GLAUCOMA DETECTION BASED ON DEEP NEURAL NETWORKS

A Project Report

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

L. Ruthvika (16071A0528)

K. Madhav (16071A0532)

G. Laasya (16071A0554)

G. Yashna (16071A0519)

Under the Guidance of

Mr. M. Ravikanth

Assistant Professor

Department of Computer Science and Engineering

VNR Vignana Jyothi Institute of Engineering & Technology

(Affiliated to JNTU Hyderabad & Recognized by AICTE)

Bachupally, Hyderabad.

April, 2020

VALLURUPALLI NAGESHWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institute, NAAC Accredited With 'A++' Grade, NBA Accredited, Approved by AICTE, New Delhi, Affiliated to JNTUH)

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad-500 090.

CERTIFICATE

This is to certify that **L. Ruthvika** (16071A0528), **K. Madhav** (16071A0532), **G. Laasya** (16071A0554) and **G. Yashna** (16071A0519) have successfully completed their major project work at CSE Department of VNR VJIET, Hyderabad entitled **"GLAUCOMA DETECTION BASED ON DEEP NEURAL NETWORKS"** in partial fulfilment of the requirements for the award of B. Tech Degree during the academic year 2019-2020. This work is carried out under my supervision and has not been submitted to any other University/Institute for the award of any degree/diploma.

Mr. M. Ravikanth
Internal Guide
Department of CSE
VNRVJIET

Dr. B. V. Kiranmayee
Professor & HOD
Department of CSE
VNRVJIET

DECLARATION

We, **L. Ruthvika** (16071A0528), **K. Madhav** (16071A0532), **G. Laasya** (16071A0554) and **G. Yashna** (16071A0519) hereby declare that the project entitled "**GLAUCOMA DETECTION BASED ON DEEP NEURAL NETWORKS"** carried out under the guidance of **Mr. M. Ravikanth** is submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. This is a bonafide record carried out by us. The results embodied in the project report have not been submitted to any other university or institute for the award of any other degree or diploma.

L. Ruthvika	**K. Madhav**	**G. Laasya**	**G. Yashna**
16071A0528	**16071A0532**	**16071A0554**	**16071A0519**
IV B. Tech CSE	**IV B. Tech CSE**	**IV B. Tech CSE**	**IV B. Tech CSE**
VNRVJIET	**VNRVJIET**	**VNRVJIET**	**VNRVJIET**
HYDERABAD	**HYDERABAD**	**HYDERABAD**	**HYDERABAD**

ACKNOWLEDGEMENT

Over a span of three and a half years, VNRVJIET has helped us transform ourselves from mere amateurs in the field of computer science into skilled Engineers capable of handling any given situation in real time.

We are highly indebted to the Institute for everything that it has given us. We would like to express our gratitude towards the Principal of our Institute, **Dr. Challa Dhanunjaya Naidu** and the Head of the Computer Science and Engineering Department, **Dr. B. V. Kiranmayee** for their kind co-operation and encouragement which helped us complete the project in the stipulated time.

Although we have spent a lot of time and put in a lot of effort into this project, it would not have been possible without the motivating support and help of our project coordinators **Mrs. V. Baby** and **Dr. Ch. Suresh** and also our project guide **Mr. M. Ravikanth.** We thank them for their guidance, constant supervision and for providing necessary information to complete this project.

Our thanks and appreciations also go to all the faculty members, staff members of VNRVJIET, to our parents who have supported and helped us at every stage of our lives and all our friends who have helped us put this project together.

ABSTRACT

Glaucoma is one of the many eye diseases that can lead to blindness if it is not detected and treated in proper time. It is often associated with the increase in the intraocular pressure (IOP) of the fluid (known as aqueous humour) in the eye, and it has been nicknamed as the “Silent Thief of Sight”. Glaucoma affects 40% of blindness in Singapore and is the second leading cause of blindness in the world.

Glaucoma is a condition that affects the optic nerve which can lead to irreversible and progressive loss of vision. The early detection of Glaucoma occurrence can be very beneficial for clinical treatment. Despite the fact that there are instruments for leading optic nerve investigation, they don't analyse this malady consequently. In recent times, deep convolutional neural networks show superior efficiency in image classification compared to previous handcrafted image classification methods based on features. This proposes a method that trains from a dataset of fundus digital pictures and fabricates a model. This model is utilized to anticipate Glaucoma and its severity utilizing profound CNN (Convolution Neural Networks). We built a system with CNN architecture which will be able to recognize the multifaceted features which are present in the task of classification.

The overall objective is to apply image processing techniques on the digital fundus images of the eye for the analysis of glaucoma and normal eye. By extracting information of pixel average value from the images, it is possible to obtain the necessary value for classification.

Contents Page. No

Contents Page. No

Contents Page. No

Contents Page. No

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Introduction

Glaucoma is one of the most severe disorders of the eye. It's actually a group of disorders, often characterized by the presence of higher pressure in the intraocular chamber (IOP), which causes damage on the optical nerve. W. H. O. says, it’s the world's second-largest cause regarding blindness. Glaucoma is popularly known as a snitch sight robber because it typically has no signs before irreversible vision loss resembling a hushed slayer.

The word 'glaucoma' can be known back to byzantine and Greek times. It is mainly derived from the Greek word "glaukos" which first appeared in Homer's works in 762 BCE where it is declared as a "sparkling silver glow" and later as a definition of sky-blue or green colours.

(a) Normal Vision *(b) Glaucoma Vision*

Fig. 1.1.1 Glaucoma Vision

Around 400 BCE Hippocrates, distinguished Greek physicist and Father of Medicine, makes use of the expression "glaucosis" in his work "Aphorisms" to spell it out conditions linked to eyesight dimming. Later on inside century which 17th an English ophthalmologist by title Richard Banister, known as the 'Father of British Ophthalmology,' has provided the initial, clear description regarding absolute glaucoma, and establishes the link between increased eyeball tension and glaucoma. Von Helmholtzis important 1850 innovation for the diagnosis had been permitted by the ophthalmoscope of glaucomatous alterations in the fundus. In 1857 treatment that is experimental glaucoma ended up being accomplished the very first successful glaucoma treatment ended up being postponed for many years since the reasons for

glaucoma could not be adequately identified. Technology today still doesn't know the precise causes of glaucoma however with early diagnosis of glaucoma, complete losing eyesight with therapy may be prevented.

Inside 1970s, automatic diagnostic glaucoma was implemented. Later in the 1990s the focus went on to glaucoma detection at a very early stage using structural improvements in optic disks to diagnose glaucoma before it affects the eyesight. OCT was suggested in very early 2000 to gather information that's retinal interior levels so the infection may be identified early so it can be treated over time to stop loss of sight.

Lot of potential patients with glaucoma globally into the coming years show escalation that's alarming. Figure 1.1.2 displays a graph representing projected patients with glaucoma in the coming decades. This season worldwide number of patients with glaucoma was 60 million based on data acquired through the 2010 and 2020 records of the world population by UN. According to research performed in 2013, it was stated that there were about 64.5 million glaucoma patients global in 2013 using the Bayesian Meta Regression Model which it could increase to 76 million in 2020 and grow to a million in 2040 further. However, evidence shows that manual glaucoma prediction and detection is quite difficult and delicate in nature, and relies totally on professional abilities. Provided the massive amount of increases in glaucoma affected people, we need to develop efficient automatic systems with the capacity of detecting glaucoma in first stages, so very early treatment will avoid loss of sight.

This research is a meta-analysis of glaucoma affected population within the last decade.

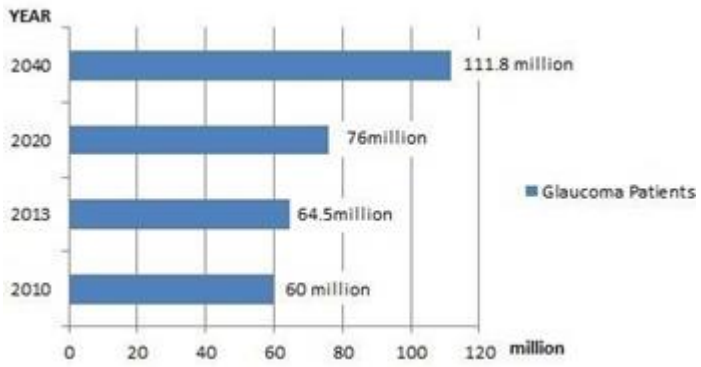


Fig. 1.1.2. Estimated Increase in Glaucoma Patients Worldwide

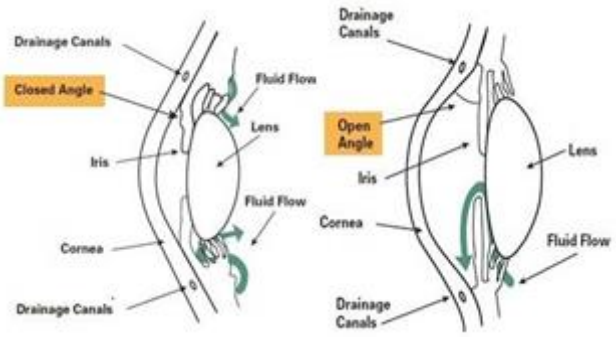
Glaucoma Global Statistics

W.H.O. has announced Glaucoma become the world's second-largest reason for blindness plus it includes 15% world’s loss of sight cases that comprise 5.2 million individuals worldwide. Considering epidemiological evidence, the WHO performed a study in 2002 and published the findings that 12.3 percent for the blindness was caused by glaucoma and 4.4 million individuals lost their vision. Estimate says approximately 60 million people suffering with glaucomatous neuropathy in 2012 plus an extra 8.4 million people go blind per year due to glaucoma. Such estimates are anticipated to increase to 80 million by 2020 and 11.2 million by 2020. Glaucoma prevalence varies region-wise, as open-angle glaucoma ended up being within North American and individuals that are European, Pakistan, Asia have actually almost half of total glaucoma patients among Asian countries. Natives of China, India, Singapore, and Mongolia have predominance of glaucoma closed and open angle.

Glaucoma Causes

If the stress in optical eye is higher than or corresponding to 21 mmHg raises the possibility of ocular hypertension. The Ocular Hypertension Treatment Survey says, 4.5 to 9.4 percent of Americans 40 years or older have actually ocular hypertension, which raises the risk of developing a sight-threatening glaucoma. Increased IOP could be the trigger that is primary for the presence and progression of glaucoma. However much fluid is produced due to IOP, the eye usually discharges equivalent amount of fluid and retains a balance. The fluid made for a glaucoma attention just isn't discharged through the eye. This raises retinal force and causes stress on nerve that's optic. Harm to the optic nerve weakens the perception and awareness of objects that may induce loss of sight.

Types of glaucoma



(a) Angle-Closure glaucoma (b) Open-angle glaucoma

Fig.1.1.3 Open-angle glaucoma VS Angle-Closure glaucoma

There are a wide number of glaucoma types. Table 1 discusses certain group forms. Globally, the absolute most common style of glaucoma is glaucoma which open-angle comprising at the very least 90 percent of all cases of glaucoma. Globally prevalent is the open-angle glaucoma. Figure 1.1.3 (a) represents glaucoma with angle closure while 1.1.3 (b) shows glaucoma with open-angle. Open-angle glaucoma progresses gradually and has now no symptoms visible. Angle-closure glaucoma progresses quickly and contains clear main signs or symptoms. Open-angle glaucoma include progressive losing peripheral vision, typically in higher level phases of both eyes and tunnel vision, while severe angle-closure glaucoma signs and symptoms include eye pain, nausea.

Glaucoma diagnosis

Glaucoma is identified by the visualisation regarding the image that's retinal interior structure. The retina is weakened, either by one or both eyes which are glaucomatous. Glaucoma does not have any warning signs and roughly 50percent of people with glaucoma are unaware of the illness; hence, a thorough, dilated study of the eye is essential for early diagnosis and treatment that's early. An eye fixed is looked over at least one time annually. Measuring the eye that is inner is suggested by the tonometry test. Most forms of glaucoma are identified as having higher than 20 mm Hg. Measures examined help diagnose as a type of glaucoma.

1.1.1 Deep Learning

Deep learning is one of the branches of machine learning that is based mainly on synthetic neural systems, whilst the neural networks imitate the brain of the human being. For this reason, Deep learning can be a kind of replica for human mind. Through deep learning, we do not have to program everything explicitly. Deep learning meaning is not a new concept. It has been around for quite some time now. Nowadays it is on speculation, because we don't have so much computing power and a lot of information before. While the computing ability has increased exponentially within the last two decades, deep learning and machine learning came into the picture. An idea that is formal of learning is neurons.

Deep learning is actually a particular as a type of learning which achieves power which great with flexibility by understanding how to make global representations as nested hierarchy comprising concepts, with every concept identified in reference to simpler principles, and furthermore abstract representations determined in terms of rather fewer abstract ones.

Around 100 billion neurons estimated to be in the human brain together this is often a image of an neuron that is individual, linked by 1000s of their neighbours.

How do we reproduce these neurons in a device? So, we build a system that's artificial, a synthetic neural web where nodes or neurons are present. We now have some neurons for input value and some for output value, and there might be plenty of neurons within the layer that are hidden in between.

Architecture:

1. Deep Neural system – It's a neural network with some extent of complexity (having numerous hidden layers between input and production layers). They are able to pattern and process relationships which are non-linear.

2. Deep Belief Network-a Deep Neural Network type. This is usually a system of multi-layered beliefs.

Steps for doing DBN:

a. utilising the Contrastive Divergence algorithm to understand a layer of functions from the devices being observable.

b. Treat activations of features that were formerly trained as noticeable devices, and discover features then.

c. Finally, as soon as the learning for the final layer that's hidden is completed, the entire DBN is educated.

3. Neural Network (perform equivalent function for each dimension of a sequence)- Allows parallel and computation that is sequential. They are going to remember things that are essential from the feedback they received, and therefore help them to be more accurate.

1.1.2 Deep Learning for Glaucoma Detection

Glaucoma is the world's second-largest cause of blindness, affecting around 2.7 million people in the US alone. It is a complex collection of diseases, which can lead to blindness if left untreated. In Australia, it is a especially large problem, where only 50 percent of all people who have it are actually diagnosed and receive the care they need.

Deep learning, also called highly organized hierarchical learning, is a significant part of the family of machine learning methods. It allows computational models consisting of multiple layers of processing to be fed with raw data, and to automatically learn several levels of abstract data representation for detection and classification. Many are looking at new ways to use AI to help ophthalmologists and optometrists use eye images better, and potentially help speed up the vision recognition process for glaucoma.

Glaucoma is usually treated using a number of tests, such as intraocular pressure measures and field vision examinations, as well as fundus and OCT imaging. OCT provides an efficient way of visualizing and quantifying structures in the eye, namely the layer of retinal nerve fibers (RNFL), which changes with disease progression.

Because that method works well, quantifying the RNFL in OCT images involves an additional procedure. Usually, these techniques often clean the input data in a number of ways, such as flipping all eyes into the same orientation (left or right) to minimize data variation to boost classifier efficiency. Our approach removes these additional steps and indicates that there is no need for these potentially time-consuming stages to detect glaucoma. Contrary to most of the existing detection techniques, specific CNN architectures are provided for glaucoma assessment. This method does not require any collection of features or precise measurements of geometric optic nerve head structures such as CDR effectively capturing the deep glaucoma characteristics based on deep CNN is our main interest.

1.2 Current Systems

Some computational tools found in our present framework are automatic Clinical Decision help Systems (CDSS) in ophthalmology, including CASNET / glaucoma, which are made to establish decision which efficient systems for detecting condition pathology in human eyes. For decades, these CDSS utilized glaucoma as being a case that is prevalent by extracting structural, contextual or textual features from retinal images. The image that's retinal methods are derived from computational techniques to minimise the variation which can be achieved because of different monitoring growth of structural faculties within the eyes by the clinicians.

Another technique employing structural features in order to identify glaucomatous progression is proper orthogonal decomposition (POD). Pixel level information such as location or area specific and the texture characteristic is known as the pixel intensity spatial variation. Glaucomatous image detection can also be achieved using texture features and the characteristics of higher order spectra (HOS).

One of the most famous current systems is considering loss of the Retinal Nerve Fiber Layer (RNFL) as a major parameter for the detection of glaucoma, which is not very normal. The fundus image is extracted from the co-occurrence gray level matrix for feature extraction, so that the ONH (Optic Nerve Head) area is removed and the rest of the RNFL is divided into ISNT-based sub-sectors.

Support Vector Machines (SVM) were used as another common method which was powerful. Also, SVMs are found to be highly efficient when managing high dimensional spaces. Phase knowledge tends to be amiss, because the Radon Transform does not retain it. In addition, some of the information found in the image is lost by used projections. RT further increases difficulty of mathematics and introduces error. They have reached 98.8 per cent and 95 per cent classification accuracy.

Achieving optic disc segmentation using vasculature observations present in the retinal region is also a prevalent approach by red channel analysis. Clustering is first implemented using Simple Linear Iterative Clustering (SLIC) followed by the k-Means Clustering algorithm along with the edge detection filter Gabor. Predictions are made using CDR. The dataset used consists of 100 images with an F-score value of 96%.

Hand crafted methods

There are three different tests which can detect the glaucoma:

1. Ophthalmoscopy
2. Tonometry
3. Perimetry

Two routine eye tests include: Tonometry and Ophthalmoscopy for regular glaucoma check-ups. Most glaucoma tests take a lot of time, and need special skills and diagnostic equipment as well. New techniques for early phase diagnosis of glaucoma are therefore urgently needed with precision, speed and even with less skilled people. Computer-based tools over the last year have made screening of glaucoma simpler. Imaging technologies such as fundus cameras, optical coherence tomography (OCT), retina tomography in Heidelberg (HRT), and laser polarimetry scanning have been widely used for eye diagnostics.

HRT, laser confocal tomography scanning, and OCT will display damage to the retinal nerve fibre even before the visual field injury. The equipment, however, is very costly and only certain hospitals can afford it. Therefore, many ophthalmologists will use fundus cameras as an option for diagnosing glaucoma. Technique for processing images enables the extraction of features that can provide valuable details for diagnosing glaucoma.

1.2.1 Drawbacks of Current System

- Shortcomings such as location, thresholding can lead to inacceptable results and eventual errors in the diagnosis of glaucoma.
- Precise definition of these undefined borders is a challenging task.
- Robustness to major intra class variations is difficult to achieve.
- They need Digital Fundus images (DFIs) of high quality, but most optometrist scans do not capture such high quality.
- The appropriate training datasets are not large enough, either in magnitude or quality.

1.3 Proposed System

We will be developing a Deep Learning architecture in our system, with CNN working at its core to automate glaucoma detection.

Deep learning systems, such as Convolutional Neural Networks, may deduce a

hierarchical representation of images to distinguish patterns of diagnostic decisions between Glaucoma and non-Glaucoma. Various architectures of deep learning, such as ResNet50 will be used for tasks of image classification. This model that is trained using convolutional neural networks should be more accurate in diagnosing glaucoma than models previously published.

