

**Department of Electrical and Computer Engineering**

**North South University**

**Senior Design Project**

**Glaucoma Detection based on Deep Learning**

**Md. Samin Ahmed ID# 1911998642**

**Sadia Sultana ID# 1921218042**

**Obantika Roy Anti ID# 1921383042**

**Fabiha Tazri Okita ID# 1922086042**

**Faculty Advisor:**

**Dr. K. M. A. Salam**

**Professor Department of Electrical and Computer Engineering**

**North South University**

**Summer, 2023**

# LETTER OF TRANSMITTAL

December, 2023

To

Dr. Rajesh Palit

Chairman,

Department of Electrical and Computer Engineering

North South University, Dhaka

Subject: **Submission of Capstone Project Report on “Glaucoma Detection based on Deep Learning”**

Dear Sir,

With due respect, we would like to submit our **Capstone Project Report** on **“Glaucoma Detection based on Deep Learning”** as a part of our BSc program. The report deals with Early stage of the Glaucoma Detection. This project was very much efficient for us as it detects glaucoma on its initial stage. It will be helping us in real life as well as in medical field. We made every effort to be as competent as possible in order to meet all of the requirements for this report.

It will be delightful for us if you kindly observe our report and give your valuable opinion. We made every effort to be as competent as possible in order to meet all the requirements for this report. It would be our pleasure if you found this report to be insightful and helpful in gaining a clear understanding of the topic.

Sincerely Yours,

.........................................................

Md. Samin Ahmed

ECE Department

North South University, Bangladesh

........................................................

Sadia Sultana

ECE Department

North South University, Bangladesh

.........................................................

Obantika Roy Anti

ECE Department

North South University, Bangladesh

.........................................................

Fabiha Tazri Okita

ECE Department

North South University, Bangladesh

# APPROVAL

Md. Samin Ahmed (ID # 1911998642), Sadia Sultana (ID # 1921218042), Obantika Roy Anti (ID # 1921383042), Fabiha Tazri Okita (ID # 1922086042), from Electrical and Computer Engineering Department of North South University, are working on the Senior Design Project titled “**Glaucoma Detection based on Deep Learning**” under the supervision of **Dr. K. M. A. Salam** partial fulfillment of the requirement for the degree of Bachelors of Science in Computer Science and Engineering and has been accepted as satisfactory.

**Supervisor’s Signature**

…………………………………….

**Dr. K. M. A. Salam**

**Professor**

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

**Chairman’s Signature**

…………………………………….

**Dr. Rajesh Palit**

**Professor**

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

# DECLARATION

This is to certify that this work is entirely original to us. No portion of this work has ever been partially or totally submitted elsewhere for the award of another degree or diploma. Without the official agreement of the project supervisor, all information pertaining to the project must remain confidential. All prior works are identified and acknowledged in this report are relevant. The supervisor's declared anti-plagiarism guideline has been upheld.

Students’ names & Signatures

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**1. Md. Samin Ahmed**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**2. Sadia Sultana**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**3. Obantika Roy Anti**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**4. Fabiha Tazri Okita**

# ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to Dr. K. M. A. Salam, Professor, Department of Electrical and Computer Engineering, North South University, Bangladesh, for his invaluable support, precise guidance, and advice regarding the experiments, research, and theoretical studies carried out during whole courses CSE499A and CSE499B of the current capstone project as well as in the preparation of the current report.

Furthermore, we would like to thank the Department of Electrical and Computer Engineering, North South University, Bangladesh for facilitating the research. We would also like to thank our friends, mates, and seniors without whom this project would have been completed. We are grateful to our parents for their unconditional support and countless sacrifices.

# ABSTRACT

The study on glaucoma detection using deep learning algorithms is presented in this abstract. Early detection and treatment of glaucoma, a main cause of permanent blindness, present difficult problems. The existing approaches for glaucoma detection rely on time-consuming, subjective manual examinations by ophthalmologists. We suggest a deep learning-based method for automated glaucoma detection to solve this problem.

Our work makes use of a sizable dataset of retinal pictures from people with and without glaucoma. We improve elements important for the diagnosis of glaucoma during the preprocessing of the photos. On the basis of this dataset, deep learning models, such as convolutional neural networks, are trained to identify patterns and distinguishing characteristics related to glaucomatous alterations in the retinal and optic nerve structures.

The outcomes of our tests show that the suggested deep learning strategy for glaucoma detection is effective. Our models successfully distinguish between glaucomatous and non-glaucomatous pictures with a high degree of accuracy, sensitivity, and specificity. Our method constantly beats them when we compare the performance of our models to that of current state-of-the-art methodologies.

The discipline of ophthalmology and public health will be significantly impacted by this research. Our method can help ophthalmologists make a quick and accurate diagnosis of glaucoma, resulting in earlier interventions and improved management of the condition. This may help glaucoma sufferers live better lives and possibly prevent visual loss. Additionally, the suggested deep learning-based method can be incorporated into current healthcare systems, offering a practical and affordable glaucoma screening and diagnosis option glaucoma screening and diagnosis.

In summary, this study offers a unique deep learning-based method for glaucoma identification. The obtained results demonstrate the potential of automated methods in enhancing the precision and effectiveness of glaucoma diagnosis. The suggested strategy has important implications for clinical practice and public health, opening the door to improved glaucoma therapy and the avoidance of permanent vision loss.

TABLE OF CONTENTS

[LETTER OF TRANSMITTAL 2](#_Toc155182279)

[APPROVAL 4](#_Toc155182280)

[DECLARATION 5](#_Toc155182281)

[ACKNOWLEDGEMENTS 6](#_Toc155182282)

[ABSTRACT 7](#_Toc155182283)

[LIST OF FIGURES 10](#_Toc155182284)

[LIST OF TABLES 12](#_Toc155182285)

[Chapter 1 Introduction 13](#_Toc155182286)

[1.1 Background and Motivation 13](#_Toc155182287)

[1.2 Purpose and Goal of the Project 15](#_Toc155182288)

[1.3 Organization of the Report 15](#_Toc155182289)

[Chapter 2 Research Literature Review 16](#_Toc155182290)

[2.1 Existing Research and Limitations 16](#_Toc155182291)

[Chapter 3 Methodology 22](#_Toc155182292)

[3.1 System Design 22](#_Toc155182293)

[3.2 Software Components 25](#_Toc155182294)

[3.3 Software Implementation 38](#_Toc155182295)

[Chapter 4 Investigation/Experiment, Result, Analysis and Discussion 42](#_Toc155182296)

[Chapter 5 Impacts of the Project 61](#_Toc155182297)

[5.1 Impact of this project on societal, health, safety, legal and cultural issues 61](#_Toc155182298)

[5.2 Impact of this project on environment and sustainability 62](#_Toc155182299)

[Chapter 6 Project Planning and Budget 65](#_Toc155182300)

[Chapter 7 Complex Engineering Problems and Activities 66](#_Toc155182301)

[7.1 Complex Engineering Problems (CEP) 66](#_Toc155182302)

[7.2 Complex Engineering Activities (CEA) 67](#_Toc155182303)

[Chapter 8 Conclusions 68](#_Toc155182304)

[8.1 Summary 68](#_Toc155182305)

[8.2 Limitations 68](#_Toc155182306)

[8.3 Future Improvement 69](#_Toc155182307)

[References 70](#_Toc155182308)

[Appendices 71](#_Toc155182309)

# LIST OF FIGURES

[Figure 1: Fundus Image 13](file:///C:\Users\HP\Downloads\Cse499_Blackbook.docx#_Toc155182310)

[Figure 2: Fundus Camera 14](#_Toc155182311)

[Figure 3: The flowchart of our project’s design 22](#_Toc155182312)

[Figure 4: The flowchart of our model appliance 22](#_Toc155182313)

[Figure 5: Use Case Diagram 23](#_Toc155182314)

[Figure 6: System Diagram 24](#_Toc155182315)

[Figure 7: DRISHTI image 26](#_Toc155182316)

[Figure 8: RIM ONE image 27](#_Toc155182317)

[Figure 9: ACRIMA image 28](#_Toc155182318)

[Figure 10: Graphical representation of glaucoma and normal image 29](#_Toc155182319)

[Figure 11: Image Count for train, validation and test dataset 32](#_Toc155182320)

[Figure 12: Login Page 36](#_Toc155182321)

[Figure 13: Detector Page 37](#_Toc155182322)

[Figure 14: Validation Graph of InceptionV3 45](#_Toc155182323)

[Figure 15: Confusion Matrix of InceptionV3 46](#_Toc155182324)

[Figure 16: Test score of InceptionV3 46](#_Toc155182325)

[Figure 17: Validation Graph of CNN 47](#_Toc155182326)

[Figure 18: Confusion Matrix of CNN 48](#_Toc155182327)

[Figure 19: Test score of CNN 48](#_Toc155182328)

[Figure 20: Validation Graph of MobilenetV2 49](#_Toc155182329)

[Figure 21: Confusion Matrix of MobilenetV2 50](#_Toc155182330)

[Figure 22: Test score of MobilenetV2 50](#_Toc155182331)

[Figure 23: Validation Graph of Restnet50 51](#_Toc155182332)

[Figure 24: Confusion Matrix of Restnet50 52](#_Toc155182333)

[Figure 25: Test score of Restnet50 52](#_Toc155182334)

[Figure 26: Validation Graph of VGG16 53](#_Toc155182335)

[Figure 27: Confusion Matrix of VGG16 54](#_Toc155182336)

[Figure 28: Test score of VGG16 54](#_Toc155182337)

[Figure 29: Validation graph of Custom CNN Model 55](#_Toc155182338)

[Figure 30: Confusion Matrix of Custom CNN Model 56](#_Toc155182339)

[Figure 31: Confusion Matrix of Custom CNN Model 56](#_Toc155182340)

[Figure 32: Comparative analysis of models 58](#_Toc155182341)

[Figure 33: Gantt chart 65](#_Toc155182342)

# LIST OF TABLES

[Table I. List of Software Tools 38](#_Toc153664946)

[Table II. A Complex Engineering Problem Attributes 66](#_Toc153664947)

[Table III. A Complex Engineering Problem Activities 67](#_Toc153664948)

# Chapter 1 Introduction

## Background and Motivation

Glaucoma is a common eye illness that is irreversible and the second-leading cause of visual weakness. It damages the optic nerve. The symptoms can start so slowly that they are hard to notice. Glaucoma can be detected at an early stage by inspecting the condition of the retinal fundus image. There are generally four categories or types of glaucoma. The four types are primary open-angle glaucoma, angle closure glaucoma, normal tension glaucoma, and secondary glaucoma. Among them, open-angle glaucoma is the most common. The number of people affected by glaucoma was 3 million in 2010, and it is expected to rise to 76.0 million by 2020. Glaucoma is a disease that primarily affects people between the ages of 40 and 60. Its global rate is 5%, and it is rising steadily. It is also known as the "silent thief of sight". There’s no cure for glaucoma, but early treatment can often stop the damage and protect against vision loss.

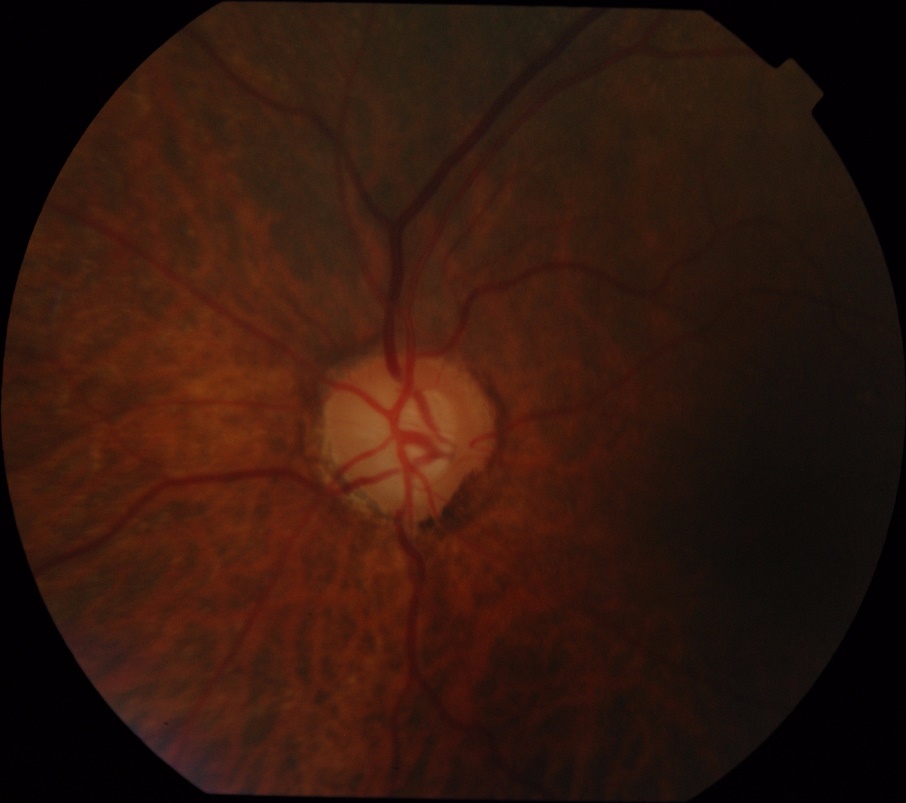


Figure 1: Fundus Image

The proposed autonomous system's goal is to create a complete decision-support system for early glaucoma detection. The main motive behind doing the project is knowing that our work can help save sight and improve the quality of life for individuals affected by glaucoma. As early detection of glaucoma is crucial for effective treatment, by developing an advanced deep learning model, we can identify subtle signs of the disease at its earliest stages. Developing an accurate and efficient deep learning model for its detection can significantly improve early diagnosis and treatment outcomes. Fundus cameras is a a specialized medical device designed for capturing high-resolution images of the fundus to monitor glaucoma.



Figure 2: Fundus Camera

The motivation lies in the potential to save people's vision and enhance their quality of life.

By creating a reliable glaucoma detection system, we can contribute to the overall improvement of public health.

