

RETINAL IMAGE PROCESSING USING NEURAL NETWORK WITH DEEP LEARNING

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in Information Technology

by

**SANTHOSH KUMAR M (Reg No: 38120075)
SHIVA M (Reg No: 38120078)**



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119

MARCH - 2022



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI– 600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SANTHOSH KUMAR M (REG NO: 38120075) SHIVA M (REG NO: 38120078)** who have done the Project work as a team who carried out the project entitled “**RETINAL IMAGE PROCESSING USING NEURAL NETWORK WITH DEEP LEARNING**” under my supervision from November 2021 to April 2022.

Internal Guide
Dr.V. MARIA ANU M.E., Ph.D.,

Head of the Department
Dr.R. SUBHASHINI M.E., PH.D.,

Submitted for Viva Voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **SANTHOSH KUMAR M (REG NO: 38120075) SHIVA M (REG NO: 38120078)** hereby declare that the Project Report entitled “**RETINAL IMAGE PROCESSING USING NEURAL NETWORK WITH DEEP LEARNING**” done by us under the guidance of **Dr.V. MARIA ANU M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr.R.Subhashini M.E., Ph.D. , Head** of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.V.Maria Anu M.E., Ph.D.**, his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

This study aims to develop a system to distinguish retinal disease from fundus images. Precise and programmed analysis of retinal images has been considered as an effective way for the determination of retinal diseases such as diabetic retinopathy, hypertension, arteriosclerosis, etc. In this work, we extracted different retinal features such as retinal, optic disc and lesions and then applied convolutional neural network-based models for the detection of multiple retinal diseases with fundus photographs involved in structured analysis of the retina (STARE) database. It described the innovative solution that provides efficient disease detection and deep learning with convolutional neural networks (CNNs) has achieved great success in the classification of various retinal diseases. A variety of neuron-wise and layer-wise visualization methods were applied using a CNN, trained with a publicly available retinal disease given image dataset. So, it observed that neural networks can capture the colors and textures of lesions specific to respective diseases upon diagnosis, which resembles human decision-making. And this model to deploy Django web framework. We experimented with different retinal features as input to convolutional neural networks for effective classification of retinal images.

TABLE OF CONTENTS

CHAPTER NO:	TITLE	PG NO:
1	INTRODUCTION	1
	1.1 DEEP LEARNING	1
	1.2 MACHINE LEARNING	3
	1.3 OBJECTIVES	3
	1.4 EXISTING SYSTEM	3
	1.4.1 Drawbacks	3
	1.5 PROPOSED SYSTEM	4
	1.5.1ADVANTAGES	4
2	LITERATURE SURVEY	5
3	AIM AND SCOPE OF THE MODEL	8
	3.1 AIM	8
	3.2 SCOPE OF THE PROJECT	8
4	METHODOLOGY AND IMPEMETATION	9
	4.1 ENVIRONMENT AND REQUIREMENTS	9
	4.2 SPLITTING THE DATASETS	9
	4.3 CONSTRUCTION OF DETECTING MODEL	9
	4.4 DATA FLOW DIAGRAM	10
	4.5 SYSTEM ARCHITECTURE	12
	4.6 WORK FLOW DIAGRAM	13
	4.7 USECASE DIAGRAM	14
	4.8 CLASS DIAGRAM	14
	4.9 ACTIVITY DIAGRAM	15
	4.10 ER DIAGRAM	16
	4.11 ANACONDA NAVIGATTOR	17
	4.11.1 JUPYTER NOTEBOOK	18
	4.11.2 WORKING PROCESS	18
	4.11.3 PYCHARM	19
	4.12 METHODOLOGY	19
	4.13 CNN MODEL STEPS	20
	4.13.1 Conv2d	20
	4.13.2 Max Pooling 2D layer	20
	4.13.3 Arguments	21
	4.13.4 Input shape	21
	4.13.5 Output shape	21
	4.13.6 Flatten layer	21
	4.13.7 Dense layer	22
	4.13.8 Arguments	22
	4.13.9 Input shape	23
	4.13.10 Output Shape	23
	4.13.11 Dropout layer	23
	4.13.12 Arguments	23
	4.13.13 Image Data Generator	24
	4.13.14 Training Process	24
	4.13.15 Epoches	24
	4.13.16 Validation Process	24

4.14	ARTIFICIAL NEURAL NETWORK	24
4.14.1	Key points related to the Architecture	24
4.14.2	Key Advantages of Neural Network	25
4.15	ARCHITECTURE OF CNN	25
4.15.1	CONVOLUTION NEURAL NETWORK	25
4.15.2	Model API	26
4.15.3	Types of Keras Models	26
4.15.4	Model Subclassing in Keras	26
4.16	Rectified Linear Unit (ReLU)	28
4.16.1	Normalisation	29
4.16.2	During Training	29
4.16.3	During Inference	29
4.16.4	Arguments	30
4.17	TYPES OF CNN	30
4.17.1	ALEXNET	31
4.17.2	Architecture of Alex Net	31
4.18	LENET	32
4.18.1	Architecture of Le Net-5	32
4.19	MODULE DISCRIPTIONS	34
4.19.1	IMPORT THE GIVEN IMAGE FROM DATASET	34
4.19.2	TO TRAIN THE MODULE BY GIVEN IMAGE DATASET	35
4.10.3	WORKING PROCESS OF LAYER IN CNN MODEL	36
4.20	DEPLOY	39
4.20.1	Deploying the model in Django Framework and predicting output	39
4.20.2	Django	39
4.20.3	HTML	39
4.20.4	CSS	40
5	RESULT AND FUTURE WORK	41
5.1	RESULT	41
5.2	Future Work	42
	CONCLUTION	43
6.1	Summary	43
6.2	APPENDIX	43
	REFERENCE	47
	Publication Report	49
	IEEE COPYRIGHT AND CONSENT FORM	50
	Acceptance Letter of the IEEE Conference	53

LIST OF FIGURES

FIG.NO	TITLE	PAGE.NO
4.2	DATA FLOW DIAGRAM	10
4.7	SYSTEM ARCHITECTURE	12
4.8	WORKFLOW DIAGRAM	13
4.9	USECASE DIAGRAM	14
4.10	CLASS DIAGRAM	14
4.11	ACTIVITY DIAGRAM	15
4.12	ER – DIAGRAM	16

LIST OF ABBREVIATION:

CNN – Convolution Neural Network
ANN – Artificial Neural Network
ReLU- Rectified Linear Unit

CHAPTER 1

1.INTRODUCTION.

1.1 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 brain so deep learning is also a kind of mimic of human brain. 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 brain 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.

1.2 MACHINE LEARNING:

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning

dynamically interacts with its environment and it receives positive or negative feedback to improve its performance. Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y).