CNNs effectively remove the need for manual extraction of the element, so you don't anymore have to recognize the features used to classify images. This automatic extraction of a function makes deep learning models highly accurate and suitable for the computer vision tasks such as classification of objects.

Problem Statement

Glaucoma is an ailment that disproportionately affects many individuals who are under 40 to 60 years of age. Its worldwide average is 5 per cent, which is slowly increasing. Early detection of glaucoma is especially important because it allows prompt care to avoid significant loss of the visual field. A practical vision loss test requires special equipment that is available only in hospitals in the area and is therefore unsuitable for screening. Sickness position (GLAUCOMA) in the light of profound learning by using image planning on (FUNDUS, OCT).

System Architecture:

The proposed System Architecture explains the following;

For fundus photography performed at extremely distinct lighting levels, patients of varying age groups, gender and nationality are considered, and images are collected to form the dataset. Considering these varied data leads to changes in the dimensions of the pixels within the images that could produce pointless variations. To counteract this, image pre-processing methods such as image cropping, colour normalization, measurements rendering, data increase etc. were performed on the images. The image data is converted to a hierarchical data format to facilitate pre-processing, followed by data augmentation, and then trained accordingly. Images were increased in real-time for improvisation in network localization capacity and also to help reduced overfitting. Techniques such as horizontal flip, vertical flip, zoom in of 20 percent were performed to make the data standardized as the number of images were not distributed equitably in the dataset for each sort. Data increase makes the model more robust to slight variations, thus preventing overfitting of the model. The dataset was formatted to 128

* 128 pixels, allowing the retention of complex features to be identified

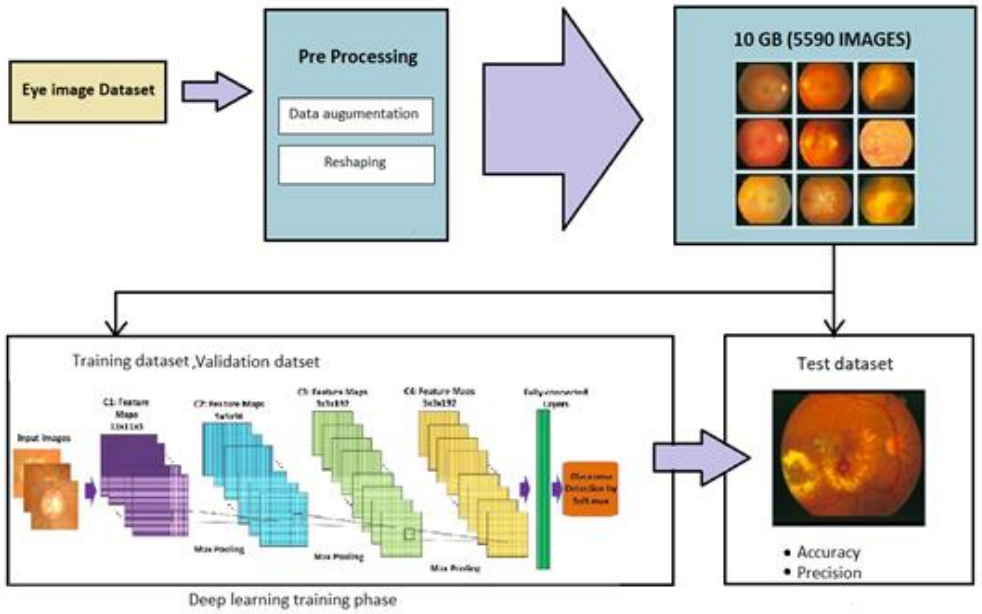


Fig 1.3 System Architecture

1.3.1 Advantages of Proposed System

Numerous research scholars have employed a variety of methods to identify Glaucoma disease, but the neural networks method of convolution has yielded better results among all. So, CNN (Convolution Neural Networks) is used in the framework that we are proposing to construct the model that predicts Glaucoma from only a picture that is given as input. Changing the application of handcrafted features, a CNN can be used being a function is learned because of its hierarchy that can easily be well useful for purposes of image classification. Accuracy for the category that's CNN-based of method are greater, while the hierarchy approach becomes offered to discover a whole lot of more complex features, plus the interpretation together with distortion features in higher layers. Inside work, we can explore the employment of the strategy that is CNN-based Glaucoma evaluation according to this presumption.

Unlike existing methods, in which features were handcrafted from the optical disk, CNN will automatically extract the features from raw images in our method, which will eventually be fed to the classifier. The classifier then performs a classification into the respective labels on the images. The manual analysis of the ophthalmic images is a time-consuming process. The accuracy often varies with the professional differences in expertise and skill. Our method offers far better responses to the disadvantages of

Glaucoma perception involving manual processes. Most established detection techniques require either feature selection or very precise measurement of geometric ONH structures, which is CDR. Using automation, the processes of identification, examination, diagnosis, treatment time, and the associated risk reduction can be made effective. It has been proved useful for Convolutional Neural Networks (CNNs) and other DL (Deep Learning) frameworks.

2. FEASIBILITY STUDY

A feasibility analysis includes issuing a summons to judgment on whether a proposal is feasible. The two parameters for judging viability are cost and value that is expected to be delivered. A research which well-designed incorporate a history of the company or project, service summary, accounting documents, operations and administration information, advertising analysis and policies, monetary data, legal sort of demands and taxation responsibilities. Such studies generally speaking precede the start of technical development and implementation of the projects. A feasibility research evaluates the possibility success of the task; hence recognized objectivity becomes a component that is very important the analysis's credibility to possible investors and lending organizations.

2.1 Technical Feasibility

Technical feasibility means evaluation of the proposed system's hardware and software specifications. Deep Learning technology is utilized in this project. Uses frameworks and libraries such as TensorFlow-GPU, Jupyter Notebook, Keras, Numpy. These play a major role in ensuring that project specifications are met with reasonable accuracy and within a minimum of time. They can turn concepts into working structures. Part that is major of evaluation is because of technical feasibility evaluations. It requires under consideration the technical requirements associated with the project that is proposed. Then, the specifications which are technological contrasted with the company's technical capability. The project of systems is regarded as theoretically feasible as soon as the internal capability which technical adequate to generally meet the specs associated with project.

The analyst will figure out or perhaps a current tool which are technological be updated or included with in a manner that satisfies the necessity being considered. It is where device analysts' knowledge is valuable, because they would be able to address the technological feasibility question using their own experience and their interaction with vendors.

2.2 Economic Feasibility

Economic feasibility helps in determining the viability, expense, and benefits of projects before allocating financial resources. This assessment usually includes an analysis of project costs / benefits. The application is structured in such a way that it requires minimal resources and reduces costs, because manual labour would be minimally required. The technologies used help to understand the consumer without investing in something.

2.3 Legal Feasibility

The proposed system does not conflict with legal requirements such as data protection acts or laws relating to social media. It ensures lawful access to data, and gives priority to data protection. This is the study to learn whether it supports the legal and ethical criteria of the proposed project. To avoid the major problems in the production and execution of the project defining the specifications that must be addressed at subsequent stages of the process as far as possible.

2.4 Operational Feasibility

The framework integrates design-independent features such as automation, and no manual maintenance or troubleshooting necessary. Operational feasibility depends on the training data which is collected and used to train the DL model. The accuracy of the training data depends on the scale, quality and parameters.

2.5 Scheduling Feasibility

Creation of the project took place in a timely manner by knowing project time schedules and maintaining a clear timetable for project growth.

3. LITERATURE SURVEY

The human eye is considered a big part of the organs as it serves as the source of sight session. The retina of the eye is the most important and critical part of the system of human vision. Glaucoma is a severe condition that causes blindness and affects the retina. According to the report reported in "https:/www.ncbi.nlm.nih.gov/pubmed/24974815," it is estimated that glaucoma will impact a population of about 76.0 million by 2020 extending to 111.8 million by 2040. Glaucoma typically does not show any signs or symptoms until it reaches an advanced stage. The injury is severe and permanent by the time it is identified, with the damage done by the optic nerve resulting in decreased vision due to a loss of about 40 percent of axons. However, if it is detected too early, it might be possible to postpone the vision loss caused by glaucoma. For the most part, the causes of glaucoma also are related to the production of Intraocular Pressure (IOP) in the eyes that result from blockage in the drainage flow of intraocular fluid. The increased IOP damage caused to the optic nerve which carries visual sensory information from the eye to the brain. Harm caused by the perception capacity of optic nerve fiber and the identification of artifacts that can lead to blindness.

Unlike existing methods, in which features were handcrafted from the optical disk, CNN will automatically extract the features from raw images in our method, which will eventually be fed to the classifier. The classifier then classifies the images into their respective labels. The manual analysis of the ophthalmic images is a time-consuming process. The accuracy often varies with the professional differences in expertise and skill. Our solution offers much better responses to the Glaucoma perception disadvantages that require manual processes. Most proven detection techniques involve either feature selection or very precise measurement of geometric ONH structures, which is CDR. Using automation, the phases of identification, examination, diagnosis, treatment time, and the associated risk reduction can be made effective. It has been proven useful for Convolutional Neural Networks (CNNs) and other DL (Deep Learning) systems.

Septiarini, Harjoko et al. considered failure of the Retinal Nerve Fiber Layer (RNFL) as a major parameter for the detection of glaucoma which is not very normal. The fundus image is extracted from the co-occurrence gray level matrix for feature extraction, so that the ONH (Optic Nerve Head) area is removed and the rest of the RNFL is divided into ISNT-based sub-sectors.

Rahul Sarma et al. used support vector machines (SVM), which were efficient in the memory. Also, SVMs are observed to be highly effective when handling high dimensional spaces. Phase knowledge tends to be amiss, because the Radon Transform does not retain it. In addition, some of the information found in the image is lost by used projections. RT further increases difficulty in mathematics and introduces error. They have reached 98.8 per cent and 95 per cent classification accuracy.

Zhao, Zuanlin Chen et al. introduced a mix of unsupervised learning for feature representation purposes, as well as supervised CDR regression learning. Using MFPPNet, which employs 3 dense connectivity blocks along with pyramid pooling, extraction and representation of the function is achieved. Dataset used for research consists of 934 images from 443 clinical samples, validation carried out on an ORIGA dataset with 0.90 AUC.

Khairina, MKom et al. took statistical characteristics for classification, such as mean, entropy and 3rd moment. They also considered smoothness, uniformity and standard deviation. They extracted these features to perform classification and fed them to KNN classifiers. The dataset they used includes 84 images with 95.24 per cent accuracy.

With Pavithra G. Et al. suggested a device that can be easily implemented on hardware kits which, in effect, can be directly linked to the optical instruments and predicted along with the diagnostics. Using histogram equalization, they process the image, find the ROI and finally estimate the Cup-to-Disk ratio, based on which prediction is based. No particular reference was made to dataset or experimental results.

The authors Atheesan et al. achieved optical disc segmentation by red channel analysis, using vasculature observations present in the retinal region. Clustering is first implemented using Simple Linear Iterative Clustering (SLIC) followed by the k-Means Clustering algorithm along with the edge detection filter Gabor. Predictions are made using CDR. The dataset used consists of 100 images with an F-score value of 96%.

Guangzhou An, Omodaka et al. used parametric region inferences swept by the curve characterizing receiver operations (AUC) with a 10-fold cross validation on a random-forest classifier. This RF was derived from color fundus images, maps of RNFL thickness, macular maps of GCC showing thickness, maps of discs showing divergence in maps of RNFL and GCC. They implemented 19-layered VGG19 CNN architecture. They have used a 357 OCT scan image dataset with an estimated accuracy of 0.958 in AUC cross validation.

3.1 About Artificial Intelligence

Artificial Intelligence is a computer science branch which aims at creating smart machines. It has become an integral part of the industrial technology.

Artificial intelligence-related work is highly scientific, and specialized. Artificial intelligence's core problems include programming machines for certain characteristics, such as:

- Knowledge
- Reasoning
- Problem solving
- Perception
- Learning
- Planning
- Ability to manipulate and move objects

A facet that's central of scientific studies are manufacturing this is certainly software applications. Devices can act and respond as people frequently only when they usually have numerous knowledge this is really world-related. To make use of information manufacturing, artificial intelligence needs usage of objects, categories, properties, and connections between them. Initiating common sense, logic and also the capability to solve problems in computers is a tough task.

3.2 About Machine Learning

Machine learning is a feature that's crucial besides AI. Discovering without supervision of any sort requires the capacity to understand patterns in input resources, while learning with proper guidance includes classification and regression that is numerical. Classification describes the group to which an item belongs, and regression handles getting an assortment of numerical feedback or result examples, hence finding features that enable the generation of proper outputs from inputs being particular. Mathematical study of and output of machine understanding formulas actually branch which well-defined of computer system research often referred to as computational learning theory.

3.3 About Deep Learning

Deep learning is one of the branches of machine learning. It is an area focused on learning by analysing machine algorithms, and developing on its own. While machine learning uses simpler concepts, profound learning works with artificial neural networks, designed to mimic how people think and learn. Neural networks had been constrained by computational resources until recently, and were thus constrained in complexity.

Profoundly discovering architectures particularly deep neural systems, deep-belief sites, recurrent neural networks and convolutional neural companies had been placed on areas computer which including, message recognition, normal language processing, sound recognition, social network filtering, device interpretation, bioinformatics, product design, medical image analysis, content assessment and game programs.

“Artificial neural networks” is motivated by the processing of information therefore the circulation of communication nodes within biological systems. ANN's have distinct mind which is similar to the biological one. Particularly, neural networks are usually static and symbolic, whereas mental performance which biological fluid (plastic) and similar to most living organisms. However, advances in big data analytics have allowed for larger, sophisticated neural networks, allowing computers to analyse, learn and respond faster than humans to complex situations. Deep learning has assisted in classifying images, translating languages, understanding voice. It can be used to solve any problem regarding pattern recognition and without human intervention.

3.3.1 Working of a Deep Learning Model

Deep learning uses architectures of the neural network and that's why deep discovering designs are often referred to as deep neural networks. The phrase "deep" usually refers to the number of levels concealed in the network. Standard neural networks make up only 2-3 concealed layers, while deep systems might have up to 150. Deep learning models are trained utilizing huge units of labelled data and neural system architectures that understand functions directly from the data without calling for removal which manual.

One of the more common types of deep-neural sites is called neural convolution (CNN or ConvNet). A CNN integrates learned features with feedback data, and utilizes convolution that is 2D, causing the structure perfect for 2D data handling, particularly photos.

CNNs take away the need for manual extraction of this factor, which means you do not have to recognize the features to classify pictures. CNN runs by extracting functions from the images. The features being relevant maybe not pre-trained; whilst network teaches around pair of pictures, these are generally taught. This automated extraction of the function makes discovering that is deep highly accurate for computer system vision tasks including category of objects.

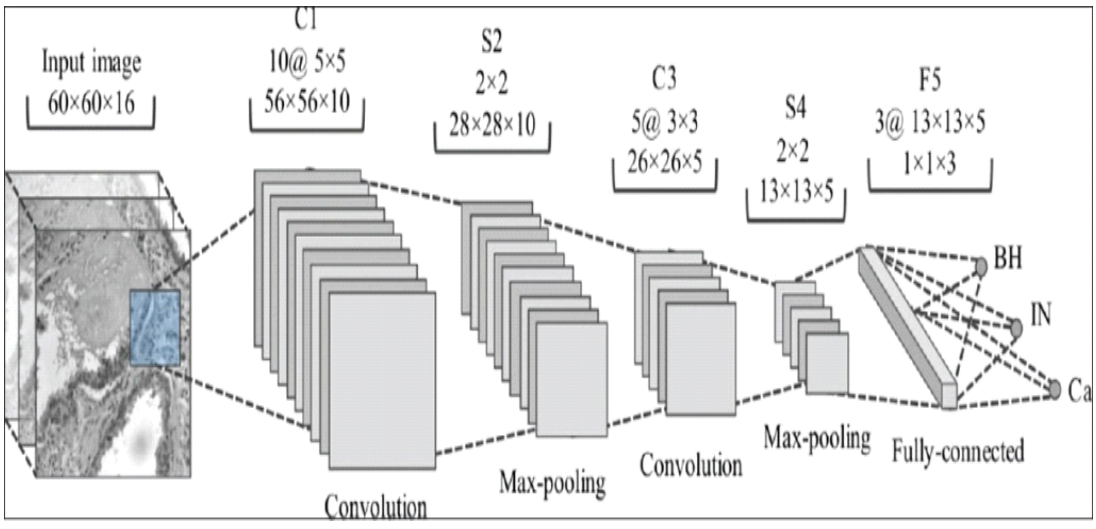


Fig3.3.1 Convolutional Neural Network

3.3.2 Types of Deep Learning

Deep learning can then be described as neural networks in one of four fundamental network architectures with a large number of parameters and layers:

- Unsupervised Pre-trained Networks
- Convolutional Neural Networks
- Recurrent Neural Networks
- Recursive Neural Networks

Unsupervised Pre-trained Networks:

Education feed-forward that is deep companies is hard in objective purpose as a result of regional optima, and because complex models tend to be susceptible to overfitting. Unattended pre-training initializes a discriminative neural network from one that's been trained utilizing an unattended criterion, such as for instance a deep belief network or perhaps a deep auto encoder. Often this process can help with both optimizations therefore the problems that tend to be overfitting.

Convolutional Neural Networks:

A Convolutional Neural Network is basically a neural network broadened or expanded by using shared loads over the space. CNN was created to recognize photos by giving convolutions which are inward to look at edges of an entity this is certainly acknowledged on the picture. A ConvNet's architecture is comparable to compared to the mind that is individual structure of Neurons, which was affected by the “Visual Cortex Organization” of the brain.

Within a restricted part of the visual area given that Receptive Field, specific neurons react to stimuli only. A selection of these areas overlaps to cover the entire visual environment. Through the use of filters which can be appropriate ConvNet has the capacity to capture the Spatial and Temporal dependencies in a picture with success. The design does a much better fit into the image dataset because of the reduced total of how many variables included additionally, reusing the weights of the network. The system is trained to better understand the image’s sophistication, in short.

Recurrent Neural Networks:

A Recurrent Neural system is essentially a regular neural system that's been extended eventually by providing edges that feed to the next step in identical time rather than

into the next layer. In sound signals or even a text, RNN was created to recognize sequences.

The definition of "recurrent neural system" is used indiscriminately to mention two large classes of networks having a comparable general structure, in which one is finite impulse and the other is infinite impulse. Both network groups represent temporal behaviour that's dynamic.

An impulse that is finite network is really a directed acyclic graph which can be unrolled and changed by a solely feed-forward neural network, whereas an infinite impulse recurrent system is a directional cyclic graph that cannot be unrolled.

Recursive Neural Networks:

A Recursive Neural Network is more as a network that's hierarchical where the input series will not obviously have a time dimension but the information must certainly be processed hierarchically inside a tree style. For example, RNNs have-been effective in learning series and tree structures in the processing of all-natural language, primarily phrasing and sentencing representations that are constant on word embedding. RNNs had been first implemented in order to learn about distributed representations which can be conceptual, particularly rational ideas. Since the 1990s, models and frameworks that are general developed in additional works.