## Purpose and Goal of the Project

1) The proposed autonomous system's goal is to create a complete decision

support system for early glaucoma detection.

2) Extensive experiments to assess the performance of the proposed glaucoma

detection system on real-world datasets, as well as comparisons with existing state-of-the-art techniques for performance evaluation.

3) Disease can be treated in its early stages.

4) Disease can be treated more precisely and efficiently.

5) These systems take a fundus image as input and perform a series of computations to provide feedback to ophthalmologists, such as labeling regions of interest such as the optic disc or providing a probability indicating the patient's risk of developing glaucoma. These algorithms improve disease detection accuracy.

## Organization of the Report

Here is the arrangement of different sections of the report-

1. Chapter 1 presents the introduction of this project.
2. Chapter 2 presents the literature reviews related to this project.
3. Chapter 3 presents the methodology of this project.
4. Chapter 4 presents the experiment, result, analysis and discussion of this project.
5. Chapter 5 presents the experiment, result, analysis and discussion of this project.
6. Chapter 6 presents the planning and budget of this project.
7. Chapter 7 presents the complex engineering problems and activities.
8. Chapter 8 presents the conclusion.

# Chapter 2 Research Literature Review

## 2.1 Existing Research and Limitations

We have researched some papers and found their results and limitations.

The research paper [1]

**Title –** “**Glaucoma Detection based on Deep Learning Network in Fundus Image**.”[05 November 2019]

**Methodology-**

1. In this paper two methods have been used named M-Net and DENET. M-Net is multi-level segmentation network which solves both optic disk and optic cup segmentation. Based on optic disc cup segmentation, the vertical cup to disc ratio is calculated.

**2.** The architecture of M-Net consists of three main parts.

1. Multi-scale U-shape network

2. Side-output layer

3. Multi-label loss function

**3.**The DENET which contains four streams.

1. Global Image Stream

2. The segmentation guided network.

3. Disc region stream

4. Disc polar stream

**Result:**

The M-Net obtains 89% accuracy over SCES dataset and 79% accuracy over new collected dataset.

The DENET obtains the best performances on the SCES dataset of 91% accuracy. and the new collected dataset of 82% accuracy.D-Net, achieves the higher performances on both datasets, which demonstrates the capability of deep learning technique.

**Remarks:**

To use DENET architecture, machine learning had to get developed. ML is a framework to automate (through algorithms) statistical models, like a linear regression model, to get better at making predictions. As we are looking forward to deep neural networks ,DENET is not preferable for this project.

On the other hand, M-Net is a segmentation-based method, which solves the OD and OC segmentation jointly into a one-stage multi label framework

The research paper [2]-

**Title-** “**Automatic detection of glaucoma via fundus imaging and artificial intelligence: A review**”

**Methodology-**

A variety of classification models and techniques are proposed in several works, along with the classifier are-

1. Support Vector Machine Classifiers (SVM),
2. Clustering classifiers,
3. Linear mixed-effects statistical modelling (LME)
4. Random Forest classifier (RF),
5. Damped Least-Squares Recurrent Deep Neural Learning Classification (DLRNL),
6. K-nearest neighbours (K-NN),
7. Neural network (NN)

A ranges of publicly available and private databases were used-

1. **DRISHTI dataset:** The DRISHTI database (was one of the most popular databases being used seven times.
2. High-Resolution Fundus (HRF) dataset.
3. GlaucomaDB dataset.
4. ACRIMA dataset.
5. DRIONS dataset.
6. RIM ONE dataset

**Results-**

1. The k-nearest neighbours’ algorithm (K-NN) has accuracy of 99.2%, sensitivity of 86.7% and specificity of 84%.
2. Random Forest classifier has an accuracy of 95.3%, sensitivity of 96.31% and specificity of 95.33%.
3. Neural network classifiers-The binary classification has accuracy of jh89.3%, sensitivity of 89.5%, specificity of 88.9% and AUC 82%.

**Remarks-**

1. Here the papers that are analyzed proposed a one-step AI framework that did not require any segmentation of the fundus images, but we are thinking of segmentation of the fundus image.
2. Here AI is used for detection of glaucoma but we will use the concept of deep learning concept.
3. We use the following models like CNN Architecture, VGGNET-16, ResNet-50, Inception V3 etc.
4. They did not utilize the modality of fundus images.
5. Till now we have collected the datasets like ACRIMA, ORIGA, RIM ONE, FUNDUS\_TRAIN\_VAL\_Data etc.

The research paper [3]-

**Title- “Glaucoma Detection based on Deep Convolutional Neural Network”**

**Methodology-**

**1.** The procedure for detecting glaucoma, We consider the ROI image***(an area of an image that you want to filter)*** as the suggested deep Convolutional Neural Network's input in this paper (CNN).

**2**. Deep learning architectures called convolutional neural networks (CNNs) have lately been used successfully for picture segmentation and classification applications. Multilayer neural networks (NN) have evolved into DL architectures, which use various design and training methods to be competitive. These strategies include scalability, hierarchical feature learning, and spatial invariance.

**3.** In this study, here a deep convolutional neural network (CNN) that takes the ROI image as its input and outputs a smaller starting image that processes considerably faster than the disc and cup segments.

**Result:**

**1.** We compare CNN predictions to the most advanced reconstruction-based method in order to verify the efficiency of our deep CNN on the accuracy of glaucoma diagnosis.

**2.** CNN is used as the foundation for the deep learning architecture. CNN's network has six weighted layers, the first four of which are convolutional and the latter two of which are completely connected. A soft-max classifier for glaucoma prediction receives the output of the last fully connected layer. In our suggested learning architecture, response-normalization layers and overlapping layers are used, similar to. Our primary goal in this project is to successfully capture the deep features of glaucoma using deep CNN.

**3**. From the 650 photos in the ORIGA dataset, 99 images are randomly chosen for the training set, while the remaining 551 images are used for testing. We will use the 650 images from ORIGA for the SCES dataset's training, and the entire dataset's 1676 images will be used for the test. Each fundus image is diagnosed by a doctor, and our algorithm predicts the labels with probabilities.

**Remarks:**

The DL framework for glaucoma detection presented in this paper is built on deep CNN and is capable of capturing discriminative characteristics that more accurately represent the hidden patterns associated with glaucoma. But simultaneously, The VGGNET-16, ResNet-50, GoogleNet, and Ensemble Model algorithms will be used in addition to the CNN method. To successfully capture the deep features of glaucoma using deep CNN is our main objective in this study. After that, we will apply the other methods.

**The research paper-** [4]

**Title-** “**Machine learning classifiers for detection of glaucoma**”

**Methodology-**

1. They propose Support Vector Machine (SVM) method to segregate, train the models using a high-end graphics processor unit (GPU) and augmented the hull convex approach to boost the accuracy of the image processing mechanisms along with distinguishing the different stages of glaucoma. .

2. In this paper they detect glaucoma by the analysis of cup-disk ratio. The first part involves to build a machine learning model using a support vector machine and K-means classifier and the rest involves building a convolutional neural network for the same.

3. Convex hull using Sklansky algorithm is implemented and they used Drishti datasets

**Result:**

1. The CDR values of the test image were passed through the model and the predicted labels were matched with the labels in ground truth.
2. The SVM classifier was used first and an accuracy of 0.8539 was obtained.
3. Then the same pair of CDR values of the training set and its labels were passed through the K-means classifier with K=2 i.e. 2 clusters representing glaucoma and non-glaucoma sets. When the CDR values of the test data were passed through the same an accuracy of 0.7077 was obtained

**Remarks:**

1. As they used machine learning approach, we are thinking of working with deep learning.’
2. Our expected result is 90 to 95% accuracy.
3. We are thinking of use CNN architecture, VGGNET16 etc. algorithms for our deep learning approach

# Chapter 3 Methodology

## 3.1 System Design

The flowchart of our project’s design and Model Appliance:

**Project’s design:**

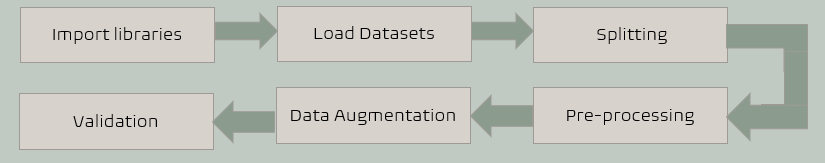


Figure 3: The flowchart of our project’s design

**Model Appliance:**

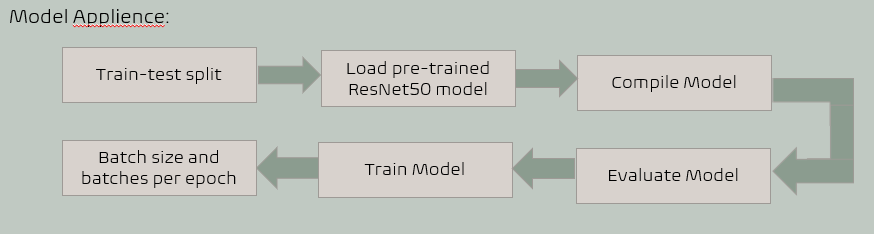


Figure 4: The flowchart of our model appliance

**Use Case Diagram:**

**A diagram of a username

Description automatically generated**

Figure 5: Use Case Diagram

**System Diagram:**

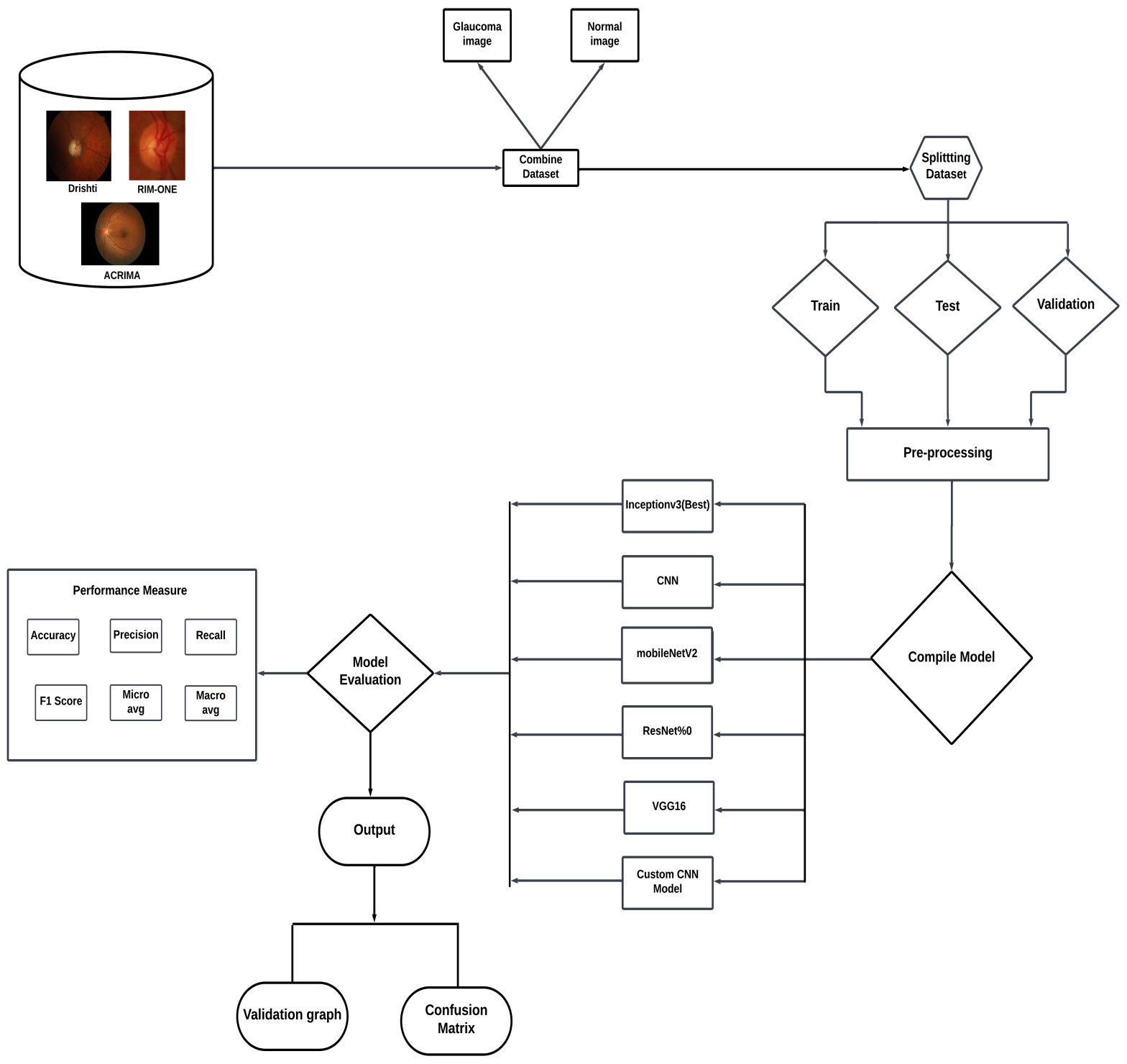
****

Figure 6: System Diagram

## 3.2 Software Components

We have used the following software components in our project-

1. **Google Colab:**

A cloud-based Python development environment that works in a web browser is called Google Colab (short for Google Collaboratory). You may develop, run, and share Python code using its Jupyter Notebook-like user interface. For machine learning and deep learning projects, Google Colab is especially helpful because it provides free access to GPU and TPU resources. Additionally, because Google Drive is integrated, you may save and load notebooks and data files straight from your Google Drive account.

1. **Python:**

Python is a high-level, interpreted programming language known for its readability, simplicity, and versatility. We have used Python Language as it is easy to understand and simple. Python emphasizes code readability, and executes the code directly. Python supports object-oriented, imperative, and functional programming paradigms. It is a general-purpose language that can be used for a wide range of applications, including web development, data science, machine learning, artificial intelligence, automation, scripting. Python community contributes to an extensive ecosystem of libraries and frameworks, such as Django for web development, NumPy and Pandas for data science, TensorFlow and PyTorch for machine learning. It is a platform-independent, meaning that Python code can run on different operating systems with little to no modification.

