DEVELOPMENT OF A WEB-BASED SYSTEM FOR DETECTING EYE DEFECTS USING DEEP LEARNING MODEL

BY

ENECHUKWU RANSOME CHUKWUBUIKEM

RUN/CMP/20/9564

A PROJECT PROPOSAL SUBMITTED TO
THE DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF NATURAL SCIENCES
REDEEMER'S UNIVERSITY, EDE, OSUN STATE, NIGERIA

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD
OF THE DEGREE OF BACHELOR OF SCIENCE (BSc.) IN COMPUTER SCIENCE.

AUGUST, 2023.

DECLARATION

I Enechukwu Ransome Chukwubuikem (RUN/CMP/20/9564), hereby declare that this project was carried out by me in the Department of Computer Science, Faculty of Natural Sciences, Redeemer's University, Ede, Osun State, Nigeria. To the best of my knowledge, this work has not been presented elsewhere for the award of a degree or any other purpose.

Enechukwu, Ransome Chukwubuikem

…… ……

Student *Signature & Date*

CERTIFICATION

We, the undersigned hereby certify that this project was carried out by Enechukwu, Ransome Chukwubuikem RUN/CMP/20/9564 in the Department of Computer Science, Faculty of Natural Sciences, Redeemer's University, Ede, Osun State, Nigeria. To the best of our knowledge, this work has not been presented elsewhere for the award of a degree or any other purpose.

Dr. O. O. Olaniyan

…….. ……

Supervisor *Signature & Date*

Dr. T. A. Olowookere

…….. ……

Co-Supervisor *Signature & Date*

Prof. A. O. Ogunde

…… ……

Head of Department *Signature & Date*

Prof. C. O. Akanbi

…… ……

External Examiner *Signature & Date*

DEDICATION

First and foremost, I would be dedicating this project to God Almighty, thanking him for his guidance and grace. To my parents, of blessed memories, who instilled in me the values of hard work, perseverance, and faith, I am forever grateful. Though they are no longer with me, their love and support continue to inspire me. I hope to honor their legacy through my work.

ACKNOWLEDGMENTS

I would like to express my genuine and deep gratitude to my sisters for their unwavering love and support, which has played a vital role in the success of my program at this esteemed institution. I extend my heartfelt appreciation to my supervisor, Mrs. T. O. Ojewunmi, for dedicating her time, effort, and guidance throughout this project, ensuring that its objectives were met.

I also want to acknowledge the collective efforts of both the academic and non-academic staff of the Department of Computer Science. Starting from the Academic Staff, Dr(Mrs) A. A. Kayode, Dr(Mrs) O. O. Olaniyan, Dr(Mrs) B. O. Oguntunde, Prof. A. O. Ogunde, Dr. M. O. Odim, Prof. S. A. Arekete, Dr. T. A. Olowookere, Mrs Adio, Mr Fagba, Mr Oyetade, Mr Adekunle, Mr Olorunfemi, Mr Victor, Mrs Onifade, Mr Agbolade, Dr. Abolarinwa. Their contributions have immensely enriched my experience during my stay, and I am truly grateful for their invaluable support. To each and every one of them, I extend my sincerest wishes for abundant blessings.

My sincerest thanks goes out to my friend, Daniel Enamudu a.k.a Danikoko, thank you so much brother. For all the late night google meets, and for all the spare time you sacrificed just to help out, i say a very big thank you.

To Adaimoabasi, thank you so much for your input, even with a very tight deadline, you were able to help. May God raise men and women to help you when you need them, Amen.

To Akinyode Ilerioluwa, thank you my friend for your assistance throughout the entirety of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This project offers a novel approach to automate the detection of eye issues through the combination of deep learning techniques and web-based technologies. Glaucoma, cataracts, and diabetic retinopathy are among the prevalent eye disorders that, if left untreated, can seriously impair vision. In this article, a convolutional neural network (CNN)-based deep learning model was described that accurately detects if there is an eye defect in the eye or not. CNNs are trained on a sizable dataset of eye images. The dataset, which was compiled from credible medical sources and contains images of both healthy and injured eyes, ensures the model's robustness and generalizability. Thorough testing and validation are performed on an independent dataset of eye scans to evaluate the system's functionality. The deep learning model's accuracy, sensitivity, specificity, and overall diagnostic abilities are rigorously scrutinised in order to produce correct results. The Inception-V3 model had an accuracy of 86%. This research aims to provide an efficient and cost-effective alternative for early identification of ocular problems in order to permit rapid intervention and maybe prevent long-term visual losses. The web-based platform's usability and accessibility raise the possibility that it will be widely adopted, making it an effective tool for enhancing eye healthcare services, especially in regions with little resources.

Keywords: Healthcare, Deep Learning, Convolutional Neural Networks, Eye Defects, Glaucoma, Cataracts, and Diabetic Retinopathy.

CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

India has 15 million blind people, and the sad reality is that 75% of these cases were treatable at one point in time. There are 10,000 patients for every doctor in India. Vision impairment can result from a variety of eye disorders, including cataracts, corneal ulcers, and trachoma, among others. According to studies, the primary causes of blindness in India are early-stage illnesses that are left untreated. Only if these eye illnesses are correctly recognised at an early stage can their progression be prevented. There are numerous clearly perceptible symptoms associated with these eye conditions. Analysing a variety of symptoms is necessary in order to correctly diagnose eye illnesses. We present a novel approach to provide an automated eye illness identification model utilising visually discernible symptoms, combining deep learning techniques like convolution neural network and digital image processing techniques like segmentation and morphology. Using the suggested procedure, four eye diseases—crossed eyes, bulging eyes, cataracts, uveitis, and conjunctivitis—are examined and grouped. The deep neural network model that has been proposed helps with the early diagnosis of eye problems. The model advises individuals to visit an ophthalmologist for screening if necessary (Journal, 2022).

1.2 Problem Statement

Despite the advancements in medical technology, early detection of eye defects remains a challenge. Traditional methods of eye defect detection are time-consuming and require a high level of expertise. Therefore, there is a need for a more efficient and accurate method of eye defect detection. Machine learning has shown potential in detecting defects in various fields, including manufacturing and computer vision. Thus, there is a need to

explore the use of machine learning in eye defect detection. The research aims to develop a machine learning model that can accurately detect eye defects from images. The model should be able to identify different types of eye defects, such as cataracts, glaucoma, and macular degeneration, and provide a high level of accuracy in detection. The model should also be able to handle different types of images, including fundus images, OCT images, and retinal images (Benbarrad *et al*., 2021). The research will contribute to the development of a more efficient and accurate method of eye defect detection, which can aid in early diagnosis and treatment of eye diseases (Han, 2022).

1.3 Aims and Objectives of the Study

The aim of the study is to develop a web-based system for detecting Eye Defects using a deep learning model.

The specified objectives to achieve this aim are to:

i. collect clinical eye defect image data
ii. design a deep learning model for detecting defects in the eye,
iii. implement the deep learning models in (ii), and
iv. evaluate the performance of the deep learning models by checking accuracy of diagnosis and treatment of visual impairments.
v. deploy the deep learning model in a web-based system.

1.4 Scope of study

The aim and objectives of this study have been clearly stated. In order to achieve them, it is essential to identify the relevant research areas. By taking this approach, chances of achieving the desired outcomes can be maximised. The proposed project will be limited to the detection of four (4) eye defects. They are;

i. Bulging Eyes

ii. Crossed Eyes

iii. Glaucoma

iv. Uveitis

1.5 Overview of Methodology

To achieve the aim and objectives of this project:

I. Data collection was from online datasets due to the time complexity of real-time data sources

II. CNN (Convolutional Neural Network) architecture based on ResNet and Inseption-V3 architecture develops the models developed for detecting skin diseases.

III. There was a comparative result of the two architecture models to check for the best result.

IV. Implementation was done using the Python programming language.

V. Evaluation was done using standard performance metrics, including accuracy, precision, recall and f1-score.

VI. system was developed using flask framework

1.6 Significance of the Study

Developing data models for eye defects is of paramount importance as;

i. the model will easily detect eye defects at their early stages,

ii. this system will help streamline the process of identification, making it faster and more

efficient than traditional methods.

iii. there will be a significant reduction in the number of people suffering from eye defects,

iv. the model can further achieve a low mortality rate.

1.7 Definition of Terms

Below are some important terms we will be using in this project:

Bulging Eye (Proptosis): Eye protrusion or eye protrusion is frequently associated with thyroid conditions such Graves' disease or hyperthyroidism.

Crossed Eyes: The eyes do not look in the same direction simultaneously when this condition is present.

Glaucoma: A serious and developing eye condition called glaucoma has the potential to permanently harm the optic nerve.

Uveitis: The middle layer of tissue that makes up the eye's wall (the uvea) is affected by the inflammatory disorder known as uveitis.

Convolutional Neural Network (CNN): It is an effective deep learning method that is being utilised more frequently for a range of computer vision tasks, including medical picture analysis. It enables health care workers to make better judgement calls and enhance patient care.

Deep learning: In this particular area of machine learning, complex features are automatically extracted from huge datasets using multi-layered neural networks. This cutting-edge technology has the power to completely alter how data is used and perceived. Its applications can be found in a variety of sectors, including healthcare, banking, and transportation.

1.8 Organization of the Project

I. Chapter One (Introduction) presents the background of study, statement of problem, the aim and the objectives of the project.

II. Chapter Two (Literature Review) provides a review of literature of the study and also, a review of existing algorithms used as benchmarks for this study.

III. Chapter Three (Methodology) provides in depth description and explanation of the algorithms to be used in this study.

IV. Chapter Four (Implementation of design) shows the implementation of the system, the process involved in developing the eye detection system and the software infrastructure used to achieve the set of objectives and documentation.

V. Chapter Five (Conclusion and Recommendations) comprises the concluding part of the project.

CHAPTER TWO

LITERATURE REVIEW

2.0 Introduction

This chapter discusses the literature review and a few related works. Further explanations on certain concepts are expatiated here.

2.1 The Human Eye

Visible light is reacted to by the human eye, a sensory organ that enables utilisation of visual information for a variety of functions such as seeing things, maintaining balance, and regulating circadian rhythm. Here are some key facts about the human eye (Wikipedia Contributors, 2019):

2.1.1 Structure of the Human Eye

Two eyes, one on each side of the face, are a feature of humans. The orbits are bony spaces in the skull where the eyes are located. The extraocular muscles, which regulate eye movements, number six. The whitish sclera, coloured iris, and pupil make up the front visible portion of the eye. On top of this lies a thin covering known as the conjunctiva. The anterior portion of the eye is another name for the front part (Wikipedia Contributors, 2019).

2.1.2 Functions of the Human Eye

The eye's various components all work together to aid with vision. Light first travels through the cornea, which is the transparent front layer of the eye. The cornea helps the eye focus by bending light into a dome shape. The pupil, an aperture within the eye, allows some of this light to enter. The coloured portion of the eye, the iris, regulates how much

light the pupil lets in. Next, light passes through the lens, which is the eye's transparent interior. To properly focus light on the retina, the lens and cornea work together. A layer of tissue at the back of the eye called the retina contains unique cells called photoreceptors, which convert light into electrical signals when it strikes the retina. The optic nerve carries these electrical signals from the retina to the brain. The visuals you see are then created by the brain from the signals. Tears are also necessary for the health of your eyes (National Eye Institute, 2022).

2.1.3 Anatomy of the Human Eye

The cornea, iris, pupil, lens, retina, optic nerve, and vitreous are just a few of the parts that make up an eye. You have centre vision thanks to the macula, a small, heightened-sensitive region of the retina. The quantity and kind of pigments in the iris determine eye colour (Boyd and Turbert, 2018).

2.1.4 Importance of the Human Eye

A healthy pair of eyes means clear vision, which plays a major role in day-to-day life and quality of life. Eye exams can help detect eye problems early, when they are most treatable (Patel, 2018).

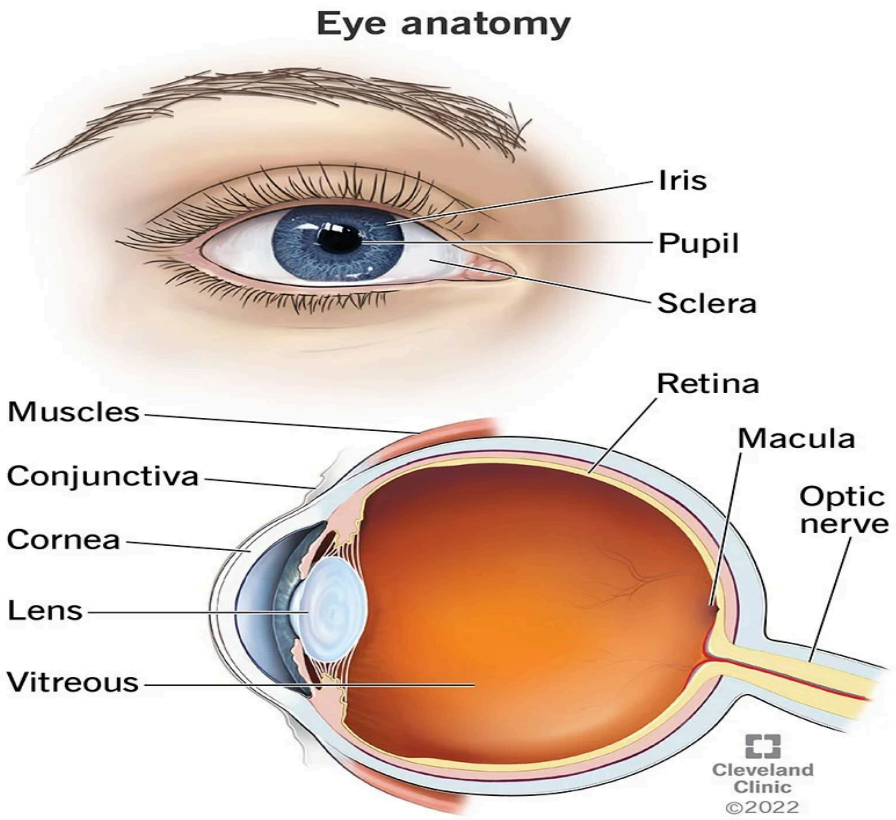

Figure 2.1: Descriptive Image of an Eye

2.2 Issues on Human Eye

There are several issues that can affect the human eye. Here are some of the most common ones:

2.2.1 Cataracts

This is a condition where the lens of the eye becomes cloudy, leading to blurry vision and difficulty seeing at night. It is most common in older adults, but can also be caused by injury or certain medications. Cataracts, thankfully, are a regular eye issue that may be repaired surgically. The extent of your eyesight loss and how it impacts the quality of life and capacity of functioning well will determine if surgery is required (Patel, 2018).

2.2.2 Bulging Eyes

Eyes that bulge or extend from their natural position may indicate a dangerous medical problem. The medical words for bulging eyes are proptosis and exophthalmos. Some people are born with protruding eyes, while others develop them as a result of an underlying medical problem. Without lifting your eyelid, the white component of your eye should not be visible above your iris (coloured area of the eye). If the white of your eye can be seen between your iris and upper eyelid, this might be an indication of irregular bulging. Your treatment approach will be determined by the underlying reason for the eye bulging. The unexpected bulging of just one eye is a medical emergency. Obtain medical assistance right away. It might be an indication of a major medical concern (Pietrangelo, 2019).

2.2.3 Crossed Eyes

Strabismus, or crossed eyes, is a disorder in which the eyes do not line up. Your eyes will stare in multiple directions if you're suffering from this ailment. And each eye will be drawn to a different item. The illness is more frequent in children, but it can happen at any age. Crossed eyes in adults as well as older kids can be as a result of a number of inherent medical disorders, such as cerebral palsy or stroke. Crossed eyes are typically treatable with corrective lenses, surgery, or a combination of the two (Chitra, 2019).

2.2.4 Refractive Errors

Refractive errors are a form of vision issue that makes seeing clearly difficult. They occur when the shape of your eye prevents light from properly concentrating on your retina (the light-sensitive layer of tissue at the back of your eye). The most common type of vision problem is refractive error. Over 150 million Americans have refractive error, yet many are

unaware that they might see well. That is why regular eye exams are so important. Your eye doctor can prescribe eyeglasses or contact lenses to help you see clearly if you have a refractive error. Eyeglasses, contact lenses, and surgery are the most prevalent means of treatment (National Eye Institute, 2019).

2.2.5 Glaucoma

Glaucoma is a category of eye disorders that cause optic nerve damage. The optic nerve transmits visual information from the eye to the brain and is essential for normal eyesight. High eye pressure is frequently associated with optic nerve damage. However, glaucoma can develop even with normal eye pressure. Glaucoma can strike at any age, although it is more frequent in elderly people. It's one of the primary causes of blindness in adults over 60. Many types of glaucoma have no symptoms. Because the effect is so gradual, you may not detect a change in vision until the problem has advanced. Regular eye checkups that involve measures of your eye pressure are essential. Vision loss can be delayed or avoided if glaucoma is detected early. You will require treatment or monitoring for the remainder of your life if you have glaucoma. Glaucoma is capable of being cured surgically, using lasers, or with eye drops (Mayo Clinic, 2022).

2.2.6 Uveitis

Uveitis is an inflammation of the eye. If your eye doctor tells you that you have it, you may be wondering how you acquired it. It is possibly caused by a different disease. When you see your eye doctor, they will most likely inquire about your medical history and any additional signs you are experiencing. They'll perform this to see whether another ailment has caused the eye problem (Bernstein, 2016).

