Chapter-1 INTRODUCTION

The functionality of our eyes determines how well we can see the world around us. The eyes are delicate organs that require routine care to be in good health and perform at their peak. Our eyes need to be taken care of in a variety of ways, including protecting them from harm, maintaining proper hygiene, and seeking expert help when necessary.

1.1 Introduction

There are various conditions that affect the eyes and may result in vision issues or even blindness and such conditions are caused due to eye diseases. From infants to the elderly, they can affect anyone, and they can be brought on by a number of things, such as genetics, environmental influences, and way of life decisions. Refractive errors, cataracts, glaucoma, dry eye, conjunctivitis, age-related macular degeneration (AMD), and age-related macular edema are a few common eye conditions. Blurred or distorted vision, light sensitivity, discomfort, redness, or discharge from the eyes are a few symptoms that these disorders might produce. If a person notices any of these symptoms or changes in their vision, it's crucial to consult a doctor right away so that they can get the necessary diagnosis and therapy. Through proper eye care and routine eye exams, which can spot early disease signs and help maintain their vision and general eye health, many eye diseases can be avoided or managed. In order to correctly diagnose an eye disease, it is important to analyse all the different range of symptoms.

Deep learning (DL), machine learning (ML), and artificial intelligence (AI) have the potential to play significant roles in the prevention, diagnosis, and treatment of eye diseases, according to studies and research done in recent years with the growth in technology. These technologies have aided in the early diagnosis of diseases; AI and machine learning (ML) can be used to analyse medical pictures like retinal scans to find early indicators of eye ailments such diabetic retinopathy, glaucoma and age-related macular degeneration. By being able to identify the characteristics and causes of eye disorders, AI ML also plays a crucial role in individualised therapy, medicine recommendations, and preventative care. This research studies and categorises and

assorts eye illnesses, including cataracts, glaucoma and diabetic retinopathy, using the proposed model.

The following are the description of eye diseases that we are trying to classify using the proposed CNN model:

(1) **Cataract** is a clouding of the natural lens in the eye, which can obstruct or haze vision. It is a typical ailment that frequently develops as a normal part of ageing but can also be brought on by other circumstances. As a person ages, protein breakdown and clumping in the eye's lens can result in cloudiness and opaqueness in the lens. The risk of getting cataracts can rise when people are exposed to UV radiation from the sun, tanning beds, or other sources. Changes in blood sugar levels in people with diabetes increase their chance of acquiring cataracts. The risk of cataracts can be raised by several drugs, including corticosteroids. Despite the fact that cataracts can arise for a number of reasons, surgery can be used to treat them.

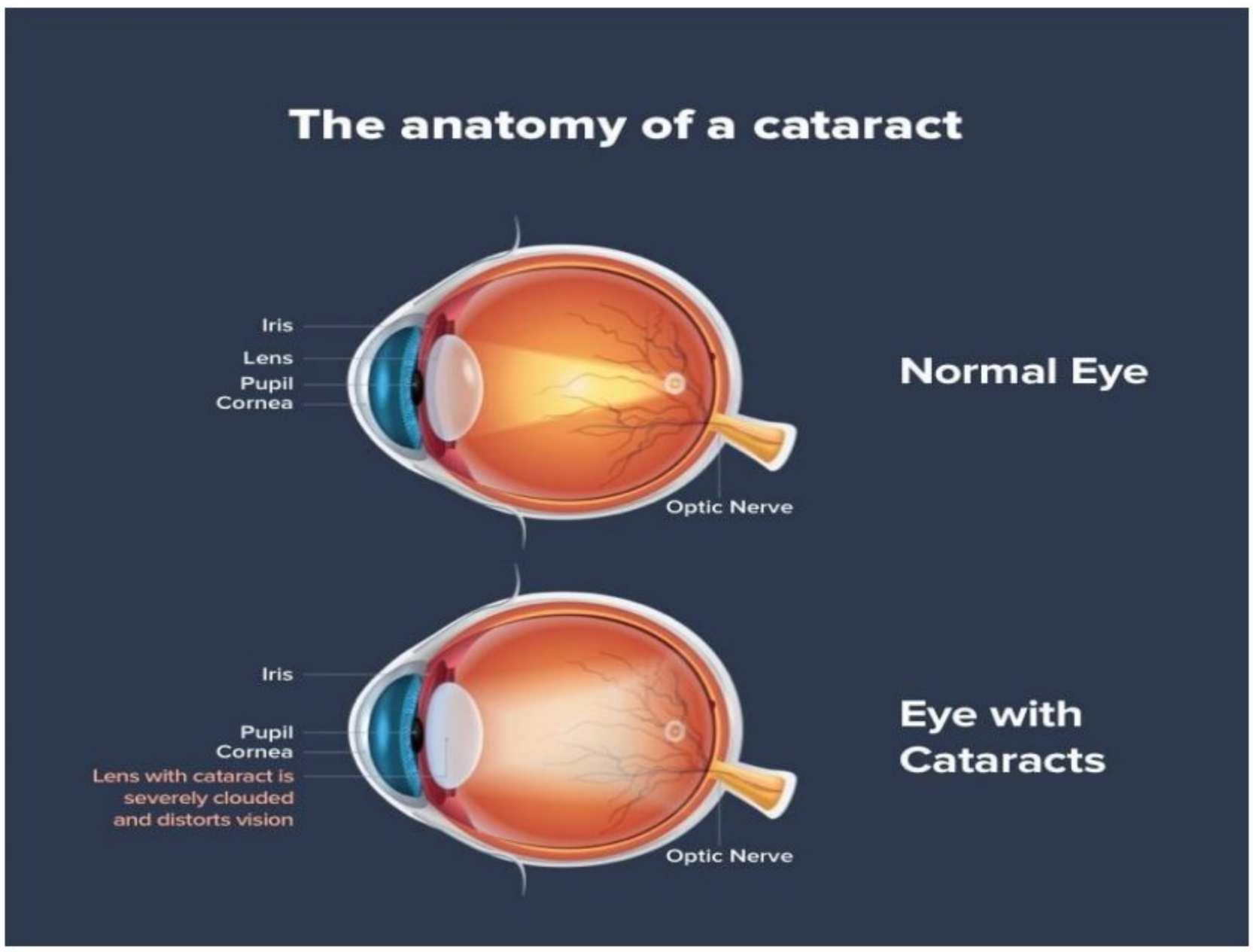

Fig 1.1 : Normal Eye vs Cataract Eye anatomy

(2) **Diabetic retinopathy**, is a disease which generally occurs to people with Type 1 and Type 2 diabetes, as the name suggests it is an impediment of diabetes, in which blood vessels that are found in the light sensitive tissue at the back of the eye are affected the most.

Although the exact cause of diabetic retinopathy is unknown, high blood sugar levels over a longer time can harm the blood vessels in the retina. This harm can result in blood vessels that leak or become blocked, which can impair vision or even result in blindness. If a person doesn't care of his or her blood sugar levels or does not properly manage or regulate their own blood sugar levels, or even high blood pressure as well as raised cholesterol levels, and smoking can become other major causes of diabetic retinopathy. Generally this aimenet is a greater risk for persons with a longer time period of diabetes.

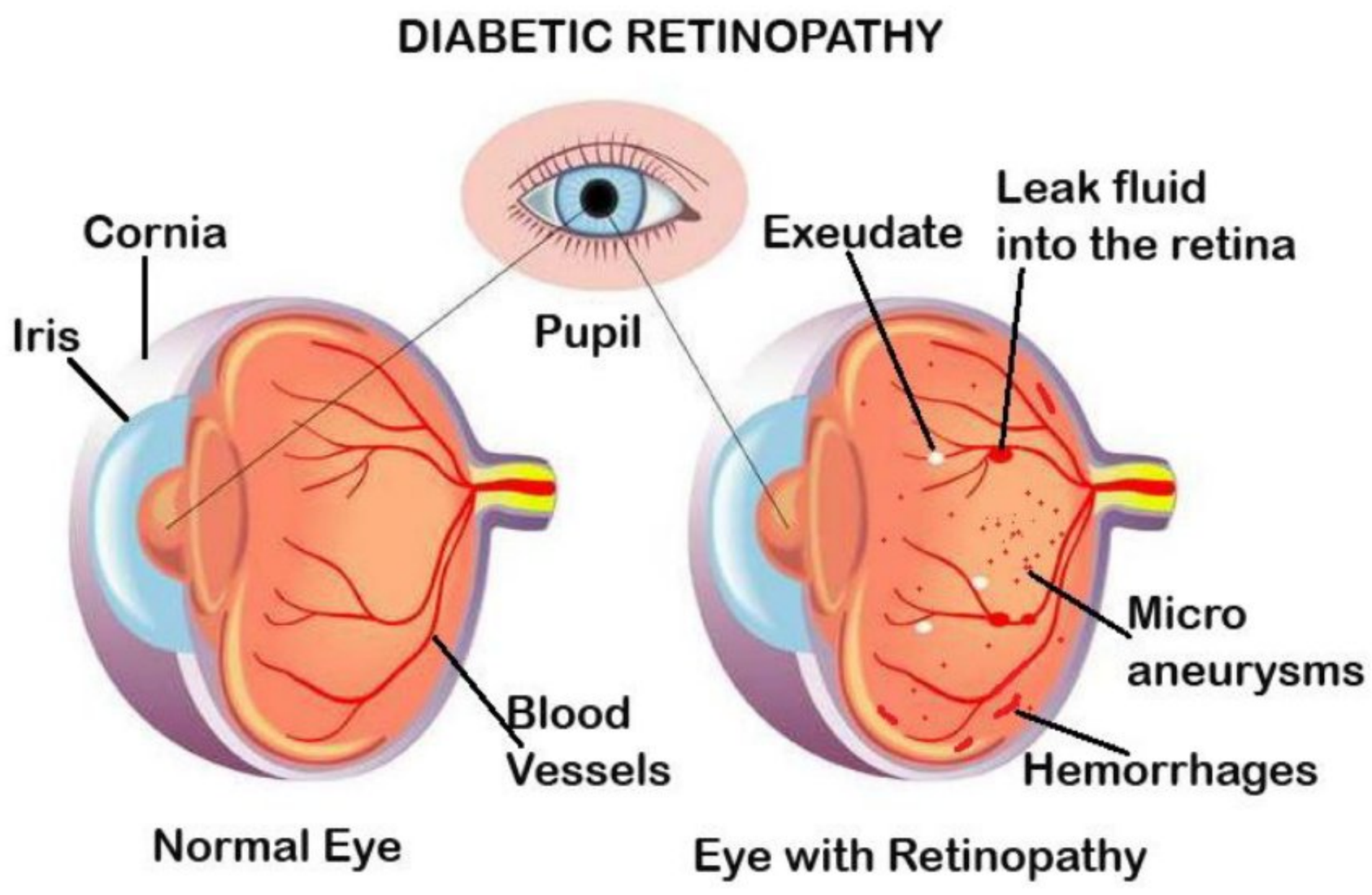

Fig 1.2: Normal Eye VS Diabetic Retinopathy Anatomy

(3) **Glaucoma,** it is the disease in which the optic nerve is affected. Optic nerve is the nerve which connects the eye and the brain of a person. Although it can happen with normal or low intraocular pressure (IOP), it is frequently related to elevated IOP. Globally, glaucoma is one of the main causes of permanent blindness. Although the precise cause of glaucoma is unknown, it is believed to be a result of

a combination of genetic and environmental factors. The additional causes of glaucoma are age, genetics, ocular injuries or surgeries, and long-term corticosteroid use. Regular eye exams are essential for detecting and tracking the progression of glaucoma since early diagnosis and treatment can reduce the risk of vision loss and improve results.

