

EYE-DISEASE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS (CNN) AND TENSORFLOW

A PROJECT REPORT

In partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING (SPECIALIZATION IN DATA SCIENCE)**

Under the guidance of

MAHENDRA DUTTA

By

SIDDHARTHA SHAW



Mckv Institute of Engineering

In association with



(ISO 9001:2015)

DECLARATION

We hereby declare that the project work presented in the proposal entitled **“EYE-DISEASE CLASSIFICATION”** using Convolutional Neural Networks (CNN) and TensorFlow is submitted in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (SPECIALIZATION IN DATA SCIENCE)** at **MCKV Institute of Engineering, Liluah, Howrah, West Bengal**. This is an original and authentic work carried out under the guidance of **Mr. Mahendra Dutta**. The project leverages deep learning to enhance diagnostic efficiency, providing a potential solution for early detection and improved patient outcomes.

Date: 27.08.24

Name of the Student: SIDDHARTHA SHAW



Ardent Computech Pvt. Ltd (An 9001:2015 Certified)

CERTIFICATE

This is to certify that the proposal for the minor project entitled **“Eye-Disease Classification using Convolutional Neural Networks (CNN) and TensorFlow”** is a record of genuine work carried out by **Siddhartha** under my guidance at **Ardent Computech Pvt. Ltd.** The project addresses critical health issues, focusing on the accurate detection of eye diseases through deep learning techniques. In my

opinion, the report, in its present form, is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (SPECIALIZATION IN DATA SCIENCE)** and adheres to the regulations of **Ardent Computech Pvt. Ltd.** To the best of my knowledge, the results presented in this report are original and make a meaningful contribution to healthcare technology.

Guide / Supervisor

MR. MAHENDRA DUTTA

Project Engineer Ardent Computech Pvt. Ltd (An ISO 9001:2015
Certified)

ACKNOWLEDGEMENT

The success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project work. I would like to show our greatest appreciation to **MAHENDRA DUTTA**, Project Engineer at

Ardent Computech Pvt. Ltd. I always feel motivated and encouraged every time by his valuable advice and constant inspiration. Words are inadequate in offering our thanks to the other trainees, project assistants and other members at **Ardent Computech Pvt. Ltd.** for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

Abstract:

Eye diseases are a major global health concern, especially in developing countries with limited resources. Early detection and treatment are essential to prevent vision loss. This study addresses this challenge by employing Convolutional Neural Networks (CNN) and transfer learning to classify eye diseases, including diabetic retinopathy, cataract, and glaucoma. CNNs, known for their efficacy in pattern recognition and medical diagnostics, are used to create a novel classification algorithm. The model, validated using K-Fold Cross Validation, achieves a high accuracy of 94% with transfer learning, compared to 84% with traditional CNN methods. Performance metrics such as accuracy, recall, specificity, precision, F1 score, and ROC curves further validate the model's effectiveness. Notably, the classifier demonstrates an impressive 99.89% accuracy in detecting glaucoma and diabetic retinopathy, highlighting its potential to improve early diagnosis and treatment in eye care.

Eye Diseases Classification

CHAPTERS

PAGE NO

Chapter 1: Introduction

6 – 7

Chapter 2: Objective

8

Chapter 3: Related Work

9

Chapter 4: Literature Survey

10 - 11

Chapter 5: Methodology

12 - 21

- Proposed Model
- Data Collection
- Data Preprocessing
- Data Modulation

Chapter 6: Data Visualization and Analysis

22 - 25

Chapter 7: Evaluation and Analysis

25-31

Chapter 8 Result Analysis

32

Chapter 9: Future Scope and Limitation

33

Chapter 10: Conclusion

34

Chapter 11: References

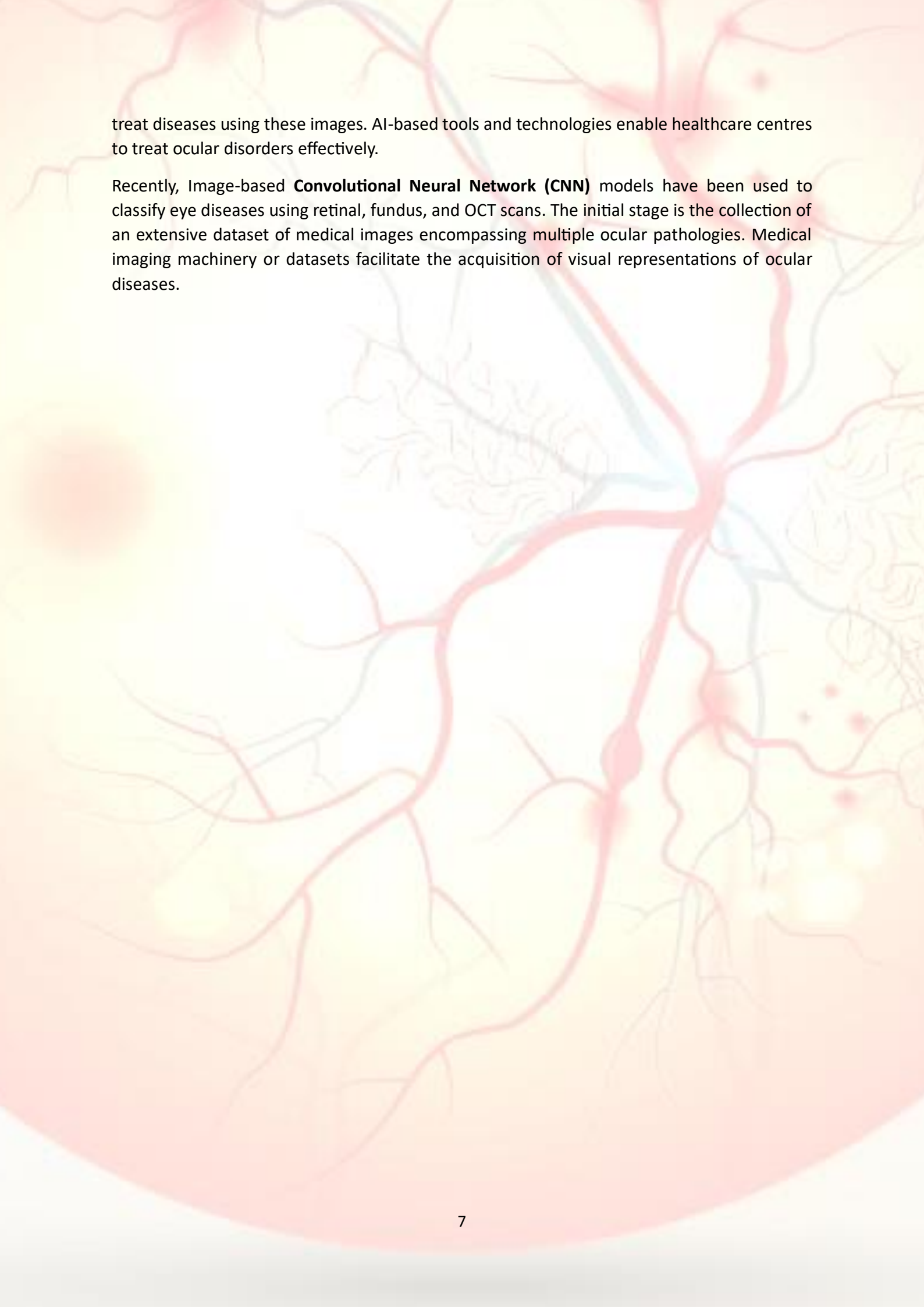
35

Chapter 1:

Introduction

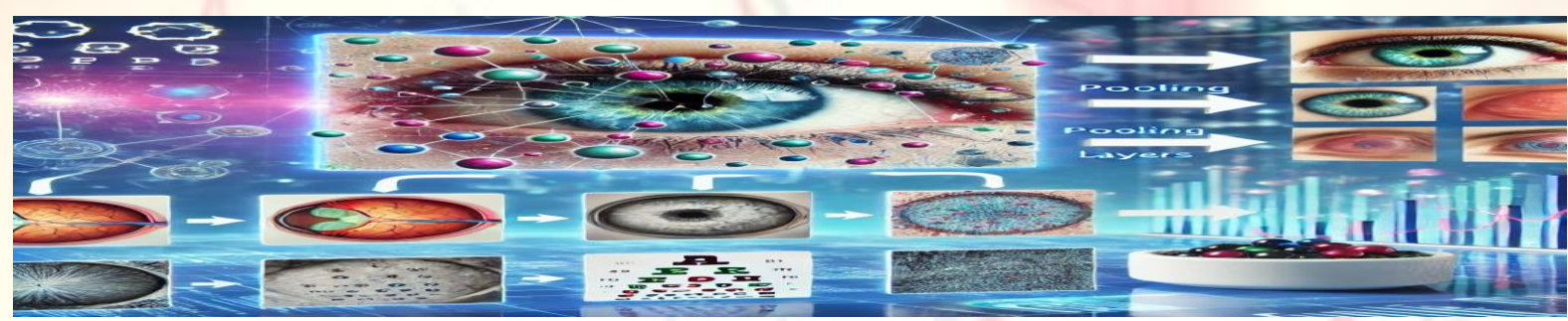
According to the first report of the **World Health Organization** in **2019**, it is estimated that there are one thousand two hundred million persons with an eye condition. Some of the conditions are refraction error (one hundred and twenty-three million), eye cataracts (sixty-five million), Glaucoma (seven million), corneal opacities (four million), macular degeneration (ten million), diabetic retinopathy (three million), and presbyopia (eight hundred and twenty-six million). Patients with vision problems suffer from significantly reduced quality of life. Diabetes retinopathy and glaucoma are examples of eye diseases. Globally, 64.3 million people had glaucoma in 2013. However, it is restricted due to the lack of awareness, shortage of ophthalmologists, and high consultation costs., Hence, automated screening is critical. This paper will utilize machine learning techniques to distinguish a normal eye from one with a disease. Diabetic eye disease (DED) categorization relies heavily on image processing; hence offered a comprehensive study on the topic. Picture quality enrichment, image segmentation, image augmentation (geometric transformation), and classification comprise DED's suggested automated classification framework. The best results were achieved by combining conventional approaches to image processing with a freshly developed convolution neural network (CNN) structure. When applied to DED classification problems, the newly constructed CNN combined with the conventional image processing method showed the best accuracy performance. The experimental data indicated satisfactory levels of precision, sensitivity, and specificity.

