

INSPECTION OF EYE DISEASES USING CORNEAL TOPOGRAPHY

**A Project Report submitted in partial fulfillment of the requirements for the
award of the degree of**

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Shifa Mehreen	121910313005
Chakrapani Anisetti	121910313015
Ch. Rohith Reddy	121910313033
R. Sai Vinay	121910313041

Under the esteemed guidance

of

Mr. V. Venkata Vidya Sagar

Asst. Professor



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

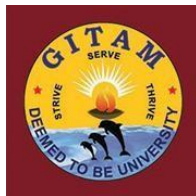
(Deemed to be University)

VISAKHAPATNAM

OCTOBER 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM





(Deemed to be University)



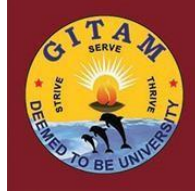
DECLARATION

I/We, hereby declare that the project report entitled “**INSPECTION OF EYE DISEASES USING CORNEAL TOPOGRAPHY**” is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 28-10-2022

Registration No(s).	Name(s)	Signature(s)
121910313005	Shifa Mehreen	
121910313015	Chakrapani Aniseti	
121910313033	Chitikela Rohith Reddy	
121910313041	Rajapantula Sai Vinay	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM
(Deemed to be University)



CERTIFICATE

This is to certify that the project report entitled “**INSPECTION OF EYE DISEASES USING CORNEAL TOPOGRAPHY**” is a bonafide record of work carried out by **Shifa Mehreen (121910313005), Chakrapani Aniseti (121910313015), Chitikela Rohith Reddy (121910313033), Rajapanthula Sai Vinay (121910313041)** students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide

Head of the Department

Mr. V Venkata Vidya Sagar

Dr. R. Sireesha

Asst. Professor

Professor

ACKNOWLEDGEMENT

We would like to thank our project guide **Mr. V. Venkata Vidya Sagar**, Assistant Professor, Department of CSE for his stimulating guidance and profuse assistance. We shall always cherish our association for his guidance, encouragement and valuable suggestions throughout the progress of this work. We consider it a great privilege to work under his guidance and constant support.

We also express our thanks to the project reviewers **Dr. G. Venkateswara Rao**, Professor and **Dr. S.V.G. Reddy**, Associate Professor, Department of CSE, GITAM (Deemed to be University) for their valuable suggestions and guidance for doing our project. We consider it a privilege to express our deepest gratitude to **Dr. R. SIREESHA**, Head of the Department, Computer Science and Engineering for his valuable suggestions and constant motivation that greatly helped us to successfully complete this project.

Our sincere thanks to **CH. VIJAY SHEKAR**, Dean, GITAM School of Technology, GITAM (Deemed to be University) for inspiring us to learn new technologies and tools. Finally, we deem it a great pleasure to thank one and all that helped us directly and indirectly throughout this project.

SHIFA MEHREEN

121910313005

CHAKRAPANI ANISETTI

121910313015

CH. ROHITH REDDY

121910313033

R. SAI VINAY

121910313041

TABLE OF CONTENTS

1.	Abstract	1
2.	Introduction	2
3.	Literature Review	3
4.	Problem Identification & Objectives	3
5.	System Methodology	5
6.	Overview of Technologies	8
7.	Implementation	10
8.1	Coding	11
8.2	Testing	14
9.	Results & Discussions	33
10.	Conclusion & Future Scope	34
11.	References	34

ABSTRACT

Ocular diseases nowadays are highly prevalent among kids and elders, and they drastically impact the derivatives of the human body and ecosystem. Eye disorders are one significant result foreseen.

WHO found approximately 62 million people suffering from visual imparity in recent surveys in India in 2010, and most people found eye diseases like conjunctivitis, cataract, glaucoma, and macular degeneration. Therefore the need to mitigate the trauma is high in demand. The objective of the present study is to determine the Corneal Topography by better methods.

The early study of the Keratoconus stages helps predict the disorder and correction of the vision. The data sets' predictive performances help the ophthalmologist refine health checks and treat the patients more accurately. Analyzing fundus images is tedious, time-consuming, and vastly depends on the doctor's experience. Still, with the advent of automated computer machines and ML algorithms, the prediction of these diseases and the diagnosis are made more accessible.

Technology is a plethora in today's medical research as medical outcomes have become accurate and usage improved constantly. Our attempt with SMART KC and ML algorithms is a pipeline to the present-day study.

In our work, we presented a device that will help ease the task of doctors: the SmartKC. It is a smartphone-based device that outputs corneal topography. It is cost-efficient, portable, and accessible. It will allow the prediction of a Keratoconus via corneal topography. Surgeons best use corneal topography in cataract surgery.

The ML prediction models shown in our project allow us to predict diseases like cataracts, glaucoma, and AMD using fundus imaging. It identifies the cataract in the patient's eye, and the Smart KC acquires the corneal topography before the surgery.

INTRODUCTION

Eyes are dissimilar in shape, color, and size. They are the prettiest and most important sensory organs of the human body. They enable us to view the beautiful world around us.

Working of an eye - when the light passes through the cornea and reaches our lens, the pupils grow bigger or smaller, depending on the amount of light that passes through. The cornea and lens perform a refraction to build an inverted image on the retina. The retina converts this information into electrical impulses, and optic nerves read the signals and send it to the brain. The brain uses its visual cortex to understand what both eyes see clearly.

It's like a camera, where each part does a different job bringing in a clear image of what we see. Even if one part of the eye lags the job, there will be an issue with the brain and interprets some leading visual disparity.

Diseases due to food deficiencies, genetics, or pollution are prevalent in third-world nations like India. Eye diseases are one of them. Some defects may even lead to blindness if not cured in time. These need a cure at an early age, usually with eyesight issues and the other defects occur with growing age, in an elderly stage. Some of India's most common eye diseases include cataracts, Glaucoma, and Macular Degeneration. Hence, recognizing eye problems and adequately addressing them as soon as possible helps to prevent them from impairing the eyes' normal functioning.

Cataract is an eye condition in which clouding occurs behind the iris. This develops over time. Some of the early stages include blurred vision, fading etc. Glaucoma is caused when immense pressure applies to the eye resulting in optic nerve damage, which causes permanent blindness. Macula is the center of the retina. When the position of the macula shifts, it causes blurring; this is called Dry Macular Degeneration. If blood vessels behind the retina grow abnormally, it is called Wet Macular Degeneration. Although it is rare, it is a severe condition. Keratoconus is a condition where the cornea becomes thinner and starts to bulge in a conical shape. It causes nearsightedness.

Our eyesight can impact by various factors, including different medical conditions like diabetes or high blood pressure. An eye disease that steals vision might strike at any time. They are usually hard to spot at first and go unnoticed most of the time. Hence, it is essential to get your eyes checked by an ophthalmologist. They may even identify any underlying diseases like heart disease from an eye exam.

Most eye diseases detect subtle changes in the appearance and structure of eye anatomy. These diseases mainly occur in the cornea and retinal part of the eye. These doctors have various instruments

to detect such conditions. Like retinal digital photography and corneal topography, etc. By far, digital imaging is the best way to detect eye diseases. Most eye diseases are incurable. The only possible way is opting for eye transplantation, but not everyone can afford this process. so early detection and prevention recommend.

LITERATURE REVIEW

This Documentation deals with the detection of Keratoconus disease by a proprietary software resident on "SmartKC". This deals with using a physical device attached to an external smartphone rear camera, by which we can get the image (data) of the cornea. The entire process bears a prototype model.

Before SmartKC there were many Software's designed, say "Keratron", among them. Keratron requires capturing four images of the eye to find the center, which goes manually. As we do it manually, there is a high risk of errors. In SmartKC, we detect the center using a cursor pointer, and it takes one image making the work faster than Keratron.

Keratoconus is detected using Sophisticated Devices, which are hard to carry and are not readily accessible to everyone. The most recent device ophthalmologists use Optical Coherence Tomography (OCT). Tomography is the process of building a 3D map, whereas topography plots a 2D map. So here we are, designing a prototype from which we can make a relatively more straightforward version.

Before this model, there were other 3D-printed devices called Placido, which contained concentric blue and black Placido rings. These are costly and require a proper arrangement of LED lights. We are trying to build a prototype model which replaces LED strips with a LED Bulb and GI Wire to replicate the ringed pattern.

PROBLEM IDENTIFICATION AND OBJECTIVES

THE PROBLEM:

Most eye diseases go unnoticed initially and can cause a huge risk to the eye if not treated early. Keratoconus is one such disease. Keratoconus is thinning and bulging of the cornea in conical shape. This Disease is caused due to vigorous rubbing of eyes or heredity. Since identification of symptoms for eye diseases in the early stages is very important to treat the disease properly; we have to have instruments that are of the best quality, very accurate and easy to use. So far there is no means to cure Keratoconus. Early detection and prevention are the best measures. Other diseases, like Cataract, Glaucoma, etc which are also prevalent in India. These diseases can be found through retinal imaging.

To enrich the technology in the medical sector is to improve software with better innovations to old devices using robots, automation and high-end technical devices in order to reach faster, accurate and better diagnoses.

THE SOLUTION:

Keratoconus is usually detected using high-end devices such as Optical coherence tomography. But it is not available for all, so we are trying to design a model which can be used through mobile devices, through which they can do their self-diagnosis or can be used in the hospitals and diagnose the patient before any serious issues.

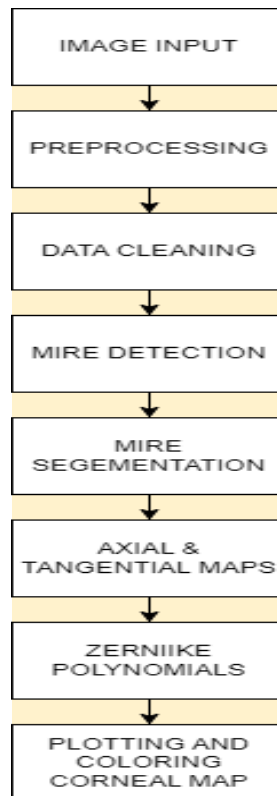
ML techniques explore a perspective smart prototype facilitating a portable, accurate, multiple operations equipped in a single device and enhances improved diagnosis. We capture the image of the eye, using Placido and the image acts as input to Software and give a corneal topography, which helps a refined diagnosis.

We have built a CNN model using the dataset - ODIR -5k, retinal images, for the other diseases, to compare a retina image to a particular disease.

