DIAGNOSIS OF EYE FUNDUS DISEASE CLASSIFICATION AND SEGMENTATION USING ARTIFICIAL INTELLIGENCE TECHNIQUES

A PROJECT REPORT

Submitted by

Surya Anush M [211420104278]

Prashanth S [211420104201]

Sai Charan Raj G [211420104235]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report "**DIAGNOSIS OF EYE FUNDUS DISEASE CLASSIFICATION AND SEGMENTATION USING ARTIFICIAL INTELLIGENCE TECHNIQUES**" is the bonafide work of **"Sai Charan Raj G , Prashanth S ,Surya Anush M"** who carried out the project work under my supervision.

Signature of the HOD with date

DR L. JABASHEELA M.E., Ph.D.,

PROFESSOR AND HEAD

Department of Computer Science and Engineering,
Panimalar Engineering College,
Chennai - 123

Signature of the Supervisor with date

DR. S. BALAJI, B. Tech, M.E, PhD,

PROFESSOR

Department of Computer Science andEngineering,

Panimalar Engineering College,

Chennai - 123

Submitted for the Project Viva-Voce Examination held on .....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We Surya Anush M [211420104278], Sai Charan Raj G [211420104235], Prashanth S [211420104201] hereby declare that this project report titled "**DIAGNOSIS OF EYE FUNDUS DISEASE CLASSIFICATION AND SEGMENTATION USING ARTIFICIAL INTELLIGENCE TECHNIQUES**", under the guidance of Dr. Balaji S is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

Surya Anush M [211420104278]

Sai Charan Raj G [211420104235]

Prashanth S [211420104201]

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Project Coordinator** Dr. V Subedha and **Project Guide** Dr. Balaji S and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

Surya Anush [211420104278]
Sai Charan Raj G [211420104235]
Prashanth S [211420104201]

ABSTRACT

Eye fundus diseases are critical conditions that can lead to severe vision impairment and even permanent blindness if not diagnosed and treated promptly. Manual diagnosis of these diseases is time-consuming and heavily reliant on the expertise of ophthalmologists. This research aims to develop an efficient and accurate diagnostic system for eye fundus disease classification and segmentation using artificial intelligence techniques. The study involves the compilation of a comprehensive dataset of eye fundus images, encompassing various types of diseases, including diabetic retinopathy, age-related macular degeneration, glaucoma, and others. Each image is accompanied by corresponding ground-truth annotations provided by expert ophthalmologists for segmentation. The experimental results demonstrate the effectiveness and reliability of the proposed system in accurately classifying eye fundus diseases and segmenting affected regions within the images. The AI models achieve high accuracy and provide valuable insights into the presence and extent of various fundus diseases.

TABLE OF CONTENT

	LIST OF FIGURES	
FIG NO.	**FIGURE NAME**	**PAGE NO**

CHAPTER 1

INTRODUCTION

The diagnosis of eye fundus diseases plays a crucial role in the early detection and management of various ocular conditions, such as diabetic retinopathy, macular degeneration, and glaucoma. Timely and accurate identification of these diseases is essential to prevent vision loss and improve patient outcomes. In recent years, the integration of artificial intelligence (AI) techniques has revolutionized the field of ophthalmology, offering advanced tools for the classification and segmentation of eye fundus images.Artificial intelligence, particularly machine learning algorithms, has shown remarkable capabilities in analyzing large datasets of eye fundus images. These techniques enable automated identification and classification of subtle pathological changes that might be challenging for human observers to detect. The development of AI models for eye fundus disease diagnosis involves training algorithms on diverse datasets, encompassing a wide range of retinal pathologies and normal variations. This training allows the AI models to learn patterns and features indicative of specific diseases, paving the way for robust and accurate automated diagnostics.

1.1 Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the

International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammer bacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data Scientist

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

Required Skills for a Data Scientist

- **Programming**: Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning**: Natural Language Processing, Classification, Clustering.
- **Data Visualization**: Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms**: MongoDB, Oracle, Microsoft Azure, Cloudera.

1.2 ARTIFICIAL INTELLIGENCE:

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Some popular accounts use the term “artificial intelligence” to describe machines that mimic “cognitive” functions that humans associate with

the human mind, such as “learning” and “problem solving”, however this definition is rejected by major AI researchers.

1.3 Overview of the system

Define a problem

Gathering image data set

Evaluating algorithms

Detecting results

The steps involved in Building the data model is depicted below.

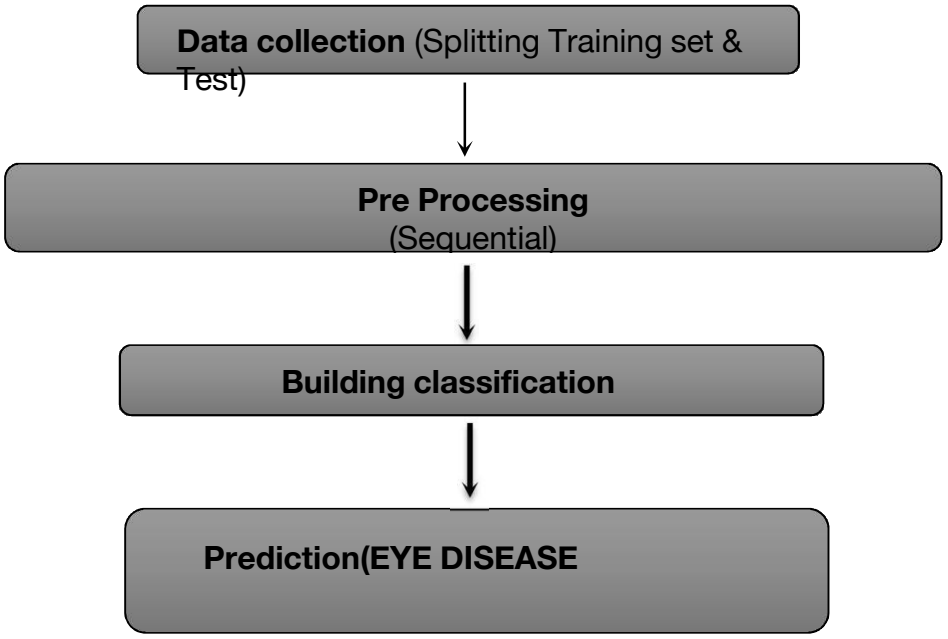


Fig: 1.1 data flow diagram for CNN model

1.4 Existing System

Diabetic retinopathy (DR) is a complication of diabetic Mellitus, developing retinal lesions that impair vision. The DR detection in the early stages avoids permanent vision loss. The treatments provide relief, and the vision loss due to DR is irreversible. The manual grading of DR is time-consuming and prone to human errors. The other real-time problem is exchanging patient fundus image information with hospitals worldwide while upholding the organizations' privacy concerns. When training a deep learning (DL) network, two critical factors to keep in mind are creating a collaborative platform and protecting patient data privacy. Therefore, an automated DR detection technique is required while protecting patient data and privacy. In this work, we propose a novel DR severity grading technique based on Federated Learning (FL), a recent advancement in DL called DRFL. FL is a new research paradigm that allows DL models to be trained collectively without disclosing clinical information. In DRFL, we combined the Federated averaging (FedAvg) technique and the median of the categorical cross-entropy loss. In comparison to FedAvg, the median cross-entropy is better suited for either under-fitted or over-fitted clients. Also, we propose a novel central server that extracts multi-scale features from the fundus images to identify small lesions present in the fundus image.

1.5 Project goal

The goal in the diagnosis of eye fundus disease classification and segmentation using artificial intelligence (AI) techniques is to enhance the efficiency, accuracy, and speed of identifying and categorizing various pathological conditions affecting the retina. The eye fundus, or the back of the eye, contains critical information about the health of the retina and is instrumental in diagnosing conditions such as diabetic retinopathy, macular degeneration, and glaucoma. Leveraging AI techniques in this domain aims to revolutionize the

traditional methods of diagnosis, providing a more sophisticated and automated approach to the analysis of fundus images.

Dropout layer

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

Image Data Generator

It is that rescales the image, applies shear in some range, zooms the image and does horizontal flipping with the image. This Image Data Generator includes all possible orientation of the image.

Epochs

It tells us the number of times model will be trained in forward and backward pass.

Validation process

Validation_data is used to feed the validation/test data into the model. Validation_steps denotes the number of validation/test samples.

1.6 Objectives

1. Develop a robust artificial intelligence (AI) model for automated diagnosis of eye fundus diseases with a focus on accurate disease classification.

2. Investigate and implement advanced machine learning algorithms to enhance the efficiency and reliability of eye fundus disease detection in medical images.

3. Explore innovative AI techniques for the segmentation of distinct regions within the eye fundus images, enabling precise localization of pathological features.

4. Evaluate the performance of the developed AI model in terms of sensitivity, specificity, and overall diagnostic accuracy to ensure its clinical utility.

5. Investigate the integration of deep learning architectures to improve the depth of feature extraction for more nuanced detection and differentiation of eye fundus diseases.

1.7 Scope

The scope of utilizing artificial intelligence (AI) techniques in the diagnosis of eye fundus diseases encompasses a comprehensive approach to disease classification and segmentation. By leveraging advanced machine learning algorithms, deep learning models, and image processing techniques, AI systems can analyze intricate features present in fundus images, aiding in the accurate identification and categorization of various eye conditions.

CHAPTER 2

LITERATURE SURVEY

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period.

Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them. Loan default trends have been long studied from a socio-economic stand point.

Review of Literature Survey

Title: Ocular Diseases Diagnosis in Fundus Images using a Deep Learning: Approaches, tools and Performance evaluation

Author: Yaroub Elloumi a,b,c , Mohamed Akila,* , Henda Boudegga

Year : 2023

Ocular pathology detection from fundus images presents an important challenge on health care. In fact, each pathology has different severity stages that may be deduced by verifying the existence of specific lesions. Each lesion is characterized by morphological features. Moreover, several lesions of different pathologies have similar features. We note that patient may be affected simultaneously by several pathologies. Consequently, the ocular pathology detection presents a multiclass classification with a complex resolution principle. Several detection methods of ocular pathologies from fundus images have been proposed. The methods based on deep learning are distinguished by higher performance detection, due to their capability to configure the network with respect to the detection objective. This work proposes a survey of ocular pathology detection methods based on deep learning. First, we study the existing methods either for lesion segmentation or pathology classification. Afterwards, we extract the principle steps of processing and we analyze the proposed neural network structures. Subsequently, we identify the hardware and software environment required to employ the deep learning architecture. Thereafter, we investigate about the experimentation principles involved to evaluate the methods and the databases used either for training and testing phases. The detection performance ratios and execution times are also reported and discussed.