The 10 methods below may be put on most of these architectures.

- Back-Propagation
- Stochastic Gradient Descent
- Learning Rate Decay
- Dropout
- Max Pooling
- Batch Normalization
- Long Short-Term Memory:
- Skip-gram
- Continuous Bag of Words
- Transfer Learning

3.4 Applications of Deep Learning

Deep Learning is a diversified field and its applications are as follows:

- **Automatic speech recognition**

Large-scale automatic speech recognition may be the first efficient, and most convincing, example of deep learning. LSTM RNNs will find out "Very Deep Learning" having multi-second intervals that involves message events divided by a large numbers of discrete time stages, where one-time step corresponds to around 10 ms. LSTM with forgotten gates competes with old-fashioned target recognizers on specific tasks.

- **Visual art processing**

The applying this is certainly developing of discovering processes to numerous aesthetic arts tasks is closely for this development made in picture recognition. a) identifying the style span of certain painting, b) Neural type transfer-capturing the type of a given artwork and putting it on to an arbitrary picture or video in a visually pleasing fashion, and c) producing striking imagery cantered on arbitrary fields of visual data for instance.

- **Natural language processing**

Neural networks have been made use of since the very early 2000s to utilize the language models. LSTM assisted to develop the device translation additionally the simulation of languages. Most recent innovations generalize word embedding to embedding of sentence. Google Translate (GT) works on the massive memory which is long-term, end-to-end. Google Neural Machine Translation (GNMT) utilizes a device which is example-based strategy where the system "learns from an incredible number of instances" and translates "complete sentences at a time, rather than bits." Google Translate aids more than one hundred languages. The network encodes the "semantics associated with sentence, without merely translations which are memorizing phrase to phrase." GT uses English as an intermediary between many sets of languages.

- **Drug discovery and toxicology**

A huge percentage of candidate drugs don't gain endorsement by regulators. These types of failures are due to inadequate effectiveness (on-target), unintended interaction

(off-target impacts) or unexpected toxic results. Work has examined the application of deep discovering in nutritional elements, home goods and medications to predict the objectives which are biomolecular off-targets, and poisonous aftereffects of environmental chemical substances. AtomNet is a deep learning framework for reasonable medication design, predicated on a construction. With disease targets like the Ebola virus and sclerosis and many more AtomNet had been able to regularly predict novel candidate biomolecules. In 2019 generative communities that are neural used to produce molecules which were experimentally tested into mice all the way through.

- **Customer relationship management**

Deep reinforcement learning, described in terms of RFM variables, has been used to estimate the importance of potential direct marketing behaviour. It was shown that the estimated value function has a natural meaning as a lifetime value for the customers.

- **Recommendation systems**

Recommendation systems have used profound learning to derive functional features from content-based music and journal recommendations for a latent factor model. Multiview deep learning was implemented to learn tastes of users from different domains. The model uses a hybrid approach focused on collaboration and information, and improves suggestions for different tasks.

- **Bioinformatics**

In bioinformatics an automobile encoder ANN had been accustomed anticipate gene ontology, annotations, and interactions with gene-function. Deep learning has been used in health informatics to anticipate quality which rest on wearables information and forecasts of health risks from electronic wellness record information. Deep learning has also shown efficacy in health.

- **Medical Image Analysis**

Deep learning has been shown to deliver competitive results in medical applications such as classification of cancer cells, identification of lesions, segmentation of organs and improvement of the image.

- **Mobile advertising**

It is often difficult to identify the right user audience for mobile ads, as many data points have to be identified and evaluated before any ad server can construct a target segment and use it in ad serving. Deep learning has been used to interpret massive, multi-dimensional ad datasets. Throughout the process of Internet advertisement request / serve / click several data points are obtained. This knowledge can form the machine learning basis for improving ad selection.

- **Image restoration**

Deep learning is applied effortlessly to reverse problems like denoising, super-resolution, artwork, and colouring of movies. These programs consist of discovering techniques particularly "Shrinkage Fields for effective Image Restoration," which trains on a picture dataset, and Deep Image Prior, which trains on a restored image.

- **Financial fraud detection**

Deep learning for the detection of financial fraud and anti-money laundering is being successfully implemented. "Advanced anti-money laundering detection systems can identify and recognize data relationships and similarities and learn to identify anomalies or distinguish and forecast particular incidents further down the road." The approach leverages both supervised learning methods, such as classifying suspicious transactions, and unsupervised learning, such as identifying anomalies.

- **Military**

The US Department of defence applied deep learning through observation to train robots in new tasks.

4. ALGORITHM DESCRIPTION

4.1 Process of building machine learning model

Key part of the project is building a model of a neural network using the python language tensorflow-gpu kit.

General Machine Learning Modelling Process:

Data collection: This will be a very step that is critical since the quality and volume of information you get will straight influence exactly how effective your predictive model will soon be.

- Organizing data: where we load our information to a suitable location and prepare them for use in our machine training which learning.
- Model creation: there are numerous of models that researchers and information researchers have actually produced over time. Others are particularly appropriate image data, some for sequences (such as for example text, or music), some for numerical information, others for text-based information.
- Training Model with Organized Data: we'll make use of our data within step to build up our model's capacity to predict incrementally
- Deploy Trained Model: This is the final step.

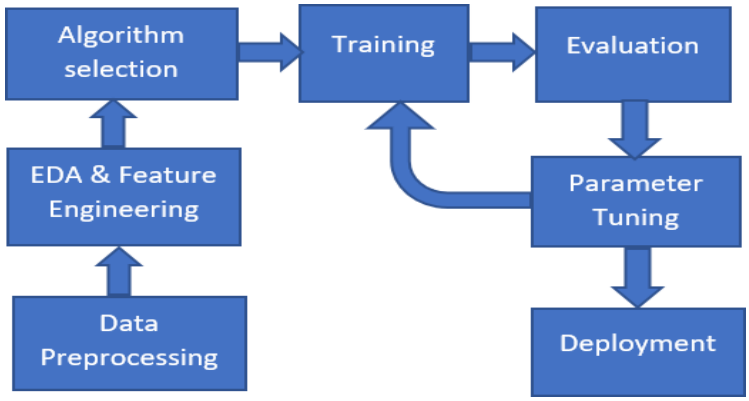


Fig 4.1.1 Process of building a machine learning model

Below Flowchart in Fig, 4.1.2 describes the flow of our ranking process. First phase is to gather data. The dataset is then processed and organized into separate files. The next step is to pre-process the data. The model is then constructed using the training data to correctly identify the images to what form they belong. The model is then tested using a reference dataset.

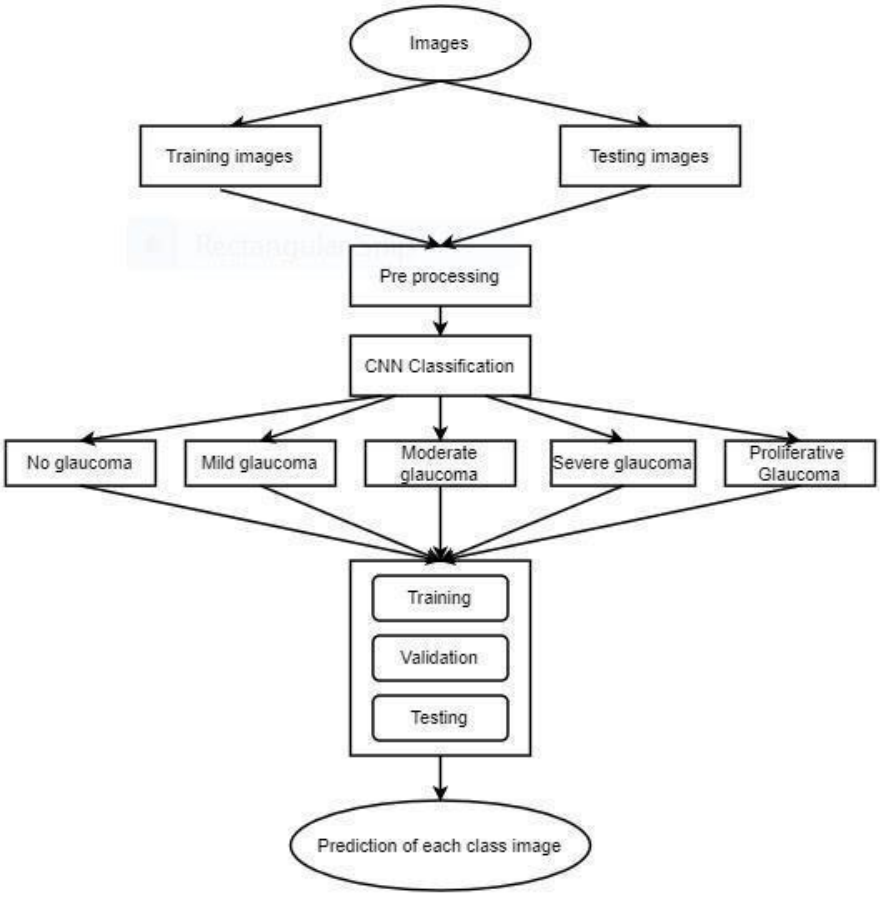


Fig 4.1.2 Flow chart of the proposed system

4.2 Convolutional Neural Networks

A model needs to understand from an experience and find out how to conduct tasks, such as predicting outcomes or classifying types from the various formats of data sources provided to it, and using deep learning techniques, this technique can be most effectively modified. This involves various kinds of structures each containing different deep layers of features or neurons or nodes. The present layer of design which imparts association with the past layer, accepts its yield as the learning contribution to completing additional operation. Convolutional Neural Networks have a large engineering framework that includes layers of neurons that are used intensively in the field of PC vision for various purposes. CNN consists of different hidden layers whose functionalities contrast with each other. A CNN model is a combination of two or three methods, one distinguishing the information characteristics and the other characterizing the information depending on the understanding being prepared. Classification of an input is done after it eliminates the characteristics from the layer where all the neurons

communicate. Extraction of defined characteristics prompts more detailed results. The CNNs with consist of neurons with learnable weights and biases. Some inputs are obtained by each neuron, performs a product and optionally employs a non-linearity to it. One differentiable function score remains expressed within the network: from the raw image pixels using one end to class scores on the other side. And so, they continue to have a loss function on the last connected (totally linked) layer (e.g. SVM or Softmax) and all the methods built for learning regular neural networks nonetheless apply.

Model Architecture: A CNN contains usually the layer that is convolutional, the pooling layer (optional), and the production layer.

Prediction: A feature that's key of sites is an iterative learning apparatus that introduces records (rows) towards the system one-by-one and makes modifications to the weights linked to the input values each and every time. The period starts once again after every one of the complete situations are addressed.

Training: An accumulation proper network behaviour inputs (denoted by p) and target outputs (denoted by t) are essential for the training stage. A neural network training process involves adjusting the weight and bias values of the network to maximize network efficiency as specified by network experience.

Launching Model:

After selection of training algorithms, we start training the model on organized data. Output of this process gives us model performance.

Evaluation: Accuracy of the model here is taken into consideration by creating a uncertainty matrix of True Positives and True Negatives.

Training a CNN is a complex activity from the outset. It needs a large amount of labelled knowledge-which can be a challenging problem. In addition, the required computational resources are huge in size. However, a workaround for training a CNN without any planning consists of fine-tuning an existing CNN which has been trained using a comparatively larger labelled dataset from another program (e.g. ImageNet).

Transfer Learning is the method of overcoming the paradigm of independent learning

and applying the information gained for one use case to other similar ones. Essentially, it is using a model that has been designed for some purpose and using it for another purpose. Transfer learning often refers to a pre-trained ANN being modified to perform a new task. Using pre-trained deep CNNs and subsequent fine tuning of network loads applying new labelled images may lead to better yet performance metrics and a reduction that's reflected in time, memory and computational operations training resources. In this approach, CNN's alternative design model is thought out and focused on its exhibition which is ResNet for greater productivity and precision in the need of performance. Figure 4.2.1 displays the planned architecture for the CNN.

This model consists of an input layer with 128 * 128 input size and is followed by convolution layers with different window sizes followed by an activation feature and pool size 3x3 max pooling layers. Convolution2D are filters which specify the number of filters to use, the kernel size is the length and width of the kernel used, padding specifies whether or not to use an additional layer of zeros around the image.

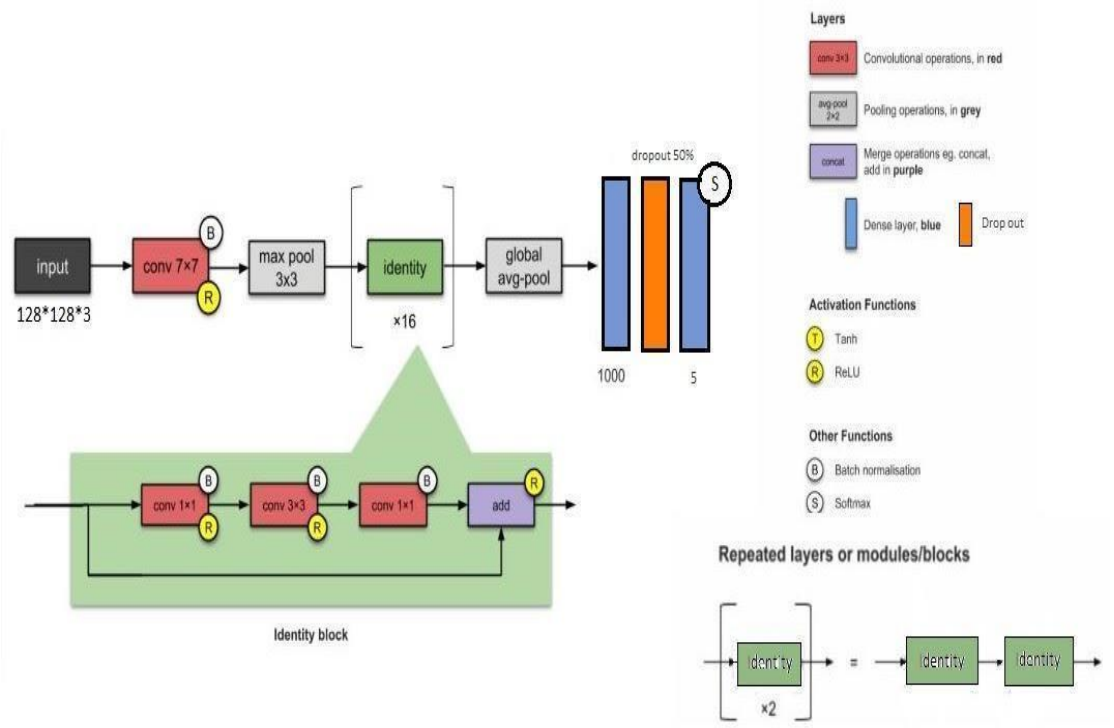


Fig 4.2.1 CNN architecture

We used 2 activation functions here: 1. ReLu, 2. Softmax. These functions help us to determine whether to activate a neuron, or not. Those often help to introduce non-linearity to the neuron production. Max pooling is used by taking the largest element

from the rectified feature map to reduce the spatial volume of the input image.

Batch Standardization is used to normalize each layer's activations by converting the inputs to mean 0 and unit variance. This allows the pattern to be regularised. Then we apply the function Flatten to transform a 2D array into a 1D array. To remove overfitting, we use the Dropout feature to drop 50 per cent neurons. In the classification stage dense layers and Softmax regression are used. The activation function Softmax uses the output neurons to produce probabilities of each type. We used optimizer Adam.

4.3 Working of CNN

CNN image classification takes, procedures and classifies an input image under particular categories (e.g.: dog, pet, tiger, lion). Computers see an input image as a pixel array, and that is dependent on the resolution of image. It shall see h x w x d (h = Height, w = Width, d = Dimension) on the basis of the image quality. E.g.: a picture of 6 x 6 x 3 RGB matrix array (3 means RGB values) and a 4 x 4 x 1 matrix array which grayscale.

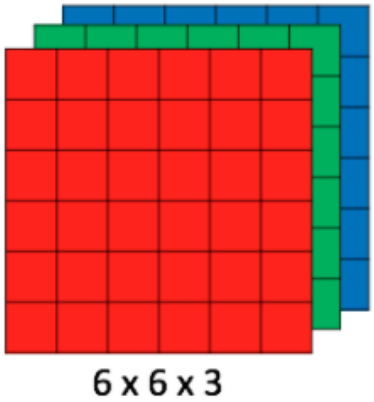


Fig 4.3.1 - Array of RGB Matrix

Theoretically, each input image can undertake a number of convolution levels with filters (kernels), pooling, entirely connected layers and use Softmax to recognize an object with probabilistic values from 0 to 1.

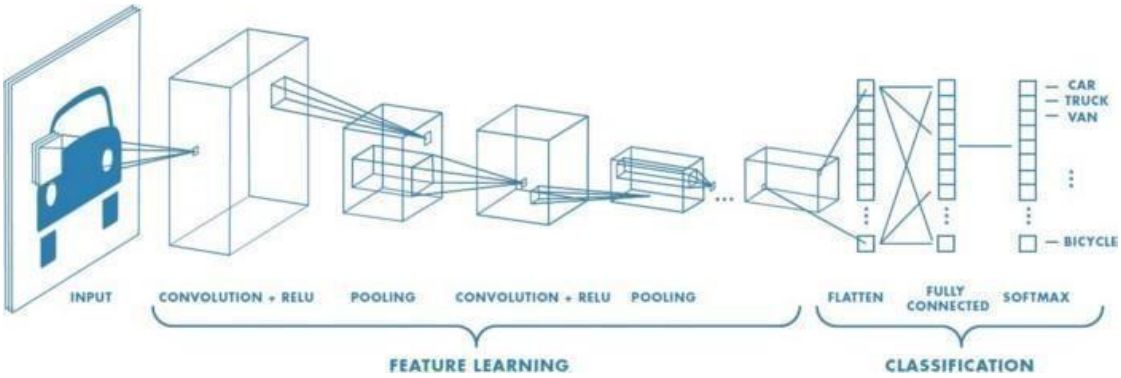


Fig 4.3.2 - Neural network with many convolutional layers

Convolution Layer:

When programming a CNN, the input actually is a form of tensor (picture number) x (picture circumference) x (photo level) x (image depth). The picture is abstracted to a function chart, with kind (number of photos) x (feature chart width) x (function map level) x (feature chart stations) upon moving through the convolutional level. A convolutional level must have listed here attributes within a neural community

- Convolutional kernels, defined by height and width (hyper-parameter).
- Input stations and output channels number (hyper-parameter).
- The Convolution filter depth (the feedback networks) needs to be add up to the number networks (circumference) for the feedback characteristic chart.