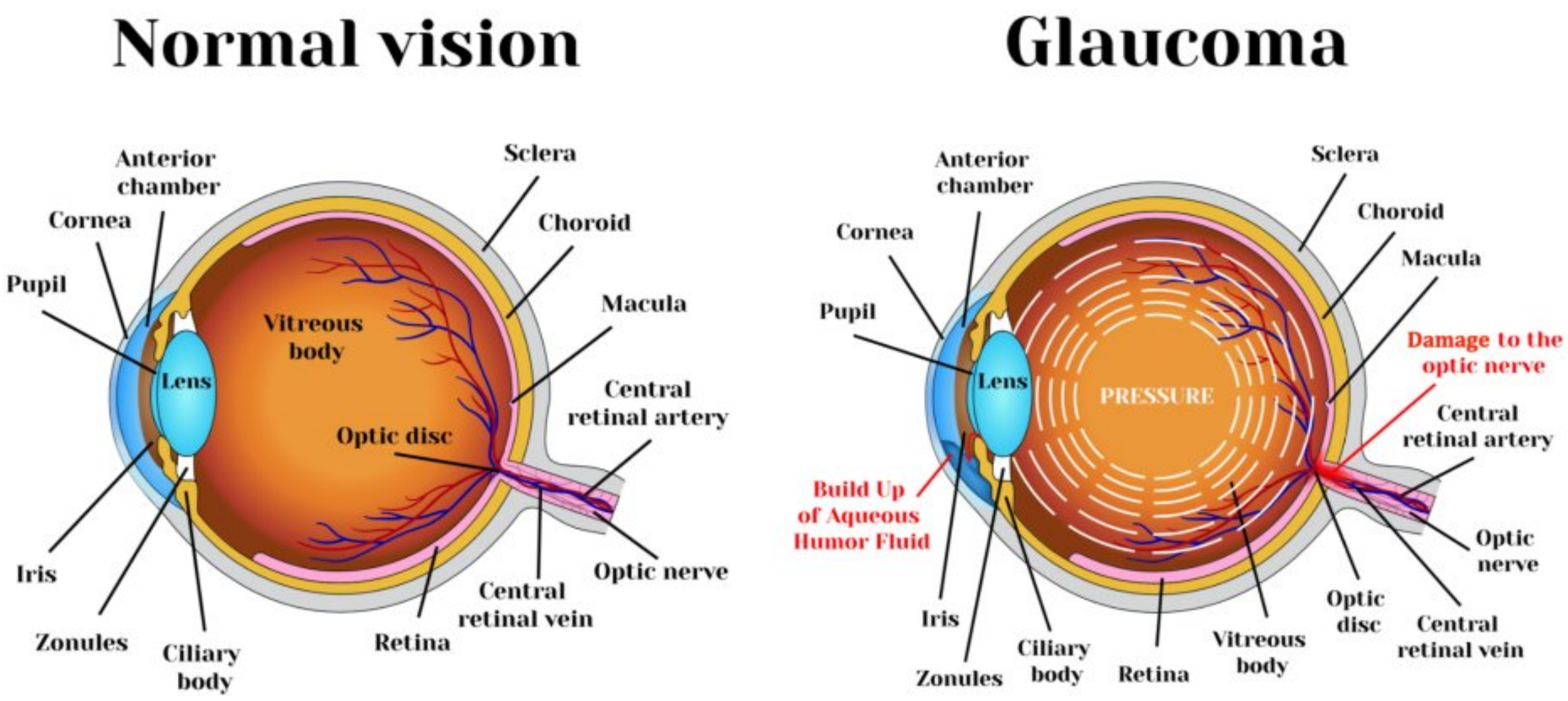

Fig 1.3: Normal Eye vs Glaucoma Anatomy

Traditional methods for predicting eye diseases can be time-consuming, subjective, and subject to human error, such as manual examination and picture interpretation. Inaccurate or inconsistent outcomes might result from this, as well as delays in diagnosis and treatment. On the contray, artificial intelligence (AI), deep learning-based techniques, can offer several benefits over traditional techniques, as they are quicker and more efficient, show improved accuracy, aid in early detection and individualised treatments, and AI algorithms can provide consistent and objective analysis of medical images, lowering the risk of human error and variability. Therefore, in this study, creation of a deep learning based CNN model is presented, that will aid in the classification of eye diseases.

1.2 Problem Statement

People frequently neglect their eye irritations or changes until they become urgent when it comes to the human eye, unless there is a vision issue. The most probable cause for this issue worldwide is - absence of such an application which is explicitly designed for identifying prior or premature signs of disorders that occurs in the eye, if such an application must have existed, then the patients could have got appropriate information regarding such ailments and they could have better assessed treatment for the same. This research's methodology addresses the above problem by enlightening the end user to examine his own eye to educate them more about their eye ailment and its current state. Conventional procedures for diagnosing eye problems are expensive, time-consuming, and call for highly specialised knowledge. CNNs have demonstrated promising results in a variety of image-based medical applications, including the diagnosis of eye disorders, thanks to breakthroughs in deep learning techniques. However creating a CNN-based system for diagnosing eye disorders that is accurate and effective involves overcoming a number of obstacles, including choosing the right architecture, correcting class imbalances, coping with noisy and low-quality images, and training the model on a huge and varied dataset. Consequently, the goal of this research is to create a solid and trustworthy CNN-based system for the rapid and widespread screening and monitoring of patients in clinics and hospitals, as well as the early detection and diagnosis of a variety of eye disorders.

1.3 Objectives

The three main objectives of this project are:

- The goal of the project "Eye diseases classification using deep learning" is to develop automated and precise methods for the early detection and diagnosis of different eye disorders utilising medical images such as eye retinal images.

- The objective is to lessen ophthalmologists' workload, enhance the precision and effectiveness of screening and diagnosis, and eventually enhance patient outcomes.

- Using Convolutional neural networks (CNNs), which is a type of deep learning technique, this research will be demonstrating three different types of eye diseases and inform patients beforehand so they can take required treatment at the right

time.

- Using deep learning techniques, to result in more effective patient management and treatment plans.

1.4 Methodology

The main methodology of this project revolves around using CNN for successful identification of four different categories of eye diseases.

six main steps that aides in this are :

(1) Collecting the database.

(2) Preprocessing the dataset.

(3) Splitting the dataset into training and testing.

(4) Making of the CNN model.

(5) Model evaluation and results.

1.5 Language Used

Python 3 - The main reasons for the use of this programming language for the implementation of the project are stated below:

Simple and dependable:

Python generates readable and condensed code. Python's simplicity enables programmers to create solid solutions because machine learning, deep learning and artificial intelligence are based on complex algorithms and adaptable workflows. Developers can concentrate solely on resolving an DL, ML issue rather than on the language's finer technical details.

Extensive selection of libraries and frameworks:

Many modules and frameworks are available for Python. Programmers use these frameworks and libraries to speed up development. Here are a few examples:

Deep learning frameworks include Keras, TensorFlow, and Scikit-learn.

- For scientific computing and data-analysis we can use numpy.

- Matplotlib is used for creating static, animated, and interactive visualisations.
- Pandas is a data analysis tool that can be used for a variety of purposes.
- Keras is used for adding layers and neurons in our CNN model.
- Tensorflow is used for training and inference of our deep neural network model.

Independency of platform

A programming language or framework that is "platform independent" enables developers to write things on one system and then use them with little (or no) modification on another. The popularity of Python is due to the fact that it is a platform-neutral language. It can be used with a number of operating systems, including Windows, Linux, and macOS. Python software can be disseminated and utilised without the requirement for a Python interpreter by using Python code to create standalone executable programmes for the majority of popular operating systems.

This study has employed Python as a programming language for this project. Python contains a large number of deep learning libraries, which expedites method development and makes it easier. These libraries include a range of deep learning techniques as well as GPU support to reduce training times, such as Keras, TensorFlow, etc.

The proposed model is built on the Jupyter notebook. Jupyter notebook provides a server-client programme that enables users to edit and run all their notebook papers through a web browser. This Jupyter notebook can be made to run locally on the desktop without any internet connection and it can be deployed on a remote server as well that can be made accessible via the internet.

The Jupyter notebook provides a computational engine that runs the code in a notebook document and this engine is known as kernel. Python code is generally executed by the ipython kernel, which is mentioned in the detailed documentation, the link for the which is provided in the references. Official kernels are available for a wide variety of other languages.

Whenever a person opens anaconda and Jupyter notebook, then the notebook document's related kernel is automatically launched. To generate results, the person has to execute run the notebook, which can be done cell by cell or through the menu Cell-> Run All

option and this will execute the file. Depending on the type of computations, the kernel may use lot of CPU and RAM, and the RAM is not released until the kernel is terminated.

1.6 Organisation

The organisation of the report is as following:

In chapter 1, the introduction of eye diseases and factors causing these diseases has been discussed. The chapter presented the importance of eye disease prediction and identification and research methodology for eye diseases prediction

In chapter 2,the literature survey has been discussed. This chapter gives a brief introduction and an abstract of various researches which were previously carried out by other researchers around the globe for the same, and brief about their techniques and approaches to their limitations. Also a comparison table is made for the same purpose.

In chapter 3,the system development has been discussed. This chapter discusses the DL algorithm used and a brief explanation about artificial neurons, forward and backward propagation and CNN. How the data set is first imported and then split into training and testing sets and then the CNN model is trained on them for the purpose of testing new dataset and then results are predicted. Using an accuracy metric, performance analysis is depicted.

In chapter 4, this chapter has discussed the performance of the model for which an accurate use of accuracy metric is done, this chapter also depicts here the losses this model made while training and how it achieved the final accuracy after several rounds of forward and backward propagation.

In chapter 5, in this chapter the conclusion of this project with the discussion on the algorithm used to build this project and proposed solutions with the brief of existing limitations of this project and future scope of the project, are being discussed.

Chapter-2 LITERATURE SURVEY

Mohammad Monirujjaman Khan et al. (2022) the study presented in this research puts forward a fanatical ocular detection method which is mainly based on deep learning. For this study, they picked a cutting-edge image classification algorithm like VGG-19 and trained the model on the ODIR dataset, which contains 5000 photos. The photos consist of eight different classes of the fundus images. Many varieties of ocular illnesses are constitued by these classes. Nevertheless the dataset for these classes they picked was very imbalanced. And to solve this problem they sought to turn this multiclass classification problem into a binary classification problem and take the same number of images for both categories in order to solve this challenge.The accuracy of the VGG-19 model was 98.13% for the normal (N) versus pathological (M) myopia class, 94.03% for the normal (N) against cataract (C), and 90.94% for the normal (N) versus glaucoma classes (G). [1]

Sushma K Sattigeri et al. (2022) their study provides a novel approach to provide an automated eye illness identification model utilising visually discernible symptoms. It uses deep learning techniques like convolution neural networks and digital image processing techniques like segmentation and morphology. Using the suggested procedure, four eye diseases—crossed eyes, bulging eyes, cataracts, uveitis, and conjunctivitis—are examined and grouped.The dataset for the purpose was collected partially from kaggle and partially from local optometrist's assistance. Two different models are used to train for single-eye and dual-eye pictures. Using two-eye pictures, one model forecasts disorders including crossed eyes and bulging eyes. The eye configuration of single eye achieved accuracy of 96% and that of two eye achieved 92.31%.[2]

Nouf Badha, et al.(2022). This study focuses on identifying eye infections of Glaucoma disease and for this purpose the authors of this paper used a variety of machine learning (ML) classifiers, including Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes (NB), Multi-layer perceptron (MLP), Decision Tree (DT), and Random Forest (RF), as well as Deep Learning (DL) models, such as Convolutional Neural Network (CNN) based on Resnet152 model.. On the dataset for Ocular Disease Intelligent Recognition, the proposed technique is evaluated. The collected findings

demonstrated that, in comparison to the other ML classifiers, the RF and MLP classifiers had the best accuracy of 77%. For the identical task and dataset, the deep learning model (CNN model: Resnet152) offers an even higher accuracy of 84%. [3]