Title : Automatic Detection of Diabetic Eye Disease Through Deep Learning Using Fundus Images

Author : Sarki, Rubina, Ahmed, Khandakar, Wang, Hua and Zhang

Year : 2020

Diabetes Mellitus, or Diabetes, is a disease in which a person's body fails to respond to insulin released by their pancreas, or it does not produce sufficient insulin. People suffering from diabetes are at high risk of developing various

eye diseases over time. As a result of advances in machine learning techniques, early detection of diabetic eye disease using an automated system brings substantial benefits over manual detection. A variety of advanced studies relating to the detection of diabetic eye disease have recently been published. This article presents a systematic survey of automated approaches to diabetic eye disease detection from several aspects, namely: i) available datasets, ii) image preprocessing techniques, iii) deep learning models and iv) performance evaluation metrics. The survey provides a comprehensive synopsis of diabetic eye disease detection approaches, including state of the art field approaches, which aim to provide valuable insight into research communities, healthcare professionals and patients with diabetes

Title : Data Driven Approach for Eye Disease Classification with Machine Learning

Author: Sadaf Malik , Nadia Kanwal , Mamoona Naveed Asghar

Year :2019

Medical health systems have been concentrating on artificial intelligence techniques for speedy diagnosis. However, the recording of health data in a standard form still requires attention so that machine learning can be more accurate and reliable by considering multiple features. The aim of this study is to develop a general framework for recording diagnostic data in an international standard format to facilitate prediction of disease diagnosis based on symptoms using machine learning algorithms. Efforts were made to ensure error-free data entry by developing a user-friendly interface. Furthermore, multiple machine learning algorithms including Decision Tree, Random Forest, Naive Bayes and Neural Network algorithms were used to analyze patient data based on multiple features, including age, illness history and clinical observations. This data was formatted according to structured hierarchies

designed by medical experts, whereas diagnosis was made as per the ICD-10 coding developed by the American Academy of Ophthalmology. Furthermore, the system is designed to evolve through self-learning by adding new classifications for both diagnosis and symptoms. The classification results from tree-based methods demonstrated that the proposed framework performs satisfactorily, given a sufficient amount of data. Owing to a structured data arrangement, the random forest and decision tree algorithms' prediction rate is more than 90% as compared to more complex methods such as neural networks and the naïve Bayes algorithm.

Title : Detection of glaucoma using artificial intelligence in fundus image: A narrative review

Author: Eman Hassan Hagar

Year : 2023

Glaucoma is a serious disease usually called ″silent thief of sight″. The disease develops with no observable signs and symptoms leading to blindness if not kept under control and observed in the early stages. A lot of work has been done over the years to increase the accuracy of detecting and predicting glaucomatous changes within the eyes. Artificial intelligence models using fundus imaging modalities are among the most promising tools to detect and predict glaucoma with high accuracy

Title : OCULAR EYE DISEASE PREDICTION USING MACHINE LEARNING

Author: Rachana Devanaboina, Sreeja Badri, Madhuri Reddy Depa, Dr.Sunil Bhutad

Year : 2021

The eye is the most important sense organ which enables us to see the world. Ocular eye diseases are some of the major problems for vision. In this ocular eye disease comes the most common disease, Cataract. Cataract is a misty form that affects the vision of the eye which causes blurriness. It is mostly found in elderly people due to their age. Computer-aided diagnosis is a bit complicated task for the detection of ocular eye diseases. In the present paperwork, we predict the ocular eye diseases based on Machine Learning algorithms which include Convolution Neural Networks (CNN) and image pre-processing. The accuracy of the outcome is displayed through the confusion matrix.

CHAPTER 3

PROPOSED SYSTEM

The proposed system aims to develop an advanced diagnostic tool for eye fundus disease classification and segmentation using cutting-edge artificial intelligence techniques. Leveraging the power of deep learning algorithms, the system offers an efficient and accurate solution to assist ophthalmologists in diagnosing and treating various eye fundus diseases promptly. A comprehensive dataset of eye fundus images is collected, comprising diverse cases of different eye fundus diseases, including diabetic retinopathy, age-related macular degeneration, glaucoma, and others. The dataset is annotated by expert ophthalmologists to provide ground-truth segmentation masks for each image. The proposed system employs Convolutional Neural Networks (CNNs) for eye fundus disease classification. The CNN architecture is trained on the pre-processed dataset, enabling it to learn distinctive disease patterns and features from the images. The dataset is split into training, validation, and testing sets. During the training process, both the disease classification CNN and the segmentation U-Net are optimized using back propagation and gradient descent techniques to minimize the loss and maximize accuracy.

3.1 Advantages

- We classify the disease and segmented.
- We build a production level application for deployment purposes.
- Huge amount data above 5000 images.
- Find out 4 classification.

3.2 Python

Introduction

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple .

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the C-Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

3.3 Deep Learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human disease so deep learning is also a kind of mimic of human disease. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. A formal definition of deep learning is- neurons Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. In disease approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbors. The question here is how it recreates these neurons in a computer. So, it creates an artificial structure called an artificial neural net where we have nodes or neurons. It has some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

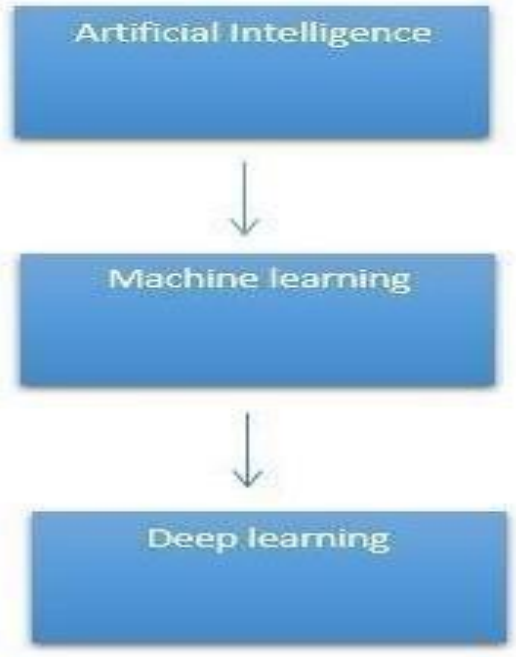


Fig 3.1 Deep Learning

It need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). It needs to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Choose the Deep Learning Algorithm appropriately. Algorithm should be used while training the dataset. Final testing should be done on the dataset

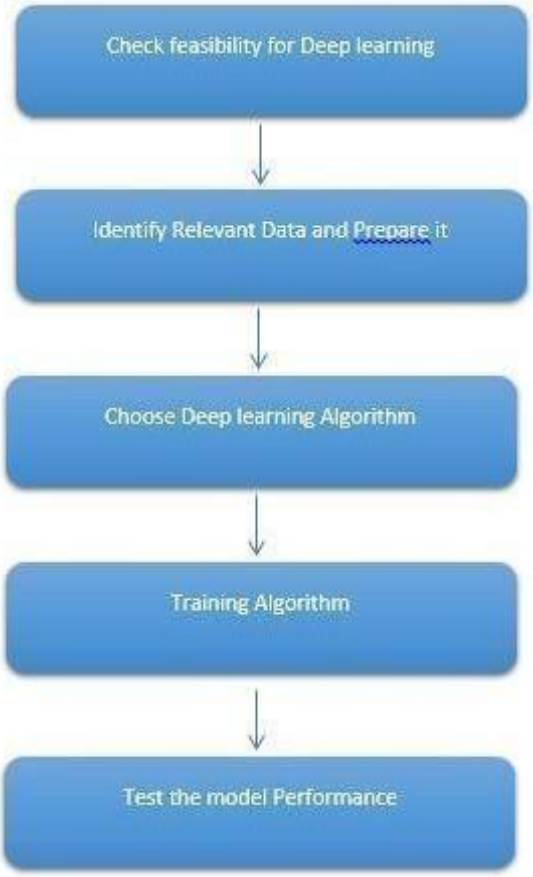


Fig 3.2 Steps for Model

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs

have various differences from biological disease. Specifically, neural networks tend to be static and symbolic, while the biological disease of most living organisms is dynamic (plastic) and analogue.

3.4 Machine Learning

It seems like your request is quite brief. "ML" typically stands for "Machine Learning." Machine learning is a subset of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to perform a specific task without being explicitly programmed for that task. If you have a specific question or topic related to machine learning that you'd like more information about, please provide more details, and I'll do my best to assist you!

3.5 Methodology:

Preprocessing and Training the model (CNN): The dataset is preprocessed such as Image reshaping, resizing and conversion to an array form. Similar processing is also done on the test image. A dataset consisting of about 4 different Cataract, Diabetic retinopathy, Glaucoma, Normal out of which any image can be used as a test image for the software.

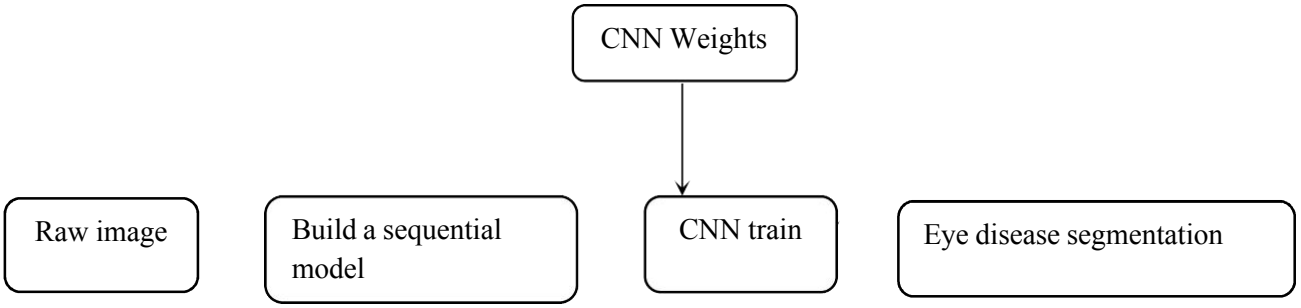


Fig 3.3 CNN Model of Preprocessing and Testing

The train dataset is used to train the model (CNN) so that it can identify the test image and the disease it has CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D. After the model is trained successfully, the software can identify the different Cataract, Diabetic retinopathy, Glaucoma, Normal Classification image contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the

CNN Model steps:

Conv2d:

The 2D convolution is a fairly simple operation at heart: you start with a kernel, which is simply a small matrix of weights. This kernel "slides" over the 2D input data, performing an elementwise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel.

This is all in pretty stark contrast to a fully connected layer. In the above example, we have 5×5=25 input features, and 3×3=9 output features. If this were a standard fully connected layer, you'd have a weight matrix of 25×9 = 225 parameters, with every output feature being the weighted sum of every single input feature. Convolutions allow us to do this transformation with only 9 parameters, with each output feature, instead of "looking at" every input feature, only getting to "look" at input features coming from roughly the same location. Do take note of this, as it'll be critical to our later discussion.

MaxPooling2D layer

Down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size)

for each channel of the input. The window is shifted by strides along each dimension.

The resulting output, when using the "valid" padding option, has a spatial shape (number of rows or columns) of: output_shape = math.floor((input_shape - pool_size) / strides) + 1 (when input_shape >= pool_size)

The resulting output shape when using the "same" padding option is: output_shape = math.floor((input_shape - 1) / strides) + 1

Input shape

- If data_format='channels_last': 4D tensor with shape (batch_size, rows, cols, channels).
- If data_format='channels_first': 4D tensor with shape (batch_size, channels, rows, cols).

Output shape

- If data_format='channels_last': 4D tensor with shape (batch_size, pooled_rows, pooled_cols, channels).
- If data_format='channels_first': 4D tensor with shape (batch_size, channels, pooled_rows, pooled_cols).

CNN Model steps

Conv2d

The 2D convolution is a fairly simple operation at heart: you start with a kernel, which is simply a small matrix of weights. This kernel “slides” over the 2D input data, performing an elementwise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel.

This is all in pretty stark contrast to a fully connected layer. In the above example, we have 5×5=25 input features, and 3×3=9 output features. If this were a standard fully connected layer, you'd have a weight matrix of 25×9 = 225 parameters, with every output feature being the weighted sum of every single input feature. Convolutions allow us to do this transformation with only 9 parameters, with each output feature, instead of "looking at" every input feature, only getting to "look" at input features coming from roughly the same location. Do take note of this, as it'll be critical to our later discussion.

MaxPooling2D layer

Down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size) for each channel of the input. The window is shifted by strides along each dimension.

The resulting output, when using the "valid" padding option, has a spatial shape (number of rows or columns) of: output_shape = math.floor((input_shape - pool_size) / strides) + 1 (when input_shape >= pool_size)

The resulting output shape when using the "same" padding option is: output_shape = math.floor((input_shape - 1) / strides) + 1

Input shape

- If data_format='channels_last': 4D tensor with shape (batch_size, rows, cols, channels).
- If data_format='channels_first': 4D tensor with shape (batch_size, channels, rows, cols).