2.2.7 Presbyopia

Presbyopia is the progressive loss of your eyes' capacity to see things well up close. It is a natural aspect of the ageing process. Indeed, the name "presbyopia" is derived from a Greek word that means "old eye." Presbyopia may become apparent after the age of 40. You will most likely find yourself holding reading items more distantly so as to view things well. Wearing corrective lenses, undergoing refractive surgery, or receiving lens implants are all treatment possibilities (Boyd, 2019).

2.3 Detection techniques for Eye defects

Below are some detection techniques for Eye defects:

2.3.1 Artificial Visual Processing Techniques

The main goal of visual-based approaches is to understand the world both in its natural and artificial representations. In the latter, the process to identify images is mostly an attempt to look for a mathematical/logical connection between the input images and representations of the environment. The mathematical/logical connection is a transition from the input image (s) to the model, which reduces the information contained in the image to relevant information for the application domain.

The hierarchy of image representation and the background functions/algorithms can be further simplified as low-level and high-level image processing (Czimmermann *et al.*, 2020).

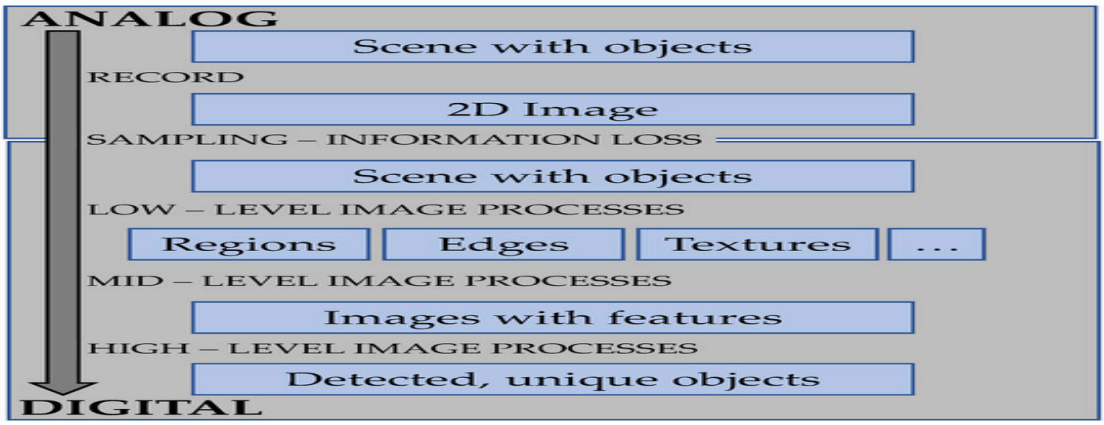

Figure 2.2: The theoretical levels of image representation for image analysis (Source: Czimmermann *et al.*, 2020)

2.3.2 Ophthalmic imaging

Ophthalmic imaging has seen explosive growth in recent years. Current retinal imaging techniques have contributed immensely to our appreciation of the pathophysiology and treatment of retinal disorders. In this section, two main ocular imaging modalities, fundus imaging and OCT imaging, were discussed. As the retinal image quality improves, the sensitivity and specificity of ocular malady detection and/or grading improves. The improved capabilities of digital technology to acquire, edit, archive and transmit retinal images and continued collaborations in this area are set to further improve retinal imaging for the benefit of patient management. Areas of continuous improvement with retinal imaging include portable, functional imaging, cost-effective fundus imaging, longer wavelength OCT imaging and adaptive optics (Muchuchuti and Viriri, 2023).

2.3.3 Deep Learning Models

Educational paradigms in their most elemental level Deep learning algorithms may be educated to distinguish between a wide variety of abnormalities if they are shown with a large number of photos depicting a variety of eye conditions. This is done by exposing the

algorithms to the images. This is achieved by presenting the algorithms with a significant number of different pictures. These models make it feasible to discover flaws in real time and may also be included into automated inspection procedures if that is something that is desired (Czimmermann *et al*., 2020).

2.3.4 Automated Visual Inspection

This method combines classic approaches to computer vision with methods that rely on human vision in order to make the process of locating errors in a wide variety of different spheres easier. This is done through a process known as "feature engineering," which comprises the establishment of a set of heuristic rules for the detection of objects in images. Feature engineering is the approach that is used to achieve this goal (Logan, 2017).

2.3.5 Visual Based Defect Detection

In order for this technology to perform successfully, the deployment of computer vision algorithms is required. This technology is used to detect and classify defects that are visible in digital photographs. It is able to detect irregularities in the cornea, such as scratches, cracks, and other imperfections on the surface of the eye. This ability allows it to diagnose a variety of conditions affecting the cornea. Because of these capabilities, it is capable of diagnosing a wide variety of corneal diseases (Ren *et al.*, 2021).

2.4 Artificial Intelligence

Artificial Intelligence (AI) is the science and engineering of creating intelligent devices, mainly computer programs. Machines may be able to learn from their mistakes, adapt to new inputs, and perform tasks that are similar to those performed by humans. Artificial intelligence, in its most basic form, is a field that combines computer science with large

datasets to solve problems. It also includes machine learning and deep learning as sub-fields (IBM Cloud, 2023).

It refers to the capability of machines to mimic or supplement human cognition through actions such as reasoning and learning from data. The term "artificial intelligence" (AI) is an abbreviation for the word "artificial intelligence." One definition of what is known as artificial intelligence (AI) is the possibility that robots could one day be able to duplicate or perhaps surpass human cognition (Abhishek, 2022).

Artificial intelligence is not a single discipline but rather a synthesis of many subfields. Some of the subfields that make up AI include machine learning, neural networks, robotics, and natural language processing. AI is not a single discipline but rather a synthesis of many subfields (Abhishek, 2022).

2.4.1 Types of Artificial Intelligence

Here are some types of Artificial Intelligence:

2.4.1.1 Machine Learning

Focuses on teaching computers to learn from data and then use that data to either make predictions or judgements (Accenture, 2021).

2.4.1.2 Neural Networks

Constructed with the purpose of mimicking the form and operation of the human brain. Image identification and the processing of natural languages are only two examples of the applications they have found (Accenture, 2021).

2.4.1.3 Expert Systems

Systems that are created to act in a manner that is analogous to that of a human being who is a specialist in a certain field (Accenture, 2021).

2.4.1.4 Robotics

Includes the application of artificial intelligence in the creation of robots that are capable of carrying out activities that are traditionally carried out by people, such as working on an assembly line or performing surgery (Accenture, 2021).

2.4.2 Applications of Artificial Intelligence

I. Applications of artificial intelligence may be found in many different fields, such as healthcare, finance, transportation, and manufacturing, among others (Kelley, 2022).

II. Artificial Intelligence is utilised for a variety of functions, including the identification of fraudulent activity, the assessment of credit applicants, customer support, and the optimisation of supply chains (Kelley, 2022).

III. Artificial Intelligence is also employed in activities such as navigation, object identification, and decision-making in robotics, drones, and autonomous vehicles (Kelley, 2022).

2.4.3 Challenges and Opportunities

I. The advent of Artificial Intelligence brings out both difficulties and prospects. Concerns relating to ethics, prejudice, and the loss of jobs are some examples of the difficulties (Burns, 2022).

II. Artificial Intelligence on the other hand, also opens the door to prospects for innovation, enhanced efficiency, and improved decision-making (Burns, 2022).

III. As artificial intelligence continues to advance, it is essential to explore the ethical implications of this technology and to make certain that it is applied in a way that is both responsible and useful (Burns, 2022).

2.5 Machine Learning

Machine learning is a subfield of artificial intelligence (AI) that focuses on the creation of algorithms and models that enable computers to learn and make predictions or judgements without being explicitly programmed to do so. This is in contrast to traditional AI, which is concerned with teaching computers how to do these things. To put it another way, machine learning is a method that enables computers to learn independently, without being specifically programmed (Advani, 2020).

In order to recognise patterns, correlations, and trends, the algorithms that power machine learning "learn" from previous data, which is sometimes referred to as "training data." The algorithms do an analysis on the data, extract characteristics, and construct models that are capable of making forecasts or choices based on data that has not yet been observed (Wikipedia Contributors, 2019).

2.5.1 Types of Machine Learning

2.5.1.1 Supervised learning

This is the most common type of machine learning. The algorithm is given a small portion of the total dataset to train with. The training dataset is similar to the whole dataset in characteristics but requires labelled parameters for the problem. The algorithm then has to discover a relationship between the input and output. The solution generated by the algorithm is then used on the remaining dataset, which it learns from in the same way as the training dataset. So, supervised learning improves itself on new data (Potentiaco, 2019).

2.5.1.2 Unsupervised learning

This algorithm works on unlabeled data. In supervised learning, the algorithm has the advantage of finding the exact nature of the relationship between any two datasets. However, in unsupervised learning, the algorithm deduces a connection between the datasets without input from the programmer. Therefore, it can quickly adapt to the dataset by changing its hidden structures dynamically (Potentiaco, 2019).

2.5.1.3 Semi supervised learning

For the aim of the training process, this approach makes use of both labelled and unlabeled data. It accomplishes this goal by directing the learning process in order to improve performance, which it achieves by using the very little number of labelled data that is currently accessible (Potentiaco, 2019).

2.5.1.4 Reinforced learning

Modelled on how human beings learn from data in their daily lives. The algorithm is made to improve itself by using a trial and error method to learn from new situations. Favorable results are reinforced or encouraged, while unfavourable results are rejected or punished (Potentiaco, 2019).

2.5.2 Machine Learning Techniques

2.5.2.1 Support Vector Machines (SVM)

Support vector machines (SVM) is a type of supervised learning that asks users to select the hyperplane that results in the highest accurate classification of data points. It is typically utilised in activities that are associated with classification (Mathworks, 2019).

2.5.2.2 Decision Trees

In the field of machine learning, decision trees are a specific kind of algorithm that entail the process of recursively separating data into smaller subsets based on the characteristics that are considered to be the most important. Both classification and regression problems can be solved with their help (Castañón and Mathworks, 2019).

2.5.2.3 Neural Networks

Neural networks are a special sort of algorithm that is used in machine learning; they take their name from this field's inspiration, which is the structure and activities of the human brain (Castañón, 2019).

2.5.2.4 Ensemble Methods

The efficiency of ensemble methods can be enhanced by combining the results of multiple models into a single comprehensive framework. For illustration purposes, the terms "bagging," "boosting," and "stacking" spring to mind (Laa, 2006).

2.5.2.5 Dimensionality Reduction

A strategy known as dimensionality reduction is a method that entails lessening the number of variables that are used in an analysis while keeping the data that is most significant. It can make complex data easier to understand and increase the performance of models (Laa, 2006).

2.5.2.6 Clustering

Clustering is a method of unsupervised learning that includes grouping data points that are similar together depending on how similar they are to each other or how far apart they are from one another (Burns, 2021).

2.5.2.7 Classification

Classification is a form of supervised learning that includes making an educated guess about the value of a category output variable based on one or more of the variables that were used to train the model (IBM, 2023).

2.5.2.8 Regression

Predicting a continuous output variable based on one or more input variables is the goal of the learning method known as regression, which falls under the category of supervised learning (Castillo, 2021).

2.6 Deep Learning

Deep learning is a subfield of machine learning and artificial intelligence that makes use of artificial neural networks, which are complex algorithms fashioned after the structure and function of the human brain. Deep learning is considered to be one of the most promising areas of research in the field of artificial intelligence and machine learning. Researchers working in the field of artificial intelligence were the ones who initially established the concept of deep learning (Ricochette, 2018).

Artificial neural networks are used in this specialised form of machine learning to carry out intricate calculations on enormous volumes of data. Deep learning models can learn complicated patterns and make precise predictions since they are trained on huge amounts of data and gain knowledge from experiences (Biswal, 2023).

It has made it possible for technologies like image recognition, voice control for consumer electronics, and autonomous cars to advance. There is no longer a need for manual feature extraction because they automatically extract pertinent characteristics from the data (Mathworks, 2019).

Compared to more conventional machine learning methods, deep learning has a number of benefits. It can gradually learn high-level features from data, obviating the need for human feature extraction and domain expertise. They can also resolve issues from beginning to end, negating the requirement for dissecting the issue down into smaller pieces (Mahapatra, 2019).

2.6.1 Deep Learning Architectures

2.6.1.1 Residual Network (ResNet 50)

A new variation of the ResNet architecture was presented in the year 2015, and was called ResNet-50. There are a total of 50 layers in this deep convolutional neural network (CNN). The utilisation of residual blocks is the primary contribution made by ResNet. These blocks enable the network to learn residual functions, which in turn makes it simpler to train very deep networks without having to deal with the issue of vanishing gradients. The ResNet-50 network is well-known for its outstanding performance in a variety of computer vision applications, which led to its widespread adoption. Information can travel straight from one layer to another, bypassing any intermediate levels, thanks to residual connections, which allow for this. This enables the training of very deep networks with increased precision (Mukherjee, 2022).

2.6.1.2 AlexNet

The deep convolutional neural network known as AlexNet was one of the first of its kind and was built in 2012 by Alex Krizhevsky and colleagues. In 2012, it was declared the victor of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The artificial neural network known as AlexNet has a total of eight layers, with five convolutional layers and three fully connected levels. It was the first time that ideas such as ReLU activation functions, dropout regularisation, and data augmentation were presented. The development of computer vision's AlexNet led to a big leap forward in terms of the performance of picture classification tasks, which was considered the "state of the art" (Calin, 2020).

2.6.1.3 Inception v3 (GoogLeNet-v3)

Christian Szegedy and his colleagues unveiled their Inception series of deep convolutional neural networks in 2015. One of those networks, Inception v3, is also referred to by its former name, GoogLeNet-v3. Its primary objective was to overcome the computational inefficiencies of earlier versions such as GoogLeNet (Inception v1). Inception v3 utilises a parallel combination of 1x1, 3x3, and 5x5 convolutions, which assists in the effective capture of multi-scale information. In addition to this, it employs the "bottleneck" layer strategy in order to lessen the amount of work that must be done computationally. Image categorization is one of the many computer vision tasks that Inception v3 excels in, and it also does well in other computer vision tasks (Culurciello, 2017).

2.6.1.4 VGG-19 (Visual Geometry Group - 19 layers)

The Visual Geometry Group at the University of Oxford created the deep convolutional neural network known as VGG-19 in 2014. It consists of 19 layers, 16 of which are convolutional and 3 of which are fully coupled to one another. The VGG-19 network is well-known for its ease of use and consistency, as it employs 3x3 convolutional filters

everywhere across the network. VGG-19 is extensively used as a baseline model in a variety of computer vision tasks due to the fact that it is simple to develop and has good performance. This is the case despite the fact that it includes a large number of parameters (Architectures, 2013).

2.6.1.5 GoogleNet (Inception v1)

Christian Szegedy and his colleagues debuted their deep convolutional neural network, GoogleNet, in 2014. It is also known by its previous name, Inception v1. It triumphed to take first place in the ILSVRC competition in 2014. The innovative architecture of GoogleNet, which makes use of something called "Inception modules," contributed to the network's rise to prominence. In order to successfully capture a variety of scale information, these modules employ the utilisation of numerous filter sizes within a single layer (1x1, 3x3, and 5x5). In comparison to past versions, the architecture was designed to be more accurate while also increasing the computing efficiency (Alzubaidi et al., 2021).

2.7 Related Works

Kim *et al.* (2022) proposed an automatic method using deep learning algorithms to segment the optic disc and cup and to estimate the key measures. The proposed methods involve three steps: the estimation of the key measures, the segmentation of the optic disc and cup from the ROI using the proposed Multiscale Average Pooling Net (MAPNet), and the detection of the position of the optic disc from a fundus image using a Region of Interest (ROI) using Mask R-CNN. According to the segmentation findings using 1099 fundus images, the optic disc has a Jaccard Index (JI) of 0.9381 and a Dice Coefficient

(DC) of 0.9679, whereas the cup has a JI of 0.822 and a DC of 0.8996. The usual errors are 0.0451 for the optical disc (CD), 0.0376 for the CD area, and 0.0376 for the rim to disc (RD) ratio, respectively. The usual radius ratio errors for a disc, cup, and rim are 0.2257 for a disc, 0.500 for a cup, and 0.2166 for a rim. The approach demonstrates the ability to operate inside the therapeutic route and performs well when estimating the key metrics when fully applied. Important criteria to look for in ocular fundus images include the cup to optic disc (CD) ratio, CD area ratio, neuroretinal rim to optic disc (RD) area ratio, and rim thickness.

Tang et al. (2020) proposed a machine learning-based algorithm used to estimate the physiological elongation of ocular axial length (AL) in myopic children. The most typical kind of myopia is axial myopia. This research comprised a total of 491 males (48.57%) and 520 females (51.43%). All individuals were 11.18 2.49 years old on average. The mean SER varied from 0 to 8.00 D and was 3.21 1.61 D. The AL varied from 21.77 to 29.84 mm, with an average of 24.95 0.99 mm. The findings indicate that two linear machine learning (ML) models and one SVM model performed better than the classic statistical regression model in terms of prediction ability. Since the ML technique utilises a linear regression, the correlations between AL and the other variables are linearly dependent. The SVM with a linear kernel function likewise produced rather excellent results. Higher-order models that are more intricate, such an SVM with quadratic and cubic kernel functions, cannot be used for our application and cannot produce satisfactory results. When the input variables are altered in our models, the AL predicted adjusts correspondingly. The error and 95% CI varied within incredibly small bands. The robust linear model produced results with a high degree of precision because there were no significant differences between the

means of ALpredicted (24.95 0.99 mm) and ALtrue (24.95 0.91 mm) for the entire sample ($t = 0.007$, $P = 0.994$), and no significant differences were found among the various subgroups (all $P > 0.05$).