Grzegorz Meller,et al.(2020). The main objective of this project is to delineate a convolutional neural network-based advanced classification model that can recognise eye disorders from images of eye-fundus. This model will be able to distinguish various eye diseases. It made use of the "Ocular Disease Intelligent Recognition (ODIR)" dataset, a structured ophthalmology database of 5,000 individuals with age, colour fundus pictures of the left and right eyes, and doctors' diagnostic keywords. Patient labels include normal (N), diabetes (D), glaucoma (G), cataract (C), AMD (A), hypertension (H), myopia (M), and various diseases/abnormalities (O).Using a relatively simple network and only images labelled as N (normal) or C (cataract), a simple CNN model was trained to determine whether an eye had a normal fundus or a cataract. After 12 epochs , the model achieved 93% accuracy, and when the experiment was run on the entire ODIR dataset, 50% validation accuracy was attained. [4]

Hao Gu et al. (2020) In this paper demonstrates a design of a novel approach that is based on hierarchical deep learning network is proposed. This network aids in identification of corneal diseases and is composed of a family of multi-task and multi-label learning classifiers which represents different levels of eye diseases which are indeed derived from a predefined hierarchical eye disease taxonomy.Then, in order to understand the fine-grained heterogeneity of eye illness traits, they presented a multi-level eye disease-guided loss function. Using a retrospective dataset of 5,325 ocular surface pictures, the proposed algorithm was trained directly on an end-to-end basis. In a prospective, a dataset which is majorly clinic-based consists of 510 outpatients who are freshly enrolled with diseases of infectious like keratitis, non-infectious keratitis, corneal dystrophy or degeneration, and corneal neoplasm, the algorithm's performance was lastly evaluated against 10 ophthalmologists. For each form of corneal disease, the algorithm's area under the ROC curve was over 0.910, and generally speaking, its sensitivity and specificity were on par with or better than the average values of all ophthalmologists.[5]

Pawan Kumar Upadhyay et al.(2022).The automatic detection of retinal disorders using OCT pictures is one such deep learning application that is covered in this work. The new method for detecting retinal diseases has been proposed, and it effectively handles the four-class problem by differentiating between the images of choroidal neovascularization

(CNV), diabetic macular edema (DME), DRUSEN, and NORMAL class. The pre-trained VGG-16 model was pruned in order to build more precise neuron connections, develop more optimised layers, and perform important tasks for occult disease detection in the proposed network (CCNN). The model achieved 97.16% accuracy.

Author(s)/Reference	**Journal/ conference, year**	**Methodology**	**Disadvantage**
Mohammad Monirujjaman Khan at el/1	2022	A deep learning based approach- VGG-19 is used to classify ODIR dataset into multiple classes. Dataset has 5000 images of fundus	The dataset with the classes was highly unbalanced.
Sushma K Sattigeri et al/2	2022	A deep learning based approach -CNN is used to classify a dataset picked from kaggle of 5 types of eye diseases.	The neural classifier model was fed images for training with different informational indexes from images.
Nouf Badha et al/3	2022	ML based algorithms such as KNN, SVM, NB, DT, RF are used to detect human eye infections of Glaucoma disease. Later DL based CNN Resnet152 model is used for better accuracy.	The machine learning models picked for the classification of ODIR images performed poorly and cannot be relied on for real medical diagnosis..

Grzegorz Meller et al/4	2020	Deep learning based CNN model is used for the purpose of classifying ODIR images dataset.	Many image variants or classes could not have their representation in the validation set of the model.
Hao Gu et al/5	2020	A novel hierarchical deep learning network - CNN is used for the purpose of classifying ocular surface images consisting of corneal diseases	The algorithm is trained exclusively to detect the ocular surface diseases listed in the methods section, so it may miss eye diseases with a normal ocular surface.
Pawan Kumar Upadhyay/ 6	2022	A Coherent Convolution Neural Network (CCNN) is used for the purpose of classifying four different classes of eye diseases.	There are cases where models fail to detect the retinal disease accurately as the model accuracy relies on tuning of hyperparameters.

Table 2.1 : Comparison of various Research Papers

Chapter-3: SYSTEM DEVELOPMENT

3.1 Feasibility Study

The purpose of the feasibility study is to assess if the project is feasible—that is, whether it can be executed effectively and conveniently—from an operational, technical, economic, and organisational perspective.

1. Economic Feasibility:

The project was created exclusively with open-source software and libraries. The creators of these libraries have all made them freely accessible online. As a result, there are no costs associated with project development.

2. Operational Feasibility:

This is a feasible operating concept that could help prediction of eye diseases.

1.5.1 Software Requirements:

Python 3 was used to build the project for the reasons outlined above. Kaggle was used to collect the dataset. GitHub and StackOverflow were consulted as resources in the case that programming grammar errors occurred. An open-source, high-GPU Jupyter Notebook or an interface similar to that - called Google Colaboratory. A completely cloud-based, setup-free Jupyter notebook environment, Google Colab is available for free. With Colab, users may access advanced computational resources from the browser, write and run code, store and share analysis, and more. Python data analysis scripts can be iterated and written using Jupyter Notebook. Code lines can be written and executed one at a time rather than having to write and rewrite an entire programme.

3.2 Model Development

The process of creating an DL model includes gathering of data from various trustworthy sources and processing it by an DL model to make it suitable for modelling, designing the model, developing the model's algorithm, and calculating performance metrics, selecting

the best performance parameters for the model evaluation. Once the model is put into use, model maintenance becomes extremely important.

3.3 SFD Diagram and DFD Diagram

3.3.2 System Flow Diagram

System flow diagram is one of the very famous ways of graphical depictions in which data flow in any system is designed and it is based on software engineering practices being followed by developers and researchers. The diagram shows a series of steps that describe where the input and output for the system come from and where they go to as part of the diagram. The graphic makes it possible to manage how the system decides which events to process and how data enters the system. The system flow diagram, which leaves out the minor components and shows the important components of the system in order of importance, is essentially a visual depiction of data flow.

With its visual representations and sequential overall process, it offers a lot of advantages to both the user and the organisations.

- Different engineers and programmers utilise system flow diagrams to visualise decisions and potential results from the system, assisting system designers in designing in accordance with it.
- System flow diagrams employ symbols that are very simple as well as effortless for end users to understand, showing how the process works and where choices are made.
- The diagrams assist in visualising the key inputs for a system that considers and generates the key output utilising key steps, which is advantageous for the organisation for their system requirements.
- Many well-known organisations utilise system flow diagrams for business objectives since they are relatively simple to grasp thanks to the use of symbols, which removes the difficulty of utilising complicated processes or codes that are typically seen using a sophisticated algorithm or in pseudo-codes.

- A keyboard, mouse, or other types of sensors serve as a flow diagram's data inputs, while a monitor or printer serves as the output device through which we see the results.
- System flow diagrams are applied in many real-world applications, such as automatic washing machines and airline reservation systems, to efficiently handle all system processes.

The following is the SFD of this study:

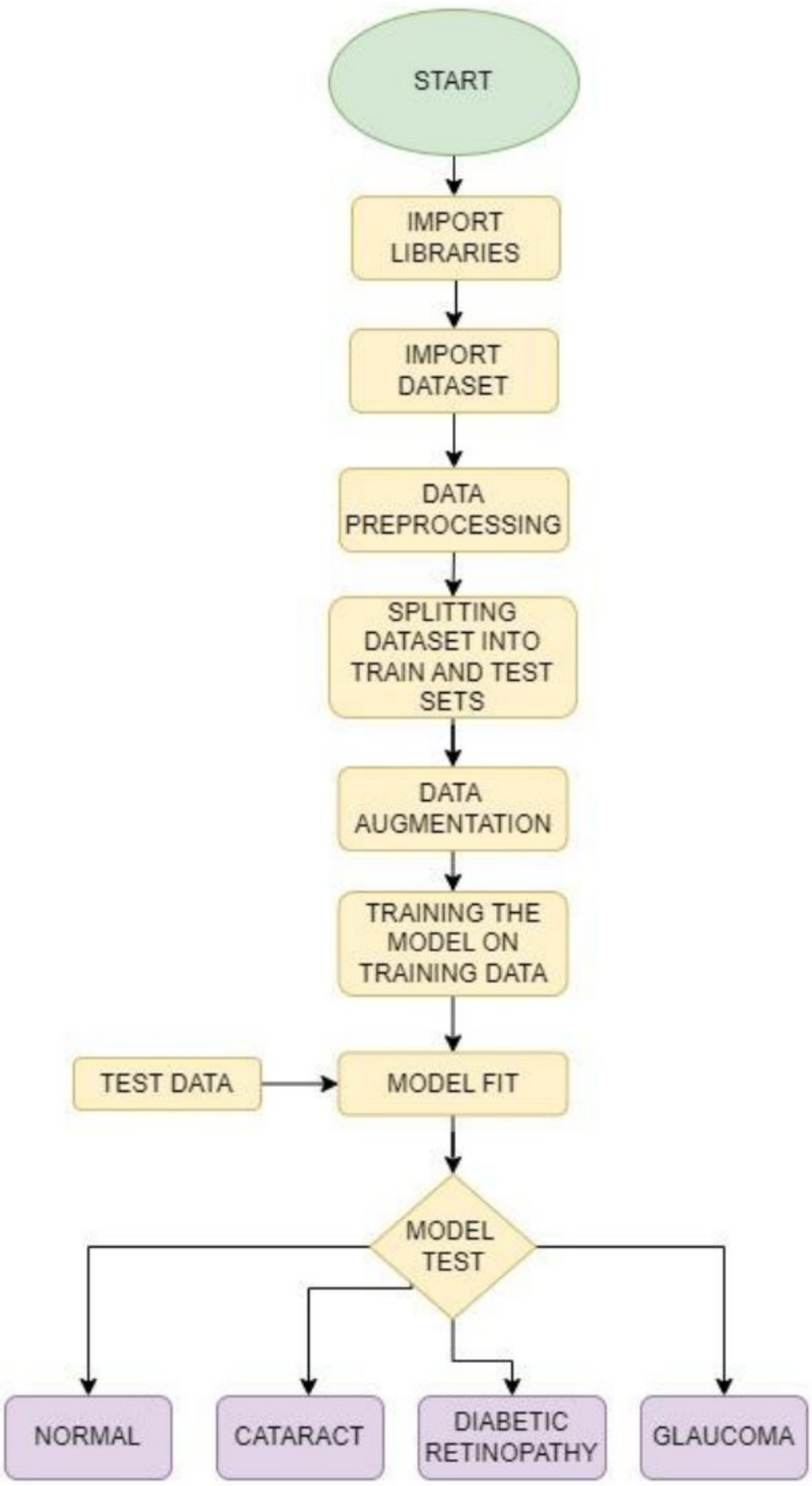

Fig 3.1: System flow diagram

3.3.3 Data Flow Diagrams

The Data flow diagram (DFD) is a prototypical visual representation which tracks and represents the movement and overall flow of data through a proposed system of a study or research or a new method. An orderly and comprehensible DFD can graphically represent the pertinent quantity of the system demand. It can be drawn manually, or automatically with the advent of AI based softwares available online, or both.

Data flow diagrams denotes the flow of information, by depicting the entrance of the the data to the system, as well as its exit from the system, it also denotes the system or the method which is generally a process that modifies that data, as well as depicts the place where the information is kept.

One of the major objective of DFD is to outline the boundaries and scope of the system as a whole. It can be best utilised in cases of system redesign or in cases of understanding the foundation of the system from the user point of view, as it can act as a communication tool between the user that is any participant and a system analyst or architect as it will serve them information in the sequence required. Many a times a DFD is also coined as a bubble chart or data flow graph.