1. **Dataset:**

The Glaucoma identification dataset we are utilizing, which we downloaded focuses on using deep learning methods for the identification of glaucoma. Glaucoma is an eye condition that affects the optic nerve, impairing vision and, if untreated, potentially leading to blindness.

There are three types of datasets, which are ACRIMA, RIM-ONE, and DRISHTI. We combined this data set into two parts- 1. Glaucoma and 2. Normal. The dataset is split into three parts: Train, Test, and Val (Validation).

**DRISHTI dataset:**

The first dataset "**DRISHTI**" consists of total 101 retinal fundus images. In that dataset there are labeled two type images and one is for glaucoma images and another normal eye images. The glaucoma contains 70 images and normal contains 31 images.



Figure 7: DRISHTI image

**RIM-ONE dataset:**

The first dataset "**RIM ONE**" consists of total 485 retinographies images. In that dataset there are labeled two type images and one is for glaucoma images and another normal eye images. The glaucoma contains 172 images and normal contains 313 images.

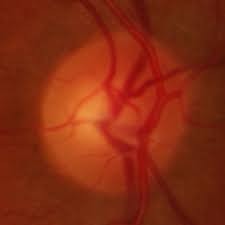


Figure 8: RIM ONE image

**ACRIMA dataset:**

On the other hand, the last one "**ACRIMA**" consists of total 705 fundus images. In that dataset there are labeled two type images and one is for glaucoma images and another normal eye images. The glaucoma contains 396 images and normal contains 309 images.

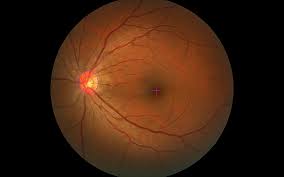


Figure 9: ACRIMA image

In order to automatically understand and recognize patterns, characteristics, and abnormalities connected to glaucoma, we want to build a deep learning model on this dataset. In order to categorize new, unseen images as either positive or negative for glaucoma, the model will learn from the labeled instances in the training set.

The ultimate objective of this work is to build a precise and trustworthy deep-learning model that can help in the early diagnosis of glaucoma, potentially assisting ophthalmologists in making prompt diagnoses and offering suitable therapies to stop or manage the condition.

There are 1289 images in total in the combined dataset. Where sub folder glaucoma has 638 images and normal has 653 images.

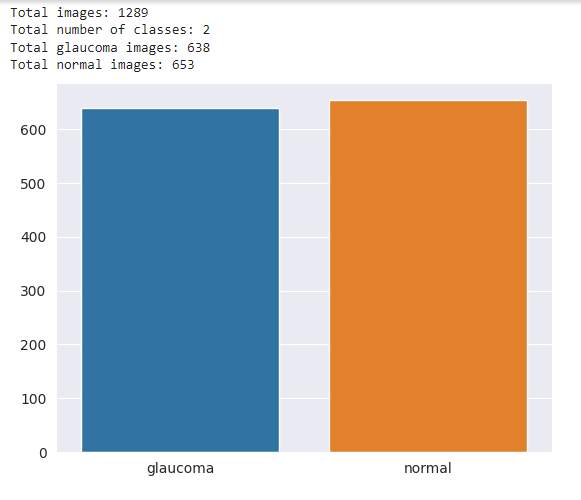


Figure 10: Graphical representation of glaucoma and normal image

To apply models on our applied dataset supportive libraries has been imported. Those libraries are-

**1. os:** The "os" library in Python provides a way to interact with the operating system. It allows you to perform various tasks related to file and directory manipulation, such as creating, deleting, renaming, or checking the existence of files and directories. It also provides functions to work with paths, environment variables, and execute system commands.

**2. cv2:** "cv2" refers to OpenCV, which stands for Open-Source Computer Vision Library. OpenCV is a popular computer vision library that provides a wide range of functions and algorithms for image and video processing. It offers tools for image manipulation, feature detection, object recognition, and various other computer vision tasks. The "cv2" library is a Python binding for OpenCV, allowing you to use OpenCV functionalities within your Python code.

**3. numpy:** “numpy” is a fundamental library for numerical computing in Python. It provides a powerful N-dimensional array object that enables efficient manipulation of large, multi-dimensional arrays and matrices. NumPy offers a wide range of mathematical functions and operations for array processing, including linear algebra, Fourier transforms, random number generation, and more. It is extensively used in scientific computing, data analysis, and machine learning applications.

**4. sklearn model selection:** The "sklearn" library, also known as scikit-learn, is a popular machine learning library in Python. It provides a comprehensive set of tools for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, and model selection. The "model\_selection" module within scikit-learn offers functions and classes for model selection and evaluation, such as splitting datasets into training and test sets, cross-validation, hyperparameter tuning, and performance metrics computation. It provides a convenient and standardized way to assess and compare the performance of different machine learning models.

**5. keras:** Keras is a high-level deep learning library that runs on top of other deep learning frameworks, such as TensorFlow or Theano. It provides a user-friendly and intuitive interface for

building and training neural networks. Keras allows you to define and customize various types of neural network architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more. It offers a wide range of pre-processing and data augmentation utilities, as well as support for GPU acceleration. Keras simplifies the process of designing and training deep learning models and has gained popularity for its ease of use and flexibility.

**6. tensorflow:** TensorFlow is an open-source library for machine learning and numerical computation developed by the Google Brain team. It is one of the most popular and widely used frameworks for building and deploying machine learning models. TensorFlow provides a comprehensive ecosystem of tools, libraries, and resources that aid in the development of various AI applications.

**Data Augmentation and Preprocessing:**

For data augmentation and pre-processing, we have used the following methods. Those are-

**Rescale:** Rescaling is an augmentation method that involves adjusting the scale or intensity of an image. It is commonly used to normalize the pixel values within a specific range, such as rescaling the pixel values from the original range of 0-255 to a new range of 0-1. Rescaling helps in standardizing the input data and making it suitable for certain machine learning algorithms or models.

**Shear:** Shear is a strategy utilized in deep learning and computer vision to enhance the quality of training data. It involves modifying images by shifting pixels along a diagonal axis, resulting in changes in the angles and shapes of objects, thereby improving the model's resilience. Training on these altered images equips the model to better identify objects from a range of angles and orientations, ultimately boosting its real-world performance. Data augmentation shear is just one of many tools employed to generate diverse training examples, ensuring deep learning models can proficiently handle a wide spectrum of real-world scenarios.

**Zoom**: Zoom augmentation involves magnifying or shrinking a portion of an image, giving the appearance of zooming in or out. This augmentation technique can be useful for simulating different viewpoints or capturing details at different scales. It helps to introduce variability and increase the robustness of a machine learning model by training it on images with varying zoom levels.

**Flip:** Flip augmentation involves flipping an image horizontally or vertically. Horizontal flipping involves reversing the order of pixels in each row, while vertical flipping reverses the order of rows in the image. Flipping is a common augmentation technique used to create variations of an image, especially when the orientation of the object is not crucial. It helps in training models that need to be invariant to horizontal or vertical flips, increasing the generalization capability of the model.

**ImageDataGenerator():** Keras ImageDataGenerator allows augment images in real-time while model is still training. Any random transformations on each training image as it is passed to the model. Main methods used in this utility are:

* rotation\_range: Degree range for random rotations.
* width\_shift\_range and height\_shift\_range: Fraction of total width or height by which the image can be shifted.
* shear\_range: Shear angle in radians.
* zoom\_range: Range for random zoom.
* horizontal\_flip and vertical\_flip: Boolean indicating whether to perform random horizontal and vertical flips.
  + - * ImageDataGenerator().flow()
      * flow\_from\_directory()

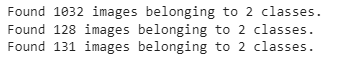
****

Figure 11: Image Count for train, validation and test dataset

**A total of 1289 images are used for training, validation, and testing purposes, with 638 glaucoma images and 653 normal images. We split the dataset into train, test, and validation. The ratio of splitting is 80%, 10% and 10%.**

**We applied six models for glaucoma detection.**

Those are-

1. **InceptionV3**
2. **MobileNetV2**
3. **Convolutional Neural Network (CNN)**
4. **ResNet50 Model**
5. **VGG16 Model**
6. **Custom CNN Model**
7. **InceptionV3:**
   1. InceptionV3 is a CNN architecture developed by Google.
   2. It is a deep learning model with 48 layers
   3. It utilizes inception modules, which consist of multiple parallel convolutional layers with different filter sizes.
   4. These modules allow the network to capture features at various scales and resolutions.
   5. InceptionV3 excels in image recognition tasks and is known for its accuracy and efficiency.
   6. It serves as a powerful base model for transfer learning, where its pre-trained weights are leveraged for tasks like fine-tuning and feature extraction.
8. **MobileNetV2:**
9. MobileNetV2 is a CNN architecture optimized for mobile and embedded devices.
10. The MobileNetV2 architecture is made up of a number of stacked inverted residual blocks.
11. MobileNetV2 is typically pre-trained on a large dataset of general-purpose images, such as ImageNet.
12. It employs depth wise separable convolutions to reduce computational complexity.
13. This model strikes a balance between accuracy and model size, making it suitable for resource-constrained environments.
14. MobileNetV2 is used for real-time image processing on mobile devices, including image classification and object detection tasks.
15. **Convolutional Neural Network (CNN):**
16. CNN is a class of deep neural networks designed for image processing tasks.
17. It uses convolutional layers to automatically extract features from input images.
18. Pooling layers reduce spatial dimensions while retaining important information.
19. Activation functions introduce non-linearity for capturing complex relationships.
20. Fully connected layers make predictions based on learned features.
21. Widely used in image classification, object detection, and image segmentation.
22. **ResNet50 Model:**
23. ResNet-50 is a powerful and versatile deep learning model that has been widely used in a variety of applications.
24. ResNet-50, short for Residual Network with 50 layers, is a deep convolutional neural network (CNN) architecture that belongs to the ResNet family.
25. The ResNet-50 architecture consists of four main stages, each of which consists of multiple residual blocks. The residual blocks are the key building blocks of ResNet, and they are responsible for the model's ability to learn deep features from images.
26. ResNet-50 has been pre-trained on a large dataset of images from the ImageNet database.
27. It is a popular choice for image classification tasks, and it has also been used for other tasks such as object detection and semantic segmentation.
28. **VGG16 Model:**
29. VGG16, short for Visual Geometry Group 16, is a convolutional neural network (CNN) architecture and consists of 16-layer deep learning model that has achieved state-of-the-art results on a variety of image classification tasks.
30. VGG16 was inspired by the success of AlexNet.
31. It consists of five blocks of convolutional layers, each followed by a max-pooling layer.
32. It has been pre-trained on a large dataset of images from the ImageNet database.
33. VGG16 can be used for a variety of image classification tasks, such as object recognition, scene classification, and image segmentation.
34. **Custom CNN Model:**
35. A custom CNN model is a convolutional neural network (CNN) architecture that is designed and trained specifically for a particular task or dataset.
36. It gives more accurate and efficient performance on the target application.
37. It can achieve higher accuracy than pre-trained models when the task and data align closely with the model's design.
38. It can be adapted to handle variations and complexities within the dataset, improving generalization.
39. Custom models can be tailored to the unique requirements of specific applications, such as medical image analysis or object detection in natural environments.

**Front-end design:**

An interface has been used for detecting glaucoma online by glaucoma detector. The link of the website for detecting glaucoma by using fundus image-[Glaucoma-Detector](https://glaucoma-detection.streamlit.app/?fbclid=IwAR1eSKZqaeef76FERLqJ49RKgOyR5bG9dcVmJfARzzHKcPMXjDfGG30rOoY)

**Glaucoma Detector:**

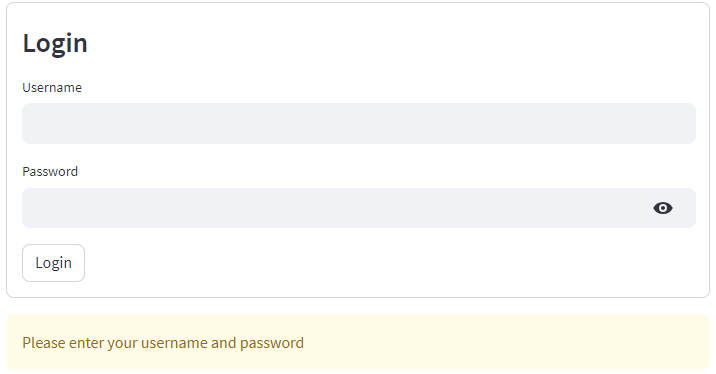


Figure 12: Login Page

Here shows user interfaces for a glaucoma detector. Firstly, the user enters the “Username” and “Password” in the login screen There is a button labeled “Login” below the fields.

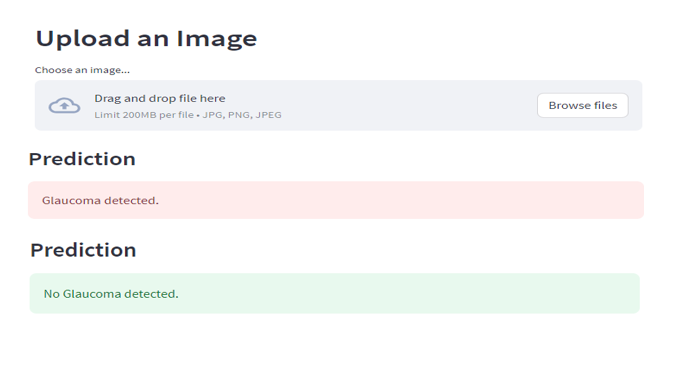


Figure 13: Detector Page

The image is taken from a web browser window and then uploads an image of their eye. The glaucoma detector then analyzes the image and predicts a result, which is displayed in the "Prediction" box. If glaucoma is found, it will say that there is “Glaucoma detected”. However, if glaucoma is not found, it will say that there is “No Glaucoma detected”.