Medical imaging encompasses diverse technologies to provide visual representations of the internal structures of the human body. It is essential for non-invasive assessments of anatomical structures and physiological processes. Healthcare experts diagnose, monitor, and



treat diseases using these images. AI-based tools and technologies enable healthcare centres to treat ocular disorders effectively.

Recently, Image-based **Convolutional Neural Network (CNN)** models have been used to classify eye diseases using retinal, fundus, and OCT scans. The initial stage is the collection of an extensive dataset of medical images encompassing multiple ocular pathologies. Medical imaging machinery or datasets facilitate the acquisition of visual representations of ocular diseases.



Chapter 2:

Objective

1. Accurately classify eye diseases using CNN and TensorFlow.
2. Support early diagnosis and timely treatment of eye diseases.
3. Automate and enhance the accuracy of disease detection.
4. Leverage deep learning techniques for complex image analysis.
5. Improve accessibility to eye care through automated solutions.

Chapter 3:

Related Work

Several studies have focused on creating expert systems to automate disease diagnosis, aiming for quick and accurate results (Hasan et al., **2021**). In the field of medical imaging, Convolutional Neural Networks (CNNs) have shown great promise. The complexity of retinal images has led researchers to develop machine learning systems to handle these challenges effectively.

Early research on Optical Coherence Tomography (OCT) was introduced by Ginsburg (**2006**), who used OCT imaging to detect intraocular lenses and refractive surgery. They applied machine learning techniques to identify specific eye diseases. Since then, various machine learning solutions have been developed to detect eye diseases such as age-related macular degeneration, diabetic retinopathy, and to automatically localize the optical disc using image classification with support vector machines (Farooq and Sattar, **2015**).

In Al-Mohtaseb et al. (**2021**), a study was done to explore the connection between symptoms and diagnoses of dry eye disease. The researchers used Independent Component Analysis (ICA) and Pearson correlations. They found that the ICA components indicated that there was little remaining data, so no consistent relationship was identified among the most commonly used symptoms and signs.

In An et al. (**2019**), a machine learning technique for detecting glaucoma was introduced. The study used OCT and color fundus images to identify abnormal features in the retina. They applied a segmentation algorithm to create thickness and deviation maps. Convolutional Neural Networks (CNNs) with transfer learning were used on various input images, including gray-scale optic disc images and retinal layers. The CNNs were trained with data augmentation, and the results were combined using a Random Forest model. This approach achieved high accuracy in detecting glaucoma based on image features.

Chapter 4:

Literature Survey

Various eye conditions, such as trachoma, cataracts, and corneal ulcers, can significantly impair vision, making early diagnosis crucial to prevent progression. These illnesses often present a range of visually discernible symptoms that require comprehensive analysis for accurate diagnosis. To address this, researchers have proposed automated eye disease recognition systems using machine learning techniques like Deep Convolutional Neural Networks (DCNN) and Support Vector Machines (SVM). Studies have shown that DCNN models outperform SVM in recognizing these diseases.

For example, one study developed a deep neural network model to differentiate between eye diseases like diabetic retinopathy and glaucoma, achieving 80% accuracy. Another study introduced a CNN-based system, LCD Net, which successfully classified retinal fundus images as either healthy or diseased. Additionally, research has highlighted the effectiveness of Transfer Learning (TL) and combined Deep Learning (DL) and Machine Learning (ML) approaches in automating diabetic eye disease identification. Overall, CNN has emerged as a leading tool for detecting eye diseases and diagnosing other pathological signs using retinal images, proving to be more effective than traditional models.

To expedite diagnosis, healthcare systems are increasingly relying on artificial intelligence (AI) solutions. However, for machine learning to be accurate and reliable, health data must be recorded consistently. A standardized framework for recording diagnostic data is essential to help machine learning algorithms accurately predict diseases based on symptoms. One study proposed a solution that includes a user-friendly interface to ensure error-free data entry and utilized various machine learning techniques, such as Decision Trees, Random Forests, Naive Bayes, and Neural Networks, to analyze patient data. The Random Forest and Decision Tree algorithms achieved a prediction accuracy of over 90%.

Another study focused on classifying different eye diseases using patient data from Mecca hospital in Sudan, employing Naive Bayes, SVM, and the J48 Decision Tree. The J48 model achieved a high accuracy of 98.75%. Additionally, a review highlighted the use of image processing and machine learning techniques, like SVM and PCA, for detecting and classifying eye diseases. The review emphasized the potential of AI and deep learning (DL) technologies in screening for diabetic eye disease, suggesting that AI will become essential in improving the effectiveness and accessibility of screening programs, ultimately helping to prevent vision loss.

S NO.	AUTHOR	JOURNAL	TITLE OF THE PAPER	KEYWORDS AND PUBLISHED YEAR	OUTCOME
1.	Ushma K Sattigeri, Harshith N, Dhanush Gowda N, K A Ullas, Aditya M S	IRJET	Eye Disease Identification Using Deep Learning	Deep Learning, Convolutional Neural Network, Deep Neural Network and published in July 2022.	The paper contributes to the field of medical image analysis and deep learning by demonstrating the potential of these techniques for identifying eye diseases.
2.	Dragos D. Taralunga, Bogdan C. Florea, Anamaria Radoi	APPLIED SCIENCES	Artificial Intelligence-Driven Eye Disease Classification Model	artificial intelligence; Shuffle Net V2; single shot detection; machine learning and published in 2023.	The paper contributes to the field of medical image analysis and AI by demonstrating the potential of these techniques for developing accurate and efficient eye disease classification models.
3.	Mohamed Elkholy, Marwa A. Marzouk	ORIGINAL RESEARCH ARTICLE(FRONTIERS)	Deep learning-based classification of eye diseases using Convolutional Neural Network	CNN, OpenCV, and published in January 2024.	The paper contributes to the field of medical image analysis and deep learning by demonstrating the potential of these techniques for identifying eye diseases.
4.	Tareq Babaqi, Manar Jaradat, Ayse Erdem Yildirim, Saif H. Al-Nimer, and Daehan Won	REARSCHGATE	Eye Disease Classification Using Deep Learning Technique	Eye Disease, Image classification, Data Mining, CNN, Transfer learning and published in July 2023.	The paper contributes to the field of medical image analysis and deep learning by demonstrating the potential of these techniques for identifying eye diseases.
5.	OMAR BERNABÉ , ELENA ACEVEDO , ANTONIO ACEVEDO , RICARDO CARREÑO , AND SANDRA GÓMEZ	IEEE ACCESS	Classification of Eye Diseases in Fundus Images	Artificial intelligence, convolutional neural networks, machine learning, and supervised learning and published in 2021.	The paper contributes to the field of medical image analysis and AI by demonstrating the potential of these techniques for developing accurate and efficient eye disease classification models.

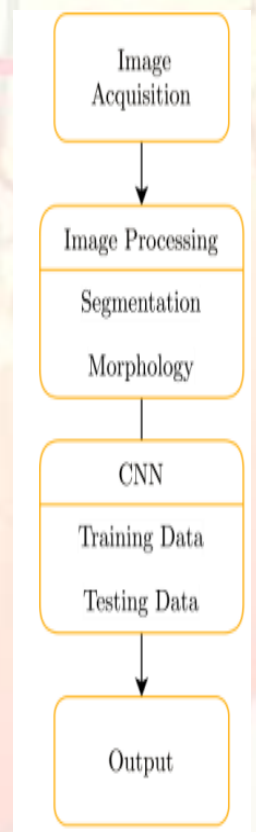
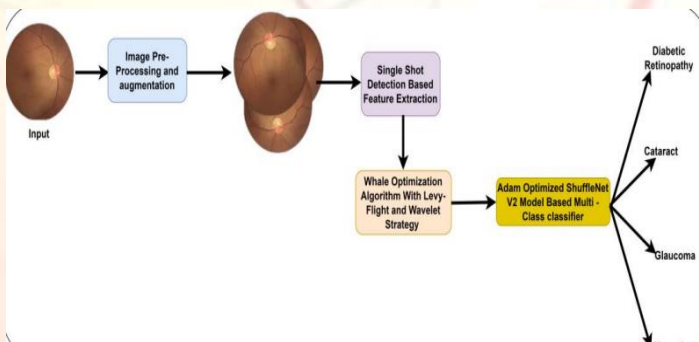


Chapter 5:

Methodology

3.1 Proposed Model:

In this section, I present the proposed model for Eye-disease classification using CNN and TensorFlow and I also use some models and diagrams, that describe the process of how classification is done.