Convolutional layers twist the input and transfer the end result which is concise. That is identical to a neuron's reaction to a certain stimulus in the visual cortex. Convolutional neural process processes information only for its receptive field. Although completely connected feed forward neural networks can be employed to learn features and classify data, application of this architecture to pictures just isn't useful. Given the very large input sizes which are associated with images, where each pixel is a certain variable, a huge number of neurons will be sufficient, even in a shallow (opposite to deep) architecture. For example, a completely connected second layer for (small) dimensions 100 x 100 picture has 10,000 weights per neuron. This problem is fixed by the convolution process because it decreases the number of free parameters, allowing the network becoming deeper with less parameters. As an example, irrespective of the picture size, tiling areas of size 5 x 5, each with similar provided loads, just require 25 learnable parameters. This way, by making use of back propagation, it addresses the question of vanishing gradients in training main-stream multi-layer neural networks with several levels.

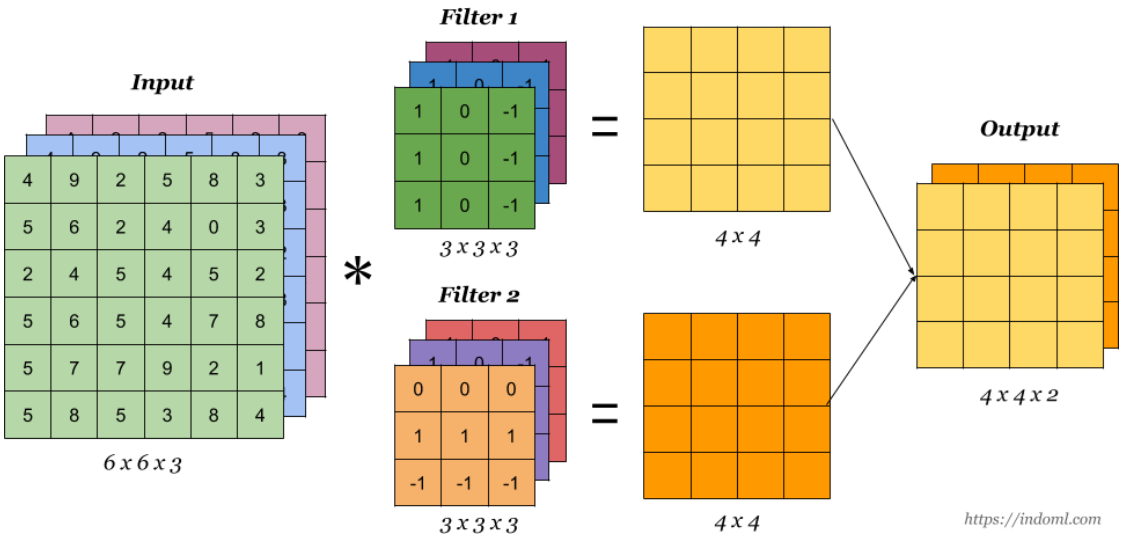


Fig 4.3.3 Convolution Layer

Pooling Layer:

Part of pooling levels will decrease the parameters whenever pictures are way too large. Spatial pooling is also known as subsampling or down sampling which lowers each map's dimensionality but keeps information this is certainly essential. Spatial pooling takes various forms:

- Max Pooling shown in Fig.4.3.4
- Average Pooling shown in Fig.4.3.4
- Sum Pooling
- Stochastic Pooling

Max pooling is taken from the rectified function map with the largest dimension. The average pooling could also take on the largest dimension. Sum of all function map elements is named as amount pooling.

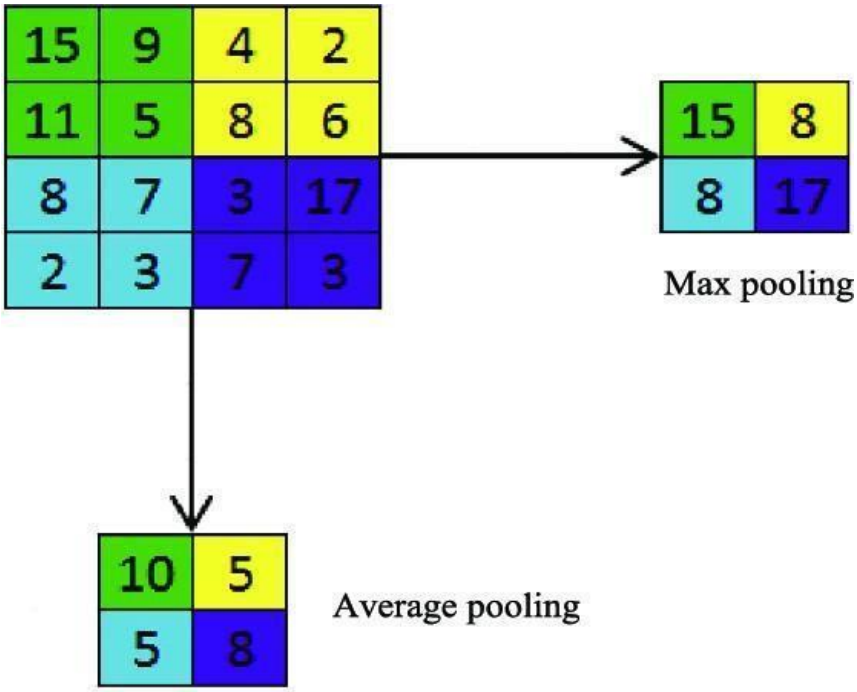


Fig 4.3.4 - Max Pooling and Average Pooling

Activation Function:

For Convolutional Neural Network, activation functions are actually very important to discovering and making sense of something really complicated and non-linear dynamic mapping between inputs and response variable. They reveal our system to properties that are non-linear. Their particular purpose is to change a node's input signal to an output signal in a CNN. Now the output signal is employed as an input for the next stack layer. Specifically, in CNN we perform some amount of input (X) products and their corresponding weights (W) and add f(x) (activation function) to it to get the output of that layer and feed it to the next layer as data. Fig. 4.3.5 shows the activation method performance.

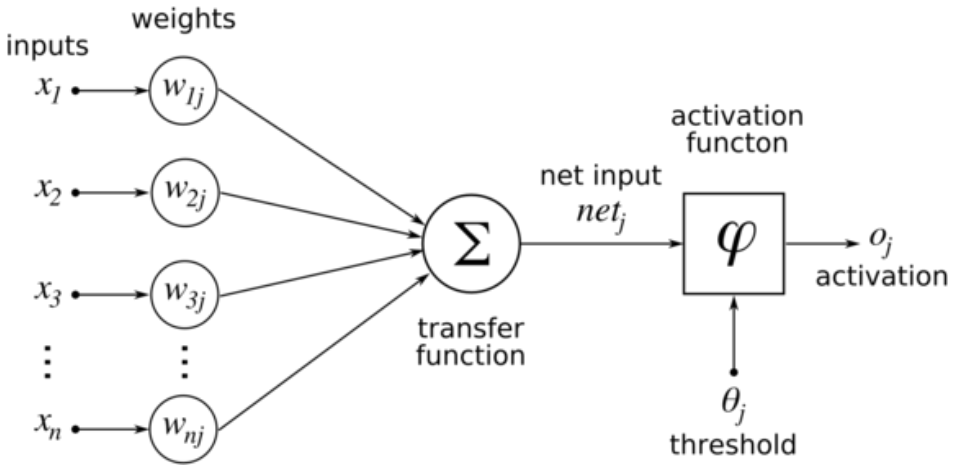


Fig 4.3.5 Activation Function

There are different activation functions as show in below figure Fig.4.3.6

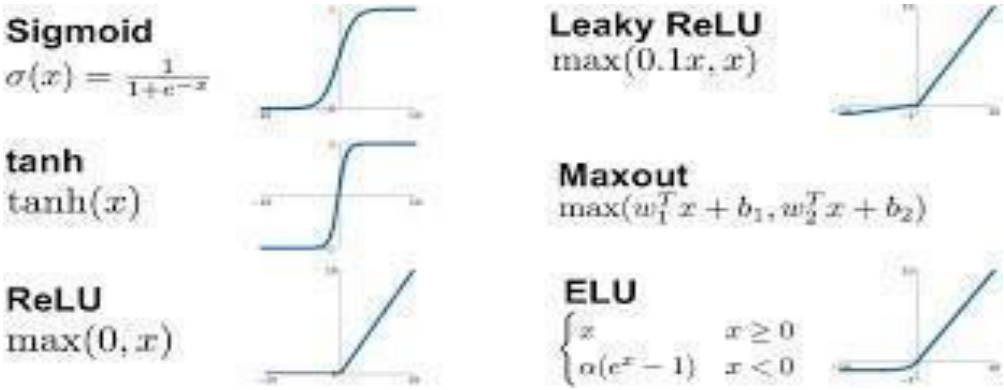


Fig 4.3.6 Different activation functions

Dense Layer:

The name indicates layers in a network layer which are completely (dense) connected by the neurons. Every neuron in a layer receives an input from all of the previous layer's neurons — thus, they are densely connected. Each neuron shall give the next layer exactly one output, an example is shown in the Fig below. 4.3.7 The thick layer is secret layer 2, here.

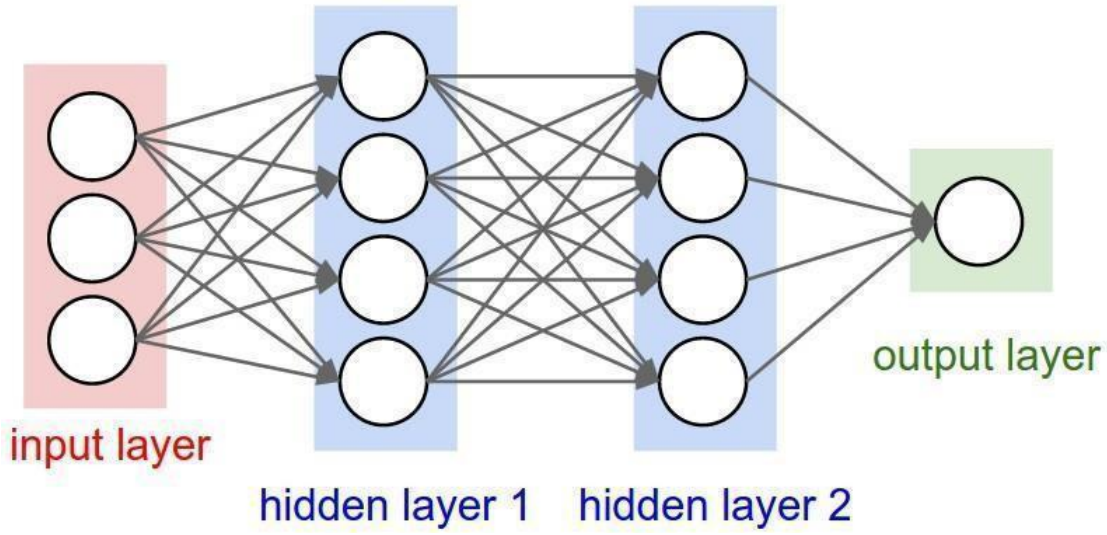


Fig 4.3.7 - Dense Layer

Optimizers:

Optimization algorithms assist us minimize (or optimize) an objective function (Error function) E(x) that is just a mathematical function that relies on the inner learning parameters associated with system which are used to determine the largest values (Y) from the set of predictors (X) utilized in the system. As an example — we call the neural network's Weights (W) and Bias (b) values as its internal learning parameters that are used in processing output values and trained and modified in the direction of optimal result, i.e. minimizing the loss through the network's training stage and also playing a major part in Neural Network Mode training process.

A Model's internal variables play an extremely essential part in training a Model effectively and efficiently, and creating accurate results. That is why we utilize various Optimization techniques and algorithms to update and assess the optimum and correct values of the parameters of such model that impact the learning process and performance of our model.

Different optimizers available are:

AdaGrad — Adaptive Gradient Algorithm. Upgradation rule of it is:

$$v_t^w = v_{t-1}^w + (\nabla w_t)^2$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t^w + \epsilon}} * \nabla w_t$$

$$v_t^b = v_{t-1}^b + (\nabla b_t)^2$$
$$b_{t+1} = b_t - \frac{\eta}{\sqrt{v_t^b + \epsilon}} * \nabla b_t$$

RMSProp — Root Mean Square Propagation. Upgradation rule of it is:

$$v_t^w = \beta * v_{t-1}^w + (1-\beta)(\nabla w_t)^2$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t^w + \epsilon}} * \nabla w_t$$

$$v_t^b = \beta * v_{t-1}^b + (1-\beta)(\nabla b_t)^2$$
$$b_{t+1} = b_t - \frac{\eta}{\sqrt{v_t^b + \epsilon}} * \nabla b_t$$

Adam — Adaptive Moment Estimation. Upgradation rule of it is:

$$m_t = \beta_1 * m_{t-1} + (1-\beta_1) * \nabla w_t$$
$$v_t = \beta_2 * v_{t-1} + (1-\beta_2) * (\nabla w_t)^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$

5. SYSTEM ANALYSIS

5.1 System Requirements

5.1.1 Hardware Requirements

- Microsoft Windows 7/8/10 (32-bit or 64-bit).
- 4 GB RAM minimum, 8 GB RAM recommended.
- Nvidia’s 4 GB graphics card minimum

5.1.2 Functional Requirements

Functional requirements of this project could be considered as

- Actual accuracy should be in approximation with the achieved accuracy of the model.
- User should be able to check the health status of his/her eye.
- Model to decide whether the eye is infected with glaucoma or not when provided with an input image.

5.1.3 Non-Functional Requirements

- Non-functional specifications identify standards that define parameters that can be used to assess a system's operation instead of actual behaviours. It should be in contrast to functional requirements which define specific behaviours or functions.
- Accuracy: To predict the correct result, the method needs to be accurate
- Precision: the result should not be uncertain.
- Usability: The program should be made easy to be used by the user. It must provide the user with the best experience when using and handling an application.
- Stability: The device must work continuously over time without failure or breakdown
- Scalability: The system should meet that consumer and surrounding environment needs • Efficiency: power, performance and response times should be managed by the system.
- Performance: The device should be functioning quickly and accurately.

5.1.4 Dependencies

5.1.4.1 Python 3.6.5

Python is among the development that is high-level which is commonly used for numerous basic purposes. It was developed in 1991 by Guido van Rossum, and further developed by the Python Software Foundation. Primarily, it had been made emphasizing readability of code. Its syntax assists programmers in conveying principles taking less lines of code. Python lets you run faster and implement systems more effectively. There are two primary versions of Python— Python 2 and Python 3. Both are used for very popular and specific.

5.1.4.2 Jupyter Notebook

The Jupyter Notebook is an open-source application which can be used to build and distribute live code, calculations, visualizations, and text documents.

5.1.4.3 Google Colab

Colaboratory (known as Colab) is a free Jupyter notebook system running in the cloud and stored on Google Drive the notebooks. It allows notebook development with kernels Python 2, Python 3, IR, Swift, R, and Julia. Google's processing units for the tensor also work on Colab with Julia.

5.1.4.4 TensorFlow-GPU

TensorFlow is known to be a second-generation framework for Google Brain. On 11 February 2017 version 1.0.0 was released. The Reference Implementation works on single computer systems, whereas TensorFlow are performed on multiple CPUs and GPUs (with optional CUDA and SYCL extensions on layouts devices that are processing general-purpose processing). TensorFlow is present on platforms for 64 little bit Linux, macOS, Microsoft windows and mobile devices Android ios.

Its design that's powerful enables to-be quickly distributed across a number of systems, and desktops. Also, across host clusters to devices which can be mobile edge devices.

TensorFlow computations tend to be represented as graphs of stateful dataflow. The term TensorFlow originates from the operations done by these networks that are neural that are on multidimensional information arrays. These arrays are known as tensors. In 2016, Jeff Dean reported during the Google I / O meeting that 1,500 GitHub repositories detailed TensorFlow, of which only 5 were Google-based.

If a TensorFlow operation has implementations for both CPU and GPU, priority will be given to the GPU devices when the operation is allocated to a device. Matmul for instance has both the CPU and the GPU kernels. To run matmul, you pick gpu:0 on a machine with devices cpu:0 and gpu:0.

5.2 Types of Users

5.2.1 Administrator

A CLI (Command-Line Interface) with Python is provided to the administrator for this Deep Learning framework to provide the model with input images. The algorithm analyses the image and predicts the degree of Glaucoma if it is affected.
For further performance study, the admin must also log all the findings and the related predictive tests.

5.3 Modules

5.3.1 Dataset preparation and pre-processing

5.3.1.1 Data collection

The high-resolution Funduscopic photos of the retina taken from the Kaggle website of the data science group (https:/www.kaggle.com/) was included in the datasets used for the method. Since our project is in the field of biomedicine and needs accurate tests, we chose a big dataset. Each image is rated to a scale of 0 to 4 for the severity of the disease.

5.3.1.2 Data Visualization

A large amount of knowledge in graphical form is easier to understand and interpret. The following figure shows the distribution of images in the dataset by diagnostic class.

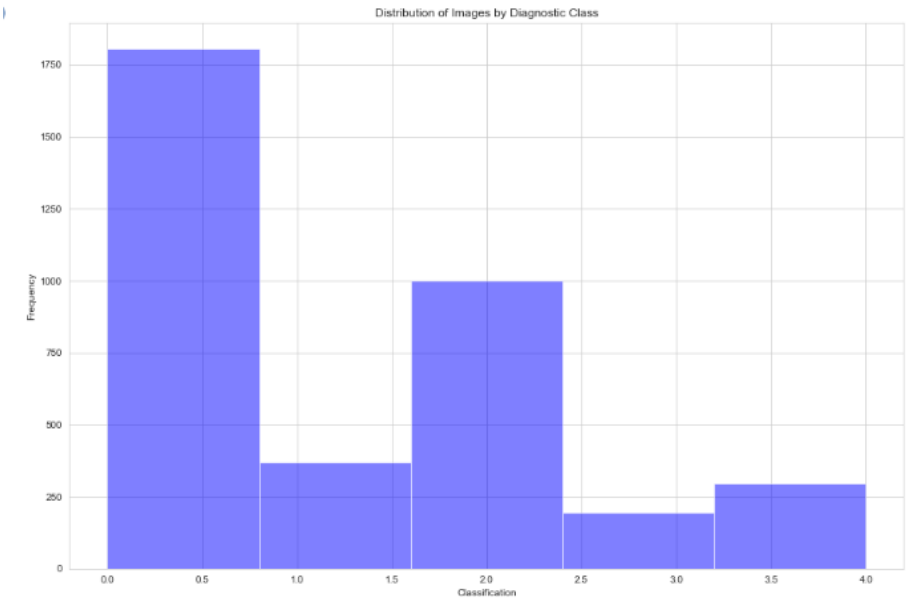


Figure 5.3.1 - Distribution of images by diagnostic class

5.3.1.3 Data pre-processing

For fundus photography performed at extremely distinct lighting rates, patients of varying age groups, gender and nationality are considered, and photographs are collected to form the dataset. Considering these varied data leads to changes in the dimensions of the pixels within the images that could produce pointless variations. To counter this, methods of image pre-processing such as image cropping, color normalization, measurements rendering, data increase etc. were performed on the images. To allow pre-processing, the data is converted to a hierarchical data format from an image format, accompanied by details increase, later trained accordingly. Images were increased in real-time for improvisation in network localization capacity and also to help reduce overfitting. Techniques such as horizontal flip, vertical flip, zoom in of 20 percent were performed to render the data standardized as the number of images were not distributed equitably throughout the dataset for each sort. Data increase makes the model more resistant to minor variations, thereby preventing overfitting of the model. The dataset was formatted to 128 * 128 pixels which allowed the retention of intricate recognizable features.