**Key Features:**

This glaucoma detector website has the following features:

1. User-Friendly system
2. Fundus image detector
3. Cost-friendly for the user

Table I. List of Software Tools

|  |  |  |
| --- | --- | --- |
| **Tools** | **Functions** | **Why selected this tool** |
| **1.Google colab** | Colab comes with many popular Python packages and libraries pre-installed, eliminating the need for manual installations. It makes easy to import and export notebooks and data files | Colab allows you to share notebooks with others, facilitating collaboration and code sharing. It supports real-time colaboration. It provides convenient storage |
| **2. Python** | Python is a cross-platform language and it can be run on a variety of operating systems, including Windows, macOS, and Linux. | Python is a cross-platform language, which means that it can be run on a variety of operating systems. It well-established and mature language with a large and active community. It is easy to learn and use, making it a good choice for beginners. It has a rich library ecosystem, including libraries for data analysis, machine learning, and deep learning. |

## Software Implementation

The development of software for glaucoma detection involves the integration of various modules and components to achieve accurate and reliable results. Here are some common software modules used in glaucoma detection systems:

**Image acquisition module:**

Responsible for capturing high-quality images of the eye, typically using imaging devices such as fundus cameras.

**Image preprocessing module:**

Cleans and enhances the acquired images to improve the quality. This may involve tasks such as contrast enhancement, and image registration.

**Deep learning module:**

Utilizes algorithms, deep learning architectures to analyze the extracted features and classify the likelihood of glaucoma. Common technique is convolutional neural networks (CNNs).

**Risk assessment module:**

Integrates the results from the deep learning module to provide an overall risk assessment or probability score for the presence of glaucoma. This can help in prioritizing cases for further clinical evaluation.

**User interface (UI):**

Provides a user-friendly interface for clinicians and healthcare professionals to interact with the system. This predicts whether there is glaucoma or not.

**Reporting Module:**

Generates comprehensive reports summarizing the analysis results, including visualizations of segmented regions, extracted features, and the final glaucoma diagnosis or risk assessment.

**Quality control module:**

Monitors and ensures the quality of the acquired images and the overall performance of the system. This may involve checks for image artifacts, system calibration, and validation against known datasets.

Developing an application for glaucoma detection involves creating a software solution that can analyze eye images to identify potential signs of glaucoma. Here is a high-level overview of the steps involved in developing a glaucoma detection application:

1. **Data Collection:**

* Three types of datasets DRISHTI, RIM-ONE and ACRIMA datasets of eye images labeled with information about whether each eye has glaucoma or not.
* Then these datasets are combined and finally creating categories of normal and glaucoma images
* . These datasets are further splitted into test, train, and validation sets.

1. **Data Preprocessing:**

* The images are preprocessed to ensure they are in a suitable format for the model.
* Common preprocessing steps include resizing images, normalization, and augmentation.
* Data augmentation is applied using Image Data Generator.

1. **Model Selection:**

* Appropriate model architectures are selected for glaucoma detection.
* Various deep learning models, including CNN, InceptionV3, MobileNetV2, ResNet50, VGG16, and a custom CNN model, are selected and trained using labeled data with appropriate loss functions.

1. **Model Training:**

* Preprocessed dataset is trained using labeled data with appropriate loss functions and performance are evaluated.

1. **Evaluation:**

* The model's performances are assessed using metrics such as accuracy, accuracy, precision, recall, and f1 score.

1. **Validation:**

* The best-performing models are validated. Validation graphs and confusion matrices are also generated to assess model performance.

1. **Integration with a User Interface:**

* Design and implement a user interface for your application. This can be a web-based interface, a mobile app, or a desktop application. Users should be able to upload eye images for analysis.

1. **Ongoing Improvement:**

* Continuously models are updated and dataset to enhance accuracy and reliability.

Developing a glaucoma detection application requires expertise in machine learning, software development, and collaboration with domain experts in ophthalmology. Additionally, adherence to ethical standards and medical regulations is essential to ensure the safety and effectiveness of the application.

# Chapter 4 Investigation/Experiment, Result, Analysis and Discussion

In glaucoma detection experiments, various variables are considered to assess the performance of different models. Here's a description of the key experiments addressing multiple variables in glaucoma detection:

**Data Collection and Preprocessing:**

**Variables Addressed:**

1. Image resolution
2. Dataset size
3. Image augmentation techniques

**Description:**

1. Experiments involve collecting retinal images from diverse sources with varying resolutions.
2. Dataset sizes are adjusted to explore the impact of data volume on model performance.
3. Different image augmentation techniques, such as rescale, resize, and flip, are applied to enhance model robustness.

**Model Selection and Architecture:**

**Variables Addressed:**

1. Model architecture (CNN, InceptionV3, MobileNetV2, ResNet50, VGG16, custom CNN)

**Description:**

1. Various pre-trained models (InceptionV3, MobileNetV2, ResNet50, VGG16) and a custom CNN are chosen to compare their effectiveness.

**Training Process:**

**Variables Addressed:**

1. Number of training epochs
2. Early stopping criteria

**Description:**

1. Models are trained over varying numbers of epochs to understand convergence and overfitting.
2. Early stopping is employed to prevent overfitting and determine the optimal number of training epochs.

**Evaluation Metrics:**

**Variables Addressed:**

1. Accuracy
2. Precision
3. Recall
4. F1 score

**Description:**

1. Models are evaluated using multiple metrics precision, recall, and F1 score to provide a comprehensive assessment of their performane.

**Validation and Generalization:**

**Variables Addressed:**

1. Model generalization across datasets

**Description:**

1. Models are tested on validation datasets to assess their ability.

**Visualizations and Interpretability:**

**Variables Addressed:**

1. Validation graphs
2. Confusion matrices

**Description:**

1. Validation graphs illustrate model training and validation performance over epochs.
2. Confusion matrices provide insights into model errors.

**Various results of the** **Applied Models**:

**InceptionV3:**

Imported InceptionV3 from tensorflow.keras.application. where our input shape of the images is (256,256,3). We set out pooling to average. Also used relu as an addition layer on the top of the model. This model give us best accuracy then all other models. Able to detect both glaucoma and normal images very well,highest accuracy and precision and recall are also satisfactory.

**Validation Graph of InceptionV3**:

A graph of a graph of a graph

Description automatically generated with medium confidence

Figure 14: Validation Graph of InceptionV3

**Confusion Matrix:**

**A blue and white squares with white text

Description automatically generated**

Figure 15: Confusion Matrix of InceptionV3

**Result:**

From this model we find out 72 % accuracy. The precision, recall and f1 score are 0.73,0.73 and 0.72.

**A screenshot of a graph

Description automatically generated**

Figure 16: Test score of InceptionV3

**CNN (Convolutional Neural Network):**

Imported CNN from tensorflow.keras.layers, where our input shape of the images is (256,256,3). We set out pooling to average. Also used relu as an addition layer on the top of the model. This model give second best accuracy from all others model.

**Validation Graph of CNN:**

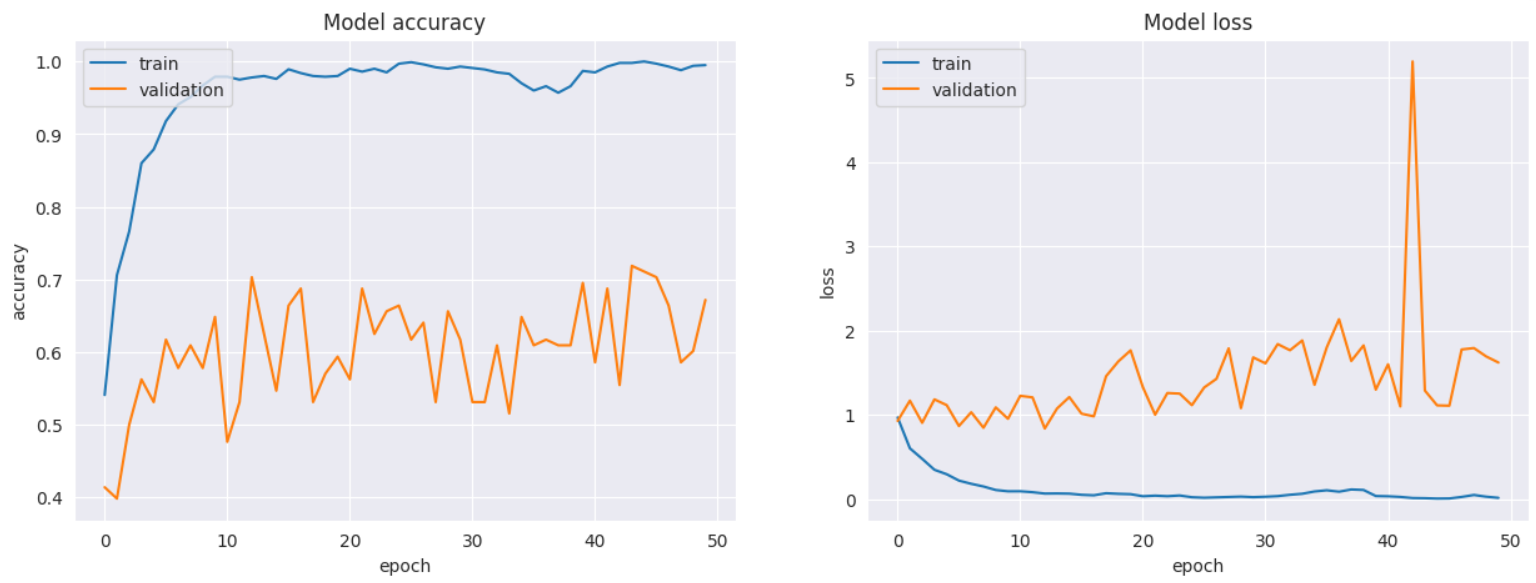


Figure 17: Validation Graph of CNN

**Confusion Matrix:**

**A screenshot of a graph

Description automatically generated**

Figure 18: Confusion Matrix of CNN

**Result:**

From this model we find out 69 % accuracy. The precision, recall and f1 score are 0.69,0.69 and 0.69.

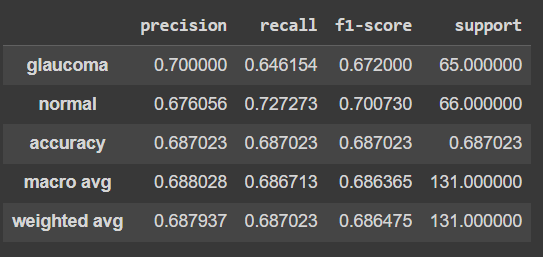
****

Figure 19: Test score of CNN

**MobilenetV2 :**

Imported mobilenetV2 from tensorflow.keras.application , where our input shape of the images is (256,256,3). We set out pooling to average. Also used relu as an addition layer on the top of the model.

**Validation Graph of MobilenetV2:**

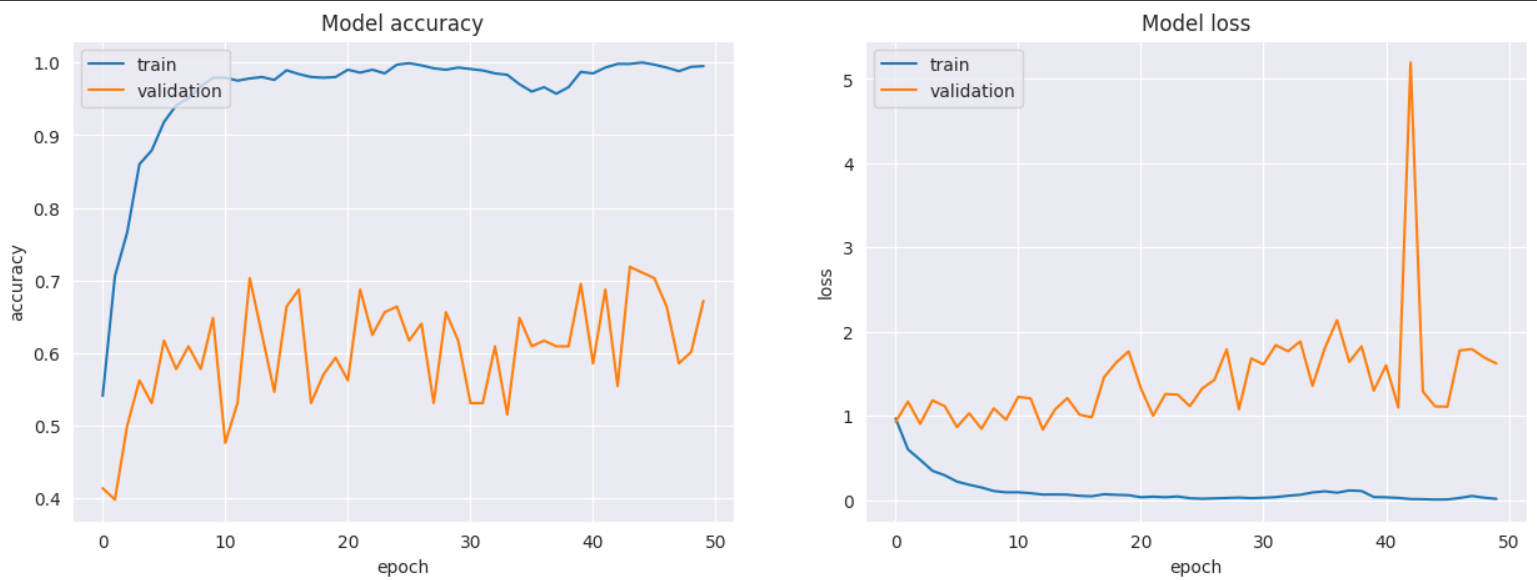
****

Figure 20: Validation Graph of MobilenetV2

**Confusion Matrix:**

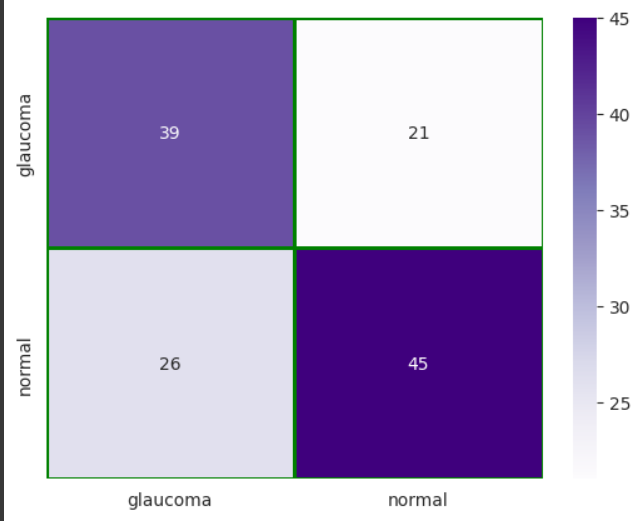
****

Figure 21: Confusion Matrix of MobilenetV2

**Result:**

From this model we find out 64 % accuracy. The precision, recall and f1 score are 0.64,0.64 and 0.64.