Each level of abstraction for a system or piece of software can be performed using the DFD. In reality, DFDs can be divided into stages that signify a growth in the flow of information and functional granularity. In DFD, levels are denoted by the numbers 0, 1, or 2. The data flow diagram in this study has made use of three main levels: 0-level DFD, 1-level DFD, and 2-level DFD.

3.3.3.1 Data Flow Diagram - Level 0

The complete programme need is shown as a single bubble in the context diagram, sometimes referred to as the core system model, with input and output data marked by incoming and exiting arrows. The system is then broken down and represented as a DFD with several bubbles. The system components that each of these bubbles represents are then broken down and documented as ever-more-detailed DFDs. To ensure that the programme at hand is thoroughly grasped, this step may be performed as many times as necessary. Maintaining the

amount of inputs and outputs between levels is crucial; this idea as levelling was given by DeMacro.

The following image depicts the Level 0 DFD for this study:

Fig 3.2: DFD Level 0

3.3.3.2 Data Flow Diagram - Level 1

The key processes identified in the level 0 DFD are divided into sub-processes at this level to give a more in-depth picture of the system. On the level 1 DFD, each subprocess is represented as a distinct process. Additionally, each sub-process's data flows and data stores are displayed.

The level 1 DFD for this study is as shown:

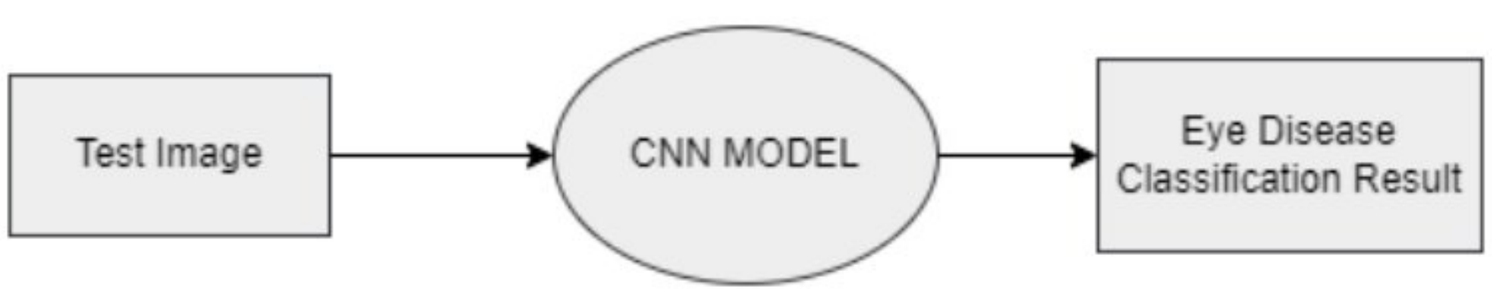

Fig 3.3: DFD Level 1

3.3.3.3 Data Flow Diagram - Level 2

By further subdividing the sub-processes mentioned in the level 1 DFD, this level offers an even more in-depth look at the system. On the level 2 DFD, each subprocess is represented as a distinct process. Each sub-process's data flows and data stores are also displayed.

The level 2 DFD for this study is as shown:

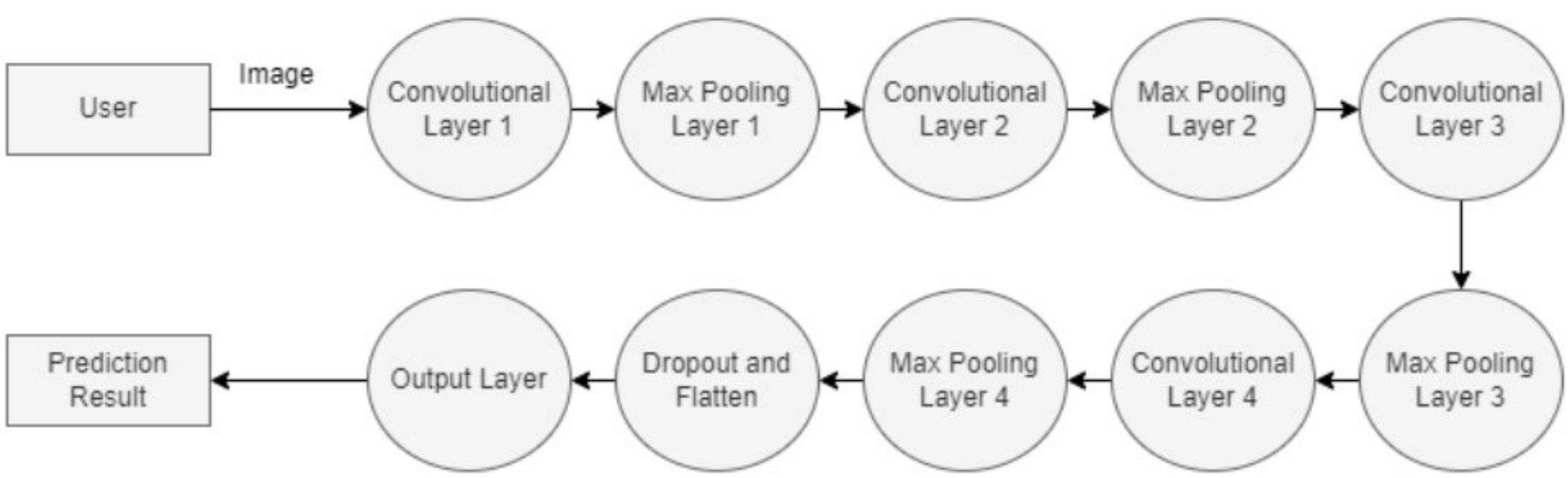

Fig 3.4: DFD Level 2

3.4 Data Set Used in the Major Project

For the purpose of this study, the dataset used for model training is picked up from kaggle. The dataset consists of four categories of retinal images out of these, three categories are of different eye diseases - cataract, glaucoma and diabetic retinopathy and the fourth category is of normal retinal eye image. These images are collected from various sources like IDRiD, Oculus recognition, HRF etc. There are a total of 4217 images, out of which 1038 images are of cataract, 1098 images of diabetic retinopathy, 1007 images of glaucoma, 1074 of normal. The retinal images consist of left and right eye images in almost equal proportions. The total size of the dataset is 772.22 MB

The Kaggle.com site's dataset was gathered there. Kaggle is a branch of Google and is a website community for data scientists and ML practitioners. Customers can browse and post data sets on Kaggle, explore and build models, interact with other data scientists, and compete to solve any data science challenges. Because kaggle is a reputable website with over a million users globally, one can be sure that the dataset is unique and authentic.

3.5 Algorithm/Pseudo code of the model

In order to analyse and classify the eye diseases DL based CNN model has been employed, successfully built using tensorflow aided keras library. The following is the proposed model's pseudocode :

3.5.1 Pseudocode of the proposed method:

i. Importing the libraries

ii. Importing dataset

iii. Data Preprocessing all the images are reshaped and recolored into one uniform size of 180px X 180px and RGB colour respectively.

iv. Splitting the dataset into training, validation and test sets The retinal images are split into 80% training and 20% test sets precisely 3374 retinal images into training set and 843 retinal images into test set.

v. Data augmentation: Random rotation, contrast and zooming is done

vi. Training the model: Keras sequential method is used for the purpose of adding layers.

vii. Evaluation of model

The following is the screenshot that depicts importing of all libraries:

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
import cv2, PIL, glob, pathlib
import tensorflow as tf
from tensorflow import keras
from keras import models, layers
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sn
```

Fig 3.5: Importing Libraries

The following is the screenshot that depicts importing of dataset :

```
In [2]: for dirname, _, filenames in os.walk(r'C:\Users\mohit\Desktop\MajorProject2'):
            for filename in filenames:
                pass
```

```
In [4]: normal = pathlib.Path(r"C:\Users\mohit\Desktop\MajorProject2\dataset\normal")
        glaucoma = pathlib.Path(r"C:\Users\mohit\Desktop\MajorProject2\dataset\glaucoma")
        DiabRetino=pathlib.Path(r"C:\Users\mohit\Desktop\MajorProject2\dataset\DiabRetino")
        cataract = pathlib.Path(r"C:\Users\mohit\Desktop\MajorProject2\dataset\cataract")
```

```
In [5]: images_dict = {"normal": list(normal.glob("*.jpg")),
                      "glaucoma": list(glaucoma.glob("*.jpg")),
                      "DiabRetino":list(DiabRetino.glob("*.jpeg")),
                      "cataract":list(cataract.glob("*.jpg"))}
        labels_dict = {
            "normal":0, "glaucoma":1, "DiabRetino":2, "cataract":3
        }
```

Fig 3.6: Data Preparation and importing the dataset

The following is the screenshot that depicts preprocessing of data:

```
In [7]: X, y = [], []
        for label, images in images_dict.items():
            for image in images:
                image = cv2.imread(str(image))
                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                image = cv2.resize(image, (180, 180))
                if image is not None:
                    X.append(image)
                    y.append(labels_dict[label])

In [8]: X = np.array(X)
        y = np.array(y)

In [10]: X = X/255
```

Fig 3.7: Data Preprocessing

The following figure depicts the split in dataset into training and testing sets

```
In [11]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

In [12]: X_train = X_train.astype("float32")
         X_test = X_test.astype("float32")
```

Fig 3.8: Splitting the data set into training and testing sets

The following is the depiction of data augmentation:

```
In [15]: #data augmentation
         data_argumentation = keras.Sequential([
             keras.layers.experimental.preprocessing.RandomRotation(0.2),
             keras.layers.experimental.preprocessing.RandomContrast(0.3),
             keras.layers.experimental.preprocessing.RandomZoom(0.3),
             keras.layers.experimental.preprocessing.RandomZoom(0.7)
         ])
```

Fig 3.9: Data Augmentation

The following figure shows model training and compilation:

```
In [16]: model = keras.Sequential([
             data_argumentation,
             layers.Conv2D(64, (5, 5), padding="same",   input_shape=(180, 180, 3), activation="softmax"),
             layers.MaxPooling2D(),
             layers.Conv2D(32, (5, 5), padding="same", activation="relu"),
             layers.MaxPooling2D(),
             layers.Conv2D(16, (5, 5), padding="same", activation="relu"),
             layers.MaxPooling2D(),
             layers.Conv2D(8, (5, 5), padding="same", activation="relu"),
             layers.MaxPooling2D(),
             layers.Dropout(0.2),
             layers.Flatten(),
             layers.Dense(50, activation="sigmoid"),
         ])
         model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=['accuracy'])
```

Fig 3.10: Building the model, and compilation

The following shows the model is trained on training data on 150 epochs.

