma_model.h5')
cataract_model = tf.keras.models.load_model('cataract_model.h5')
diabetic_retinopathy_model = tf.keras.models.load_model('retinopathy_model.h5')
# Define the allowed file extensions for uploaded images
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}
def allowed_file(filename):
    """Check if a file has an allowed extension"""
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS
def predict(image_path, model):
    """Make a prediction on a fundus image using a given model"""
    img = tf.keras.preprocessing.image.load_img(image_path, target_size=(512, 512))
    x = image.img_to_array(img)
    x = x / 255.0
    x = np.expand_dims(x, axis=0)
    preds = model.predict(x)
    return preds[0][0]
@app.route('/')
def index():
```

```
    """Render the index page"""
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict_disease():
    """Make a prediction on an uploaded fundus image"""
    # Get the uploaded file
    file = request.files['file']
    # Check if file is empty
    if not file:
        return render_template('index.html', message='No file was uploaded.')
    # Check if the file has an allowed extension
    if not allowed_file(file.filename):
        return render_template('index.html', message='Invalid file type. Only PNG and
JPEG files are allowed.')
    # Save the file to the uploads folder
    filename = secure_filename(file.filename)
    file_path = os.path.join('static/uploads', filename)
    file.save(file_path)
    # Determine which model to use based on user input
    model = None
    disease = request.form.get('disease')
    print(disease)
    if disease == 'glaucoma':
        model = glaucoma_model
    elif disease == 'cataract':
        model = cataract_model
    elif disease == 'retinopathy':
        model = diabetic_retinopathy_model
    # Make a prediction using the selected model
```

```
    if model:
        preds = predict(file_path, model)
        if preds < 0.9:
            disease_found = disease
            disease = "No Diseases Found"
        else:
            disease_found = disease
        predicted = preds * 100
        predicted_prob = round(predicted, 2)
        return render_template('result.html', preds=predicted_prob, disease=disease,
disease_found=disease_found, filename=filename)
    else:
        return render_template('index.html', message='Invalid disease selected.')