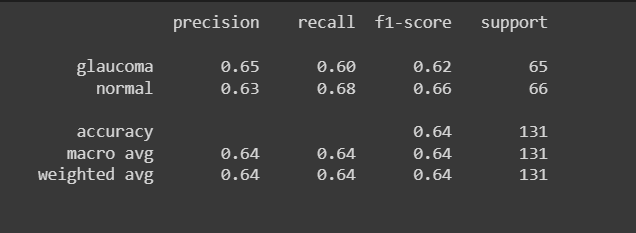
****

Figure 22: Test score of MobilenetV2

**Restnet50:**

Imported Restnet50 from tensorflow.keras.layers , where our input shape of the images is (256,256,3). We set out pooling to average. Also used relu as an addition layer on the top of the model.

**Validation Graph of Restnet50:**

****

Figure 23: Validation Graph of Restnet50

**Confusion Matrix:**

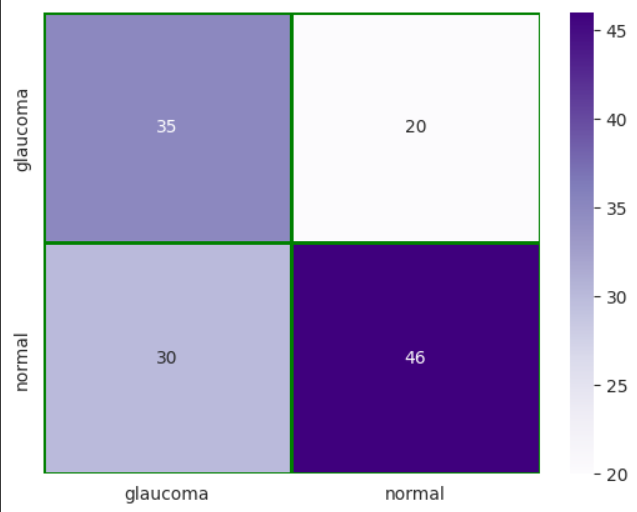
****

Figure 24: Confusion Matrix of Restnet50

**Result :** From this model we find out 62 % accuracy. The precision, recall and f1 score are 0.63,0.62 and 0.62.

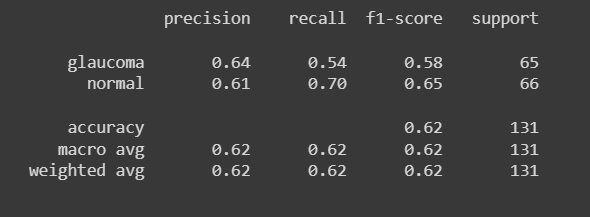
****

Figure 25: Test score of Restnet50

**VGG16 :**

Imported VGG16 from tensorflow.keras.layers, where our input shape of the images is (256,256,3). We set out pooling to average. Also used relu as an addition layer on the top of the model.

**Validation Graph of VGG16:**

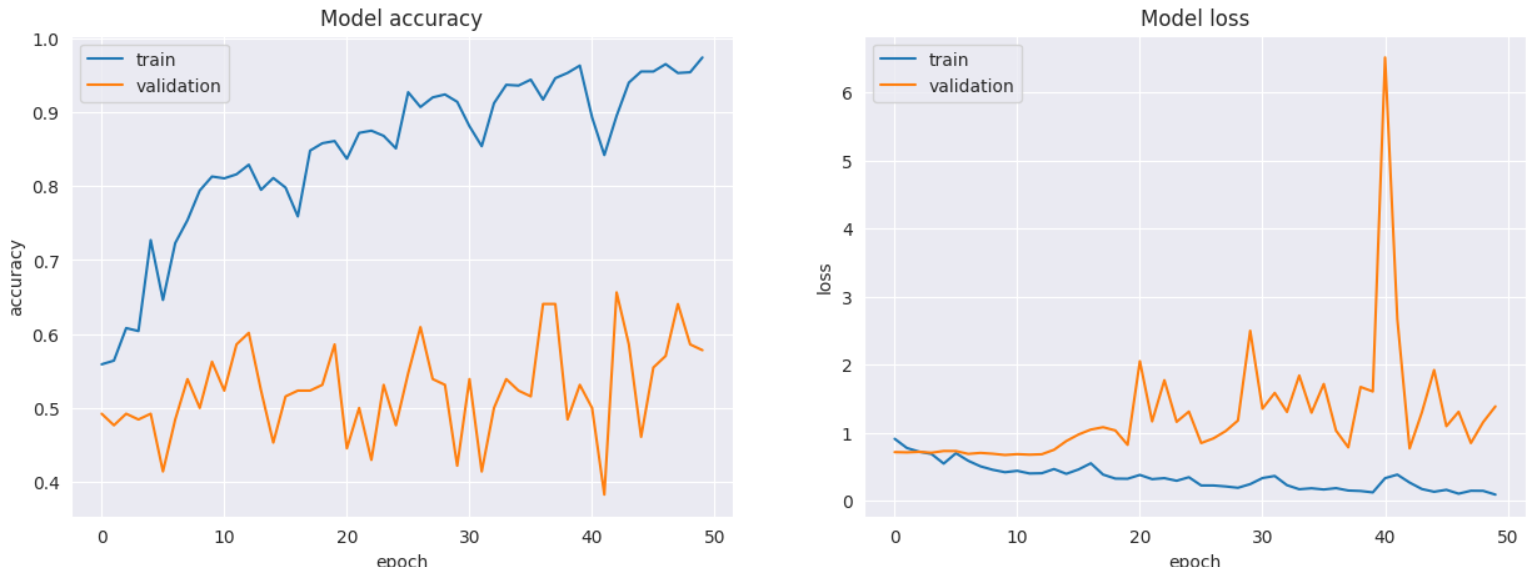
****

Figure 26: Validation Graph of VGG16

**Confusion Matrix:**

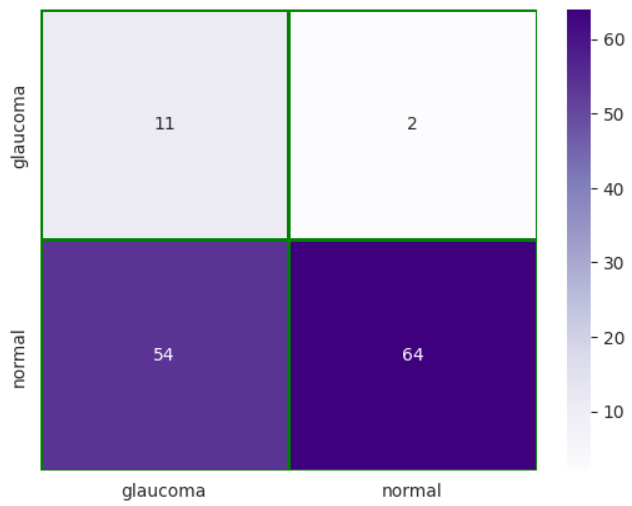
****

Figure 27: Confusion Matrix of VGG16

**Result :**

From this model we find out 57 % accuracy. The precision, recall and f1 score are 0.70,0.57 and 0.49.

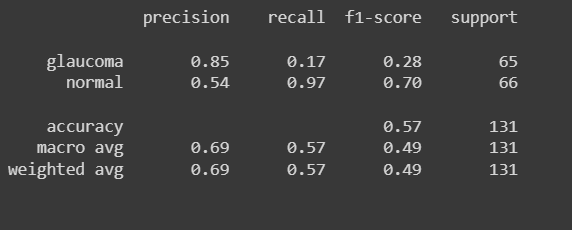
****

Figure 28: Test score of VGG16

**Custom CNN :**

Imported CUSTOM CNN from tensorflow.keras.layers, where our input shape of the images is (256,256,3). We set out pooling to average. Also used relu as an addition layer on the top of the model.

**Validation graph of Custom CNN Model:**

**A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated**

Figure 29: Validation graph of Custom CNN Model

**Confusion Matrix:**

**A screenshot of a chart

Description automatically generated**

Figure 30: Confusion Matrix of Custom CNN Model

**Result :**

From this model we find out 47 % accuracy. The precision, recall and f1 score are 0.47,0.47 and 0.47.

**A screenshot of a graph

Description automatically generated**

Figure 31: Confusion Matrix of Custom CNN Model

**Result using comparative analysis and Discussion:**

**InceptionV3 Model (Accuracy: 72%):**

1. InceptionV3 achieved the highest accuracy, indicating it is the most effective at making correct predictions.
2. Precision, recall, and F1 score are all at 0.73 and 0.72, showing a well-balanced performance with relatively few false positives and false negatives. This means it has a strong capability to correctly identify positive and negative cases with high precision and recall.

**CNN Model (Accuracy: 69%):**

1. The CNN model performed well with an accuracy of 69%, slightly below InceptionV3 but still solid.
2. Precision, recall, and F1 score are all at 0.69, indicating a well-balanced performance. This suggests it can effectively distinguish between positive and negative cases with a good trade-off between precision (few false positives) and recall (few false negatives).

**MobileNetV2 Model (Accuracy: 64%):**

1. MobileNetV2 achieved a decent accuracy of 64%, though it lags behind InceptionV3 and the CNN model.
2. Precision, recall, and F1 score are all at 0.64, indicating a consistent but slightly lower level of performance. This model performs reasonably well in terms of both precision and recall.

**ResNet50 Model (Accuracy: 62%):**

1. The ResNet50 model demonstrates moderate accuracy at 62%, but it's not as high as InceptionV3 or the CNN model.
2. Precision, recall, and F1 score are close to each other but slightly lower, at 0.63, 0.62, and 0.62, respectively. This suggests that while precision is relatively good, recall and the F1 score are somewhat lower.

**VGG16 Model (Accuracy: 57%):**

1. VGG16 had the lowest accuracy among the models at 57%, indicating that it's the least accurate in making correct predictions.
2. Precision is relatively high at 0.70, which means it doesn't often produce false positives. However, recall and F1 score are lower at 0.57 and 0.49, respectively, implying that it tends to miss some positive cases (false negatives).

**Custom CNN Model (Accuracy: 47%):**

1. The custom CNN model has the lowest accuracy among all models, at 47%, suggesting it needs improvements.
2. Precision, recall, and F1 score are all at 0.47, indicating a balanced but relatively low overall performance. This model is not as effective at correctly identifying positive and negative cases.

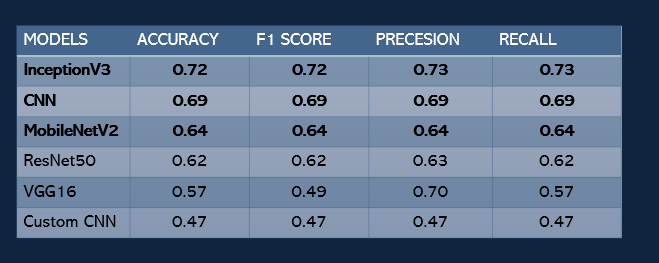


Figure 32: Comparative analysis of models

As a result of the highest accuracy and well-balanced precision-recall trade-off, InceptionV3 is the most successful best fitted model overall. Its 72% accuracy rate demonstrates its ability to classify data accurately, and its 0.73, 0.72, and precision, recall, and F1 score values imply that it reduces the number of false positives and false negatives, making it a reliable option for accurate predictions.

However, selecting a model should involve more than just performance indicators. Important factors to take into account include things like training time, computational resources, and application-specific requirements. In contexts with limited resources, InceptionV3 could require a significant amount of processing power and time for training. Thus, in order to choose the best model, a practical resource assessment is essential.

**Benefits of using InceptionV3** **are the following-**

* 1. Able to detect both glaucoma and normal images very well.
  2. Have highest accuracy
  3. Precision and recall are also satisfactory**.**
  4. Relatively efficient, means that it can be trained and used on large datasets without requiring a lot of computational resources.
  5. Used for a variety of image classification tasks, such as classifying objects, scenes, and faces.
  6. Used to develop models that can accurately classify cancer cells and other medical conditions.
  7. Used to develop models that can accurately identify objects.

It's also important to recognize that other models can be useful in different situations even if they don't perform as well as InceptionV3. They might have benefits like faster training times or less computational requirements, making them better suited for particular applications. To improve their performance and possibly close the performance difference with the best model, more optimization work can be done.

In the end, selecting a model should be a wise option that takes into consideration practical and performance limitations to make sure it fits the demands of the particular task and the resources at hand.

**Novelty of Project**

1. The key idea of this project is to determine glaucoma using popular deep neural networks named InceptionV3, CNN and MobileNetV2.
2. The dataset we used here is a combination of three popular datasets used to detect glaucoma.
3. Finally, we made the detection procedure much easier with the help of a website.

# Chapter 5 Impacts of the Project

## 5.1 Impact of this project on societal, health, safety, legal and cultural issues

Glaucoma is a leading cause of blindness worldwide, affecting over **70 million** people. It is a progressive eye disease that damages the optic nerve, leading to loss of vision. There is no cure for glaucoma, but it can be treated to slow or prevent vision loss.

In our project we have used  deep learning process to detect glaucoma in its early stages. This project has the potential to revolutionize glaucoma care by making it easier and more affordable to diagnose and treat the disease.