Fig 1.1 Process of Machine Learning

Supervised Machine Learning **is the** majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output **is $y = f(X)$** . The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include **logistic regression, multi-class classification, Decision Trees and support vector machines etc.** Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into **Classification** problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

1.3 OBJECTIVE:

1. The goal is to develop a deep learning model for Retinal image classification by convolutional neural network algorithm.
2. For potentially classifying the results in the form of best accuracy by comparing the CNN architectures

1.4 EXISTING SYSTEM:

They proposed for modality specific attention network (MSAN) that used fundus and OCT images simultaneously to diagnose ophthalmology, the noise in the OCT image may disturb the ROI extraction, they implement a Gaussian filter with the kernel size for the OCT image denoising and they introduced a deep learning method for fundus image classification based on retinal nerve fiber layer detecting. The anatomy-based deep learning methods they have to detect or segment anatomy for training deep learning models and accuracy of anatomy extracting affects the performance of classification and analyzing the quantitative performance of our proposed MSAN, It is necessary to investigate whether different attention modules work as expected, their Experimental results on a clinically acquired multi-modal retinal image dataset demonstrate that MSAN outperforms other well-known single-modal and multi-modal retinal image classification methods. Deep learning-based approach is presented for the early detection of diabetic retinopathy from retinal images.

1.4.1 Drawbacks:

1. They are not using AlexNet CNN and LeNet CNN for reducing errors
2. They are not considering the relation between different diseases, and they project cost and time is heavy.
3. They are not using any model deployment process.

1.5 PROPOSED SYSTEM:

The proposed approach consists stage pretreatments were performed in retinal images from data sets and standardize them to size then classification was made by Convolutional Neural Network which is a deep learning algorithm and success was achieved to deep learning technique so that a person with lesser expertise in software should also be able to use it easily. It proposed system to predicting retinal disease. It explains about the experimental analysis of Samples of images are collected that comprised of different retinal. The primary attributes of the image are relied upon the shape and texture-oriented features. An efficient disease detection and deep learning with convolutional neural networks (CNNs) has achieved great success in the classification of various retinal diseases. A variety of neuron-wise and layer-wise visualization methods were applied using a CNN, trained with a publicly available retinal disease given image dataset. The sample screenshots display the retinal disease detection using color-based classification model. And to deploy this model in web application for Django framework.

1.5.1 ADVANTAGES:

1. Increasing throughput & reducing subjectiveness arising from human experts in detecting the retinal decease.
2. It is essential to detect a particular disease. In our country many farmers are not so educated to get correct information about all diseases.
3. The below Activation Functions are used in the Modules
4. ReLu - The goal of using the rectifier function is to make our photos more non-linear. We want to do this since photos are naturally non-linear.
5. SoftMax - SoftMax classifiers calculate the probability of each class label, whereas hinge loss calculates the margin. We humans find it far easier to comprehend probabilities than margin scores.

CHAPTER 2

LITERATURE SURVEY

Title : Retinal image preprocessing, enhancement, and registration

Author: Carlos Hernandez-Matasa, Antonis A. Argyrosa,b , Xenophon Zabulisa

Year : 2018

The first fundus images were acquired after the invention of the ophthalmoscope. The concept of storing and analyzing retinal images for diagnostic purposes exists ever since. The first work on retinal image processing was based on analog images and regarded the detection of vessels in fundus images with fluorescein [1]. The fluorescent agent enhances the appearance of vessels in the image, facilitating their detection and measurement by the medical professional or the computer. However, fluorescein angiography is an invasive and time-consuming procedure and is associated with the cost of the fluorescent agent and its administration. Digital imaging and digital image processing have proliferated the use of retinal image analysis in screening and diagnosis. The ability to accurately analyze fundus images has promoted the use of noninvasive, fundus imaging in these domains. Moreover, the invention of new imaging modalities, such as optical coherence tomography (OCT) and scanning laser ophthalmoscopy (SLO), has broadened the scope and applications of retinal image processing. This review regards both fundus imaging, as implemented by fundus photography and SLO and OCT imaging.

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

Author: RUBINA SARKI, KHANDAKAR AHMED, (Senior Member, IEEE), HUA WANG, (Member, IEEE), AND YANCHUN 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 : Retinal Image Analysis Based on Deep Learning

Author: Baidaa Al-Bander

Year : 2018

The interpretation of ophthalmic images is typically performed by trained clinical experts. However, due to the volume and complexity of these images, and the large variation in pathology, in addition to the variation among experts, there has been increasing interest in computer-assisted assessment and diagnosis of such images. There has been particular interest in finding a cost-effective approach with high sensitivity and specificity, independent of human intervention, and robust enough to be applied to large populations in a timely manner to identify retinal diseases. This thesis introduces novel deep learning methodologies based on convolutional neural networks (CNNs) to address key challenges in different retinal image analysis tasks. Three retinal image analysis objectives have been considered in this research project: fovea and optic disc (OD) localisation, choroid and optic disc/cup segmentation, and disease and lesion classification tasks. In the first retinal image analysis task, simultaneous detection of the centres of the fovea and the optic disc from colour fundus images is considered as a regression problem.

Title : Classification of Diabetic Retinopathy Images by Using Deep Learning Models

Author: Suvajit Dutta, Bonthala CS Manideep, Syed Muzamil Basha, Ronnie D. Caytiles¹ and N. Ch. S. N. Iyengar²

Year : 2018

Diabetes or more precisely Diabetes Mellitus (DM) is a metabolic disorder happens because of high blood sugar level in the body. Over the time, diabetes

creates eye deficiency also called as Diabetic Retinopathy (DR) causes major loss of vision. The symptoms can originate in the retinal area are augmented blood vessels, fluid drip, exudates, hemorrhages, and micro aneurysms. In modern medical science, images are the indispensable tool for precise diagnosis of patients. In the meantime evaluation of contemporary medical imageries remains complex. In recent times computer vision with Deep Neural Networks can train a model perfectly and level of accuracy also will be higher than other neural network models. In this study fundus images containing diabetic retinopathy has been taken into consideration. The idea behind this paper is to propose an automated knowledge model to identify the key antecedents of DR. Proposed Model have been trained with three types, back propagation NN, Deep Neural Network (DNN) and Convolutional Neural Network (CNN) after testing models with CPU trained Neural network gives lowest accuracy because of one hidden layers whereas the deep learning models are out performing NN. The Deep Learning models are capable of quantifying the features as blood vessels, fluid drip, exudates, hemorrhages and micro aneurysms into different classes. Model will calculate the weights which gives severity level of the patient's eye. The foremost challenge of this study is the accurate verdict of each feature class thresholds.

Title : Retinal image preprocessing, enhancement, and registration

Author: Carlos Hernandez-Matasa , Antonis A. Argyrosa,b , Xenophon Zabulisa

Year : 2018

The diabetes retinopathy is the application of medical image processing. The retinal images are evaluated to diagnose the DR. It is however, time consuming and resource demanding to manually grade the images such that the severity of DR can be defined. When the tiny blood vessels present within the retina are damaged, only then can one notice this problem. Blood will flow from this tiny blood vessel and features are formed from the fluid that exists on retina. The kinds of features involved here due to the leakage of fluid and blood from the blood vessels are considered to be the most important factors to study this problem.

CHAPTER 3

AIM AND SCOPE OF THE MODEL

3.1 AIM:

- Load the data
- Loading the given dataset
- Import required libraries packages
- Pre-process the data
- Reshape, data augmentations
- Define model
- Sequential or Functional
- Number of layers to be used, Number of nodes to be used in the model, Evaluation metrics
- Compile the model
- Define loss function, optimizer, weights and bias
- Fit the model
- Train data, Test data, epoch, Batchsize.