Output shape

- If data_format='channels_last': 4D tensor with shape (batch_size, pooled_rows, pooled_cols, channels).
- If data_format='channels_first': 4D tensor with shape (batch_size, channels, pooled_rows, pooled_cols).

Flatten layer

It is used to flatten the dimensions of the image obtained after convolving it. Dense: It is used to make this a fully connected model and is the hidden layer. Dropout: It is used to avoid over fitting on the dataset and dense is the output layer contains only one neuron which decide to which category image belongs.

Flatten is used to flatten the input. For example, if flatten is applied to layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, 4)

Flatten has one argument as follows

keras.layers.Flatten(data_format = None)

data_format is an optional argument and it is used to preserve weight ordering when switching from one data format to another data format. It accepts either channels_last or channels_first as value. channels_last is the default one and it identifies the input shape as (batch_size, ..., channels) whereas channels_first identifies the input shape as (batch_size, channels, ...)

Dense layer

Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True). These are all attributes of Dense.

Input shape

N-D tensor with shape: (batch_size, ..., input_dim). The most common situation would be a 2D input with shape (batch_size, input_dim).

Output shape

N-D tensor with shape: (batch_size, ..., units). For instance, for a 2D input with shape (batch_size, input_dim), the output would have shape (batch_size, units)

3.6 Artificial Neural Networks

Artificial Neural Networks contain artificial neurons which are called **units**. These units are arranged in a series of layers that together constitute the whole Artificial Neural Network in a system. A layer can have only a dozen units or millions of units as this depends on how the complex neural networks will be required to learn the hidden patterns in the dataset. Commonly, Artificial Neural Network has an input layer, an output layer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyze or learn about.

3.7 Architecture of CNN

CONVOLUTIONAL NEURAL NETWORK

A Convolutional neural network (CNN) is one type of Artificial Neural Network. A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data.

In other non-vision applications, a one dimensional convolution mayslide vertically over an input matrix.

Padding

To handle the edge pixels there are several approaches:

- Losing the edge pixels
- Padding with zero value pixels
- Reflection padding

Reflection padding is by far the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image.

Strides

It is common to use a stride two convolution rather than a stride one convolution, where the convolutional kernel strides over 2 pixels at a time, for example our 3x3 kernel would start at position (1,1), then stride to (1,3), then to 1, 5) and so on, halving the size of the output channel/feature map, compared to the convolutional kernel taking strides of one.

Rectified Linear Unit (ReLU)

A Rectified Linear Unit is used as a non-linear activation function. A ReLU says if the value is less than zero, round it up to zero.

Normalisation

Normalisation is the process of subtracting the mean and dividing by the standard deviation. It transforms the range of the data to be between -1 and 1 making the data use the same scale, sometimes called Min-Max scaling.It is common to normalize the input features, standardising the data by removing the mean and scaling to unit variance. It is often important the input features are centred around zero and have variance in the same order. With some data, such as images the data is scaled so that it's range is between 0 and 1, most simply dividing the pixel values by 255.

Batch normalisation

Batch normalisation has the benefits of helping to make a network output more stable predictions, reduce overfitting through regularisation and speeds up training by an order of magnitude.

Batch normalisation is the process of carrying normalisation within the scope activation layer of the current batch, subtracting the mean of the batch's activations and dividing by the standard deviation of the batches activations.
Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

- epsilon is small constant (configurable as part of the constructor arguments)
- gamma is a learned scaling factor (initialized as 1), which can be disabled by passing scale=False to the constructor.
- beta is a learned offset factor (initialized as 0), which can be disabled by passing center=False to the constructor.

During inference (i.e. when using evaluate() or predict() or when calling the layer/model with the argument training=False (which is the default), the layer

normalizes its output using a moving average of the mean and standard deviation of the batches it has seen during training. That is to say, it returns gamma * (batch - self.moving_mean) / sqrt(self.moving_var + epsilon) + beta.

self.moving_mean and self.moving_var are non-trainable variables that are updated each time the layer in called in training mode, as such:

- moving_mean = moving_mean * momentum + mean(batch) * (1 - momentum)
- moving_var = moving_var * momentum + var(batch) * (1 - momentum)

Arguments

- **axis**: Integer, the axis that should be normalized (typically the features axis). For instance, after a Conv2D layer with data_format="channels_first", set axis=1 in BatchNormalization.
- **momentum**: Momentum for the moving average.
- **epsilon**: Small float added to variance to avoid dividing by zero.
- **center**: If True, add offset of beta to normalized tensor. If False, beta is ignored.
- **scale**: If True, multiply by gamma. If False, gamma is not used. When the next layer is linear (also e.g. nn.relu), this can be disabled since the scaling will be done by the next layer.
- **beta_initializer**: Initializer for the beta weight.
- **gamma_initializer**: Initializer for the gamma weight.
- **moving_mean_initializer**: Initializer for the moving mean.
- **moving_variance_initializer**: Initializer for the moving variance.
- **beta_regularizer**: Optional regularizer for the beta weight.
- **gamma_regularizer**: Optional regularizer for the gamma weight.

- **beta_constraint**: Optional constraint for the beta weight.
- **gamma_constraint**: Optional constraint for the gamma weight.

Call arguments

- **inputs**: Input tensor (of any rank).
- **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode.

3.8 Types of CNN

- ALEXNET
- LENET

AlexNET

AlexNet architecture consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 softmax layer. 2. Each convolutional layer consists of convolutional filters and a nonlinear activation function ReLU. 3. The pooling layers are used to perform max pooling. AlexNet contained eight layers; the first five were convolutional layers, some of them followed by max-pooling layers, and the last three were fully connected layers. It used the non-saturating ReLU activation function, which showed improved training performance over tanh and sigmoid.

Architecture of AlexNet:

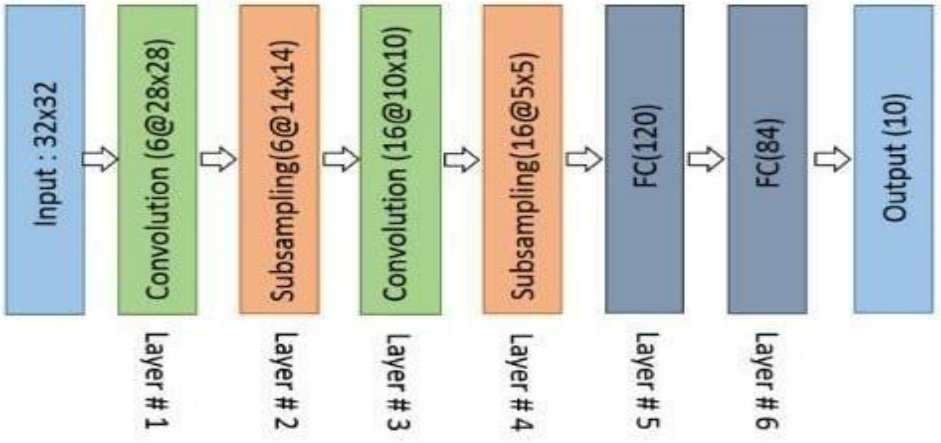


Fig 3.4 Architecture of AlexNet

Convolutional layers

Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

Pooling layers:

Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a

filter region. These are typically used to reduce the dimensionality of the network.

Dense or Fully connected layers

Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP.

LENET

As a representative of the early convolutional neural network, LeNet possesses the basic units of convolutional neural network, such as convolutional layer, pooling layer and full connection layer, laying a foundation for the future development of convolutional neural network. As shown in the figure (input image data with 32*32 pixels) LeNet-5 consists of seven layers. In addition to input, every other layer can train parameters. In the figure, Cx represents convolution layer, Sx represents sub-sampling layer, Fx represents complete connection layer, and x represents layer index.

Fig 3.5 Architecture of LeNet

3.9 LIST OF MODULES

1. Data Analysis
2. Manual Architecture
3. LeNet Architecture
4. U-Net Architecture
5. Deployment

3.10 MODULE DESCRIPTION

IMPORT THE GIVEN IMAGE FROM DATASET

We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function. Here we set train, test, and validation also we set target size, batch

size and class-mode from this function we have to train using our own created network by adding layers of CNN.

TO TRAIN THE MODULE BY GIVEN

1. Data Analysis

Data analysis is the process of cleaning, changing, and processing raw data, and extracting actionable, relevant information that helps businesses make informed decisions. The procedure helps reduce the risks inherent in decision-making by providing useful insights. The data analysis process, or alternately, data analysis steps, involves gathering all the information, processing it, exploring the data, and using it to find patterns and other insights.

Manual Architecture

Creating a manual architecture for image segmentation typically involves designing a neural network or algorithm that can process an input image and produce pixel-level segmentation masks or regions of interest. Here's a simplified manual architecture for image segmentation:

1. Input Image: The architecture takes an input image as its primary input. This can be a grayscale or color image depending on your application.

2. Preprocessing: Preprocess the input image to enhance features and reduce noise. Common preprocessing steps include resizing, normalization, and data augmentation.

3. Convolutional Neural Network (CNN): Use a convolutional neural network as the backbone of your segmentation architecture. CNNs are highly effective at capturing local patterns and spatial features in images.

- You can use a pre-trained CNN architecture like VGG, ResNet, or U-Net as a starting point, or design a custom architecture.

4. Encoder-Decoder Architecture: Many segmentation architectures follow an encoder-decoder structure

- Encoder: The encoder part of the network extracts features from the input image through a series of convolutional layers. These layers reduce spatial dimensions while increasing the number of feature maps.

- Decoder: The decoder part of the network upsamples the feature maps to the original image size while reducing the number of channels. This process helps generate the segmentation mask.

5. Skip Connections: To improve segmentation accuracy, consider adding skip connections that concatenate feature maps from the encoder to the corresponding layers in the decoder. This allows the network to capture both high-level and low-level features.

6. Convolutional Transpose (Deconvolution) Layers: Use convolutional transpose layers (sometimes called deconvolution layers) to upsample feature maps in the decoder. These layers expand the spatial resolution of the feature maps.

7. Activation Function: Apply an activation function, typically a softmax or sigmoid, to the final layer of the decoder to produce the segmentation mask. For binary segmentation, sigmoid is often used; for multi-class segmentation, softmax is common.

8. Loss Function: Define an appropriate loss function to measure the difference between the predicted segmentation mask and the ground truth mask. Common loss functions for segmentation include binary cross-entropy and categorical cross-entropy.

9. Optimization Algorithm: Use an optimization algorithm like stochastic gradient descent (SGD), Adam, or RMSprop to update the network's weights and minimize the loss function.

10. Training Data: Train the network using a dataset of annotated images. The dataset should include input images and corresponding segmentation masks.

11. Post-processing: Apply post-processing techniques if needed, such as morphological operations (erosion, dilation) to refine the segmentation mask.

12. Inference: During inference, feed an unseen image through the trained network to obtain the segmentation mask.

13. Evaluation: Evaluate the segmentation accuracy using metrics like Intersection over Union (IoU), Dice coefficient, or pixel accuracy.

Le-Net Archietecture

LeNet, short for "LeNet-5," is a classic convolutional neural network (CNN) architecture developed by Yann LeCun in the early 1990s. While LeNet is renowned for its role in image classification tasks, it can be adapted for image segmentation, including applications like Eye fundus diseases cancer segmentation. Here's a high-level overview of how LeNet can be modified for this purpose:

1. Input Image

2. Preprocessing

3. LeNet Architecture

4. Convolutional Layers

5. Pooling Layers

6. Encoder-Decoder Modification

7. Decoder

8. Activation Function

9. Loss Function

10. Optimization Algorithm

11. Training Data

12. Post-processing (Optional)

13. Evaluation

U-Net Architecture

The U-Net architecture is a popular deep learning architecture for image segmentation tasks, including Eye fundus diseases cancer segmentation. It was originally developed for biomedical image segmentation and has since found applications in various medical image analysis tasks. The name "U-Net" is derived from the U-shaped architecture of the network.

