
# COURSEWORK REPORT

**CS7052 Machine Learning**

Case Study**:**

***Glaucoma Disease Detection***

Course Mentor**:**

**Dr. Elaheh Homayounvala**

Submitted By:

**Pranit Mehta (21051814)**

**Machine Learning**

**CS7052**

*Date: 09-12-22*

# Table of Contents

# Table of Figures

# Glaucoma Disease Detection - Deep Learning Base Methodology

## 1. Abstract

To categorize Glaucoma illnesses, we built an architecture based on the Deep Learning (DL) approach, namely a Convolution Neural Network (CNN). We employed two deep learning algorithms, MobileNetV2 and DensNet201, for identifying and classifying Glaucoma. We used Kaggle dataset for 938 photos of Glaucoma illness from 10 different classes. The Glaucoma dataset was assessed based on the val_loss and val_accuracy of both models. In addition, we gave the classification report in which we compared the accuracy, recall, and f1-score of the two illness classification models. DensNet201 has the lowest val_loss rate and the highest val_accuracy, which are 0.31 and 91.62 %, respectively.

# Introduction

Glaucoma is the second leading cause of blindness worldwide, with 75 million cases in 2020, and it is expected to increase by 30 million in 2040 (Abdullah et al., 2021), (Aamir et al., 2020). Glaucoma can be broadly classified into two categories: Open Angle Glaucoma and Closed Angular Glaucoma. Among them, Open-Angle Glaucoma has a greater number of registered cases all over the world and has no symptoms (Mayro et al., 2020). Open-Angle Glaucoma is a chronic Eye disease that is caused due to the blockage in the flow of liquid called Aqueous humor (Li et al., 2019). Aqueous humor flows inside the eye directly below the optic nerve and keeps the eye pressure low. The blockage in the eye creates an accumulation of liquid and destroys the optic nerve, which carries the information to the brain. When information from the eye to the brain is lost, it results in permanent blindness (Saxena et al., 2020). Glaucoma has no permanent solution in the current medical field, so early detection is the only possibility. Due to its risk of spreading and diagnosis complexity, my research mainly focuses on addressing the glaucoma diagnosis. Figure 1 shows the difference between the normal and Glaucoma eyes (Open Angle Glaucoma) (Gupta et al., 2022).

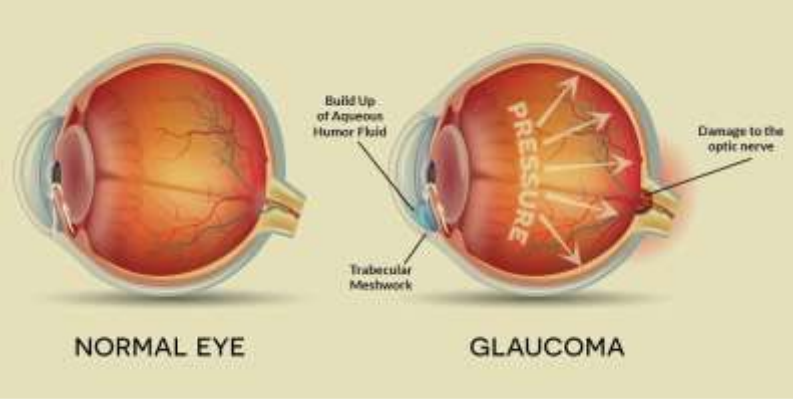



Figure 1 Difference between a Normal Eye and an Eye Affected by Glaucoma

To detect Glaucoma in its earliest stages, the patient is subjected to various tests, such as the Field Examination, Optic head examination, and Pressure Checkup. Among these, the Optic head Examination is the most reliable and efficient method for detecting the presence of Glaucoma due to its cost-effectiveness and simple structure (Thakoor et al., 2019). In the Optic Head Examination, the doctor takes a picture of the human eye structure in the form of a Fundus image (Medical image containing information about eye structure) using a Fundus Camera (Krishna Adithya et al., 2021). Then the doctor tries to estimate the boundaries of the Optic

Disc and Optic Cup to calculate the Optic Cup-to-Disc ratio to predict whether a patient has Glaucoma.

Figure 2 and Figure 3 shows the visual representation of the Fundus image and Optic Disc and Optic Cup boundaries.