3.2 SCOPE OF THE PROJECT:

Most of the related works classifies the disease with two classifiers, i.e. whether he/she has a healthy eye or has a diabetic retinopathy defective eye. The scope in our research is to split this class into defective and non-defective stages of diabetic retinopathy. An eye can be classified into 4 classes. Besides, extracting hand crafted features from raw images after different processing is one of the best scopes we have worked in this research. The objective of using this method is to get some features which are specifically responsible for the diabetic retinopathy.

CHAPTER 4

METHODOLOGY AND IMPLEMENTATION

4.1 ENVIRONMENTAL REQUIREMENTS:

1. Software Requirements:

Operating System : Windows / Linux
Simulation Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III
Hard disk : minimum 80 GB
RAM : minimum 2 GB

4.2 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.

4.3 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.

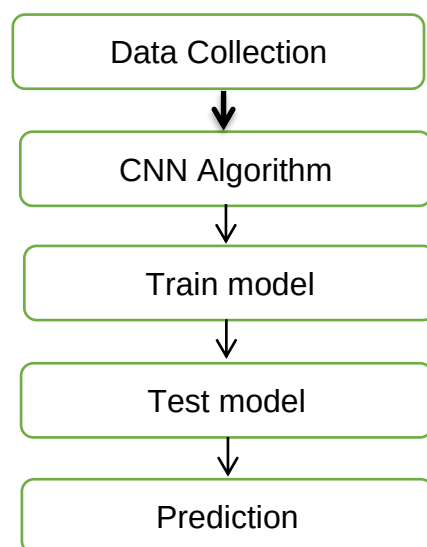


Fig.4.1 Steps of dataflow diagram

4.4 DATA FLOW DIAGRAM:

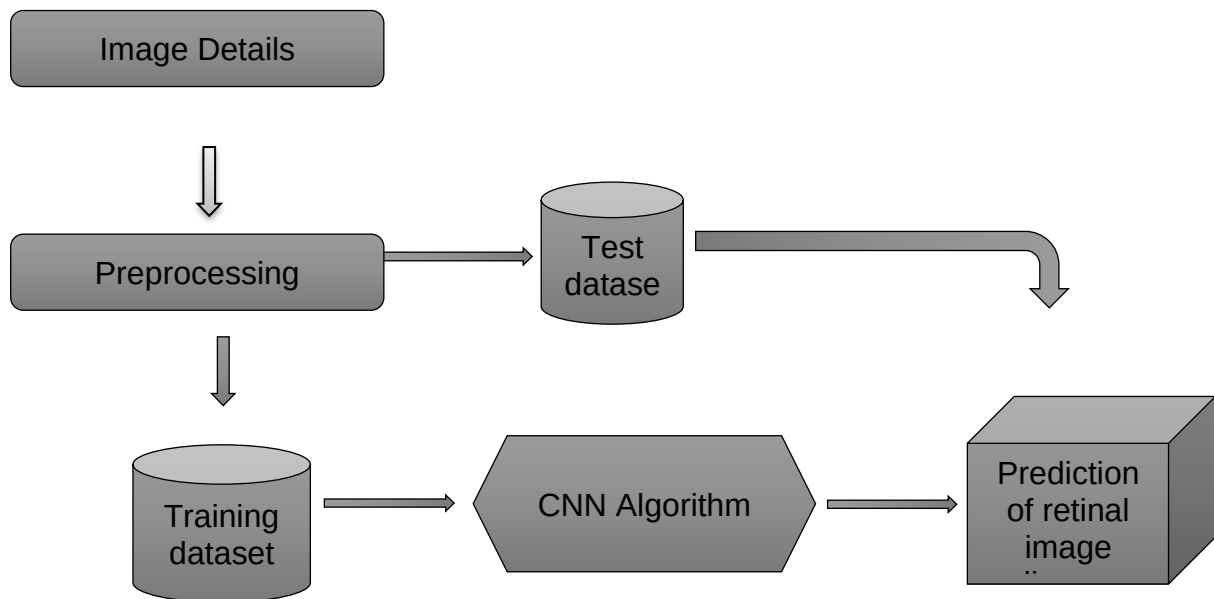


Fig.4.2 Process of dataflow diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to Systems Design.

Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

Data store: files or repositories that hold information for later use, such as a database table or a membership form.

Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like "Billing details."

DFD levels and layers A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond.

Level 0:

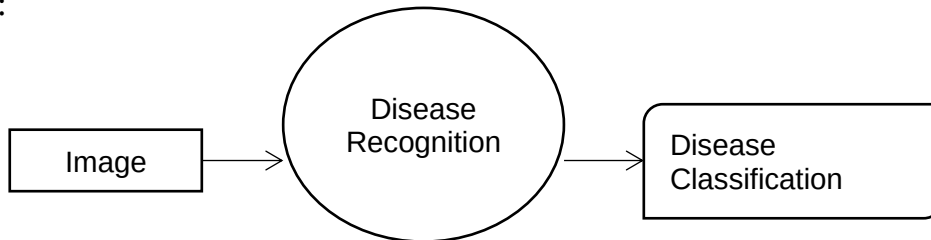


Fig 4.3 Level 0 DFD

The necessary level of detail depends on the scope of what you are trying to accomplish. DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled.

Level 1:

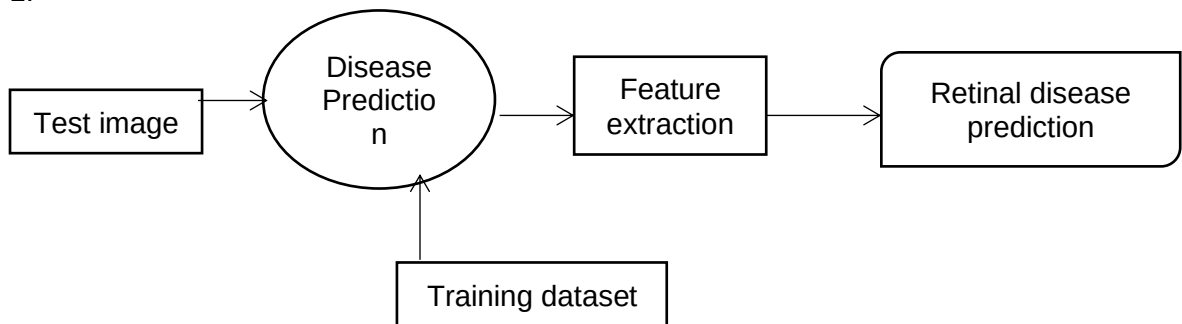


Fig 4.4 Level 1 DFD

It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.