Here's an overview of the U-Net architecture for Eye fundus diseases cancer segmentation:

Encoder-Decoder Structure

U-Net follows an encoder-decoder structure. It consists of two main parts: the encoder and the decoder.

The encoder captures features from the input image at multiple scales by using convolutional and pooling layers. It gradually reduces the spatial dimensions while increasing the number of feature maps.

The decoder then takes these features and upsamples them to the original image size while reducing the number of feature maps. This helps generate a detailed segmentation mask.

Skip Connections

One of the key innovations of U-Net is the use of skip connections that connect corresponding layers between the encoder and decoder.

These skip connections allow the network to capture both high-level and low-level features, which is crucial for accurate segmentation.

Skip connections concatenate feature maps from the encoder to the corresponding layers in the decoder.

Contracting and Expansive Paths

The encoder path is often referred to as the contracting path because it reduces spatial dimensions.

The decoder path is called the expansive path because it increases the spatial dimensions.

Skip connections connect the contracting and expansive paths, facilitating the flow of information between them.

Final Layer

The final layer of the U-Net architecture typically consists of a convolutional layer with a softmax activation function for multi-class segmentation or a sigmoid activation function for binary segmentation.

The output of this layer is the segmentation mask, where each pixel is classified into the desired classes (e.g., tumor or background).

Loss Function

Common loss functions for U-Net-based segmentation tasks include binary cross-entropy loss for binary segmentation or categorical cross-entropy loss for multi-class segmentation.

Training Data

To train a U-Net model for Eye fundus diseases cancer segmentation, you need a dataset of annotated Eye fundus diseases images. The dataset should include input mammograms or other relevant images and corresponding pixel-level segmentation masks indicating the regions of interest (e.g., tumors).

Inference

During inference, you feed an unseen Eye fundus diseases image through the trained U-Net model to obtain the segmentation mask, which highlights the areas of interest, such as tumors or lesions.

Post-processing

Post-processing steps, such as morphological operations (e.g., erosion, dilation) or connected component analysis, can be applied to refine the segmentation mask and remove any artifacts.

CHAPTER 4

PROJECT REQUIREMENTS

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
    A. Hardware requirements
    B. software requirements

4.1 Functional Requirements

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like tensorflow, keras, matplotlib.

4.2 Non-Functional Requirements

Process of functional steps,

1. Problem define
2. Preparing data

3. Evaluating algorithm
4. Improving results
5. Prediction the result

Environment Requirements

Framework : Keras.

Software Requirements

Operating System : Windows / Linux

Simulation Tool : Anaconda with Jupyter Notebook

Language : Python

Hardware requirements

Processor : Intel i3

Hard disk : minimum 400 GB

RAM : minimum 4 GB

4.3 Django

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

4.4 HTML

HTML stands for HyperText Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

Basic Construction of an HTML Page

These tags should be placed underneath each other **at the top of every HTML page** that you create.

Fig 4.1 Basic Structure of HTML Page

<!DOCTYPE html> — This tag **specifies the language** you will write on the page. In this case, the language is HTML 5.

<html> — This tag signals that from here on we are going to write in HTML code.

<head> — This is where all the **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

<body> — This is where the **content of the page** goes.

Further Tags

Inside the <head> tag, there is one tag that is always included: <title>, but there are others that are just as important:

<title>

This is where we **insert the page name** as it will appear at the top of the browser window or tab.

<meta>

This is where information *about* the document is stored: character encoding, name (page context), description.

HeadTag

<head>
<title>My First Webpage</title>
<meta charset="UTF-8">
<meta name="description" content="This field contains information about your page. It is usually around two sentences long.">.
<meta name="author" content="Conor Sheils">
</header>

Adding Content

Next, we will make <body> tag.

The HTML <body> is where we add the content which is designed for viewing by human eyes.

This includes **text, images, tables, forms** and everything else that we see on the internet each day.

How to Add HTML Headings To Your Web Page
In HTML, headings are written in the following elements:

- <h1>
  - <h2>
    - <h3>
      - <h4>
        - <h5>
          - <h6>

As you might have guessed <h1> and <h2> should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

Add Text In HTML

Adding text to our HTML page is simple using an element opened with the tag <p> which **creates a new paragraph**. We place all of our regular text inside the element <p>.

Element	Meaning	Purpose
<b>	Bold	Highlight important information

****	Strong	Similarly to bold, to highlight key text
<i>	Italic	To denote text
****	Emphasised Text	Usually used as image captions
<mark>	Marked Text	Highlight the background of the text
<small>	Small Text	To shrink the text
<strike>	Striked Out Text	To place a horizontal line across the text
<u>	Underlined Text	Used for links or text highlights
<ins>	Inserted Text	Displayed with an underline to show an inserted text
<sup>	Superscript Text	Another typographical presentation style

When we write text in HTML, we also have a number of other elements we can use to **control the text or make it appear in a certain way**.

Add Links In HTML

As you may have noticed, the internet is made up of lots of links.

Almost everything you click on while surfing the web is a link **takes you to another page** within the website you are visiting or to an external site.

Links are included in an attribute opened by the **<a>** tag. This element is the first that we've met which uses an attribute and so it **looks different to previously mentioned tags**.

```
<a href="http://www.google.com">Google</a>
```

Image Tag

In today's modern digital world, images are everything. The **<img>** tag has everything you need to display images on your site. Much like the <a> anchor element, <img> also contains an attribute.

The attribute *features information* for your computer regarding the **source**, **height**, **width** and **alt text** of the image

```
<img src="yourimage.jpg" alt="Describe the image" height="X" width="X">
```

4.5 CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.

CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

CSS Syntax

```
Selector {
```

```
        Property 1 : value;

        Property 2 : value;

        Property 3 : value;

    }
```

- Selector: selects the element you want to target
- Always remains the same whether we apply internal or external styling
- There are few basic selectors like tags, id's, and classes
- All forms this key-value pair
- Keys: properties(attributes) like color, font-size, background, width, height,etc
- Value: values associated with these properties

CSS Comment

- Comments don't render on the browser
- Helps to understand our code better and makes it readable.
- Helps to debug our code
- Two ways to comment:
    - Single line

CSS How-To

- There are 3 ways to write CSS in our HTML file.
    - Inline CSS
    - Internal CSS
    - External CSS

- Priority order

Inline > Internal > External

Inline CSS

- Before CSS this was the only way to apply styles
- Not an efficient way to write as it has a lot of redundancy
- Self-contained
- Uniquely applied on each element
- The idea of separation of concerns was lost

Internal CSS

- With the help of style tag, we can apply styles within the HTML file
- Redundancy is removed
- But the idea of separation of concerns still lost
- Uniquely applied on a single document
- Example:

External CSS

- With the help of <link> tag in the head tag, we can apply styles
- Reference is added
- File saved with .css extension
- Redundancy is removed
- The idea of separation of concerns is maintained
- Uniquely applied to each document

CSS Selectors

- The selector is used to target elements and apply CSS
- Three simple selectors
    - Element Selector
    - Id Selector
    - Class Selector
- Priority of Selectors

CSS Colors

- There are different colouring schemes in CSS
- **RGB**-This starts with RGB and takes 3 parameter
- **HEX**-Hex code starts with # and comprises of 6 numbers which are further divided into 3 sets
- **RGBA**-This starts with RGB and takes 4 parameter

CSS Background

- There are different ways by which CSS can have an effect on HTML elements
- Few of them are as follows:
    - Color – used to set the color of the background
    - Repeat – used to determine if the image has to repeat or not and if it is repeating then how it should do that
    - Image – used to set an image as the background
    - Position – used to determine the position of the image
    - Attachment – It basically helps in controlling the mechanism of scrolling

CSS BoxModel

- Every element in CSS can be represented using the BOX model
- It allows us to add a border and define space between the content
- It helps the developer to develop and manipulate the element.

CHAPTER 5

FEASIBILITY STUDY

Splitting the dataset

The data use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. It has the test dataset (or subset) in order to test our models and it will do this using Tensor flow library in Python using the Keras method.

Construction of a Detecting Model

Deep learning needs data gathering have lot of past image data's. Training and testing this model working and predicting correctly.

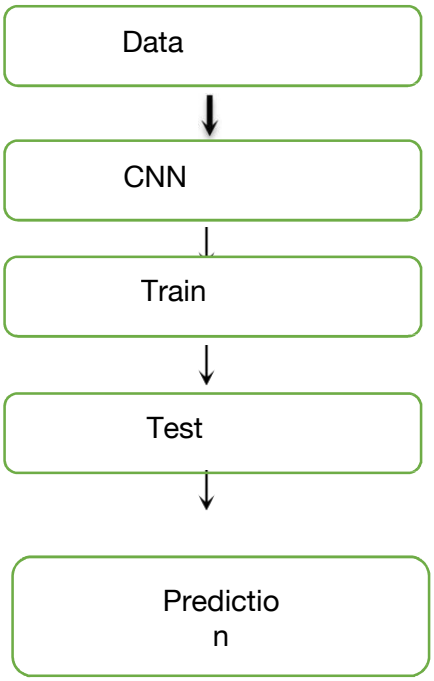


Fig 5.1Steps of dataflow diagram

5.1 Data Flow Diagram

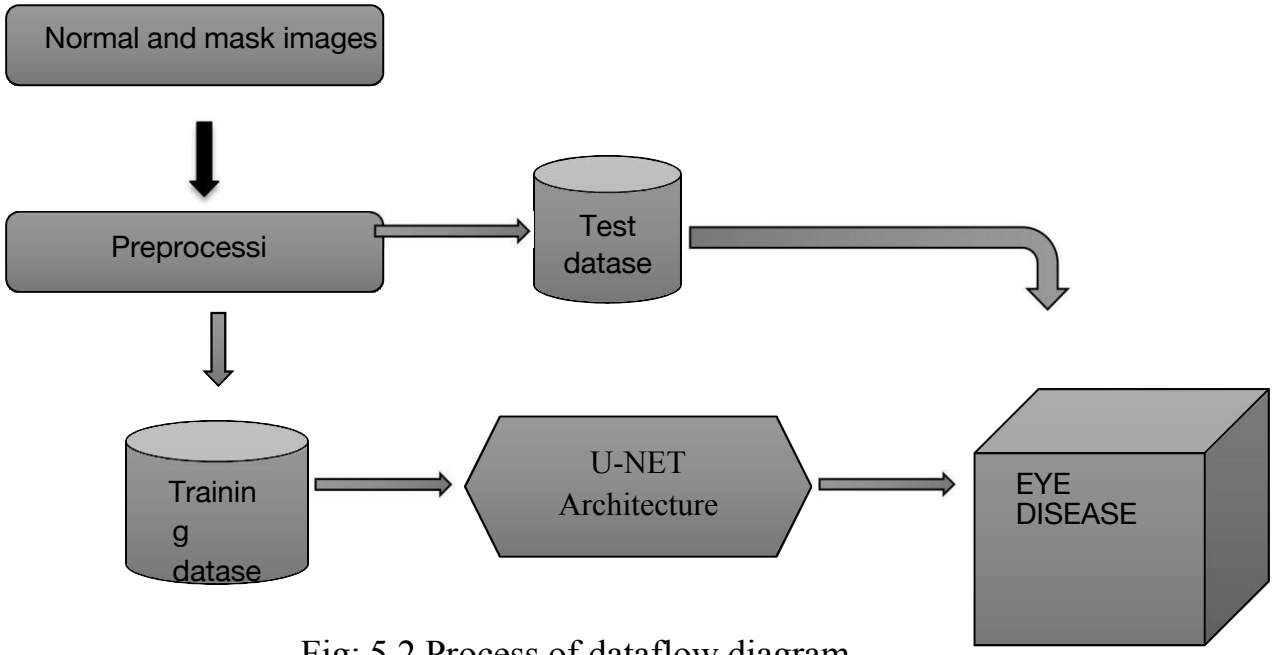


Fig: 5.2 Process of dataflow diagram

5.2 DESIGN ARCHITECTURE

Design is meaningful engineering representation of something that is to be built. Software design is a process design is the perfect way to accurately translate requirements in to a finished software product. Design creates a representation or model, provides detail about software data structure, architecture, interfaces and components that are necessary to implement a

system.