5.3.2 Dataset splitting:

The dataset contains 5,590 images. All images are divided into collections of cars, measures, and validations. The known outputs are stored inside the training dataset on which the model relies. Later on, the data is generalized to other data based on the set. The validation set is used to evaluate a given model but this is for standard evaluation. So the model absorbs this knowledge once in a while, but never "Know" from it. We use the validation gathering results and update hyper parameters of low rates. We have the test dataset to try out our model's prediction. This is only used once a model is fully trained (using the sets of trains and validations). High quality training dataset is useful to train the classifier model but not all images are labelled in our data. We have found a total of 3,362 images that are labelled for this purpose. We considered two reasons splitting our dataset into sets of work, validation and checking. First, the total number of samples in our details and second the actual model that we are preparing.

5.3.3 Modelling

5.3.3.1 Model training

The main intent behind proposing this solution is the classification of the images and get the severity scale to what degree the eye has been affected by the Glaucoma disease. A robust and enormous training is necessary to obtain such composite classification of pictures. Therefore, our solution is to use profoundly convolutional neural networks to carry out training and research. Our main concern is to apprehend the derez characteristics of the disease effectively based on deep CNN. On CNN, initial training was introduced before significant progress was made. This was a time-saving process performed without hesitation to produce a comparatively speedy classification test. The network was then trained for a further 5 epochs after 2 epochs of training on the images. We used a residual learning system to facilitate the significantly deeper network training. Neural networks were seriously overfitted. To solve this problem, we stopped training the neural network early on before the training dataset was overfitted and finally improved the generalization of deep neural networks.

5.3.3.2 Model evaluation and testing:

Model Evaluation plays a major role in the product development process. Evaluation helps to find the best model that speaks to our information, and how well the model picked will work later. To improve the model, we modified the hyper-parameters of the model to try to improve the precision and also to look at the matrix of uncertainty and try to increase the number of true positives and true negatives. When planning the model, we apply the prepared equivalent model to anticipate the using the test details, i.e. the unseen data. We built up a uncertainty matrix when this is completed, this shows to us how well our model is being prepared. A misunderstanding matrix has four measurements, 'True Positives,' 'real Negatives,' 'False Positives,' and 'False downsides.' This confusion matrix generated includes information about the classifications predicted and those that are real with a category system. We are inclined to obtain more qualities in the True negatives and true positives in order to obtain a model that is slowly correct. We have followed the metric of accuracy in order to test the implementation of our CNN classifier.

6. SYSTEM DESIGN

6.1 UML Diagrams Introduction

Abbreviates UML for Unified Language Modelling. It is a visually reflective form of architectural design and implementation of complex software solutions. UML makes use of graphic notation techniques to construct visual models for software-persistent systems.

- There is a misconception that UML is a method of software development but UML is just a basic modelling language that helps to better understand how to construct a system. UML specification clarified the mechanism as follows:
- Provides instructions on the systematic flow of the tasks of a project, specifies what objects are to be made, manages the errands of individual designers and the community as a whole, and offers a benchmark for tracking and measuring the products and exercises of an undertaking.

6.2 Use case Diagram

6.2.1 Definition

Use case diagrams which include actors, use cases and their relationships. The diagram is used to design an application's system / subsystem. A case diagram of solitary use shows a specific device use. Use case diagrams to collect the specifications. These involve internal and external factors that are design specifications primarily. Using these diagrams, a system can perform a high level of requirement analysis. The functionalities are encapsulated in use cases at a stage where the specifications of a program are analysed.

Using case diagrams to determine the flow of events in a system as defined. However, the use case diagram never demonstrates how they are handled. A well-organized case of usage often depicts the precondition, post state and exceptions. These additional components are used when doing the testing to generate test cases.

6.2.2 Use case diagram for Glaucoma detection

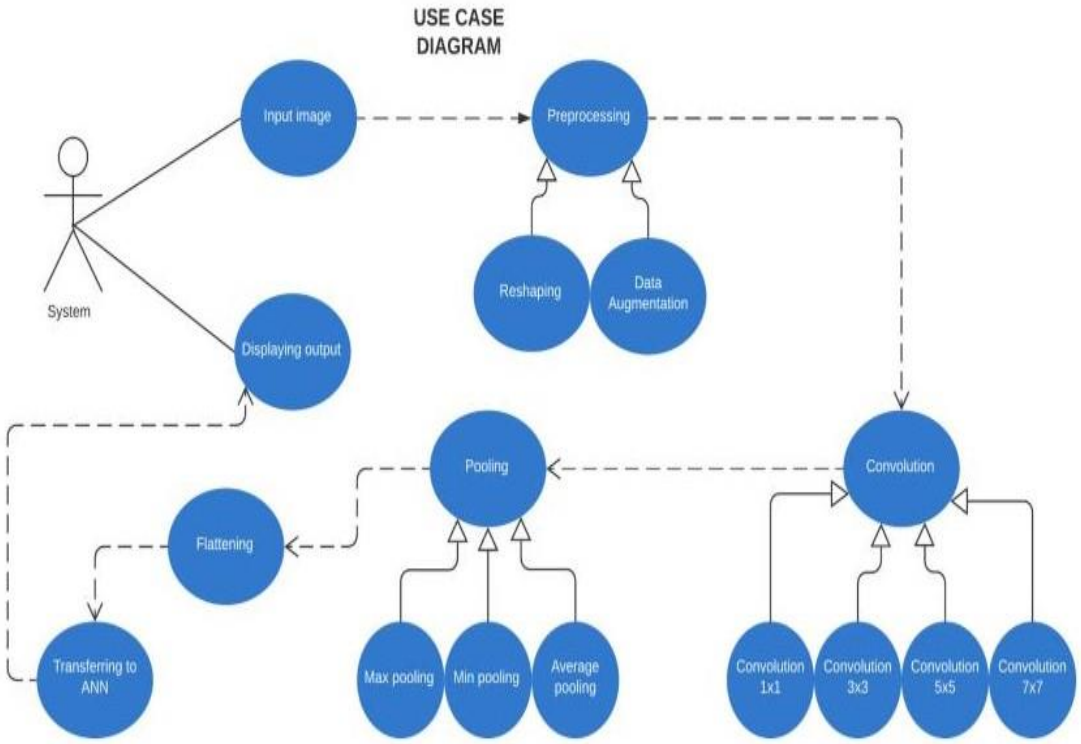


Fig:6.2 Use Case Diagram for Glaucoma Detection

Description:

The use case diagram for glaucoma detection contains one actor:

- System

The device has two use cases, input image, and output view. The program is distributed feedback in the form of an image comprising a retinal eye scan. This image is processed by shaping it into the desired angle and then augmented in order to increase the effectiveness. Using convolution, the images with similar qualities are integrated, and then pooling between max pooling, min pooling, average pooling is done according to the pixel brightness.

This picture is then taken for flattening, which transfers to ANN. The machine attempts to identify the picture based on the information it has, and shows the output for glaucoma recognition.

6.3 Class Diagram

6.3.1 Definition

A class diagram is one of the static and structured UML diagrams which project the layout of a system by indicating the classes contained in the system, the attributes linked to that class, the operations (or methods) performed by each class, and the relational interaction between objects.

It's really important to understand the difference between class and object. Classes are things which typically define the different types of objects, i.e. classes are like object constructors, while functional class instances are called objects. They contain the same components as each object is taken to be built from the same set of blueprints.

Class diagram primarily comprises of three major elements:

- Class- A class is an entity prototype capable of inheriting the same properties, relationships, operations, and semics.
- Attribute- A specific class attribute indicates that the class characteristics created by the users of the program are of interest.
- Relationship- Relationship is a characteristic feature that defines the structure and functioning of model elements together by adding semicolons to the diagram.

6.3.2 Class diagram for Glaucoma detection

Description:

The class diagram consists of various classes

Pre-processing-system whose data is to be used as an input to a specific process

1. Image: This is an instance of data for the entire phase

2. Illumination correction(): Delete irregularities in image pixel illumination

3. Roiextraction(): The recognition of sections of fundus scopic images for glaucoma detection purposes

4. Reshaping(): Extracted area shift to ideal shapes

5. Histogramequilization(): To minimize the most rising contrast in the picture CNN-Algorithm that recieves input in the form of image, assigns priority to each aspect of image.

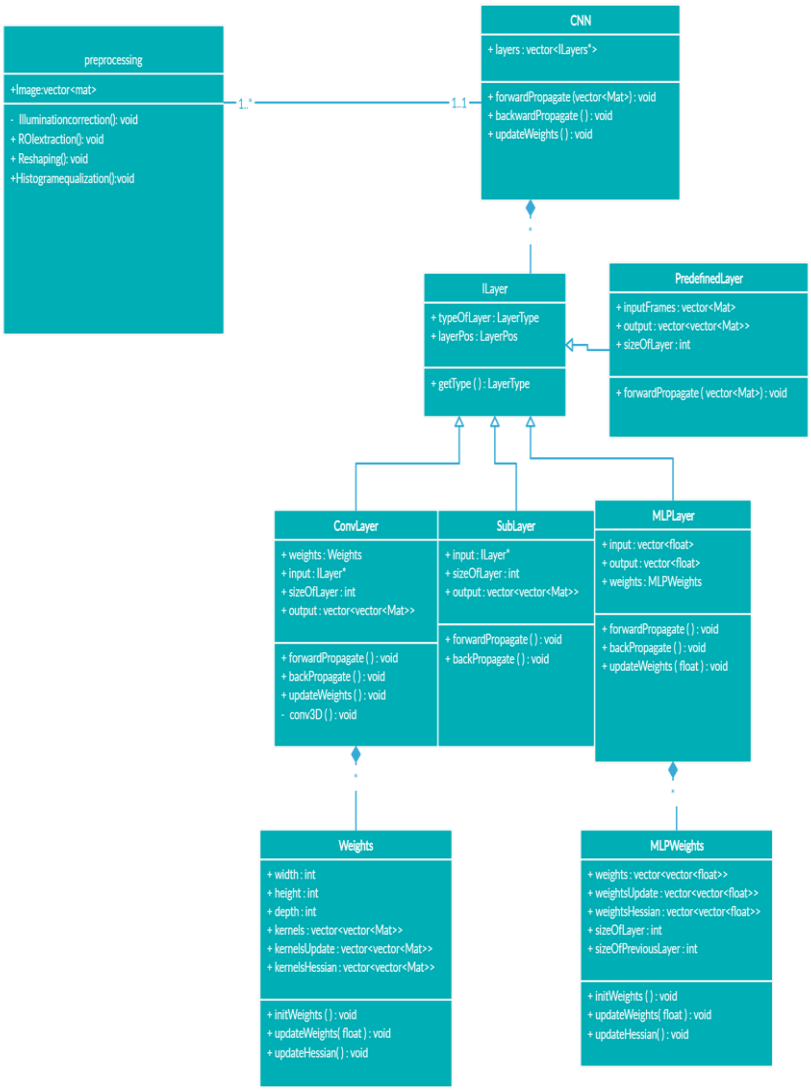


Fig:6.3 Class diagram for glaucoma detection

- forwardpropagate(): obtain output to compare it with real value to find error
- backwardpropagate(): find derivative of error according to each weight to be subtracted from actual weight value
- updateweights(): update weights after propagation

Ilayer-Ilayer stands for layer of data. The CNN input layer is composed of the image data. Usually a 3-D matrix represents that image data. To make a single column this matrix must be reshaped.

For example, if we have an image with 28 x 28 = 784 dimensions, we need to convert it to 784 x 1 even before feeding it into data. Let's assume that we have "n" training examples then the dimensional input should be (784, n).

- getType(): to acquire the type of the layer

- PredefinedLayer- it is used to maintain the dimensions of output just as same as input
- ConvLayer- Rotates the data and transfers the findings to the next sheet. The layer can't learn from just one filter, it learns to use different features in a parallel way with an input taken.
- forwardpropagate(): obtain output to compare it with real value to find error
- backwardpropagate(): find derivative of error according to each weight to be subtracted from actual weight value
- updateweights(): update weights after propagation
- con3D(): rotate input for layers to learn
- SubLayer
- MLPLayer-MLP stands for multilayer perceptron (MLP). It is a class of artificial neural feed forward networks (ANN). It comprises at least three layers of nodes, which are one layer of input, one or more hidden layers, one layer of output. For training the MLP, a supervised method of learning called back propagation is employed. It shows the characteristics of a fully connected layer, because all the perceptrons are interconnected.
- For modern and advanced computer vision activities, MLP is now considered insufficient, because it comes with many drawbacks. Complete parameter counts tend to increase to higher values causing such high dimensional redundancy. As regards spatial data, another drawback of MLP is that it accepts flattened vectors as inputs.
- forwardpropagate(): obtain output to compare it with real value to find error
- backwardpropagate(): find derivative of error according to each weight to be subtracted from actual weight value
- updateweights(): update weights after propagation
- Weights- numeric value that can be regulates or altered in accordance to reduce the error that occurs while backward propagation
- InitWeights(): to accept the weight in the form of numeric integer value
- updateWeight(float): to change the updated weights into float values for accuracy
- updateHessian(): analyzing successive gradient vectors
- MLPWeights- Initializes the weight parameters in order to achieve optimization in the least possible time
- InitWeights(): to accept the weight in the form of numeric integer value

- updateWeight(float): to change the updated weights into float values for accuracy
- updateHessian(): analyzing successive gradient vectors

6.4 Sequence Diagram

6.4.1 Definition

UML interaction diagrams consist primarily of three separate diagrams, i.e. sequence diagrams, pacing diagrams, and diagrams for communication. These diagrams have a very good utility to explain relations between defined parts of the system. Sequence diagrams are commonly used by both developers and readers, as it is the simplest form among the different forms.

Order diagram defines an interaction by reflecting on the order of the exchanged messages along with their corresponding lifeline event specifications.

Messages written with horizontal arrows, with the name of the message written above them, indicate contact. Interrelationship among a system's components is very important from the perspective of implementation and execution.

The process of requesting an item to perform an operation is called a call. There are three types of calls:

- Synchronous calls are represented using solid arrowheads
- Open arrowheads display asynchronous messages
- Reaction messages are delineated by dashed lines.

Sequence diagrams are used primarily to view the sequence of calls in a system to perform a specific functionality.

6.4.2 Sequence diagram for Glaucoma detection

Description:

The user begins the program, and attempts to send the data store images as an input. If the device is activated it receives the image for further processing from the data store. The pre-processing and region-of-interest extraction measures take place. The fundamentally significant sections or regions of the fundus picture that can detect glaucoma are extracted and stored in the data store.

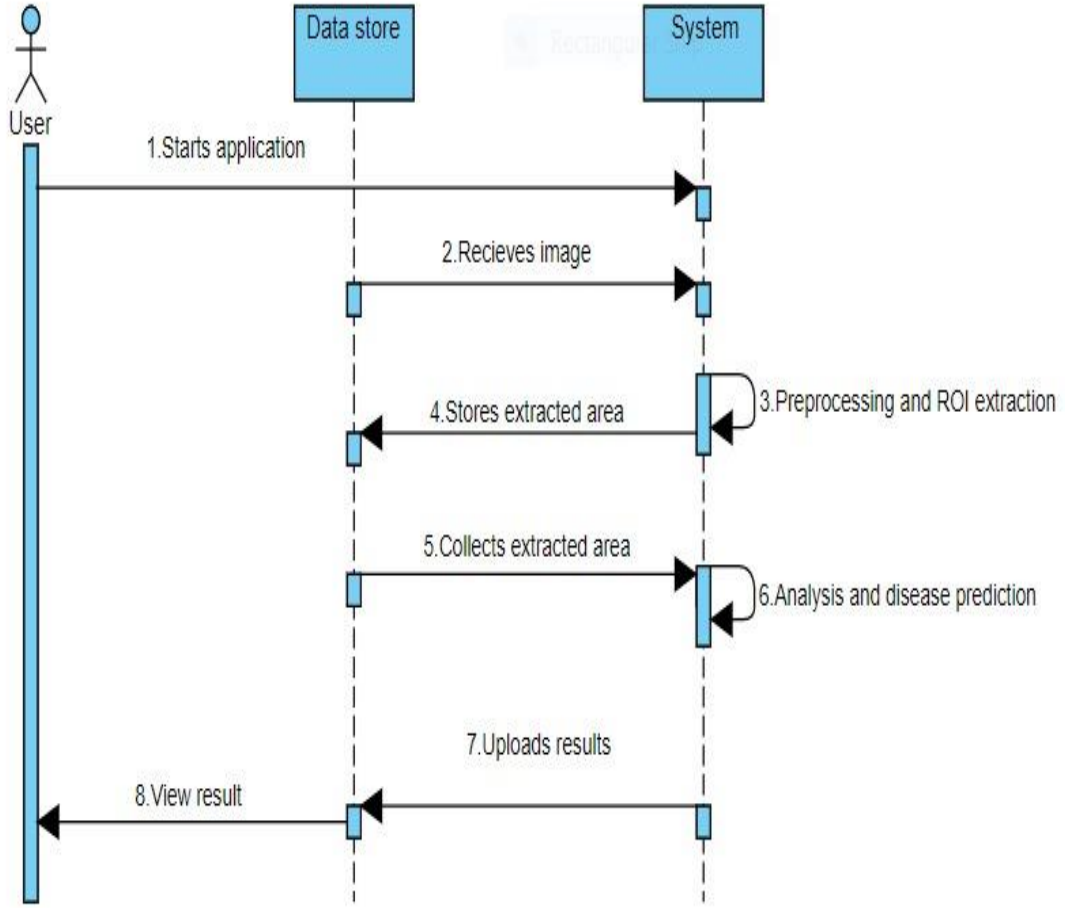


Fig:6.4 Sequence diagram for glaucoma detection

The data store then gathers interest for analysis to the extracted area. The system uses the extracted data to examine and predict illness along with the intensity of effect that the disease has caused on the individual. Then, it uploads the results to the data store for future training. The data store will then display the result to the user using this system for status check.

6.5 Activity Diagram

6.5.1 Definition

Among the UML action diagrams, operation diagram is a diagram showing the progression of control or movement of artifacts between various processes with importance for the order and movement conditions. It portrays the control flow from point to point to end showing the various decision paths that occur during the processing of the task. We may represent both sequential processing and simultaneous processing of activities using task diagrams

Activity can be described in an activity diagram as a parameterised behaviour represented as a coordinated action stream. A node can be defined as the execution of a subordinate etiquette, such as an arithmetic calculation, an operation request, or control of specific object contents.

6.5.2 Activity diagram for Glaucoma detection

Description:

System operation diagram focuses on the flow of data control through the network. It starts from collecting the information and runs all the way until the result is shown by the client and the device ends.