Level 2:

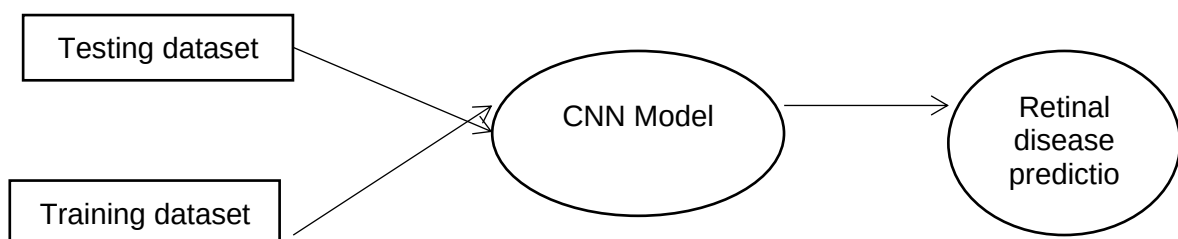


Fig 4.5 Level 2 DFD

It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

Level 3:

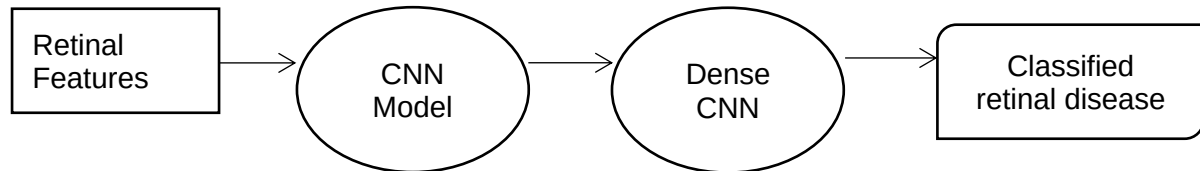


Fig 4.6 Level 3 DFD

4.5 SYSTEM ARCHITECTURE:

The presented method consists of stage pretreatments in retinal images from data sets and size standardization, then classification using Convolutional Neural Network, a deep learning algorithm, and success was achieved to deep learning technique so that a person with less expertise in software could easily use it

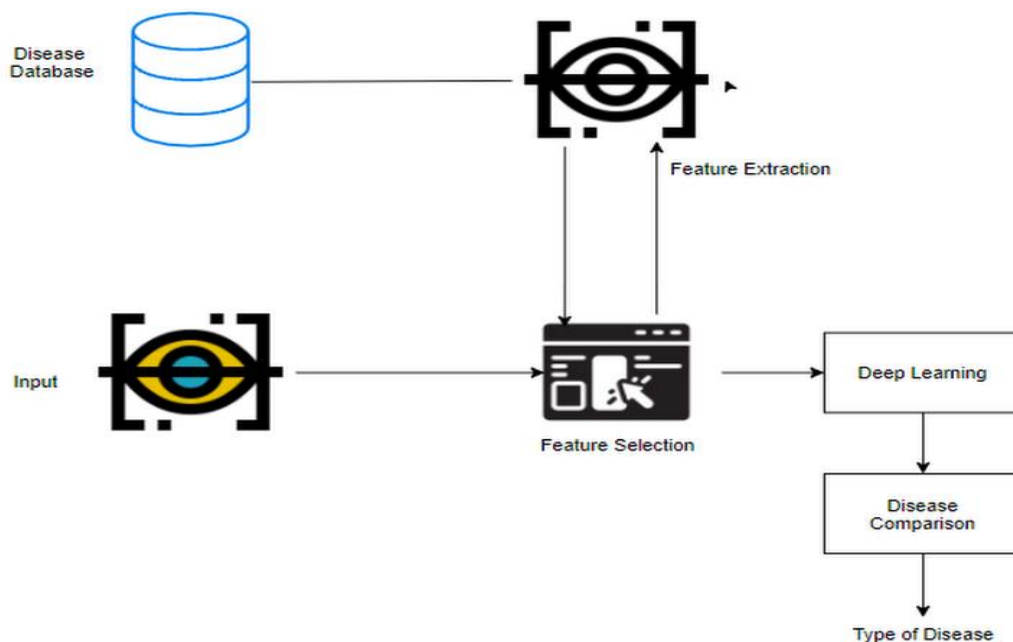


Fig.4.7 System Architecture

4.6 WORKFLOW DIAGRAM:

The form and texture-oriented elements of the picture are relied upon as the image's principal qualities. The categorization of diverse retinal illnesses has been greatly aided by effective disease detection and deep learning with convolutional neural networks (CNNs). It offered a technique for diagnosing and forecasting retinal illness. It discusses the experimental study of samples of pictures that are made up of distinct retinal types. Using a CNN trained on a publicly accessible retinal illness picture dataset, a number of neuron- and layer-wise visualisation approaches were employed. The retinal illness identification utilising a color-based classification model is shown in the example screenshots. And to use this model in a Django framework web application.

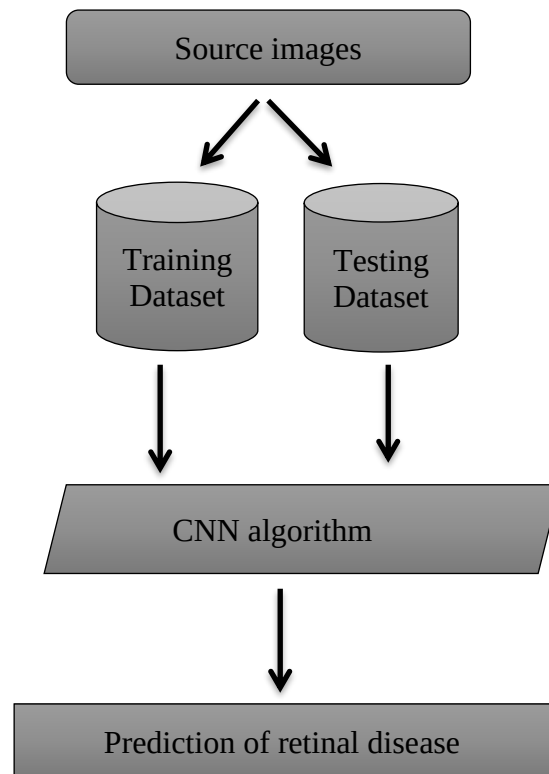


Fig.4.8 Workflow Diagram

4.7 USECASE DIAGRAM:

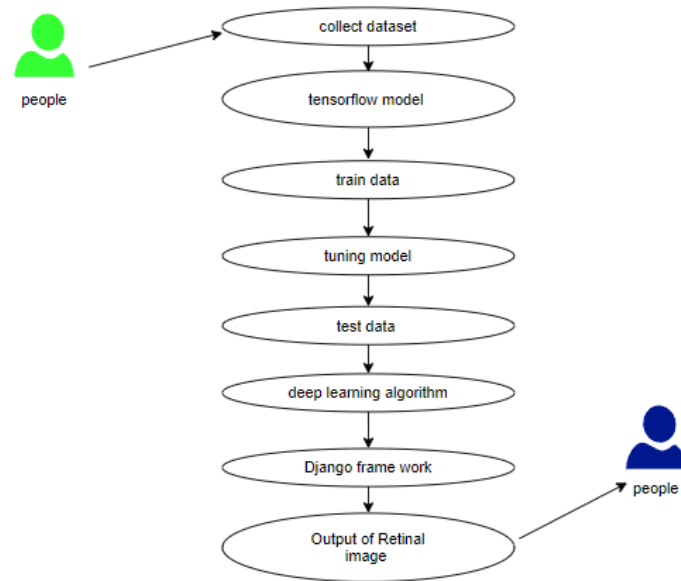


Fig.4.9 Usecase Diagram

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.