Badah *et al.* (2022) reviewed that glaucoma is a serious eye disease that affects a lot of people around the world. They explained that deep learning architectures have been widely used in recent years for image recognition tasks aim to detect human eye infections of Glaucoma disease by first employing a variety of machine learning (ML) classifiers, including the Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Naive Bayes (NB), Multi-layer Perceptron (MLP), Decision Tree (DT), and Random Forest (RF), and then a Deep Learning (DL) model, such as Convolutional Neural Network (CNN) based on Resnet152 model. On the dataset for Ocular Disease Intelligent Recognition, the suggested technique is evaluated. The collected findings demonstrated that, in contrast to the other ML classifiers, the RF and MLP classifiers had the best accuracy of 77%. For the identical objective and dataset, the deep learning model (CNN) model: Resnet152) offers an even higher accuracy of 84%. Additionally, we see that our top-performing model outperforms several cutting-edge methods in terms of results.

Bian *et al*. (2020) proposed a model for segmenting the optic disc and optic cup using a cascade neural network technique. The model utilises anatomical knowledge to guide the segmentation process and achieves a segmentation result of 93% for the optic disc and 88% for the optic cup. However, one disadvantage of the model is that if the first stage optic disc segmentation gives incorrect information, it can affect the second stage as well. This approach is significant for diagnosing glaucoma, as accurately segmenting the optic

disc and optic cup is crucial for calculating the cup-to-disk ratio (CDR). Glaucoma is a disease that affects the optic nerve and can lead to vision loss if not detected and treated early. Segmenting the optic disc and optic cup helps in identifying abnormalities and diagnosing glaucoma. The use of deep learning techniques, specifically convolutional neural networks, has become prevalent in the field of medical image segmentation, including the segmentation of the optic disc and optic cup. These techniques have shown promising results in accurately identifying and segmenting these structures in retinal fundus images.

A survey of computer-aided diagnosis for eye illnesses was conducted by Zhang *et al*. (2014). Both physicians and researchers have given computer assisted diagnosis (CAD), which may automate the identification procedure for eye illnesses, a great deal of attention. It not only lessens the load on the professionals by offering an unbiased view with insightful information, but it also provides patients with early identification and simple access. They examined ocular CAD procedures for diverse kinds of data. They explored the databases and algorithms to find various eye disorders for each type of data. Their benefits and drawbacks are examined and evaluated. They looked at three different data sources that are often incorporated into CAD approaches today: clinical, genetic, and imaging data. The most current advancements in techniques for diagnosing and treating eye illnesses (including diabetic retinopathy, glaucoma, age-related macular degeneration, and pathological myopia) are thoroughly researched and summarised. The clinical value of fully autonomous CAD systems that are able to embed clinical information and integrate diverse data sources still show enormous promise for future breakthroughs, even though

CAD for ocular illnesses has demonstrated considerable development over the previous years.

By creating fundus photographs with four distinct shooting axes, Takahashi *et al*. (2017) taught the GoogleNet DCNN to analyse a single fundus shot intelligently. A better degree of accuracy in the diagnosis of diabetic retinopathy is seen in the research on synthetic fundus pictures. This shows that more diverse fundus pictures ought to be used to diagnose DR. With the ability to scan 80% of the fundus region, ultra-wide field scanning laser ophthalmoscopy has just been developed as a new technique. More sophisticated algorithms are required to support this ongoing trend in AI diagnostic research development. Retrospective analysis of 9,939 posterior pole images from 2,740 diabetic individuals was conducted. Four fields in each eye were photographed yearly at Jichi Medical University between May 2011 and June 2015 using nonmydriatic 45° field colour fundus photography. 95% of the images were trained using a manually adjusted Davis grading of three extra neighbouring images, while the remaining 5% were trained using a modified totally randomly initialised GoogLeNet deep learning neural network. Using actual prognoses, we assessed 4,709 of the 9,939 posterior pole fundus images. Additionally, the updated GoogleLeNet learnt 95% of the images. Prevalence and bias-adjusted Fleiss' kappa (PABAK) of AI staging of the remaining 5% of the photos were the primary outcome variables.

Liu *et al.* (2019) did a study and looked at a deep learning system (DLS) created to categorise glaucomatous optic neuropathy (GON) using the automated analysis of retinal fundus images. The Chinese Glaucoma Study Alliance, the Handan Eye investigation, and several internet databases provided the images utilised in this inquiry. The researchers

selected 241,032 images in total to be the training dataset. On June 9, 2009, the photographs were entered into the databases. On July 11, 2018, they were purchased. On December 15, 2018, the photographs became the focus of research. Through the analysis of several validation datasets, the generalizability of the DLS was assessed. These datasets made it possible to evaluate the DLS in a clinical setting with no exclusions. Additionally, fundus images from several sources that offered a range of picture quality were used to evaluate the DLS. Furthermore, a population-based research that appropriately reflected the distribution of glaucoma patients within the cohort was used to evaluate the DLS. Last but not least, a second dataset with a varied ethnic distribution was used for extra analysis. a digital education system (DLS) was put into place with the intention of transferring the learned information and tested models in order to successfully apply them to fundus pictures gathered from various sources. A prediction visualisation test was carried out to identify the precise fundus image locations that the DLS uses for diagnostic reasons in order to improve understanding of the decision-making process used by the DLS. A total of 269,601 fundus pictures from the initial pool of 274,413 fundus images obtained using the Computerised Grading System for Glaucoma Analysis (CGSA) were successfully graded for glaucomatous optic neuropathy (GON) after undergoing an initial assessment of image quality. A population of 68,013 patients was randomly selected for the sample size of 241,032 pictures (consisting of 29,865 verified GON instances [12.4%], 11,046 probable GON cases [4.6%], and 200,121 unlikely GON cases [83%]) for the purpose of training the GD-CNN model. The remaining 28,569 pictures from the CGSA dataset were used for the validation and assessment of the GD-CNN model. In primary local validation datasets, it was discovered that the GD-CNN model's area under the receiver operating

characteristic curve (AUC) was 0.996 (95% confidence interval [CI], 0.995-0.998). The model furthermore showed a sensitivity of 96.2% and a specificity of 97.7%. Pathologic or high myopia was shown to be the main reason for both false-negative and false-positive grading by GD-CNN and manual grading, with prevalence rates of 46.3% and 32.3% for GD-CNN and 44.2% and 34.0% for manual grading, respectively.

Using images of the retinal fundus, Varadarajan *et al*. (2018) suggested a review and developed a deep learning system to detect refractive error. They came to the conclusion that the attention maps constantly emphasised the fovea as a feature that was crucial for prediction by examining the areas of an image that were most useful for prediction. They assessed how retinal fundus imaging may be used to obtain novel information, like refractive error. The UK Biobank and the Age-Related Eye Disease work (AREDS) clinical trials' 45- and 30-degree field-of-view retinal fundus pictures, respectively, were used in this work. In the UK Biobank and AREDS, subjective refraction and autorefraction were used to assess refractive error. In order to forecast refractive error, they created a deep learning system.

Domen *et al*. (2019) presented a deep-learning-based computer vision system for detecting surface defects. The system utilises a CNN to classify images as either defective or non-defective and outperforms traditional machine-vision approaches. The proposed system has significant implications for industrial applications, as detecting defects early in the manufacturing process can save time and resources while ensuring product quality and safety. The authors compared their approach to a classical machine-vision approach and demonstrated that their system outperforms the traditional approach. The proposed system

is significant for industrial applications, as surface defects can lead to product failure and safety hazards. Detecting defects early in the manufacturing process can save time and resources while ensuring product quality and safety. They utilised a dataset of images of various surface defects, including scratches, dents, and cracks. They trained the CNN on this dataset and achieved high accuracy in detecting surface defects. Also, they compared their approach to a segmentation-based deep-learning approach and demonstrated that their classification-based approach outperforms the segmentation-based approach.

Table 2.1: Summary of Related Works

Author(s) (year)	Title	Results	Methods	Limitations

Zhang *et al.*, (2014)	A survey on computer aided diagnosis for ocular diseases	The clinical importance of fully automatic CAD systems which are able to embed clinical knowledge and integrate heterogeneous data sources still show great potential for future breakthrough.	Clinical, genetic and imaging datasets	Researchers have to have background knowledge on ocular diseases

Takahashi *et al*., (2017)	Applying artificial intelligence to disease staging: Deep learning for improved staging of diabetic retinopathy	The PABAK to modified Davis grading was 0.64 (accuracy, 81%; correct answer in 402 of 496 photographs). The PABAK to real prognosis grading was 0.37 (accuracy, 96%).	GoogleNet DCNN	Since the models were trained on a single sort of surface flaw, it was difficult to use them to identify flaws in various materials or surfaces. This made the technology less applicable to different industrial situations.

Varadarajan *et al.,* (2018)	Deep learning for predicting refractive error from retinal fundus images	By analysing attention maps to determine the parts of a photograph most relevant for prediction, they concluded that attention maps consistently highlighted the fovea as a feature that was important for prediction.	Deep learning algorithm	The accuracy was not clearly stated.

Liu *et al.,* (2019)	Development and Validation of a Deep Learning System to Detect Glaucomatous Optic Neuropathy Using Fundus Photographs	The AUC of the GD-CNN model in primary local validation data sets was 0.996 (95% CI, 0.995-0.998), with sensitivity of 96.2% and specificity of 97.7%.	Use of a deep learning system.	Deep learning models were sometimes referred to as "black boxes" since it was difficult to understand how they made decisions. This lack of interpretability prevented models from being used in specific applications, such as safety-critical systems, by making it difficult to comprehend why they generate particular predictions.

Domen *et al.*, (2019)	Deep-Learning-Based Computer Vision System for Surface-Defect Detection	Model selection and Model training	Metrics such as accuracy, precision, recall, or F1 score, along with a comparison with existing methods or benchmarks	Limited dataset used
Tang *et al.,* (2020)	Machine learning-based algorithm used to estimate the physiological elongation of ocular axial length (AL) in myopic children.	The results show that most Machine learning (ML) models had a predictive ability that surpassed that of the traditional statistical regression model.	Support Vector Machine model (SVM)	More complicated models, such as an SVM with quadratic and cubic kernel functions, cannot achieve good performance

Kim *et al.,* (2022)	Identifying those at risk of Glaucoma: A Deep learning approach for Optic Disc and Cup Segmentation and their Boundary Analysis	0.9381 Jaccard Index (JI), 0.9679 Dice Coefficient (DC), 0.8222 JI, 0.8996 DC for cup	Mask R-CNN, Multiscale Average Pooling Net (MAPNet).	The dataset used was too small
Bian *et al.,* (2022)	Model for segmenting the optic disc and optic cup using a cascade neural network technique.	Segmentation result of 93% for the optic disc and 88% for the optic cup	Optic disc and optic cups	The first stage optic disc segmentation gives incorrect information, it can affect the second stage as well.

Badah *et al*., (2022)	Automatic Eye Disease Detection Using Machine Learning and Deep Learning Models	RF and MLP classifiers achieved the highest accuracy of 77% in comparison to the other ML classifiers	Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naıve Bayes (NB), Multi-layer perceptron (MLP), Decision Tree (DT) and Random Forest (RF), Convolutional Neural Network (CNN) based on Resnet152 model	Too many models

Tulbure *et al.*, (2022)	A review on modern defect detection models using DCNNs – Deep convolutional neural networks	Literature review of modern defect detection models using DCNNs. Literature research, data extraction, data synthesis, analysis & evaluation	Comparisons of different DCNN models, their performance metrics, strengths, weaknesses, and potential areas for improvement	Publication bias, availability and quality of paper.

CHAPTER THREE

METHODOLOGY

3.0 Introduction

This chapter explains the methods that were implemented to develop this project. It also describes how the data is collected and the algorithms implemented during this project.

3.1 Data Collection

The dataset used in this research was collected online from Kaggle. It contains images grouped into 4 different categories as shown in the table consisting of 446 images all together.

The dataset will be organised into a folder Eye defect Detection Dataset 446 Images, also containing subfolders showing the 4 different categories as Bulging Eyes, Glaucoma, Crossed Eyes, Uveitis.

3.2 Existing System Analysis

GoogleNet is a classic deep learning model offered by Szegedy *et al*. (2020). A convolutional neural network with a depth of 22 layers is called GoogleNet. GoogleNet collects more features and enhances training outcomes in comparison to deeper networks to increase training performance. A CNN architecture designed by Sandler et al. called MobileNetv2 aims to work well on mobile devices. Unlike traditional CNN systems, MobileNetv2 is based on an inverted residual structure with residual connections between the bottleneck layers. Additionally, this structure appears to have an intermediary expansion layer with rather deep curves to filter the generated features as it is a nonlinear source. In the MobileNetv2 architecture, there is a first fully convolutional layer with 32 filters, followed by 19 more bottleneck layers. ShufeNet is a computationally affordable

convolutional neural network created specifically for mobile devices by Zhang et al. Pointwise group convolution and channel shufe are two innovative processes that must be used in this revolutionary architecture in order to significantly reduce computational costs while maintaining accuracy. Convolutional neural networks like DarkNet were developed with the intention of being simple and practical. This network is built using a variety of ideas, including Network in Network, Inception, and Batch Normalisation. DarkNet19 prioritises convolutional layers above fully linked layers in its topology. DarkNet19's structure is made up of five max pooling layers and 19 convolutional layers as a result.

3.3 Architecture of the System

The architecture is divided into three modules: pre-processing, training and testing. Figure 3.1 shows the architecture of the proposed system

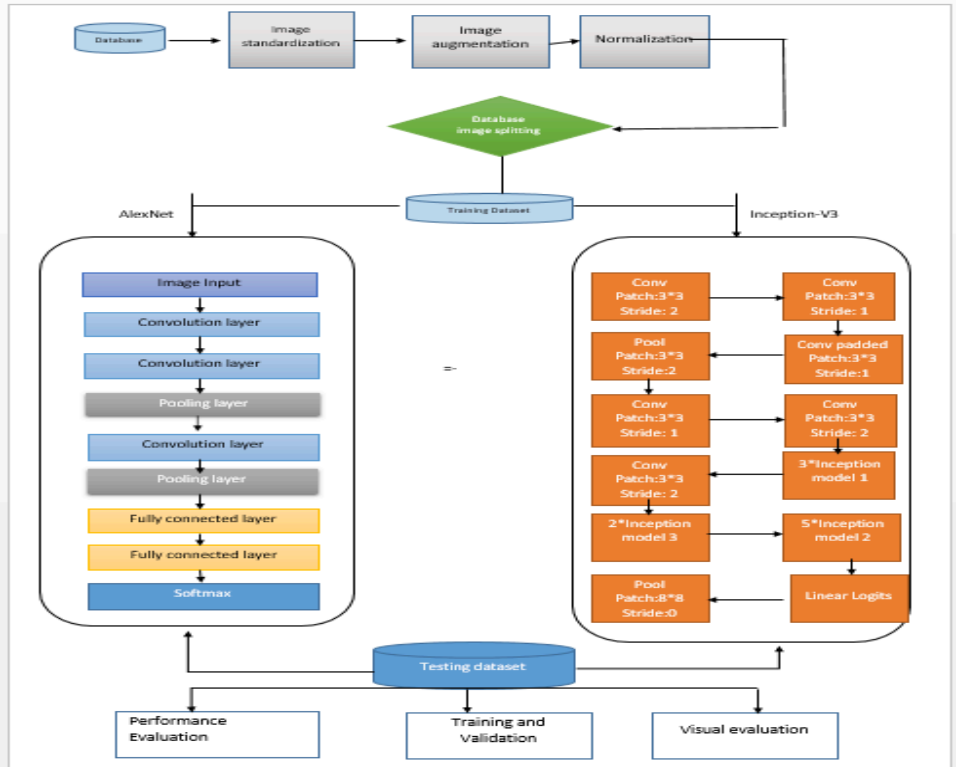

Figure 3.1: Architecture of System

3.3.1 Pre-Processing Module

The dataset is transformed to make it simpler for the algorithm to understand its features. It is frequently used to simplify a model while increasing accuracy. This procedure entails dataset partitioning, picture augmentation, image standardisation, and normalisation.

3.3.1.1 Image Standardisation

The process of standardising the photos entails dividing the individual pixel values by the normal deviation of the pixel values after deducting the mean pixel values from each one. As a result of this procedure, standardised photographs with a mean and standard deviation near to 0 and 1 are produced. Either per-image (sample-wise) or per-dataset (feature-wise) standardisation can be carried out (Brownlee, 2019).

3.3.1.2 Image Augmentation

Deep learning uses the process of "image augmentation" to extract new images from the ones it already has. To increase the diversity of the images and the models' capacity for generalisation, this phase entails making minor adjustments to the already-existing images (Lee, 2022).

3.3.1.3 Normalisation

Using a procedure known as data rescaling, image data pixels can be projected to a preset range, often (0,1) or (-1,1). Scaling the pixel values to make them more intelligible or recognisable is a technique called normalisation (Brownlee, 2019).

3.3.1.4 Splitting the Dataset

The dataset is next divided into two sections. It serves as both the testing set and the training set for the model, respectively.

3.3.2 Training Module

The deep learning model is the Convolutional Neural Network (CNN). For comparison purposes, CNN is implemented using the AlexNet and Inception-V3 architectures.