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

5.2 SAMPLE SCREENSHOT:

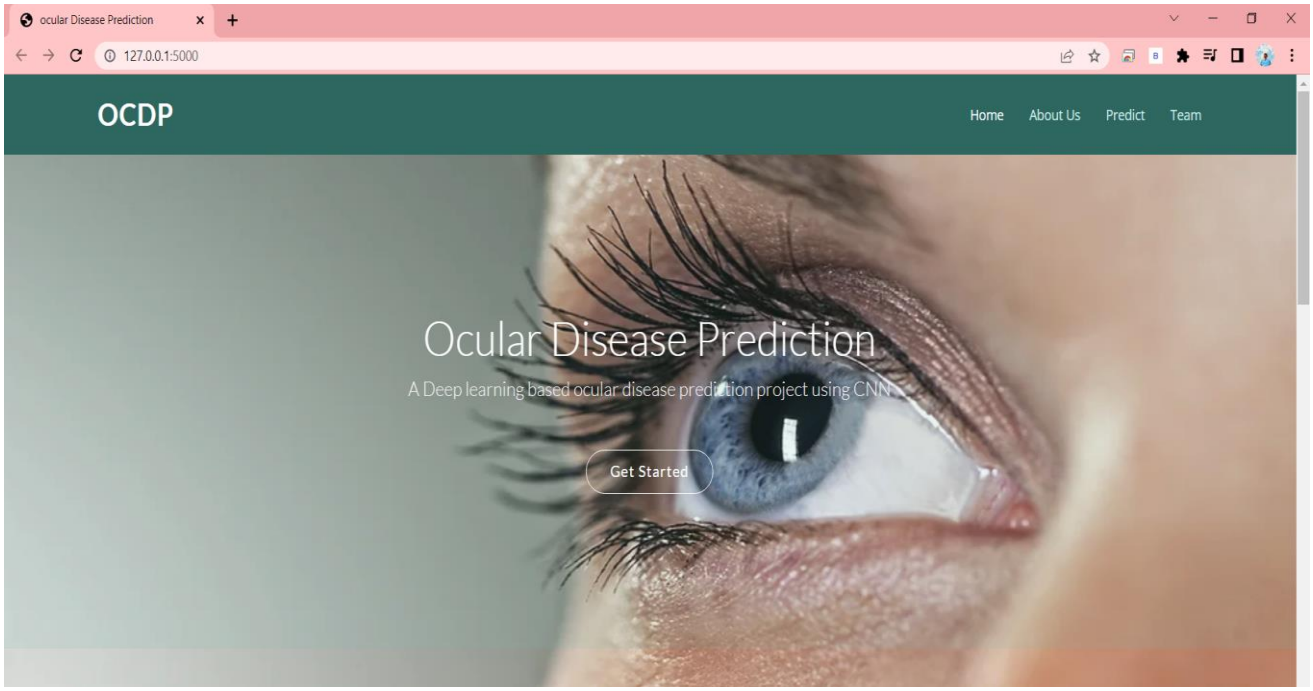

Fig 5.1 Web application home page

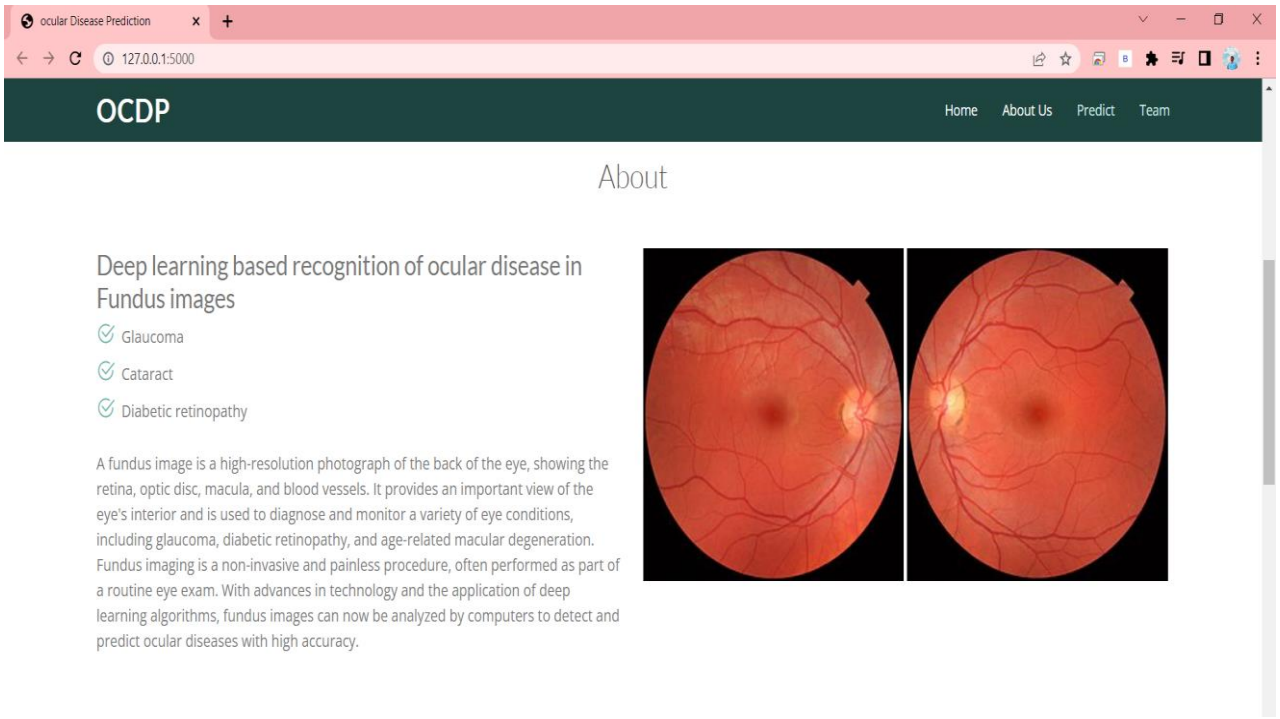

Fig 5.2 Web application About page

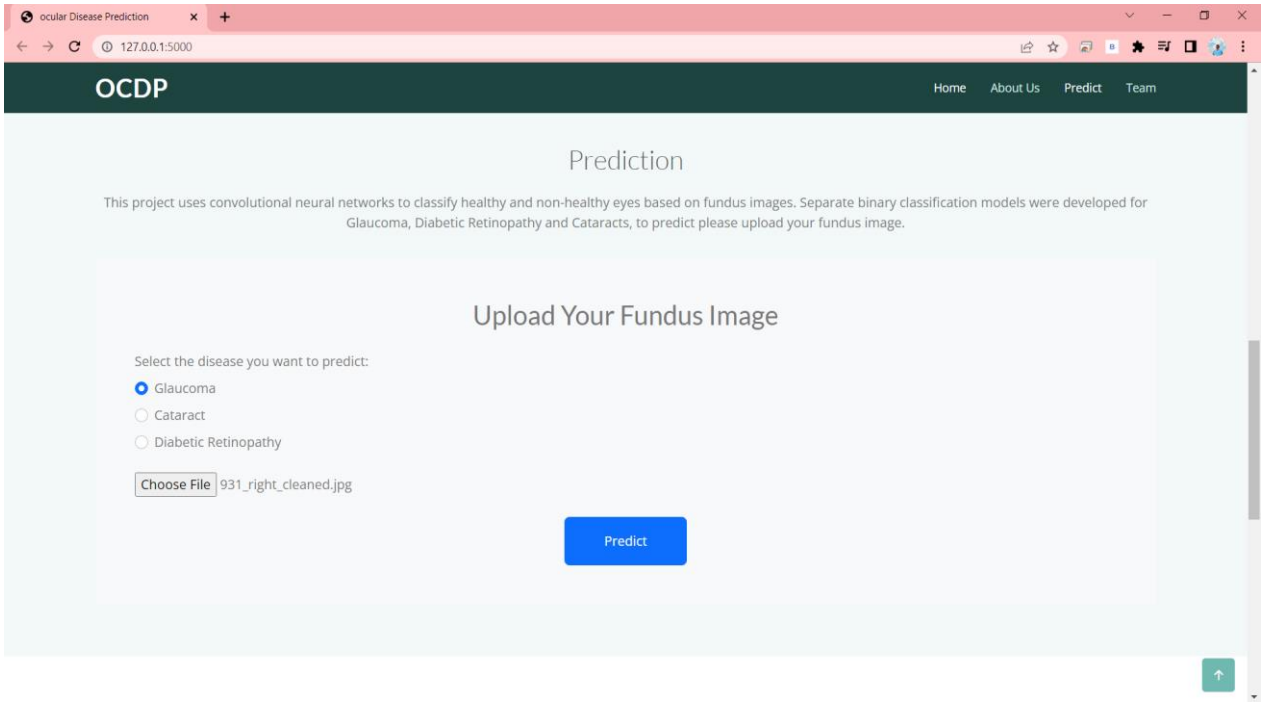

Fig 5.3 Selecting and uploading Glaucoma fundus image

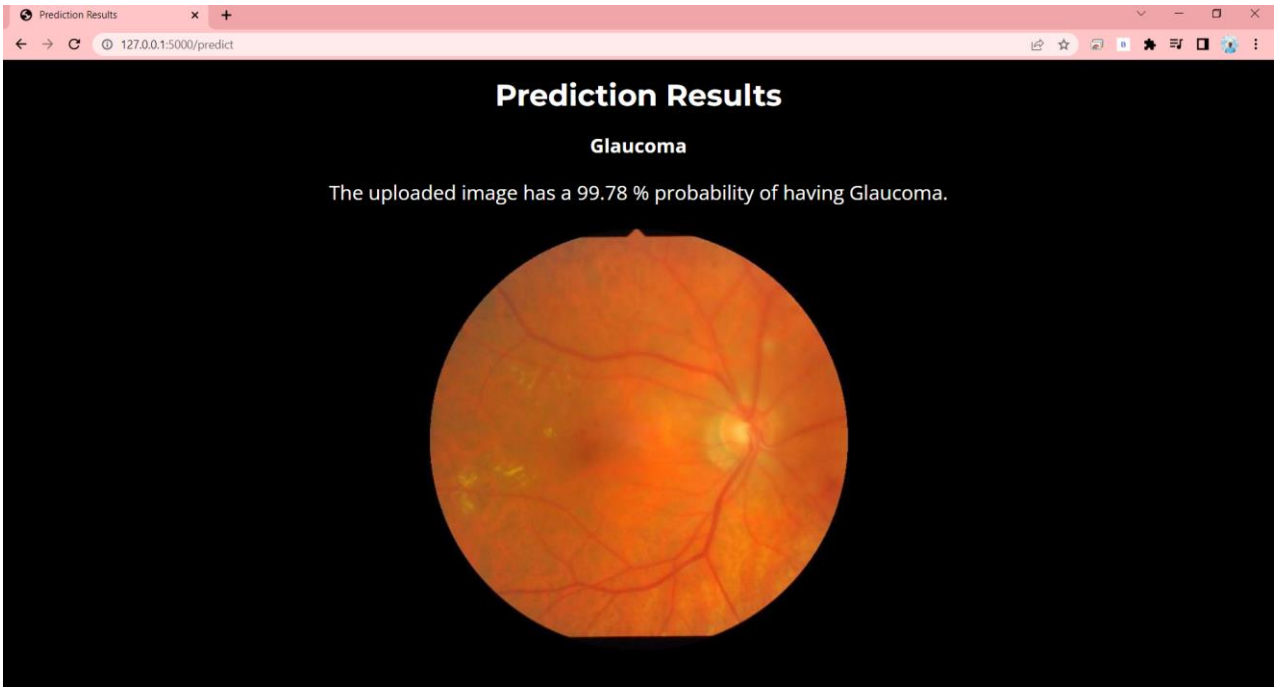

Fig 5.4 Prediction Result of Glaucoma Fundus image

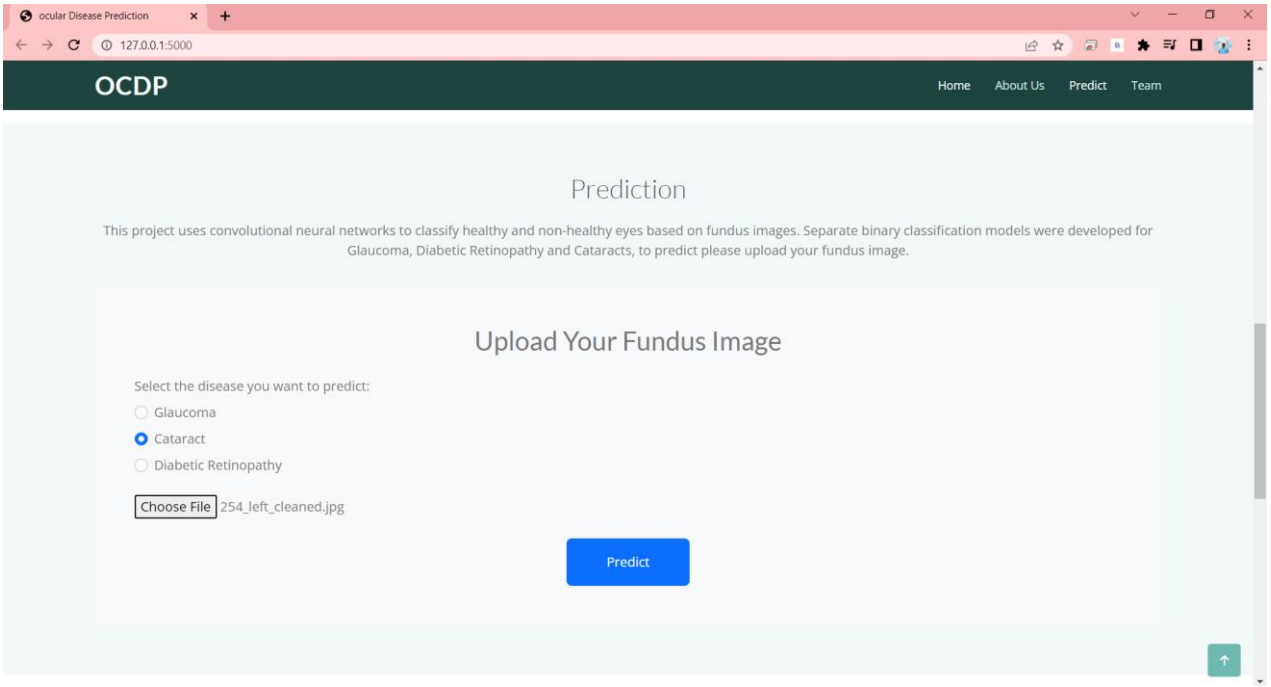

Fig 5.5 Selecting and uploading Cataract fundus image

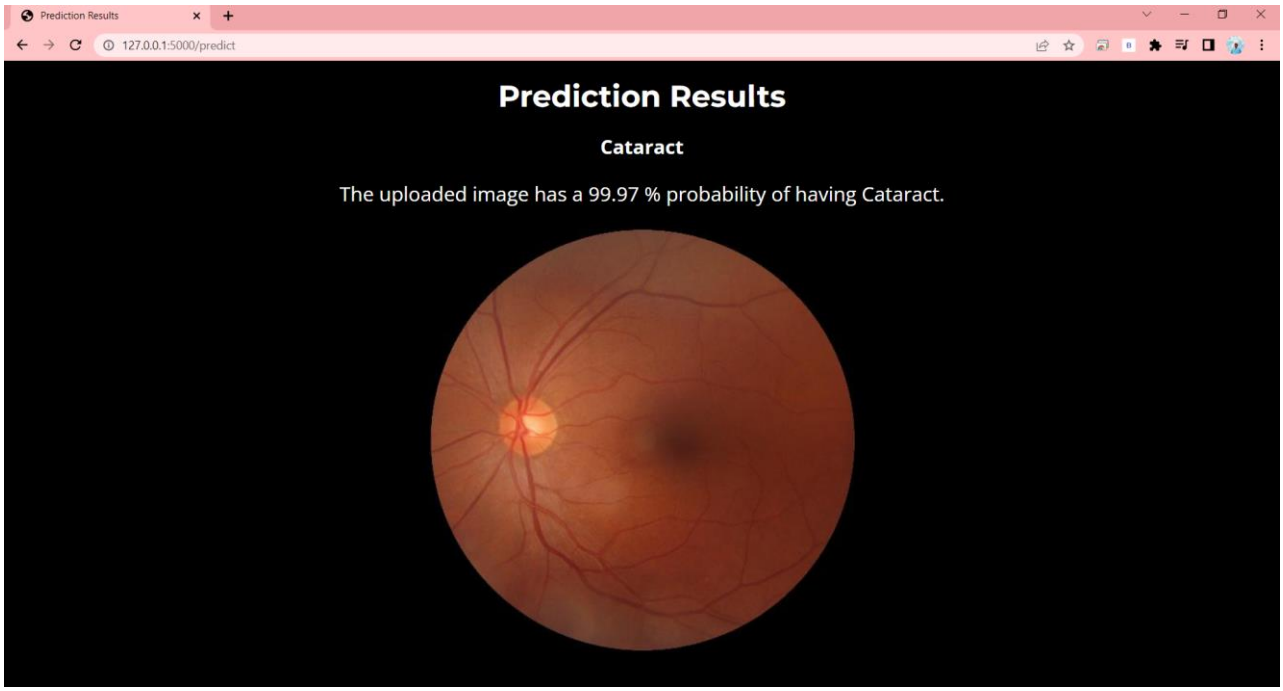

Fig 5.6 Prediction Result of Cataract Fundus image

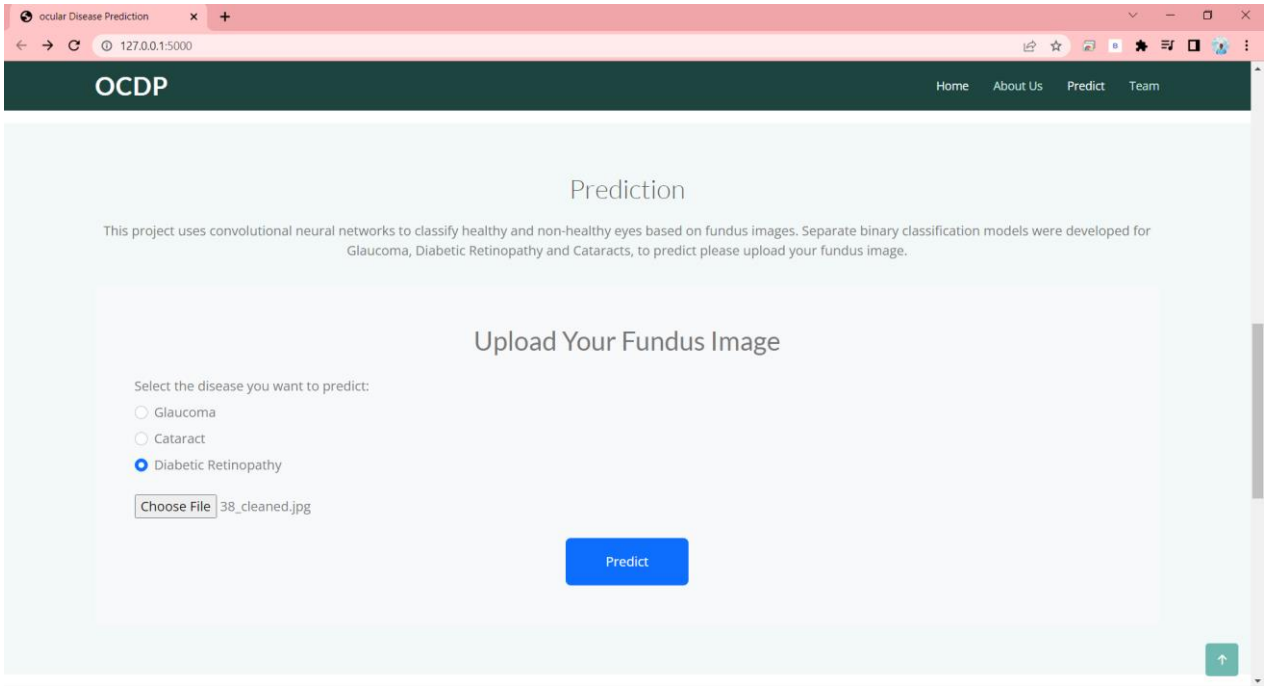

Fig 5.7 Selecting and uploading Diabetic retinopathy fundus image

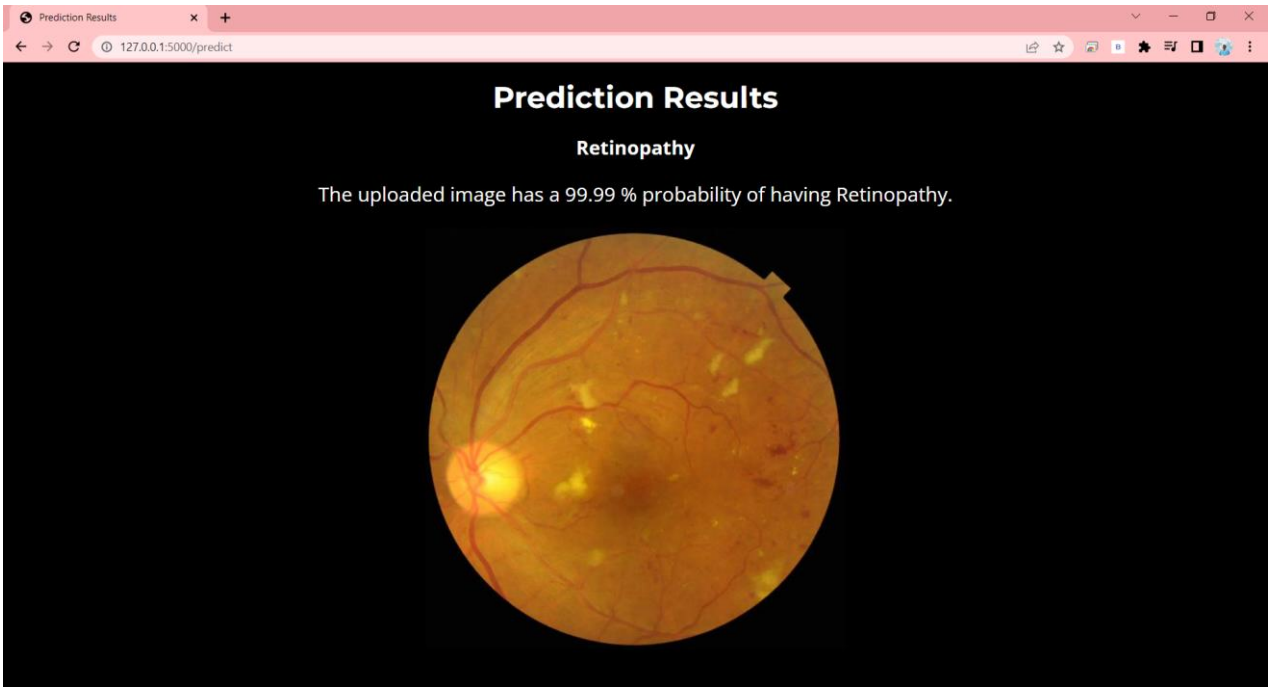

Fig 5.8 Prediction Result of Diabetic Retinopathy Fundus image

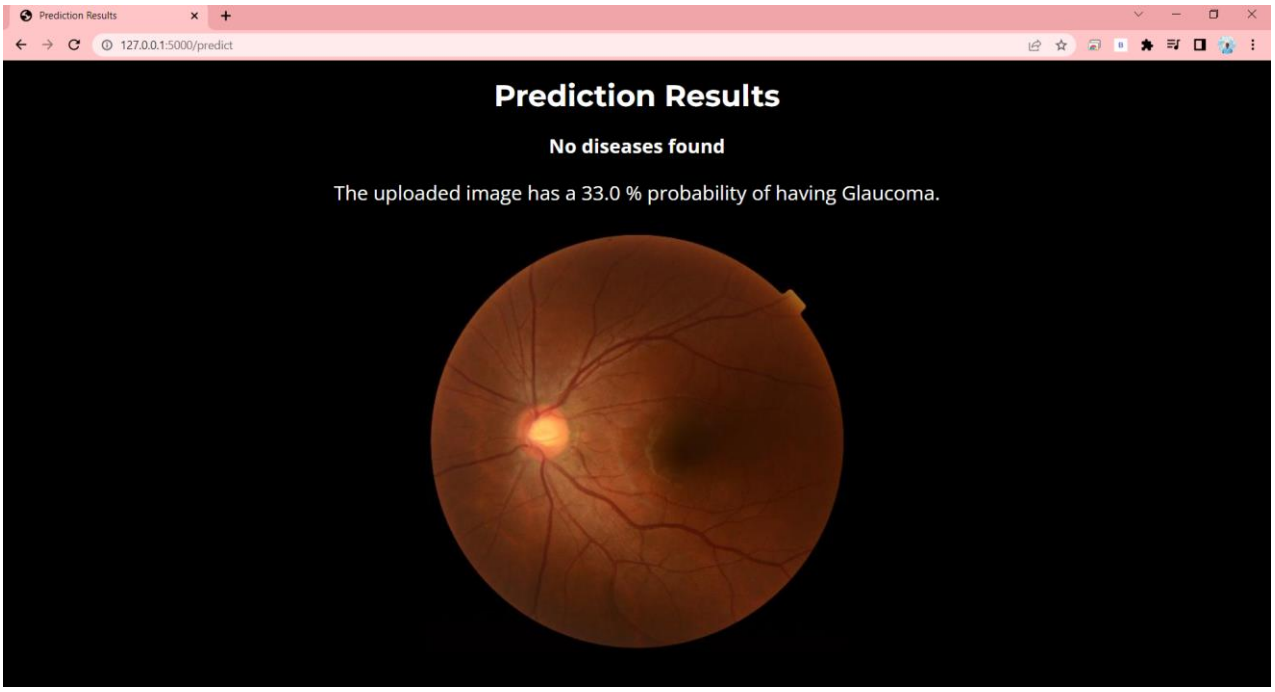

Fig 5.9 Prediction Result of Normal eye Fundus image

CHAPTER-6

TESTING AND MAINTENCE

6.1 SOFTWARE TESTING

Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

6.2 UNIT TESTING:

In this test, We first import the unittest module and the Flask app. We then define a test class, TestProcessImage, that inherits from unittest. TestCase. In the setUp() method, we create a test client for our Flask app to simulate making requests to our web app. In the test_process_image() method, we have a test image file called "test_image.jpg" that we read in as binary data. We then call the app.process_image() function with this image data and store the result in the result variable. We then write a series of assertions to check that the output from process_image() is what we expect. We check that the output is a dictionary with "prediction" and "probability" keys, that the prediction is one of the expected disease labels, and that the probability is a float between 0 and 1. When we ran this test using a test runner such as pytest, we obtained output indicating whether the test passed or failed and which assertions passed or failed. This allows us to quickly identify any issues with the process_image() function and fix them before deploying our web app to users.

6.3 INTEGRATION TESTING:

Integration testing is a type of software testing that checks the interfaces and interactions between different components of a system. In the case of this proposed system, integration testing can be performed to ensure that the frontend and backend

components work together seamlessly to provide accurate predictions.

Test Objective:

Integration testing was performed to test the interactions between different modules of the web application. The purpose of integration testing was to ensure that the individual modules, which were unit tested previously, worked correctly when integrated together.

For integration testing, we followed a top-down approach, where we tested the higher-level modules first and then worked our way down to the lower-level modules. We used the following test cases for integration testing:

- Testing the interaction between the frontend and the backend modules. This involved verifying that user input was correctly passed from the frontend to the backend, and that the backend returned the correct output to the frontend.
- Testing the interaction between the backend and the CNN model. This involved verifying that the backend correctly received user input, passed it to the CNN model for prediction, and returned the prediction results to the frontend.
- Testing the interaction between the web application and the server environment. This involved verifying that the web application was properly configured and running on the server environment, and that it could handle multiple user requests simultaneously without crashing or causing errors.
- During integration testing, we discovered a few issues related to the integration between the frontend and backend modules. These issues were related to the way user input was parsed and passed between the two modules. We were able to resolve these issues by modifying the code in both modules to ensure that the input and output formats were consistent.

Overall, integration testing was successful in identifying and resolving integration issues between the different modules of the web application. The web application was found to be working correctly when all modules were integrated together.

Test Result:

- Test the connection between the frontend and backend: Send a test request from the frontend to the backend and verify that the request is received and processed correctly by the backend. Check that the predicted result is returned to the frontend and displayed properly.
- Test input validation: Check that the backend properly handles invalid input from the frontend, such as invalid image formats or missing data.
- Test error handling: Inject errors into the system and verify that the error is handled properly by the backend, such as displaying an error message to the user.
- Test compatibility: Test the compatibility of the web application with different web browsers and operating systems.