4.8 CLASS DIAGRAM:

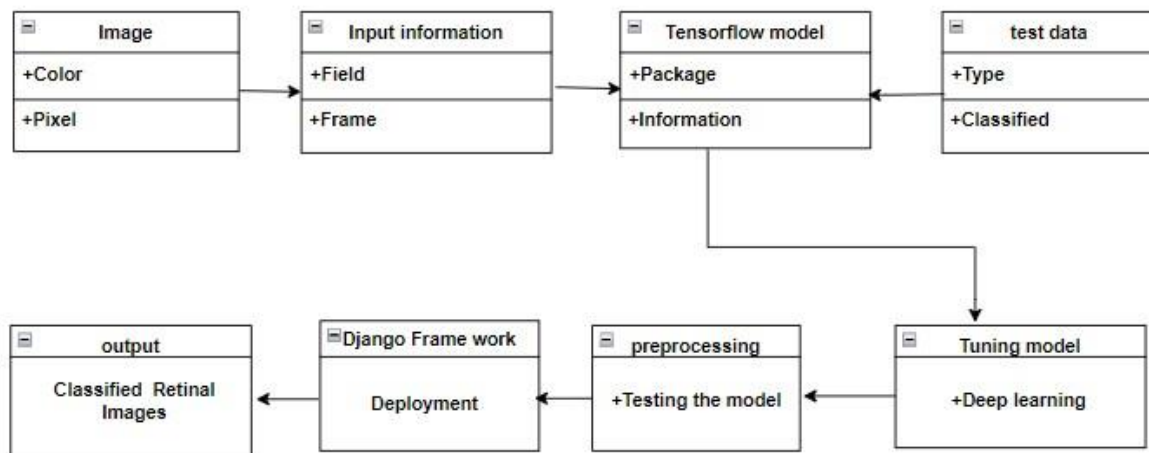


Fig.4.10 Class Diagram

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.

4.9 ACTIVITY DIAGRAM:

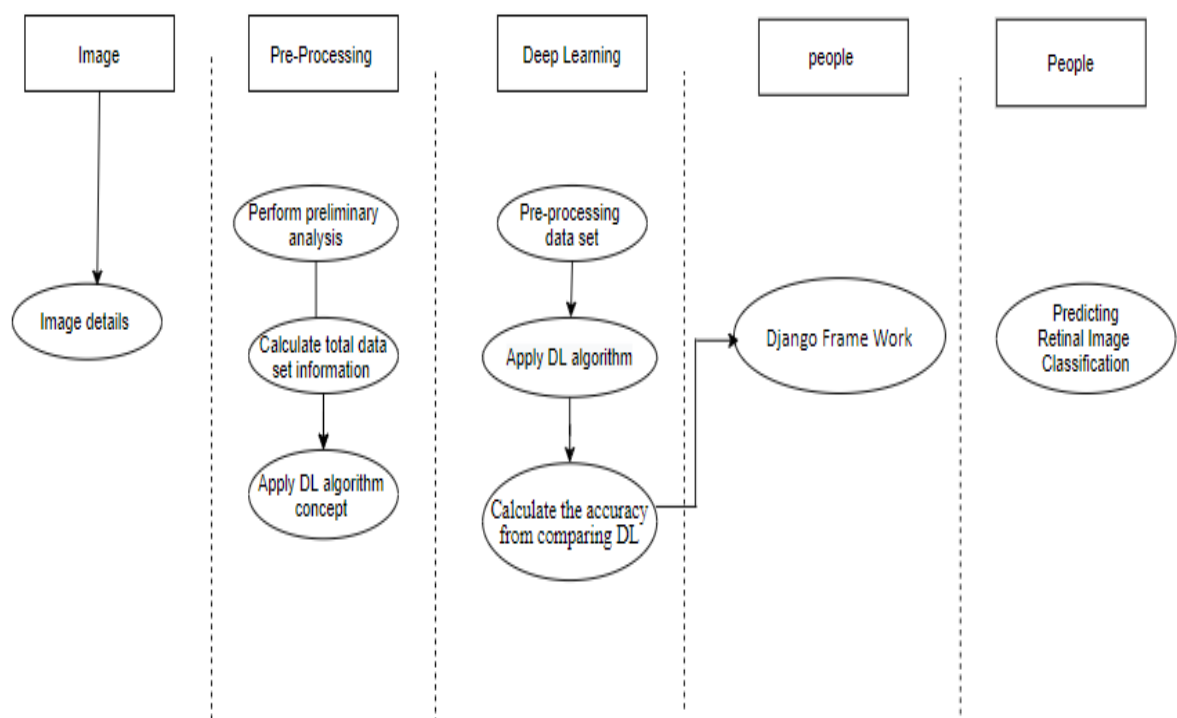


Fig.4.11 Activity Diagram

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.

4.10 ER DIAGRAM:

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.

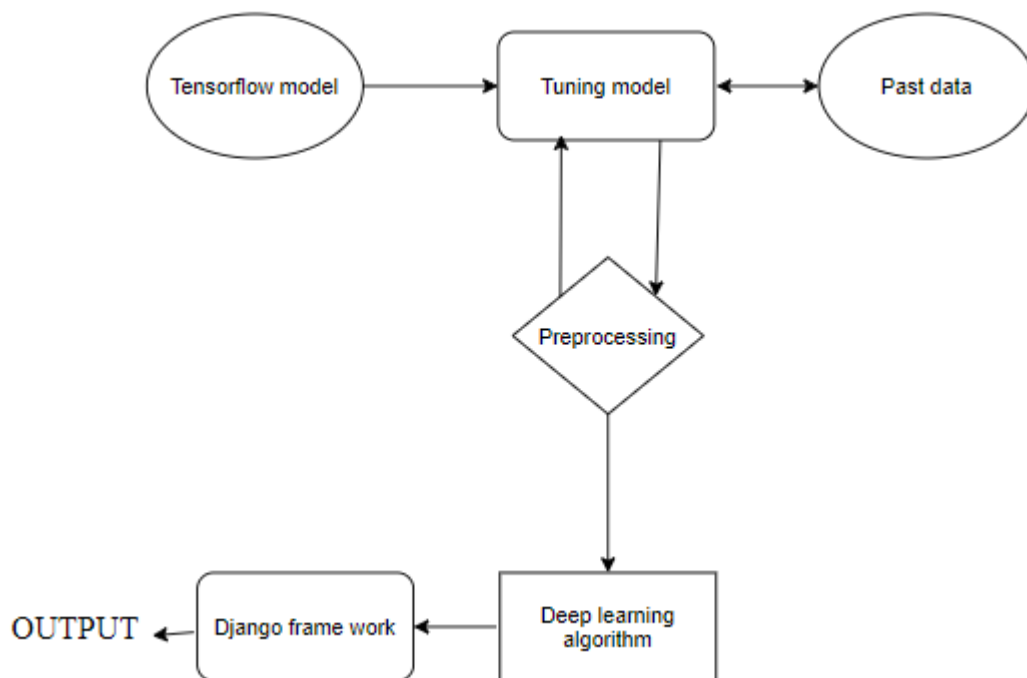


Fig.4.12 ER Diagram

4.11 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. Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science. Anaconda is a 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. 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.