3.3.2.1 Convolutional Neural Network

The input layer, convolutional layer, pooling layer, and fully connected layers are only a few of the layers that make up CNN. The pooling layer samples the input image to reduce processing, the convolutional layer applies filters to the image to extract features, and the fully connected layer generates the final prediction. Through training, the network discovers the best filters. CNNs are utilised in tasks involving image recognition and processing, including computer vision, image and video recognition, classification, media recreation, recommendation systems, and natural language processing. They are utilised in image recognition systems since they were created primarily to process pixel data (Mishra, 2020). Figure 3.2 shows the different layers of a convolutional neural network.

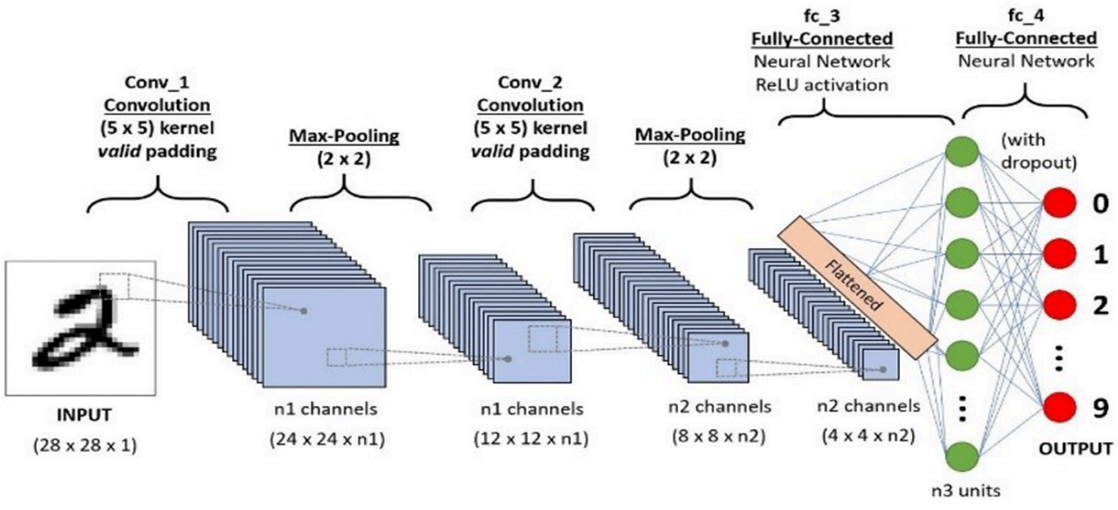

Figure 3.2: Convolutional Neural Network (Source: Saha, 2021).

CNNs are made to mimic the actions of the visual cortex and take use of the significant spatially local correlation found in real-world images. They are ideal for computer vision jobs and applications where object recognition is crucial, like self-driving cars and facial recognition, because they can detect simpler patterns first and more complex patterns later. They are taught by modifying the weights of the filters in each layer using a back propagation approach. In order to reduce the error between the anticipated output and the actual output, the network is fed with a large number of labelled images throughout the training procedure (Awati, 2022).

3.3.2.2 Inception-V3 Architecture

Through the application of transfer learning, Inception-v3, an expansion of the GoogLeNet architecture, has produced good classification results in a number of applications. Convolution kernels of various sizes are used by the Inception-v3 model to extract various degrees of feature images. As a result, the model is able to lower the size of the grid between various modules. Additionally, to maintain a consistent mesh size and split large-scale kernels into smaller-scale kernels, the inception module uses zero-padding convolution (Saha, 2018).

3.3.2.3 ResNet Architecture

ResNet-50 is a novel variant of the architecture. This deep convolutional neural network (CNN) has 50 layers altogether. The main contribution of ResNet is the use of leftover blocks. These blocks provide the network the ability to learn residual functions, which in turn makes training very deep networks less complicated because the problem of vanishing gradients is avoided. The ResNet-50 network was widely used because of its well-known exceptional performance in several computer vision applications. Bypassing any

intermediary levels, information can transfer directly from one layer to another owing to residual connections. This makes it possible to more precisely train very deep neural networks (Varangaonkar, 2018).

Mathematically, the softmax function is written as:

$$\sigma(z)\vec{}_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (3.1)$$

3.4 Software Development Life Cycle of the System.

The Waterfall approach was chosen for this project because the criteria are clearly defined, explicit, and fixed. The waterfall model is a method for sequential software development that is well-liked in software engineering and product development. With distinct endpoints or goals for each stage, the software development lifecycle (SDLC) is tackled in a linear, sequential manner. In order to effectively generate a product, the model moves through a number of stages that must be completed (Lewis,2019). .The SDLC stages are shown in Figure 3.3 below.

3.4.1 Stages of Waterfall Software Development Life Cycle

The phases involved are;

1. Requirement analysis
2. System Design
3. Implementation
4. Testing
5. Deployment
6. Maintenance

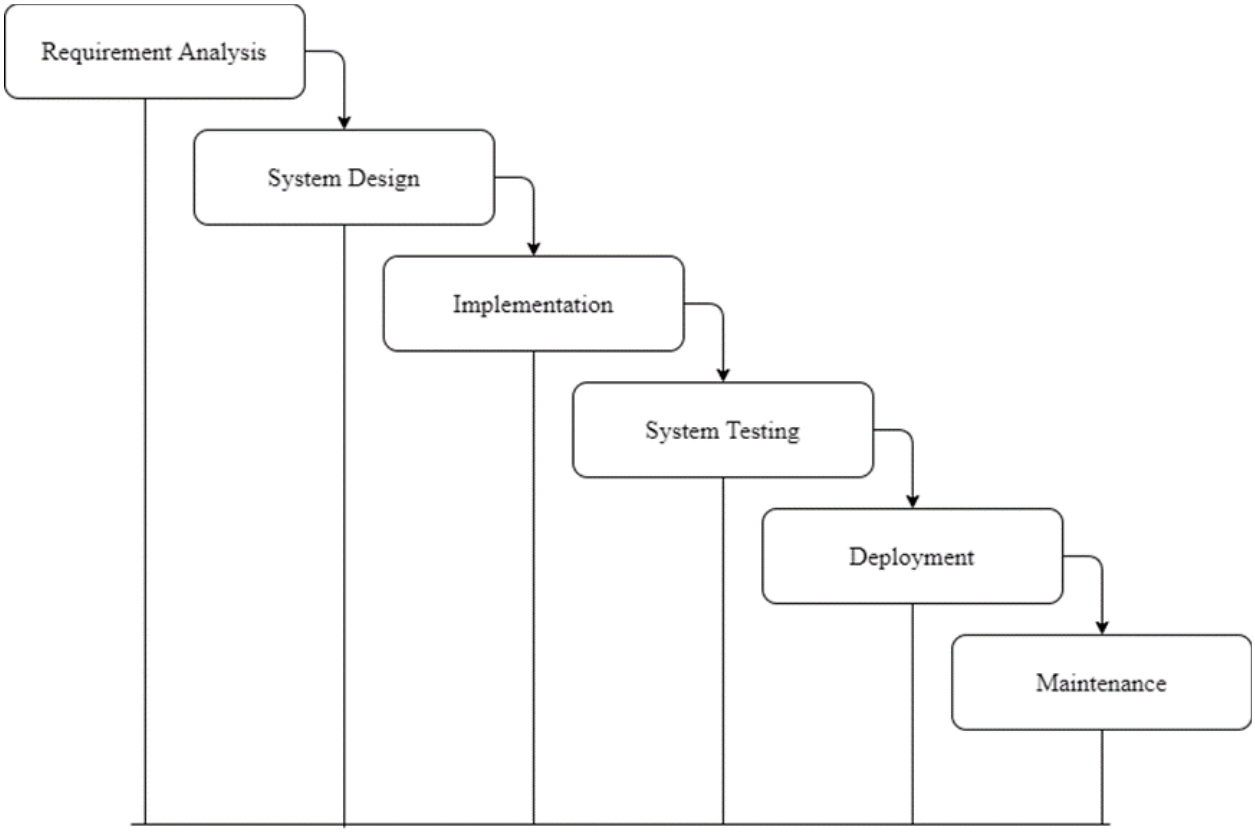

Figure 3.3: Diagrammatic representation of the waterfall model (Source: Lalband & Kavitha, 2019)

3.4.2 Flowchart of the Model

A flowchart displays the control flow from start to end, illustrating the numerous decision paths that are present while the activity is being performed. It is depicted in figure 3.4.

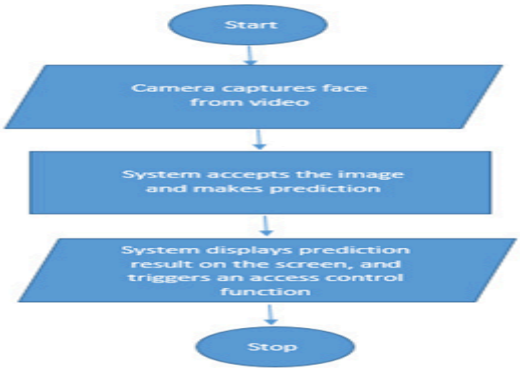

Figure 3.4: Flowchart of the system.

3.4.3 Use Case Diagram

The design of the proposed system is depicted using the Use case diagram in Figure 3.5 below.

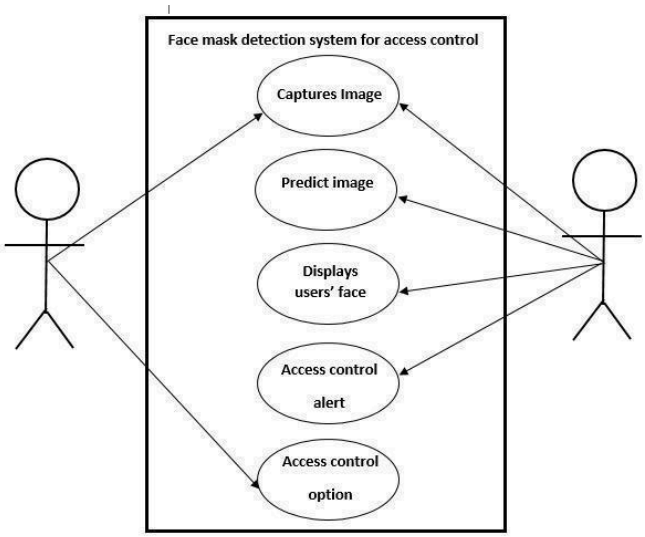

Figure 3.5: Use Case Diagram of the system.

3.5 Performance Metrics

They are used to know how well the deep learning model operated on the input data. Using it helps the programmer tune the parameters of the dataset to yield better performance from the model. The metrics for this research are accuracy, precision, recall and fl-score.

3.5.1 Confusion Matrix

The confusion matrix would provide the essential performance indicators. The confusion matrix is a two-by-two matrix table structure that allows the user to see how well an algorithm performs. It depicts various combinations of Actual Vs Predicted values. To define them, there are four (4) possible combinations:

- **True Positive (TP)** refers to the values that were both positive and projected

to be positive

- **False Positive (FP)** represents the values that were genuinely negative but were incorrectly expected to be positive.
- **False Negative (FN)** are the positive readings incorrectly forecasted as unfavourable.
- **True Negative (TN)** are the values that were both genuinely negative and projected to be negative.

Therefore, the confusion matrix can be used to deduce some measures for evaluating a system's performance. Some of which are:

- **Accuracy** is the number of correct predictions divided by the total number of guesses in the dataset

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+FP+TN+FN)} \quad (3.2)$$

- **Precision** refers to the number of true positives in the positive class prediction. The genuine positive is simply divided by the outcomes.

$$\text{Precision} = \frac{TP}{(TP+FP)} \quad (3.3)$$

- **Recall** is the number of positive class predictions from the dataset's positive samples. It is just true positive divided by expected outcomes.

$$\text{Recall} = \frac{TP}{(TP+FN)} \quad (3.4)$$

- **The F1 score** is a statistical metric for evaluating performance. An F1 score (from 0 to 9), with 0 being the lowest and nine being the greatest) is the average of an individual's ability based on precision and recall

$$\text{F1 Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3.5)$$

CHAPTER FOUR

SYSTEM IMPLEMENTATION, RESULTS AND DISCUSSIONS

4.0 Introduction

This chapter presents the implementation procedure, the results of the study, metrics that define the success of our models, and evaluation of the models, and the deployment of the system.

4.1 Implementation Procedure

This section focuses on the procedures and outcomes of a series of experiments conducted using Eye disease images downloaded from the Kaggle website.

```
In [1]: import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

        # Input data files are available in the read-only "../input/" directory
        # For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

        import os
        for dirname, _, filenames in os.walk('./Eye_disease'):
            for filename in filenames:
                print(os.path.join(dirname, filename))
```

Figure 4.1: Import Dataset

Four groups of 446 sets of eye illness photographs (in JPEG format)—bulging_eyes, crossed_eyes, glaucoma, and uveitis—are accessible. Each image category (Bulging_Eyes, Crossed_Eye, Glaucoma, Uveitis) has its own subfolder within the two folders designated as "train" and "test," respectively, for the dataset. The first step in the implementation procedure was to load the data and the shape, as shown in Figure 4.1. The data was loaded via the pandas' library in Python, as shown in Figures 4.2 and 4.3.

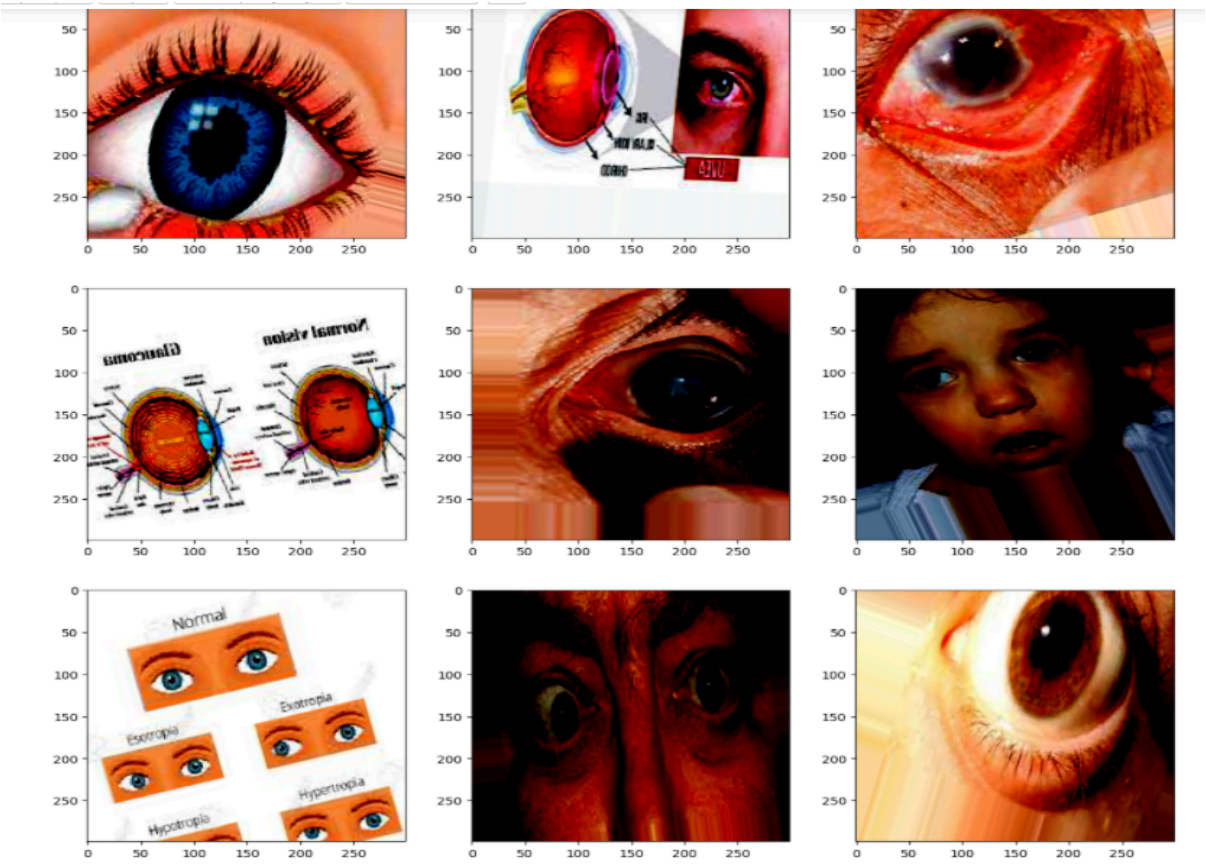

Figure 4.2: Output showing the Eye_disease dataset

All the needed library was imported into the environment, as shown in Figure 4.3. The needed libraries are matplotlib, seaborn, keras, sklearn, cv2, and os. The next steps in the implementation procedures were the data pre-processing.

```
In [1]: import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

        # Input data files are available in the read-only "../input/" directory
        # For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

        import os
        for dirname, _, filenames in os.walk('./Eye_disease'):
            for filename in filenames:
                print(os.path.join(dirname, filename))
```

```
In [2]: # General Libs
        from tensorflow import keras
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        from tensorflow.keras.preprocessing import image
        #from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
        from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2, preprocess_input
        from tensorflow.keras.layers import Dense, Flatten
        from tensorflow.keras.models import Model
        from tensorflow.keras.optimizers import Adam
        import numpy as np
        import random
        import matplotlib.pyplot as plt
        %matplotlib inline
```

Figure 4.3: Imported library

4.2 Data Pre-processing

In Figure 4.4, the data distribution shows that there are more images with Glaucoma than Bulging_Eyes, Crossed_Eye and Uveitis. The dataset was divided into training and testing in 80:20 split.