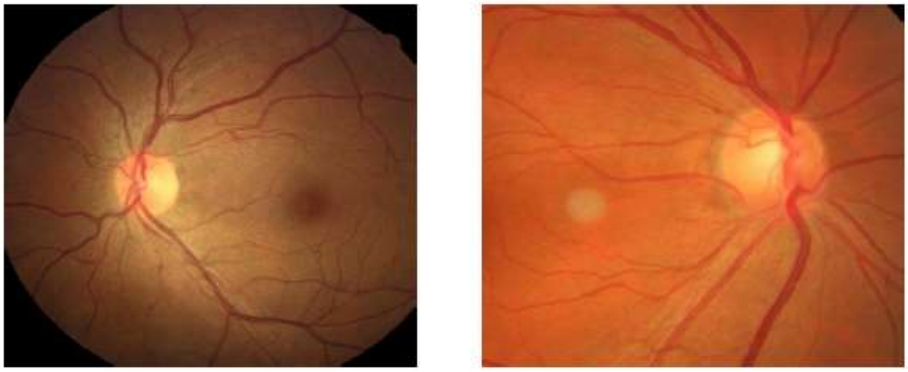



Figure 2 Illustration of Fundus images of a Glaucoma eye and a normal eye

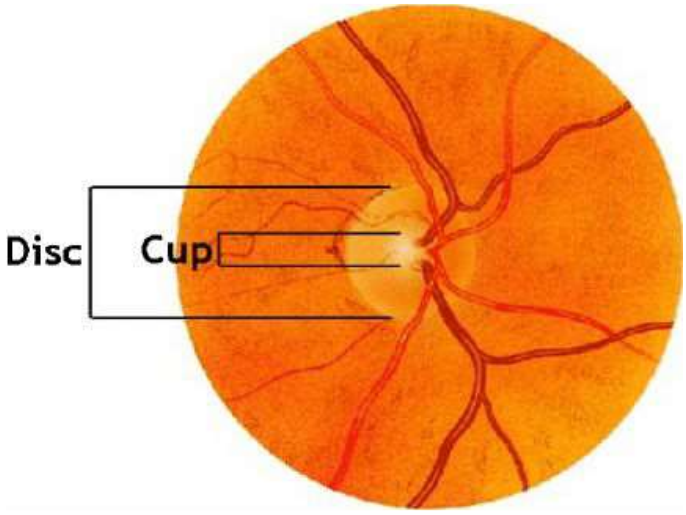



Figure 3 Example of Optic Disc and Cup labeled in a Fundus image

The leading cause of blindness and visual impairment worldwide, Glaucoma currently has no effective cure. The possibility that this condition may be the root cause of blindness wasn't first accepted, but years of study have provided conclusive evidence to support this hypothesis (Vaghjiani et al., 2020). Vision loss from Glaucoma may be avoided if the disease is caught early enough. So, we may state that Glaucoma is the salient chronic eye condition that is the reason for blindness. The urban population is where this chronic illness is having the most

impact. Recent studies have predicted that by 2020, 79 million people will have fallen prey to this illness (Gheisari et al., 2021). Since detecting Glaucoma is so crucial, it's vital to conduct screening eye exams. However, this is laborious and time-consuming when applied to a population as vast as the general population.

When ophthalmologists use a computer-aided system (CADx) to do eye screenings, they can save time and money while improving the quality of care they provide. The diagnosis and analysis of Glaucoma, a chronic eye illness, may be streamlined with these tools, saving time and energy. Because the border of CUP is impacted by glaucoma eye illness, it is difficult for ophthalmologists to distinguish between Glaucoma retinal images and normal glaucoma images, which is why the CADx system should be adopted in clinics and hospitals (Bisneto et al., 2020). A cup-to-disc ratio (CDR) may be used to measure this impact. Retinal fundus imaging and glaucoma stage categorization are challenging because of high-volume screening as shown in Figure 4.

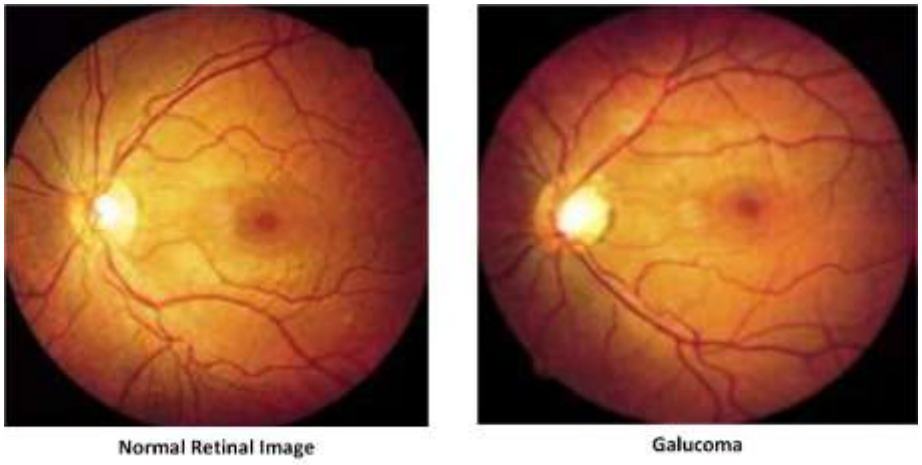



Figure 4 Glaucoma manifesting itself visually: "(a) normal retinal image, (b) glaucoma"

Reduction in the size of the periphery of the retina (PPA). It is, moreover, a sensor for the detection of ocular Glaucoma. In the event of peripapillary atrophy (PPA), the border of the disc shifts when neighboring intensities do (Serte and Serener, 2019). The figure displays the result of PPA on a region-of-interest (ROI) picture of the retinal fundus in a patient with glaucomatous optic nerve damage. PP manifests as a disc boundary variant analogous to rim thinning as shown in Figure 5.