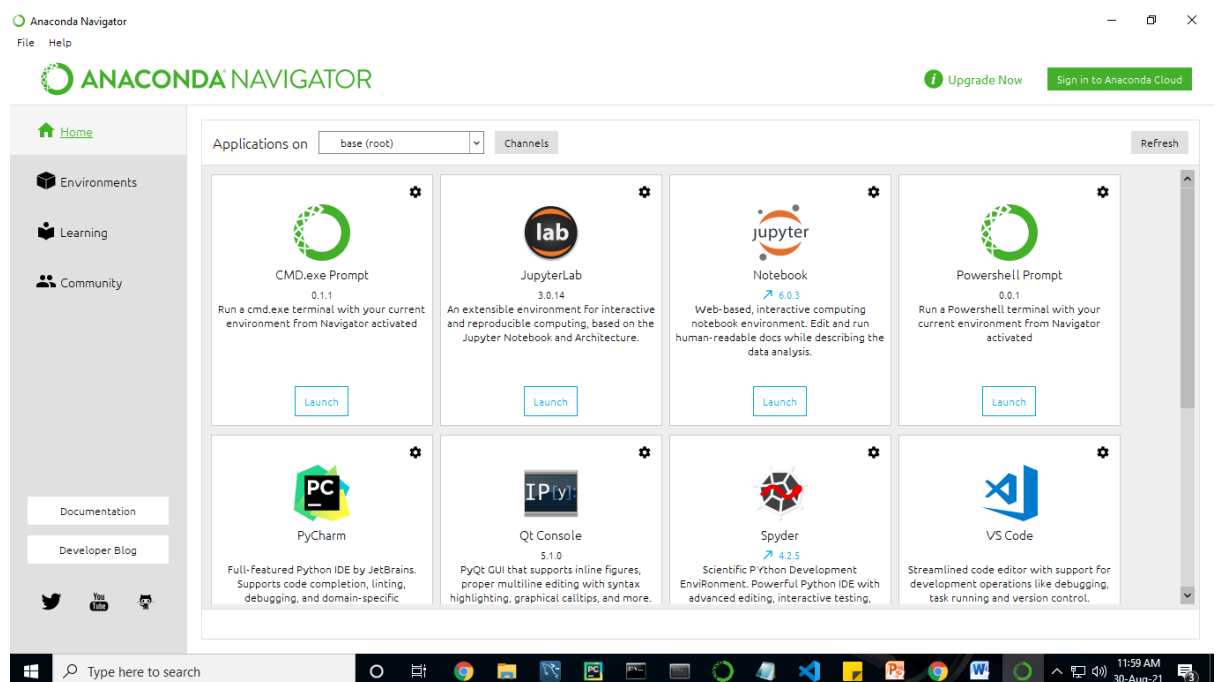


Fig.4.13 Anaconda Navigator 1

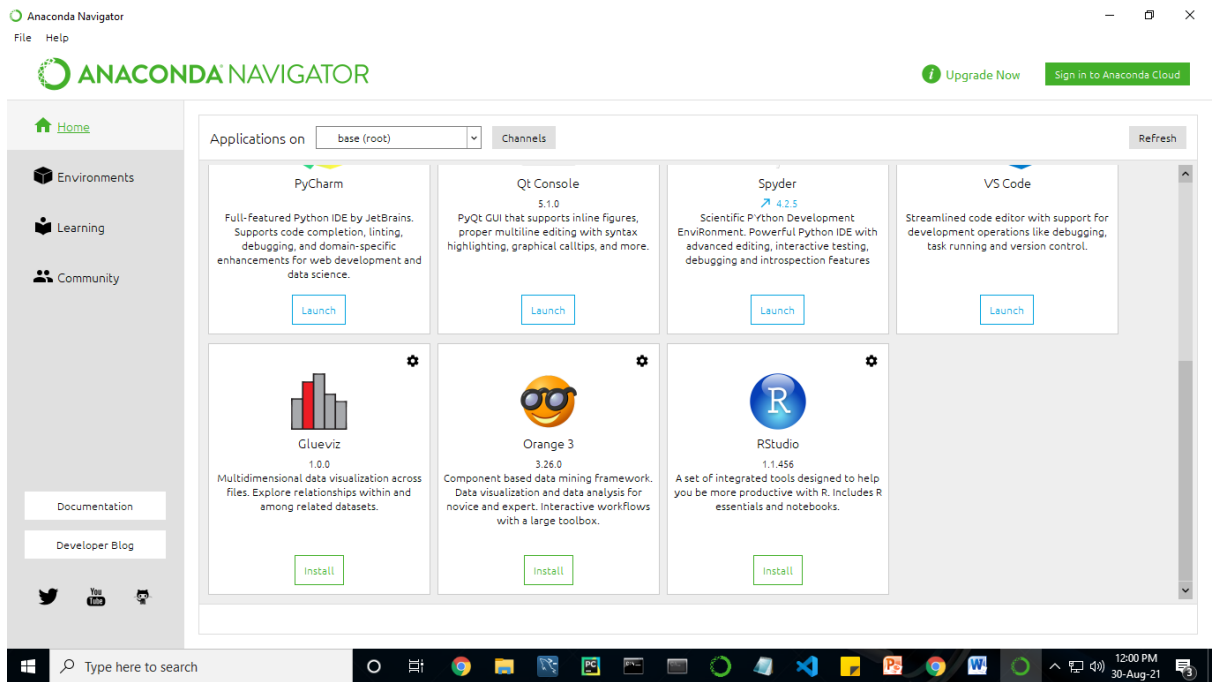


Fig.4.14 Anaconda Navigator 2

4.11.1 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.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

4.11.2 WORKING PROCESS:

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

2. Load a dataset and understand its structure using statistical summaries and data visualization.
3. Machine learning models, pick the best and build confidence that the accuracy is reliable.

4.11.3 PYCHARM:

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. Code faster and with more easily in a smart and configurable editor with code completion, snippets, code folding and split windows support.

4.12-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 images of retinal disease images, out of which any image can be used as a test image for the software.

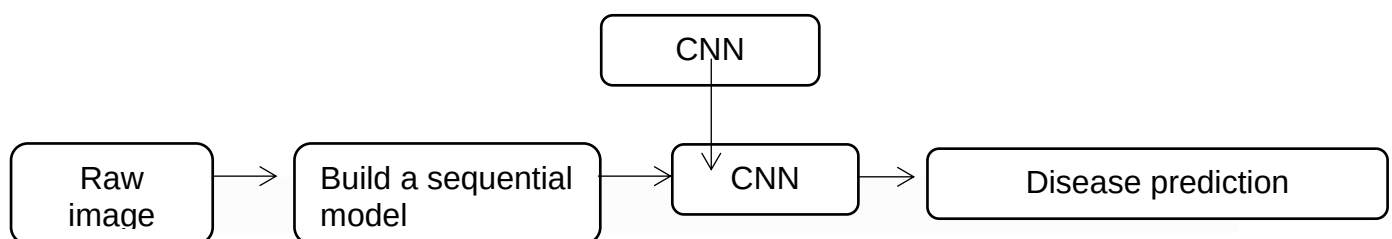


Fig.4.15 Methodology

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 disease if the retinal image contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the disease.