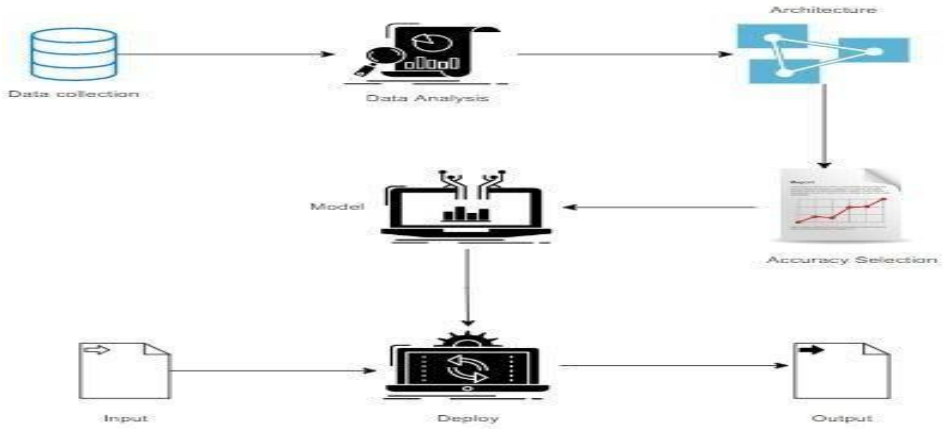


Fig 5.3 Design Architecture

5.3 Work flow diagram

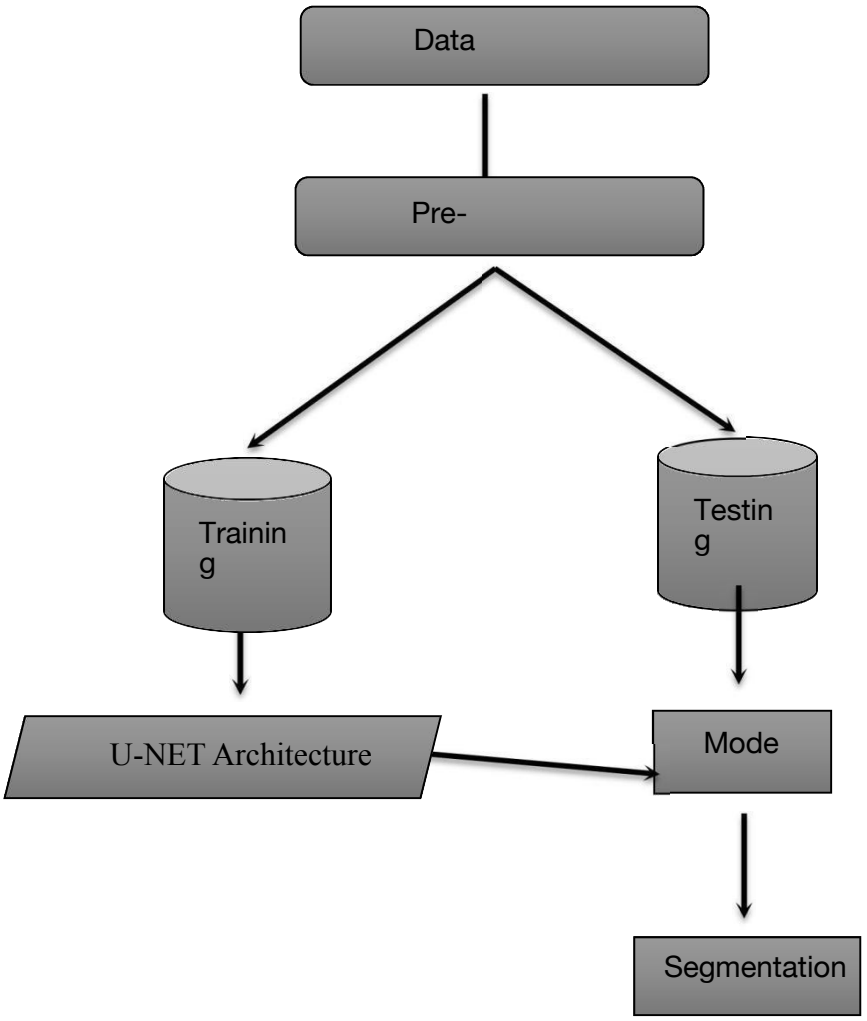


Fig 5.4 Workflow Diagram for Eye Disease

5.4 USECASE DIAGRAM

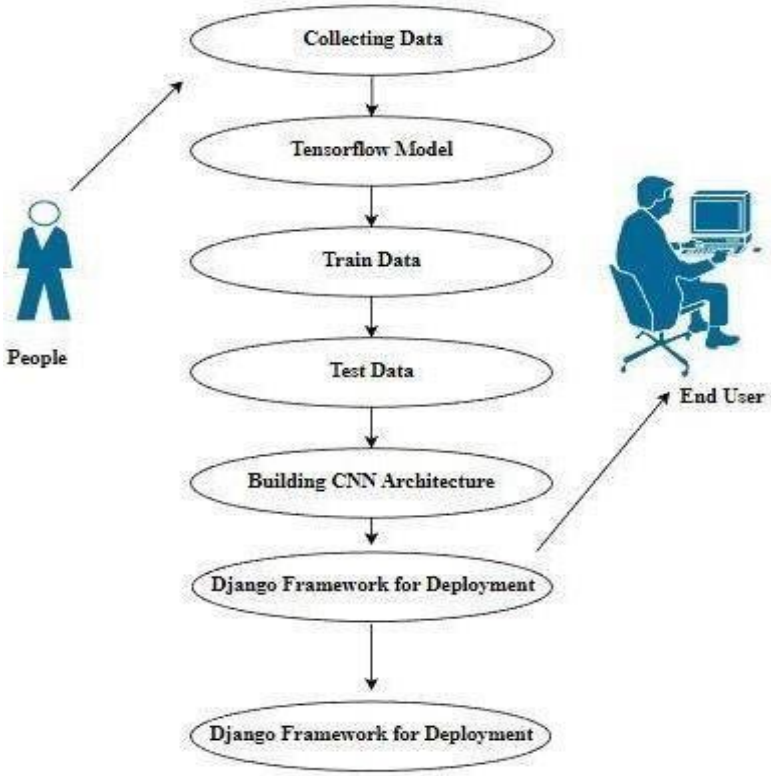


Fig 5.5 Usecase Diagram for Eye Disease

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

5.5 CLASS DIAGRAM

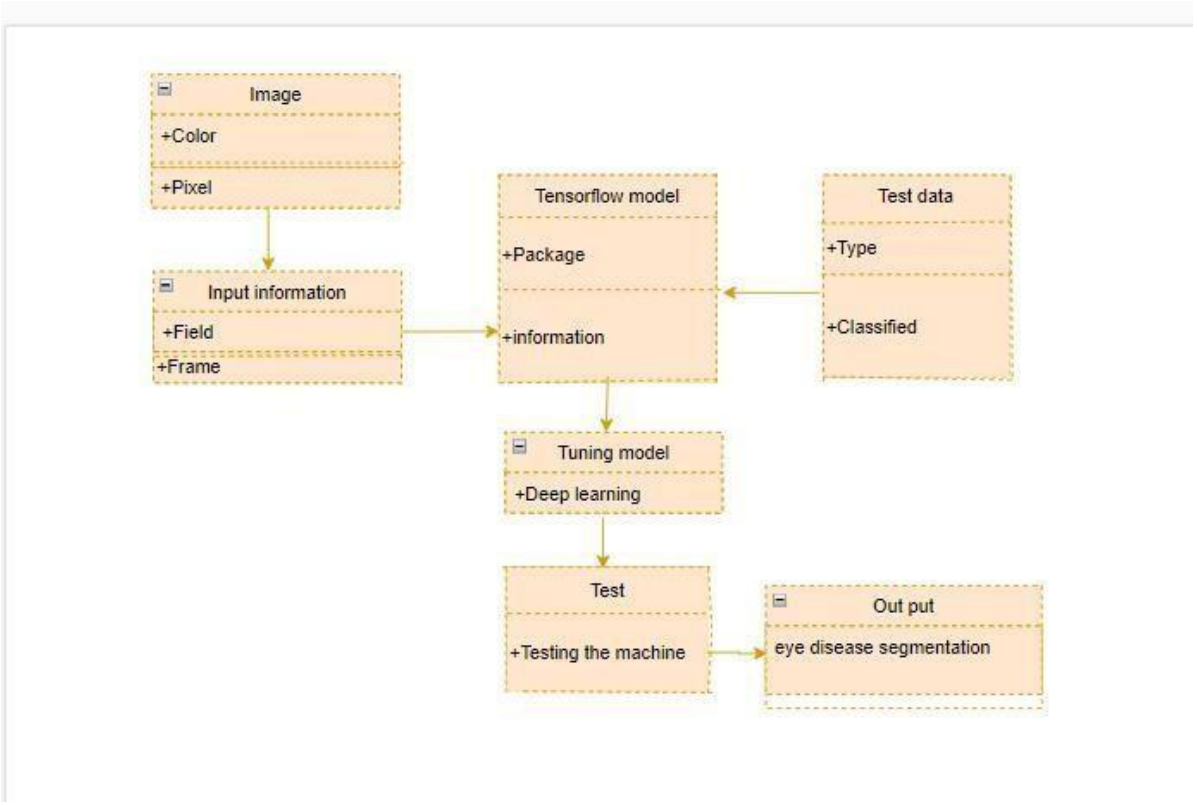


Fig 5.6 Class Diagram for Eye Disease

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before

making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

5.6 ACTIVITY DIAGRAM

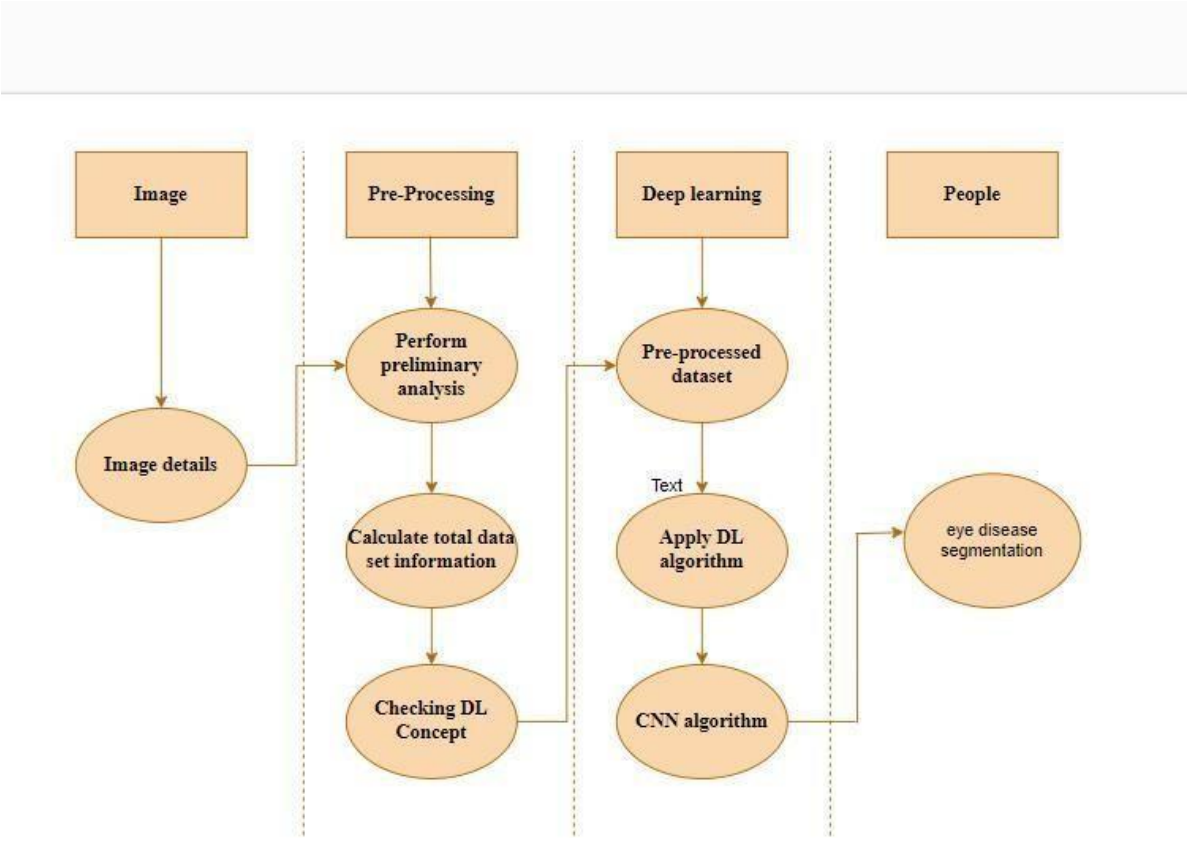


Fig 5.7 ActivityDiagram for Eye Disease

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