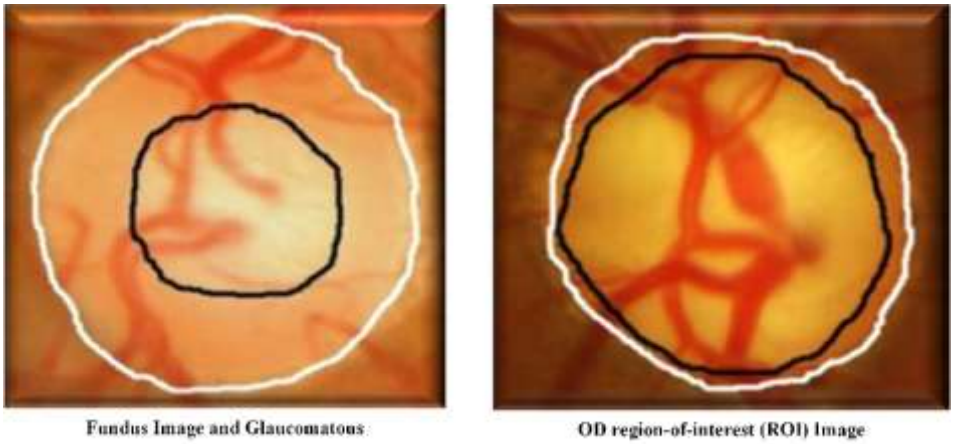



Figure 5 Representative Normal Sample "(a) fundus image and glaucomatous (b) OD region-of-interest (ROI) image"

An early study gave a mechanized answer and classified this chronic illness based on the cup-to-disc ratio (CDR) (Bechar et al., 2018). Therefore, it may be challenging to segregate CUP and Od areas near the disc's edge. Therefore, several image processing methods should be available after feature extraction if we are not planning on segmenting the OD and CUP areas. After creating the picture features, we may use them without further knowledge or training to choose the discriminative features (Kausu et al., 2018). Therefore, it is safe to state that ophthalmologists will have difficulty detecting Glaucoma utilizing CADx devices.

Instead of relying on segmentation methods, Glaucoma may be identified by analyzing the picture characteristics in and around the OD area. Therefore, a method where it may be presupposed that morphological change in OD produced by the illness might lessen the requirement to precisely detect cup and disc boundaries (Raja et al., 2022). Because of this, deep learning's foundational characteristics may be recovered automatically, even by those who lack specialized knowledge in a certain area of image processing.

In this research, we have used the python programming language with the Google Co-lab Platform, where we have developed our models. We have used resources of dataset from google, and for the implementation, we have used the online platform of Google Colab.

# 3 Demonstration of the technique

This experiment shows that deep tuning, or fine-tuning all the trainable layers in the CNN, is the optimal strategy for improving performance. To test how well the CNN performs after being fine-tuned for a period ranging from 1 to 30 iterations, we also conduct a more basic experiment in which all layers are set to trainable mode. It should be noted that this experiment's performance was evaluated using the validation set.

## 3.1 Experiments Result

In the implementation of the CNN models, we have used python programming. In the initial phase, we imported several libraries from Kera's. After this, we mounted our dataset with google drive. In the dataset, we have 938 images based on the 10 glaucoma disease classes. We have used the google colab platform for the implementation. In the co-lab, we used the 12GB RAM resource during the training and testing of the dataset. We have divided our dataset into the 70% for training and 30% for testing of the dataset, 30 epochs, and 30 batch sizes for training time.

## 3.2 Compiling Model

In this section, we have evaluated the val_loss and val_accuracy of the CNN models used in this report, such as MobileNetV2 and DensNet201. After evaluation, we have found that DensNet201 finds less val_loss and gains the best accuracy compared to another CNN model (MobileNetV2), which is 0.0899 and 98.58%, respectively, as shown in Table 1.

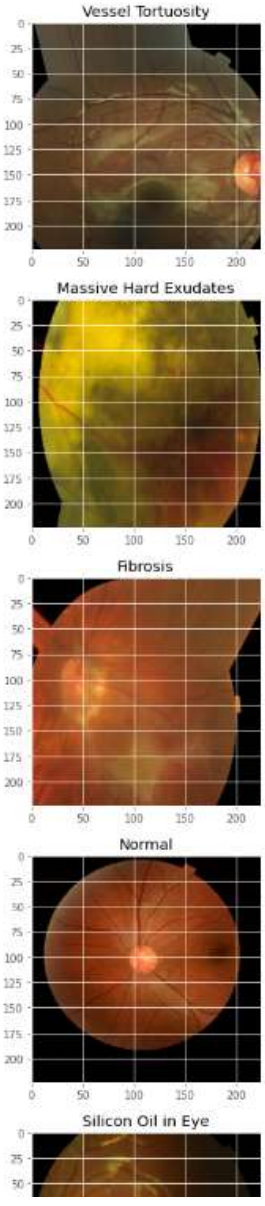
Table 1 The evaluated results of CNN models