3.2 Data Collection:

In this first phase, we collect the “eye-disease-classification” dataset from **Kaggle** and cleaning the data.

Step-1: Import the libraries

```
from keras.layers import Dense, Input, Conv2D, LSTM, MaxPool2D, UpSampling2D
from sklearn.model_selection import train_test_split

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import tensorflow as tf
from tensorflow import keras
from pathlib import Path
import PIL
import os
```

Step-2: Set the path and go to further steps

```
folder_path = "/content/drive/My Drive/project/dataset"
```

```
from pathlib import Path
import pandas as pd
from tqdm import tqdm

folder_path = "/content/drive/My Drive/project/dataset"
glaucoma = Path(folder_path + '/glaucoma')
cataract = Path(folder_path + '/cataract')
normal = Path(folder_path + '/normal')
diabetic_retinopathy = Path(folder_path + '/diabetic_retinopathy')

disease_type = [glaucoma, cataract, normal, diabetic_retinopathy]
df = pd.DataFrame()

for types in disease_type:
    if types.exists(): # Check if the directory exists
        for imagepath in tqdm(list(types.iterdir()), desc=str(types)):
            df = pd.concat([df, pd.DataFrame({
                'image': [str(imagepath)],
                'disease_type': [disease_type.index(types)]
            })], ignore_index=True)
```

```
df.head()
```

	image	disease_type
0	/content/drive/My Drive/project/dataset/glauco...	0
1	/content/drive/My Drive/project/dataset/glauco...	0
2	/content/drive/My Drive/project/dataset/glauco...	0
3	/content/drive/My Drive/project/dataset/glauco...	0
4	/content/drive/My Drive/project/dataset/glauco...	0

Step 3: inspect of the data

```
df.describe()

   disease_type
count  4218.000000
mean    1.536747
std     1.116713
min     0.000000
25%    1.000000
50%    2.000000
75%    3.000000
max     3.000000

df.shape
(4218, 2)

df.columns
Index(['image', 'disease_type'], dtype='object')

df.count() # number of rows

```

	0
image	4218
disease_type	4218

```
dtype: int64

df.disease_type.value_counts()

```

disease_type	count
3	1098
2	1075
1	1038
0	1007

```
dtype: int64
```

3.3 Data Preprocessing:

Step 1: Check the null value and Duplicate values

```
# check if null values present are not.
```

```
df.isnull().sum()
```

	0
image	0
disease_type	0

```
dtype: int64
```

```
## check for duplicates
df.duplicated().sum()
```

```
0
```

Step 2: Count the diseases types-values

```
df['disease_type']=df['disease_type'].map({0: 'glaucoma', 1: 'cataract' , 2: 'normal', 3:
df.disease_type.value_counts()
```

	count
disease_type	
diabetic_retinopathy	1098
normal	1075
cataract	1038
glaucoma	1007

Step 3: In this part, we first normalize the data and then use feature selection to split our dataset into features (X) and labels (y).

```
from sklearn.feature_selection import RFE, RFECV
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

```
#train-test data
```

```
from sklearn.model_selection import train_test_split
```

```
# Split your data into features (X) and labels (y)
```

```
x = df['image'] # 'image' contains image data
```

```
y = df['disease_type'] # 'disease_type' contains labels
```

```
# Split the data into training and testing sets
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42
```

```
x
image
0 /content/drive/My Drive/project/dataset/glauco...
1 /content/drive/My Drive/project/dataset/glauco...
2 /content/drive/My Drive/project/dataset/glauco...
3 /content/drive/My Drive/project/dataset/glauco...
4 /content/drive/My Drive/project/dataset/glauco...
...
4213 /content/drive/My Drive/project/dataset/diabet...
4214 /content/drive/My Drive/project/dataset/diabet...
4215 /content/drive/My Drive/project/dataset/diabet...
4216 /content/drive/My Drive/project/dataset/diabet...
4217 /content/drive/My Drive/project/dataset/diabet...
```

4218 rows × 1 columns

dtype: object

```
y
disease_type
0 glaucoma
1 glaucoma
2 glaucoma
3 glaucoma
4 glaucoma
...
4213 diabetic_retinopathy
4214 diabetic_retinopathy
4215 diabetic_retinopathy
4216 diabetic_retinopathy
4217 diabetic_retinopathy
```

4218 rows × 1 columns

dtype: object

Step 4: Data generation refers to the process of creating synthetic or augmented data to increase the size and diversity of a dataset. This is especially useful when the available data is limited or imbalanced. In the context of image classification, it is used.

For this part, we divided it into a **train-data** part and a **valid-data** part

```
#data generators for tarining
```

```
# Convert the 'disease_type' column to string
df['disease_type'] = df['disease_type'].astype(str)

train_data = datagen.flow_from_dataframe(dataframe=df,
                                         x_col='image',
                                         y_col='disease_type',
                                         target_size=(224, 224),
                                         class_mode='categorical',
                                         batch_size=32,
                                         shuffle=True,
                                         subset='training')
```

Found 3375 validated image filenames belonging to 4 classes.

```
# Data generators for validation
```

```
df['disease_type'] = df['disease_type'].astype(str)

valid_data = datagen.flow_from_dataframe(dataframe=df,
                                         x_col = 'image',
                                         y_col = 'disease_type',
                                         target_size=(224,224),
                                         class_mode = 'categorical',
                                         batch_size = 32,
                                         shuffle = True,
                                         subset = 'validation')
```

Found 843 validated image filenames belonging to 4 classes.

The code sets up data generators for training and validation using specific parameters:

1. **x_col** and **y_col**: Define the columns for image paths and labels in the DataFrame.
2. **target_size=(224, 224)**: Resizes images to 224x224 pixels.
3. **class_mode='categorical'**: Prepares data for multi-class classification.
4. **batch_size=32**: Processes 32 images per batch for efficient training.
5. **shuffle=True**: Randomizes data to prevent overfitting.
6. **subset='training' / 'validation'**: Separates the data into training and validation sets.

3.4 Data Modulation: In this section, we use some TensorFlow and Keras techniques to summarize and analyse the model architecture differently--

1st Approach:

VGG19. Standing for “Visual Geometry Group 19,” VGG19 is not a secret code but a powerful convolutional neural network (CNN) architecture renowned for its effectiveness in image classification and recognition.

VGG19 is a convolutional neural network (CNN) architecture that was developed by the Visual Geometry Group (VGG) at the University of Oxford. It is known for its simplicity and effectiveness in image classification tasks. The model was introduced in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" by Karen Simonyan and Andrew Zisserman.

Architecture:

VGG19 consists of **19 layers** in total, which includes **16 convolutional layers** and **3 fully connected layers**. Additionally, it incorporates **5 max-pooling layers** and a **SoftMax layer** for classification tasks.

Convolutional Layers: The architecture uses small convolutional filters (3x3) throughout the network, which allows it to capture fine details in images while maintaining a manageable number of parameters.

Pooling Layers: Max pooling layers are used to reduce the spatial dimensions of the feature maps, which helps in reducing the computational load and controlling overfitting.

Key Features:

Parameter Count: VGG19 has approximately **138 million parameters**, making it a relatively large model. This high number of parameters contributes to its ability to learn complex features from images.

Pre-trained Models: VGG19 is often used with pre-trained weights from the ImageNet dataset, which contains over a million images across 1000 categories. This allows the model to leverage learned features for various image classification tasks, making it suitable for transfer learning.

Performance: VGG19 has been shown to perform well on various benchmarks, achieving high accuracy in image classification tasks. Its architecture has influenced many subsequent models in the field of deep learning.

```
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, BatchNormalization, Dropout,
from keras.models import Sequential

# labeling this

labels= [key for key in train_data.class_indices ]
num_classes= len(disease_type)

# 1st approach

from tensorflow.keras.applications.vgg19 import VGG19
image_size=224
vgg = VGG19(weights="imagenet",include_top = False,input_shape=(image_size,image_size,3))
```

2nd Approach:

ResNet50 is a deep convolutional neural network architecture that is part of the Residual Network (ResNet) family, developed by Kaiming He and his colleagues. It was introduced in the paper "Deep Residual Learning for Image Recognition" and has become one of the most popular models for image classification tasks due to its innovative design and effectiveness.

Architecture:

Depth: ResNet50 consists of **50 layers**, which include convolutional layers, batch normalization layers, and fully connected layers. The depth of the network allows it to learn complex features from images.

Residual Connections: One of the key innovations of ResNet50 is the use of **residual connections** (or skip connections). These connections allow the input to bypass one or more layers and be added to

the output of a later layer. This helps to mitigate the vanishing gradient problem, making it easier to train very deep networks.

Key Features:

Feature Extraction: The convolutional layers in ResNet50 are responsible for extracting features from input images. Each convolutional layer is followed by batch normalization and ReLU activation functions, which help stabilize the learning process and introduce non-linearity. **Performance:** ResNet50 has achieved state-of-the-art results on various image classification benchmarks, including the ImageNet dataset. Its architecture allows it to generalize well to different tasks and datasets.

In summary, ResNet50 is a powerful and influential deep-learning model that has significantly impacted the field of computer vision, particularly in image classification tasks. Its innovative architecture and effectiveness make it a popular choice for a wide range of applications.