4.13 CNN Model steps:

4.13.1 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.

The kernel repeats this process for every location it slides over, converting a 2D matrix of features into yet another 2D matrix of features. The output features are essentially, the weighted sums (with the weights being the values of the kernel itself) of the input features located roughly in the same location of the output pixel on the input layer.

Whether or not an input feature falls within this “roughly same location”, gets determined directly by whether it’s in the area of the kernel that produced the output or not. This means the size of the kernel directly determines how many (or few) input features get combined in the production of a new output feature.

This is all in pretty stark contrast to a fully connected layer. In the above example, we have $5 \times 5 = 25$ input features, and $3 \times 3 = 9$ output features. If this were a standard fully connected layer, you’d have a weight matrix of $25 \times 9 = 225$ 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.

4.13.2 MaxPooling2D layer

Downsamples 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: $\text{output_shape} = \text{math.floor}((\text{input_shape} - \text{pool_size}) / \text{strides}) + 1$ (when $\text{input_shape} \geq \text{pool_size}$)

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

4.13.3 Arguments

- `pool_size`: integer or tuple of 2 integers, window size over which to take the maximum. (2, 2) will take the max value over a 2x2 pooling window. If only one integer is specified, the same window length will be used for both dimensions.
- `strides`: Integer, tuple of 2 integers, or None. Strides values. Specifies how far the pooling window moves for each pooling step. If None, it will default to `pool_size`.
- `padding`: One of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.
- `data_format`: A string, one of `channels_last` (default) or `channels_first`. The ordering of the dimensions in the inputs. `channels_last` corresponds to inputs with shape (batch, height, width, channels) while `channels_first` corresponds to inputs with shape (batch, channels, height, width). It defaults to the `image_data_format` value found in your Keras config file at `~/.keras/keras.json`. If you never set it, then it will be "channels_last".

4.13.4 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).

4.13.5 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).

4.13.6 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, ...)`

4.13.7 Dense layer

Dense implements the operation: $\text{output} = \text{activation}(\text{dot}(\text{input}, \text{kernel}) + \text{bias})$ 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.

Note: If the input to the layer has a rank greater than 2, then Dense computes the dot product between the inputs and the kernel along the last axis of the inputs and axis 0 of the kernel (using `tf.tensordot`). For example, if input has dimensions `(batch_size, d0, d1)`, then we create a kernel with shape `(d1, units)`, and the kernel operates along axis 2 of the input, on every sub-tensor of shape `(1, 1, d1)` (there are `batch_size * d0` such sub-tensors). The output in this case will have shape `(batch_size, d0, units)`.

Besides, layer attributes cannot be modified after the layer has been called once (except the trainable attribute). When a popular kwarg `input_shape` is passed, then keras will create an input layer to insert before the current layer. This can be treated equivalent to explicitly defining an `InputLayer`.

4.13.8 Arguments

- `units`: Positive integer, dimensionality of the output space.
- `activation`: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: $a(x) = x$).
- `use_bias`: Boolean, whether the layer uses a bias vector.
- `kernel_initializer`: Initializer for the kernel weights matrix.
- `bias_initializer`: Initializer for the bias vector.
- `kernel_regularizer`: Regularizer function applied to the kernel weights matrix.
- `bias_regularizer`: Regularizer function applied to the bias vector.
- `activity_regularizer`: Regularizer function applied to the output of the layer (its "activation").
- `kernel_constraint`: Constraint function applied to the kernel weights matrix.

- **bias_constraint:** Constraint function applied to the bias vector.

4.13.9 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).

4.13.10 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).

4.13.11 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 $1/(1 - \text{rate})$ such that the sum over all inputs is unchanged.

Note that the Dropout layer only applies when training is set to True such that no values are dropped during inference. When using `model.fit`, training will be appropriately set to True automatically, and in other contexts, you can set the kwarg explicitly to True when calling the layer.

(This is in contrast to setting `trainable=False` for a Dropout layer. `trainable` does not affect the layer's behavior, as Dropout does not have any variables/weights that can be frozen during training.)

4.13.12 Arguments

1. **rate:** Float between 0 and 1. Fraction of the input units to drop.
2. **noise_shape:** 1D integer tensor representing the shape of the binary dropout mask that will be multiplied with the input. For instance, if your inputs have shape (batch_size, timesteps, features) and you want the dropout mask to be the same for all timesteps, you can use `noise_shape=(batch_size, 1, features)`.

seed: A Python integer to use as random seed.

4.13.13 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.

4.13.14 Training Process:

`train_datagen.flow_from_directory` is the function that is used to prepare data from the `train_dataset` directory `Target_size` specifies the target size of the image. `Test_datagen.flow_from_directory` is used to prepare test data for the model and all is similar as above. `fit_generator` is used to fit the data into the model made above, other factors used are `steps_per_epochs` tells us about the number of times the model will execute for the training data.

4.13.15 Epochs:

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

4.13.16 Validation process:

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

4.14 ARTIFICIAL NEURAL NETWORK:

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets that look like the figure below. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers.

Artificial Neural Network(ANN) uses the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problems.

4.14.1 Key Points related to the architecture:

The network architecture has an input layer, hidden layer (there can be more than 1) and the output layer. It is also called MLP (Multi Layer Perceptron) because of the multiple layers. The hidden layer can be seen as a “distillation layer” that distills some

of the important patterns from the inputs and passes it onto the next layer to see. It makes the network faster and efficient by identifying only the important information from the inputs leaving out the redundant information

3. The activation function serves two notable purposes:

- It captures non-linear relationship between the inputs
- It helps convert the input into a more useful output.

4.14.2 Key advantages of neural Networks:

ANNs have some key advantages that make them most suitable for certain problems and situations:

1. ANNs have the ability to learn and model non-linear and complex relationships, which is really important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex.
2. ANNs can generalize — After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.
3. Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables (like how they should be distributed). Additionally, many studies have shown that ANNs can better model heteroskedasticity i.e. data with high volatility and non-constant variance, given its ability to learn hidden relationships in the data without imposing any fixed relationships in the data. This is something very useful in financial time series forecasting (e.g. stock prices) where data volatility is very high.

4.15 ARCHITECTURE OF CNN:

4.15.1 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.

4.15.2 Models API:

There are three ways to create Keras models:

- The Sequential model, which is very straightforward (a simple list of layers), but is limited to single-input, single-output stacks of layers (as the name gives away).
- The Functional API, which is an easy-to-use, fully-featured API that supports arbitrary model architectures. For most people and most use cases, this is what you should be using. This is the Keras "industry strength" model.
- Model subclassing, where you implement everything from scratch on your own. Use this if you have complex, out-of-the-box research use cases.

4.15.3 Types of Keras Models

Models in keras are available in two types:

- Keras Sequential Model
- Keras Functional API

1. Sequential Model in Keras

It allows us to create models layer by layer in sequential order. But it does not allow us to create models that have multiple inputs or outputs. It is best for simple stack of layers which have 1 input tensor and 1 output tensor.