```
In [17]: history = model.fit(X_train, y_train, epochs=150)
         101/101 [==============================] - 203s 2s/step - loss: 0.8224 - accuracy: 0.6519
         Epoch 142/150
         101/101 [==============================] - 203s 2s/step - loss: 0.8192 - accuracy: 0.6625
         Epoch 143/150
         101/101 [==============================] - 185s 2s/step - loss: 0.8226 - accuracy: 0.6560
         Epoch 144/150
         101/101 [==============================] - 176s 2s/step - loss: 0.8102 - accuracy: 0.6778
         Epoch 145/150
         101/101 [==============================] - 177s 2s/step - loss: 0.8154 - accuracy: 0.6613
         Epoch 146/150
         101/101 [==============================] - 177s 2s/step - loss: 0.8139 - accuracy: 0.6691
         Epoch 147/150
         101/101 [==============================] - 178s 2s/step - loss: 0.8277 - accuracy: 0.6535
         Epoch 148/150
         101/101 [==============================] - 188s 2s/step - loss: 0.8191 - accuracy: 0.6659
         Epoch 149/150
         101/101 [==============================] - 176s 2s/step - loss: 0.8180 - accuracy: 0.6638
         Epoch 150/150
         101/101 [==============================] - 177s 2s/step - loss: 0.7964 - accuracy: 0.6697
```

Fig 3.11: Model training for 150 epochs on train set

The following figure shows model summary:

```
In [18]: model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 180, 180, 3)       0

 conv2d (Conv2D)             (None, 180, 180, 64)      4864

 max_pooling2d (MaxPooling2D  (None, 90, 90, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 90, 90, 32)        51232

 max_pooling2d_1 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 45, 45, 16)        12816

 max_pooling2d_2 (MaxPooling  (None, 22, 22, 16)       0
 2D)

 conv2d_3 (Conv2D)           (None, 22, 22, 8)         3208

 max_pooling2d_3 (MaxPooling  (None, 11, 11, 8)        0
 2D)

 dropout (Dropout)           (None, 11, 11, 8)         0

 flatten (Flatten)           (None, 968)               0

 dense (Dense)               (None, 50)                48450

=================================================================
Total params: 120,570
Trainable params: 120,570
Non-trainable params: 0
_________________________________________________________________
```

Fig 3.12: Model Summary

The following depicts the test accuracy:

```
In [20]: score = model.evaluate(X_test, y_test)
```

```
26/26 [==============================] - 7s 258ms/step - loss: 0.7893 - accuracy: 0.6741
```