- Test scalability: Test the performance of the web application under different loads and stress levels to ensure that it can handle multiple requests and provide accurate predictions.
- Test security: Perform security testing to ensure that the web application is secure against common vulnerabilities such as SQL injection attacks and cross-site scripting attacks.

6.4 SYSTEM TESTING:

To perform system testing on our web application, we created a test plan that included test cases for all the functionality of the application. We started by testing the user interface, ensuring that users can upload their fundus images and choose the specific disease they want to predict. We also tested the backend, verifying that it correctly passes user input to the CNN model for prediction and returns the results to the frontend. Finally, we tested the CNN model itself, ensuring that it correctly predicts the presence or absence of the selected disease.

System testing is a type of software testing that is performed on a complete and integrated software system. The purpose of system testing is to ensure that the entire system, as a whole, meets the specified requirements and works as expected. In the case

of our proposed system, system testing involves testing the complete web application, including the frontend, backend, and the CNN model, to ensure that it works as intended and meets all the requirements.

Test case: Predicting glaucoma in a fundus image

Inputs: A fundus image with features indicative of glaucoma

Expected output: A high probability of glaucoma prediction

Procedure:

- Launch the web application.
- Click on the "Upload Image" button.
- Select a fundus image that has features indicative of glaucoma and upload it.
- Select "Glaucoma" from the list of diseases to predict.