| MobileNetV2 | | | DensNet201 | | |
|---|---|---|---|---|---|
| epoch | val_loss | val_accuracy | epoch | val_loss | val_accuracy |
| 1 | 1.3372 | 0.6418 | 1 | 1.4089 | 0.6738 |
| 2 | 0.6876 | 0.7943 | 2 | 0.8630 | 0.7979 |
| 3 | 0.4683 | 0.9078 | 3 | 0.5858 | 0.8759 |
| 4 | 0.3157 | 0.9362 | 4 | 0.4574 | 0.8901 |
| 5 | 0.2422 | 0.9504 | 5 | 0.3516 | 0.9220 |
| 6 | 0.2012 | 0.9610 | 6 | 0.2539 | 0.9574 |
| 7 | 0.1905 | 0.9645 | 7 | 0.2091 | 0.9574 |
| 8 | 0.1587 | 0.9610 | 8 | 0.1933 | 0.9539 |
| 9 | 0.1597 | 0.9681 | 9 | 0.1763 | 0.9681 |
| 10 | 0.1277 | 0.9645 | 10 | 0.1669 | 0.9610 |
| 11 | 0.1444 | 0.9504 | 11 | 0.1554 | 0.9681 |
| 12 | 0.1273 | 0.9574 | 12 | 0.1468 | 0.9681 |
| 13 | 0.1145 | 0.9681 | 13 | 0.1499 | 0.9645 |
| 14 | 0.1188 | 0.9681 | 14 | 0.1476 | 0.9610 |
| 15 | 0.1097 | 0.9716 | 15 | 0.1283 | 0.9787 |
| 16 | 0.1061 | 0.9752 | 16 | 0.1553 | 0.9716 |
| 17 | 0.1228 | 0.9681 | 17 | 0.1210 | 0.9858 |
| 18 | 0.0966 | 0.9716 | 18 | 0.1225 | 0.9823 |
| 19 | 0.1079 | 0.9716 | 19 | 0.1264 | 0.9752 |
| 20 | 0.1353 | 0.9574 | 20 | 0.1152 | 0.9787 |
| 21 | 0.0979 | 0.9752 | 21 | 0.1204 | 0.9787 |
| 22 | 0.1043 | 0.9681 | 22 | 0.1155 | 0.9823 |
| 23 | 0.1273 | 0.9610 | 23 | 0.1165 | 0.9787 |
| 24 | 0.0973 | 0.9681 | 24 | 0.1116 | 0.9752 |
| 25 | 0.1079 | 0.9752 | 25 | 0.1108 | 0.9752 |
| 26 | 0.1025 | 0.9716 | 26 | 0.1053 | 0.9787 |
| 27 | 0.1452 | 0.9574 | 27 | 0.1099 | 0.9716 |
| 28 | 0.0824 | 0.9752 | 28 | 0.0963 | 0.9858 |
| 29 | 0.0921 | 0.9681 | 29 | 0.1116 | 0.9716 |
| 30 | 0.0920 | 0.9787 | 30 | 0.0899 | 0.9858 |

Below is the accuracy and loss after 10 epochs:

```
+ Code  + Text
Epoch 3/10
21/21 [==============================] - 299s 14s/step - loss: 0.9560 - accuracy: 0.6951 - val_loss: 0.7165 - val_accuracy: 0.8440
Epoch 4/10
21/21 [==============================] - 306s 14s/step - loss: 0.6978 - accuracy: 0.7805 - val_loss: 0.5356 - val_accuracy: 0.8901
Epoch 5/10
21/21 [==============================] - 294s 14s/step - loss: 0.6358 - accuracy: 0.8110 - val_loss: 0.4741 - val_accuracy: 0.8972
Epoch 6/10
21/21 [==============================] - 280s 13s/step - loss: 0.5407 - accuracy: 0.8552 - val_loss: 0.3597 - val_accuracy: 0.9255
Epoch 7/10
21/21 [==============================] - 284s 13s/step - loss: 0.4340 - accuracy: 0.8659 - val_loss: 0.3021 - val_accuracy: 0.9326
Epoch 8/10
21/21 [==============================] - 282s 13s/step - loss: 0.3925 - accuracy: 0.8841 - val_loss: 0.2926 - val_accuracy: 0.9291
Epoch 9/10
21/21 [==============================] - 236s 11s/step - loss: 0.3627 - accuracy: 0.8811 - val_loss: 0.2741 - val_accuracy: 0.9326
Epoch 10/10
21/21 [==============================] - 237s 11s/step - loss: 0.3113 - accuracy: 0.9162 - val_loss: 0.2357 - val_accuracy: 0.9362
10/10 [==============================] - 66s 6s/step
['Vessel Tortuosity', 'Massive Hard Exudates', 'Fibrosis', 'Normal', 'Silicon Oil In Eye', 'Fibrosis', 'Vessel Tortuosity', 'Fibrosis',
                              precision    recall  f1-score   support
```

The output of few images from the dataset is shown below.