Our project has the following societal, health, safety, legal, and cultural impacts:

**Societal impact:**

The glaucoma detection project has the potential to improve the quality of life for millions of people around the world. By detecting glaucoma early, the project can help to prevent blindness and improve vision for people with the disease. This can lead to increased productivity, reduced healthcare costs, and improved quality of life.

**Health impact:**

Our project has the potential to save lives. By detecting glaucoma early, the project can help to prevent blindness and other serious eye diseases. This can lead to improved health outcomes for people with glaucoma and their families.

**Safety impact:**

The glaucoma detection project has the potential to improve safety. By detecting glaucoma early, the project can help to prevent accidents and injuries that are caused by impaired vision. This can lead to a safer environment for everyone.

**Legal impact:**

The glaucoma detection project has the potential to change the law. By making it easier and more affordable to diagnose and treat glaucoma, the project could lead to new laws that protect the rights of people with the disease. This could include laws that require employers to provide accommodations for people with glaucoma and laws that require insurance companies to cover glaucoma treatment.

**Cultural impact:**

The glaucoma detection project has the potential to change the way we think about glaucoma. By making the disease more visible, the project could help to break down the stigma associated with glaucoma and encourage people to get screened for the disease. This could lead to earlier diagnosis and treatment, which could save lives and improve vision.

Overall, our project has the potential to make a significant impact on society. The project has the potential to improve the quality of life for millions of people, save lives, improve safety, change the law, and change the way we think about glaucoma.

## 5.2 Impact of this project on environment and sustainability

Our 'Glaucoma Detection based on Deep Learning' project has the potential to have a positive impact on the environment and sustainability in several ways.

First, the project could help to reduce the number of people who become blind due to glaucoma. Blindness is a major environmental problem, as it can lead to social isolation, economic hardship, and increased vulnerability to accidents. By reducing the number of people who become blind, the glaucoma detection project could help to improve the environment for everyone.

Second, the project could help to reduce the amount of waste generated by the healthcare system. Currently, glaucoma is diagnosed and treated using a variety of methods, including eye exams, imaging tests, and medications. These methods can generate a significant amount of waste, including medical waste, paper waste, and plastic waste. The glaucoma detection project could help to reduce this waste by making it easier and more affordable to diagnose and treat glaucoma.

Third, the project could help to improve the sustainability of the healthcare system. The healthcare system is a major contributor to environmental problems, as it consumes a significant amount of resources, including energy, water, and land. The glaucoma detection project could help to improve the sustainability of the healthcare system by making it more efficient and less wasteful.

Overall, the glaucoma detection project has the potential to have a positive impact on the environment and sustainability. The project could help to reduce blindness, reduce waste, and improve the sustainability of the healthcare system.

So we can summarize the environmental and sustainability benefits mentioned above as these:

* **Economic benefits:** The project could lead to economic benefits by reducing the cost of healthcare, increasing productivity, and improving quality of life.
* **Social benefits:** The project could lead to social benefits by reducing social isolation, improving economic opportunity, and enhancing quality of life.
* **Cultural benefits:** The project could lead to cultural benefits by raising awareness of glaucoma, reducing stigma, and promoting inclusion.

Overall, the glaucoma detection project has the potential to make a significant positive impact on the environment, sustainability, and society.

# Chapter 6 Project Planning and Budget

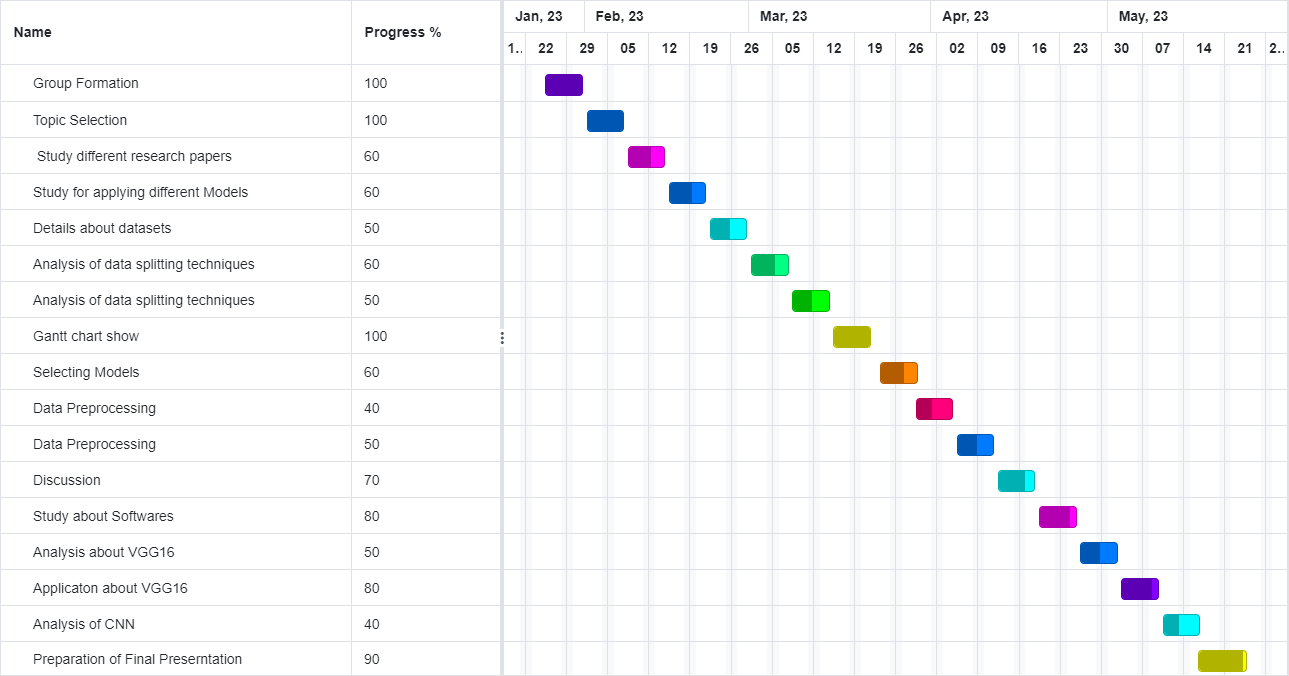


Figure 33: Gantt chart

The best aspect is that using software solutions to carry out the full process doesn't cost anything. Because it runs only on computer programs and digital solutions, people cannot pay for it. In keeping with a dedication to affordability and accessibility, this project makes use of technology to offer its benefits without putting any financial strain on users.

# Chapter 7 Complex Engineering Problems and Activities

## 7.1 Complex Engineering Problems (CEP)

## 

Table II. A Complex Engineering Problem Attributes

|  |  |  |
| --- | --- | --- |
| **Attributes** | | **Addressing the complex engineering problems (P) in the project** |
| P1 | Depth of knowledge required (K3-K8) | The project requires knowledge of Engineering Fundamentals and Deep Learning(K3), Machine Learning, Dataset, Python, Data Processing, Image processing, Neural Network(K4), System Designing (K5), Model Design, Model Compilation & Evaluation, Performance Score & High Power Computing Device (K6), Involve User Survey, Data Filtering, Engineering Impacts (K7), Research on Scientific Research Papers for Glaucoma & neural networks (K8). |
| P2 | Range of conflicting requirements | The Output of the system’s training depends on data augmentation. Testing depends on image quality. |
| P3 | Depth of analysis required | No unique way to detect. Depth of analysis needed to select a specific model from many alternatives.  (InceptionV3/CNN/mobilenetv2/Resnet50/VGG16/Custom CNN model) |
| P4 | Familiarity of issues | Inconsitant model performance for different dataset  Some dataset suits some models better. |
| P5 | Extent of applicable codes | There is no existing code or standard for this project. |
| P6 | Extent of stakeholder involvement | There are several stakeholders needs to be involved including the patients, hospital authorities. |
| P7 | Interdependence | Project involves a number of interdependent sub-systems such as dataset, data augmentation, models, training and testing. |

## 7.2 Complex Engineering Activities (CEA)

Table III. A Complex Engineering Problem Activities

|  |  |  |
| --- | --- | --- |
| **Attributes** | | **Addressing the complex engineering activities (A) in the project** |
| A1 | Range of resources | This project involves human resource, modern tools (compilation and evaluation of models), etc. |
| A2 | Level of interactions | Involves interactions between different stakeholders including group members to customize the models running simulation and tests, analysis of the best fitted model and other information. |
| A3 | Innovation | Employs innovative skills of engineering by introducing technology in a different manner in the medical sector |
| A4 | Consequences to society  / Environment | Impact in our society since it helps to detect early symptoms of glaucoma |
| A5 | Familiarity | Needs to be familiar with machine learning, neural network, data processing, glaucoma, image processing etc. |

# Chapter 8 Conclusions

## 8.1 Summary

We find this encouraging, especially given the neural approach's entirely supervised nature, which learns from raw pixel data and does not rely on any prior domain knowledge on vessel structure. During learning, a network autonomously extracts low-level features that are invariant to small geometric variations, then gradually transforms and combines them into higher order features. So, our project is working on different models using fundus images for detecting glaucoma. It will help to detect glaucoma at the early stages which will be very helpful in the medical field. And it will also help to know which models can be the more accurate ones.

## 8.2 Limitations

Glaucoma detection through deep learning faces several key limitations. Firstly, the computational intensity of InceptionV3, with its substantial number of parameters, may pose challenges for deployment on resource-constrained devices. While our system achieved accuracy below 85% with three applied models, further testing on diverse datasets is crucial for more accurate metrics. Future iterations should explore additional models and datasets to enhance results. The system's performance may be hindered by the limited and homogeneous dataset, leading to suboptimal model performance and potential overfitting concerns.

Additionally, the model size of InceptionV3 may limit deployment on devices with constrained storage, whereas MobileNetV2, designed for lightweight applications, could be more suitable. In real-time applications, InceptionV3's longer inference times may be a constraint compared to the faster MobileNetV2. Balancing accuracy and efficiency are pivotal, and the selection of the appropriate convolutional neural network architecture is crucial for project success.

Continuous model updates are necessary, given the potential obsolescence of pretrained models and the emergence of new versions. Ensuring optimal performance requires updated codes, techniques, and faster runtime. Despite these challenges, our glaucoma detection project stands to benefit significantly from the strengths of InceptionV3, with an awareness of its limitations guiding future improvements and refinements.

## 8.3 Future Improvement

Our primary focus is to develop a new, non-invasive, and affordable glaucoma screening tool that can be used for mass usage. Our tool will use artificial intelligence to analyze images of the retina, the light-sensitive tissue at the back of the eye, to identify signs of glaucoma. We believe that our glaucoma detection tool has the potential to revolutionize the way glaucoma is diagnosed and managed. AI models will continually adapt and learn from evolving medical knowledge and new data. Mechanisms are developed for models to stay current and accurate over the time. By making early detection more accessible and affordable, we can help to prevent millions of people from losing their sight.

# References

|  |  |
| --- | --- |
| [1] | J. C. Y. X. J. L. Huazhu Fu, "Glaucoma Detection based on Deep Learning," in *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*, 2019, pp. 119-137. |
| [2] | M. a. B. M. W. P. b. V. K. Lauren J. Coan, "Automatic detection of glaucoma via fundus imaging and artificial intelligence: A review," *Survey of Ophthalmology,* vol. 68, no. 1, pp. 17-41, 2023. |
| [3] | X. Chen, Y. Xu, D. W. K. Wong, T. Y. Wong and J. Liu, "Glaucoma detection based on deep convolutional neural network," in *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015. |
| [4] | L. S. B. H. Reshma Verma, "Machine learning classifiers for detection of glaucoma," *IAES International Journal of Artificial Intelligence (IJ-AI),* vol. 12, pp. 806-814, 2023. |
| [5] | J. C. Y. X. J. L. Huazhu Fu, "Glaucoma Detection based on Deep Learning," in *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*, 2019, pp. 119-137. |

# Appendices

**Drive Mount:**

from google.colab import drive

drive.mount('/content/drive')

# from google.colab import drive

# drive.flush\_and\_unmount()

**Importing necessary libraries:**

import matplotlib.pyplot as plt

import numpy as np

import os

import shutil

from tensorflow import keras

import seaborn as sns

import random

from keras.models import load\_model

from keras.preprocessing import image

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

import cv2

import pandas as pd

from sklearn.metrics import confusion\_matrix, classification\_report

import seaborn as sb

import tensorflow as tf

from sklearn.metrics import classification\_report,confusion\_matrix

import seaborn as sb

**Data collection & exploration:**

# current\_dir = os.getcwd()

# print(current\_dir)

**Here are the following codes for splitting three types of datasets:**

**a. DRISTHI**

train\_glaucoma\_dir = "/content/drive/MyDrive/PROJECT/Glaucoma Final/archive/Training-20211018T055246Z-001/Training/Images/GLAUCOMA"

train\_normal\_dir = "/content/drive/MyDrive/PROJECT/Glaucoma Final/archive/Training-20211018T055246Z-001/Training/Images/NORMAL"

test\_glaucoma\_dir = "/content/drive/MyDrive/PROJECT/Glaucoma Final/archive/Test-20211018T060000Z-001/Test/Images/glaucoma"

test\_normal\_dir = "/content/drive/MyDrive/PROJECT/Glaucoma Final/archive/Test-20211018T060000Z-001/Test/Images/normal"

dristhi\_glaucoma\_images = os.listdir(train\_glaucoma\_dir)+os.listdir(test\_glaucoma\_dir)

dristhi\_normal\_images = os.listdir(train\_normal\_dir)+os.listdir(test\_normal\_dir)

# Look at the number of samples in each dataset

print("Dristhi dataset contains :")

print(f"\t{len(dristhi\_glaucoma\_images)} images representing an eye with glaucoma")

print(f"\t{len(dristhi\_normal\_images)} images representing a normal eye")

print("Sample Dristhi glaucoma images:")

plt.subplots(figsize=(15, 10))

for i in range(1, 5):

    plt.subplot(1, 4, i)

    plt.imshow(load\_img(f"{os.path.join(train\_glaucoma\_dir, dristhi\_glaucoma\_images[i - 1])}"))

plt.show()

print("\nSample Dristhi normal images:")

plt.subplots(figsize=(15, 10))

for i in range(1, 5):

    plt.subplot(1, 4, i)

    plt.imshow(load\_img(f"{os.path.join(train\_normal\_dir, dristhi\_normal\_images[i - 1])}"))

plt.show()