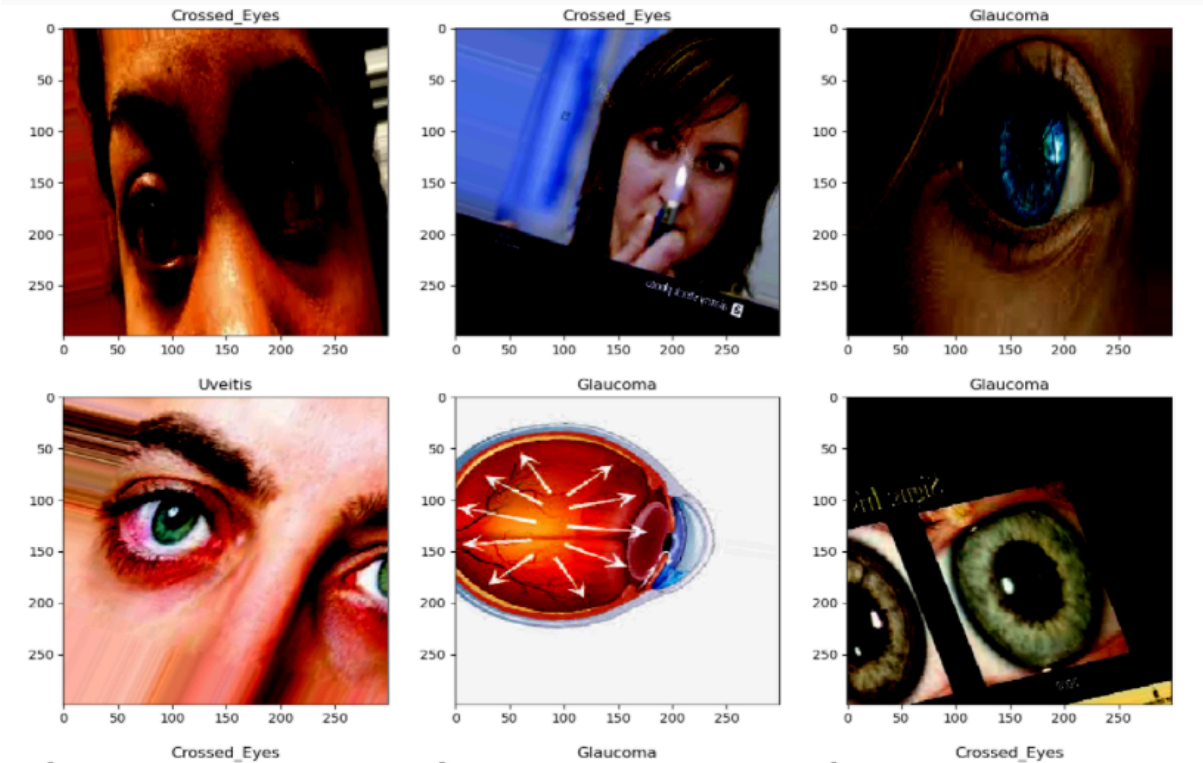

Figure 4.4: Output showing the image dataset

4.2.1 Image Standardisation

The regular image was cropped by 9% to eliminate the white borders overlay text in order to produce the data. White patches were employed to conceal the eye's personal information.

4.2.2 Data Augmentation

In order to meet the large data requirements of sophisticated machine learning architectures and avoid the model from becoming overly specialised, the project adopted the approach of data augmentation through rotation. As a result, the model was able to acquire a sizable number of parameters required for feature extraction, training, and classification of ocular pictures. Some training photographs were randomly transformed in order to accomplish data augmentation. The photographs were also magnified by a random factor of 20%, rotated by a random angle of 30 degrees, moved horizontally by a random distance equal to 10% of their width, moved vertically by a random distance equal to 10% of their height, and turned horizontally. Figure 4.5 displays the data augmentation code.

```
#Using keras ImageGenerator and flow_from_directory
#data_generator = ImageDataGenerator(preprocessing_function=preprocess_input, validation_split=0.2)
# With augmentation
data_generator = ImageDataGenerator(
        validation_split=0.2,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        preprocessing_function=preprocess_input,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
val_data_generator = ImageDataGenerator(preprocessing_function=preprocess_input,validation_split=0.2)
```

Figure 4.5: Data Augmentation

4.2.3 Normalisation

Figure 4.5 shows how each eye image was defined in a window size of 5 x 5; this eliminates noise from the dataset. A gradient operator is comparable to an averaging operator, which is frequently used to eliminate noise. To sort the medians of the three consecutive windows in ascending order, we first calculate the median for each subwindow. The centre pixel and its neighbours in each window were changed to the median values from the three windows that came after. The previous windows' values were used to calculate these values. To keep the clarity of the image, this procedure is repeated for every pixel. In order to reduce the quantity of grey tones in the input photographs, the watershed process was used. To lessen the impact of the varied lighting conditions, we do a greyscale normalisation.

This can be seen in Figure 4.6 and Figure 4.7. Furthermore, on [0...1] data, the CNN converges faster than on [0...255].

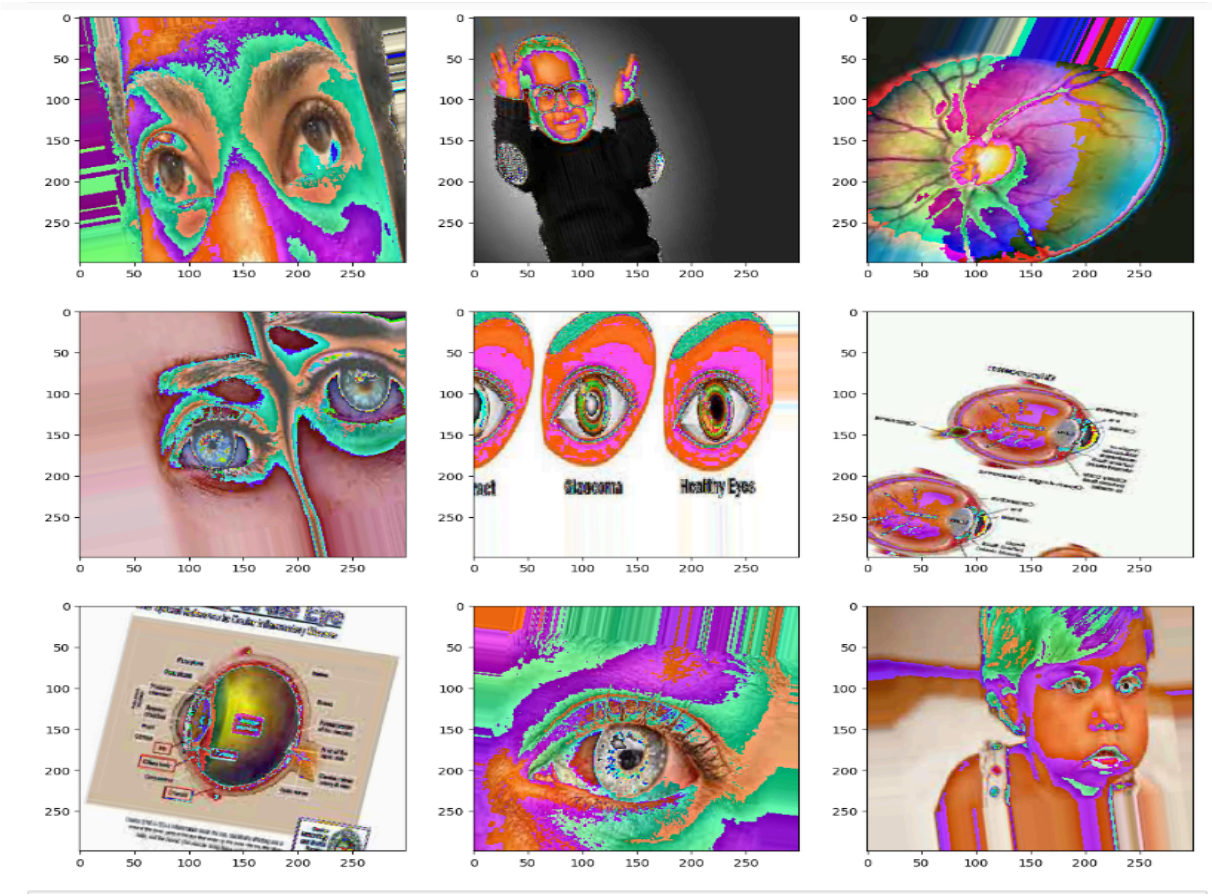

Figure 4.6: Batch train generator Image

```
# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test = np.array(x_test) / 255
```

Figure 4.7: Data Normalisation

4.3 Splitting Train and Validation Set

A training set and a validation set were created from the data set. Our CNN models will be trained on the training set utilising batch size and epochs, and then their accuracy will be assessed on the validation set. Precision, recall, f1-score, and accuracy are taken into

account when comparing different models. We divided the 446 photos in our dataset for different eye illnesses into two subsets: 80% for training and 20% for validation. On a portion of these photos, we evaluated how accurate the model was.

4.4 Experimentation with Inception Resnet Architecture for two-class prediction

The Inception Resnet version 2 architectures were used in this research. The rest net employed 64 feature kernel filters for the first and second convolutional layers, each having a filter size of 33. The input picture is given to the first and second convolution layers with the dimensions modified to 224x224x64. The output of this stage is passed on to the max pooling layer with a stride of 2. In the third and fourth convolutional layers, there are 124 feature kernel filters with a 3x3 filter size. A max pooling layer with stride 2 follows these layers and reduces the output to 56x56x128. The fifth, sixth, and seventh layers are convolutional layers with 3x3 kernel sizes. Each of them makes use of 256 feature maps. A stride 2 max pooling layer is put after the first three layers. The eighth through thirteenth layers are composed of sets of 3x3 kernel and 512 kernel convolutional layers. Following these layers is a max pooling layer with a stride of 1. A cell, an input gate, an output gate, and a forget gate make up the CNN unit.

4.5 Classification of CNN Architecture for Binary Prediction

Bulging_Eyes, Crossed Eye, Glaucoma, and Uveitis photographs of the eyes were used for training and testing. A total of 356 (80%) photos with extracted features were supplied into the CNN Architecture's input.

4.5.1 Training the Model

The Adam optimizer is used to construct the model. The remaining 20% of our training dataset is utilised for validation, with the remaining 80% being used for training. There are

356 photos in our training dataset and 140 photos in the validation sets. 50 epochs were used to train the model, and 50 batches were used to train our classifier.

4.6 Performance Evaluation

For this model, many kinds of illnesses, including cataract, uveitis/conjunctivitis, crossed eyes, and bulging eyes, have been compared. Two different models are used for the training for images with one eye and two eyes. One model predicts diseases namely crossed eye and bulging eye using two-eye images as in Figure 4.8 and Figure 4.9. The other model predicts diseases like conjunctivitis/uveitis using single- eye images as in Figure 4.10 and Figure 4.11.

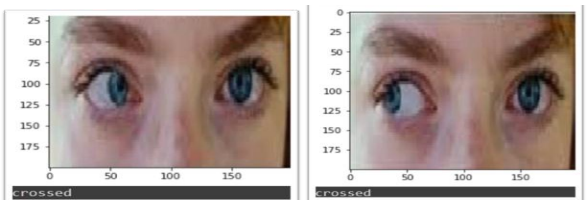

Figure 4.8: Model Prediction of Crossed eye

Table 4.1: Model Accuracy

Eye Configuration	Accuracy (%)	Epochs
Single Eye	86.00	16 early stops
Two Eyes	92.31	

As mentioned in the Table 4.1, we have obtained an accuracy of 96% for the dataset of single- eye images and accuracy of 92.31% for the dataset of two-eyed images.

Figure 4.9: Model Prediction of Bulging eye

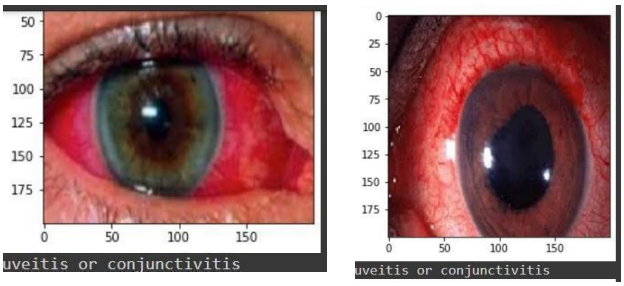

Figure 4.10: Model Prediction of Uveitis or Conjunctivitis

4.7 Model Performance Evaluation for CNN (Inception_net)

The training accuracy scores, classification report, and confusion matrix were used to evaluate the CNN model's performance. We also calculated the true positive rate (recall), precision, false-positive rate, false-negative rate (miss rate), and log loss using the confusion matrix.

4.7.1 Accuracy

When fed with the training set, the model was able to obtain an 86% accuracy rate. Table 4.2 shows the accuracy of the CNN model.

Table 4.2: Accuracy of the CNN Model

MODEL	ACCURACY (%)
CNN	86

4.7.2 Precision

Table 4.3 shows the precision values of the model when classifying the eye defects. The table 4.3 shows us that the model had a precision value of 0.60 when classifying Bulging Eyes, a precision value of 0.96 when classifying Crossed Eyes, a value of 0.77 when classifying Glaucoma and a value of 0.93 when classifying Uveitis.

Table 4.3: Precision of the Model

MODEL	PRECISION			
	BULGING EYES	CROSSED EYES	GLAUCOMA	UVEITIS
CNN	0.60	0.96	0.77	0.93

4.7.3 Recall

Table 4.4 shows the recall values of the model when classifying the eye defects. The table 4.4 shows us that the model had a recall value of 0.90 when classifying Bulging Eyes, a recall value of 0.87 when classifying Crossed Eyes, a value of 1.00 when classifying Glaucoma and a value of 0.70 when classifying Uveitis.

Table 4.4: Recall of the Model

MODEL	RECALL			
	BULGING EYES	CROSSED EYES	GLAUCOMA	UVEITIS
CNN	0.90	0.87	1.00	0.70

4.7.4 F1 Score

Table 4.5 shows the f1 score of the model when classifying the eye defects. The table 4.5 shows us that the model had a score of 0.72 when classifying Bulging Eyes, a score of 0.91 when classifying Crossed Eyes, a score of 0.87 when classifying Glaucoma and a score of 0.80 when classifying Uveitis.

Table 4.5: F1 Score of the Model

MODEL	F1 SCORE			
	BULGING EYES	CROSSED EYES	GLAUCOMA	UVEITIS
CNN	0.72	0.91	0.87	0.80

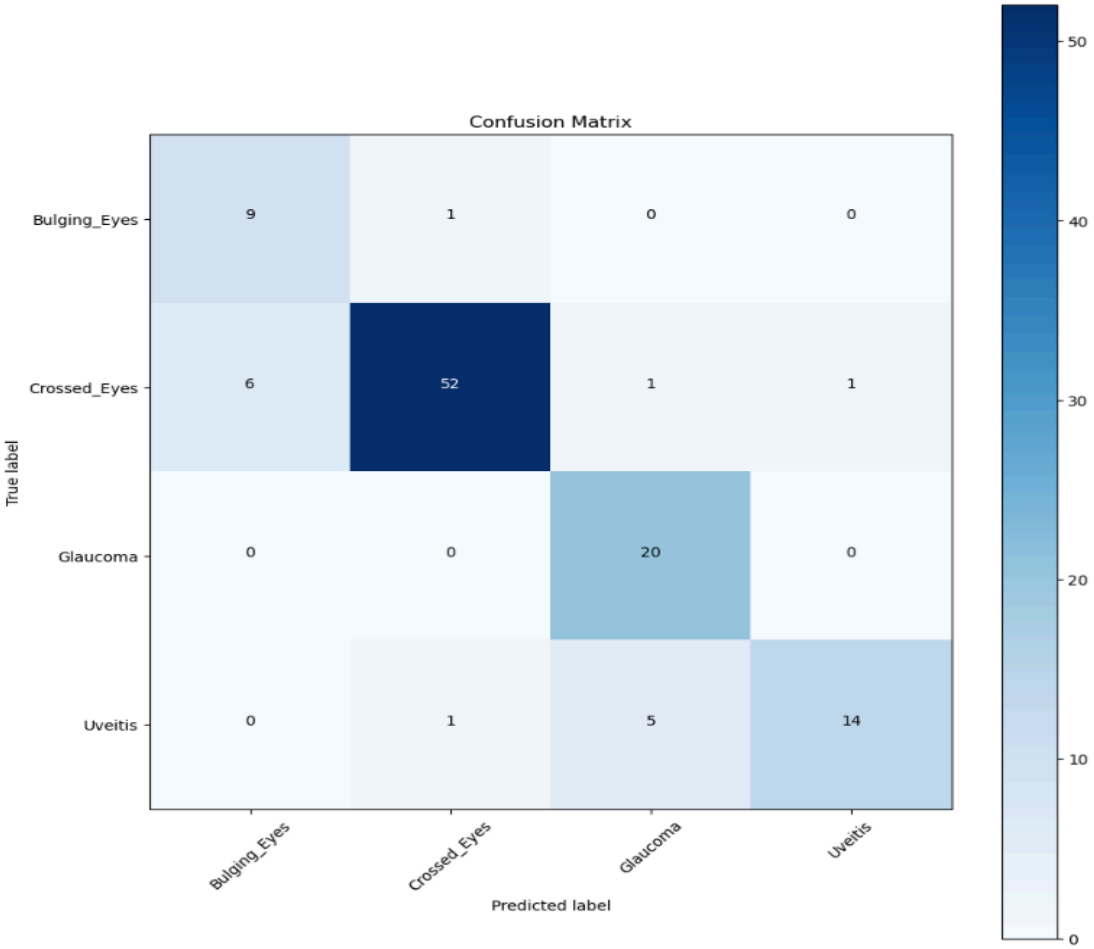

Figure 4.11: Confusion Matrix

In addition, the confusion matrix is shown in Figure 4.12. The confusion matrix in Figure 4.13 gives us additional information about the performance of the model. True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN). Figure 4.14 showed the model plot of 86 % training and testing accuracy, as well as training and testing loss. The model plot of training loss and testing loss was shown in Figure 4.15.