```
In [21]: print('Test accuracy:', score[1]*100)
```

```
Test accuracy: 67.41293668746948
```

Fig 3.13: Test Accuracy

The following figure shows the overall architecture of training of model's subsequent layers:

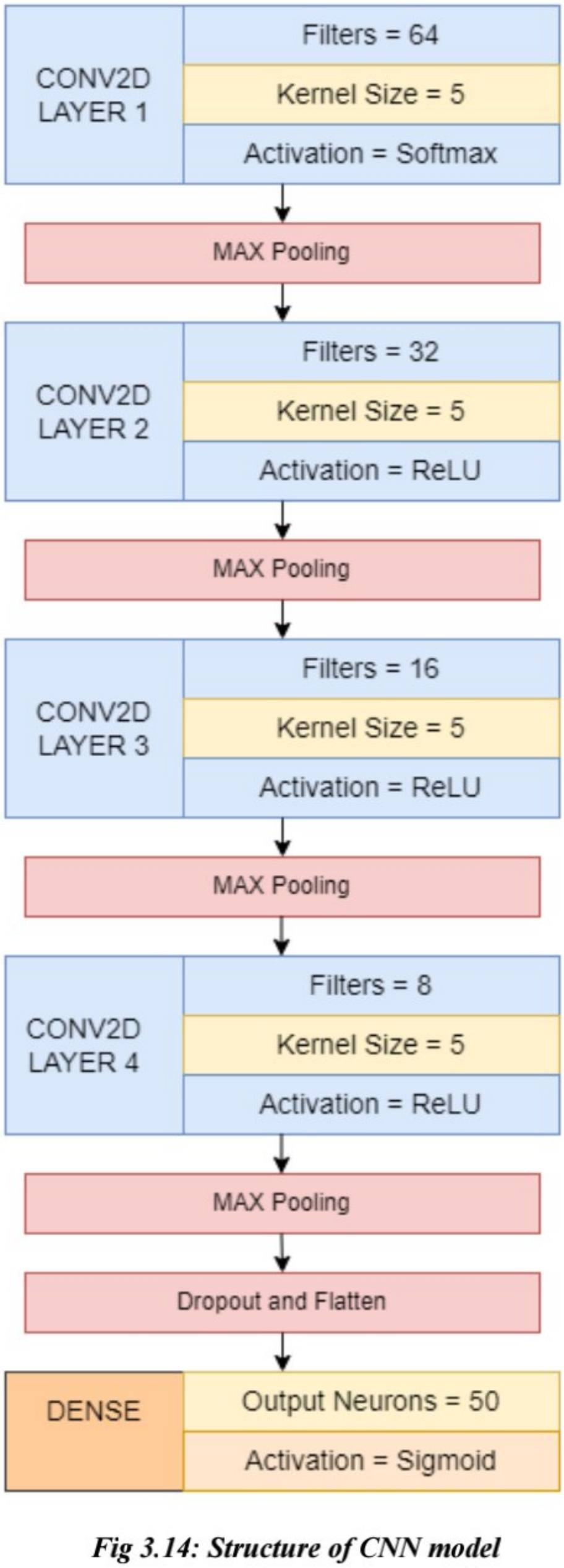

Fig 3.14: Structure of CNN model

3.5.2 Data Augmentation and its importance

Data augmentation is a very impressive and efficient technique which makes use of a method that creates additional data points from the existing data; this approach helps in enhancing the overall amount of data.For the purpose of creating new data points in the deep learning models, this approach makes some minor changes in the section already existing imported dataset. The process of artificially deriving fresh data from previously collected training data is known as data augmentation. Techniques include cropping, cushioning, flipping, rotating, and resizing.This method helps in strengthening the overall performance of the model as well as problems of overfitting and missing values of data are avoided very efficiently. For the purpose of training the proposed model data augmentation is performed, using the keras libraries experimental library - preprocessing method. And random rotation, contrast and zooming techniques are employed.

3.5.3 Artificial Neural Network

The phrase "artificial neural network" refers to a branch of artificial intelligence that was inspired by biology and is based on the brain. Artificial Neural networks is a man made or as the name suggests artificial computational network which is based on the biological neural network of the humans, which are the basic building blocks of structure of human brain. This neural network works in parallel design constructs as that of the human brain, as an ANN also make use of neurons that are further linked to each other in different layers of a network. In AI,these neurons are termed as nodes. The following image depicts the human neuron and the artificial neuron:

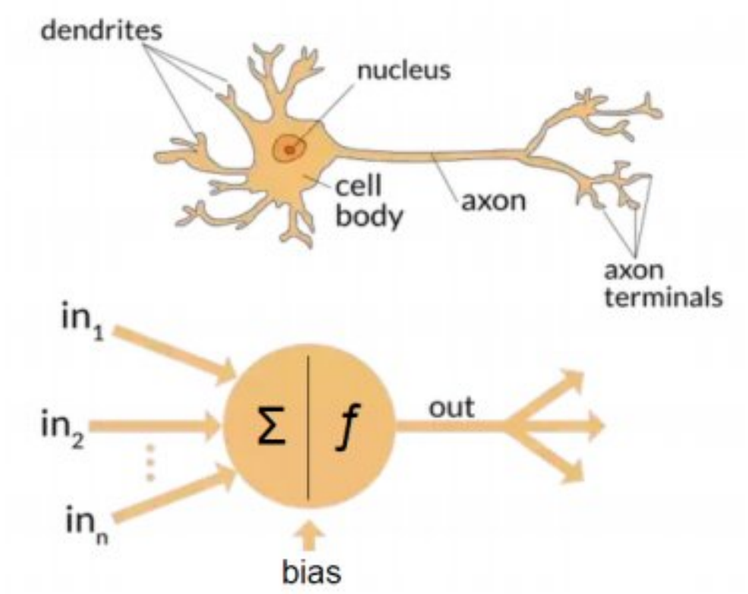

Fig 3.15: Human Vs Artificial Neuron

The following table shows the relationship between biological neuron and an artificial neuron:

Human Neuron	**Artificial Neuron**
Cell nucleus	Nodes
Dendrites	Inputs
Synapse	Weights
Axon	Output

Table 3.1: Relationship between human neuron and artificial neuron

A neural network can be visualised as a very large number of artificial neurons, also known as nodes or units are made to stack in a form of hierarchy of layers. There are several different kinds of layers that can be found in an artificial neural network and can be better understood as the following diagram.

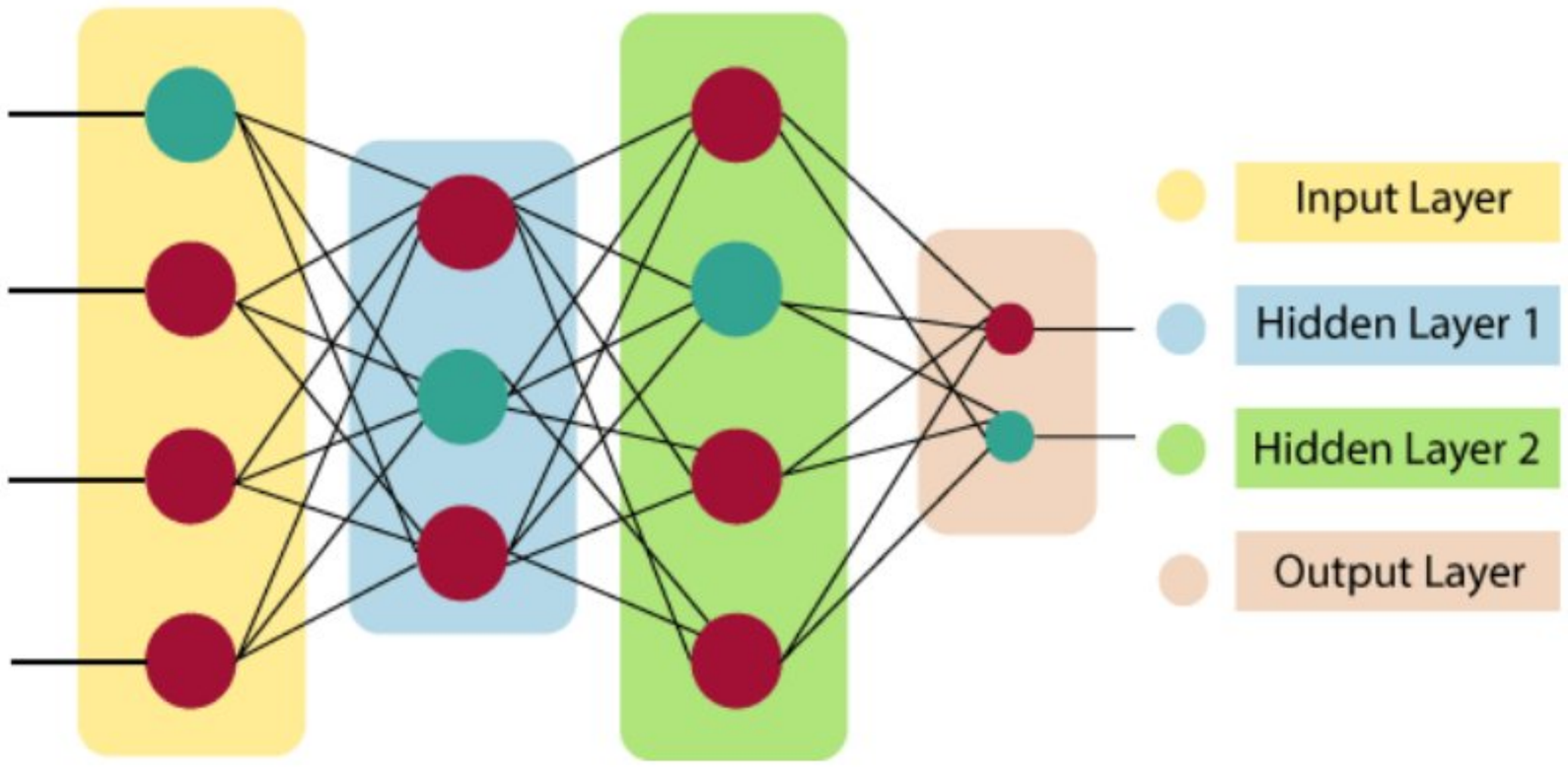

Fig 3.16 : ANN 3 layers architecture

Input layer : As the name of the layer itself implies it is the place where the user given inputs or the programmer provides the input data for further processing and the input can be of different formats.

Hidden layer : It is the place where all the computations are performed, in order to make all the necessary processing and feature extraction, to understand the hidden and buried patterns in the input data. This layer lies between the input layer and output layer of the ANN.

Output layer : The function of this layer is to receive the various outputs which are generated by the hidden layer by several processing and then to further communicate these output nodes to some activation function and then as a result. The artificial neural network computes some weighted total of all the inputs and then also incorporates a bias, whenever ann is fed with inputs. Further there is presence of a transfer function which is used to visualise all this processing and this computation for the purpose of producing the output, it then passes the weighted total as an input to an activation function. The purpose of the activation function is to decide upon the activation of a node, and the output layer only receives those nodes which are activated by the activation function. There are various different kinds of activation functions and they are used according to the type of classification or regression problem they are used in.

3.5.4 Forward and Backward Propagation

Forward propagation - as the name implies, forward propagation feeds the network with input data moving forward.All the inputs moves in forward direction, from input layer the hidden layer receives the input nodes and then process them in accordance with the activation function and further forwards these nodes or transfers these nodes to the upcoming immediate next layer..

In ANN emphasis is laid on supplying all the input nodes only in forward direction because it is the only way the output will get produced in such an architecture. If the data starts flowing in some other direction then this might create a cycle or a loop and will hinder the formation of output nodes. Therefore to prevent this the data must flow only in forward direction during the process of output generation. This is what is implied by

forward propagation which is also referred to as feed forward propagation, as it helps in spread of information.

Processing takes place at each neuron in a hidden or output layer in two steps:

Preactivation: It is the linear transformation of weights with respect to the inputs available. It is a weighted sum of inputs. In preactivation the neurons decide upon the activation of input nodes in intermediate layers of ANN architecture. And the function is the same, to decide whether or not a node should be activated and transmitted to layers ahead which is based on the aggregated total and the type of activation function used.

Activation: The activation function receives the estimated weighted sum of the inputs. It is basically a mathematical function which gives the network more nonlinearity. The sigmoid, hyperbolic tangent(tanh), ReLU, and Softmax activation functions are the four most widely used and well-liked activation functions.

One of the major foundations and back support of neural networks is **backpropagation.** In this method all the weights of the neural network are adjusted or tweaked again on the basis of the error rate which is record at the end of preceding epoch or iteration during training, and by properly tuning and tweaking of the weights, the error rate recorded can be lowered to much levels and overall model's reliability is enhanced. This also aids in ultimate broadening of the model's applicability.

In neural networks, backpropagation is the abbreviation for "backward propagation of errors." It is the most common technique used for developing and training the whole neural network. This method helps in calculation of the gradient of loss function with respect to each weight in the neural network.

The following are the steps for backpropagation:

Step 1: Firstly all the input nodes must enter through the preconnected path.

Step 2: With the help of the weights all the input nodes are modelled. Generally the weights initially are selected at random.

Step 3: Determine each neuron's output from the input layer through the hidden layer and out to the output layer.

Step 4: Determine the output error

Backpropagation Error = Actual Output - Desired Output

Step 5: Return from the hidden layer to the output layer and change the weights to lower the error.

Step 6: Continue the procedure until the intended result is obtained.

3.5.5 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a subset of deep learning neural networks that are frequently employed in pattern recognition, computer vision, and image processing.Since CNN is based on image processing, one must understand what exactly an image is. A digital image is a binary representation of any visual data which is made up of an arrangement most similar to a grid. Image can be thought of as a grid of pixels, and each and every value of the pixel indicates some information like the colour it is and how bright it is. Convolutional, pooling, and fully linked layers are some of the layers that make up CNNs. In mathematics, a procedure called convolution on two functions creates a third function that expresses how the shapes of the first two are changed. CNN operates on similar concepts. The role of CNN or ConvNet is to transform the images into a format that is simpler to interpret without sacrificing details that are essential for making accurate predictions.

When humans first see an image, their brains process a huge amount of information. They as a whole system encompass practically the whole vision field, like the human brain, each neuron has a specific receptor and is linked to other nodes in order to transmit the information gathered. Analogue to how each neuron in the biological vision system reacts to stimuli only in the restricted region of the visual field known as the receptive field, each neuron in a CNN analyses data only in its receptive field. The layers first pick up on lines, curves, and other simpler patterns before moving on to more complex patterns like faces and objects. By using a CNN, one can enable human like sight for computers.

A CNN typically has three layers: starting from convolutional layer then next subsequent laye is a pooling layer and there can be repetitions of such layers and finally a fully connected layer

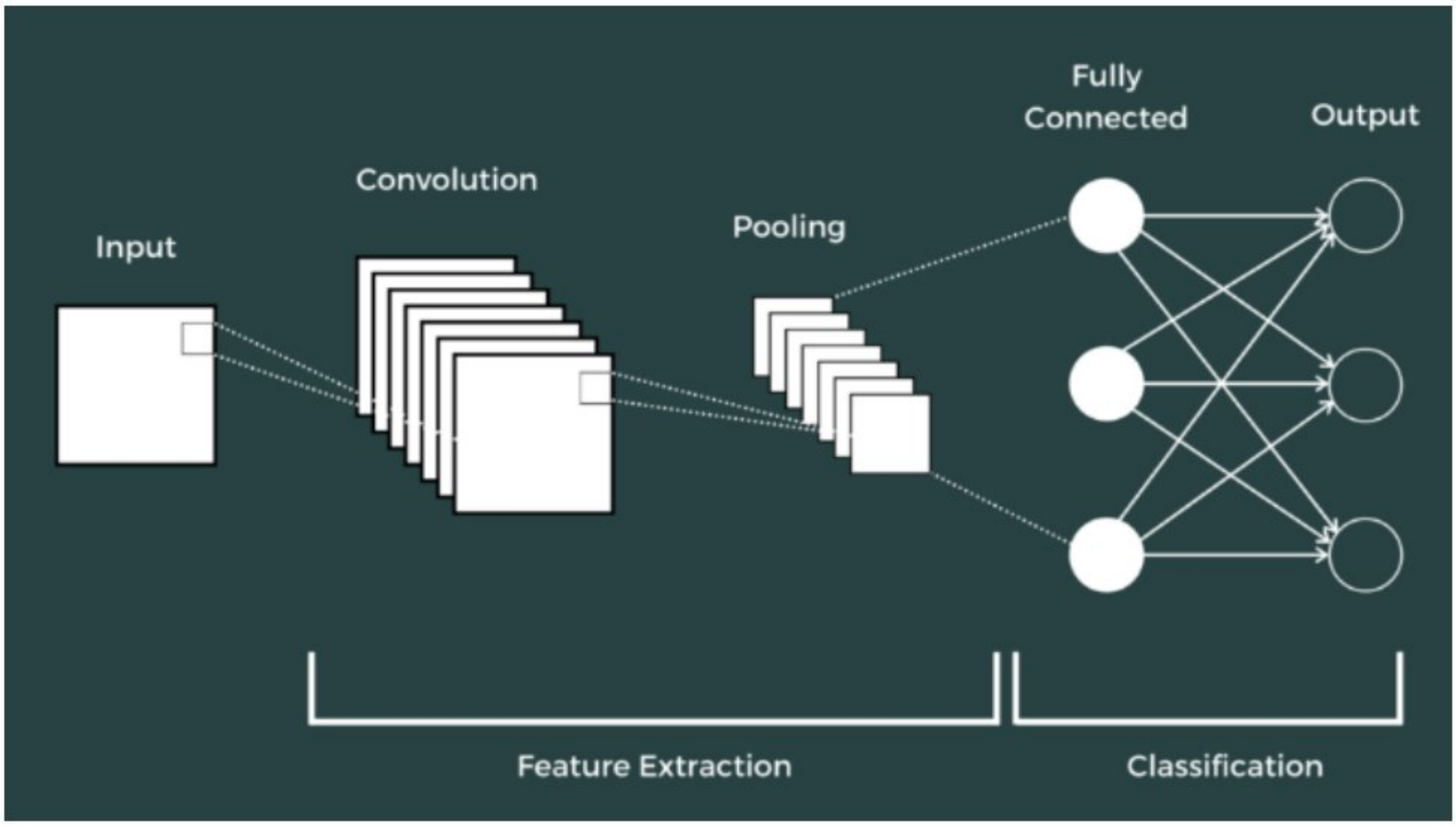

Fig 3.17: CNN Architecture

3.5.6 ReLu in Hidden layers

A frequent form of activation function in deep neural networks, such as Convolutional Neural Networks (CNNs), is the ReLU (Rectified Linear Unit) layer. The element-wise rectification process, which is applied by the ReLU function, simply outputs the input value if it is positive and 0 otherwise. In comparison to sigmoid, ReLU boasts of training sophisticated models and neural networks with constant values. Relu shows greater effectiveness: ReLU has an advantage over other sigmoid functions in that it runs significantly faster and with a lot lower run time max(0,f). This is due to the fact that it does not suffer from the vanishing gradients problem.

3.5.7 Pooling

The pooling layer acts as a stand-in for the output of the network at particular locations by computing a combined statistic from the nearby outputs. This helps to reduce the spatial size of the representation, which minimises the volume of processing and weights required.

The pooling procedure is applied to each layer of the representation independently.One of the pooling operations is in addition to the average of the rectangular neighbouring values in matrix and the L2 norm of the rectangular neighbourhood, is based on the distance that exists from the central pixel. The most used approach is max pooling, which reports the largest output from the neighbourhood.

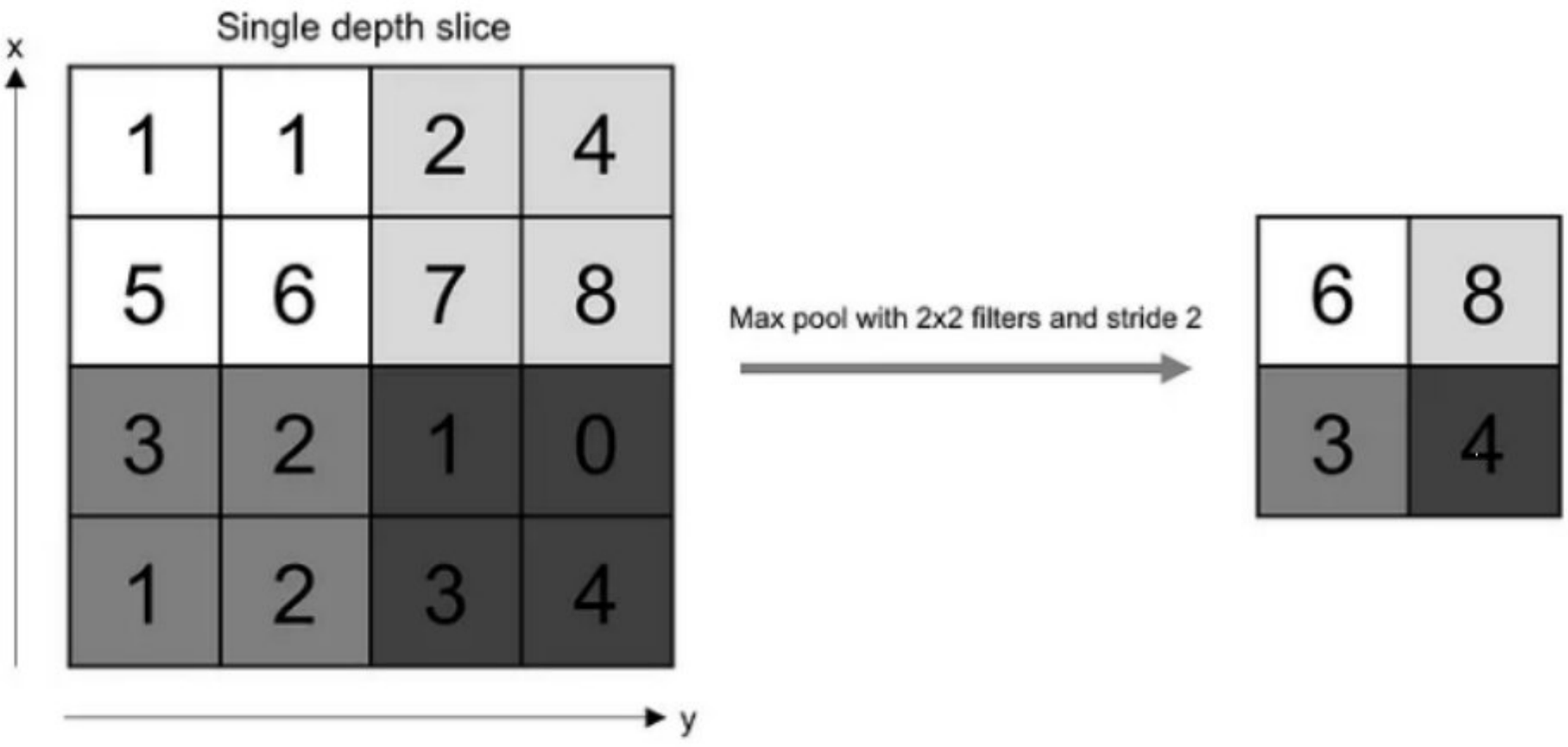

Fig 3.18: Pooling Operation Example

3.5.8 Flattening

A Convolutional Neural Network (CNN) technique known as flattening converts the output of the convolutional and pooling layers into a one-dimensional vector that is sent to a fully connected layer for additional processing. The output of a convolutional or

pooling layer is typically a three-dimensional tensor with dimensions (height, width, depth), which is further turned into a one-dimensional vector in order to send it to a fully connected layer. By simply stacking the values of each channel on top of one another, the flattening operation converts the tensor into a long vector.

3.5.9 Full Connection

In typical Fully Connected Convolutional Neural Networks (FCNN), all neurons in this layer are fully connected to all neurons in the layer before it and the layer after it. This connection enables the computation of the units of this layer using the standard matrix multiplication and bias effect method. The representation that exists across the input and the output is mapped using the FC layer.

3.5.10 Activation Function For Final Layer

A Convolutional Neural Network (CNN)'s final layer's activation function is determined by the type of problem being solved. The final layer of a CNN commonly employs a sigmoid activation function for binary classification issues. The network's output is translated by the sigmoid function into a probability value between 0 and 1, which represents the likelihood that the input belongs to the positive class. The final layer of a CNN commonly uses a softmax activation function for multi-class classification issues. The network's output is normalised by the softmax function so that the total of the probabilities for each class equals 1. As a result, the network may generate a probability distribution for each potential class and select the class with the highest probability as the output. The final layer of a CNN commonly employs a linear activation function for regression problems, which yields a continuous output value. Depending on the particular requirements of the problem, alternative activation functions, such as tanh or ReLU, may occasionally also be utilised in the final layer of a CNN.

3.5.10 Sparse Categorical Cross-Entropy

When working on a deep learning or machine learning problem, the model is optimised during training utilising loss/cost functions. Almost always, the objective is to decrease the loss function. The lower the loss, the better the model performs. Cross-entropy loss is one of the most significant cost functions. It aids in the optimisation of classification models.

Logits are transformed into probabilities by Softmax. The Cross-Entropy is used to calculate the deviation from the truth values by taking the output probabilities (P).

Making the model output as close as feasible to the desired output (truth values) is the goal. In order to reduce the Cross-Entropy loss, the model weights are iteratively modified as necessary during model training. Model training is the process of changing the weights, and as the model continues to train and the loss is reduced, we say the model is learning.

Cross-Entropy Loss Function is also known as logistic, logarithmic, or logarithmic loss. Loss of Cross-Entropy Logarithmic loss, logistic loss, and logarithmic gain are other names for function. The desired outcome for each class, which can be either 0 or 1, is compared to each class's anticipated probability, and a score/loss is computed which regulates the likelihood depending on the extent to which it diverges from its true expected value. Significant variations close to 1 receive a large score because of the penalty's logarithmic nature, whereas slight variations close to 0 receive a small score.

In order to change the model weights during training, cross-entropy loss is used. The goal is to reduce loss; hence, the better the model, the smaller the loss. A cross-entropy loss of zero indicates a flawless model.

The following is the equation of cross entropy loss function

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \quad \text{for n classes,}$$

where t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

Fig: 3.19 Mathematical equation for cross entropy loss function

This is also the equation for the sparse categorical entropy loss function which is used in the proposed model. In case of sparse categorical entropy truth labels are integer encoded.

Chapter- 4: PERFORMANCE ANALYSIS

In machine learning, it is crucial to be able to accurately assess the model being created in order to ensure that the predictions (such as those for disease prediction or cost estimation in the future) adequately describe the intended phenomenon. But with so many options available (Accuracy, Precision, Recall, etc.), it can be difficult for data scientists to decide which performance statistic to utilise. To be able to measure the performance of the model objectively and in the appropriate context, the suitable metric must be chosen for a given model. This chapter discusses the performance parameters that have been chosen for the purpose of evaluation of the proposed CNN model.

4.1 Selection of best performance parameters :

One can assess the effectiveness of ML, DL algorithms, classification algorithms, and regression algorithms using a variety of metrics. One must exercise caution while selecting the measures to measure the effectiveness of ML because

- The statistic that one selects will determine exactly how the effectiveness of ML algorithms is assessed and compared.

- The metric that one selects will have a significant influence on how one weighs the relative value of different variables in the outcome.

Classification is a process in machine learning and data science in which a model is built for the purpose of differentiating among different classes of input data, and the input data can be binary as well muli class, depending on its nature, during classification process the model comprehends from various different feature and identify features that make certain value map to its respective class. In machine learning, deep learning and AI there exists a lot of algorithm which are used to classify a dataset into its apt category, and this is done

with the help of a pre labelled or pre classified set of data which a programmer makes before training, and is called a training set. All ML algorithms then train upon this training data to learn hidden features and patterns in order to classify input test sets into their respective classes.

This study makes use of multilabel classification using deep learning based convolutional neural networks.

Class-balance and anticipated results are two factors that should be taken into account when choosing the optimal metrics to assess a given classifier's performance on a certain dataset. One particular performance metric may only assess a classifier from one angle while frequently underestimating others. As a result, it is impossible to choose a single statistic to assess a classifier's effectiveness in general. For multiclass classification there are various metrics which can be used: Classification report, confusion matrix, Class Prediction Error, Cohen's kappa, Cross entropy, Area Under Curve, Log Loss, etc.

For this Study confusion matrix and classification report are picked for the purpose of evaluation.

4.1.1 Confusion Matrix

It is the simplest approach for determining how well a proposed model is performing on a classification, and its output can include two or more different types of classes. A confusion matrix is just a square matrix or a table having two main dimensions, Actual values and Predicted values, as well as True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) for each dimension. The actual values are the values with real labels and the predicted values are the values that our model predicts. For Binary Classification the following diagram depicts a simple confusion matrix:

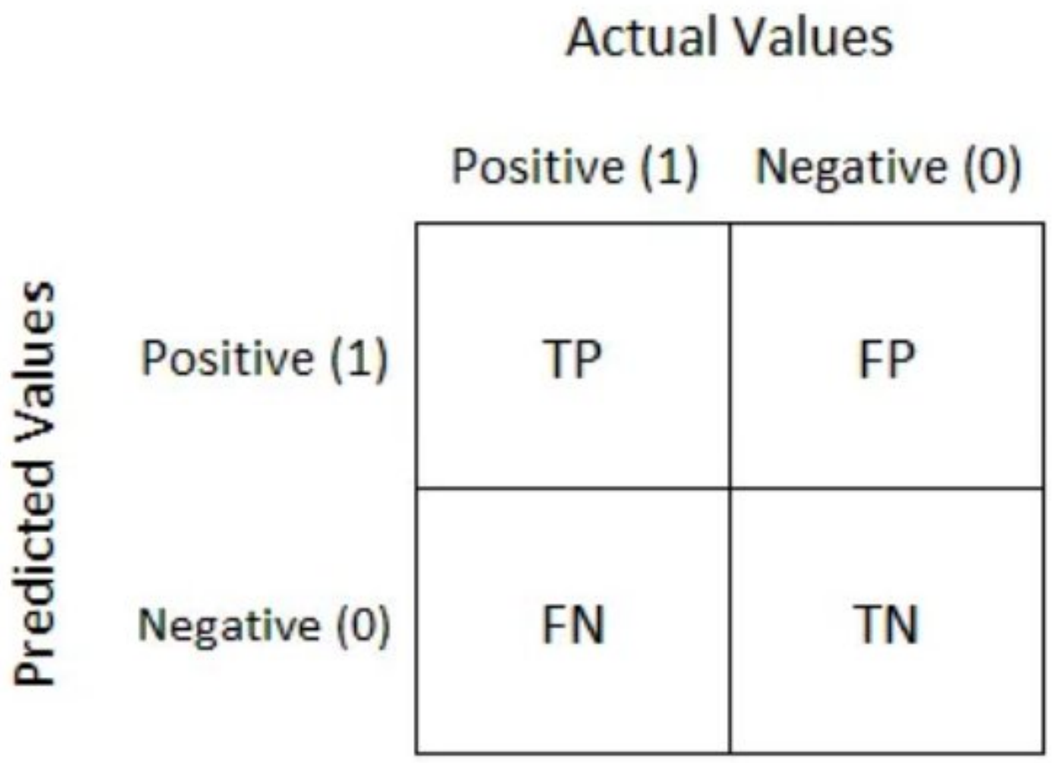

Fig 4.1: A simple binary class classification Confusion matrix

Recall, specificity, precision, accuracy, and—most importantly—AUC-ROC curves may all be measured with great success with the help of this tool.

The following are the descriptions for TP, FP, FN, TN:

- **True Positive:** It is the case, when the actual and predicted values are both same, or the predicted values match their true labels.
- **False Positive:** It is the case, when the predicted values come out to be true, but in actual they were labelled as false.
- **False Negatives:** It is the case, when the predicted values come out to be negative but in actual they were labelled as true.
- **True Negative:** It is the case, when the predicted values are negative and their corresponding actual values are also negative.

The following figure depicts the confusion matrix for the predicted and actual values.

For the purpose of plotting the matrix, confusion_matrix method of sklearn library is used, and seaborn and matplot library further helps in better visualisation of the results of the model.