**b. RIM-ONE**

rimOne\_dir = '/content/drive/MyDrive/PROJECT/Glaucoma Final/RIM-ONE\_DL\_images/RIM-ONE\_DL\_images/partitioned\_randomly/'

train\_glaucoma\_dir = rimOne\_dir + "training\_set/glaucoma"

train\_normal\_dir = rimOne\_dir + "training\_set/normal"

test\_glaucoma\_dir = rimOne\_dir + "test\_set/glaucoma"

test\_normal\_dir = rimOne\_dir + "test\_set/normal"

rimOne\_glaucoma\_images = os.listdir(train\_glaucoma\_dir)+os.listdir(test\_glaucoma\_dir)

rimOne\_normal\_images = os.listdir(train\_normal\_dir)+os.listdir(test\_normal\_dir)

# Look at the number of samples in each dataset

print("Rim One dataset contains :")

print(f"\t{len(rimOne\_glaucoma\_images)} images representing an eye with glaucoma")

print(f"\t{len(rimOne\_normal\_images)} images representing a normal eye")

print("Sample Rim-One glaucoma images:")

plt.subplots(figsize=(15, 10))

for i in range(1, 5):

    plt.subplot(1, 4, i)

    plt.imshow(load\_img(f"{os.path.join(train\_glaucoma\_dir, rimOne\_glaucoma\_images[i - 1])}"))

plt.show()

print("\nSample Rim-One normal images:")

plt.subplots(figsize=(15, 10))

for i in range(1, 5):

    plt.subplot(1, 4, i)

    plt.imshow(load\_img(f"{os.path.join(train\_normal\_dir, rimOne\_normal\_images[i - 1])}"))

plt.show()

**c. ACRIMA:**

# acrima\_dir = "/content/drive/MyDrive/PROJECT/Glaucoma Final/Database/Images"

# acrima\_main\_dir = '/content/drive/MyDrive/PROJECT/Glaucoma Final/Database/Images'

# os.makedirs(acrima\_main\_dir + '/glaucoma')

# os.makedirs(acrima\_main\_dir + '/normal')

# for f in os.listdir(acrima\_dir):

#   if 'g' in (f.split('.')[0]):

#     file\_path = os.path.join(acrima\_dir,f)

#     img = cv2.imread(file\_path)

#     cv2.imwrite('/content/drive/MyDrive/Glaucoma Datasets/Database/Database/glaucoma/' + f, img)

#   else:

#     file\_path = os.path.join(acrima\_dir,f)

#     img = cv2.imread(file\_path)

#     cv2.imwrite('/content/drive/MyDrive/Glaucoma Datasets/Database/Database/normal/' + f, img)

# acrima\_dir = current\_dir + "/drive/MyDrive/datasets/acrima/Database"

glaucoma\_dir = "/content/drive/MyDrive/PROJECT/Glaucoma Final/Database/glaucoma"

normal\_dir = "/content/drive/MyDrive/PROJECT/Glaucoma Final/Database/normal"

normal\_images = os.listdir(normal\_dir)

glaucoma\_images = os.listdir(glaucoma\_dir)

# Look at the number of samples in each dataset

print("Acrima dataset contains : ")

print(f"\t{len(glaucoma\_images)} images representing an eye with glaucoma")

print(f"\t{len(normal\_images)} images representing a normal eye")

print("Sample glaucoma images:")

plt.subplots(figsize=(15, 10))

for i in range(1, 5):

    plt.subplot(1, 4, i)

    plt.imshow(load\_img(f"{os.path.join(glaucoma\_dir, glaucoma\_images[i - 1])}"))

plt.show()

print("\nSample normal images:")

plt.subplots(figsize=(15, 10))

for i in range(1, 5):

    plt.subplot(1, 4, i)

    plt.imshow(load\_img(f"{os.path.join(normal\_dir, normal\_images[i - 1])}"))

plt.show()

**Combining datasets:**

## define your paths for glaucoma####

# g\_path1 = '/content/drive/MyDrive/Glaucoma Datasets/archive/Test-20211018T060000Z-001/Test/Images/glaucoma'

# g\_path2 = '/content/drive/MyDrive/Glaucoma Datasets/Database/Database/glaucoma'

# g\_path3 ='/content/drive/MyDrive/Glaucoma Datasets/RIM-ONE\_DL\_images/RIM-ONE\_DL\_images/partitioned\_randomly/training\_set/glaucoma'

# g\_path4='/content/drive/MyDrive/Glaucoma Datasets/RIM-ONE\_DL\_images/RIM-ONE\_DL\_images/partitioned\_randomly/test\_set/glaucoma'

# g\_path5='/content/drive/MyDrive/Glaucoma Datasets/archive/Training-20211018T055246Z-001/Training/Images/GLAUCOMA'

# g\_dest='/content/drive/MyDrive/Glaucoma Datasets/combined/glaucoma'

# os.makedirs(g\_dest)

# g\_list=[g\_path1,g\_path2,g\_path3,g\_path4,g\_path5]

# for i in g\_list:

#   shutil.copytree(i, g\_dest, dirs\_exist\_ok=True)

##################################################

#normal

# n\_path1='/content/drive/MyDrive/Glaucoma Datasets/Database/Database/normal'

# n\_path2='/content/drive/MyDrive/Glaucoma Datasets/RIM-ONE\_DL\_images/RIM-ONE\_DL\_images/partitioned\_randomly/training\_set/normal'

# n\_path3='/content/drive/MyDrive/Glaucoma Datasets/RIM-ONE\_DL\_images/RIM-ONE\_DL\_images/partitioned\_randomly/test\_set/normal'

# n\_path4='/content/drive/MyDrive/Glaucoma Datasets/archive/Training-20211018T055246Z-001/Training/Images/NORMAL'

# n\_path5='/content/drive/MyDrive/Glaucoma Datasets/archive/Test-20211018T060000Z-001/Test/Images/normal'

# n\_dest='/content/drive/MyDrive/Glaucoma Datasets/combined/normal'

# os.makedirs(n\_dest)

# n\_list=[n\_path1,n\_path2,n\_path3,n\_path4,n\_path5]

# for i in n\_list:

#   shutil.copytree(i,n\_dest, dirs\_exist\_ok=True)

# print(len(os.listdir(n\_dest)))

pip install pathlib

import pathlib

base\_dir = '/content/drive/MyDrive/PROJECT/Glaucoma Final/combined'

base\_dir = pathlib.Path(base\_dir)

glaucoma = [fn for fn in os.listdir(f'/content/drive/MyDrive/PROJECT/Glaucoma Final/combined/glaucoma/')]

normal = [fn for fn in os.listdir(f'/content/drive/MyDrive/PROJECT/Glaucoma Final/combined/normal')]

data=[glaucoma,normal]

dataset\_classes =['glaucoma','normal']

image\_count = len(list(base\_dir.glob('\*/\*.jpg')))+len(list(base\_dir.glob('\*/\*.png')))

print(f'Total images: {image\_count}')

print(f'Total number of classes: {len(dataset\_classes)}')

count = 0

data\_count = []

for x in dataset\_classes:

  print(f'Total {x} images: {len(data[count])}')

  data\_count.append(len(data[count]))

  count += 1

sns.set\_style('darkgrid')

sns.barplot(x=dataset\_classes, y=data\_count)

plt.show()

**Spliiting Ratio of Dataset 80:10:10 (Train:Test:Validation) -**

!pip install split-folders

import splitfolders #to split dataset

import pathlib

base\_ds = '/content/drive/MyDrive/PROJECT/Glaucoma Final/combined'

base\_ds = pathlib.Path(base\_ds)

img\_height=256

img\_width=256

batch\_size=32

splitfolders.ratio(base\_ds, output='images', seed=1321, ratio=(.8,.1,.1), group\_prefix=None)

**Data augmentation done using Image Data Generator**

from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rescale=1./255,

shear\_range = 0.15,

zoom\_range = 0.15,

horizontal\_flip = True)

train\_ds = datagen.flow\_from\_directory(

    'images/train',

    target\_size = (img\_height, img\_width),

    batch\_size = batch\_size,

    class\_mode='categorical',

    shuffle=False)

val\_ds = datagen.flow\_from\_directory(

    'images/val',

    target\_size = (img\_height, img\_width),

    batch\_size = batch\_size,

    class\_mode='categorical',

    shuffle=False)

test\_ds = datagen.flow\_from\_directory(

    'images/test',

    target\_size = (img\_height, img\_width),

    batch\_size = batch\_size,

    class\_mode='categorical',

    shuffle=False)

# len(train\_ds.classes)

**Important functions:**

def plot\_train\_history(history):

    plt.figure(figsize=(15,5))

    plt.subplot(1,2,1)

    plt.plot(history.history['accuracy'])

    plt.plot(history.history['val\_accuracy'])

    plt.title('Model accuracy')

    plt.ylabel('accuracy')

    plt.xlabel('epoch')

    plt.legend(['train', 'validation'], loc='upper left')

    plt.subplot(1,2,2)

    plt.plot(history.history['loss'])

    plt.plot(history.history['val\_loss'])

    plt.title('Model loss')

    plt.ylabel('loss')

    plt.xlabel('epoch')

    plt.legend(['train', 'validation'], loc='upper left')

    plt.show()

# plot\_train\_history(model\_info)

def glaucoma\_prediction(test\_image):

  image = img\_to\_array(test\_image)

  image = np.expand\_dims(image, axis = 0)

  result = np.argmax(model.predict(image))

  return result

**Here is the code of applying InceptionV3 by importing libraries in the Google Colab:**

from keras.applications import InceptionV3

from keras.models import Sequential

from keras.layers import GlobalAveragePooling2D, Dense, Dropout, BatchNormalization

from keras.callbacks import ModelCheckpoint

keras.backend.clear\_session()

# Load pre-trained InceptionV3 model

base\_model = InceptionV3(weights='imagenet', include\_top=False, input\_shape=(256, 256, 3))

# Add custom top layers

model = Sequential()

model.add(base\_model)

model.add(GlobalAveragePooling2D())

model.add(Dense(512, activation='relu'))

model.add(BatchNormalization())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(2, activation='softmax'))

# Set the first layers to non-trainable (optional)

for layer in model.layers[0].layers:

    layer.trainable = False

# Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Define a ModelCheckpoint callback

checkpoint = ModelCheckpoint('best\_weights.h5', save\_best\_only=True, save\_weights\_only=True, monitor='val\_loss', mode='min', verbose=1)

# Train the model with the callback

inception = model.fit(train\_ds, epochs=50,steps\_per\_epoch = int(round(1032/32)),

 validation\_data=val\_ds,validation\_steps = int(round(128/32)), callbacks=[checkpoint])

# Load the best weights

model.load\_weights('best\_weights.h5')

# Summary of the model

model.summary()

plot\_train\_history(inception)

# score=model.evaluate(test\_ds)

# print("Loss:",score[0],"Accuracy:",score[1])

pred= np.round(model.predict(test\_ds, verbose=1))

test\_labels=test\_ds.labels

test\_pred\_labels=[]

for i in range(len(pred)):

  test\_pred\_labels.append(np.argmax(pred[i]))

conf\_matrix= confusion\_matrix(test\_pred\_labels,test\_labels)

print (conf\_matrix)

sb.heatmap(conf\_matrix,cmap='Purples', annot=True,xticklabels=['glaucoma','normal'],yticklabels=['glaucoma','normal'],linewidths=1,

                linecolor='green').plot()

plt.show()

test\_report = classification\_report(test\_ds.labels,test\_pred\_labels, target\_names=['glaucoma','normal'], output\_dict=True)

test\_df = pd.DataFrame(test\_report).transpose()

test\_df

**Here is the code of applying MobileNetV2 by importing libraries in the Google Colab:**

checkpoint = tf.keras.callbacks.ModelCheckpoint('best\_weights\_resnet50.h5', save\_best\_only=True, save\_weights\_only=True, monitor='val\_accuracy', mode='max', verbose=1)

from keras.applications import MobileNetV2

from keras.models import Sequential

from keras.layers import GlobalAveragePooling2D, Dense, Dropout, BatchNormalization

keras.backend.clear\_session()

# Load pre-trained MobileNetV2 model

base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(256, 256, 3))

# Add custom top layers

model = Sequential()

model.add(base\_model)

model.add(GlobalAveragePooling2D())

model.add(Dense(512, activation='relu'))

model.add(BatchNormalization())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(2, activation='softmax'))

# Set the first layers to non-trainable (optional)

for layer in model.layers[0].layers:

    layer.trainable = False

# Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Summary of the model

model.summary()

mobilenet\_v2\_model\_info=model.fit(train\_ds,

steps\_per\_epoch = int(round(1032/32)),

epochs = 50,

validation\_data = val\_ds,

validation\_steps = int(round(128/32)),

                                  callbacks = [checkpoint])

score=model.evaluate(test\_ds)

print("Loss:",score[0],"Accuracy:",score[1])

plot\_train\_history(mobilenet\_v2\_model\_info)

pred= np.round(model.predict(test\_ds, verbose=1))

test\_labels=test\_ds.labels

test\_pred\_labels=[]

for i in range(len(pred)):

  test\_pred\_labels.append(np.argmax(pred[i]))

conf\_matrix= confusion\_matrix(test\_pred\_labels,test\_labels)

print (conf\_matrix)

sb.heatmap(conf\_matrix,cmap='Purples', annot=True,xticklabels=['glaucoma','normal'],yticklabels=['glaucoma','normal'],linewidths=1,

                linecolor='green').plot()

plt.show()

test\_report = classification\_report(test\_ds.labels,test\_pred\_labels, target\_names=['glaucoma','normal'])

print(test\_report)