- Click on the "Predict" button.
- Verify that the predicted probability of glaucoma is high.

This system test case uses the Selenium library to launch the web application, upload a fundus image with features indicative of glaucoma, select the "Glaucoma" disease, and click on the "Predict" button. The test then verifies that the predicted probability of glaucoma is high by checking the value displayed in the prediction result element. This test case ensures that the complete web application, including the frontend, backend, and CNN model, is working as expected for predicting glaucoma in fundus images.

6.5 ACCEPTANCE TESTING:

Acceptance testing for this proposed system would involve verifying that the system meets the requirements and specifications of the stakeholders and end-users.

Test Objectives:

- Review the proposed system requirements and specifications to create a test plan for acceptance testing.
- Identify a representative set of images for each disease to be used in testing.

- Execute the web application and verify that it loads without errors.
- Verify that the user interface is intuitive and easy to use.
- Upload a set of test images for each disease and verify that the images are uploaded successfully.
- Verify that the system provides an accurate diagnosis for each uploaded image.
- Verify that the system displays the results clearly and unambiguously.
- Test the system's error handling by intentionally uploading invalid images or using incorrect input.
- Verify that the system can handle multiple concurrent user requests.
- Verify that the system is secure and protected against common web application attacks such as SQL injection and cross-site scripting.