OBJECTIVES:

- To obtain Corneal Topography for cornea related disease like Keratoconus
- To build a sample prototype model for Smart KC
- Build Software → to capture image then convert the input image into a graph (data visualization) so as to find if KC or not
- Consider other diseases - Cataract/Glaucoma/Macular Hypertension (considered only these since, more prevalent in India) and To build an ML model to predict whether the image belongs to that disease or not.

SYSTEM METHODOLOGY:



FOR SMART KC:

Prototype: SmartKC:

Paper model:



Our Eye viewed using the model:



Actual working 3D- Model Image:



Software: SmartKC:

This model will be used to take in the input image with the device, record it and go through the software designed and then to output a corneal topography that will be accurate and easy to study, finally checking if that person has KC or Not. A specific smartphone's camera is used to

STEPS:

- Data Collection → taking image as input

Image Analysis Pipeline:

- Pre-Processing Image → 1. Image cropping, focus on the eye and not the eyelids and other parts of eye, 2. Mire Segmentation is important to get correct selection of the cornea, to measure radii 3. Center detection, get center of the eye image
- Mire - Placido mapping → Placido ring location and working distance is calculated
- Corneal Topography estimation → 1. Arc-Step method, allows the measurement of both curvature and height at each point on the cornea with a resolution and accuracy of less than one micron. 2. Zernike Smoothing, functions used for pupil planes in the classical optical imaging
- Corneal Topography and other preprocessing Images → output folder = date stamp

PROCESS:

- Patients focus on the center of the circles and the operator captures the proper images, and the one with clear mires is then selected. → Oculyzer test (used to

monitor cornea and generate reliable results in no time.)

- (capture eye images with mires being sharp and in-focus limbus to limbus coverage, along with the patient name, id, and gender (so no confusion with the patients))
- Take image input and run the code with it → running script lines in python
- Image is shown
- Select center of the image correctly
- Later, the code starts to execute and perform various functions on the image and once the execution of all the functions is done, it shows “Test Complete!”
- Now, when we look at our root directory of the project, we will find a folder with the particular date stamp having the final output - corneal topography and mid-process images like the preprocessing and mire detection cleaning images too.

PROCESS FLOW CHART

Patient eye → Operator capture image → input image = sharp mires → run code → select center → code executes on the image → test complete! → folder = date stamp → final output = corneal topography

If data collected we could run a CNN model on the tangential and axial images and predict if the patients have kc or not.

Due to constraints with the dataset, we tried to develop a similar CNN model for the retinal images, for other such eye diseases

FOR OTHER DISEASES:

Dataset Used: ODIR -5K → has fundus images of 5K patients collected from different medical institutes in China.

Fundus imaging is defined as the process whereby reflected light is used to form a two-dimensional representation of the three-dimensional retina, the semi-transparent, layered tissue lining the interior of the eye projected onto an imaging plane.

Diseases like Glaucoma, Cataracts, that hamper vision can be found using this retinal imaging

ML Model:

We Use CNN → since, image data

ML MODEL FLOW CHART:

Dataset (ODIR-5K) → Data Preprocessing → Data Cleaning → Data Segmentation → Model VGG19 → Output - Predictions and Classification into Class A or Class B

SMART KC SOFTWARE:

Software:

It is important that the software being used for medical purposes promotes safe innovation, is accurate and protects patient safety.

For the software behind this minimal and sophisticated device, the Smart KC, we need to run many files in the background for it to give appropriate results.

Keratron is the device that is being used for detecting KC.

Using Smart KC, we hope to acquire accurate results to that of the Keratron diagnosis by an ophthalmologist at the hospital.

We could use a mobile app for data collection. This app would allow:

1. the operators to capture good quality, aligned mires and blur-free images.
2. record patient details like patient id, name, gender, and age.

Capturing good-quality images is important for disease detection; we have to ensure that the sources of error are extremely less. Since we are using a portable low-cost setting with minimal external hardware, the job gets even more challenging.

OVERVIEW OF TECHNOLOGIES

SOFTWARE:

Python - Python is a high level language with huge libraries, capacity to handle enormous quantities of data, and beginner friendliness. It is mostly utilized in the fields of machine learning and data science. It also offers all the functionality that is included in other general purpose languages. It is object-oriented, enables functional programming, has dynamic typing, and Python's code is very legible.

Numpy - NumPy is a Python library. It is based on numerical computation. Because the library is open source, anybody may use it. Since it offers us superior data structures than conventional python, it is widely employed in data science. Python lists are less efficient than arrays, which are used in NumPy. In addition to having various mathematical

functions that may be used directly on them, they enable vectorization. This offers a great deal of usefulness and adaptability.

opencv-contrib-python - OpenCV contrib is a specialized module present in the Python programming language, which is exclusively needed for the system to run SURF feature descriptions alongside the OpenCV module present in the open-source library.

scikit-image - scikit-image is the open-source image processing package that Python programmers may use. It is made to work with Python's NumPy and SciPy libraries. It comprises algorithms for segmentation, color space manipulation, analysis, feature recognition, and more.

scikit-learn - A Python open source package called Sklearn or Scikit-Learn is used to implement all of the main machine learning algorithms currently available. Instead of starting from scratch, it gives programmers access to prebuilt algorithms, which saves a ton of time.

scipy - SciPy stands for Scientific Python. It provides more utility functions for optimization, statistics and signal processing. SciPy is a scientific computation library that uses NumPy underneath. Like NumPy, SciPy is open source so we can use it freely.

zernike polynomials - In ophthalmology, Zernike polynomials are used to describe optical imperfections of the cornea or lens from an ideal spherical shape, which result in refraction errors. They are also commonly used in adaptive optics, where they can be used to characterize atmospheric distortion.

pandas - Pandas is another free and open-source Python library. Data analysis is done using it. Since much of the data accessible today is structured, Pandas includes a particular data structure called Data Frame that is very effective at managing tabular data. Pandas also offers a lot of capability in processing and altering that data.

matplotlib - Matplotlib is an open source, low level graph plotting library in python that serves as a visualization utility.

seaborn - Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.

tqdm - tqdm is a library in Python which is used for creating Progress Meters or Progress Bars.

Pytorch (torch, torchvision and other dependencies for Pytorch) - PyTorch is an open sourcedeep learning framework, which provides a Python package for high-level features like tensor computation (like NumPy) with strong GPU acceleration.

random - Random is a python module that is used to generate random values

tensorflow - It is a free, open source python library which provides libraries that are used inworking with neural networks

keras - it is an open source python library which provides interface to work with neural networks

IMPLEMENTATION

Hardware:

- Chart
- Transparent paper
- Mesh
- Transparent tape
- Magnifying glass
- Wire
- LED Bulb
- Lithium-ion battery

CODING

main.py -

It is the most important file because it contains the corneal_topography_generator class. This classcontains following parameters -

model_file which contains the dimensions of the hardware device, working_distance which is the distance between camera and apex of the cornea, sensor_dims which is the dimensions of the camera, f_len which is focal length of the camera, f_gap1 which maps 1/mire-21-radius to gap 1, start_angle which is starting meridian, end_angle which is the ending meridian, jump is the difference between consecutive angles which are obtained after processing mires, upsample which denotes whether the image should be upsampled or not, n_mires which is number of mires detected, zernike_degree is the degree of zernike polynomial which is used for fitting, test_name which holds the date on which the experiment is carried out.

preprocess.py -

This file is to perform preprocessing on the mire image. It is used for manually selecting the center of the mire image and segmentation of the mire image. This module contains some functions such as threshold_image(), mouse_click() which is used to record the values of the point on which the cursor is clicked, auto_canny() which is used to find the medians which are used in median filtering, apply_crf() which contains DenseCRF(), DenseCRF() is a python interface which is used to see full health report, GaussianBlur() which is used to reduce noise from the selected frame, crop_around_center() intensifies the the target image (cornea), get_iris_diameter() which is used to calculate the diameter of iris which is used in iris segmentation.

mire_detection.py -

This file is responsible for cleaning the image data and detecting mires. mire is a pattern which is similar to fingerprints. It is the radius of curvature of the ringed projections. it contains functions such as medfilt_clean() which is used for overfiltering and cleaning pixels using median filtering, medfilt_tuple(), curve_fit_fill(), process() which is used to process the corner pixels of the image and clean_points() which is used to reduce unwanted data

camera_size.py -

This module deals with the data of the camera. it takes values such as working distance, dimensions of the camera, dimensions of the hardware device. it contains 2 functions get_slope(), get_arc_step_params()

arc_step_method.py - it is used to run the arc step method

get_maps.py -

This module is used to draw maps with respect to the mire image patterns. The 2 main maps that corneal topography uses are tangential map and axial map. This module has the following functions `generate_tan_map()` which is used to generate tangential maps. tangential maps are used to determine the power and curvature of each point on the cornea, `generate_axial_map()` which is used to degenerate axial map. axial maps are used to determine the base curve selection of the cornea

metrics.py -

This module is responsible for calculating the external factors that affect the image such as tilt.

utils.py –

This module is responsible for coloring the plots which we get. These coloring will take place based on the plot line.

Input = Image collected via our setup - SmartKC

Output = Corneal Topography heatmaps with quantitative
sim-K values

Corneal Topography Heatmaps:
Sim-K values:

To compute the corneal topography from the input image, four sets of data points are needed:

- (1) camera parameters
- (2) pixel location of the mires on the cornea
- (3) 3D location of the Placido rings with respect to the cornea
- (4) the working distance

Basically, extracts mapping between mires pixel location and corresponding Placido rings. Camera parameters - known from smartphone specifications

To compute the location of Placido rings and working distance, we need to know two measurements—
`gapbase` - distance between the camera and base of the placido attachment
`gaptop` - distance between the corneal apex and top of the placido attachment.

Corneal apex - These fully determine the two sets of data points, as from the 3D print specifications, the distance of Placido rings from the base of the attachment is known

Working distance - is the sum of `gapbase`, `gaptop` and placido attachment length.

The four sets of data points are then combined using the Arc-Step method and Zernike polynomialbased surface fitting to generate the corneal topography

Pre-processing:

Step -1: Ic: Image capture and crop

Input image resolution → 3000x4000 pixels (width*height)

Resize:

corneal region required → cropped image is considered = 500*500 pixels

Step-2: Ig: Cropped image to GrayScale:

grayscale simplifies the algorithm and reduces computational requirements.