**Here is the code of applying CNN by importing libraries in the Google Colab:**

from keras.models import Sequential

from keras.layers import Conv2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense,Dropout

from keras.layers import BatchNormalization

# Initialising the CNN

classifier = Sequential()

# Step 1 - Adding Convolution layer

classifier.add(Conv2D(32, (3, 3), input\_shape = (256,256, 3), activation = 'relu'))

# Step 2 - Adding MaxPooling layers

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Adding a second convolutional layer

classifier.add(Conv2D(32, (3, 3), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Step 3 - Flattening

classifier.add(Flatten())

# Step 4 - Full connection

classifier.add(Dense(units = 512, activation = 'relu'))

classifier.add(BatchNormalization()),

classifier.add(Dense(256,activation='relu')),

classifier.add(Dropout(0.25)),

classifier.add(Dense(units = 2, activation = 'softmax'))

# Compiling the CNN

classifier.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

classifier.summary()

model\_info=classifier.fit(train\_ds,

steps\_per\_epoch = int(round(1032/32)),

epochs = 150,

validation\_data = val\_ds,

validation\_steps = int(round(128/32)))

plot\_train\_history(model\_info)

classifier.save('/content/drive/MyDrive/Glaucoma Datasets/combined/combine\_cnn.h5')

model=load\_model('/content/drive/MyDrive/Glaucoma Datasets/combined/combine\_cnn.h5')

print("Glaucoma detection model loaded")

# test\_image = load\_img('/content/drive/MyDrive/datasets/acrima/Database/glaucoma/Im310\_g\_ACRIMA.jpg', target\_size = (256,256))

# prediction = glaucoma\_prediction(test\_image)

# if prediction == 0:

#  print("Glaucoma")

# else:

#  print("Not Glaucoma")

# test\_image = load\_img('/content/drive/MyDrive//datasets/acrima/Database/normal/Im001\_ACRIMA.jpg', target\_size = (256,256))

# prediction = glaucoma\_prediction(test\_image)

# if prediction == 0:

#  print("Glaucoma")

# else:

#  print("Not Glaucoma")

score=model.evaluate(test\_ds)

print("Loss:",score[0],"Accuracy:",score[1])

Testing set confusion matrix

from sklearn.metrics import classification\_report,confusion\_matrix

import seaborn as sb

pred= np.round(model.predict(test\_ds, verbose=1))

test\_labels=test\_ds.labels

test\_pred\_labels=[]

for i in range(len(pred)):

  test\_pred\_labels.append(np.argmax(pred[i]))

conf\_matrix= confusion\_matrix(test\_pred\_labels,test\_labels)

print (conf\_matrix)

sb.heatmap(conf\_matrix,cmap='Purples', annot=True,xticklabels=['glaucoma','normal'],yticklabels=['glaucoma','normal'],linewidths=1,

                linecolor='green').plot()

plt.show()

test\_report = classification\_report(test\_ds.labels,test\_pred\_labels, target\_names=['glaucoma','normal'], output\_dict=True)

test\_df = pd.DataFrame(test\_report).transpose()

test\_df

**Here is the code of applying ResNet50 by importing libraries in the Google Colab:**

checkpoint = tf.keras.callbacks.ModelCheckpoint('best\_weights\_resnet50.h5', save\_best\_only=True, save\_weights\_only=True, monitor='val\_accuracy', mode='max', verbose=1)

from keras.models import Model

from keras.layers import Input, Conv2D, MaxPooling2D, GlobalAveragePooling2D, Dense, BatchNormalization, Activation, Add

from keras.applications import ResNet50

keras.backend.clear\_session()

# Load pre-trained ResNet-50 model

base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(256, 256, 3))

# Add custom top layers

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(512, activation='relu')(x)

x = BatchNormalization()(x)

x = Dense(256, activation='relu')(x)

x = Dropout(0.25)(x)

predictions = Dense(2, activation='softmax')(x)

# Create the final model

resnet50\_model = Model(inputs=base\_model.input, outputs=predictions)

# Set the first layers to non-trainable (optional)

# Compile the model

resnet50\_model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Summary of the model

resnet50\_model.summary()

resnet50\_model\_info=resnet50\_model.fit(train\_ds,

steps\_per\_epoch = int(round(1032/32)),

epochs = 50,

validation\_data = val\_ds,

validation\_steps = int(round(128/32)),

                                       callbacks = [checkpoint])

plot\_train\_history(resnet50\_model\_info)

score=resnet50\_model.evaluate(test\_ds)

print("Loss:",score[0],"Accuracy:",score[1])

pred= np.round(resnet50\_model.predict(test\_ds, verbose=1))

test\_labels=test\_ds.labels

test\_pred\_labels=[]

for i in range(len(pred)):

  test\_pred\_labels.append(np.argmax(pred[i]))

conf\_matrix= confusion\_matrix(test\_pred\_labels,test\_labels)

print (conf\_matrix)

test\_report = classification\_report(test\_ds.labels,test\_pred\_labels, target\_names=['glaucoma','normal'])

print(test\_report)

sb.heatmap(conf\_matrix,cmap='Purples', annot=True,xticklabels=['glaucoma','normal'],yticklabels=['glaucoma','normal'],linewidths=1,

                linecolor='green').plot()

plt.show()

**Here is the code of applying VGG16 by importing libraries in the Google Colab:**

from keras.applications import VGG16

from keras.models import Sequential

from keras.layers import GlobalAveragePooling2D, Dense, Dropout, BatchNormalization

from keras.callbacks import ModelCheckpoint

keras.backend.clear\_session()

# Load pre-trained VGG16 model

base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(256, 256, 3))

# Add custom top layers

model = Sequential()

model.add(base\_model)

model.add(GlobalAveragePooling2D())

model.add(Dense(512, activation='relu'))

model.add(BatchNormalization())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(2, activation='softmax'))

# Set the first layers to non-trainable (optional)

for layer in model.layers[0].layers:

    layer.trainable = False

# Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Define a ModelCheckpoint callback

checkpoint = ModelCheckpoint('best\_weights.h5', save\_best\_only=True, save\_weights\_only=True, monitor='val\_loss', mode='min', verbose=1)

# vgg16\_model\_info=model.fit(train\_ds,

# steps\_per\_epoch = int(round(1032/32)),

# epochs = 50,

# validation\_data = val\_ds,

# validation\_steps = int(round(128/32)))

vgg16\_model\_info = model.fit(train\_ds, epochs=50,steps\_per\_epoch = int(round(1032/32)),

 validation\_data=val\_ds,validation\_steps = int(round(128/32)), callbacks=[checkpoint])

# Load the best weights

model.load\_weights('best\_weights.h5')

# Summary of the model

model.summary()

plot\_train\_history(vgg16\_model\_info)

pred= np.round(model.predict(test\_ds, verbose=1))

test\_labels=test\_ds.labels

test\_pred\_labels=[]

for i in range(len(pred)):

  test\_pred\_labels.append(np.argmax(pred[i]))

conf\_matrix= confusion\_matrix(test\_pred\_labels,test\_labels)

print (conf\_matrix)

sb.heatmap(conf\_matrix,cmap='Purples', annot=True,xticklabels=['glaucoma','normal'],yticklabels=['glaucoma','normal'],linewidths=1,

                linecolor='green').plot()

plt.show()

score=model.evaluate(test\_ds)

print("Loss:",score[0],"Accuracy:",score[1])

test\_report = classification\_report(test\_ds.labels,test\_pred\_labels, target\_names=['glaucoma','normal'])

print(test\_report)

**Here is the code of applying a custom CNN model by importing libraries in the Google Colab:**

checkpoint = tf.keras.callbacks.ModelCheckpoint('best\_weights.h5', save\_best\_only=True, save\_weights\_only=True, monitor='val\_loss', mode='min', verbose=1)

from keras.models import Sequential

from keras.layers import Conv2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense,Dropout

from keras.layers import BatchNormalization

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization

keras.backend.clear\_session()

# Initialize the CNN

classifier = Sequential()

# Step 1 - Add Convolution layer

classifier.add(Conv2D(64, (3, 3), input\_shape=(256, 256, 3), activation='relu'))

# Step 2 - Add MaxPooling layers

classifier.add(MaxPooling2D(pool\_size=(2, 2)))

# Adding a second convolutional layer

classifier.add(Conv2D(64, (3, 3), activation='relu'))

classifier.add(MaxPooling2D(pool\_size=(2, 2)))

# Adding a third convolutional layer for more complexity

classifier.add(Conv2D(128, (3, 3), activation='relu'))

classifier.add(MaxPooling2D(pool\_size=(2, 2)))

# Step 3 - Flatten

classifier.add(Flatten())

# Step 4 - Full connection

classifier.add(Dense(units=512, activation='relu'))

classifier.add(BatchNormalization())

classifier.add(Dropout(0.5))  # Increased dropout rate for regularization

classifier.add(Dense(units=256, activation='relu'))

classifier.add(Dropout(0.25))

# Output layer with 2 units for binary classification

classifier.add(Dense(units=2, activation='softmax'))

# Compile the CNN

classifier.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

classifier.summary()

model\_info=classifier.fit(train\_ds,

steps\_per\_epoch = int(round(1032/32)),

epochs = 130,

validation\_data = val\_ds,

validation\_steps = int(round(128/32)),

                          callbacks = [checkpoint]

)

plot\_train\_history(model\_info)

score=classifier.evaluate(test\_ds)

print("Loss:",score[0],"Accuracy:",score[1])

pred= np.round(classifier.predict(test\_ds, verbose=1))

test\_labels=test\_ds.labels

test\_pred\_labels=[]

for i in range(len(pred)):

  test\_pred\_labels.append(np.argmax(pred[i]))

conf\_matrix= confusion\_matrix(test\_pred\_labels,test\_labels)

print (conf\_matrix)

sb.heatmap(conf\_matrix,cmap='Purples', annot=True,xticklabels=['glaucoma','normal'],yticklabels=['glaucoma','normal'],linewidths=1,

                linecolor='green').plot()

plt.show()

test\_report = classification\_report(test\_ds.labels,test\_pred\_labels, target\_names=['glaucoma','normal'])

print(test\_report)

**Here is the code of web interface for testing glaucoma by using streamlit app:**

import streamlit as st

import tensorflow as tf

import cv2

import numpy as np

import pandas as pd

import streamlit\_authenticator as stauth

# Load the pre-trained Glaucoma detection model

# from keras.models import model\_from\_json

# json\_file = open('models\model (1).json')

# loaded\_model\_json = json\_file.read()

# loaded\_model = model\_from\_json(loaded\_model\_json)

# # load weights into new model

# loaded\_model.load\_weights(r"C:\Users\abdul\developement\Python-projects\streamlit-glaucoma-detection\glaucoma\_detector\models\ResNet50\_weights.h5")

# print("Loaded model from disk")

st.set\_page\_config(

        page\_title="Glaucoma Detector",

)

import yaml

from yaml.loader import SafeLoader

# Load the YAML file

with open('config.yaml') as file:

    config = yaml.load(file, Loader=SafeLoader)

authenticator = stauth.Authenticate(

    config['credentials'],

    config['cookie']['name'],

    config['cookie']['key'],

    config['cookie']['expiry\_days'],

    config['preauthorized']

)

model = tf.keras.models.load\_model('models/raw-test1.h5', compile = False)

# Define a function to make predictions on the input image

def predict\_glaucoma(image):

    # Preprocess the image (resize, normalize, etc. based on your model's requirements)

    processed\_image = preprocess\_image(image)

    # Make prediction

    # prediction = model.predict(np.expand\_dims(processed\_image, axis=0))[0]

    pred = model.predict(np.expand\_dims(processed\_image, axis=0))[0]

    return pred

# Define a function to preprocess the input image

def preprocess\_image(image):

    # Implement any necessary preprocessing (resize, normalization, etc.)

    # Example:

    processed\_image = cv2.resize(image, (224, 224))  # Resize to match model input size

    return processed\_image

# Streamlit app with enhanced styling

def main():

    # Add content for the authenticated page

    # Authenticate the user

    name, authentication\_status, username = authenticator.login('Login', 'main')

    if authentication\_status:

        st.title("Glaucoma Detector")

        st.markdown("\*\*Accuracies of Various Models :\*\*")

        data = {'Models': ['CNN', 'CNN 2','Resnet50', 'MobileNetv2','Vgg16','InceptionV3'],

            'Accuracy': [68.702,52.053,65.463, 68.103,58.779,72.591],

            'f1 score' : [70.021,47.021, 62.451, 71.087,52.661,72.471],

            'precision score' : [68.891,46.512, 63.209, 69.041,67.153,72.591],

            'recall score' : [68.802,45.021, 60.054, 70.023,58.775,72.553]}

        df = pd.DataFrame(data)

        df.set\_index('Models', inplace=True)

        df.index.name = 'Models'

        st.table(df)

        st.markdown("\*\*Confusion Matrix of best model:\*\*")

        st.image("models/download (4).png", caption='Confusion Matrix', use\_column\_width=True)

        # Add a header

        st.header("Upload an Image")

        uploaded\_file = st.file\_uploader("Choose an image...", type=["jpg", "png", "jpeg"])

        if uploaded\_file is not None:

            # Read the uploaded image

            # image = cv2.imdecode(np.fromstring(uploaded\_file.read(), np.uint8), 1)

            image = uploaded\_file.read()

            processed\_image = cv2.imdecode(np.fromstring(image, np.uint8), 1)

            # Display the uploaded image with caption

            st.image(image, caption='Uploaded Image.', use\_column\_width=True)

            # Make prediction

            prediction = predict\_glaucoma(processed\_image)

            # Display the prediction

            st.subheader("Prediction")

            # st.write(f"Probability: {prediction[0]:.2f}")

            if prediction >0.5:

                st.error("Glaucoma detected.")

            else:

                st.success("No Glaucoma detected.")

    elif authentication\_status == False:

        st.error("Authentication failed. Please try again.")

    elif authentication\_status == None:

        st.warning('Please enter your username and password')

if \_\_name\_\_ == '\_\_main\_\_':

    main()