```
# 2nd approach
```

```
from tensorflow.keras.applications import ResNet50

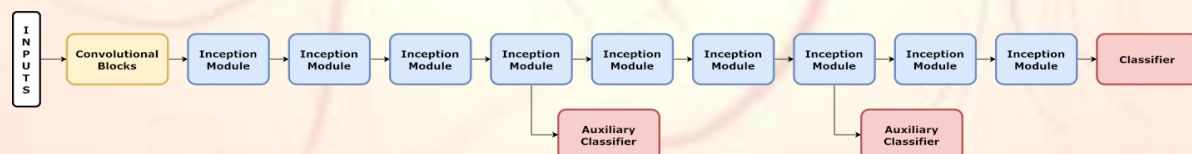
resnet = ResNet50(weights='imagenet', include_top=False, input_shape=(image_size, image_size, 3))

model = Sequential()
model.add(resnet)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736  000000000000000000 3s 0us/step
```

3rd Approach:

Inception v3 is a convolutional neural network (CNN) for assisting in image analysis and object detection and got its start as a module for Google Net. It is the third edition of Google's Inception Convolutional Neural Network, originally introduced during the ImageNet Recognition Challenge. The design of Inceptionv3 was intended to allow deeper networks while also keeping the number of parameters from growing too large: it has "under 25 million parameters", compared to 60 million for Alex Net.



Just as ImageNet can be thought of as a database of classified visual objects, Inception helps the classification of objects in the world of computer vision. The Inceptionv3 architecture has been reused in many different applications, often used "pre-trained" from ImageNet. One such use is in life sciences, where it aids in the research of leukaemia.

```
from tensorflow.keras.applications import InceptionV3

inception = InceptionV3(weights='imagenet', include_top=False, input_shape=(image_size,

model = Sequential()
model.add(inception)
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

4th Approach:

MobileNetV2 is a convolutional neural network architecture designed specifically for mobile and embedded vision applications. It was introduced by Google researchers in 2018 and built the original Mobile Net architecture, enhancing its efficiency and accuracy while maintaining a lightweight structure suitable for devices with limited computational resources.

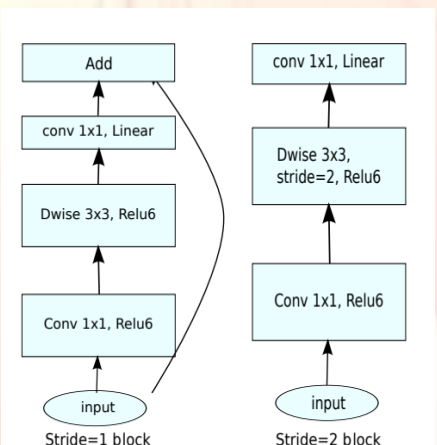
Architecture:

Depth-wise Separable Convolutions: MobileNetV2 utilizes **depth-wise separable convolutions**, which consist of two layers: a depthwise convolution and a pointwise convolution. This approach significantly reduces the number of parameters and computations compared to standard convolutions. **Depthwise Convolution:** Applies a single filter to each input channel separately, capturing spatial features. **Pointwise Convolution:** Combines the outputs of the depthwise convolutions across channels using 1x1 convolutions, allowing for interaction between the features. **Inverted Residuals:** The architecture introduces **inverted residual blocks**, which consist of a lightweight depth-wise convolution followed by a linear bottleneck. This design helps maintain a low computational cost while allowing the network to learn complex features. The inverted residual structure allows for efficient feature extraction and preserves information through residual connections, which help in gradient flow during training. **Linear Bottlenecks** MobileNetV2 employs **linear bottlenecks** at the end of each block, where the output is passed through a linear layer instead of a non-linear activation function. This design choice helps reduce information loss and improve performance, especially in deeper networks.

```
from tensorflow.keras.applications import MobileNetV2

mobilenet = MobileNetV2(weights='imagenet', include_top=False, input_shape=(image_size,
model = Sequential()
model.add(mobilenet)
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9406464/9406464 ████████████████████ 1s 0us/step
```



(d) Mobilenet V2

5th Approach:

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(image_size, image_size, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

1. Conv2D:

Role: The Conv2D layer is responsible for extracting features from images by applying filters (kernels) that detect patterns such as edges, textures, and shapes. It is a foundational layer in CNNs, commonly used for tasks like image recognition.

2. MaxPooling2D:

Role: The MaxPooling2D layer reduces the size of feature maps by selecting the maximum value within a specified pooling window. This down-sampling helps to reduce the spatial dimensions of the data, making the model more efficient and less prone to overfitting.

3. Batch Normalization:

Role: The Batch Normalization layer normalizes the inputs of each layer during training, which improves stability and speeds up convergence. It helps to reduce the impact of internal covariate shift and allows for a higher learning rate.

In summary, we can say that **Conv2D** layers extract features from images by applying filters, enabling the model to learn important visual patterns. **MaxPooling2D** layers reduce the spatial dimensions of feature maps, enhancing computational efficiency and reducing overfitting. **Batch Normalization** layers normalize the inputs to improve training stability and allow for faster convergence.

3.5 Model summary:

A **model summary** is a detailed report that outlines the architecture of a neural network, including the types and names of layers, the output shapes of each layer, and the number of trainable and non-trainable parameters, providing an overview of the model's structure and complexity.

Chapter 6:

Data Visualization and Analysis

Data Visualization: The graphical representation of data using charts, graphs, and other visual tools to make complex information more understandable and accessible.

In Machine learning (ML) and deep learning, it enhances understanding of data patterns and model performance through graphical representations. It aids in exploring data, selecting relevant features, and interpreting results, making complex information accessible and actionable for informed decision-making and effective communication among stakeholders.

Data Analysis: Data analysis is the systematic process of inspecting, cleaning, transforming, and modeling data to extract useful information, draw conclusions, and support decision-making. It involves various techniques and methodologies to interpret data from diverse sources, ultimately helping organizations make informed decisions and understand complex phenomena.

For this part, we describe the data visualization techniques and how the analysis part be done in this model.

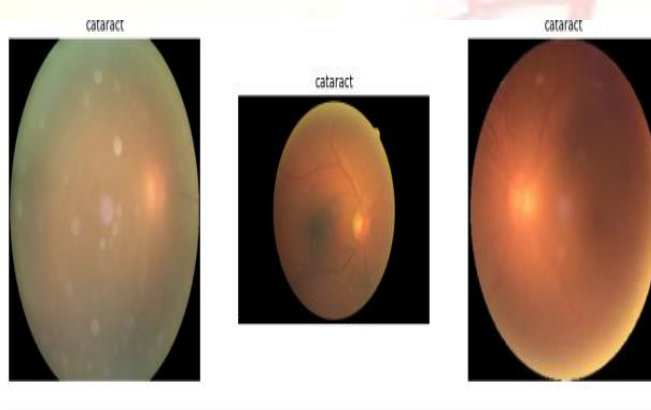


Fig 1: Cataract

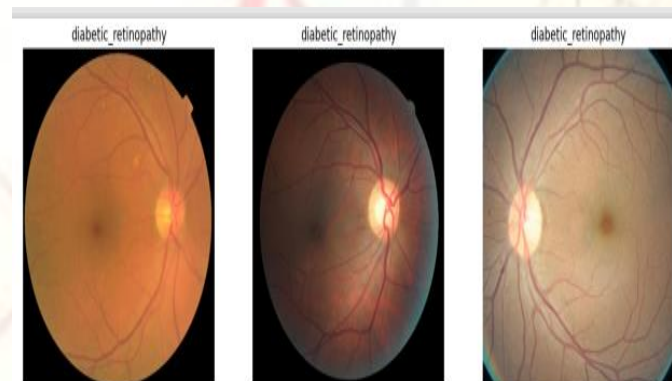


Fig 2: Diabetic retinopathy

For this, we import some libraries from **sklearn** and **Pathlib** and they are essential to do this visualization process for this model and they are **TensorFlow**, **Keras**, **Os**, and **PIL**.

TensorFlow and Keras are essential for building and training deep learning models, with TensorFlow providing a flexible framework and Keras simplifying the process with an intuitive interface. OS enables file manipulation and environment management, crucial for configuring machine learning setups. PIL (Pillow) adds image processing capabilities, allowing for opening, manipulating, and saving various image formats and preparing image data for machine learning models. These libraries work together seamlessly, creating a powerful toolkit for data scientists and machine learning practitioners to develop and visualize sophisticated models.