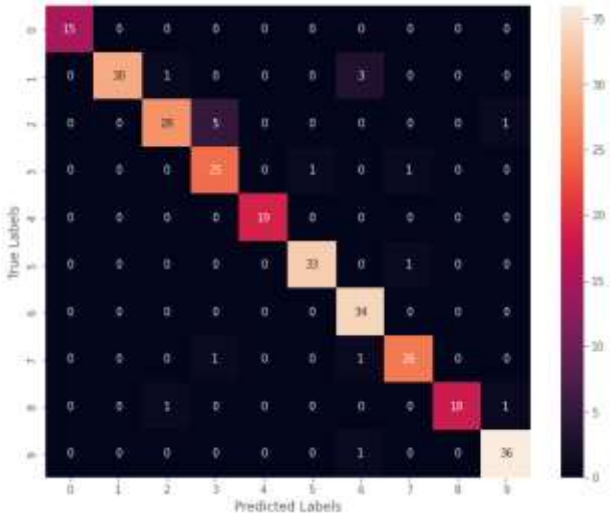


**Performance measurement technique**

The confusion matrix's code is shown below. It aids in testing recall, accuracy, false positives, true positives, and false negatives, true negatives.

```
+ Code  + Text

from sklearn.metrics import confusion_matrix
y_test = np.argmax(testY, axis=1)

conf_mtrx = confusion_matrix(y_test, y_pred)
```

```
[9] # Functions to compute True Positives, True Negatives, False Positives and False Negatives

    def true_positive(y_true, y_pred):

        tp = 0

        for yt, yp in zip(y_true, y_pred):

            if yt == 1 and yp == 1:
                tp += 1
        return tp

    def true_negative(y_true, y_pred):
        tn = 0
        for yt, yp in zip(y_true, y_pred):

            if yt == 0 and yp == 0:
                tn += 1
        return tn

    def false_positive(y_true, y_pred):
        fp = 0
        for yt, yp in zip(y_true, y_pred):

            if yt == 0 and yp == 1:
                fp += 1
        return fp

    def false_negative(y_true, y_pred):
        fn = 0
        for yt, yp in zip(y_true, y_pred):
            if yt == 1 and yp == 0:
                fn += 1
        return fn
```




Confusion matrix

### 3.3 Classification Report

Precision-recall is utilized to evaluate both the MobileNetV2 and DenseNet201 CNN models' classification precision and their ability to generalize across distinct datasets. Regarding golf swings, precision is the rate at which the classifier retrieves correct Disc Swelling and Elevation. At the same time, recall is the rate at which the classifier retrieves true Disc Swelling and Elevation and thus indicates the classifier's sensitivity when faced with plausibly incorrect Disc Swelling and Elevation. When considering both accuracy and recall, the synthesized F1-score assesses classification quality (see Table 2). The figure is generated using two-line graphs representing the loss and classification learning curves on the train and test sets, respectively. There seems to be a good match between the issue and the model, as shown by the Figure 6.

Table 2 Classification report of CNN Models with all glaucoma diseases

| Model Name | MobileNetV2 | | | | DensNet201 | | | |
|---|---|---|---|---|---|---|---|---|
| Glaucoma Diseases Name | precision | recall | f1-score | support | precision | recall | f1-score | support |
| Congenital Disc Abnormality | 1.00 | 1.00 | 1.00 | 29 | 1.00 | 1.00 | 1.00 | 29 |
| Disc Swelling and Elevation | 0.97 | 0.95 | 0.96 | 37 | 1.00 | 0.97 | 0.99 | 37 |
| Fibrosis | 0.4 | 1.00 | 0.97 | 34 | 0.94 | 1.00 | 0.97 | 34 |
| Fundus Neoplasm | 1.00 | 0.92 | 0.96 | 26 | 1.00 | 0.96 | 0.98 | 26 |
| Laser Spots | 1.00 | 1.00 | 1.00 | 33 | 1.00 | 1.00 | 1.00 | 33 |
| Massive Hard Exudates | 1.00 | 1.00 | 1.00 | 29 | 1.00 | 1.00 | 1.00 | 29 |
| Normal | 0.97 | 1.00 | 0.98 | 31 | 0.97 | 1.00 | 0.98 | 31 |
| Preretinal Hemorrhage | 0.94 | 0.94 | 0.94 | 16 | 0.94 | 1.00 | 0.97 | 16 |
| Silicon Oil in Eye | 1.00 | 0.95 | 0.97 | 20 | 1.00 | 0.95 | 0.97 | 20 |
| Vessel Tortuosity | 0.96 | 1.00 | 0.98 | 27 | 1.00 | 0.96 | 0.98 | 27 |
| accuracy | | | 0.98 | 282 | | | 0.99 | 282 |
| macro avg | 0.98 | 0.98 | 0.98 | 282 | 0.99 | 0.98 | 0.98 | 282 |
| weighted avg | 0.98 | 0.99 | 0.99 | 282 | 0.99 | 0.99 | 0.99 | 282 |