The program can retrieve images from the data store as soon as the user opens the device. The image acquired is then enhanced and reshaped to match the necessary area. The field of interest is accessed by different methods. Such regions for which attention needs to be paid are taken into consideration and reshaped and filtered based on different factors such as RGB pixel intensity and cell brightness. CNN comes into picture after changing the necessary information. CNN analyses the data which it receives in the form of a math grid. The device shows whether the image has a glaucomatous eye or not according to the results obtained and if yes, it also shows the strength of impact. After seeing the result, the machine is then stopped.

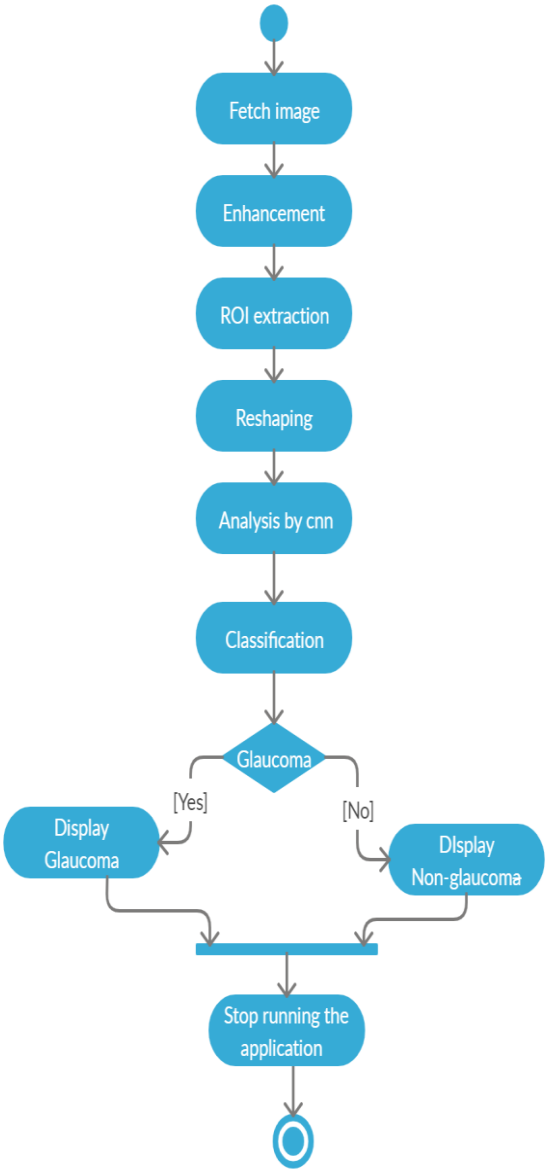


Fig 6.5: Activity diagram for glaucoma detection

7. IMPLEMENTATION

7.1 Code for splitting data:

```
# load dependencies

# misc
import datetime
import os, sys, shutil

# basics
import numpy as np
from numpy import loadtxt
import pandas as pd
from PIL import Image

# set file path variables
train_path = 'C://Users//hp//Desktop//Project-1//train_images'
test_path = 'C://Users//hp//Desktop//Project-1//test_images'

# load csv files with image file names and labels as pandas dataframes
train_data = pd.read_csv('C:/Users/hp/Desktop/Project-1/train.csv')
test_data = pd.read_csv('C:/Users/hp/Desktop/Project-1/test.csv')

# look at training data
train_data.head()

# look at test data
test_data.head()

# number of images in test & train data
print('Number of images in training set is {}'.format(len(train_data)))
print('Number of images in test set is {}'.format(len(test_data)))

# store the class information in some variables for convenience
class_labels = [0,1,2,3,4]
class_dict = {0:'No Glaucoma', 1:'Mild Glaucoma', 2:'Moderate Glaucoma', 3:'Severe Glaucoma', 4:'Proliferative Glaucoma'}
class_list = ['No Glaucoma', 'Mild Glaucoma', 'Moderate Glaucoma', 'Severe Glaucoma', 'Proliferative Glaucoma']
```

```
# look at the distribution of the training data into the 5 classes
train_data.diagnosis.value_counts()

# function to determine largest and smallest dimensions of the images
def get_dimensions(df, path):
    max_width = 0
    max_height = 0
    min_width = 0
    max_height = 0

    file_names = df['id_code']
```

```
    for index, file_name in enumerate(file_names):
        current_image = Image.open(path+'//'+file_name+'.png')

        width, height = current_image.size

        # set initial values
        if max_width == 0:
            max_width = width
            min_width = width
            max_height = height
            min_height = height

        if width > max_width:
            max_width = width
        if width < min_width:
            min_width = width

        if height > max_height:
            max_height = height
        if height < min_height:
            min_height = height

    print('Minimum width: {},'.format(min_width))
    print('Maximum width: {},'.format(max_width))
    print('************')
    print('Minimum height: {}'.format(min_height))
    print('Maximum height: {}'.format(max_height))

# look at the train images sizes
get_dimensions(train_data, train_path)

# look at the test images sizes
get_dimensions(test_data, test_path)
```

```
old_train = train_path
new_folder = 'data_organized'

dir_names = ['train', 'val', 'test']

os.mkdir(new_folder)

for d in dir_names:
    new_dir = os.path.join(new_folder, d)
    os.mkdir(new_dir)


for label in class_labels:
    print('Moving Class {} images.'.format(label))
    for d in dir_names:
        new_dir = os.path.join(new_folder, d, str(label))
        os.mkdir(new_dir)
        print('created \t')
    temp = train_data[train_data.diagnosis == label]
    train, validate, test = np.split(temp.sample(frac=1), [int(.8*len(temp)), int(.9*len(temp))])
    print('Split {} imgs into {} train, {} val, and {} test examples.'.format(len(temp),
                                                                              len(train),
                                                                              len(validate),
                                                                              len(test)))
    for i, temp in enumerate([train, validate, test]):
        for row in temp.index:
            filename = temp['id_code'][row] + '.png'
            origin = os.path.join(old_train + '/' + filename)
            destination = os.path.join(new_folder + '/' + dir_names[i] + '/' + str(label) + '/' + filename)

            shutil.copy(origin, destination)
```

7.2 Code for training the model:

Imports section

```
# load dependencies

# misc
import datetime
import os, sys, shutil

# basics
import numpy as np
from numpy import loadtxt
import pandas as pd
from tqdm import tqdm

# charting
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image
import seaborn as sns

# metrics
from sklearn.metrics import confusion_matrix, cohen_kappa_score

# keras
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
from keras import optimizers, applications
from keras.applications import VGG19
from keras import layers
from keras.layers import Dense, Dropout, GlobalAveragePooling2D, Input
from keras import models
from keras.models import Model
from keras.models import load_model
from keras import optimizers
from keras.applications import ResNet50
```

Loading the dataset

```
[ ] # set file path variables
    train_path = '/content/drive/My Drive/aptos2019-blindness-detection/train_images/'
    test_path = '/content/drive/My Drive/aptos2019-blindness-detection/test_images/'

[ ] # load csv files with image file names and labels as pandas dataframes
    train_data = pd.read_csv('/content/drive/My Drive/aptos2019-blindness-detection/train.csv')
    test_data = pd.read_csv('/content/drive/My Drive/data_organized/aptos2019-blindness-detection/test.csv')

[ ] # look at test data
    test_data.head()
```

	id_code
0	0005cfc8afb6
1	003f0afdcd15
2	006efc72b638
3	00836aaacf06
4	009245722fa4

```
[ ] # number of images in test & train data
    print('Number of images in training set is {}'.format(len(train_data)))
    print('Number of images in test set is {}'.format(len(test_data)))

Number of images in training set is 3662
Number of images in test set is 1928

[ ] # store the class information in some variables for convenience
    class_labels = [0,1,2,3,4]
    class_dict = {0:'No Glaucoma', 1:'Mild Glaucoma', 2:'Moderate Glaucoma', 3:'Severe Glaucoma', 4:'Proliferative Glaucoma'}
    class_list = ['No Glaucoma', 'Mild Glaucoma', 'Moderate Glaucoma', 'Severe Glaucoma', 'Proliferative Glaucoma']
```

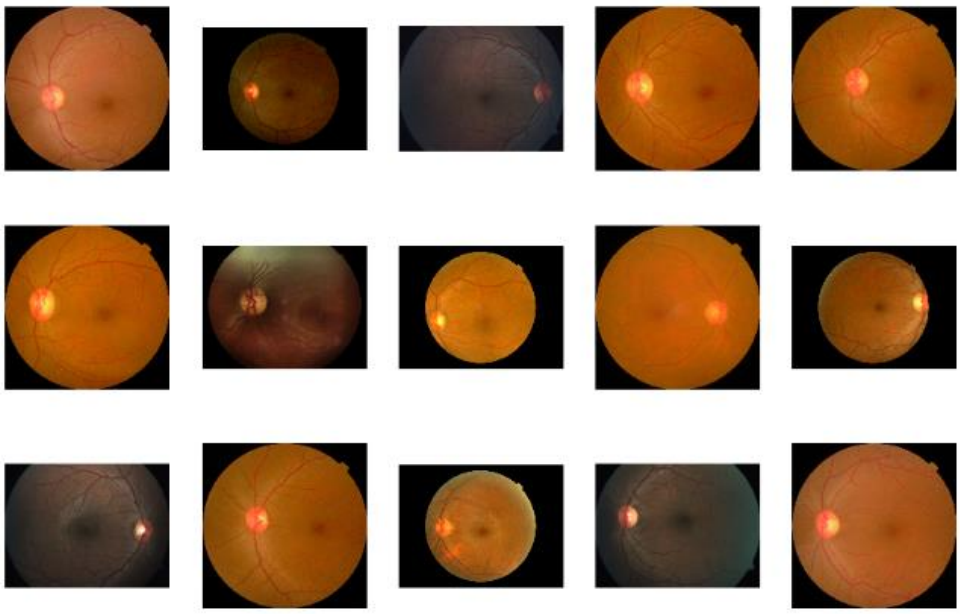
Distribution of images in the dataset by diagnostic class

```
# look at the distribution of the training data into the 5 classes
train_data.diagnosis.value_counts()

0    1805
2     999
1     370
4     295
3     193
Name: diagnosis, dtype: int64

[ ] # plot the distribution of images by class
    fig, ax = plt.subplots(1, 1, figsize=(15,10))

    plt.hist(train_data.diagnosis, 5, facecolor='blue', alpha=0.5)

    plt.xlabel('Classification')
    plt.ylabel('Frequency')
    plt.title(r'Distribution of Images by Diagnostic Class')
    plt.show()
```

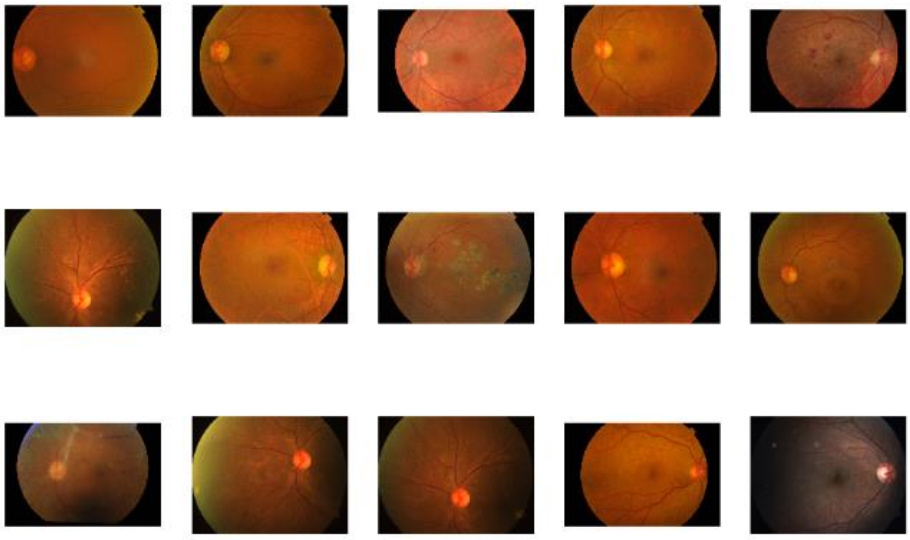
Displaying few samples of the dataset

```
[ ]  # function to show some images

     def show_15_images(df, category):
         # category is 0, 1, 2, 3, 4
         rows = 3
         columns = 5
         fig, ax = plt.subplots(rows, columns, figsize=(15,10))

         title = class_list[category]
         images = df[df['diagnosis'] == category][:15].id_code.values

         for sample in range (0, 15):
             image = Image.open(os.path.join(train_path, images[sample] + '.png'))

             col = sample // columns
             row = sample % columns
             ax[col, row].imshow(image)
             ax[col, row].axis('off')

         plt.suptitle(title)
         plt.show()
```

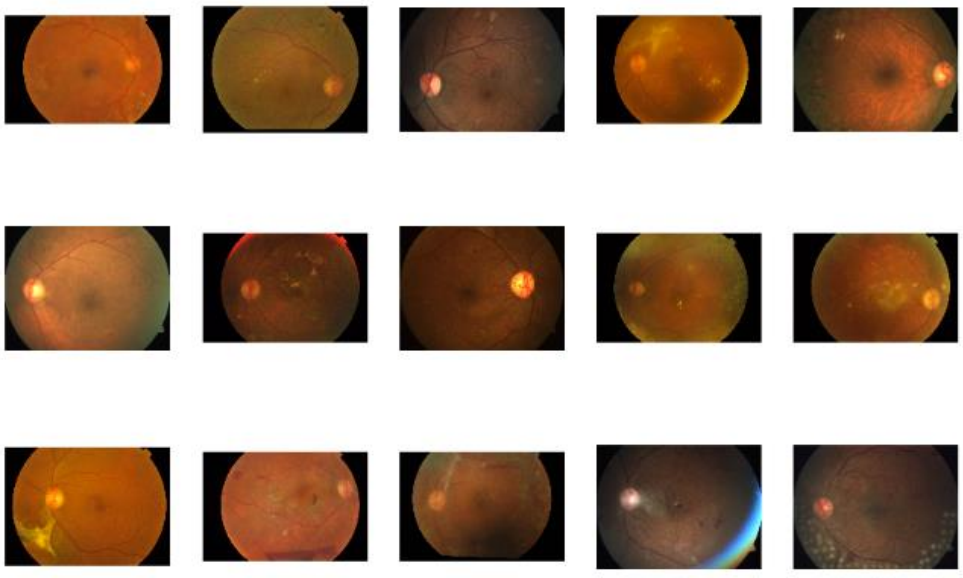
```
[ ]  # look at some class 0 images
     show_15_images(train_data, 0)
```



```
[ ]  # look at some class 1 images
     show_15_images(train_data, 1)
```



```
[ ]  # look at some class 4 images
     show_15_images(train_data, 4)
```



Resizing samples

```python
# function to determine largest and smallest dimensions of the images
def get_dimensions(df, path):
    max_width = 0
    max_height = 0
    min_width = 0
    max_height = 0

    file_names = df['id_code']

    for index, file_name in enumerate(file_names):
        current_image = Image.open(os.path.join(path, file_name + '.png'))

        width, height = current_image.size

        # set initial values
        if max_width == 0:
            max_width = width
            min_width = width
            max_height = height
            min_height = height

        if width > max_width:
            max_width = width
        if width < min_width:
            min_width = width

        if height > max_height:
            max_height = height
        if height < min_height:
            min_height = height

    print('Minimum width: {},'.format(min_width))
    print('Maximum width: {},'.format(max_width))
    print('***********')
    print('Minimum height: {}'.format(min_height))
    print('Maximum height: {}'.format(max_height))
```

```
# look at the train images sizes
get_dimensions(train_data, train_path)
```

```
Minimum width: 474,
Maximum width: 4288,
**********
Minimum height: 358
Maximum height: 2848
```

```
[ ] # look at the test images sizes
    get_dimensions(test_data, test_path)
```

```
Minimum width: 640,
Maximum width: 2896,
**********
Minimum height: 480
Maximum height: 1958
```

```
[ ] # set new paths to the directories
    train_dir = '/content/drive/My Drive/data_organized/train/'
    validation_dir = '/content/drive/My Drive/data_organized/val/'
    test_dir = '/content/drive/My Drive/data_organized/test/'
```

```
[ ] # count the images in each directory
    def count_images(path):
        count = 0
        for directory in os.listdir(path):
            count += len(os.listdir(path + "/" + directory))
        return count
```

```
[ ]  total_train = count_images(train_dir)
     total_train
```

```
2929
```

```
[ ]  total_val = count_images(validation_dir)
     total_val
```

```
365
```

```
[ ]  total_test = count_images(test_dir)
     total_test
```

```
368
```

```
[ ]  # CNN model parameters
     BATCH_SIZE = 32
     EPOCHS = 20  #this take over 16 hours to run, early stopping trials ended at about 6 epochs
     WARMUP_EPOCHS = 2
     LEARNING_RATE = 1e-4
     WARMUP_LEARNING_RATE = 1e-3
     HEIGHT = 128
     WIDTH = 128
     COLORS = 3
     N_CLASSES = 5
     ES_PATIENCE = 5
     RLROP_PATIENCE = 3
     DECAY_DROP = 0.5
```

Reshaping, data augmentation and normalization operations

```
# get all the images in the training directory and reshape and augment them
train_datagen = ImageDataGenerator(
      rescale=1/255,
      rotation_range=20,
#       width_shift_range=0.2,   # removed these for time savings
#       height_shift_range=0.2,
#       shear_range=0.2,
#       zoom_range=0.5,
      horizontal_flip=True,
      fill_mode='nearest')

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(HEIGHT, WIDTH),
        batch_size= BATCH_SIZE,
        shuffle = True,
        class_mode= 'categorical')


# get all the data in the validation directory and reshape them
val_datagen = ImageDataGenerator(rescale=1/255)

val_generator = val_datagen.flow_from_directory(
        validation_dir,
        target_size=(HEIGHT, WIDTH),
        batch_size = BATCH_SIZE,
        class_mode= 'categorical')

# get all the data in the test directory and reshape them
test_generator = ImageDataGenerator(rescale=1/255).flow_from_directory(
        test_dir,
        target_size=(HEIGHT, WIDTH),
        batch_size = 1,
        class_mode= 'categorical',
        shuffle = False)
```

Creating model

```
Found 2929 images belonging to 5 classes.
Found 365 images belonging to 5 classes.
Found 368 images belonging to 5 classes.
```

```
# created a function to create models - all using ResNet50 weights
def create_model(input_shape, n_out):
    input_tensor = Input(shape=input_shape)
    base_model = applications.ResNet50(weights=None,
                                       include_top=False,
                                       input_tensor=input_tensor)
    base_model.load_weights('resnet50/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5')

    x = GlobalAveragePooling2D()(base_model.output)
    x = Dropout(0.5)(x)
    x = Dense(2048, activation='relu')(x)
    x = Dropout(0.5)(x)
    final_output = Dense(n_out, activation='softmax', name='final_output')(x)
    model = Model(input_tensor, final_output)

    return model
```

```
# instantiate and compile a model
model = create_model(input_shape=(HEIGHT, WIDTH, COLORS), n_out=N_CLASSES)

for layer in model.layers:
    layer.trainable = False

for i in range(-5, 0):
    model.layers[i].trainable = True

metric_list = ["accuracy"]
optimizer = optimizers.Adam(lr=WARMUP_LEARNING_RATE)
model.compile(optimizer=optimizer, loss="categorical_crossentropy",  metrics=metric_list)
model.summary()
```