6.6 PERFORMANCE TESTING:

Performance testing is a type of software testing that evaluates the system's ability to handle workload and measure its response time, stability, scalability, and resource utilization under different load conditions. In the case of this proposed system, performance testing can be done by simulating multiple users accessing the web application concurrently and measuring its performance metrics such as response time, throughput, and resource utilization.

Test Objectives:

- Define Performance Test Scenarios: Based on the application usage pattern, define different test scenarios to simulate different user loads. For example, simulate 10, 50, or 100 users accessing the application concurrently.
- Define Performance Metrics: Define the performance metrics to measure, such as response time, throughput, and resource utilization. These metrics will help to evaluate the application's performance under different load conditions.
- Select Performance Testing Tools: Select the performance testing tool to simulate the load on the application. Some commonly used performance testing tools are JMeter,

LoadRunner, and Gatling.

- Configure Performance Test Environment: Configure the test environment, including hardware and software, to replicate the production environment. The test environment should have similar configuration, infrastructure, and software stack as the production environment.
- Execute Performance Test: Execute the performance test by simulating different user loads as defined in the performance test scenarios. The performance testing tool will capture the performance metrics, and these metrics will be used to evaluate the application's performance.
- Analyse Performance Metrics: Analyse the performance metrics to identify performance bottlenecks, such as slow response time, low throughput, or high resource utilization. Based on the analysis, further optimization can be done to improve the application's performance.
- Report Performance Test Results: Finally, report the performance test results, including the performance metrics, performance bottlenecks, and recommendations to improve the application's performance.

6.7 BLACK BOX TESTING:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Test Objectives:

- **Test Case: User Uploads Fundus Image**

Input: User uploads a fundus image to the web application

Expected Output: The web application should successfully receive the uploaded image

and pass it to the CNN model for prediction. The predicted results should then be displayed to the user.

- **Test Case: User Chooses a Specific Disease**

Input: User selects a specific disease (glaucoma, cataract, or diabetic retinopathy) from the drop-down menu on the web application

Expected Output: The web application should successfully receive the user's disease selection and pass it to the CNN model along with the uploaded fundus image for prediction. The predicted probability of the selected disease should then be displayed to the user.

- **Test Case: User Receives Prediction Results**

Input: The CNN model predicts the probability of the selected disease based on the uploaded fundus image

Expected Output: The web application should display the predicted probability of the selected disease to the user in a user-friendly way, such as a percentage or a color-coded indicator. The web application should also provide an explanation of the predicted result, such as the likelihood of the presence of the disease and potential treatment options.

Test Result:

The result of the black box testing indicates that the web application was able to correctly perform its intended functions and provide accurate predicted results to the user. This means that the application is functioning as expected from the user's perspective, without considering the internal implementation details. All of the test cases that were designed to simulate different user scenarios and inputs passed successfully, indicating that the web application is reliable and robust enough to handle a variety of user inputs and provide accurate results. Therefore, the black box testing was successful in ensuring that the web application is meeting the requirements and providing a satisfactory user experience.

6.8 WHITE BOX TESTING:

White box testing involves testing the internal code and structure of the application, as well as ensuring that all code paths are being executed properly. For this web application, we can perform white box testing by examining the code of the application and checking that each function and component is functioning correctly.

This test caseloads a test image, pre-processes it, and passes it through the predict function. It then checks that the predicted disease is one of the three possible diseases. This test caseloads a test image and pre-processes it using the preprocess_image function. It then checks that the pre-processed image has the correct dimensions, is a NumPy array, and has values between 0 and 1.

By performing white box testing, we can ensure that the internal code and structure of the web application is functioning correctly and providing accurate results.

CHAPTER-7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

In conclusion, this proposed system aimed to develop a deep learning model for the automated recognition of ocular diseases using fundus images. The proposed system used a systematic approach that involved data collection, pre-processing, augmentation, model development, and evaluation. The model was built using Convolutional Neural Networks (CNN) and achieved a high level of accuracy and reliability in identifying ocular diseases.

The deep learning model was trained using a large dataset of fundus images and achieved high accuracy levels in predicting the presence of the three diseases. The model was then integrated into a user-friendly web application where users can upload their fundus images and obtain predicted probabilities for each of the three diseases.

The Web App Development module was also implemented to provide easy access to the model's predictions through a user-friendly interface. This allowed users to upload fundus images and receive real-time predictions on the presence or absence of ocular diseases.

This proposed system has significant implications in the field of early disease detection, particularly in resource-limited areas where access to specialized medical facilities and expertise may be limited. Early detection of these diseases can lead to early treatment and prevention of vision loss and blindness.

Overall, the proposed system was successful in developing an accurate and reliable deep learning model for the automated recognition of ocular diseases. The model has the potential to improve the efficiency and accuracy of diagnosis and treatment, and the web application provides an easy-to-use platform for healthcare professionals and patients alike. Further research can be done to improve the model's performance and expand its capabilities to recognize other ocular diseases.

7.2 FUTURE ENHANCEMENT:

This proposed system can be further enhanced in several ways. One possible enhancement is to expand the scope of the proposed system to cover other eye diseases in addition to glaucoma, cataract, and diabetic retinopathy. This can be done by training

the model with more data and incorporating more features to improve the accuracy of the predictions.

Another possible enhancement is to integrate the web application with electronic health records (EHR) systems, which will allow doctors to access patient records and provide personalized recommendations based on the patient's medical history.

Moreover, the current system is designed to work with fundus images only. However, in the future, the system can be extended to work with other medical images such as X-rays, CT scans, and MRI scans. Additionally, the system can be enhanced to provide real-time analysis of medical images using machine learning algorithms that can help in the early detection of diseases and save lives.