$I_G = (0.299 I_R C + 0.587 I_G C + 0.114 I_B)$ where I_R , I_G , I_B are the red, green and blue color channels of I_C

Step-3: Mire detection:

Identify and segment out the mire pixels.

1. Noise reduction
2. Normalize noise reduced image
3. algorithm to enhance mire reflections.
4. Mires → similar to fingerprints
5. We used fingerprint enhancement for mire enhancement. The enhancement algorithm estimates local orientation by computing gradients, and applies oriented Gabor filters to enhance the mires.
6. Binary mire image → Enhanced image segmented using Binary thresholding
7. Estimate center of image of the cornea region

TESTING

in main.py module add the running script :

```
“python main.py --n_mires 22 --working_distance 75.0 --camera_params "4.8 6.4 4.755" --  
model_file <placido_model_dimensions> --base_dir <image_dir>”
```

```

27_10_2022\kc_eye_1
kc_eye_1_axial_map_overla...
kc_eye_1_mp_clean.png
kc_eye_1_mp.png
kc_eye_1_out_clean_inv_filt...
kc_eye_1_out_clean_inv.png
kc_eye_1_out_col.png
kc_eye_1_out_edge_inv_filt...
kc_eye_1_out_edge_inv.png
kc_eye_1_out_gray.png
kc_eye_1_out_masked.png
kc_eye_1_tan_map_overlay....
kc_eye_1_tilt_factor.png
plots.png
> data
> OUTLINE
> TIMELINE

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Microsoft Windows [Version 10.0.22621.674]
(c) Microsoft Corporation. All rights reserved.

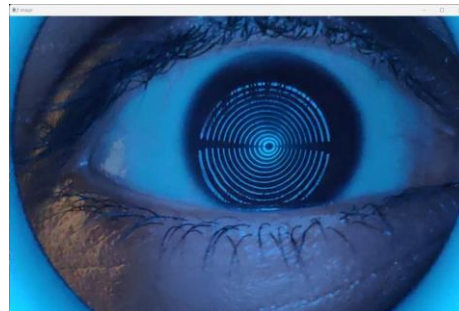
C:\Users\shifa\OneDrive\Documents\SmartKC-A-Smartphone-based-Corneal-Topographer-main>python main.py --n_mires 22 --w
orking_distance 75.0 --camera_params "4.8 6.4 4.755" --model_file "C:\Users\shifa\OneDrive\Documents\SmartKC-A-Smartp
hone-based-Corneal-Topographer-main\ring_distribution.txt" --base_dir "C:\Users\shifa\OneDrive\Documents\SmartKC-A-Sm
artphone-based-Corneal-Topographer-main"
base_dir: kc_eye_1.jpg

Center selected: 678 340
Processing ...
Processing Done!
Cleaning ...
Cleaning Done!
Test Complete!

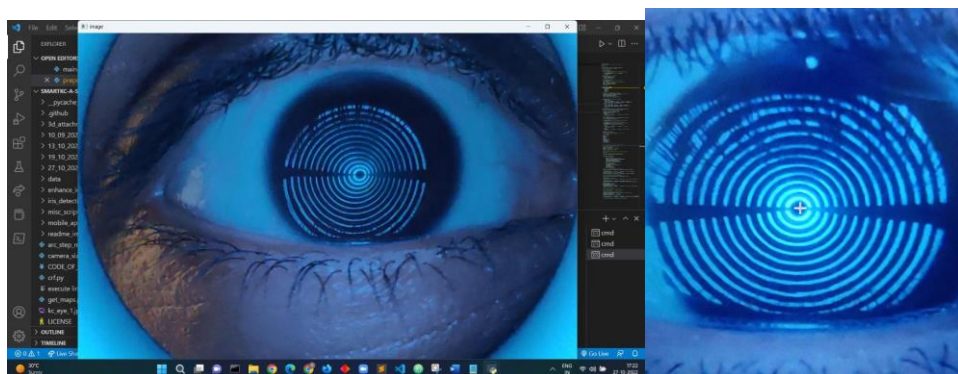
C:\Users\shifa\OneDrive\Documents\SmartKC-A-Smartphone-based-Corneal-Topographer-main>
C:\Users\shifa\OneDrive\Documents\SmartKC-A-Smartphone-based-Corneal-Topographer-main>

```

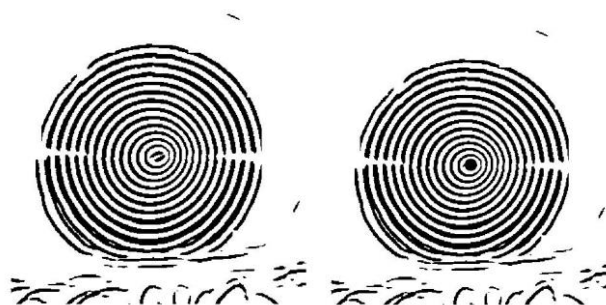
Here we used the kc_eye_1 image,



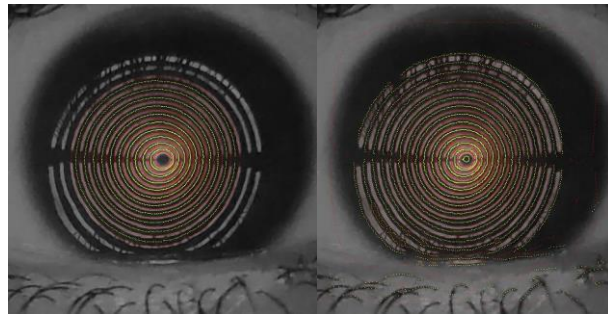
Once the running script is added, then an image will pop out on to which we have to select the center. selecting the center is performed after the kc_eye_1 image is preprocessed



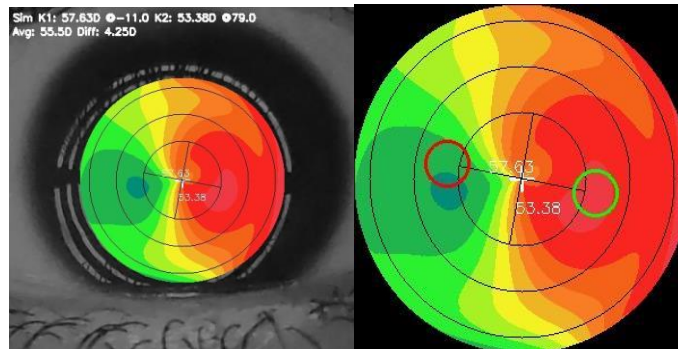
after the center is selected, the unnecessary data is removed



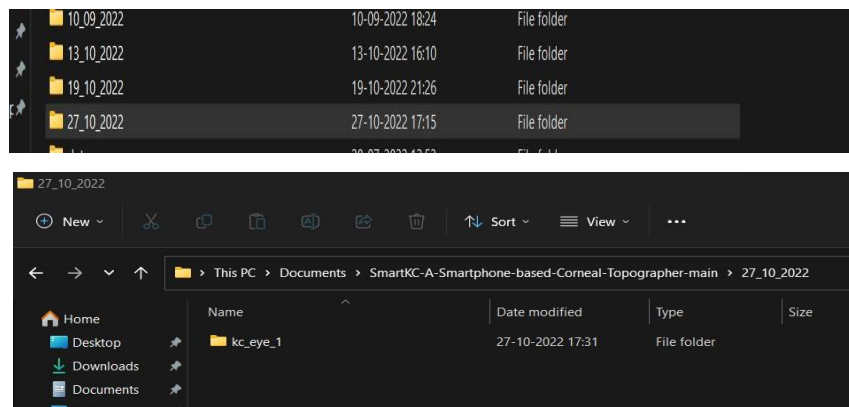
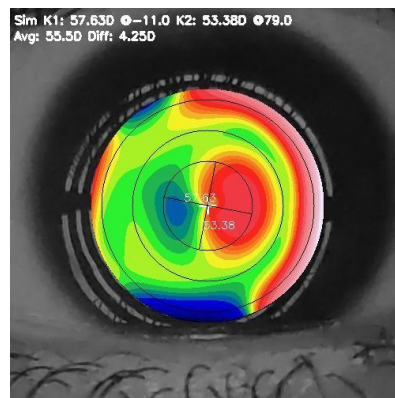
once the images are cleaned, the mires are detected and these mires are then segmented

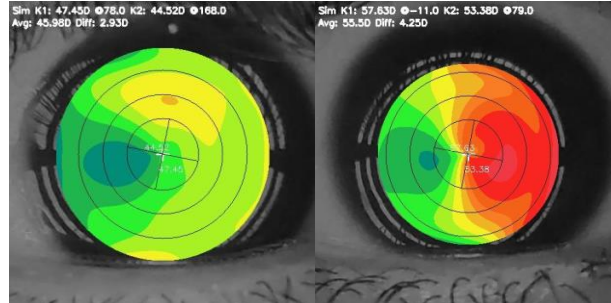
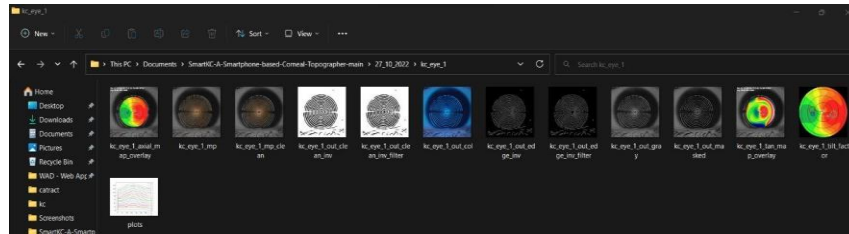


with the help of the mires we will plot the tangential and axial maps and with these maps we will build a plot for the shape of cornea



Coloring to the plot is done using the utils module and all these image gets saved in a folder under the title of the day on which the code is executed





Eye having No Keratoconus and having keratoconus respectively. Here green color denotes neutral i.e there are no abrasions on the cornea whereas the eye having keratoconus has warmer colors they represent the steeps and cooler colors indicate flats.

FOR OTHER EYE DISEASES:

Predictions are very useful in the medical field as they allow highly accurate guesses based on previous historical data, making it easier for the doctors to do their job. ML models learn patterns and recognize relationships between them, from the dataset presented to it. So, when a new data is given to the model, it identifies the data and shows if the patient has the disease or not.