Training model

```
# warm up training phase, only two epochs
STEP_SIZE_TRAIN = train_generator.n//train_generator.batch_size
STEP_SIZE_VALID = val_generator.n//val_generator.batch_size

history_warmup = model.fit_generator(generator=train_generator,
                            steps_per_epoch=STEP_SIZE_TRAIN,
                            validation_data=val_generator,
                            validation_steps=STEP_SIZE_VALID,
                            epochs=WARMUP_EPOCHS,
                            verbose=1).history
```

```
Epoch 1/2
91/91 [==============================] - 441s 5s/step - loss: 1.3500 - acc: 0.6539 - val_loss: 1.8089 - val_acc: 0.4932
Epoch 2/2
91/91 [==============================] - 433s 5s/step - loss: 0.8186 - acc: 0.7092 - val_loss: 1.8386 - val_acc: 0.4932
```

```
# create the model, use early stopping
for layer in model.layers:
    layer.trainable = True

es = EarlyStopping(monitor='val_loss', mode='min', patience=ES_PATIENCE, verbose=1)
rlrop = ReduceLROnPlateau(monitor='val_loss', mode='min', patience=RLROP_PATIENCE, factor=DECAY_DROP, min_lr=1e-6, verbose=1)

callback_list = [es, rlrop]
optimizer = optimizers.Adam(lr=LEARNING_RATE)
model.compile(optimizer=optimizer, loss="binary_crossentropy",  metrics=metric_list)
model.summary()
```

```
[ ]  STEP_SIZE_TRAIN = train_generator.n//train_generator.batch_size
     STEP_SIZE_VALID = val_generator.n//val_generator.batch_size
```

```
[ ]  history_finetunning = model.fit_generator(generator=train_generator,
                                steps_per_epoch=STEP_SIZE_TRAIN,
                                validation_data=val_generator,
                                validation_steps=STEP_SIZE_VALID,
                                epochs=3, # revert to: epochs=EPOCHS
                                callbacks=callback_list,
                                verbose=1).history
```

```
Epoch 1/3
91/91 [==============================] - 580s 6s/step - loss: 0.2080 - acc: 0.9139 - val_loss: 0.2619 - val_acc: 0.9156
Epoch 2/3
91/91 [==============================] - 573s 6s/step - loss: 0.1825 - acc: 0.9242 - val_loss: 0.2388 - val_acc: 0.8964
Epoch 3/3
91/91 [==============================] - 569s 6s/step - loss: 0.1632 - acc: 0.9341 - val_loss: 0.2036 - val_acc: 0.9096
```

Plotting accuracy and loss

```
history = {'loss': history_warmup['loss'] + history_finetunning['loss'],
           'val_loss': history_warmup['val_loss'] + history_finetunning['val_loss'],
           'acc': history_warmup['acc'] + history_finetunning['acc'],
           'val_acc': history_warmup['val_acc'] + history_finetunning['val_acc']}

sns.set_style("whitegrid")
fig, (ax1, ax2) = plt.subplots(2, 1, sharex='col', figsize=(20, 14))

ax1.plot(history['loss'], label='Train loss')
ax1.plot(history['val_loss'], label='Validation loss')
ax1.legend(loc='best')
ax1.set_title('Loss')

ax2.plot(history['acc'], label='Train Accuracy')
ax2.plot(history['val_acc'], label='Validation accuracy')
ax2.legend(loc='best')
ax2.set_title('Accuracy')

plt.xlabel('Epochs')
sns.despine()
plt.show()
```

Saving the model

```
[ ]  # save the model

     model.save('Final_Model.h5')
```

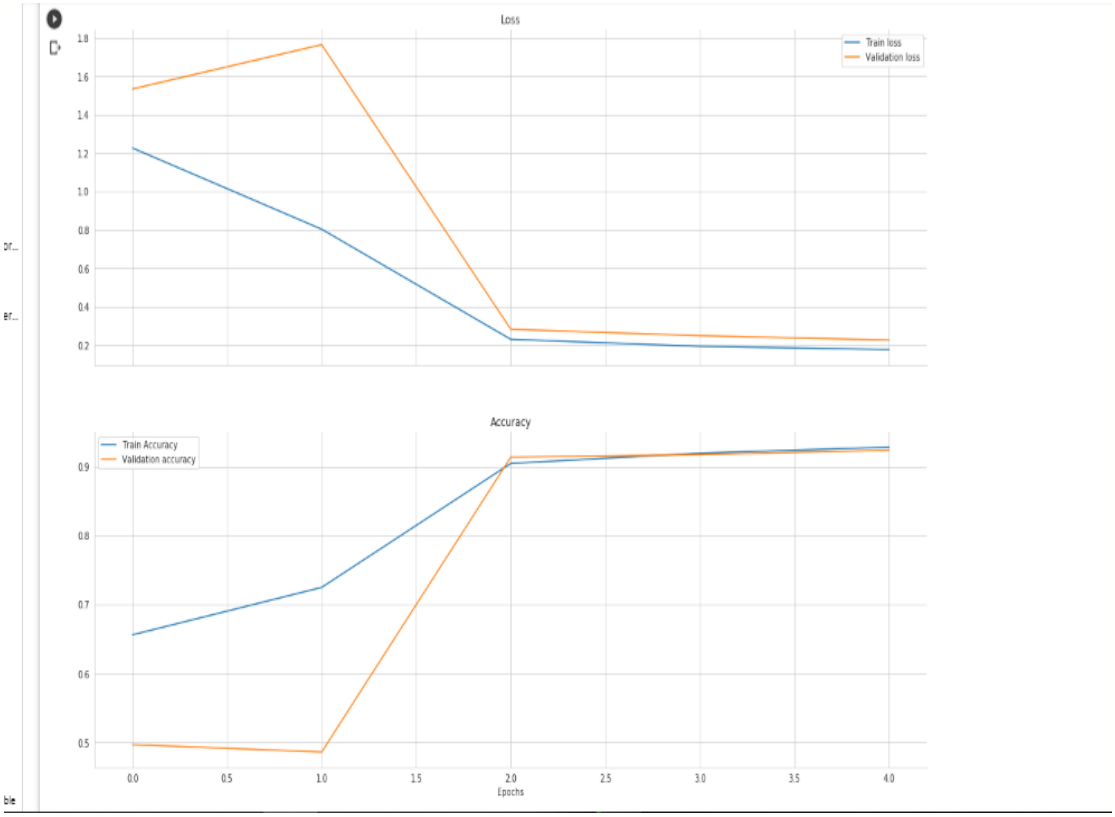


Figure 7.1 Model accuracy against train data and validation data

The Fig. 7.1 Proves the consistency of the methods we suggest. The accuracy percentage obtained from the CNN model is 93.41% and the accuracy derived for the validation is 90.96%. Fig 7.1 also demonstrates the relation between the number of epochs and the loss of the model, as we see that there is a loss in the model as we gradually increase the count in epochs. If we provide a rise in the number of epochs, then the Neural Network begins to overfit, which means that it has started to learn from static noise data and leads to a decrease in the accuracy of real time and to incorrect results. And if we reduce the number of epochs to less than 20 it begins to get under fit and produces precision of less than 80% based on the epochs used.

7.3 Layers in CNN model:

It shows

- Number count of layers we applied to the model.
- Mixture of trainable parameters available inside the model
- Output dimensions after each layer.
- Parameters available in each layer.

Layer (type)	Output Shape	Param #	Connected to
input_6 (InputLayer)	(None, 128, 128, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 134, 134, 3)	0	input_6[0][0]
conv1 (Conv2D)	(None, 64, 64, 64)	9472	conv1_pad[0][0]
bn_conv1 (BatchNormalization)	(None, 64, 64, 64)	256	conv1[0][0]
activation_246 (Activation)	(None, 64, 64, 64)	0	bn_conv1[0][0]
max_pooling2d_6 (MaxPooling2D)	(None, 31, 31, 64)	0	activation_246[0][0]
res2a_branch2a (Conv2D)	(None, 31, 31, 64)	4160	max_pooling2d_6[0][0]
bn2a_branch2a (BatchNormalizati	(None, 31, 31, 64)	256	res2a_branch2a[0][0]
activation_247 (Activation)	(None, 31, 31, 64)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 31, 31, 64)	36928	activation_247[0][0]
bn2a_branch2b (BatchNormalizati	(None, 31, 31, 64)	256	res2a_branch2b[0][0]
activation_248 (Activation)	(None, 31, 31, 64)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 31, 31, 256)	16640	activation_248[0][0]
res2a_branch1 (Conv2D)	(None, 31, 31, 256)	16640	max_pooling2d_6[0][0]
bn2a_branch2c (BatchNormalizati	(None, 31, 31, 256)	1024	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalizatio	(None, 31, 31, 256)	1024	res2a_branch1[0][0]
add_81 (Add)	(None, 31, 31, 256)	0	bn2a_branch2c[0][0] bn2a_branch1[0][0]
activation_249 (Activation)	(None, 31, 31, 256)	0	add_81[0][0]
res2b_branch2a (Conv2D)	(None, 31, 31, 64)	16448	activation_249[0][0]
bn2b_branch2a (BatchNormalizati	(None, 31, 31, 64)	256	res2b_branch2a[0][0]
activation_250 (Activation)	(None, 31, 31, 64)	0	bn2b_branch2a[0][0]
res2b_branch2b (Conv2D)	(None, 31, 31, 64)	36928	activation_250[0][0]
bn2b_branch2b (BatchNormalizati	(None, 31, 31, 64)	256	res2b_branch2b[0][0]
activation_251 (Activation)	(None, 31, 31, 64)	0	bn2b_branch2b[0][0]
res2b_branch2c (Conv2D)	(None, 31, 31, 256)	16640	activation_251[0][0]
bn2b_branch2c (BatchNormalizati	(None, 31, 31, 256)	1024	res2b_branch2c[0][0]
add_82 (Add)	(None, 31, 31, 256)	0	bn2b_branch2c[0][0] activation_249[0][0]
activation_252 (Activation)	(None, 31, 31, 256)	0	add_82[0][0]

res2c_branch2a (Conv2D)	(None, 31, 31, 64)	16448	activation_252[0][0]
bn2c_branch2a (BatchNormalizati	(None, 31, 31, 64)	256	res2c_branch2a[0][0]
activation_253 (Activation)	(None, 31, 31, 64)	0	bn2c_branch2a[0][0]
res2c_branch2b (Conv2D)	(None, 31, 31, 64)	36928	activation_253[0][0]
bn2c_branch2b (BatchNormalizati	(None, 31, 31, 64)	256	res2c_branch2b[0][0]
activation_254 (Activation)	(None, 31, 31, 64)	0	bn2c_branch2b[0][0]
res2c_branch2c (Conv2D)	(None, 31, 31, 256)	16640	activation_254[0][0]
bn2c_branch2c (BatchNormalizati	(None, 31, 31, 256)	1024	res2c_branch2c[0][0]
add_83 (Add)	(None, 31, 31, 256)	0	bn2c_branch2c[0][0] activation_252[0][0]
activation_255 (Activation)	(None, 31, 31, 256)	0	add_83[0][0]
res3a_branch2a (Conv2D)	(None, 16, 16, 128)	32896	activation_255[0][0]
bn3a_branch2a (BatchNormalizati	(None, 16, 16, 128)	512	res3a_branch2a[0][0]
activation_256 (Activation)	(None, 16, 16, 128)	0	bn3a_branch2a[0][0]
res3a_branch2b (Conv2D)	(None, 16, 16, 128)	147584	activation_256[0][0]
bn3a_branch2b (BatchNormalizati	(None, 16, 16, 128)	512	res3a_branch2b[0][0]
activation_257 (Activation)	(None, 16, 16, 128)	0	bn3a_branch2b[0][0]
res3a_branch2c (Conv2D)	(None, 16, 16, 512)	66048	activation_257[0][0]
res3a_branch1 (Conv2D)	(None, 16, 16, 512)	131584	activation_255[0][0]
bn3a_branch2c (BatchNormalizati	(None, 16, 16, 512)	2048	res3a_branch2c[0][0]
bn3a_branch1 (BatchNormalizatio	(None, 16, 16, 512)	2048	res3a_branch1[0][0]
add_84 (Add)	(None, 16, 16, 512)	0	bn3a_branch2c[0][0] bn3a_branch1[0][0]
activation_258 (Activation)	(None, 16, 16, 512)	0	add_84[0][0]
res3b_branch2a (Conv2D)	(None, 16, 16, 128)	65664	activation_258[0][0]
bn3b_branch2a (BatchNormalizati	(None, 16, 16, 128)	512	res3b_branch2a[0][0]
activation_259 (Activation)	(None, 16, 16, 128)	0	bn3b_branch2a[0][0]
res3b_branch2b (Conv2D)	(None, 16, 16, 128)	147584	activation_259[0][0]
bn3b_branch2b (BatchNormalizati	(None, 16, 16, 128)	512	res3b_branch2b[0][0]
activation_260 (Activation)	(None, 16, 16, 128)	0	bn3b_branch2b[0][0]
res3b_branch2c (Conv2D)	(None, 16, 16, 512)	66048	activation_260[0][0]
bn3b_branch2c (BatchNormalizati	(None, 16, 16, 512)	2048	res3b_branch2c[0][0]
add_85 (Add)	(None, 16, 16, 512)	0	bn3b_branch2c[0][0] activation_258[0][0]
activation_261 (Activation)	(None, 16, 16, 512)	0	add_85[0][0]

res3c_branch2a (Conv2D)	(None, 16, 16, 128)	65664	activation_261[0][0]
bn3c_branch2a (BatchNormalizati	(None, 16, 16, 128)	512	res3c_branch2a[0][0]
activation_262 (Activation)	(None, 16, 16, 128)	0	bn3c_branch2a[0][0]
res3c_branch2b (Conv2D)	(None, 16, 16, 128)	147584	activation_262[0][0]
bn3c_branch2b (BatchNormalizati	(None, 16, 16, 128)	512	res3c_branch2b[0][0]
activation_263 (Activation)	(None, 16, 16, 128)	0	bn3c_branch2b[0][0]
res3c_branch2c (Conv2D)	(None, 16, 16, 512)	66048	activation_263[0][0]
bn3c_branch2c (BatchNormalizati	(None, 16, 16, 512)	2048	res3c_branch2c[0][0]
add_86 (Add)	(None, 16, 16, 512)	0	bn3c_branch2c[0][0] activation_261[0][0]
activation_264 (Activation)	(None, 16, 16, 512)	0	add_86[0][0]
res3d_branch2a (Conv2D)	(None, 16, 16, 128)	65664	activation_264[0][0]
bn3d_branch2a (BatchNormalizati	(None, 16, 16, 128)	512	res3d_branch2a[0][0]
activation_265 (Activation)	(None, 16, 16, 128)	0	bn3d_branch2a[0][0]
res3d_branch2b (Conv2D)	(None, 16, 16, 128)	147584	activation_265[0][0]
bn3d_branch2b (BatchNormalizati	(None, 16, 16, 128)	512	res3d_branch2b[0][0]
activation_266 (Activation)	(None, 16, 16, 128)	0	bn3d_branch2b[0][0]
res3d_branch2c (Conv2D)	(None, 16, 16, 512)	66048	activation_266[0][0]
bn3d_branch2c (BatchNormalizati	(None, 16, 16, 512)	2048	res3d_branch2c[0][0]
add_87 (Add)	(None, 16, 16, 512)	0	bn3d_branch2c[0][0] activation_264[0][0]
activation_267 (Activation)	(None, 16, 16, 512)	0	add_87[0][0]
res4a_branch2a (Conv2D)	(None, 8, 8, 256)	131328	activation_267[0][0]
bn4a_branch2a (BatchNormalizati	(None, 8, 8, 256)	1024	res4a_branch2a[0][0]
activation_268 (Activation)	(None, 8, 8, 256)	0	bn4a_branch2a[0][0]
res4a_branch2b (Conv2D)	(None, 8, 8, 256)	590080	activation_268[0][0]
bn4a_branch2b (BatchNormalizati	(None, 8, 8, 256)	1024	res4a_branch2b[0][0]
activation_269 (Activation)	(None, 8, 8, 256)	0	bn4a_branch2b[0][0]
res4a_branch2c (Conv2D)	(None, 8, 8, 1024)	263168	activation_269[0][0]
res4a_branch1 (Conv2D)	(None, 8, 8, 1024)	525312	activation_267[0][0]
bn4a_branch2c (BatchNormalizati	(None, 8, 8, 1024)	4096	res4a_branch2c[0][0]
bn4a_branch1 (BatchNormalizatio	(None, 8, 8, 1024)	4096	res4a_branch1[0][0]
add_88 (Add)	(None, 8, 8, 1024)	0	bn4a_branch2c[0][0] bn4a_branch1[0][0]
activation_270 (Activation)	(None, 8, 8, 1024)	0	add_88[0][0]

res4b_branch2a (Conv2D)	(None, 8, 8, 256)	262400	activation_270[0][0]
bn4b_branch2a (BatchNormalizati	(None, 8, 8, 256)	1024	res4b_branch2a[0][0]
activation_271 (Activation)	(None, 8, 8, 256)	0	bn4b_branch2a[0][0]
res4b_branch2b (Conv2D)	(None, 8, 8, 256)	590080	activation_271[0][0]
bn4b_branch2b (BatchNormalizati	(None, 8, 8, 256)	1024	res4b_branch2b[0][0]
activation_272 (Activation)	(None, 8, 8, 256)	0	bn4b_branch2b[0][0]
res4b_branch2c (Conv2D)	(None, 8, 8, 1024)	263168	activation_272[0][0]
bn4b_branch2c (BatchNormalizati	(None, 8, 8, 1024)	4096	res4b_branch2c[0][0]
add_89 (Add)	(None, 8, 8, 1024)	0	bn4b_branch2c[0][0] activation_270[0][0]
activation_273 (Activation)	(None, 8, 8, 1024)	0	add_89[0][0]
res4c_branch2a (Conv2D)	(None, 8, 8, 256)	262400	activation_273[0][0]
bn4c_branch2a (BatchNormalizati	(None, 8, 8, 256)	1024	res4c_branch2a[0][0]
activation_274 (Activation)	(None, 8, 8, 256)	0	bn4c_branch2a[0][0]
res4c_branch2b (Conv2D)	(None, 8, 8, 256)	590080	activation_274[0][0]
bn4c_branch2b (BatchNormalizati	(None, 8, 8, 256)	1024	res4c_branch2b[0][0]
activation_275 (Activation)	(None, 8, 8, 256)	0	bn4c_branch2b[0][0]
res4c_branch2c (Conv2D)	(None, 8, 8, 1024)	263168	activation_275[0][0]
bn4c_branch2c (BatchNormalizati	(None, 8, 8, 1024)	4096	res4c_branch2c[0][0]
add_90 (Add)	(None, 8, 8, 1024)	0	bn4c_branch2c[0][0] activation_273[0][0]
activation_276 (Activation)	(None, 8, 8, 1024)	0	add_90[0][0]
res4d_branch2a (Conv2D)	(None, 8, 8, 256)	262400	activation_276[0][0]
bn4d_branch2a (BatchNormalizati	(None, 8, 8, 256)	1024	res4d_branch2a[0][0]
activation_277 (Activation)	(None, 8, 8, 256)	0	bn4d_branch2a[0][0]
res4d_branch2b (Conv2D)	(None, 8, 8, 256)	590080	activation_277[0][0]
bn4d_branch2b (BatchNormalizati	(None, 8, 8, 256)	1024	res4d_branch2b[0][0]
activation_278 (Activation)	(None, 8, 8, 256)	0	bn4d_branch2b[0][0]
res4d_branch2c (Conv2D)	(None, 8, 8, 1024)	263168	activation_278[0][0]
bn4d_branch2c (BatchNormalizati	(None, 8, 8, 1024)	4096	res4d_branch2c[0][0]
add_91 (Add)	(None, 8, 8, 1024)	0	bn4d_branch2c[0][0] activation_276[0][0]
activation_279 (Activation)	(None, 8, 8, 1024)	0	add_91[0][0]
res4e_branch2a (Conv2D)	(None, 8, 8, 256)	262400	activation_279[0][0]
bn4e_branch2a (BatchNormalizati	(None, 8, 8, 256)	1024	res4e_branch2a[0][0]