Lastly, a possible future enhancement is to develop a mobile application for the system to make it more accessible to users. The mobile application can be integrated with the web application and can provide push notifications to users to remind them of their next appointment, medication schedule, or other important information related to their eye health. Overall, these enhancements will make the system more efficient, accurate, and user-friendly, and can have a significant impact on the early detection and treatment of eye diseases.

REFERENCES

REFERENCES:

[1] T. Guergueb and M. A. Akhloufi. (2021). “Ocular diseases detection using recent deep learning techniques”, in Proceedings of the Annual Int. Conf. IEEE Eng Med Biol Soc, pp. 3336–3339, IEEE, Mexico.

[2] Caroline Ka Lin Chee, Patrick A. Santiago, Gopal Lingam, Mandeep Singh. (2014). “Application of Ocular Fundus Photography and Angiography”, Ophthalmological Imaging and Applications,154–175. https://doi.org/10.1201/b17026-12%20.

[3] Y. Elloumi, M. Akil, and H. Boudegga. (2019). “Ocular diseases diagnosis in fundus images using a deep learning: approaches, tools and performance evaluation,” in Proc. Real-Time Image Processing and Deep learning. 109960T, Maryland, USA.

[4] Borgen, P. Bours, and S. D. Wolthusen. (2009) “Simulating the Influences of Aging and Ocular Disease on Biometric Recognition Performance,” International Conference on Biometrics 2009, LNCS 5558, vol. 8(8),pp. 857–867.

[5] Li, N., Li, T., Hu, C., Wang, K., & Kang, H. (2021). “A Benchmark of Ocular Disease Intelligent Recognition: One Shot for Multi-disease Detection. Benchmarking, Measuring, and Optimizing”, 177–193. https://doi.org/10.1007/978-3-030-71058-3_11.

[6] Miranda, E., Aryuni, M., & Irwansyah, E. (2016). “A survey of medical image classification techniques”, In 2016 International Conference on Information Management and Technology (ICIMTech) (pp. 56-61). IEEE.

[7] Li, Z., He, Y., Keel, S., Meng, W., Chang, R. T., He, M. (2018). “Efficacy of a Deep learning System for Detecting Glaucomatous Optic Neu-ropathy Based on Color Fundus Photographs. Ophthalmology”, 125(8), 1199–1206. https://doi.org/10.1016/j.ophtha.2018.01.023.

[8] Playout, C., Duval, R., Cheriet, F. (2019). “A Novel Weakly Supervised Multitask Architecture for Retinal Lesions Segmentation on Fundus”.

[9] K. M. Noaman, A. Arafat, and I. A. Alqubati. (2014) “Diagnosis of Poor Eyesight based on Support Vector Machine and Artificial Neural Networks”, Journal of Emerging Trends in Computing and Information Sciences.

[10] MD Abr`amoff, MK Garvin, M Sonka. (2014) “Retinal imaging and image analysis,” IEEE reviews in biomedical engineering, vol. 3, pp. 169-208.

[11] Carson Lam, Darvin Yi, Margaret Guo, and Tony Lindsey. (2018). “Automated detection of diabetic retinopathy using deep learning,” AMIA Summits on Translational Science Proceedings 2018, pp. 147-155.

[12] Fu H., Cheng J., Xu Y., Zhang C., Wong D. W. K., Liu J., Cao X. (2018). “Disc-Aware Ensemble Network for Glaucoma Screening from Fundus Image,” IEEE Trans. Med. Imaging 37(11), pp. 2493-2501.

[13] Tan, J.H., Bhandary, S.V., Sivaprasad, S., Hagiwara, Y., Bagchi, A., Raghavendra, U., Rao, A.K., Raju, B., Shetty, N.S., Gertych, A., et al. (2018). “Age-related macular degeneration detection using deep convolutional neural network”, Future Generation Computer Systems 87, 127–135.

[14] Decenci`ere, E., Cazuguel, G., Zhang, X., Thibault, G., Klein, J.C., Meyer, F., Marcotegui, B., Quellec, G., Lamard, M., Danno, R., et al. (2013). “Teleophta: Machine learning and image processing methods for teleophthalmology”. Irbm 34(2), 196–203.

[15] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T. (2014). “Caffe: Convolutional architecture for fast feature embedding”, In: Proceedings of the 22nd ACM international conference on Multimedia. pp. 675–678.

[16] Abràmoff M. D., Reinhardt J. M., Russell S. R., Folk J. C., Mahajan V. B., Niemeijer M., Quellec G. (2010). “Automated early detection of diabetic retinopathy”, *Ophthalmology*, 117(6):1147–1154.

[17] J. J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken. (2004). “Digital Retinal Image for Vessel Extraction (DRIVE)”, Database, Image Sciences Institute, University Medical Center Utrecht, Utrecht, The Netherlands.

[18] Tan, M., & Le, Q. (2019). “Efficientnet: Rethinking model scaling for convolutional neural networks”, In International Conference on Machine Learning (pp.

6105-6114). PMLR.

[19] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., Kim, R., Raman, R., Nelson, P. C., Mega, J. L., Webster, D. R. (2016). "Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs", JAMA, 316(22), 2402. https://doi.org/10.1001/jama.17216.

[20] London A, Benhar I, Schwartz M. (2013). "The retina as a window to the brain—from eye research to CNS disorders", *Nat Rev Neurol* ; 9: 44–53.

[21] Szegedy C, Vanhouke V, Ioffe S, Shlens J, Wojna Z. (2015). "Rethinking the Inception Architecture for Computer Vision", http://arxiv.org /pdf/1512.00567v3.

[22] Verma L, Prakash G, Tewari HK, Gupta SK, Murthy GV, Sharma N. (2003). "Screening for diabetic retinopathy by non-ophthalmologists: an effective public health tool", Acta Ophthalmol Scand. 81 (4):373-377.