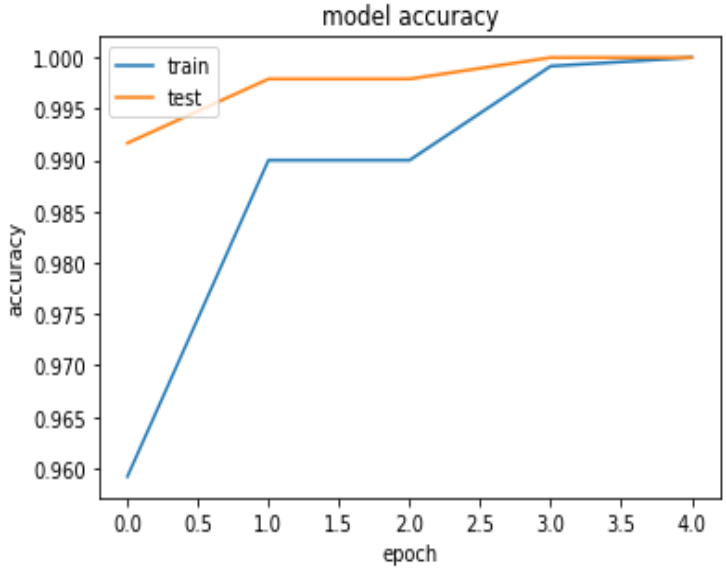

Figure 4.12: Training and Testing accuracy plot

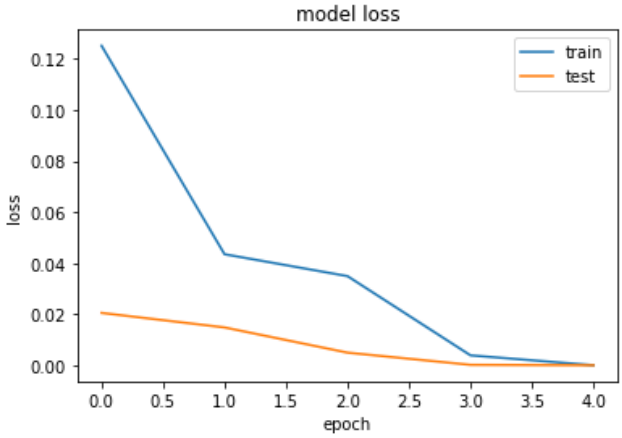

Figure 4.13: Training and Testing loss plot

Figure 4.15 demonstrates the training and validation accuracy and loss of InceptionV3 architecture where training curves are depicted using the blue line and validation curves are depicted using the red line. In case of training accuracy, starting from a value of 0.96, the accuracy reaches up to 0.99 then starts drooping in the last 23 epoch. Again, in case of validation accuracy, starting from a value of 0.971, the accuracy reaches up to 0.987 then drops to 0.97 in the last epoch

Figure 4.14: Training and Testing Accuracy plot for ResNet Model

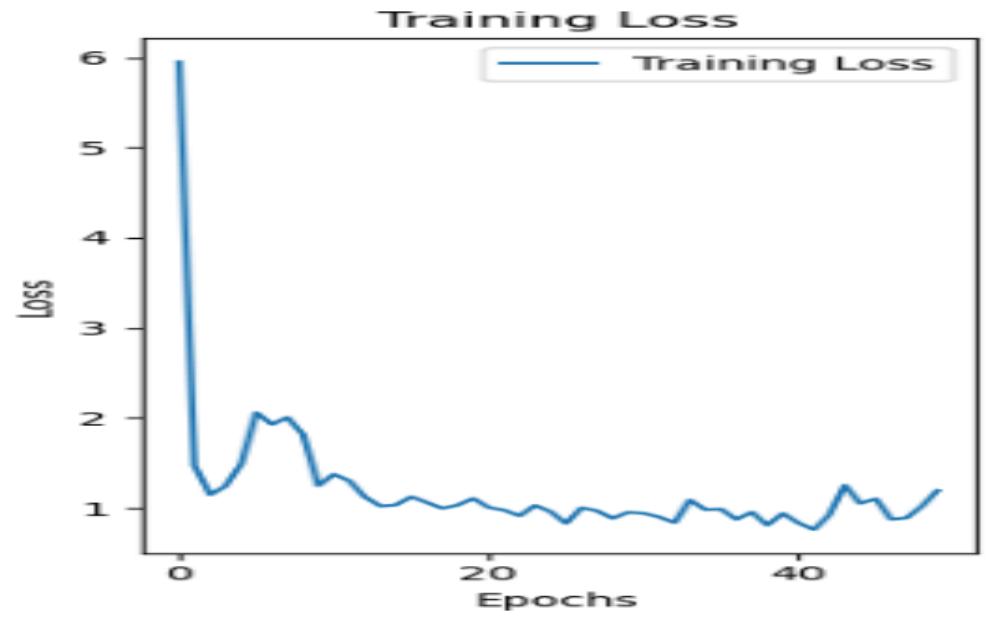

Figure 4.15: Training and Testing Loss plot for ResNet Model

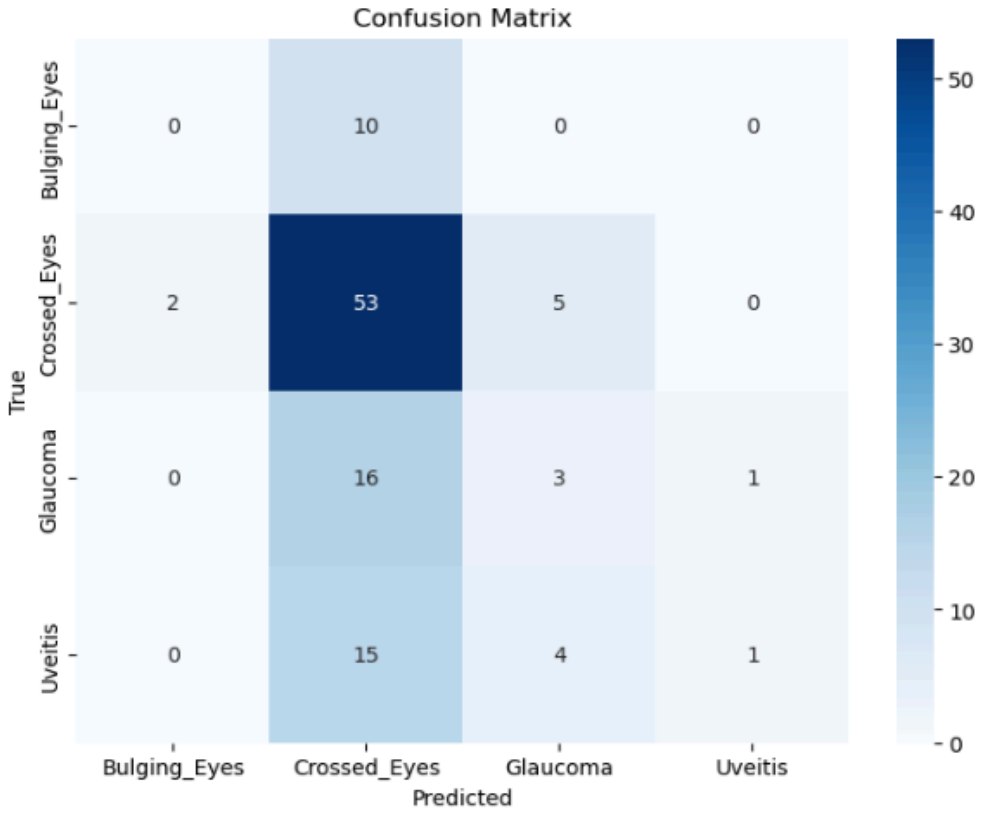

Figure 4.16: Confusion Matrix for ResNet

4.8 Experimentation and Analysis

After creating our desired model, we compared the models through different numerical results such as Validation Accuracy, Recall, Precision and F1 Score. After the experimentation we achieved the values which are shown in the Table 4.2 below. Our model did perform very close to the mentioned pre-trained models instead of it being a less layered CNN structure.

Table 4.6: Experimentation and Analysis

Architecture	Precision (%)	Recall (%)	F1 Score (%)	Accuracy (%)

CNN (Inception_net)	**91.26**	**96.79**	**90.14**	**86.00**
Resnet	**55.0**	**87.0**	**68.0**	**62.2**

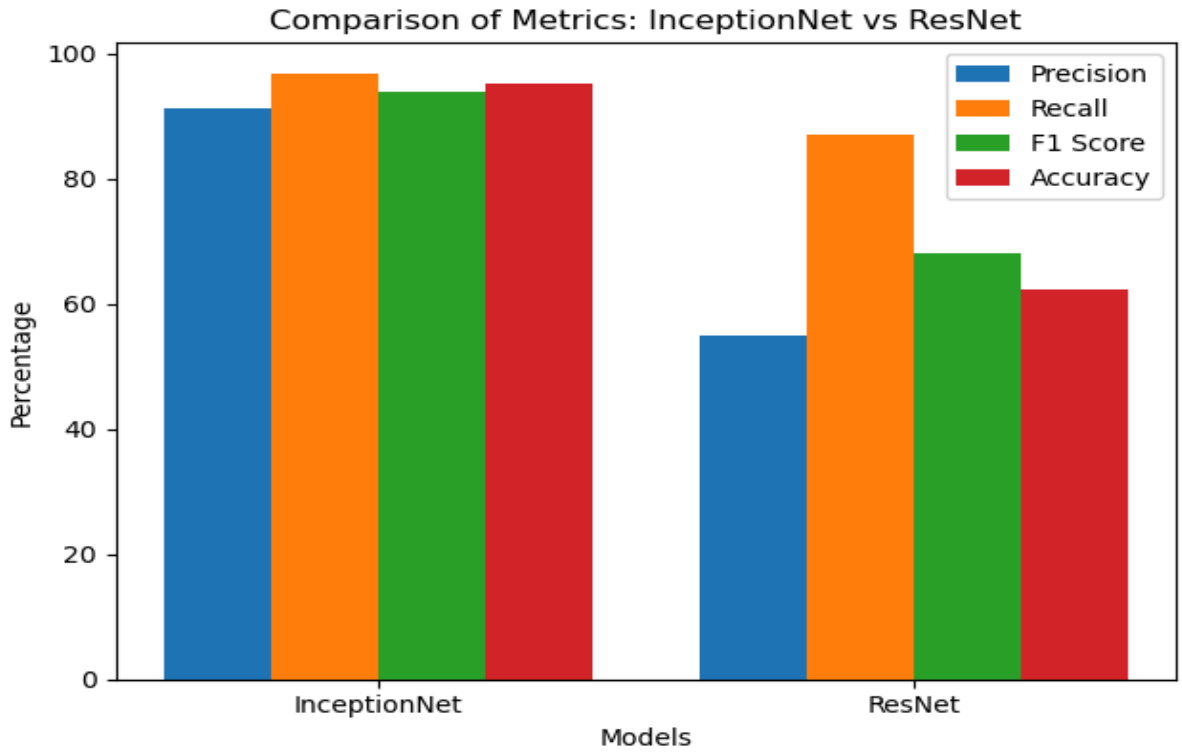

Figure 4.17: Comparison of Metrics: InceptionNet vs ResNet

4.9 System Implementation

The images are uploaded from the device to the system. Figure 4.16 shows us how the image is then uploaded/taken to the web application where it is sent to the deep learning model. The deep learning model is a pretrained model which classifies whether the eye does or does not have eye disease. The accuracy and the prediction are sent to the user. For the deep learning model, the images from the dataset are first augmented. Then, the

TensorFlow image data generator is used to increase the size of the dataset. Once the final dataset is ready, a deep learning model is trained on the images till the desired accuracy is obtained. Figure 4.17 shows us the result of the system not detecting an eye defect. Figure 4.18 shows us how the system accepts the image data. Figure 4.19 shows the result of the system detecting an eye defect