```
In [43]: class_names = [np.argmax(element) for element in y_pred]
         plt.figure(figsize=(5,3))
         fx=sn.heatmap(confusion_matrix(y_test,y_pred_classes), annot=True, fmt=".2f",cmap="GnBu")
         fx.set_title('Confusion Matrix \n');
         fx.set_xlabel('\n Predicted Values\n')
         fx.set_ylabel('Actual Values\n');
         fx.xaxis.set_ticklabels(class_names)
         fx.yaxis.set_ticklabels(class_names)
         plt.show()
```

Fig 4.2: Code for plotting confusion matrix

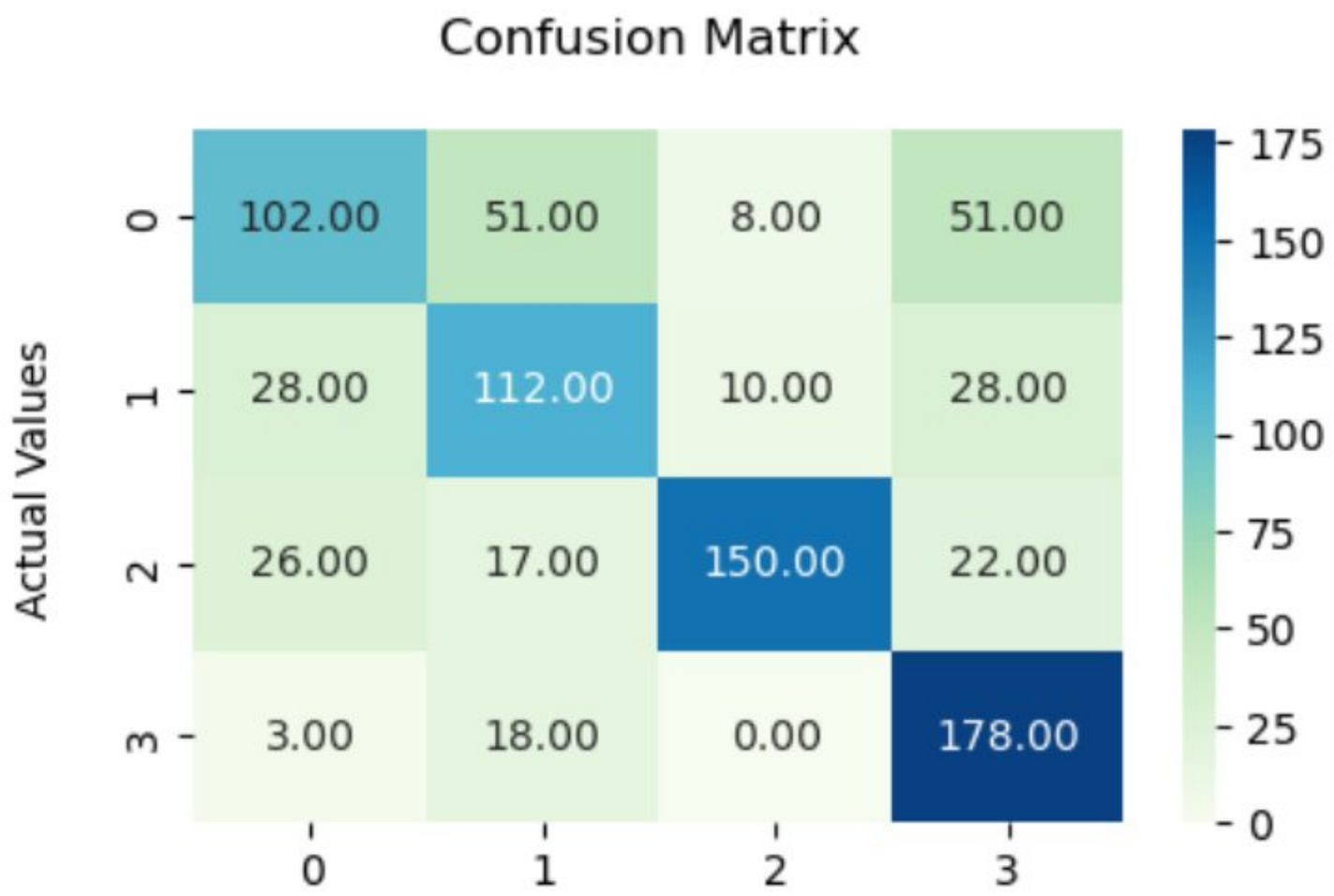

Fig 4.3: Confusion matrix

Here Classes of retinal images - normal, glaucoma, diabetic retinopathy and cataract are encoded as - 0, 1, 2, 3 respectively.

The Dark blue coloured diagonal from top left to bottom right corner of the matrix, depicts the true positive values. Therefore 102 images of 'class 0' which is 'normal' class of retinal images are predicted correctly. 112 images of 'class 1' which is 'glaucoma' class are predicted correctly. '150' images of 'class 2' which is 'diabetic retinopathy' are predicted correctly. And 178 images of 'class 3' which is 'cataract' are predicted correctly.

For class ‘0’ or **Normal retinal** images

True positives (TP): 102

True negatives (TN): 112 + 10 + 28+ 17+ 150 + 22 + 18 + 0 +178 = 535

False Positives (FP): 51 + 51 + 178 = 196

False Negatives (FN): 18 + 0 + 178 = 196

For class ‘1’ or retinal images with **Glaucoma disease**

True positives (TP): 112

True negatives (TN): 150 + 22 + 178 + 0 + 28 + 3 + 8 + 51 +102 = 542

False Positives (FP): 18 + 17 + 51 = 86

False Negatives (FN): 28 + 10 + 28 = 66

For class ‘2’ or retinal images with **Diabetic Retinopathy disease**

True positives (TP): 150

True negatives (TN): 102 + 51 + 51 + 28 + 112 + 28 + 3 +18 + 178 =571

False Positives (FP): 0 + 10 + 8 = 18

False Negatives (FN): 26 + 17 + 22 = 65

For class ‘3’ or retinal images with **Cataract disease**

True positives (TP): 178

True negatives (TN): 102 + 51 + 8 + 28 + 112 + 10 + 26 + 17 + 150 = 504

False Positives (FP): 51 + 28 + 22 = 101

False Negatives (FN): 3 + 18 + 0 = 21

4.1.2 Classification Accuracy

Its use as a performance metric for classification algorithms is very widespread. It can be described as the proportion of all true or accurate forecasts made to the number of forecast predictions made. With the help of the following mentioned formula, one can quickly calculate it using the confusion matrix, once the confusion matrix is generated.

$$\textbf{Accuracy} = \frac{\textbf{TP+TN}}{\textbf{TP+FP+FN+TN}}$$

The evaluation is the major process during any model-development process because it determines whether the model is the best match for the given problem and pertinent data. The Keras model offers a method called evaluate() that can be used with the object that stores the model, and as the name suggests is used to perform evaluation of the model. It gives testing loss and test accuracy of the model.

For the purpose of getting the accuracy of the proposed model, following code was executed and the final accuracy was around 67.4%