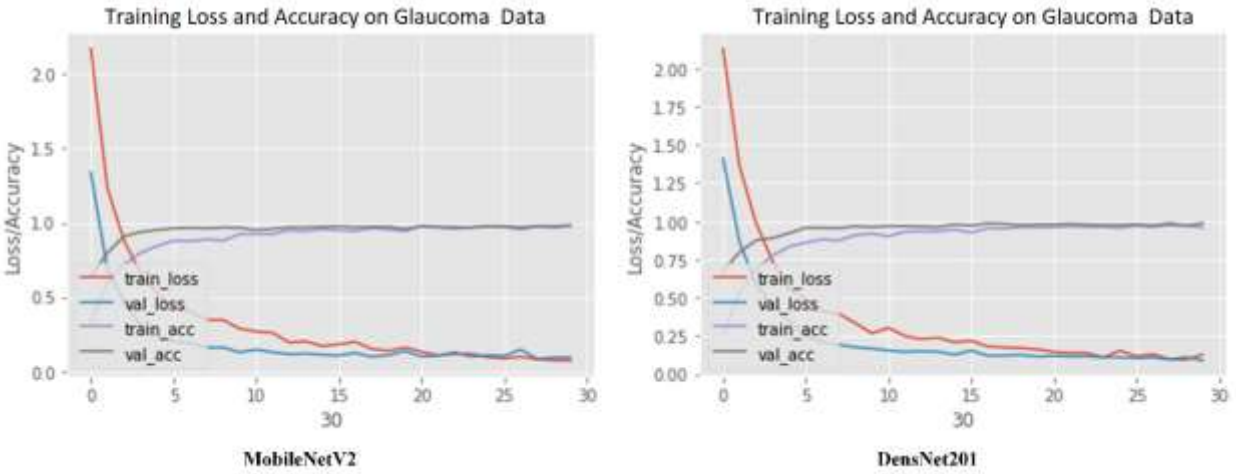
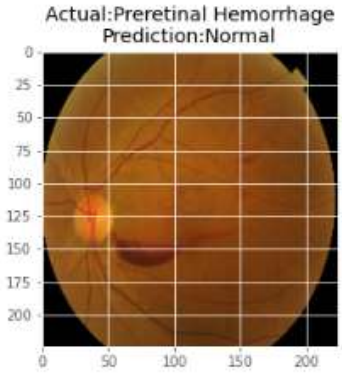
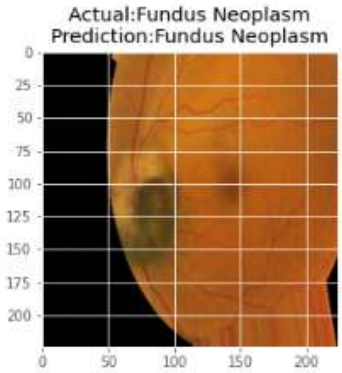



Figure 6 Precision-Recall curve of classification from all glaucoma diseases

```
plt.imshow(testX[8])
plt.title('Actual:'+ list(map(myfunc, y_test))[8] + '\n' + 'Prediction:'+ list(map(myfunc, y_pred))[8])
plt.show()
```




```
[24] plt.imshow(testX[35])
plt.title('Actual:'+ list(map(myfunc, y_test))[35] + '\n' + 'Prediction:'+ list(map(myfunc, y_pred))[35])
plt.show()
```




As can be observed from the aforementioned code and output, a small number of errors were made when predicting the model, and a small number of the dataset's images were incorrectly forecasted. Additionally, the actual forecast and anticipated value are provided together with the slight discrepancy that can't be predicted correctly in this model. Therefore, by giving the model more precise and thoroughly cleaned images and information, which will eventually assist the model to train and perform better, this model can demonstrate further development.

# 4 Conclusion

The retinal nerve fiber layer (RNFL) thickness seen in normal people necessitates routine monitoring for accurate glaucoma diagnosis (Shabbir et al., 2021). That's why it's recommended to check up with a doctor every six months. However, it is difficult to conduct such comprehensive medical examinations in countries with less access to advanced imaging technology like OCT. However, qualitative evaluations based on fundus photos have a low sensitivity for diagnosing early-stage progressive optic neuropathy because of the substantial training necessary. While our method has the potential to greatly increase the usefulness of qualitative and subjective data derived from fundus pictures alone, it still requires refinement before it can be used in clinical settings. Additional data for the training model is something we think can help RNFL thickness prediction become even more precise. However, the quality of the fundus image plays a crucial role in determining the discrepancy between OCT measurement and CNN-based estimate, even if a trained CNN can predict quantitative retinal deformation correctly.

As per my view this report helped me in understanding a CNN deep learning method which can be applied to various hospitals and ophthalmologists as it is cost-efficient, time-saving and easily accessible to small clinics and hospitals. In contrast to other methods, the suggested method primarily focuses on precisely extracting the optic cup and disc from the fundus image. It enables medical professionals to diagnose glaucoma without the need for extra examinations which not only saves time and cost to the doctor but for the patient as well.

To sum up, a good fundus shot is essential for a precise prediction. More risk factors, like intraocular pressure, a history of diabetes, hypertension, and extreme myopia, can be incorporated into the DL model to increase screening efficacy.

# References


AAMIR, M., IRFAN, M., ALI, T., ALI, G., SHAF, A., AL-BESHRI, A., ALASBALI, T. & MAHNASHI, M. H. J. D. 2020. An adoptive threshold-based multi-level deep convolutional neural network for glaucoma eye disease detection and classification. 10**,** 602.

ABDULLAH, F., IMTIAZ, R., MADNI, H. A., KHAN, H. A., KHAN, T. M., KHAN, M. A. & NAQVI, S. S. J. I. A. 2021. A review on glaucoma disease detection using computerized techniques. 9**,** 37311-37333.

AN, G., AKIBA, M., OMODAKA, K., NAKAZAWA, T. & YOKOTA, H. J. S. R. 2021. Hierarchical deep learning models using transfer learning for disease detection and classification based on small number of medical images. 11**,** 1-9.