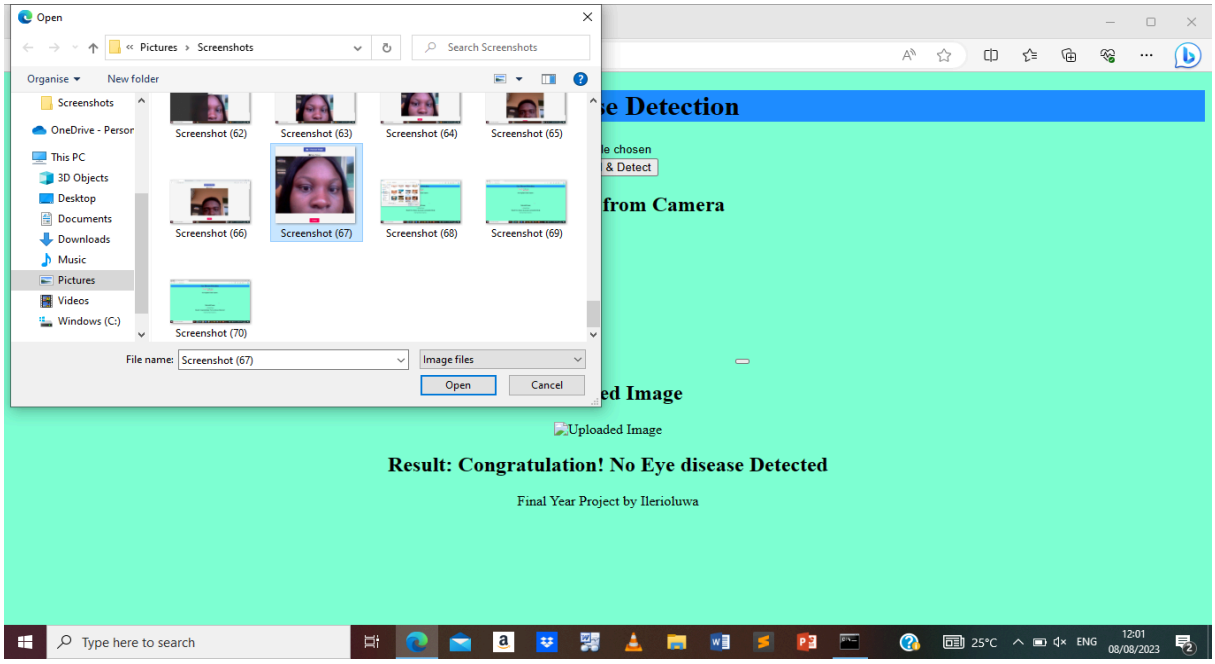

Figure 4.18: Eye Defect Detection system

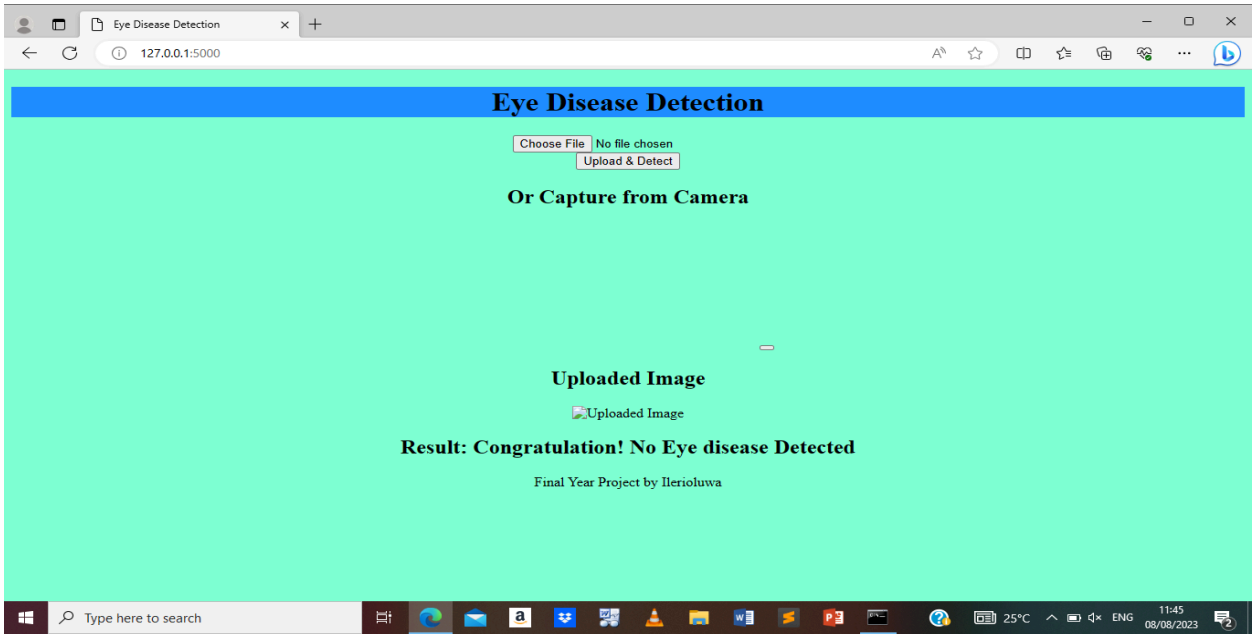

Figure 4.19: Result of eye input

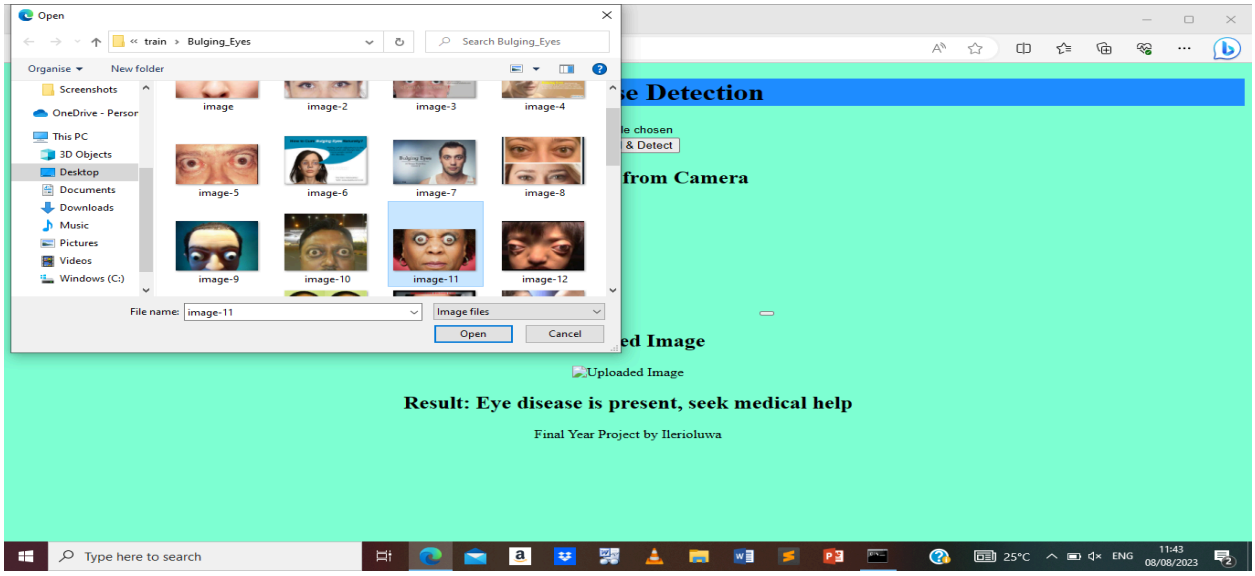

Figure 4.20: Eye Defect Detection system

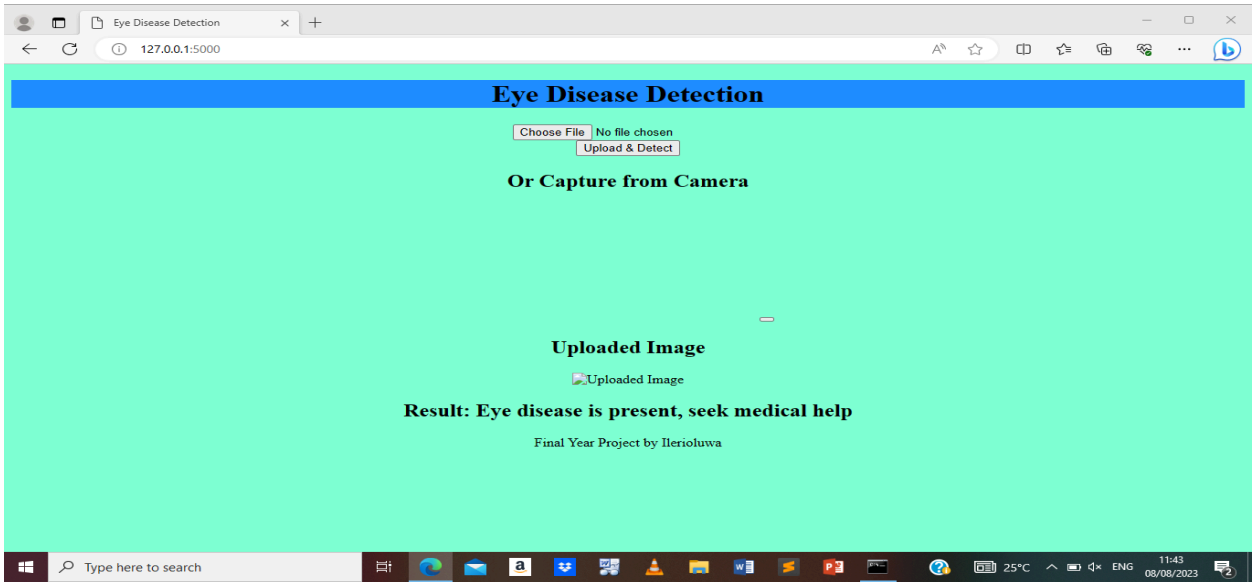

Figure 4.21: Result Eye Defect system

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.0 Introduction

This chapter summarises the whole study from conception to the resulting system and concludes the project in its entirety. The project's main goal was to develop a web based system that detects eye defects in the human eye.

5.1 Summary

Eye disease images are commonly used by medical experts like ophthalmologists, which are very helpful in detecting various eye disorders. They used this to diagnose the different types of eye diseases like Cataracts, Diabetic Retinopathy, Glaucoma etc. These eye images can be also used for the prediction of the severity of the diseases and can provide early signs or warnings. Recently, different machine learning algorithms are playing a vital role in the field of medical science, and it is no different in Ophthalmology either. In this project, we aim to automatically classify healthy and diseased eye images using deep neural networks. Because deep learning is an excellent machine learning algorithm, which has proven to be very accurate in computer vision problems. In this work, we used convolutional neural networks (CNN) to classify the eye images whether they are healthy or not. However, everyone with eye diseases should get a comprehensive dilated eye exam at least once in 4 months. This can help avoid its progression. But regular access to specialists is difficult for most people. Our model solves this problem by building hardware to be attached to a cellular device camera which acts as an indirect ophthalmoscope. The web app gives instant diagnosis on eye images using deep learning.

Every patient diagnosed with an eye disease can contact a doctor by sending them the results for further treatment plans or further diagnosis.

5.2 Conclusion

The study adopted CNN model to assist radiologists to quickly and accurately detect eye disease cases through the system. Experiments were carried out using these models. The efficiency of the models developed were evaluated using accuracy, and a comparison was made between our results and earlier studies reported in literature. The accuracy of 86% was recorded in the CNN model. The model will be able to guide the users to know their eye condition (for specified eye diseases namely crossed eye, bulged eye, conjunctivitis and cataract). The model is cost effective and has a simple interface. The model shares the details of the nearby eye doctors to use to get their eye check-up.

5.3 Limitations

The system is not portable for mobile devices yet, as it requires all of the directories to be downloaded before it can function perfectly.

5.4 Future Works

To improve the accuracy of our CNN model and shorten the training time, we have implemented convolutional layers, activation functions, max pooling, dense, and various other methods. It is our intention to add more layers with different parameters tweaked in the future to increase the accuracy of the model. Using a more challenging dataset, and then modifying it accordingly, will allow us to test its accuracy in a future implementation of the model.

REFERENCES

Abhishek, kumar. (2022, May 10). Introduction to artificial intelligence. Retrieved from Simple Talk website: https://www.red-gate.com/simple-talk/development/data-science-development/introduction-to-artificial-intelligence/

Accenture. (2021). What Is Artificial Intelligence | Accenture. Retrieved from www.accenture.com website: https://www.accenture.com/us-en/insights/artificial-intelligence-summary-index

Advani, V. (2020, April 29). What is Machine Learning? How Machine Learning Works and future of it? Retrieved from GreatLearning website: https://www.mygreatlearning.com/blog/what-is-machine-learning/

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., … Farhan, L. (2021c). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1). https://doi.org/10.1186/s40537-021-00444-8

Architectures, M. (2013). Deep Learning. Retrieved from O'Reilly | Safari website: https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html

Awati, R. (2022, September). What is convolutional neural network? - Definition from WhatIs.com. Retrieved from SearchEnterpriseAI website: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network

Badah, N., Algefes, A., AlArjani, A., & Mokni, R. (2022). Automatic Eye Disease Detection Using Machine Learning and Deep Learning Models. *Pervasive Computing and Social Networking*, 773–787. https://doi.org/10.1007/978-981-19-2840-6_58

Benbarrad, T., Salhaoui, M., Kenitar, S. B., & Arioua, M. (2021). Intelligent Machine Vision Model for Defective Product Inspection Based on Machine Learning. *Journal of Sensor and Actuator Networks*, *10*(1), 7. https://doi.org/10.3390/jsan10010007

Bernstein, S. (2016, November). Is Uveitis a Symptom of Something Else? Retrieved January 22, 2020, from WebMD website: https://www.webmd.com/eye-health/uveitis-symptom-something-else

Bian, X., Luo, X., Wang, C., Liu, W., & Lin, X. (2020). Optic disc and optic cup segmentation based on anatomy guided cascade network. *Computer Methods and Programs in Biomedicine*, *197*, 105717. https://doi.org/10.1016/j.cmpb.2020.105717

Biswal, A. (2023, February 16). Top 10 Deep Learning Algorithms You Should Know in (2021). Retrieved from Simplilearn.com website: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

Boyd, K., & Turbert, D. (2018, December 21). *Parts of the Eye*. American Academy of Ophthalmology. https://www.aao.org/eye-health/anatomy/parts-of-eye

Boyd, K. (2019, February 21). What is presbyopia? Retrieved from American Academy of Ophthalmology website: https://www.aao.org/eye-health/diseases/what-is-presbyopia

Brownlee, J. (2019, April 2). How to Normalize, Center, and Standardize Image Pixels in Keras. Retrieved from Machine Learning Mastery website: https://machinelearningmastery.com/how-to-normalize-center-and-standardize-images-with-the-imagedatagenerator-in-keras/

Burns, E. (2021, March). What Is Machine Learning and Why Is It Important? Retrieved from

SearchEnterpriseAI website: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML

Burns, E. (2022). What is artificial intelligence (AI)? Retrieved from TechTarget website: https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence

Calin, O. (2020). Deep Learning Architectures. In *Springer Series in the Data Sciences*. Cham:
Springer International Publishing. https://doi.org/10.1007/978-3-030-36721-3

Castañón, J. (2019, May). 10 Machine Learning Methods that Every Data Scientist Should Know.
Retrieved from Medium website: https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9

Castillo, D. (2021, October 29). *Machine Learning Regression Explained*. Seldon. https://www.seldon.io/machine-learning-regression-explained

Chitra Badii. (2019, August 20). Everything You Need to Know About Crossed Eyes. Retrieved from Healthline website: https://www.healthline.com/health/crossed-eyes

Culurciello, E. (2017, March 23). Neural Network Architectures. Retrieved from Medium website:
https://towardsdatascience.com/neural-network-architectures-156e5bad51ba

Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C. M., & Dario, P. (2020). Visual-Based Defect Detection and Classification Approaches for Industrial Applications—A SURVEY. *Sensors*, *20*(5), 1459. https://doi.org/10.3390/s20051459

Domen Tabernik, Samo Šela, J. Skvarc, & Danijel Skočaj. (2019). Deep-Learning-Based Computer Vision System for Surface-Defect Detection. *Deep-Learning-Based Computer*
Vision System for Surface-Defect Detection, *908*(90), 490–500. https://doi.org/10.1007/978-3-030-34995-0_44

Han, J.-H. (2022). Artificial Intelligence in Eye Disease: Recent Developments, Applications, and Surveys. *Diagnostics*, *12*(8), 1927. https://doi.org/10.3390/diagnostics12081927

How to Normalize, Center, and Standardize Image Pixels in Keras? (2021, February 23). Retrieved August 9, 2023, from GeeksforGeeks website: https://www.geeksforgeeks.org/how-to-normalize-center-and-standardize-image-pixels-in-keras/

IBM. (2023). What is Machine Learning? Retrieved from www.ibm.com website: https://www.ibm.com/topics/machine-learning

IBM. (2023). What is Artificial Intelligence (AI)? Retrieved from www.ibm.com website: https://www.ibm.com/topics/artificial-intelligence

Introduction to Convolution Neural Network. (2017, August 21). Retrieved from GeeksforGeeks website: https://www.geeksforgeeks.org/introduction-convolution-neural-network/

Journal, I. (2022). EYE DISEASE IDENTIFICATION USING DEEP LEARNING. *IRJET*. Retrieved from https://www.academia.edu/88384433/EYE_DISEASE_IDENTIFICATION_USING_DEEP_LEARNING

Kelley, K. (2022, September 29). What is Artificial Intelligence: Types, History, and Future.

Retrieved from Simplilearn.com website: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/what-is-artificial-intelligence

Kim, J., Tran, L., Peto, T., & Chew, E. Y. (2022). Identifying Those at Risk of Glaucoma: A Deep Learning Approach for Optic Disc and Cup Segmentation and Their Boundary Analysis. *Diagnostics*, *12*(5), 1063. https://doi.org/10.3390/diagnostics12051063

Laa, D. (2006, July 5). What are The Top Machine Learning (ML) Methods? Retrieved from

Tableau website: https://www.tableau.com/learn/articles/top-machine-learning-methods

Lee, W.-M. (2022, December 22). Image Data Augmentation for Deep Learning. Retrieved from Medium website: https://towardsdatascience.com/image-data-augmentation-for-deep-learning-77a87fabd2bf

Lewis, S. (2019). What is waterfall model? - Definition from WhatIs.com. Retrieved from SearchSoftwareQuality website: https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model

Liu, H., Li, L., Wormstone, I. M., Qiao, C., Zhang, C., Liu, P., … Kang, H. (2019). Development and Validation of a Deep Learning System to Detect Glaucomatous Optic Neuropathy Using Fundus Photographs. *JAMA Ophthalmology*, *137*(12), 1353. https://doi.org/10.1001/jamaophthalmol.2019.3501

Logan, S. (2017, August 26). Vision Defect Detection Systems. Retrieved August 6, 2023, from www.cameralyze.co website: https://www.cameralyze.co/blog/vision-defect-detection-systems

Mahapatra, S. (2019, January 22). Why Deep Learning over Traditional Machine Learning? Retrieved from Medium website: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063

Mathworks. (2019). What Is Deep Learning? | How It Works, Techniques & Applications. Retrieved from Mathworks.com website: https://www.mathworks.com/discovery/deep-learning.html

Mayo Clinic. (2022, September 30). Glaucoma - Symptoms and causes. Retrieved from Mayo Clinic website: https://www.mayoclinic.org/diseases-conditions/glaucoma/symptoms-causes/syc-20372839

Mishra, M. (2020, September 2). Convolutional Neural Networks, Explained. Retrieved from Medium website: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Muchuchuti, S., & Viriri, S. (2023). Retinal Disease Detection Using Deep Learning Techniques: A Comprehensive Review. *Journal of Imaging*, *9*(4), 84. https://doi.org/10.3390/jimaging9040084

Mukherjee, S. (2022, August 18). *The Annotated ResNet-50*. Medium. https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758

National Eye Institute. (2019). Refractive Errors | National Eye Institute. Retrieved from Nih.gov website:

https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/refractive-errors

National Eye Institute. (2022, April 20). How the Eyes Work | National Eye Institute. Retrieved from Nih.gov website: https://www.nei.nih.gov/learn-about-eye-health/healthy-vision/how-eyes-work

Pietrangelo, A. (2019). What You Should Know About Bulging Eyes. Retrieved from Healthline website: https://www.healthline.com/health/eyes-bulging

Patel, H. (2018, September 24). Anatomy of the Human Eye. Retrieved from News-Medical.net website: https://www.news-medical.net/health/Anatomy-of-the-Human-Eye.aspx

Potentia Analytics. (2019, December 19). What Is Machine Learning: Definition, Types, Applications and Examples. Retrieved from Potentia Analytics website: https://www.potentiaco.com/what-is-machine-learning-definition-types-applications-and-examples/

Saha, S. (2018, December 15). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | Saturn Cloud Blog. Retrieved from saturncloud.io website: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/

Ren, Z., Fang, F., Yan, N., & Wu, Y. (2021). State of the Art in Defect Detection Based on Machine Vision. *International Journal of Precision Engineering and Manufacturing-Green Technology*, *56*. https://doi.org/10.1007/s40684-021-00343-6

Ricochette, L. (2018). A simple way to understand machine learning vs deep learning – Zendesk. Retrieved from Zendesk website: https://www.zendesk.com/blog/machine-learning-and-deep-learning/

Takahashi, H., Tampo, H., Arai, Y., Inoue, Y., & Kawashima, H. (2017). Applying artificial intelligence to disease staging: Deep learning for improved staging of diabetic retinopathy. *PLOS ONE*, *12*(6), e0179790. https://doi.org/10.1371/journal.pone.0179790

Takao, S., Murata, A., & Watanabe, K. (2017). Gaze-Cueing With Crossed Eyes: Asymmetry Between Nasal and Temporal Shifts. *Perception*, *47*(2), 158–170. https://doi.org/10.1177/0301006617738719