5.7 SEQUENCE DIAGRAM

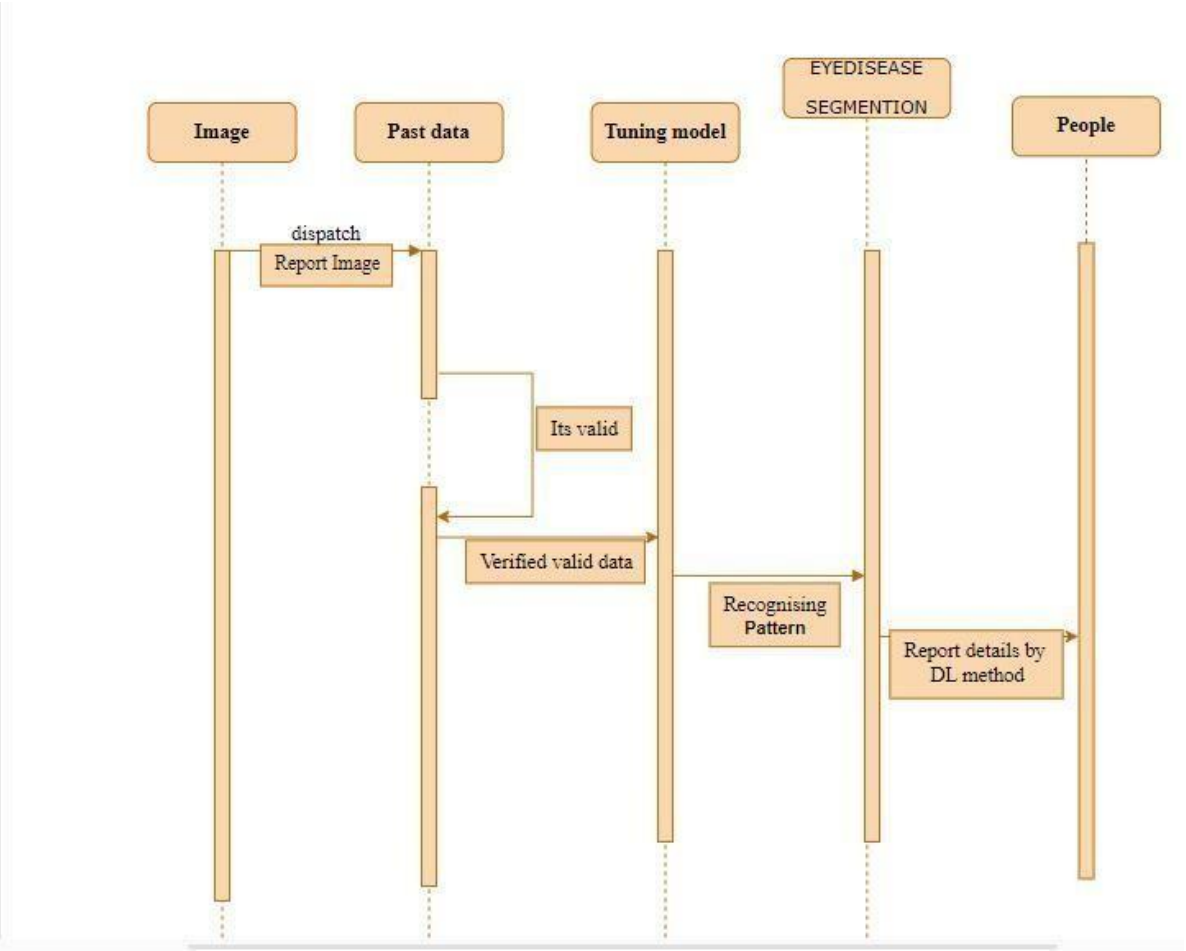


Fig 5.8 Sequence Diagram for Eye Disease

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in

my opinion the most important design-level models for modern business application development.

5.8 ER DIAGRAM

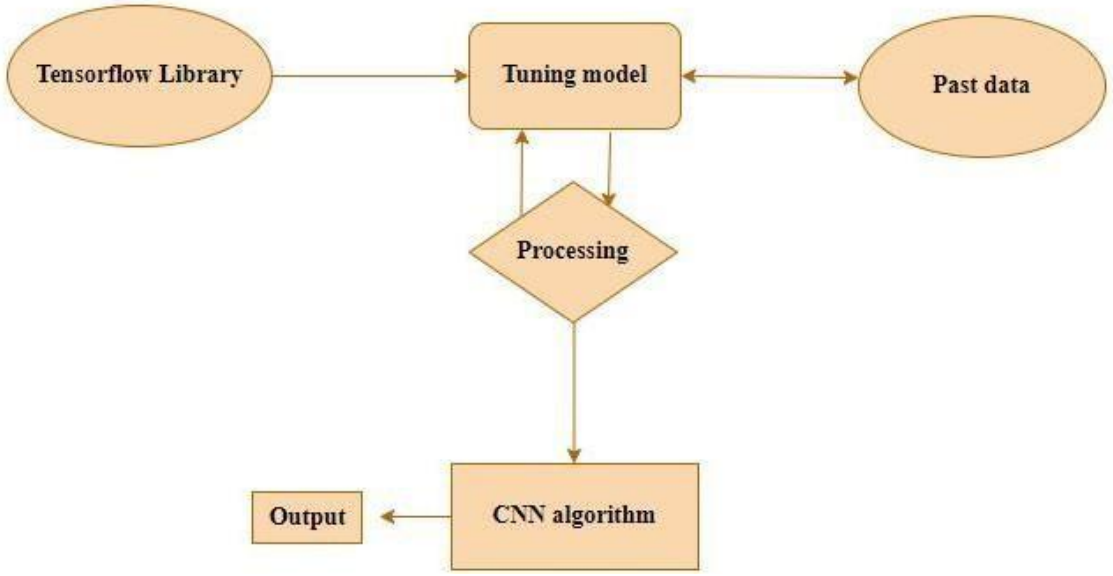


Fig 5.9 ER Diagram for Eye Disease

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

5.9 COLLABORATION DIAGRAM

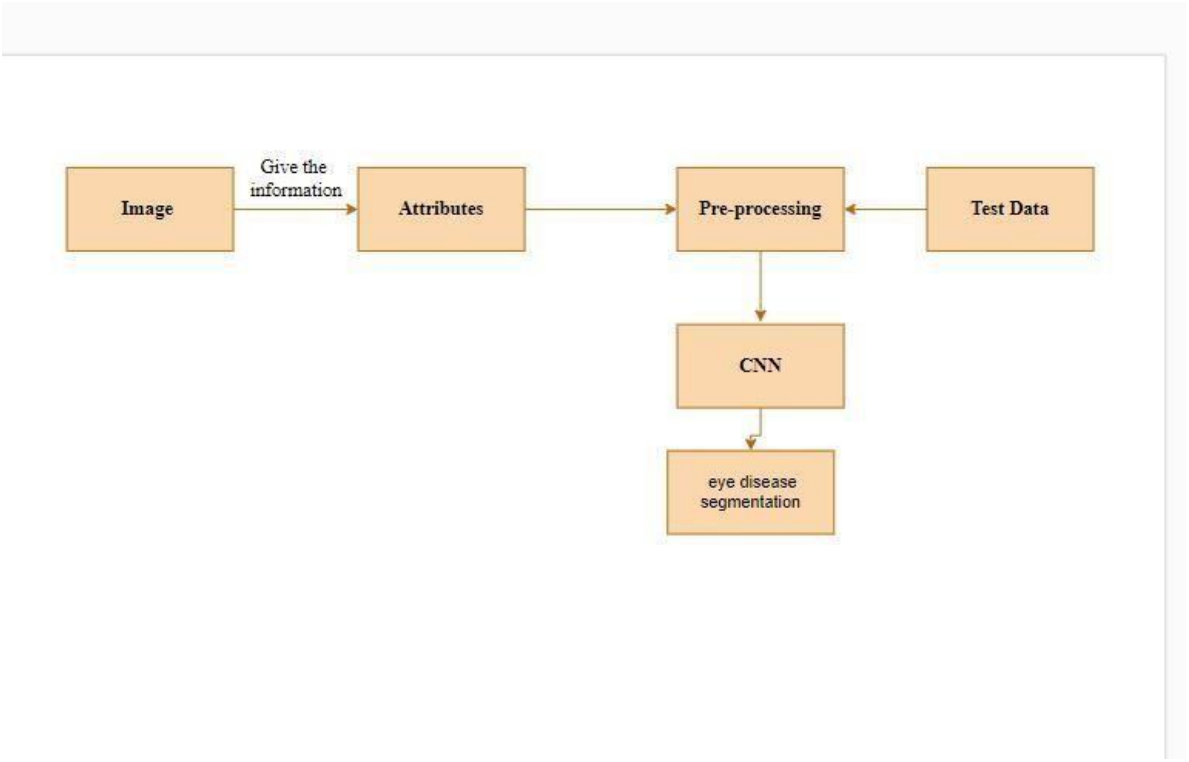


Fig 5.10 Collaboration Diagram for Eye Disease

A collaboration diagram show the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram.it can be the decomposition of a use case, use case diagram, or part of a use case diagram.