With the development of new technologies in the field of medicine, large amounts of patient data have been gathered and are available to the medical research community. However, one of the most fascinating and difficult problems for doctors is making an accurate prediction of disease. As a result, ML techniques are now widely being used by scientists conducting research in medicine.

Here, in our project we have taken one such dataset for eye diseases.

DATASET: ODIR - 5K

An organized ophthalmic database of 5,000 individuals containing age, color fundus images of the left and right eyes, and diagnostic keywords from doctors is called Ocular Disease Intelligent Recognition (ODIR).

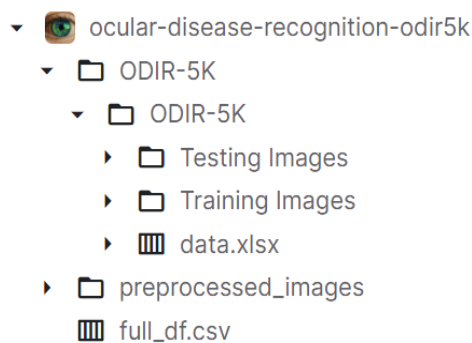
This dataset is intended to reflect a "real-life" set of patient data that Shanggong Medical Technology Co., Ltd. has gathered from various hospitals and medical facilities in China. These institutions use a variety of cameras available on the market, including Canon, Zeiss, and Kowa, to acquire fundus images, producing images with different image resolutions.



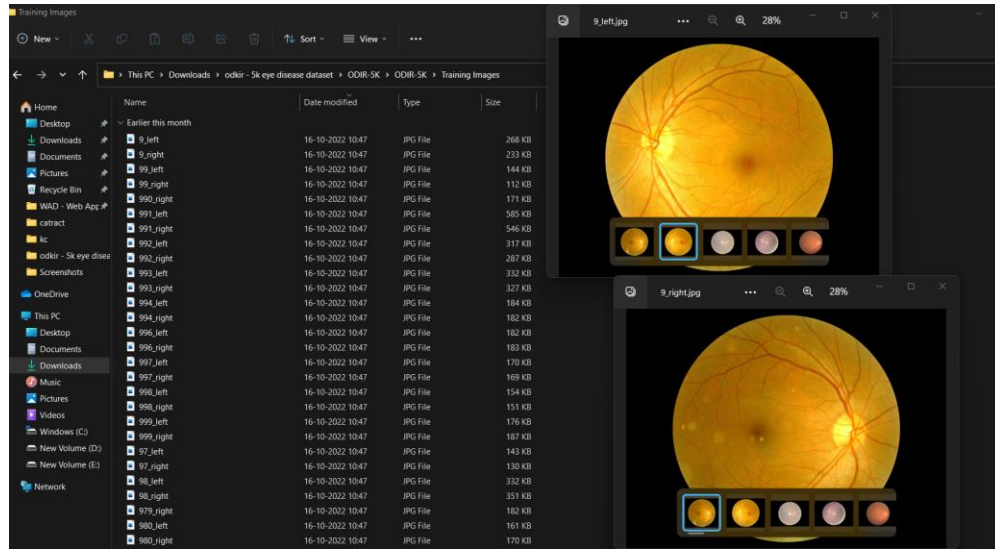
Fundus Imaging: Fundus is the bottom or base of anything. In medicine, it is a general term for the inner lining of a hollow organ. The ocular fundus is the inner lining of the eye made up of the Sensory Retina, the Retinal Pigment Epithelium, Bruch's Membrane, and the Choroid.

With quality control management, trained human readers assigned labels to the annotations. Patients are divided into eight labels, including: Pathological myopia (M), Diabetes (D), Glaucoma (G), Cataract (C), Age-related Macular Degeneration (A), Hypertension (H), Other Diseases/Abnormalities (O)

Dataset in its file directories:



It has Training and Testing Images having left and right eye's fundus images, along with an excel sheet, holding patient data.

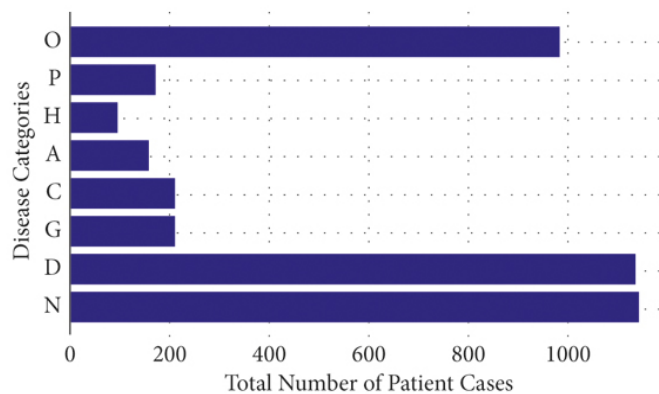


Excel Sheet:

ID	Patient Age	Patient Sex	Left-Fundus	Right-Fundus	Left-Diagnostic Keywords	Right-Diagnostic Keywords	N	D	G	C	A	H	M	O	filepath	labels	target	filename
0	0	69	Female	0_left.jpg	0_right.jpg	cataract	normal fundus	0	0	0	1	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	[N]	[1, 0, 0, 0, 0, 0, 0]	0_right.jpg
1	1	57	Male	1_left.jpg	1_right.jpg	normal fundus	normal fundus	1	0	0	0	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	[N]	[1, 0, 0, 0, 0, 0, 0]	1_right.jpg
2	2	42	Male	2_left.jpg	2_right.jpg	laser spot, moderate non proliferative retinopathy	moderate non proliferative retinopathy	0	1	0	0	0	0	1	../input/ocular-disease-recognition-odir5k/ODI...	[D]	[0, 1, 0, 0, 0, 0, 0]	2_right.jpg

This dataset contains 5000 color fundus photographs, which are divided into training and testing subsets. A little more than 3500 cases are used for training and the rest for testing. For this work, all images are resized to 224×224 .

Distribution of Dataset:



Here, we can see that the dataset is highly unbalanced for different diseases. Hence, multiclass classification wouldn't give accurate results. Hence, we solve this issue by considering binary

classification. (Taking in the same number of images for both the classes, so we do not have an issue of overfitting).

Overfitting → When a deep neural network or machine learning model performs considerably better on training data than on test data, it is said to be overfit.

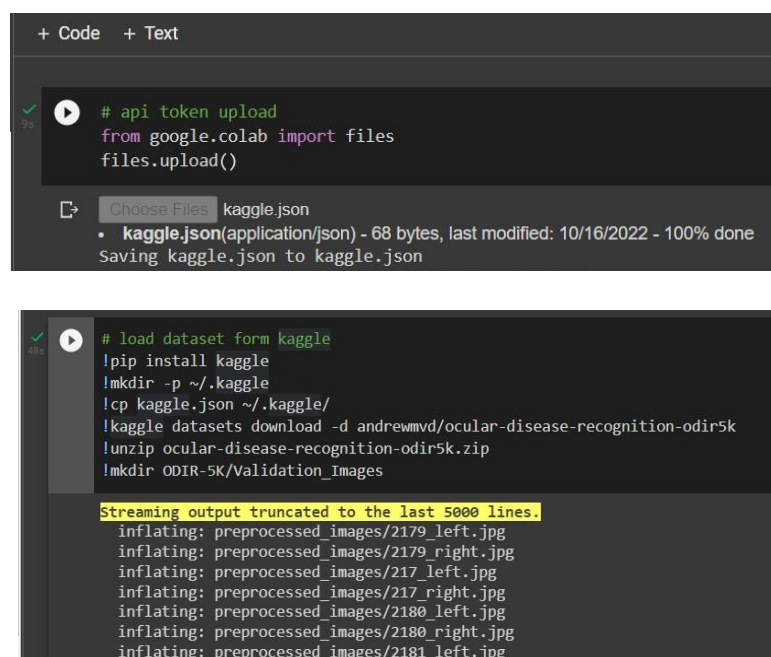
CODE:

The goal is to classify these fundus images into ocular diseases.

Cataract Vs Normal

We will be using Google Colab Notebook, to execute our ML model.

1. Firstly, we import our ODIR-5K dataset from Kaggle and upload it into Google Colab.



```
+ Code + Text

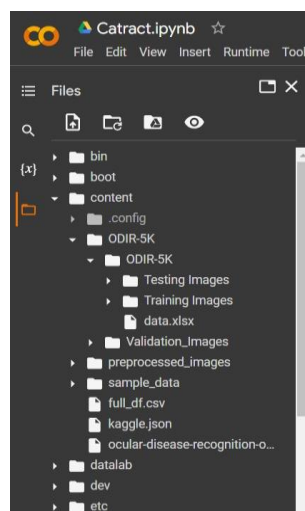
# api token upload
from google.colab import files
files.upload()

Choose Files kaggle.json
• kaggle.json(application/json) - 68 bytes, last modified: 10/16/2022 - 100% done
Saving kaggle.json to kaggle.json

# load dataset form kaggle
!pip install kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!kaggle datasets download -d andrewmvd/ocular-disease-recognition-odir5k
!unzip ocular-disease-recognition-odir5k.zip
!mkdir ODIR-5K/Validation_Images

Streaming output truncated to the last 5000 lines.
inflating: preprocessed_images/2179_left.jpg
inflating: preprocessed_images/2179_right.jpg
inflating: preprocessed_images/217_left.jpg
inflating: preprocessed_images/217_right.jpg
inflating: preprocessed_images/2180_left.jpg
inflating: preprocessed_images/2180_right.jpg
inflating: preprocessed_images/2181_left.jpg
```

File Structure in Colab:



2. import necessary packages

```
[3] import os
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import cv2
import random
from tqdm import tqdm
import matplotlib.pyplot as plt
import tensorflow
from tensorflow import keras
from keras_preprocessing.image import ImageDataGenerator
```

3. Viewing our dataset

[4] df = pd.read_csv("/content/full_df.csv")
df.sample(5)