Tang, T., Yu, Z., Xu, Q., Peng, Z., Fan, Y., Wang, K., … Zhao, M. (2020). *A machine learning-based algorithm used to estimate the physiological elongation of ocular axial length in myopic children*. *7*(1). https://doi.org/10.1186/s40662-020-00214-2

Team, K. (n.d.). Keras documentation: InceptionV3. Retrieved August 9, 2023, from keras.io website: https://keras.io/api/applications/inceptionv3

The Economic TImes. (2019). Definition of Systems Design | What is Systems Design ? Systems

Design Meaning - The Economic Times. Retrieved from The Economic Times website: https://economictimes.indiatimes.com/definition/systems-design

Tulbure, A.-A., Tulbure, A.-A., & Dulf, E.-H. (2022). A review on modern defect detection

models using DCNNs – Deep convolutional neural networks. *Journal of Advanced Research*, *35*(9098), 33–48. https://doi.org/10.1016/j.jare.2021.03.015

Varadarajan, A. V., Poplin, R., Blumer, K., Angermueller, C., Ledsam, J., Chopra, R., … Webster, D. R. (2018). Deep Learning for Predicting Refractive Error From Retinal Fundus Images. *Investigative Opthalmology & Visual Science*, *59*(7), 2861. https://doi.org/10.1167/iovs.18-23887

Wikipedia Contributors. (2019, April 29). Machine learning. Retrieved from Wikipedia website: https://en.wikipedia.org/wiki/Machine_learning

Zhang, Z., Srivastava, R., Liu, H., Chen, X., Duan, L., Kee Wong, D. W., … Liu, J. (2014). A survey on computer aided diagnosis for ocular diseases. *BMC Medical Informatics and Decision Making*, *14*(1). https://doi.org/10.1186/1472-6947-14-80

APPENDICES

Appendix A: Source Code for Inception-V3

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('./Eye_disease'):
        for filename in filenames:
    print(os.path.join(dirname, filename))
# General Libs
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
#from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2,
preprocess_input
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
# Alguns parâmetros para leitura do dataset
im_shape = (299,299)
TRAINING_DIR = './Eye_diseases/train'
TEST_DIR = './Eye_diseases/test'
seed = 10
BATCH_SIZE = 16
#Using keras ImageGenerator and flow_from_directoty
#data_generator = ImageDataGenerator(preprocessing_function=preprocess_input,
validation_split=0.2)
# With augmentation
```

```
data_generator = ImageDataGenerator(
        validation_split=0.2,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        preprocessing_function=preprocess_input,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
val_data_generator =
ImageDataGenerator(preprocessing_function=preprocess_input,validation_split=0.2)
# Generator para parte train
train_generator = data_generator.flow_from_directory(TRAINING_DIR,
target_size=im_shape, shuffle=True, seed=seed,
                                        class_mode='categorical', batch_size=BATCH_SIZE,
subset="training")
# Generator para parte validação
validation_generator = val_data_generator.flow_from_directory(TRAINING_DIR,
target_size=im_shape, shuffle=False, seed=seed,
                                        class_mode='categorical', batch_size=BATCH_SIZE,
subset="validation")
# Generator para dataset de teste
test_generator = ImageDataGenerator(preprocessing_function=preprocess_input)
test_generator = test_generator.flow_from_directory(TEST_DIR, target_size=im_shape,
shuffle=False, seed=seed,
                                            class_mode='categorical', batch_size=BATCH_SIZE)
nb_train_samples = train_generator.samples
nb_validation_samples = validation_generator.samples
nb_test_samples = test_generator.samples
classes = list(train_generator.class_indices.keys())
print('Classes: '+str(classes))
num_classes  = len(classes)
# Visualizando alguns exemplos do dataset por meio do Generator criado
plt.figure(figsize=(15,15))
for i in range(9):
        #gera subfigures
    plt.subplot(330 + 1 + i)
        batch = train_generator.next()[0]*255
        image = batch[0].astype('uint8')
    plt.imshow(image)
plt.show()
import matplotlib.pyplot as plt
plt.figure(figsize=(15, 15))
for i in range(9):
        # Generate subplots
```

```
    plt.subplot(330 + 1 + i)
        # Retrieve a batch of data from the generator
        batch = train_generator.next()
        images = batch[0]
        # Select the first image from the batch
        image = images[0]
        # Display the image
    plt.imshow(image)
plt.show()
import matplotlib.pyplot as plt
# Set up the data generators
train_generator = data_generator.flow_from_directory(TRAINING_DIR,
target_size=im_shape, shuffle=True, seed=seed,
                                        class_mode='categorical', batch_size=BATCH_SIZE,
subset="training")

# Visualize some examples from the dataset
plt.figure(figsize=(15, 15))
for i in range(9):
        # Generate subplots
    plt.subplot(330 + 1 + i)
        # Retrieve a batch of data and labels from the generator
        batch = train_generator.next()
        images = batch[0]
        labels = batch[1]
        # Select the first image and its corresponding label from the batch
        image = images[i]
        label = labels[i]
        # Convert label to class name (assuming you have a list of class names)
    class_names = ['Bulging_Eyes', 'Crossed_Eyes', 'Glaucoma', 'Uveitis']
        class_name = class_names[label.argmax()]
        # Display the image with 'nearest' interpolation
    plt.imshow(image, interpolation='nearest')
    plt.title(class_name)
plt.show()
base_model = InceptionResNetV2(weights='imagenet', include_top=False,
input_shape=(im_shape[0], im_shape[1], 3))
#
x = base_model.output
x = Flatten()(x)
x = Dense(100, activation='relu')(x)
predictions = Dense(num_classes, activation='softmax',
kernel_initializer='random_uniform')(x)
model = Model(inputs=base_model.input, outputs=predictions)
# Freezing pretrained layers
for layer in base_model.layers:
```

```
    layer.trainable=False
optimizer = Adam()
model.compile(optimizer=optimizer,loss='categorical_crossentropy',metrics=['accuracy'])
epochs = 50
# Saving the best model
callbacks_list = [
    keras.callbacks.ModelCheckpoint(
        filepath='model.h5',
        monitor='val_loss', save_best_only=True, verbose=1),
    keras.callbacks.EarlyStopping(monitor='val_loss', patience=10,verbose=1)
]
history = model.fit(
        train_generator,
        steps_per_epoch=nb_train_samples // BATCH_SIZE,
        epochs=epochs,
        callbacks = callbacks_list,
        validation_data=validation_generator,
        verbose = 1,
# Using the validation dataset
score = model.evaluate(validation_generator)
print('Val loss:', score[0])
print('Val accuracy:', score[1])
# Using the test dataset
score = model.evaluate_generator(test_generator)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
import itertools
#Plot the confusion matrix. Set Normalize = True/False
def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',
cmap=plt.cm.Blues):
    """
        This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.figure(figsize=(10,10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
        tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
        if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
       cm[np.isnan(cm)] = 0.0
        thresh = cm.max() / 2.
```

```
        for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
      plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")
   plt.tight_layout()
   plt.ylabel('True label')
        plt.xlabel('Predicted label')
# Some reports
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np
#Confution Matrix and Classification Report
Y_pred = model.predict_generator(test_generator)#, nb_test_samples // BATCH_SIZE,
workers=1)
y_pred = np.argmax(Y_pred, axis=1)
target_names = classes
#Confution Matrix
cm = confusion_matrix(test_generator.classes, y_pred)
plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')
print('Classification Report')
print(classification_report(test_generator.classes, y_pred, target_names=target_names))
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Define directories
valid_dir = './Eye_diseases/train'
test_dir = './Eye_diseases/train'
# Image dimensions and batch size
img_width, img_height = 224, 224
batch_size = 32
# Number of disease categories
num_classes = 4  # Change this to the number of your disease categories

# Validation and test data generators
valid_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_datagen = ImageDataGenerator(rescale=1.0 / 255)
valid_generator = valid_datagen.flow_from_directory(
        valid_dir,
   target_size=(img_width, img_height),
   batch_size=batch_size,
   class_mode='categorical'
)
test_generator = test_datagen.flow_from_directory(
        test_dir,
   target_size=(img_width, img_height),
   batch_size=batch_size,
```

```
    class_mode='categorical'
)
# Build a simple CNN model
model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(img_width, img_height, 3)),
    MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Load pre-trained weights if available
# model.load_weights('pretrained_weights.h5')
# Evaluate the model on the validation set
valid_loss, valid_accuracy = model.evaluate(valid_generator)
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator)
print("Validation accuracy:", valid_accuracy)
print("Test accuracy:", test_accuracy)
import numpy as np
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Define directories
train_dir = './Eye_diseases/train'
test_dir = './Eye_diseases/test'

# Image dimensions and batch size
img_width, img_height = 224, 224
batch_size = 32
# Number of disease categories
num_classes = 4
# Data augmentation
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    vertical_flip=True
)
```

```
train_generator = train_datagen.flow_from_directory(
        train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)
test_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_generator = test_datagen.flow_from_directory(
        test_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)
# Load pre-trained VGG16 model
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(img_width,
img_height, 3))
# Build a custom model on top of VGG16
model = Sequential([
    base_model,
        Flatten(),
        Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
        epochs=50
)
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator)
print("Test accuracy:", test_accuracy)
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Define directories
train_dir = './Eye_diseases/train'
test_dir = './Eye_diseases/test'
# Image dimensions and batch size
img_width, img_height = 227, 227  # Adjusted for AlexNet
batch_size = 32
```

```
# Number of disease categories
num_classes = 4

# Data augmentation
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    vertical_flip=True
)
train_generator = train_datagen.flow_from_directory(
        train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)
test_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)
# Build the AlexNet model
model = Sequential([
        Conv2D(96, (11, 11), strides=(4, 4), activation='relu', input_shape=(img_width,
img_height, 3)),
    MaxPooling2D((3, 3), strides=(2, 2)),
    Conv2D(256, (5, 5), activation='relu'),
    MaxPooling2D((3, 3), strides=(2, 2)),
    Conv2D(384, (3, 3), activation='relu'),
    Conv2D(384, (3, 3), activation='relu'),
    Conv2D(256, (3, 3), activation='relu'),
    MaxPooling2D((3, 3), strides=(2, 2)),
        Flatten(),
    Dense(4096, activation='relu'),
    Dropout(0.5),
    Dense(4096, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model
```

```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
          epochs=50
)
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator)
print("Test accuracy:", test_accuracy)
import numpy as np
from sklearn.metrics import classification_report
# ...

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator)
# Predict the class labels
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)
# Get the true class labels
true_classes = test_generator.classes
# Get the class labels mapping
class_labels = list(test_generator.class_indices.keys())
# Generate the classification report
report = classification_report(true_classes, predicted_classes, target_names=class_labels)
print("Test accuracy:", test_accuracy)
import numpy as np
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
# ...
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator)
# Predict the class labels
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)

# Get the true class labels
true_classes = test_generator.classes
# Generate the confusion matrix
confusion = confusion_matrix(true_classes, predicted_classes)
print("Test accuracy:", test_accuracy)
print("Confusion Matrix:\n", confusion)
# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels,
yticklabels=class_labels)
plt.xlabel('Predicted')
```

```
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
import numpy as np
from sklearn.metrics import classification_report
# ...
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator)
# Predict the class labels
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)
# Get the true class labels
true_classes = test_generator.classes
# Get the class labels mapping
class_labels = list(test_generator.class_indices.keys())
# Generate the classification report
report = classification_report(true_classes, predicted_classes, target_names=class_labels,
output_dict=True)
print("Test accuracy:", test_accuracy)
# Display precision, recall, and F1-score for each class
for label in class_labels:
    print(f"Class: {label}")
    print(f"Precision: {report[label]['precision']:.2f}")
    print(f"Recall: {report[label]['recall']:.2f}")
    print(f"F1-score: {report[label]['f1-score']:.2f}")
    print("--------------")
import matplotlib.pyplot as plt
# Model names
models = ['InceptionNet', 'ResNet']
# Metrics
precision = [91.26, 55.0]
recall = [96.79, 87.0]
f1_score = [93.94, 68.0]
accuracy = [95.15, 62.2]
# Bar width
bar_width = 0.2

# Set positions for bars
pos_precision = list(range(len(models)))
pos_recall = [pos + bar_width for pos in pos_precision]
pos_f1_score = [pos + bar_width for pos in pos_recall]
pos_accuracy = [pos + bar_width for pos in pos_f1_score]
# Plotting
plt.bar(pos_precision, precision, width=bar_width, label='Precision')
plt.bar(pos_recall, recall, width=bar_width, label='Recall')
plt.bar(pos_f1_score, f1_score, width=bar_width, label='F1 Score')
```

```
plt.bar(pos_accuracy, accuracy, width=bar_width, label='Accuracy')
# Set x-axis labels
plt.xticks([pos + bar_width * 1.5 for pos in pos_precision], models)
# Labeling and title
plt.xlabel('Models')
plt.ylabel('Percentage')
plt.title('Comparison of Metrics: InceptionNet vs ResNet')
plt.legend()
# Display the plot
plt.tight_layout()
plt.show()
```

APPENDIX B: Source Code for ResNet-50

```
import os
from flask import Flask, render_template, request, redirect, url_for
from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
app = Flask(__name__)
# Load your trained model
model = load_model('model.h5')
# Configure allowed extensions and upload folder
ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png'}
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
def allowed_file(filename):
        return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS
@app.route('/', methods=['GET', 'POST'])
def index():
        result = None
        image_path = None
        if request.method == 'POST':
        if 'file' not in request.files:
        return redirect(request.url)
        file = request.files['file']
        if file.filename == '':
        return redirect(request.url)
        if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)

        # Load and preprocess the uploaded image
        img = image.load_img(filepath, target_size=(299, 299))  # Change the target_size
```

```
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array /= 255.0
        # Make a prediction
        prediction = model.predict(img_array)
        if prediction[0][0] > 0.5:
                result = "Eye disease is present, seek medical help"
        else:
                result = "Congratulation! No Eye disease Detected"
        image_path = filepath
        return render_template('index.html', result=result, image_path=image_path)
if __name__ == '__main__':
   app.run(debug=True)
import os
from flask import Flask, render_template, request, redirect, url_for
from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

app = Flask(__name__)

# Load your trained model
model = load_model('model.h5')

# Configure allowed extensions and upload folder
ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png'}
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allowed_file(filename):
        return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

@app.route('/', methods=['GET', 'POST'])
def index():
        result = None
        image_path = None

        if request.method == 'POST':
        if 'file' not in request.files:
        return redirect(request.url)

        file = request.files['file']

        if file.filename == '':
```

```
        return redirect(request.url)

        if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)

        # Load and preprocess the uploaded image
          img = image.load_img(filepath, target_size=(299, 299))  # Change the target_size
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array /= 255.0

        # Make a prediction
        prediction = model.predict(img_array)
        if prediction[0][0] > 0.5:
                result = "Eye disease is present, seek medical help"
        else:
                result = "Congratulation! No Eye disease Detected"

        image_path = filepath

        return render_template('index.html', result=result, image_path=image_path)

if __name__ == '__main__':
    app.run(debug=True)
import os
from flask import Flask, render_template, request, redirect, url_for
from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

app = Flask(__name__)

# Load your trained model
model = load_model('model.h5')

# Configure allowed extensions and upload folder
ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png'}
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allowed_file(filename):
        return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS
```

```
@app.route('/', methods=['GET', 'POST'])
def index():
        result = None
        image_path = None

        if request.method == 'POST':
        if 'file' not in request.files:
        return redirect(request.url)

        file = request.files['file']

        if file.filename == '':
        return redirect(request.url)

        if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)

        # Load and preprocess the uploaded image
        img = image.load_img(filepath, target_size=(299, 299))  # Change the target_size
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array /= 255.0

        # Make a prediction
        prediction = model.predict(img_array)
        if prediction[0][0] > 0.5:
                result = "Eye disease is present, seek medical help"
        else:
                result = "Congratulation! No Eye disease Detected"

        image_path = filepath

        return render_template('index.html', result=result, image_path=image_path)

if __name__ == '__main__':
    app.run(debug=True)
import os
from flask import Flask, render_template, request, redirect, url_for
from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
app = Flask(__name__)
```

```
# Load your trained model
model = load_model('model.h5')
# Configure allowed extensions and upload folder
ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png'}
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
@app.route('/', methods=['GET', 'POST'])
def index():
    result = None
    image_path = None
    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files['file']
        if file.filename == '':
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(filepath)
            # Load and preprocess the uploaded image
            img = image.load_img(filepath, target_size=(299, 299))  # Change the target_size
            img_array = image.img_to_array(img)
            img_array = np.expand_dims(img_array, axis=0)
            img_array /= 255.0
            # Make a prediction
            prediction = model.predict(img_array)
            if prediction[0][0] > 0.5:
                result = "Eye disease is present, seek medical help"
            else:
                result = "Congratulation! No Eye disease Detected"

            image_path = filepath
    return render_template('index.html', result=result, image_path=image_path)
if __name__ == '__main__':
    app.run(debug=True)
<!DOCTYPE html>
<html>
<head>
    <title>Eye Defect Detection - Result</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
```

```
</head>
<body>
    <h1 class="title">Eye Defect Detection - Result</h1>
    <div class="result-container">
        <h2 class="result-heading">Conclusion:</h2>
        <p class="result-text">{{ conclusion }}</p>
    </div>
    <footer class="footer">
        <p class="footer-text">Developed by Ilerioluwa</p>
    </footer>
    <script src="{{ url_for('static', filename='js/main.js') }}"></script>
</body>
</html>
```