```
In [20]: score = model.evaluate(X_test, y_test)

         26/26 [==============================] - 7s 258ms/step - loss: 0.7893 - accuracy: 0.6741
```

```
In [21]: print('Test accuracy:', score[1]*100)
Test accuracy: 67.41293668746948
```

Fig 4.4: Model final accuracy

4.1.3 Classification Report

In machine learning and data science, a classification report is a performance metric which is used for the purpose of evaluation of a model. It is used to denote a trained model's classification accuracy and other parameters like - recall, F1 Score, and support. It gives a clearer picture of a trained model's overall performance.

The following is the detail description of each of the metric of classification report :

Precision : The number of positive forecasts were actually correct can be determined by precision. To do this, it divides the total number of positive predictions—correct or incorrect—by the number of samples that were correctly predicted as positive (TP).

The following represents the formula for calculating precision :

$$\textbf{Precision} = \frac{\textbf{TP}}{\textbf{TP+FP}}$$

Recall : Parallel to Precision, Recall is the measure of true positives that were accurately predicted by the model on a test set. This is accomplished by dividing the total number of samples that were properly forecasted as positive or wrongly predicted as negative (TP, FN) by the total number of samples that were correctly predicted as positive (TP). Recall is also referred to as **sensitivity, true positive rate, or hit rate**.

The following is the formula for recall :

$$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP+FN}}$$

Specificity : Similar to recall (also known as sensitivity), specificity measures the percentage of actual negatives that were accurately identified. It calculates this by dividing the number of negative samples that were properly projected to be negative or wrongly forecasted to be positive (TN, FP) by the total number of negative samples.

The following is the formula used to calculate specificity:

$$\textbf{Specificity} = \frac{\textbf{TN}}{\textbf{TN+FP}}$$

To evaluate the ML and DL models created, precision and recall are frequently used in the data science field. Medical professionals frequently consider specificity and sensitivity while assessing medical testing. These ideas are quite dissimilar yet similar at the same time. This disparity may lead to numerous misconceptions between the medical community and the data science community when the two worlds collide, as when a medical test is a machine learning model.

Precision: This indicates the proportion of instances that positive predictions came true. Recall, on the other hand, indicates how many times the model accurately anticipated based on actual positive evidence. False negatives should be avoided at all costs in any medical diagnosis, for example cancer disease detection, since they can have fatal repercussions. Here, recall serves as a more useful metric than precision. False negatives are less significant if a person has to optimise suggestions on YouTube because just a tiny percentage of recommendations is displayed in any case.

Since the model proposed in the study is based on medical images dataset, and aims to classify eye diseases among four different categories, emphasis is laid on why precision is important and as a evaluation parameter for this study, a detailed research has been carried out to find the importance and use of the same for model evaluation, and why doctors must consider precision before recall and other parameters before evaluating the trained model.

Precision and recall frequently have the opposite relationships. A model is likely to have higher precision but lower recall if it predicts fewer positive cases. However, because false positives can have more serious repercussions than false negatives, precision is typically given precedence over recall in medical classification tasks. As a result, accuracy aids in striking a compromise between the model's overall performance and accurate positive forecasts.

In case of imbalanced Datasets that is the case of medical datasets which often suffer from class imbalance, one class may have a disproportionately low number of instances compared to another. Precision is crucial in these circumstances. For medical diagnostic and treatment decisions, a high accuracy score guarantees that the model is correctly recognising the positive class even in the presence of skewed data.

Misclassification can have negative effects in medical situations. Precision reduces the likelihood of making an incorrect diagnosis of a patient as positive when they are actually negative, which helps to minimise false positives. By minimising unnecessary procedures or interventions, we can improve patient safety by maximising precision.

Cost effectiveness is aided by precision as Misclassifications may result in time- and money-consuming and needless medical procedures, examinations, or treatments. Precision guarantees highly accurate positive forecasts, cutting down on wasteful expenses related to false positives.

Since precision guarantees patient safety, Misclassification can have negative effects in medical situations. Precision reduces the likelihood of making an incorrect diagnosis of a patient as positive when they are actually negative, which helps to minimise false positives. By minimising unnecessary procedures or interventions, we can improve patient safety by maximising precision.

High precision in a medical categorization model suggests a high degree of dependability and credibility. The predictions made by the model must be trusted by patients and healthcare providers. By putting precision first, we can create models that deliver reliable, precise outcomes, strengthening the relationship between the model and its end users.

Support : Support in classification report defines the presence of each particular instance of the different classes of dataset in the test set or the set which consists of true responses.

F1 Score : It generates the harmonic mean of precision and recall. Mathematically F1 score is weighted average of precision and recall. It is more useful than accuracy during unevenly distributed classes of dataset.

$$F1 = 2 * \frac{(precision*recall)}{(precision + recall)}$$

F1 score 1 represents its highest value and reveals that models precision and recall are perfect. Whereas if F1 score is 0, this represents its lowest value and indicates that either recall is 0 or precision is 0.

F1 scores generally have almost always an equal amount of contribution of both precision and recall during model evaluation.

The classification report of the model can be calculated using sklearn's metrics module. The following figure shows the classification report for the proposed model and the model trained:

```
In [27]: from sklearn.metrics import classification_report
         y_pred=model.predict(X_test)
         y_pred_classes= [np.argmax(element) for element in y_pred]
         print("Classification Report :\n", classification_report(y_test, y_pred_classes))

         26/26 [==============================] - 7s 287ms/step
         Classification Report :
                       precision    recall  f1-score   support

                    0       0.64      0.48      0.55       212
                    1       0.57      0.63      0.60       178
                    2       0.89      0.70      0.78       215
                    3       0.64      0.89      0.74       199

             accuracy                           0.67       804
            macro avg       0.68      0.68      0.67       804
         weighted avg       0.69      0.67      0.67       804
```

Fig 4.5: Classification report for the model