The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

CHAPTER 6

SOFTWARE DESCRIPTION

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system "Conda". The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

6.1 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using

command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.
The following applications are available by default in Navigator:

JupyterLab
Jupyter Notebook
Spyder
VSCode
Glueviz
Orange 3 App
RStudio
Anaconda Prompt (Windows only)
Anaconda PowerShell (Windows only)

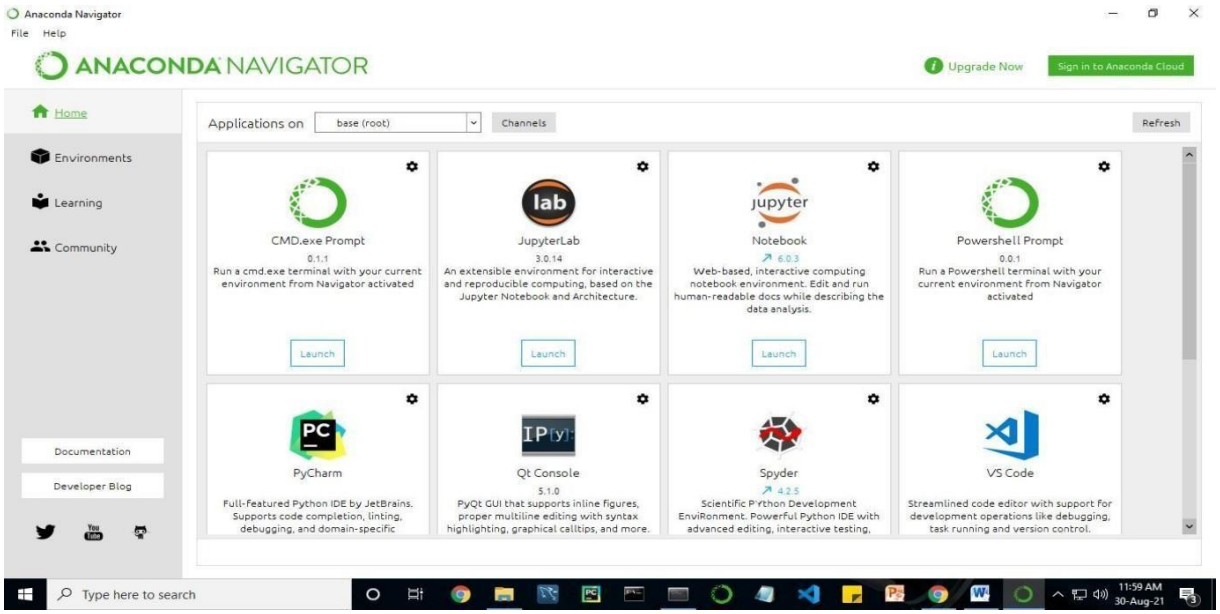
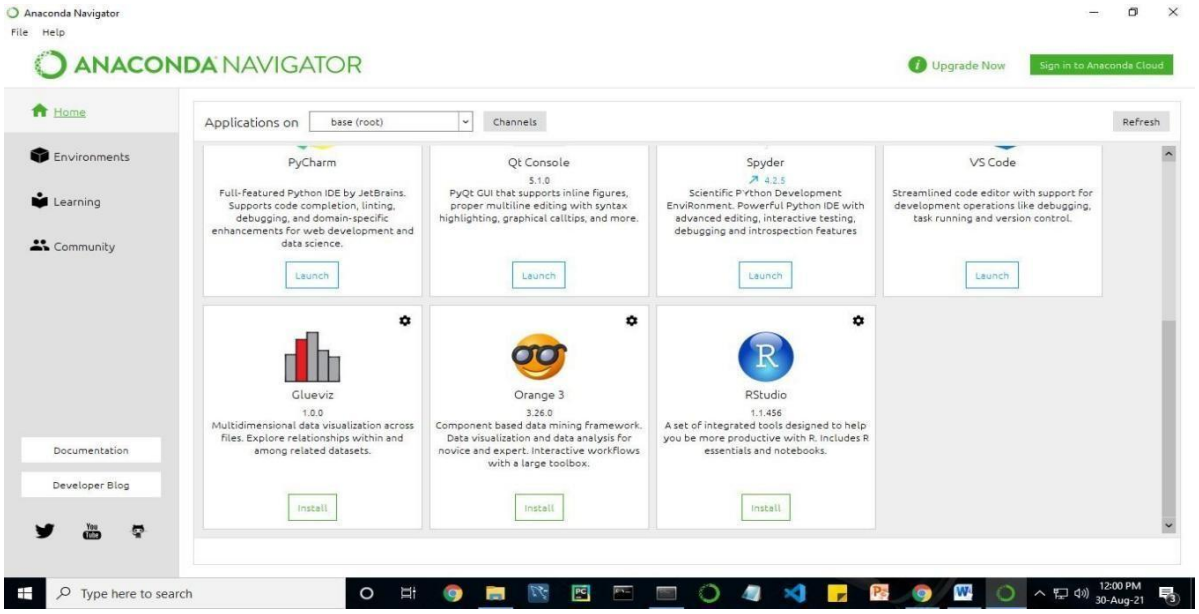


Fig 6.1 Anaconda Nvigator



Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

6.2 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

- Launch the jupyter notebook app
- In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- Click on the menu *Help -> User Interface Tour* for an overview of the Jupyter Notebook App user interface.
- You can run the notebook document step-by-step (one cell a time) by pressing *shift* + *enter*.
- You can run the whole notebook in a single step by clicking on the menu *Cell -> Run All*.
- To restart the kernel (i.e. the computational engine), click on the menu *Kernel -> Restart*. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc…).

Working Process

Download and install anaconda and get the most useful package for machine learning in Python.

Load a dataset and understand its structure using statistical summaries and data visualization.

Machine learning models, pick the best and build confidence that the accuracy is reliable.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

6.3 Visual Studio Code

Visual Studio Code (VS Code) is a versatile and lightweight source code editor that provides a wide range of features for developers. It supports various programming languages and extensions, making it a popular choice for Python development and beyond.

VS Code Features

Intelligent Coding Assistance: VS Code offers intelligent code completion, error checking, and quick fixes to enhance your coding experience. It supports Python and a variety of other languages, providing a smart and configurable editor.

Code Editing: The smart code editor in VS Code includes support for Python, JavaScript, TypeScript, HTML, CSS, and more. It features language-aware

code completion, error detection, and on-the-fly code fixes to boost your productivity.

Smart Code Navigation: Use the built-in search functionality to quickly navigate to classes, files, symbols, and IDE actions. Switching between declarations, references, and implementations is just a click away.

Refactoring Tools: VS Code supports fast and safe refactorings, including renaming, extracting methods, introducing variables, and more. Language-specific refactorings help you make project-wide code changes efficiently.

Built-in Developer Tools: VS Code comes with integrated tools like a debugger, test runner, profiler, terminal, and version control system (VCS) support for Git and others. It also integrates seamlessly with remote development tools, Docker, and more.

CHAPTER 7

RESULT AND DISCUSSION

```
TRAINING DATA FOR CATERACT:

======== Images in:  Dataset/cataract
Images_count :   301
Min_width :      256
Max_width :      256
Min_height :     256
Max_height :     256
```

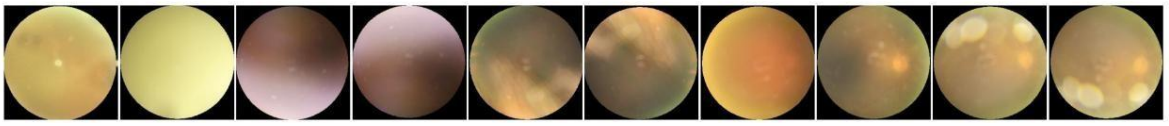


Fig 7.1 Training data for CATERACT

```
TRAINING DATA FOR GLAUCOMA:

======== Images in:  Dataset/glaucoma
Images_count :   300
Min_width :      256
Max_width :      256
Min_height :     256
Max_height :     256
```

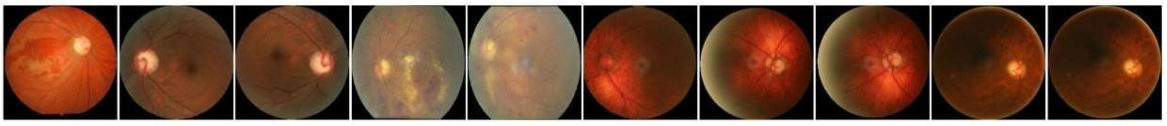


Fig 7.2 Training data for GLAUCOMA

```
TRAINING DATA FOR DIABETIC RETINOPATHY:

======== Images in:  Dataset/diabetic_retinopathy
Images_count :   300
Min_width :      512
Max_width :      512
Min_height :     512
Max_height :     512
```

Fig 7.3 Training data for DIABETIC RETINOPATHY

```
TRAINING DATA FOR SEBORRHEIC NORMAL:

======== Images in:  Dataset/normal
Images_count :   300
Min_width :      512
Max_width :      512
Min_height :     512
Max_height :     512
```

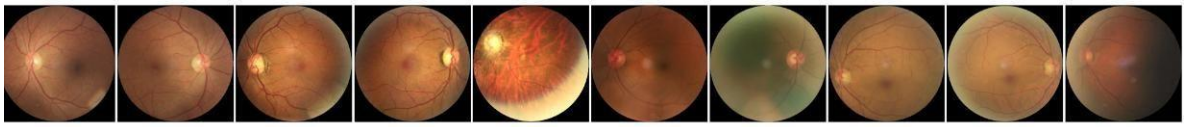


Fig 7.4 Training data for SEBORRHEIC NORMAL

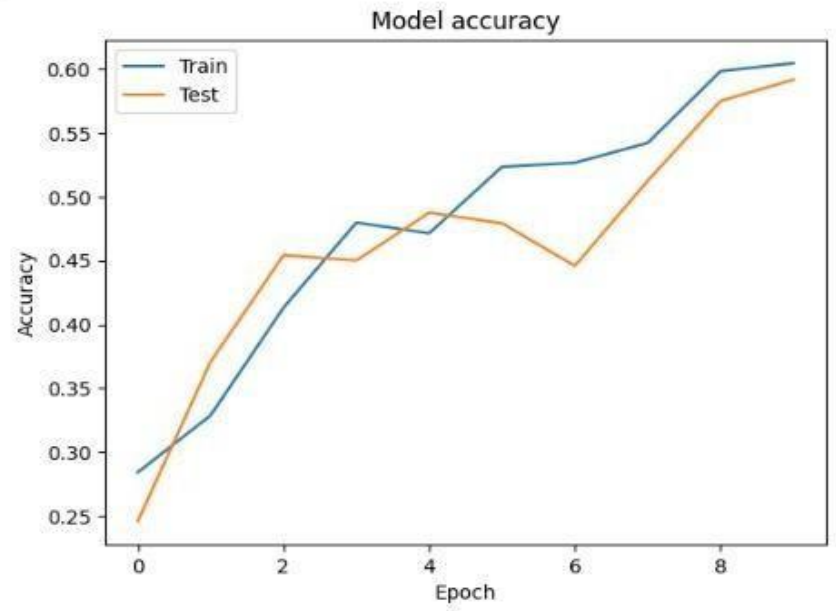
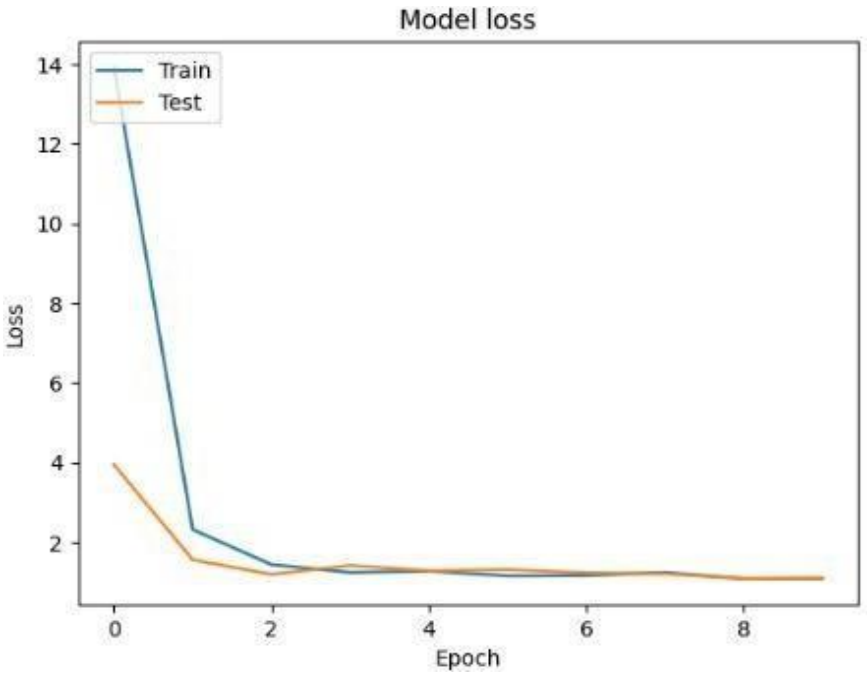