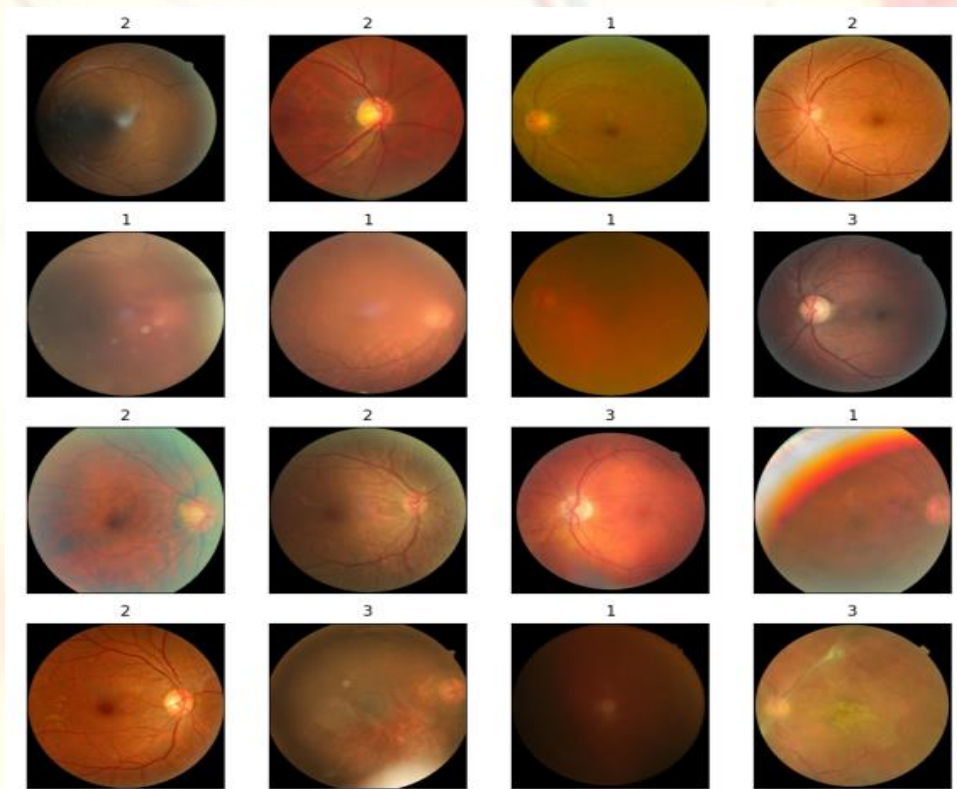


Fig 3: Collection of retinal fundus images

This appears to be a collection of retinal fundus images. The numbers next to each image likely represent different stages or severities of the observed conditions. Without further context, I cannot definitively identify the specific eye diseases or conditions depicted, but the images suggest they may include conditions like diabetic retinopathy, age-related macular degeneration, and other retinal vascular or pigmentary disorders.

These types of retinal images are commonly used in ophthalmology and optometry to diagnose, monitor, and manage various eye diseases and disorders. The collection of images appears to be part of a medical or research study related to retinal imaging and disease classification.

Data Augmentation:

Data augmentation is a technique used in deep learning to artificially increase the size and diversity of the training dataset by creating modified versions of existing data. This is particularly useful when dealing with limited data, as it helps prevent overfitting and improves the model's ability to generalize to new, unseen data.

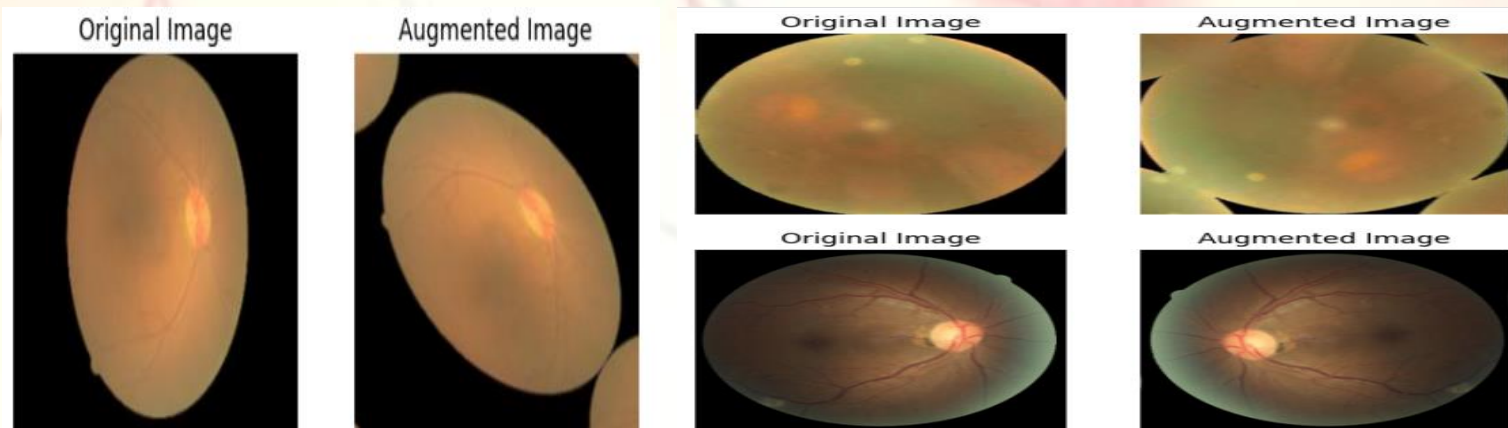


Fig 4: Original images vs Augmented images

In this section, we differentiate between original images and augmented images by import augmentation and we also import the library which is Cv2.

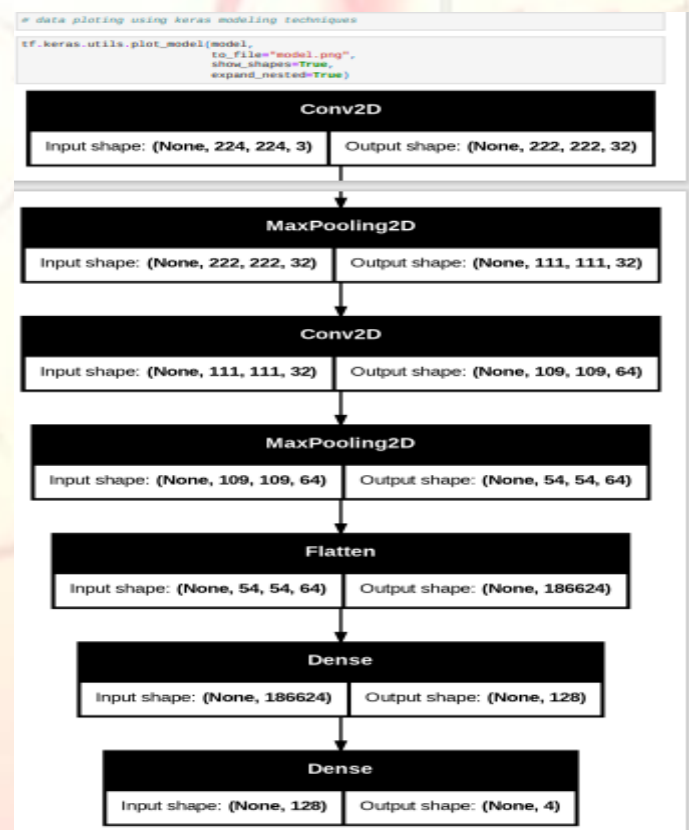
Original images in deep learning are the raw, unaltered data points used for training, while augmented images are artificially generated variations created through techniques like rotation, flipping, and color adjustments. Augmented images introduce diversity, helping the model learn to generalize better and reduce overfitting. By simulating different conditions, augmented data makes the model more robust against variations in real-world scenarios, ultimately improving its performance on a wide range of tasks, even with limited training data.

Data plotting using Keras modeling techniques:

In this, we plot this using the Keras modeling technique, and for this use **Conv2D**, **MaxPooling2D**, **Flatten**, and **Dense**.

This image depicts a neural network architecture for image classification, specifically a convolutional neural network (CNN). The network consists of a series of convolutional (Conv2D) and max-pooling (MaxPooling2D) layers, followed by a flattening layer, and finally, two dense layers.

The input shape and output shape of each layer are shown, indicating how the spatial dimensions of the feature maps are transformed as the data flows through the network. This architecture is designed to extract relevant features from the input images and classify them into desired categories.