BECHAR, M. E. A., SETTOUTI, N., BARRA, V., CHIKH, M. A. J. M. S. & PROCESSING, S. 2018. Semi-supervised superpixel classification for medical images segmentation: application to detection of glaucoma disease. 29**,** 979-998.

BISNETO, T. R. V., DE CARVALHO FILHO, A. O. & MAGALHÃES, D. M. V. J. A. S. C. 2020. Generative adversarial network and texture features applied to automatic glaucoma detection. 90**,** 106165.

GHEISARI, S., SHARIFLOU, S., PHU, J., KENNEDY, P. J., AGAR, A., KALLONIATIS, M. & GOLZAN, S. M. J. S. R. 2021. A combined convolutional and recurrent neural network for enhanced glaucoma detection. 11**,** 1-11.

GOUR, N. & KHANNA, P. J. P. R. L. 2020. Automated glaucoma detection using GIST and pyramid histogram of oriented gradients (PHOG) descriptors. 137**,** 3-11.

GUPTA, N., GARG, H. & AGARWAL, R. J. T. V. C. 2022. A robust framework for glaucoma detection using CLAHE and EfficientNet. 38**,** 2315-2328.

KAUSU, T., GOPI, V. P., WAHID, K. A., DOMA, W., NIWAS, S. I. J. B. & ENGINEERING, B. 2018. Combination of clinical and multiresolution features for glaucoma detection and its classification using fundus images. 38**,** 329-341.

KRISHNA ADITHYA, V., WILLIAMS, B. M., CZANNER, S., KAVITHA, S., FRIEDMAN, D. S., WILLOUGHBY, C. E., VENKATESH, R. & CZANNER, G. J. J. O. I. 2021. EffUnet-SpaGen: an efficient and spatial generative approach to glaucoma detection. 7**,** 92.

LI, L., XU, M., WANG, X., JIANG, L. & LIU, H. Attention based glaucoma detection: a large-scale database and CNN model. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019. 10571-10580.

MAYRO, E. L., WANG, M., ELZE, T. & PASQUALE, L. R. J. E. 2020. The impact of artificial intelligence in the diagnosis and management of glaucoma. 34**,** 1-11.

RAJA, H., AKRAM, M. U., HASSAN, T., RAMZAN, A., AZIZ, A. & RAJA, H. J. I. J. O. R. 2022. Glaucoma Detection Using Optical Coherence Tomography Images: A Systematic Review of Clinical and Automated Studies. 1-21.

SAXENA, A., VYAS, A., PARASHAR, L. & SINGH, U. A glaucoma detection using convolutional neural network. 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020. IEEE, 815-820.

SERTE, S. & SERENER, A. A generalized deep learning model for glaucoma detection. 2019 3rd International symposium on multidisciplinary studies and innovative technologies (ISMSIT), 2019. IEEE, 1-5.

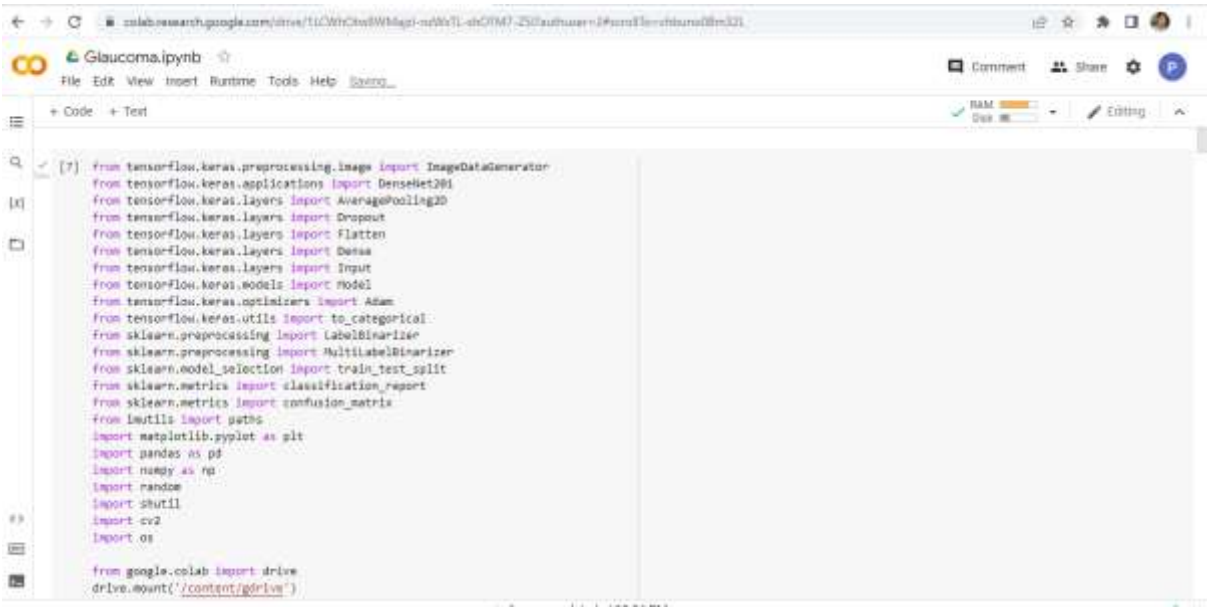
SHABBIR, A., RASHEED, A., SHEHRAZ, H., SALEEM, A., ZAFAR, B., SAJID, M., ALI, N., DAR, S. H., SHEHRYAR, T. J. M. B. & ENGINEERING 2021. Detection of glaucoma using retinal fundus images: A comprehensive review. 18**,** 2033-2076.