Fig 7.5 Model Accuracy Between Epoch and Accuracy


Model loss
Train
Test
Loss
Epoch
14
12
10
8
6
4
2
0
2
4
6
8

CHAPTER 8
APPENDICES

Module 1

Classification

```
# MANUAL NET ARCHITECTURE
import tensorflow
import tensorflow as tf
print(tf.__version__)

import os
import glob
from PIL import Image
import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Activation
cataract = 'Dataset/cataract'
diabetic_retinopathy = 'Dataset/diabetic_retinopathy'
glaucoma = 'Dataset/glaucoma'
```

```
normal = 'Dataset/normal'
def plot_images(item_dir, n=6):
    all_item_dir = os.listdir(item_dir)
    item_files = [os.path.join(item_dir, file) for file in all_item_dir][:n]

    plt.figure(figsize=(80, 40))
    for idx, img_path in enumerate(item_files):
        plt.subplot(3, n, idx+1)
        img = plt.imread(img_path)
        plt.imshow(img, cmap='gray')
        plt.axis('off')

    plt.tight_layout()

def image_details_print(data,path):
    print('======== Images in: ', path)
    for key,values in data.items():
        print(key,':\t', values)



def images_details(path):
    files=[f for f in glob.glob(path + "**/*.*", recursive=True)]
    data={}
    data['Images_count']=len(files)
    data['Min_width']=10**100
    data['Max_width']=0
    data['Min_height']=10**100
    data['Max_height']=0
```

```
    for f in files:
        img=Image.open(f)
        width,height=img.size
        data['Min_width']=min(width,data['Min_width'])
        data['Max_width']=max(width, data['Max_width'])
        data['Min_height']=min(height, data['Min_height'])
        data['Max_height']=max(height, data['Max_height'])

    image_details_print(data,path)
print("")
print("TRAINING DATA FOR CATERACT:")
print("")
images_details(cataract)
print("")
plot_images(cataract, 10)
print("")
print("TRAINING DATA FOR DIABETIC RETINOPATHY:")
print("")
images_details(diabetic_retinopathy)
print("")
plot_images(diabetic_retinopathy, 10)
print("")
print("TRAINING DATA FOR GLAUCOMA:")
print("")
images_details(glaucoma)
print("")
plot_images(glaucoma, 10)
print("")
print("TRAINING DATA FOR SEBORRHEIC NORMAL:")
```

```
print("")
images_details(normal)
print("")
plot_images(normal, 10)
# Define the path to your dataset directory
dataset_dir = 'Dataset'

# Set the batch size and number of classes
batch_size = 64
num_classes = 4
# Create an ImageDataGenerator for data augmentation and preprocessing

from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    validation_split=0.2)
# Load and preprocess the dataset using the ImageDataGenerator

train_generator =
    datagen.flow_from_directory(dataset_dir,
    target_size=(224, 224),
    batch_size=batch_size,
```

```
    class_mode='categorical',
    subset='training')
validation_generator =
    datagen.flow_from_directory(dataset_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')

MODEL=Sequential()
MODEL.add(Convolution2D(32,(3,3),input_shape=(224,224,3),activation='rel
u'))
MODEL.add(MaxPooling2D(pool_size=(2,2)))
MODEL.add(Flatten())
MODEL.add(Dense(4, activation='softmax'))
MODEL.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics
=['accuracy'])
epochs = 10
history =
    MODEL.fit(train_g
    enerator,
    validation_data=validation_generator,
    epochs=epochs # Add the callbacks here
)
import matplotlib.pyplot as plt

def graph():
    #Plot training & validation accuracy values
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
```

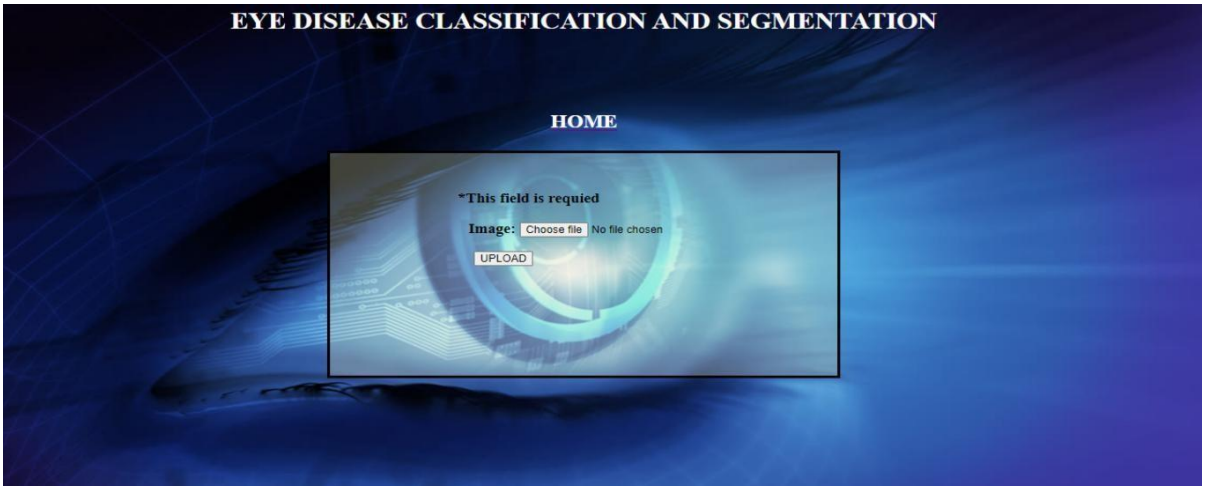
```
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

graph()
import matplotlib.pyplot as plt

def graph():
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
graph()
```

Output Screenshot


ARTIFICIAL INTELLIGENCE
TEAM MEMBERS
DOMAIN_RESULT
PROBLEM STATEMENTS
LOGOUT
TESTING
FOR HEALTH CARE DOMAIN
EYE DISEASE CLASSIFICATION AND SEGMENTATION
PERSONAL INFORMATION
DEPLOYMENT
PERSONAL_INFO_DATABASE
EYE DISEASE
NORMAL
CATARACT
EYE DISEASE


EYE DISEASE CLASSIFICATION AND SEGMENTATION
HOME
*This field is requied
Image:
Choose file
No file chosen
UPLOAD


*This field is requied
Image:
Choose file
No file chosen
UPLOAD
THE CATARACT TYPE OF EYE DISEASE DETECTED IN THIS IMAGE.
ORIGINAL_IMAGE
MASK IMAGE

CHAPTER 9
CONCLUSION AND FUTURE WORK

CONCLUSION

In conclusion, the application of artificial intelligence techniques in the diagnosis of eye fundus diseases through classification and segmentation has shown promising results in enhancing the efficiency and accuracy of medical assessments. The use of advanced algorithms and machine learning models has facilitated the automatic identification and categorization of various eye fundus pathologies, providing valuable support to healthcare professionals. These AI-driven approaches have the potential to streamline the diagnostic process, enabling early detection and intervention, thereby improving patient outcomes. However, ongoing research and validation studies are crucial to ensuring the robustness and reliability of these techniques in real-world clinical settings, and collaborative efforts between the medical and AI communities are essential for the continued development and implementation of effective tools for eye fundus disease diagnosis.

FUTURE WORK

In the realm of diagnosing eye fundus diseases, the integration of advanced artificial intelligence (AI) techniques holds great promise for future developments. The ongoing exploration of deep learning models, particularly convolutional neural networks (CNNs), offers an avenue for enhancing the accuracy and efficiency of disease classification and segmentation in eye fundus images. Future work could focus on refining existing models to handle a broader spectrum of pathologies and ensuring robustness across diverse datasets. Additionally, exploring novel architectures that incorporate multi-modal information, such as combining imaging and patient clinical data, may further improve diagnostic capabilities. Addressing challenges related to

interpretability and the integration of AI systems into clinical workflows will be crucial for facilitating the adoption of these technologies in real-world medical settings. Furthermore, continuous efforts in collecting large and diverse datasets, along with collaborative initiatives between clinicians and AI researchers, will be essential for training models that generalize well and contribute to the evolution of reliable and accessible tools for the early detection and management of eye fundus diseases.

CHAPTER 10

REFRENCES

[1] Yaroub Elloumi, Mohamed Akil, Henda Boudegga, Ocular Diseases Diagnosis in Fundus Images using a Deep Learning: Approaches, tools and Performance evaluation , 2019

[2] Sarki, Rubina, Ahmed, Khandakar, Wang, Hua and Zhang, Automatic Detection of Diabetic Eye Disease Through Deep Learning Using Fundus Images , 2020

[3] Sadaf Malik , Nadia Kanwal , Mamoona Naveed Asghar , Data Driven Approach for Eye Disease Classification with Machine Learning ,2019

[4] Eman Hassan Hagar , Detection of glaucoma using artificial intelligence in fundus image: A narrative review , 2023

[5] Rachana Devanaboina, Sreeja Badri, Madhuri Reddy Depa, Dr.Sunil Bhutad, OCULAR EYE DISEASE PREDICTION USING MACHINE LEARNING, 2021

[6] Feng Li, Yuguang Wang, Tianyi Xu, Lin Dong, Lei Yan, Minshan Jiang, Xuedian Zhang, Hong Jiang, Zhizheng Wu & Haidong Zou , Deep Learning-Based Automatic Detection System of Diabetic Retinopathy Using Fundus Images: A Review, 2021

[7] Muhammad Mateen, Junhao Wen, Mehdi Hassan,Nasrullah Nasrullah, Machine Learning Techniques for Automatic Detection of Diabetic Retinopathy: A Review, 2020

[8] Carson Lam, Darvin Yi, Margaret Guo and Tony Lindsey, Review on Automatic Detection of Diabetic Retinopathy Using Deep Learning Techniques, 2018

[9] Vasudevan Lakshminarayanan ,Hoda Kheradfallah ,Arya Sarkar andJanarthanam Jothi Balaji ,Automated Detection and Diagnosis of Diabetic Retinopathy: A Comprehensive Survey, 2021

[10] Amsa Shabbir, Aqsa Rasheed, Detection of glaucoma using retinal fundus images: A comprehensive review , 2021

[11] Qaisar Abbas ,Mubarak Albathan ,Abdullah Altameem ,Riyad Saleh Almakki andAyyaz Hussain , Deep-Ocular: Improved Transfer Learning Architecture Using Self-Attention and Dense Layers for Recognition of Ocular Diseases, 2023

[12] Gauri Ramanathan, Diya Chakrabarti, Aarti Patil, Sakshi Rishipathak, Eye Disease Detection Using Machine Learning , 2021

[13] Abdul Rahaman Wahab Sait, Artificial Intelligence-Driven Eye Disease Classification Model, 2023

[14] Muhammad Waqas Nadeem,Hock Guan Goh, Muzammil Hussain,Soung-Yue Liew,Ivan Andonovic and Muhammad Adnan Khan, Deep Learning for Diabetic Retinopathy Analysis: A Review, Research Challenges, and Future Directions, 2022

[15] Shyamala Subramanian ,Sashikala Mishra ,Shruti Patil ,Kailash Shaw and Ebrahim Aghajari **,** Machine Learning Styles for Diabetic Retinopathy Detection: A Review and Bibliometric Analysis, 2022