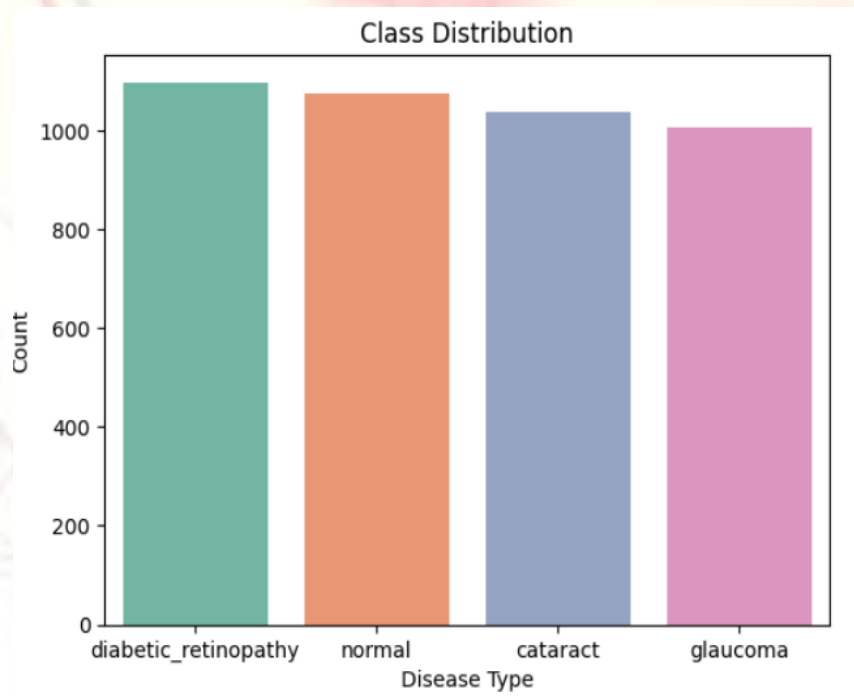


Fig 5: bar plot (Class Distribution vs Disease Type)

Chapter 7:

Evaluation and Analysis

Evaluation and **analysis** are crucial to assess model performance, robustness, and generalization. Key aspects include selecting appropriate metrics, employing validation techniques, analyzing generalization, testing robustness, and examining model explainability and interpretability. Rigorous evaluation enables informed decisions about model selection, optimization, and deployment, leading to reliable and trustworthy systems that meet the intended objectives.

Step 1: Model Accuracy check

Accuracy:

Model accuracy in deep learning refers to the ratio of correct predictions made by the model to the total number of predictions. It provides a direct measure of how well the model has learned the underlying patterns in the data and can generalize to new examples. Higher accuracy indicates better model performance.

For this, model's accuracy, we first calculate **the Checkpoint** which refers to a saved state of the model's parameters during training, allowing for model restoration, resumption of training, performance evaluation, and transfer learning, enabling efficient and reliable deep learning model development.

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# ModelCheckpoint with updated file extension
checkpoint = ModelCheckpoint("vgg19.keras", # Change .h5 to .keras
                             monitor="val_acc",
                             verbose=1,
                             save_best_only=True,
                             save_weights_only=False,
                             save_freq='epoch')

# EarlyStopping callback
earlystop = EarlyStopping(monitor="val_acc",
                           patience=5,
                           verbose=1)
```

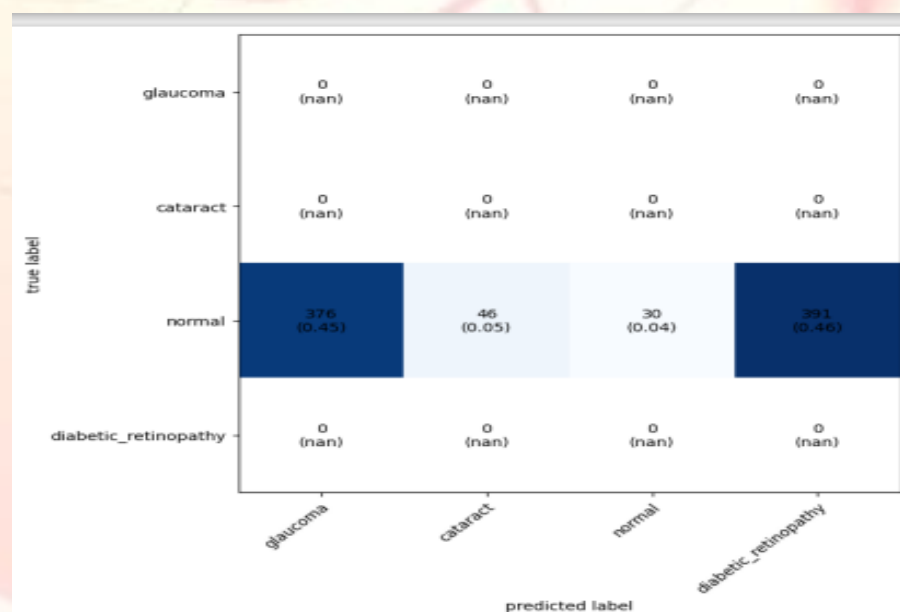
In this section, we use checkpoint and calculate it using model check and then we calculate another Early stopping callback. This callback is used to stop the training process early if the model's performance on the validation set does not improve.

```
loss, accuracy = model.evaluate(valid_data)
print("Loss:", loss)
print("Accuracy:", accuracy)
```

```
27/27 ████████████████████████████████ 70s 3s/step - accuracy: 0.0322 - loss: 2.4699
Loss: 2.48268461227417
Accuracy: 0.035587187856435776
```

Step 2: Model Evaluation

Confusion Metrix: A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model's predictions. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance.



This image appears to be a confusion matrix or a correlation matrix, which is a common way to visualize the performance of a classification model. In this case, the matrix shows the predicted labels (rows) versus the true labels (columns) for a multi-class classification problem involving different eye conditions: **glaucoma**, **cataract**, **normal**, and **diabetic retinopathy**.

The numbers within the cells represent the counts or probabilities of the model's predictions for each combination of true and predicted labels. For example, the cell in the top-left corner shows that the model correctly predicted 0.435 (or 43.5%) of the glaucoma cases, while the cell in the top-right corner shows that the model incorrectly predicted 0.443 (or 44.3%) of the glaucoma cases as "normal".

Classification Report:

A **classification report** is a detailed summary of the performance metrics for a classification model. It typically includes the following key metrics:

1. **Precision:** The ratio of true positive predictions to the total number of positive predictions (true positives + false positives). It indicates how accurate the model's positive predictions are.
2. **Recall (Sensitivity):** The ratio of true positive predictions to the total number of actual positive instances. It measures the model's ability to detect positive instances correctly.
3. **F1-score:** The harmonic mean of precision and recall, providing a balanced measure of the model's performance.
4. **Support:** The number of instances for each class in the test or evaluation dataset.