activation_280 (Activation)	(None, 8, 8, 256)	0	bn4e_branch2a[0][0]
res4e_branch2b (Conv2D)	(None, 8, 8, 256)	590080	activation_280[0][0]
bn4e_branch2b (BatchNormalizati	(None, 8, 8, 256)	1024	res4e_branch2b[0][0]
activation_281 (Activation)	(None, 8, 8, 256)	0	bn4e_branch2b[0][0]
res4e_branch2c (Conv2D)	(None, 8, 8, 1024)	263168	activation_281[0][0]
bn4e_branch2c (BatchNormalizati	(None, 8, 8, 1024)	4096	res4e_branch2c[0][0]
add_92 (Add)	(None, 8, 8, 1024)	0	bn4e_branch2c[0][0] activation_279[0][0]
activation_282 (Activation)	(None, 8, 8, 1024)	0	add_92[0][0]
res4f_branch2a (Conv2D)	(None, 8, 8, 256)	262400	activation_282[0][0]
bn4f_branch2a (BatchNormalizati	(None, 8, 8, 256)	1024	res4f_branch2a[0][0]
activation_283 (Activation)	(None, 8, 8, 256)	0	bn4f_branch2a[0][0]
res4f_branch2b (Conv2D)	(None, 8, 8, 256)	590080	activation_283[0][0]
bn4f_branch2b (BatchNormalizati	(None, 8, 8, 256)	1024	res4f_branch2b[0][0]
activation_284 (Activation)	(None, 8, 8, 256)	0	bn4f_branch2b[0][0]
res4f_branch2c (Conv2D)	(None, 8, 8, 1024)	263168	activation_284[0][0]
bn4f_branch2c (BatchNormalizati	(None, 8, 8, 1024)	4096	res4f_branch2c[0][0]
add_93 (Add)	(None, 8, 8, 1024)	0	bn4f_branch2c[0][0] activation_282[0][0]
activation_285 (Activation)	(None, 8, 8, 1024)	0	add_93[0][0]
res5a_branch2a (Conv2D)	(None, 4, 4, 512)	524800	activation_285[0][0]
bn5a_branch2a (BatchNormalizati	(None, 4, 4, 512)	2048	res5a_branch2a[0][0]
activation_286 (Activation)	(None, 4, 4, 512)	0	bn5a_branch2a[0][0]
res5a_branch2b (Conv2D)	(None, 4, 4, 512)	2359808	activation_286[0][0]
bn5a_branch2b (BatchNormalizati	(None, 4, 4, 512)	2048	res5a_branch2b[0][0]
activation_287 (Activation)	(None, 4, 4, 512)	0	bn5a_branch2b[0][0]
res5a_branch2c (Conv2D)	(None, 4, 4, 2048)	1050624	activation_287[0][0]
res5a_branch1 (Conv2D)	(None, 4, 4, 2048)	2099200	activation_285[0][0]
bn5a_branch2c (BatchNormalizati	(None, 4, 4, 2048)	8192	res5a_branch2c[0][0]
bn5a_branch1 (BatchNormalizatio	(None, 4, 4, 2048)	8192	res5a_branch1[0][0]
add_94 (Add)	(None, 4, 4, 2048)	0	bn5a_branch2c[0][0] bn5a_branch1[0][0]
activation_288 (Activation)	(None, 4, 4, 2048)	0	add_94[0][0]
res5b_branch2a (Conv2D)	(None, 4, 4, 512)	1049088	activation_288[0][0]
bn5b_branch2a (BatchNormalizati	(None, 4, 4, 512)	2048	res5b_branch2a[0][0]

activation_289 (Activation)	(None, 4, 4, 512)	0	bn5b_branch2a[0][0]
res5b_branch2b (Conv2D)	(None, 4, 4, 512)	2359808	activation_289[0][0]
bn5b_branch2b (BatchNormalizati	(None, 4, 4, 512)	2048	res5b_branch2b[0][0]
activation_290 (Activation)	(None, 4, 4, 512)	0	bn5b_branch2b[0][0]
res5b_branch2c (Conv2D)	(None, 4, 4, 2048)	1050624	activation_290[0][0]
bn5b_branch2c (BatchNormalizati	(None, 4, 4, 2048)	8192	res5b_branch2c[0][0]
add_95 (Add)	(None, 4, 4, 2048)	0	bn5b_branch2c[0][0] activation_288[0][0]
activation_291 (Activation)	(None, 4, 4, 2048)	0	add_95[0][0]
res5c_branch2a (Conv2D)	(None, 4, 4, 512)	1049088	activation_291[0][0]
bn5c_branch2a (BatchNormalizati	(None, 4, 4, 512)	2048	res5c_branch2a[0][0]
activation_292 (Activation)	(None, 4, 4, 512)	0	bn5c_branch2a[0][0]
res5c_branch2b (Conv2D)	(None, 4, 4, 512)	2359808	activation_292[0][0]
bn5c_branch2b (BatchNormalizati	(None, 4, 4, 512)	2048	res5c_branch2b[0][0]
activation_293 (Activation)	(None, 4, 4, 512)	0	bn5c_branch2b[0][0]
res5c_branch2c (Conv2D)	(None, 4, 4, 2048)	1050624	activation_293[0][0]
bn5c_branch2c (BatchNormalizati	(None, 4, 4, 2048)	8192	res5c_branch2c[0][0]
add_96 (Add)	(None, 4, 4, 2048)	0	bn5c_branch2c[0][0] activation_291[0][0]
activation_294 (Activation)	(None, 4, 4, 2048)	0	add_96[0][0]
global_average_pooling2d_3 (Glo	(None, 2048)	0	activation_294[0][0]
dropout_5 (Dropout)	(None, 2048)	0	global_average_pooling2d_3[0][0]
dense_3 (Dense)	(None, 2048)	4196352	dropout_5[0][0]
dropout_6 (Dropout)	(None, 2048)	0	dense_3[0][0]
final_output (Dense)	(None, 5)	10245	dropout_6[0][0]

==========

Total params: 27,794,309
Trainable params: 27,741,189
Non-trainable params: 53,120

7.4 Code for testing the model:

Imports section

```
[ ]  import numpy as np
     from numpy import loadtxt
     from sklearn.metrics import confusion_matrix
     import os
     import cv2
     # charting
     import matplotlib.pyplot as plt
     import seaborn as sns

     # metrics
     from sklearn.metrics import confusion_matrix, cohen_kappa_score
     from keras.models import Model
     from keras.models import load_model
```

Predicting output labels

```
[ ]  img_array=[]



     class_labels = [0,1,2,3,4]
     class_dict = {0:'No glaucoma', 1:'Mild glaucoma', 2:'Moderate glaucoma', 3:'Severe glaucoma', 4:'Proliferative glaucoma'}
     class_list = ['No glaucoma', 'Mild glaucoma', 'Moderate glaucoma', 'Severe glaucoma', 'Proliferative glaucoma']
```

```
[ ]  model=load_model('/content/drive/My Drive/Models/FinalModel.h5',compile=False)
```

```
imageRealTimeTestDir='/content/drive/My Drive/Test'
Predictions=[]
Actual_labels=[]
for foldername in os.listdir(imageRealTimeTestDir):

  for filename in os.listdir(imageRealTimeTestDir+"//"+foldername):
    img=cv2.imread(imageRealTimeTestDir+"//"+foldername+"//"+filename)
    backup=img
    img=cv2.resize(img,(128,128))
    img=np.reshape(img,[1,128,128,3])
    output=model.predict([img])
    ind=np.argmax(output[0])
    print('For image {} predicted label is {}'.format(filename,ind))
    Predictions.append(ind)
    Actual_labels.append(int(foldername))
```

Confusion matrix

```
[ ]  array=confusion_matrix(Actual_labels,Predictions)

     print(array)
```

```
[[173   6   7   0   1]
 [  3  30   3   0   1]
 [  0   6  88   6   0]
 [  0   0   3  16   1]
 [  0   0   0   2  28]]
```

```
[ ]  import seaborn as sn
     import pandas as pd
     import matplotlib.pyplot as plt
```

```
[ ]  df_cm = pd.DataFrame(array, index = [i for i in class_list],
                          columns = [i for i in class_list])

     ax=plt.subplot()

     sn.heatmap(df_cm,annot=True,fmt='d') # font size

     ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
     ax.set_title('Confusion Matrix');
     ax.xaxis.set_ticklabels(class_list)
     ax.yaxis.set_ticklabels(class_list)
     plt.show()
```

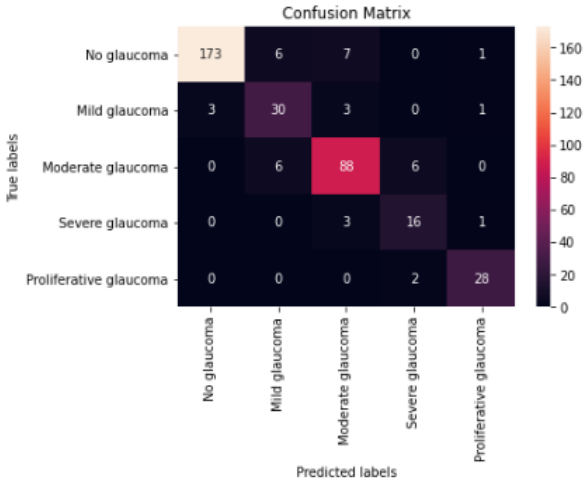


Figure 7.2 - Confusion Matrix

Precision and accuracy score

```
[ ]  from sklearn.metrics import precision_score
     from sklearn.metrics import accuracy_score
     score=precision_score(Actual_labels,Predictions,average='macro')
     print("Precision score is {}%".format(score*100))

     score=accuracy_score(Actual_labels,Predictions)
     print("Accuracy score is {}%".format(score*100))

     Precision score is 82.76839723142821%
     Accuracy score is 89.57219251336899%
```

Precision score is 82.76839723142821%

Accuracy score is 89.57219251336899%

The confusion matrix defines the efficiency parameter of a classifier and its capabilities. This matrix contains information about the classifications predicted and the actual ones conducted by a system that does classification. The uncertain feature of matrix for our proposed classification model is composed of five rows and columns each. In the matrix the output assessment parameter of the classification model was depicted in the context of classifying Glaucoma. The matrix of uncertainty obtained as output is shown in figure 7 below. The matrix rows in the order from top to bottom (rows 1 to 5) correspond to actual classes respectively, in the order of severity: no, mild, moderate, severe, and proliferative. The matrix columns from left to right (columns 1 to 5)

correspond to classes forecast in the same order of increasing severity: no, mild, moderate, severe, and proliferative.

From a sample of 187 non-Glaucoma photos 173 were correctly predicted as "No Glaucoma" thus achieving a 92.51 per cent accuracy.

From a sample of 37 mild-glaucoma images 30 were correctly predicted as "Mild Glaucoma" thus achieving 81.08 per cent accuracy.

88 is correctly predicted as "Severe Glaucoma" from a compilation of 100 low-Glaucoma images thereby achieving an accuracy of 88 per cent.

From a compilation of 20 severe-Glaucoma images 16 were correctly predicted as "Severe Glaucoma" thereby achieving 80 per cent accuracy.

From a sample of 30 proliferative-Glaucoma photos 28 were correctly predicted as "Proliferative Glaucoma" thereby achieving 93.33 percent accuracy.

8.TESTING

In several software development methodologies, after the implementation is completed, an individual process of testing is dedicated to be performed. Using this method has a privilege of making it easier to catch a sight of one's own mistakes and a fresh eye would rather spot obvious errors much faster than the one who has gone through the material multiple times.

Testing is a software execution process with the intention of detecting error. It should be error free to make our program work well. If testing is performed successfully it will remove all the program errors.

8.1 Types of Testing

In general, testing after developing a software is divided into two major wide-ranging categories. One would be functional testing and the other would be non-functional testing. One more general form of test is known to be maintenance testing.

8.1.1 Functional testing:

Functional testing indicates an assessment to the software application's functional features. You would need to test each and every application when conducting functional tests. You need to see if you get the results you want, or not.

Functional checks are carried out both manually and by means of automation software. Manual testing is simple for this form of testing but you can use software if appropriate.

8.1.2 Non-Functional Testing

Non-functional testing is known to be the testing of an application's non-functional features such as functionality, reliability, practicability, safety, etc. During the functional checks non-functional checks are carried out.

Through non-functional research, you will be able to significantly increase the consistency of your applications. Functional tests also increase the consistency so you possess the ability of making the program much preferable with non-functional tests. The non-functional testing helps the program to be polished. This sort of research isn't about whether or not the app works. Instead, it's about how well the app is running, and nothing more.

In general, non-functional experiments are not conducted by hand. In reality, manually

performing these kinds of tests is difficult. But typically, such experiments are conducted using software.

8.2 Test Cases

Test case ID	Test case name	Description	Expected Results	Actual Results	Test Results
1	Verification of data set	Test case deals with images	Images should be clear and coloured	Images should be clear and coloured	Pass
2	Data Reshaping	All the images are checked for equal dimensions.	All samples must be reshaped correctly.	Worked as expected.	Pass
3	Validation of custom architecture	To evaluate the model performance	High training and validation accuracy which are very close	Achieved accuracy for training around 51% and validation accuracy of 54%	Fail
4	Validation of RESNET50 architecture with 3 epochs	To evaluate the model performance	High training and validation accuracy which are very close	Achieved training accuracy of 95% and validation accuracy of 47%	Fail

5	Validation of RESNET50 architecture with 5 epochs	To evaluate the model performance	High training and validation accuracy which are very close	Achieved training accuracy of 93% and validation accuracy of 90%	Pass
6	Confusion matrix	The values in the confusion matrix are checked for suitability	Should give more True Positives	Worked as expected	Pass

Table 8.2 Test cases

9. CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

This project provides proof of deep neural networks that are a feasible medical image-based technique, even though they approach the task in question differently from virtually all well-documented past research. We consider this motivating, especially considering the neural approach restraining the fully supervised characteristics, which makes it unique and advantageous that it acquires knowledge from raw pixel data and does not necessarily rely on any prior vessel structure domain knowledge. We have followed the metric of accuracy in order to test the implementation of our CNN classifier. We used 1928 Glaucoma detection images in which we obtained 93.41 percent training accuracy and 90.96 percent validation accuracy.

In addition, automation assists in prediction, prevention, and early detection of the disease-related risks. We provided a deep neural network architecture with the aim of identifying the Glaucoma and also its severity levels. A deep learning framework for Glaucoma disease prediction is based in this project on significant CNN, which can get the discriminative features that better represent the hidden models relevant to Glaucoma. This project aims to incorporate a hospital model that will help predict Glaucoma disease as soon as possible using fundus photos. Identifying the disease as early as possible lets the patient get the care needed to keep the disease under control instead of having it go undetected until the very end.

9.2 Future Scope

The future aim of our project is to improve precision by training models with more high-resolution images for each class and if high computing power is given then we can check with more layers in the CNN model and with different activation functions. We want to introduce a protocol to check every fundus image that is created for Glaucoma in hospital so that a patient can take the appropriate steps as soon as possible.

BIBLIOGRAPHY

[1] Septiarini, Anindita et al. "Automated Detection of Retinal Nerve Fiber Layer by Texture-Based Analysis for Glaucoma Evaluation." Healthcare informatics research vol. 24,4 (2018): 335-345. doi:10.4258/hir.2018.24.4.335

[2] Rahul Sharma, Pradip Sircas et al. "Automated Glaucoma detection using center slice of higher order statistics", Journal of Mechanics in Medicine and BiologyVol. 19, No. 01, 1940011 (2019), https://doi.org/10.1142/S0219519419400116

[3] R. Zhao, X. Chen, L. Xiyao, C. Zailiang, F. Guo and S. Li, "Direct Cup-to-Disc Ratio Estimation for Glaucoma Screening via Semi-supervised Learning," in IEEE Journal of Biomedical and Health Informatics.

[4] Septiarini A, Khairina DM, Kridalaksana AH, Hamdani H. , "Automatic Glaucoma Detection Method Applying a Statistical Approach to Fundus Images," Healthc Inform Res. 2018 Jan;24(1):53-60. https://doi.org/10.4258/hir.2018.24.1.53

[5] G. Pavithra, G. Anushree, T. C. Manjunath and D. Lamani, "Glaucoma detection using IP techniques," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, 2017, pp. 3840-3843.

[6] Atheesan S. and Yashothara S., "Automatic Glaucoma detection by using funduscopic images," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, 2016, pp. 813-817.

[7] Guangzhou An Kazuko Omodaka, Kazuki Hashimoto, Satoru Tsuda, Yukihiro Shiga, Naoko Takada, Tsutomu Kikawa, Hideo Yokota, Masahiro Akiba,"Glaucoma Diagnosis with Machine Learning Based on Optical Coherence Tomography and Color Fundus Images",2019 Journal of Healthcare Engineering

[8] N. A. Diptu et al., "Early Detection of Glaucoma Using Fuzzy Logic in Bangladesh Context," 2018 International Conference on Intelligent Systems (IS), Funchal - Madeira, Portugal, 2018, pp. 87-93.

[9] N. Sengar, M. K. Dutta, R. Burget and M. Ranjoha, "Automated detection of suspected Glaucoma in digital fundus images," 2017 40th International Conference on Telecommunications and Signal Processing (TSP), Barcelona, 2017, pp. 749-752.

[10] J. Odstrcilik, J. Jan, R. Kolar, and J. Gazarek, "Improvement of vessel segmentation by matched filtering in colour retinal images," 2009 Springer IFMBE Proceedings of World Congress on Medical Physics and Biomedical Engineering, Munich, Germany, pp. 327-330, Sept 2009.