ID	Patient Age	Patient Sex	Left-Fundus	Right-Fundus	Left-Diagnostic Keywords	Right-Diagnostic Keywords	N	D	G	C	A	H	M	O	filepath	labels	target	filename
5699	3412	70	Female	3412_left.jpg	3412_right.jpg	normal fundus	normal fundus	1	0	0	0	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	[N]	[1, 0, 0, 0, 0, 0]	3412_left.jpg
2317	3203	74	Female	3203_left.jpg	3203_right.jpg	normal fundus	normal fundus	1	0	0	0	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	[N]	[1, 0, 0, 0, 0, 0]	3203_right.jpg
1292	1820	65	Male	1820_left.jpg	1820_right.jpg	dry age-related macular degeneration	dry age-related macular degeneration	0	0	0	0	1	0	0	../input/ocular-disease-recognition-odir5k/ODI...	[A]	[0, 0, 0, 0, 1, 0]	1820_right.jpg
5082	2743	55	Male	2743_left.jpg	2743_right.jpg	normal fundus	normal fundus	1	0	0	0	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	[N]	[1, 0, 0, 0, 0, 0]	2743_left.jpg
4261	1306	72	Male	1306_left.jpg	1306_right.jpg	glaucoma	glaucoma	0	0	1	0	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	[G]	[0, 0, 1, 0, 0, 0]	1306_left.jpg

4. Finding and storing “cataract” images into an array, left and right images are divided into individual arrays. “cataract” label → “C”. i.e, C== 1 for “cataract”.

```
[5] def has_cataract(text):
    if "cataract" in text:
        return 1
    else:
        return 0

[6] df["left_cataract"] = df["Left-Diagnostic Keywords"].apply(lambda x: has_cataract(x))
df["right_cataract"] = df["Right-Diagnostic Keywords"].apply(lambda x: has_cataract(x))

[7] left_cataract = df.loc[(df.C == 1) & (df.left_cataract == 1)]["Left-Fundus"].values
left_cataract[:15]

array(['0_left.jpg', '81_left.jpg', '103_left.jpg', '119_left.jpg',
       '254_left.jpg', '294_left.jpg', '330_left.jpg', '448_left.jpg',
       '465_left.jpg', '477_left.jpg', '553_left.jpg', '560_left.jpg',
       '594_left.jpg', '611_left.jpg', '625_left.jpg'], dtype=object)

[8] right_cataract = df.loc[(df.C == 1) & (df.right_cataract == 1)]["Right-Fundus"].values
right_cataract[:15]

array(['24_right.jpg', '81_right.jpg', '112_right.jpg', '188_right.jpg',
       '218_right.jpg', '345_right.jpg', '354_right.jpg', '477_right.jpg',
       '553_right.jpg', '560_right.jpg', '625_right.jpg', '726_right.jpg',
       '769_right.jpg', '949_right.jpg', '955_right.jpg'], dtype=object)

[9] print("Number of images in left cataract: {}".format(len(left_cataract)))
print("Number of images in right cataract: {}".format(len(right_cataract)))

Number of images in left cataract: 304
Number of images in right cataract: 290
```

5. Similarly, for “normal” images. C==0 for “normal”.

```

[10] left_normal = df.loc[(df.C == 0) & (df["Left-Diagnostic Keywords"] == "normal fundus")]["Left-Fundus"].sample(250,random_state=42).values
right_normal = df.loc[(df.C == 0) & (df["Right-Diagnostic Keywords"] == "normal fundus")]["Right-Fundus"].sample(250,random_state=42).values
right_normal[:15]

array(['2964_right.jpg', '680_right.jpg', '500_right.jpg',
       '2368_right.jpg', '2820_right.jpg', '2769_right.jpg',
       '2696_right.jpg', '2890_right.jpg', '940_right.jpg',
       '2553_right.jpg', '3371_right.jpg', '3042_right.jpg',
       '919_right.jpg', '3427_right.jpg', '379_right.jpg'], dtype=object)

```

6. merging both left and right image file arrays

```

[11] cataract = np.concatenate((left_cataract,right_cataract),axis=0)
normal = np.concatenate((left_normal,right_normal),axis=0)

[12] print(len( Cataract ),len( normal ))

594 500

```

7. creating our own dataset, for binary classification from the given huge dataset. collecting images only of cataract and normal. First we resize the images, and then converting images to arrays along with their labels to get tabular data, to fit it to our training model.

create_dataset():

```

[13] from keras_preprocessing.image import load_img, img_to_array
dataset_dir = "/content/preprocessed_images"
image_size=224
labels = []
dataset = []
def create_dataset(image_category,label):
    for img in tqdm(image_category):
        image_path = os.path.join(dataset_dir,img)
        try:
            image = cv2.imread(image_path,cv2.IMREAD_COLOR)
            image = cv2.resize(image,(image_size,image_size))
        except:
            continue

        dataset.append([np.array(image),np.array(label)])
    random.shuffle(dataset)
    return dataset

```

datasets:

```

[14] dataset = create_dataset( Cataract,1)
100%|██████████| 594/594 [00:02<00:00, 247.03it/s]

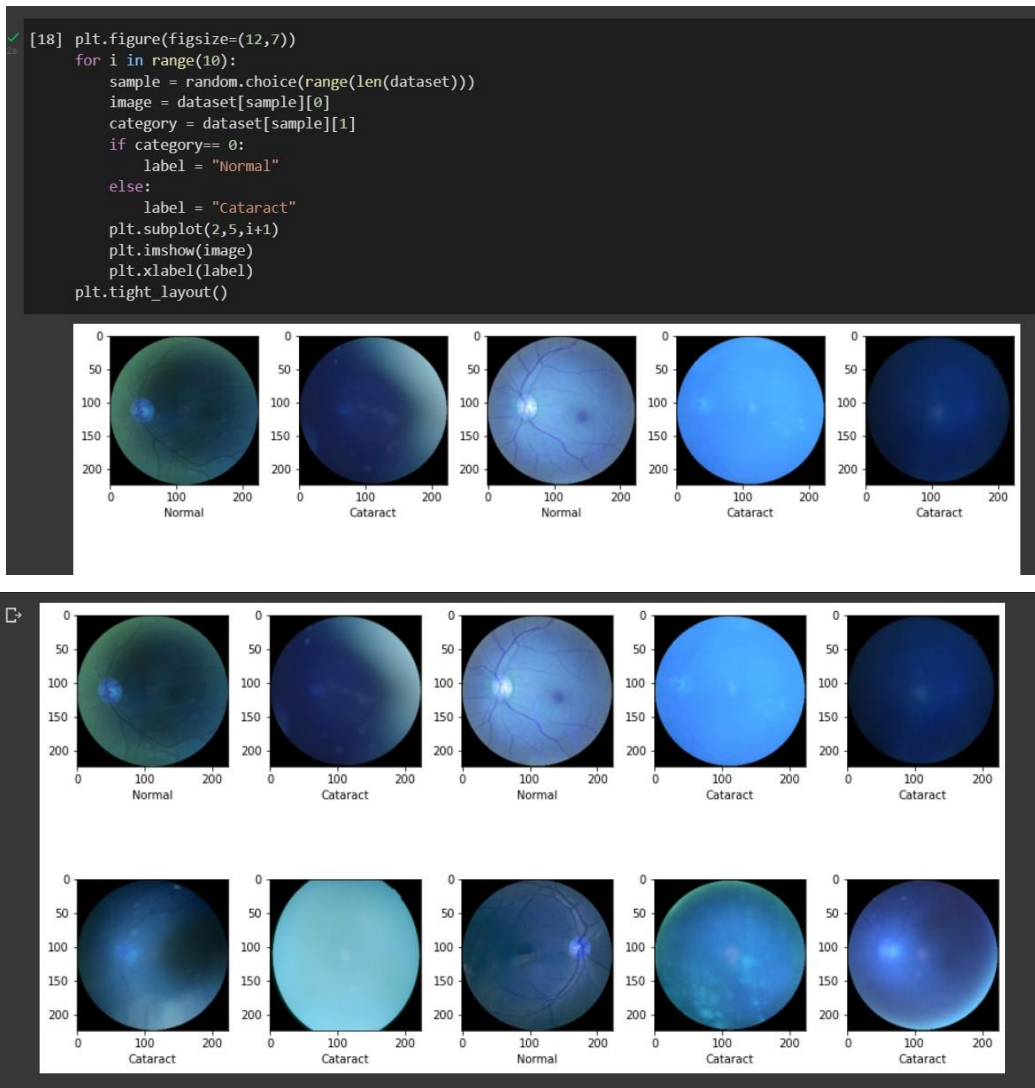
[15] len(dataset)
588

[16] dataset = create_dataset( normal,0)
100%|██████████| 500/500 [00:01<00:00, 253.74it/s]

[17] len(dataset)
1088

```

8. Viewing some images from dataset



9. Dividing dataset into x(features) and y(target)

```
[19] x = np.array([i[0] for i in dataset]).reshape(-1,image_size,image_size,3)
      y = np.array([i[1] for i in dataset])

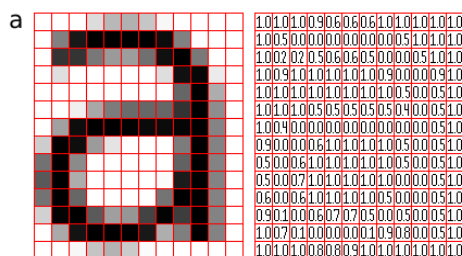
[20] from sklearn.model_selection import train_test_split
      x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

MODEL:

Since we are dealing with images, we will be using various CNN models to test accuracy, and choose the best one.

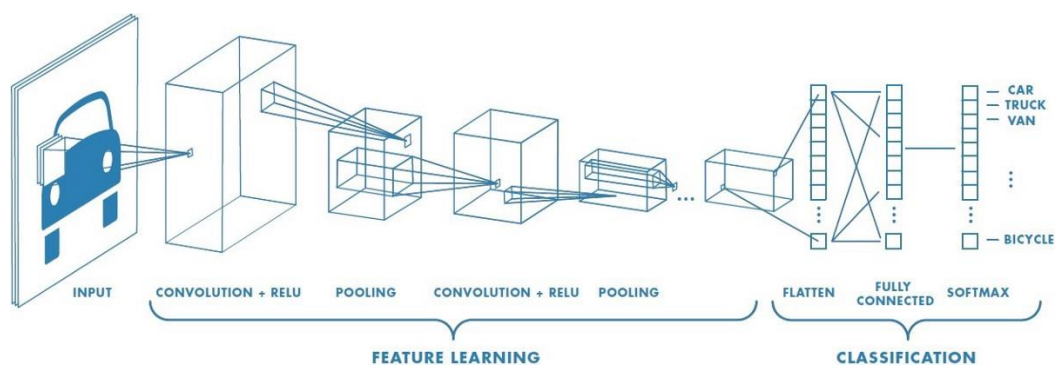
CNN: A neural network type called a convolutional neural network, or CNN or ConvNet, is particularly adept at processing input with a grid-like architecture, like an image. A digital image is a binary representation of visual data. It is made up of a grid-like arrangement of pixels, each of

which has a pixel value to indicate how bright and what color it should be.



There are various types of CNN models like the ResNet, GoogleNet/Inception Net, VGG19, VGG16, DenseNet, Xception, etc.,

CNN architecture:

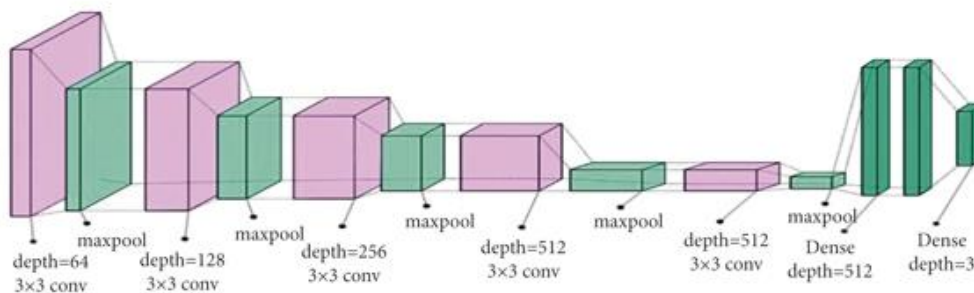


10. Creating Model:

We will be using the VGG19 model for our project here.

VGG19 → CNN model with 19 deep layers

Architecture of VGG19:



```

MODEL

[21] from tensorflow.keras.applications.vgg19 import VGG19
     vgg = VGG19(weights="imagenet", include_top = False, input_shape=(image_size, image_size, 3))

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [=====] - 1s 0us/step

[22] for layer in vgg.layers:
     layer.trainable = False

[23] from tensorflow.keras import Sequential
     from tensorflow.keras.layers import Flatten, Dense
     model = Sequential()
     model.add(vgg)
     model.add(Flatten())
     model.add(Dense(1, activation="sigmoid"))

```

- Sequential() → It's a layer by layer structure arranged in a sequence, with each layer having one tensor input and one tensor output and there is no layer sharing between them. A CNN can be instantiated as a Sequential model because each layer has exactly one input and output and is stacked together to form the entire network.
- Flatten() → It is used to convert 2D array input into 1D array.
- Dense() → It is used to add layers in the model.
- Sigmoid → we use sigmoid activation function, since we are trying to do binary classification. Sigmoid is commonly used for predicting probabilities that are always between 0 and 1. Binary classification being 0-1 classification, sigmoid is used in the output layer.

$$S(x) = \frac{1}{1 + e^{-x}}$$

11. looking at our model

```

[24] model.summary()

Model: "sequential"

```

Layer (type)	Output Shape	Param #
vgg19 (Functional)	(None, 7, 7, 512)	20024384
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 1)	25089

```

=====
Total params: 20,049,473
Trainable params: 25,089
Non-trainable params: 20,024,384

```


12. compile model

```
[25] model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])

[26] from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping
checkpoint = ModelCheckpoint("vgg19.h5",monitor="val_acc",verbose=1,save_best_only=True,
                             save_weights_only=False,period=1)
earlystop = EarlyStopping(monitor="val_acc",patience=5,verbose=1)

WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify the frequency in number of batches seen.
```

- optimizer → used to give results faster
- loss → penalty for bad prediction, binary_crossentropy → for binary classification
- metrics → judge performance of model
- callbacks → here used to avoid overfitting, modelcheckpoint → to save model weights and use them and load later to continue the training from the state saved, earlystopping → it stops to train the model, when a monitored metric has stopped improving.

13. Fit model and run epochs

```
[27] history = model.fit(x_train,y_train,batch_size=32,epochs=15,validation_data=(x_test,y_test),
                        verbose=1,callbacks=[checkpoint,earlystop])

Epoch 1/15
28/28 [=====] - ETA: 0s - loss: 1.4384 - accuracy: 0.8977WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 23s 329ms/step - loss: 1.4384 - accuracy: 0.8977 - val_loss: 1.0554 - val_accuracy: 0.9083
Epoch 2/15
28/28 [=====] - ETA: 0s - loss: 0.3549 - accuracy: 0.9575WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 207ms/step - loss: 0.3549 - accuracy: 0.9575 - val_loss: 0.5664 - val_accuracy: 0.9450
Epoch 3/15
28/28 [=====] - ETA: 0s - loss: 0.1213 - accuracy: 0.9759WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 208ms/step - loss: 0.1213 - accuracy: 0.9759 - val_loss: 0.5686 - val_accuracy: 0.9587
Epoch 4/15
28/28 [=====] - ETA: 0s - loss: 0.0322 - accuracy: 0.9908WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 209ms/step - loss: 0.0322 - accuracy: 0.9908 - val_loss: 0.5834 - val_accuracy: 0.9495
Epoch 5/15
28/28 [=====] - ETA: 0s - loss: 0.0091 - accuracy: 0.9977WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 210ms/step - loss: 0.0091 - accuracy: 0.9977 - val_loss: 0.4893 - val_accuracy: 0.9633
Epoch 6/15
28/28 [=====] - ETA: 0s - loss: 0.0061 - accuracy: 0.9828WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 216ms/step - loss: 0.0061 - accuracy: 0.9828 - val_loss: 0.5484 - val_accuracy: 0.9633
Epoch 7/15
28/28 [=====] - ETA: 0s - loss: 0.0231 - accuracy: 0.9931WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 212ms/step - loss: 0.0231 - accuracy: 0.9931 - val_loss: 0.4718 - val_accuracy: 0.9633
Epoch 8/15
28/28 [=====] - ETA: 0s - loss: 0.0043 - accuracy: 0.9989WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 212ms/step - loss: 0.0043 - accuracy: 0.9989 - val_loss: 0.4718 - val_accuracy: 0.9633
Epoch 9/15
28/28 [=====] - ETA: 0s - loss: 1.4920e-05 - accuracy: 1.0000WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 221ms/step - loss: 1.4920e-05 - accuracy: 1.0000 - val_loss: 0.6068 - val_accuracy: 0.9541
Epoch 10/15
28/28 [=====] - ETA: 0s - loss: 1.3178e-05 - accuracy: 1.0000WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 222ms/step - loss: 1.3178e-05 - accuracy: 1.0000 - val_loss: 0.6058 - val_accuracy: 0.9541
Epoch 11/15
28/28 [=====] - ETA: 0s - loss: 1.2058e-05 - accuracy: 1.0000WARNING:tensorflow:Can save best model only with val_acc available.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_acc
28/28 [=====] - 6s 224ms/step - loss: 1.2058e-05 - accuracy: 1.0000 - val_loss: 0.6067 - val_accuracy: 0.9541
```

RESULTS:

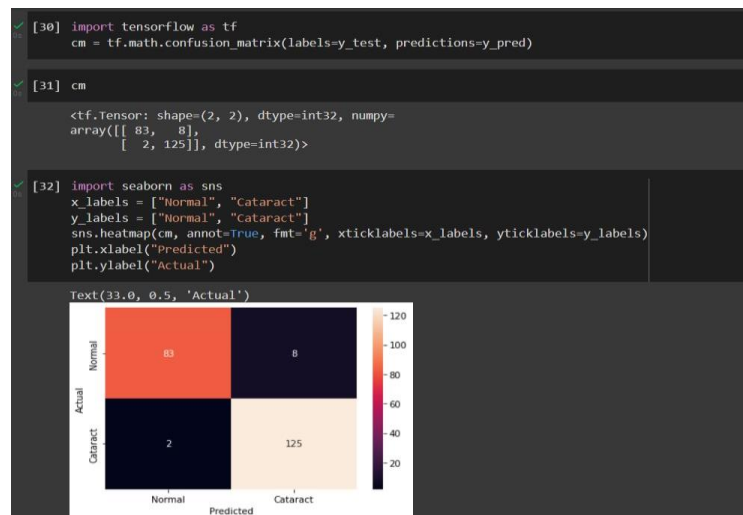
14. loss and accuracy score

```
[ ] loss,accuracy = model.evaluate(x_test,y_test)
print("loss:",loss)
print("Accuracy:",accuracy)

7/7 [=====] - 1s 173ms/step - loss: 0.6067 - accuracy: 0.9541
loss: 0.6067376136779785
Accuracy: 0.9541284441947937
```

15. Confusion matrix → a table to define the performance of the classification algorithm.

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN



16. Classification report

```
[ ] accuracy_score(y_test,y_pred)

0.9541284403669725

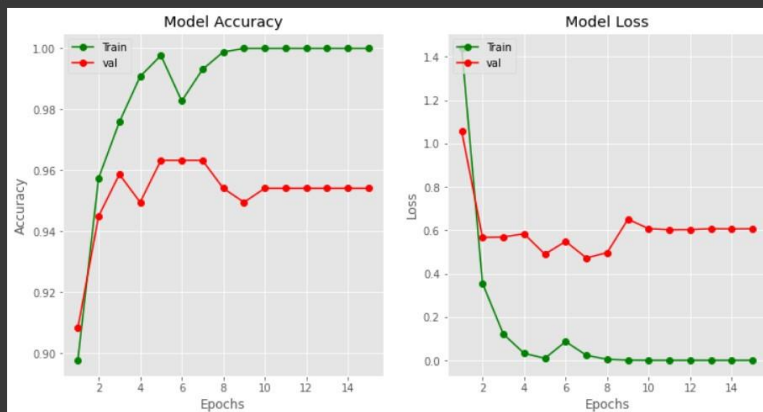
[ ] print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.98	0.91	0.94	91
1	0.94	0.98	0.96	127
accuracy			0.95	218
macro avg	0.96	0.95	0.95	218
weighted avg	0.96	0.95	0.95	218

17. Learning Curve

```
plt.style.use('ggplot')
fig = plt.figure(figsize=(12,6))
epochs = range(1,16)
plt.subplot(1,2,1)
plt.plot(epochs,history.history["accuracy"],"go-")
plt.plot(epochs,history.history["val_accuracy"],"ro-")
plt.title("Model Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend(["Train","val"],loc = "upper left")

plt.subplot(1,2,2)
plt.plot(epochs,history.history["loss"],"go-")
plt.plot(epochs,history.history["val_loss"],"ro-")
plt.title("Model Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend(["Train","val"],loc = "upper left")
plt.show()
```