```
print(classification_report(y_test,y_pred,target_names = labels))
```

	precision	recall	f1-score	support
cataract	0.00	0.00	0.00	0
diabetic_retinopathy	0.00	0.00	0.00	0
glaucoma	1.00	0.04	0.07	843
normal	0.00	0.00	0.00	0
accuracy			0.04	843
macro avg	0.25	0.01	0.02	843
weighted avg	1.00	0.04	0.07	843

This classification report shows that the model performs perfectly on the "glaucoma" class but fails to classify "cataract," "diabetic retinopathy," and "normal" classes (with precision, recall, and F1-scores all at 0). The overall accuracy is low at 4%, highlighting a significant imbalance in model performance across the classes.

Different Types of Visualization: In this section, we plot many types of visualization techniques that are used to check how model accuracy and model loss can be justified, and also this visualization helps how it will be used in this model.

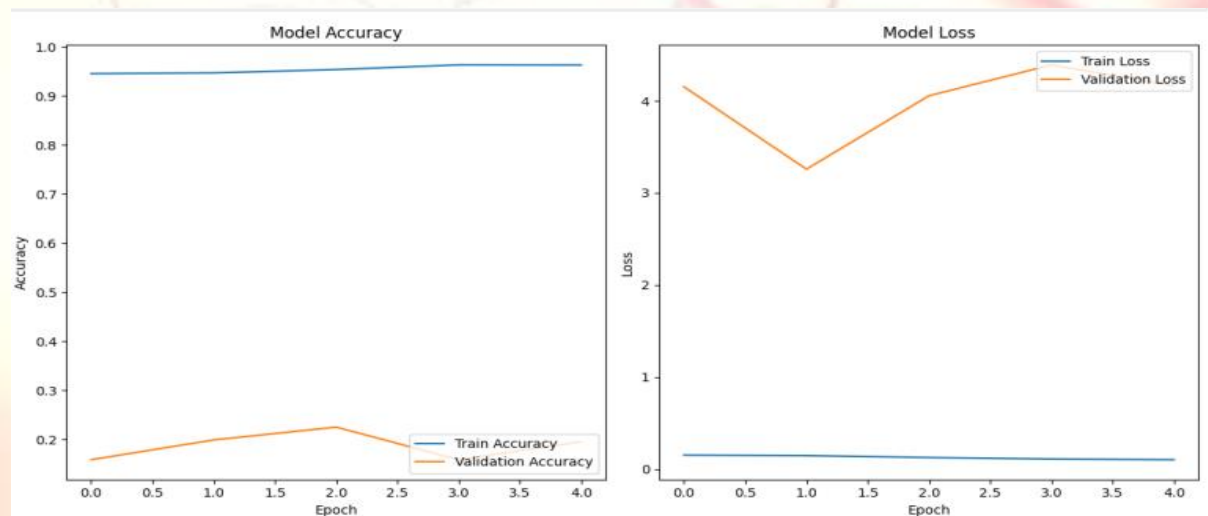


Fig 6.1: Model accuracy vs model loss

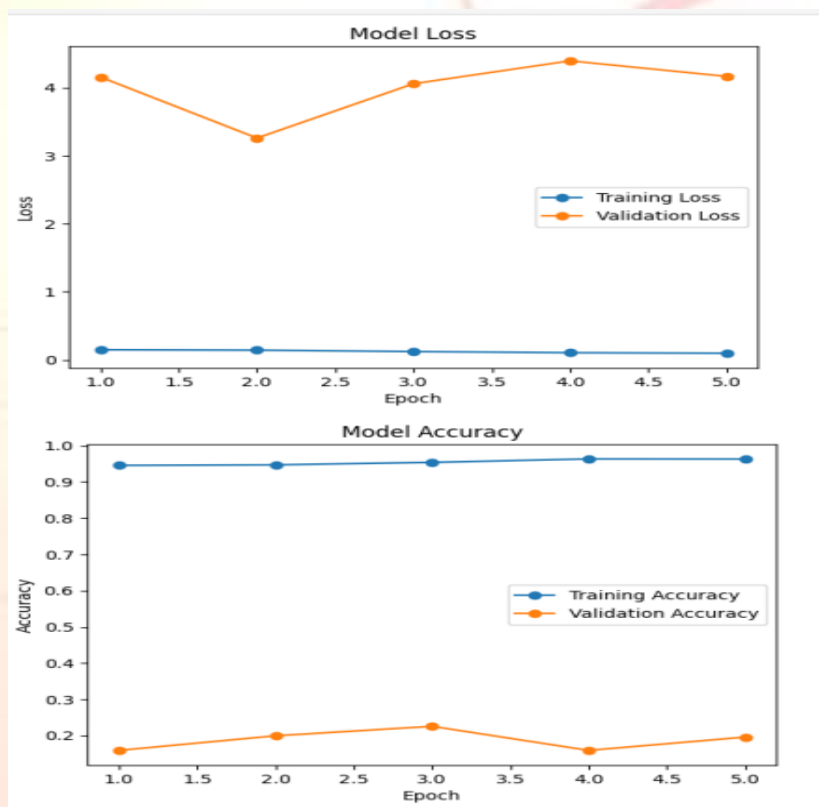
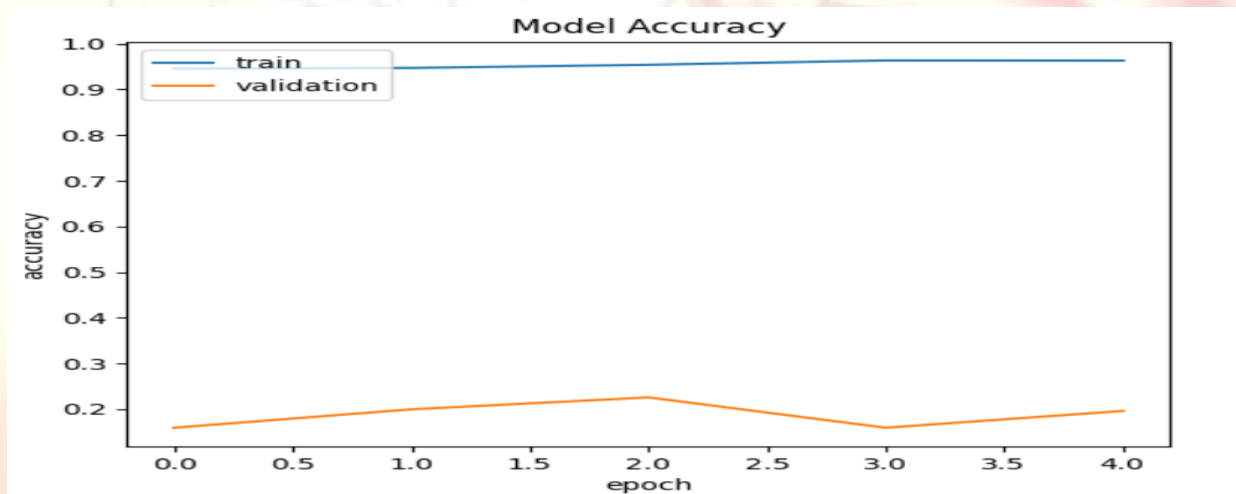


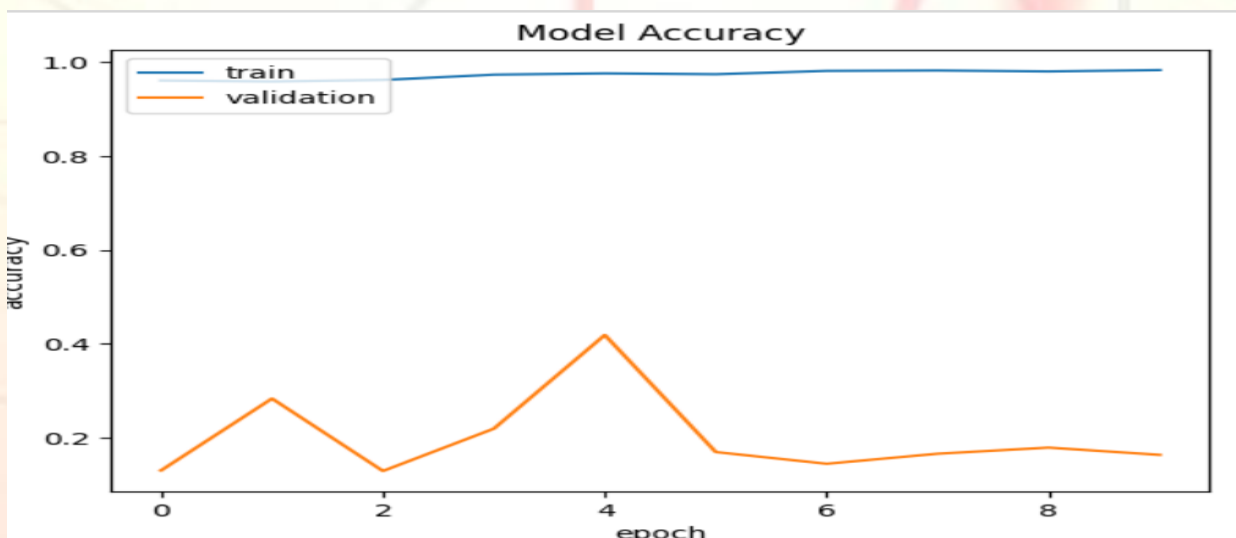
Fig 6.2: Model accuracy vs Model Loss

The graphs show that the model's training loss is consistently low, while the validation loss is high and fluctuating, indicating overfitting. The training accuracy remains high, but the validation accuracy is low and unstable, further confirming that the model is not generalizing well to unseen data.



The image shows a line graph illustrating the accuracy of the model over multiple epochs. The x-axis represents the number of epochs, and the y-axis represents the accuracy. There are two lines: one for the training accuracy and one for the validation accuracy.

The graph also shows a typical training and validation curve for a machine-learning and deep-learning model. The increasing training accuracy and generally increasing validation accuracy suggests a well-trained model, but the slight decrease in validation accuracy towards the end might indicate the need for further tuning or regularization to prevent overfitting.



The image shows a line graph illustrating the accuracy of the model over multiple epochs. The x-axis represents the number of epochs, and the y-axis represents the accuracy. There are two lines: one for the training accuracy and one for the validation accuracy.

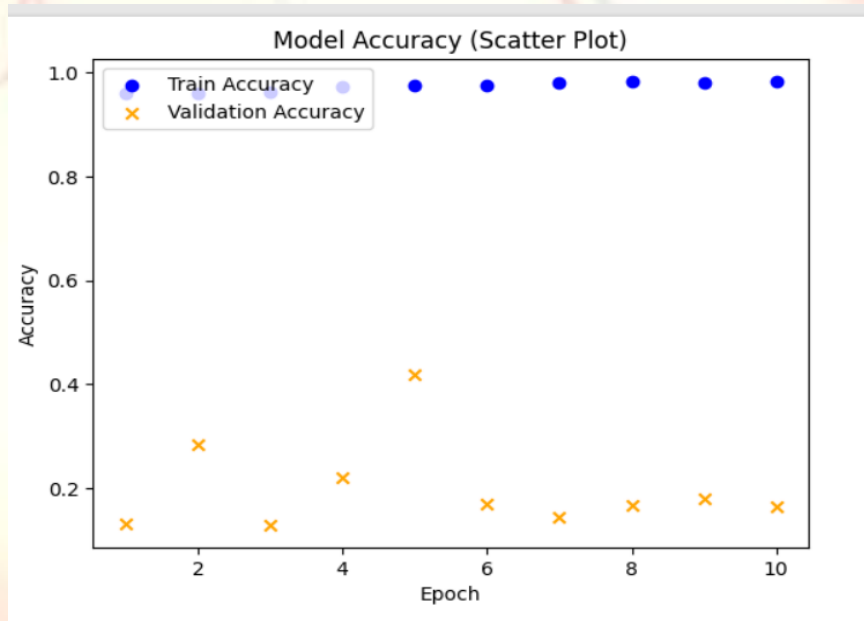


Fig 7.1: Scatter-plot (Accuracy vs Epoch)

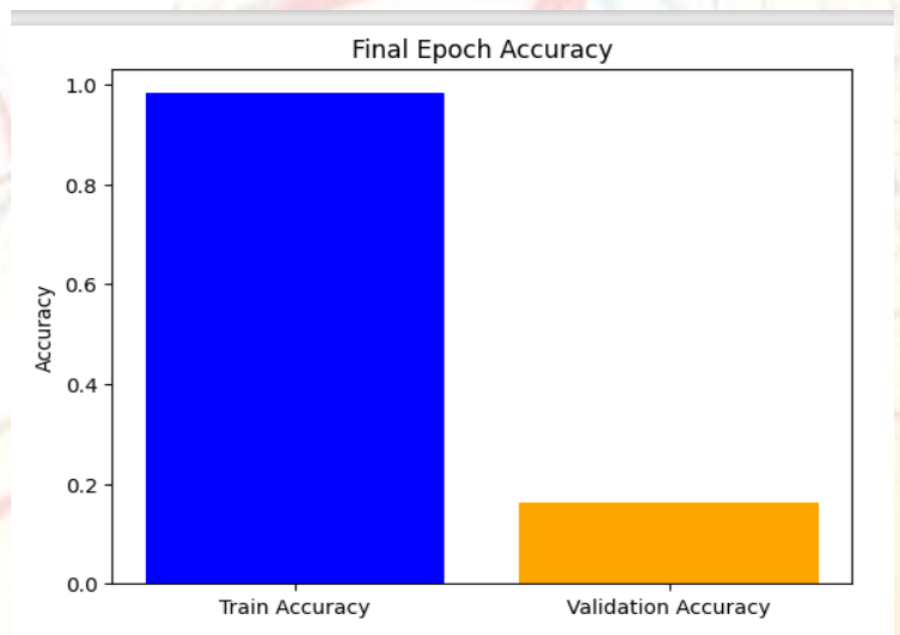


Fig 7.2: Bar-plot (Train-Accuracy vs Validation-Accuracy)

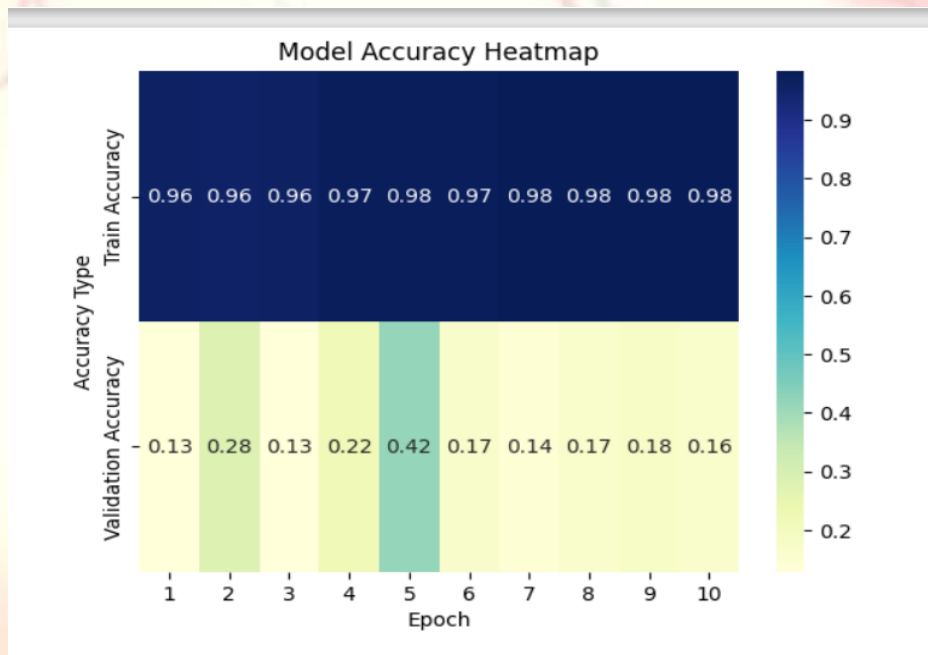


Fig 7.3: Heatmap (Accuracy-type vs Epoch)

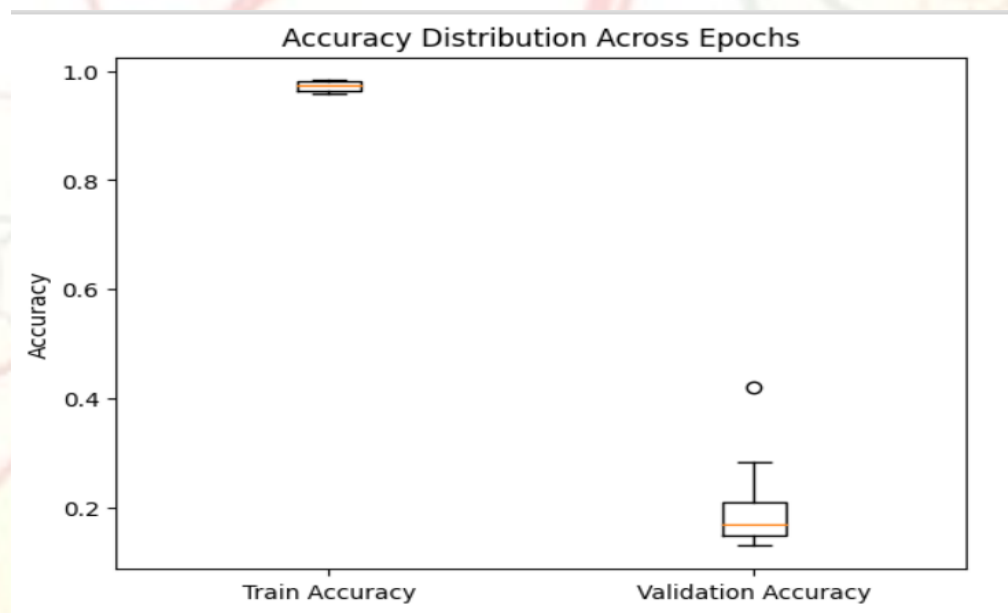


Fig 7.4: box plot (Train-Accuracy vs Validation-Accuracy)



Chapter 8:

Result Analysis

To analyse our model, evaluate metrics like accuracy, precision, recall, F1-score, and confusion matrix. Consider the different types of Visualizing plots and accuracy checking and also another essential factor of this model carefully to identify strengths and weaknesses, and make informed decisions for improvement.

Chapter 9:

Conclusions