SHOUKAT, A., AKBAR, S. J. A. I. & THINGS, I. O. 2021. Artificial intelligence techniques for glaucoma detection through retinal images: State of the art. 209-240.

THAKOOR, K. A., LI, X., TSAMIS, E., SAJDA, P. & HOOD, D. C. Enhancing the accuracy of glaucoma detection from OCT probability maps using convolutional neural networks. 2019 41st annual international conference of the IEEE engineering in medicine and biology society (EMBC), 2019. IEEE, 2036-2040.

VAGHJIANI, D., SAHA, S., CONNAN, Y., FROST, S. & KANAGASINGAM, Y. Visualizing and understanding inherent image features in CNN-based glaucoma detection. 2020 Digital Image Computing: Techniques and Applications (DICTA), 2020. IEEE, 1-3.

# Appendix I


Glaucoma.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
[7]
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import DenseNet201
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import random
import shutil
import cv2
import os
from google.colab import drive
drive.mount('/content/gdrive')


+ Code + Text
[7]
from google.colab import drive
drive.mount('/content/gdrive')
file_path = '/content/gdrive/MyDrive/Machine Learning/Project Glaucoma/'
dataset_path = os.path.join(os.path.dirname(file_path), 'Glaucoma_Dataset')
INIT_LR=1e-3
EPOCHS = 10
BS = 10
# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")
#imagePaths = list(paths.list_images(dataset_path))
imagePaths = list(paths.list_images(f"{dataset_path}"))
data = []
labels = []
# loop over the image paths
for imagePath in imagePaths:
# extract the class label from the filename
label = imagePath.split(os.path.sep)[-2]
# load the image, swap color channels, and resize it to be a fixed
# 224x224 pixels while ignoring aspect ratio
image = cv2.imread(imagePath)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image = cv2.resize(image, (224, 224))
# update the data and labels lists, respectively
data.append(image)

```
+ Code + Text
[7] print(len(data))
np.save(file_path + 'Glaucoma_Dataset/Glaucoma_features_data.npy',data)
np.save(file_path + 'Glaucoma_features_labels.npy',labels)

data=np.load(file_path + 'Glaucoma_Dataset/Glaucoma_features_data.npy')
labels=np.load(file_path + 'Glaucoma_features_labels.npy')
print(labels)
lb = LabelBinarizer()
labelsx = lb.fit_transform(labels)
#labelsx = labelsx[:1]
print(lb.classes_)
(trainX, testX, trainY, testY) = train_test_split(data, labelsx, test_size=0.30, random_state=42)
trainAug = ImageDataGenerator(rotation_range=15, fill_mode="nearest")

baseModel = DenseNet201(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(64, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(10, activation="softmax")(headModel)
# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)
# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False
```

+ Code + Text

53m

```
opt=Adam()
model.compile(loss = "categorical_crossentropy", optimizer = opt, metrics=["accuracy"])
print("[INFO] training head...")
H = model.fit(
    trainAug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX)/BS,
    validation_data=(testX, testY),
    validation_steps=len(testX),
    epochs=EPOCHS )
# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy on Glaucoma Data")
plt.xlabel(N)
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

+ Code + Text

[7]
```
from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix

def myfunc(a):
  return lb.classes_[a]
y_predz = model.predict(testX, batch_size=BS, verbose=1)
y_pred = np.argmax(y_predz, axis=1)
predicted_Labels = map(myfunc, y_pred)
print(list(lb.inverse_transform(testY))[:])
#print(np.fromiter(predicted_Labels))
print(classification_report(list(lb.inverse_transform(testY))[:],list(predicted_Labels)[:]))
```

+ Code + Text

```
from sklearn.metrics import confusion_matrix
y_test = np.argmax(testY, axis=1)

conf_mtrx = confusion_matrix(y_test, y_pred)
```

[9]
```
# Functions to compute True Positives, True Negatives, False Positives and False Negatives

def true_positive(y_true, y_pred):

    tp = 0

    for yt, yp in zip(y_true, y_pred):

        if yt == 1 and yp == 1:
            tp += 1
    return tp

def true_negative(y_true, y_pred):
    tn = 0
    for yt, yp in zip(y_true, y_pred):

        if yt == 0 and yp == 0:
            tn += 1
    return tn

def false_positive(y_true, y_pred):
    fp = 0
    for yt, yp in zip(y_true, y_pred):

        if yt == 0 and yp == 1:
            fp += 1
    return fp

def false_negative(y_true, y_pred):
    fn = 0
    for yt, yp in zip(y_true, y_pred):
        if yt == 1 and yp == 0:
            fn += 1
    return fn
```

```
Dict={}
for i in range(30):
  if list(map(myfunc, y_test))[i] not in Dict:
    Dict[list(map(myfunc, y_test))[i]] = list(map(myfunc, y_test))[i]
    plt.imshow(testX[i])
    plt.title(list(map(myfunc, y_test))[i])
    plt.show()
```