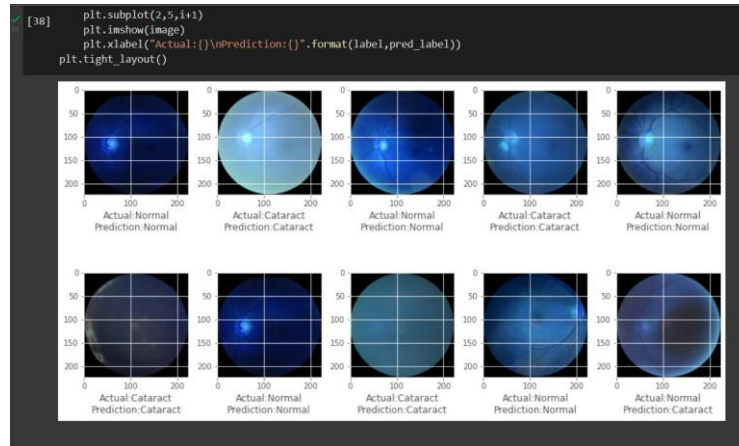
18. Prediction: evaluating model on test data

```
[38] plt.figure(figsize=(12,7))
for i in range(10):
    sample = random.choice(range(len(x_test)))
    image = x_test[sample]
    category = y_test[sample]
    pred_category = y_pred[sample]

    if category== 0:
        label = "Normal"
    else:
        label = "Cataract"

    if pred_category== 0:
        pred_label = "Normal"
    else:
        pred_label = "cataract"

    plt.subplot(2,5,i+1)
    plt.imshow(image)
    plt.xlabel("Actual:{}\nPrediction:{}".format(label,pred_label))
plt.tight_layout()
```

Similarly, we can do binary classification for other diseases too. Using the same implementation steps.

Glaucoma Vs Normal:

Viewing image files with glaucoma and normal in the dataset:

```
def has_glaucoma(text):
[ ] if "glaucoma" in text:
    return 1
    else:
    return 0

[ ] df["left_glaucoma"] = df["left-Diagnostic Keywords"].apply(lambda x: has_glaucoma(x))
df["right_glaucoma"] = df["Right-Diagnostic Keywords"].apply(lambda x: has_glaucoma(x))

[ ] left_glaucoma = df.loc[(df.G == 1) & (df.left_glaucoma == 1)]["Left-Fundus"].values
left_glaucoma[:15]

array(['95_left.jpg', '153_left.jpg', '167_left.jpg', '178_left.jpg',
       '247_left.jpg', '365_left.jpg', '583_left.jpg', '625_left.jpg',
       '746_left.jpg', '931_left.jpg', '1202_left.jpg', '1210_left.jpg',
       '1211_left.jpg', '1212_left.jpg', '1213_left.jpg'], dtype=object)

[ ] right_glaucoma = df.loc[(df.G == 1) & (df.right_glaucoma == 1)]["Right-Fundus"].values
right_glaucoma[:15]

array(['43_right.jpg', '167_right.jpg', '238_right.jpg', '247_right.jpg',
       '365_right.jpg', '583_right.jpg', '746_right.jpg', '931_right.jpg',
       '1209_right.jpg', '1210_right.jpg', '1211_right.jpg', '1212_right.jpg',
       '1213_right.jpg', '1214_right.jpg', '1215_right.jpg'], dtype=object)

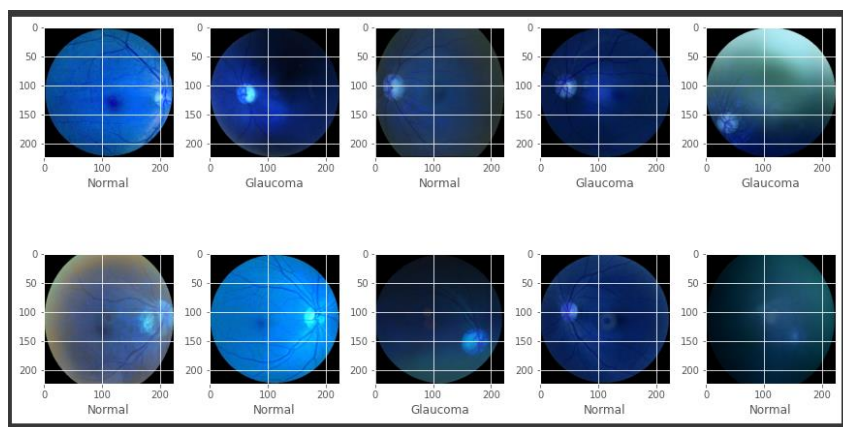
print("Number of images in left glaucoma: {}".format(len(left_glaucoma)))
print("Number of images in right glaucoma: {}".format(len(right_glaucoma)))

Number of images in left glaucoma: 332
Number of images in right glaucoma: 284

[ ] left_normal = df.loc[(df.G == 0) & (df["Left-Diagnostic Keywords"] == "normal fundus")]["Left-Fundus"].sample(250,random_state=42).values
right_normal = df.loc[(df.G == 0) & (df["Right-Diagnostic Keywords"] == "normal fundus")]["Right-Fundus"].sample(250,random_state=42).values
right_normal[:15]

array(['3343_right.jpg', '906_right.jpg', '3406_right.jpg',
       '2974_right.jpg', '3373_right.jpg', '3373_right.jpg',
       '2446_right.jpg', '2967_right.jpg', '3197_right.jpg',
       '2496_right.jpg', '3253_right.jpg', '2916_right.jpg',
       '2887_right.jpg', '363_right.jpg'], dtype=object)
```

Few Images in dataset:



Loss, Accuracy score and classification_report:

```
[ ] loss,accuracy = model.evaluate(x_test,y_test)
print("loss:",loss)
print("Accuracy:",accuracy)

7/7 [=====] - 1s 170ms/step - loss: 0.6394 - accuracy: 0.8789
loss: 0.6394011974334717
Accuracy: 0.878923773765564

[ ] from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
y_pred = (model.predict(x_test) > 0.5).astype("int32")

7/7 [=====] - 1s 166ms/step

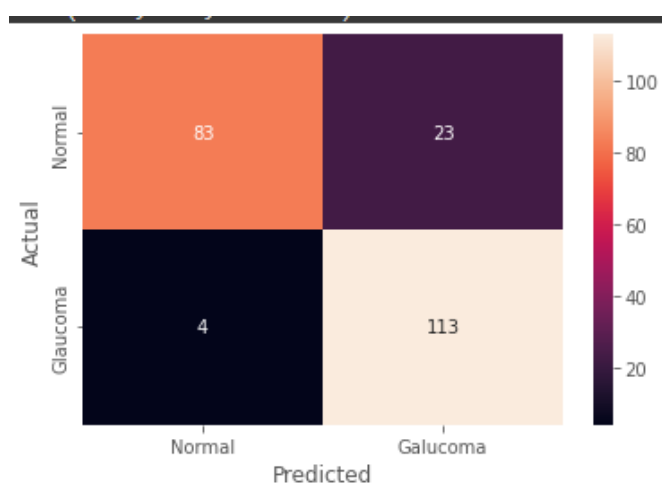
[ ] accuracy_score(y_test,y_pred)

0.8789237668161435

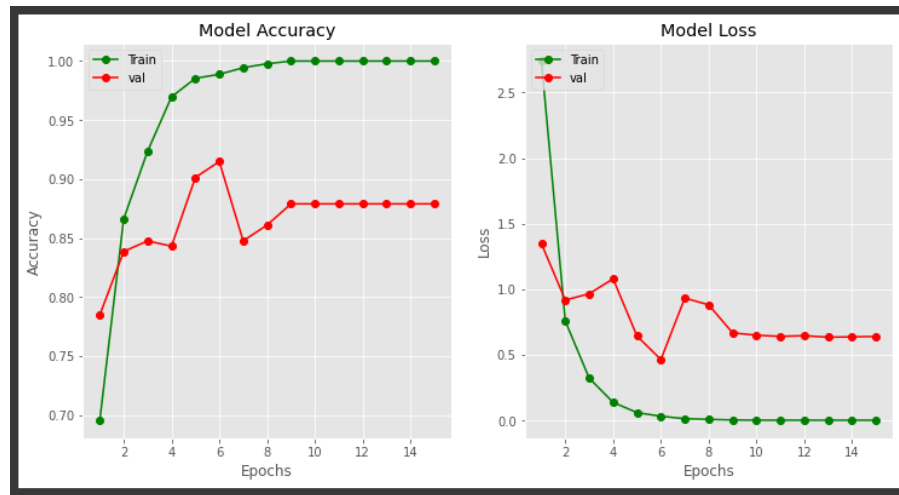
[ ] print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.95	0.78	0.86	106
1	0.83	0.97	0.89	117
accuracy			0.88	223
macro avg	0.89	0.87	0.88	223
weighted avg	0.89	0.88	0.88	223

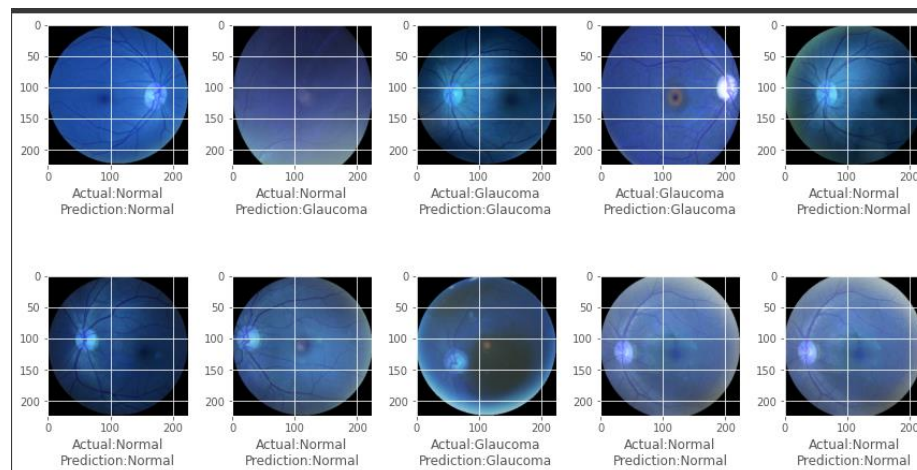
Confusion Matrix



Learning Curve



Prediction Output:



AMD Vs Normal:

Viewing image files with AMD and normal in the dataset:

```
[ ] def has_amd(text):
    if "macular degeneration" in text:
        return 1
    else:
        return 0

[ ] df["left_amd"] = df["Left-Diagnostic Keywords"].apply(lambda x: has_amd(x))
df["right_amd"] = df["Right-Diagnostic Keywords"].apply(lambda x: has_amd(x))

[ ] left_amd = df.loc[(df.A == 1) & (df.left_amd == 1)][["Left-Fundus"].values
left_amd[:15]

array(['43_left.jpg', '48_left.jpg', '53_left.jpg', '55_left.jpg',
      '102_left.jpg', '126_left.jpg', '136_left.jpg', '152_left.jpg',
      '158_left.jpg', '162_left.jpg', '164_left.jpg', '168_left.jpg',
      '170_left.jpg', '210_left.jpg', '212_left.jpg'], dtype=object)

[ ] right_amd = df.loc[(df.A == 1) & (df.right_amd == 1)][["Right-Fundus"].values
right_amd[:15]

array(['43_right.jpg', '48_right.jpg', '53_right.jpg', '55_right.jpg',
      '71_right.jpg', '102_right.jpg', '126_right.jpg', '152_right.jpg',
      '153_right.jpg', '158_right.jpg', '160_right.jpg', '162_right.jpg',
      '164_right.jpg', '168_right.jpg', '178_right.jpg'], dtype=object)

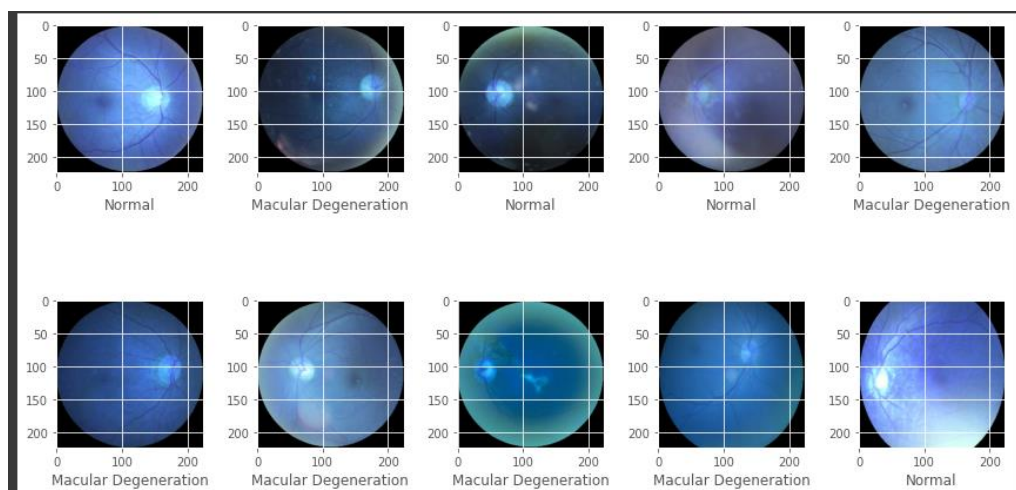
[ ] print("Number of images in left amd: {}".format(len(left_amd)))
print("Number of images in right amd: {}".format(len(right_amd)))

Number of images in left amd: 266
Number of images in right amd: 285

[ ] left_normal = df.loc[(df.A == 0) & (df["Left-Diagnostic Keywords"] == "normal fundus")][["Left-Fundus"].sample(250, random_state=42).values
right_normal = df.loc[(df.A == 0) & (df["Right-Diagnostic Keywords"] == "normal fundus")][["Right-Fundus"].sample(250, random_state=42).values
right_normal[:15]

array(['2760_right.jpg', '2981_right.jpg', '2425_right.jpg',
      '3250_right.jpg', '2505_right.jpg', '939_right.jpg',
      '2714_right.jpg', '285_right.jpg', '215_right.jpg',
      '3429_right.jpg', '2488_right.jpg', '900_right.jpg',
      '2336_right.jpg', '2974_right.jpg', '875_right.jpg'], dtype=object)
```

Few images in dataset:



Loss, Accuracy score and classification_report:

```
[28] loss, accuracy = model.evaluate(x_test, y_test)
print("loss:", loss)
print("Accuracy:", accuracy)

7/7 [=====] - 1s 172ms/step - loss: 0.6450 - accuracy: 0.8720
loss: 0.6449716687202454
Accuracy: 0.8720378875732422

[29] from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
y_pred = (model.predict(x_test) > 0.5).astype("int32")

7/7 [=====] - 1s 177ms/step

[30] accuracy_score(y_test, y_pred)

0.8720379146919431

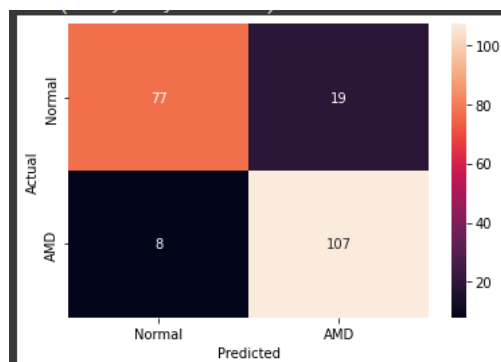
[31] print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

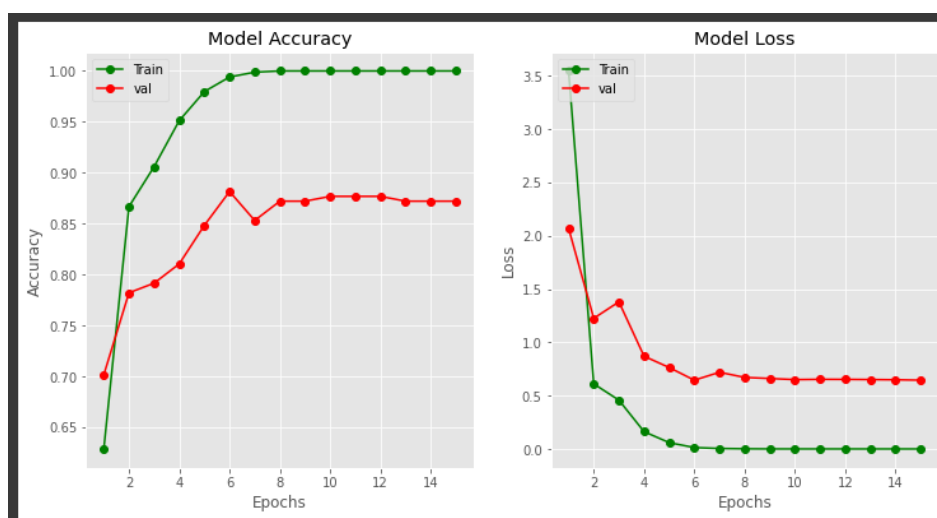
     0       0.91       0.80       0.85         96
     1       0.85       0.93       0.89        115

 accuracy      0.88
 macro avg     0.87
 weighted avg   0.87
```

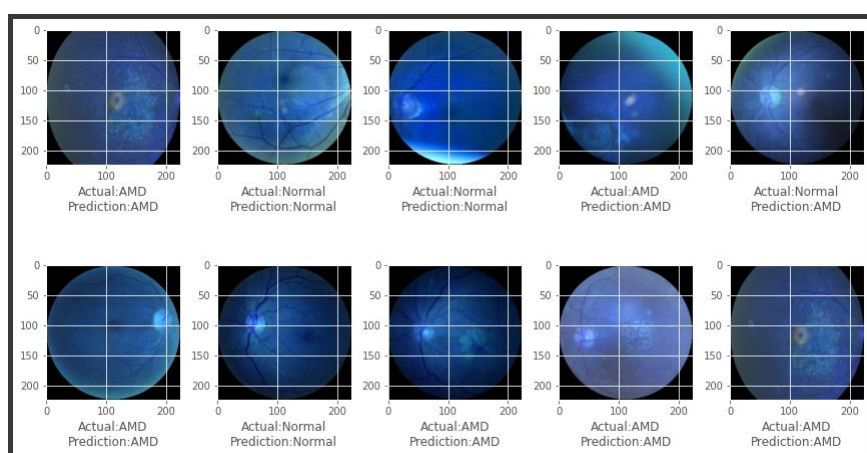
Confusion Matrix



Learning Curve:



Predicted Output:



RESULTS AND DISCUSSIONS:

- model

The prototype which we build can also detect mires if it is made with accurate measurements. thus making the placido arrangement much cheaper and easily available

- smart kc

The smartKc software takes the cornea image and detects mires from it. with the help of these mires it plots axial and tangential maps and from these it plots the map of corneal surface with the help of zernike polynomials.

- other diseases

There are various diseases that our dataset contained, we ran our VGG19 model on 3 such diseases that are commonly found in the Indian population. Using the similar implementation as Cataract Vs Normal, we were able to find the results below for Glaucoma Vs Normal and AMD Vs Normal

S.no	Classification	No of patients with that disease	Accuracy	Loss
1.	Cataract Vs Normal	588	95.41	60.67
2.	Glaucoma Vs Normal	613	87.89	63.94
3.	AMD Vs Normal	551	87.20	64.49

CHALLENGES AND LIMITATIONS:

1. Image capturing
2. Construction of Device - Placido attachment - offset, tilt, working distance
3. Dataset Collection -
 - ODIR - 5K highly unbalanced data, different image resolutions, the other disease category has more than 10 diseases included in it.
 - All these issues will hamper with the accuracy and other metric
 - SMART KC - No data found for smartKC images

CONCLUSION & FUTURE SCOPE:

We have tried to build programs for various eye diseases. For SmartKC, we were successfully able to implement a code that would output corneal topographies of the eye. We used deep learning algorithms like CNN - VGG19 to predict other diseases using binary classification, and secured an accuracy of 95.41%.

In the future scope of this project:

1. We could build mobile/web apps and predict diseases on the UI itself.
2. Collect a proper dataset for our testing prototype SmartKC and apply CNN on the corneal topography images
3. Get a better dataset to apply Multiclass classification on the other diseases from the ODIR-5K dataset.

REFERENCES:

- [1] [Siddhartha Gairola, Murthaza Bohra published in 2021, "SmartKC: Smartphone-based Corneal Topographer for Keratoconus Detection".](#)
- [2] [Larxel published in 2020, "Ocular Disease Detection"](#) - ODIR-5K Dataset
- [3] <https://www.hindawi.com/journals/cin/2022/5007111/>