The eye is a crucial sense organ, and vision loss can severely impact quality of life. Eye conditions can also signal serious health issues, but many lack access to proper eye care. The rise of deep learning and image processing offers significant potential for detecting and diagnosing these diseases. This study applied Convolutional Neural Networks (CNN) and transfer learning to identify cataracts, diabetic retinopathy, and glaucoma. While CNN alone struggled with classification, its performance greatly improved with transfer learning, using a pre-trained Efficient Net model. The results showed high accuracy, recall, specificity, precision, and F1 scores, with ROC curves demonstrating the effectiveness of the proposed classifier. This approach could greatly aid ophthalmology in addressing global eye health challenges.

Chapter 10:

FUTURE SCOPE AND LIMITATION

Future work in eye disease classification using CNN and TensorFlow could involve improving model accuracy by leveraging larger, more diverse datasets and exploring advanced architectures like Vision Transformers. Enhancing model interpretability to assist clinicians in decision-making and developing lightweight models for deployment on mobile devices are also potential areas of focus. For further research, we plan to investigate the efficacy of transfer learning on a more diverse dataset that includes other types of eye diseases. And explore other applications of transfer learning in disease detection tasks. This adding new diseases and their symptoms to the model. We will also try to deeply classify the infection of the mentioned diseases. The classification will include elementary and urgent cases. Urgent cases are cases that require urgent treatments or surgeries.

The project on eye disease classification using CNN and TensorFlow encounters several limitations. The model may overfit to the training data if the dataset is not sufficiently large or diverse, leading to poor generalization. Variations in image quality and resolution can affect the model's accuracy, and the system might struggle with rare or atypical eye diseases. Additionally, differences in imaging devices and patient demographics can impact performance. Ensuring robust model performance across different conditions remains a challenge.

REFERENCES

1. Akram A, & Debnath, R. "An automated eye disease recognition system from visual content of facial images using machine learning techniques" Turkish Journal of Electrical Engineering and Computer Science, 28(2), 917-932,2020.
2. Prasad, K., Sajith, P. S., Neema, M., Madhu, L., & Priya, P. N., "Multiple eye disease detection using Deep Neural Network". In TENCON IEEE Region 10 Conference (TENCON) (pp. 2148-2153), 2019.
3. jain, L., Murthy, H. S., Patel, C., & Bansal, D., "Retinal eyedisease detection using deep learning", Fourteenth International Conference on Information Processing (ICINPRO) (pp. 1-6). IEEE, 2018.
4. Fageeri, S. O., Ahmed, S. M. M., Almubarak, S. A., & Mu'azu, A. A., "Eye refractive error classification using machine learning techniques", International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE) (pp. 1-6), January 2017.
5. K. Babski-Reeves, B. Eksioglu, D. Hampton, eds, Proceedings of the IISE Annual Conference , Expo 2023.
6. Umesh, L., Mrunalini, M., & Shinde S., " Review of image processingand machine learning techniques for eye disease detection and classification" International Research Journal of Engineering and Technology, 3(3), 547-551,2016.