This model is not suited when any of the layer in the stack has multiple inputs or outputs. Even if we want non-linear topology, it is not suited.

2. Functional API in Keras

It provides more flexibility to define a model and add layers in keras. Functional API allows us to create models that have multiple input or output. It also allows us to share these layers. In other words. we can make graphs of layers using Keras functional API.

As functional API is a data structure, it is easy to save it as a single file that helps in recreating the exact model without having the original code. Also its easy to model the graph here and access its nodes as well.

4.15.4 Model Subclassing in Keras

Sequential model does not allow you much flexibility to create your models. Functional API also only has a little of customization available for you. But you may create your own fully-customizable models in Keras. This is done by subclassing the Model class and implementing a call method.

Input() is used to instantiate a Keras tensor.

A Keras tensor is a symbolic tensor-like object, which we augment with certain attributes that allow us to build a Keras model just by knowing the inputs and outputs of the model.

For instance, if a, b and c are Keras tensors, it becomes possible to do: `model = Model(input=[a, b], output=c)`

kernels:

Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map. These are linear transformations, each convolution is a type of affine function.

In computer vision the input is often a 3 channel RGB image. For simplicity, if we take a greyscale image that has one channel (a two dimensional matrix) and a 3x3 convolutional kernel (a two dimensional matrix). The kernel strides over the input matrix of numbers moving horizontally column by column, sliding/scanning over the first rows in the matrix containing the images pixel values. Then the kernel strides down vertically to subsequent rows. Note, the filter may stride over one or several pixels at a time, this is detailed further below.

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

Padding:

To handle the edge pixels there are several approaches:

1. Losing the edge pixels
2. Padding with zero value pixels
3. 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. For a 3x3 kernel, one pixel needs to be added around the outside, for a 7x7 kernel then three pixels would be reflected around the outside. The pixels added around each side is the dimension, halved and rounded down.

Traditionally in many research papers, the edge pixels are just ignored, which loses a small proportion of the data and this gets increasing worse if there are many deep convolutional layers. For this reason, I could not find existing diagrams to easily convey some of the points here without being misleading and confusing stride 1 convolutions with stride 2 convolutions.

With padding, the output from a input of width w and height h would be width w and height h (the same as the input with a single input channel), assuming the kernel takes a stride of one pixel at a time.

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.

With padding, the output from an input of width w , height h and depth 3 would be the ceiling of width $w/2$, height $h/2$ and depth 1, as the kernel outputs a single summed output from each stride.

4.16 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.

4.16.1 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.

4.16.2 During training (i.e. when using `fit()` or when calling the layer/model with the argument `training=True`), the layer normalizes its output using the mean and standard deviation of the current batch of inputs. That is to say, for each channel being normalized, the layer returns $\gamma * (\text{batch} - \text{mean}(\text{batch})) / \sqrt{\text{var}(\text{batch}) + \text{epsilon}} + \text{beta}$, where:

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

4.16.3 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 $\gamma * (\text{batch} - \text{self.moving_mean}) / \sqrt{\text{self.moving_var} + \text{epsilon}} + \text{beta}$.

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

1. $\text{moving_mean} = \text{moving_mean} * \text{momentum} + \text{mean}(\text{batch}) * (1 - \text{momentum})$

2. $\text{moving_var} = \text{moving_var} * \text{momentum} + \text{var}(\text{batch}) * (1 - \text{momentum})$

As such, the layer will only normalize its inputs during inference *after having been trained on data that has similar statistics as the inference data*.

4.16.4 Arguments

1. **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.
2. **momentum**: Momentum for the moving average.
3. **epsilon**: Small float added to variance to avoid dividing by zero.
4. **center**: If True, add offset of beta to normalized tensor. If False, beta is ignored.
5. **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.
6. **beta_initializer**: Initializer for the beta weight.
7. **gamma_initializer**: Initializer for the gamma weight.
8. **moving_mean_initializer**: Initializer for the moving mean.
9. **moving_variance_initializer**: Initializer for the moving variance.
10. **beta_regularizer**: Optional regularizer for the beta weight.
11. **gamma_regularizer**: Optional regularizer for the gamma weight.
12. **beta_constraint**: Optional constraint for the beta weight.
13. **gamma_constraint**: Optional constraint for the gamma weight.

Call arguments

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

4.17 TYPES OF CNN:

1. AlexNet

2. LeNet

4.17.1 ALEXNET:

AlexNet is the name of a convolutional neural network which has had a large impact on the field of machine learning, specifically in the application of deep learning to machine vision. AlexNet was the first convolutional network which used GPU to boost performance.

AlexNet architecture consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 softmax layer. Each convolutional layer consists of convolutional filters and a nonlinear activation function ReLU. The pooling layers are used to perform max pooling.

4.17.2 Architecture of AlexNet:

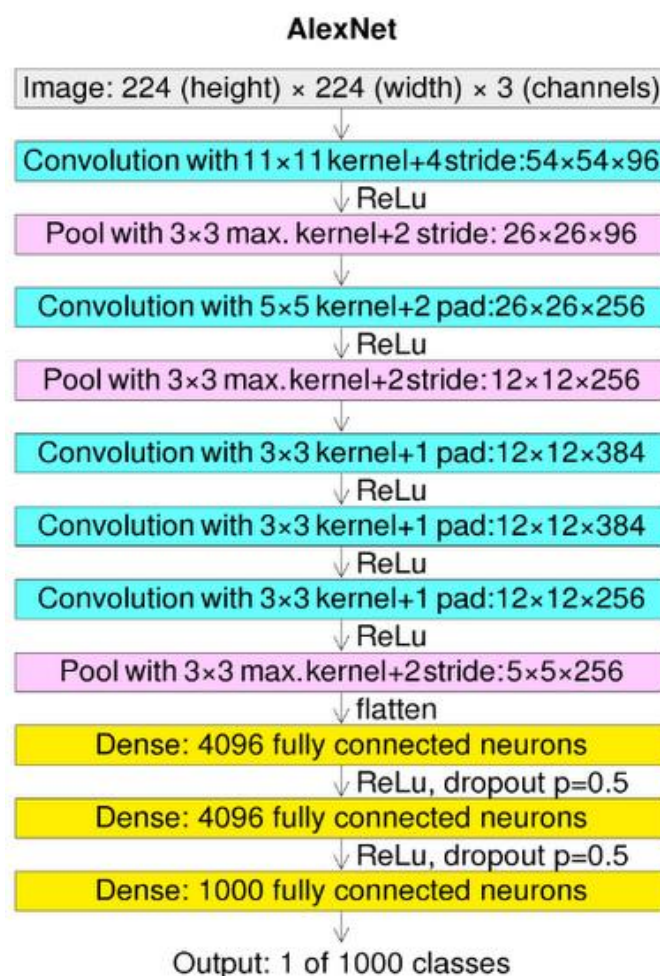


Fig.4.16 Alex Net Architecture

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.

4.18 LENET:

LeNet was one among the earliest convolutional neural networks which promoted the event of deep learning. After innumerable years of analysis and plenty of compelling iterations, the end result was named LeNet.

4.18.1 Architecture of LeNet-5:

LeNet-5 CNN architecture is made up of 7 layers. The layer composition consists of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers.

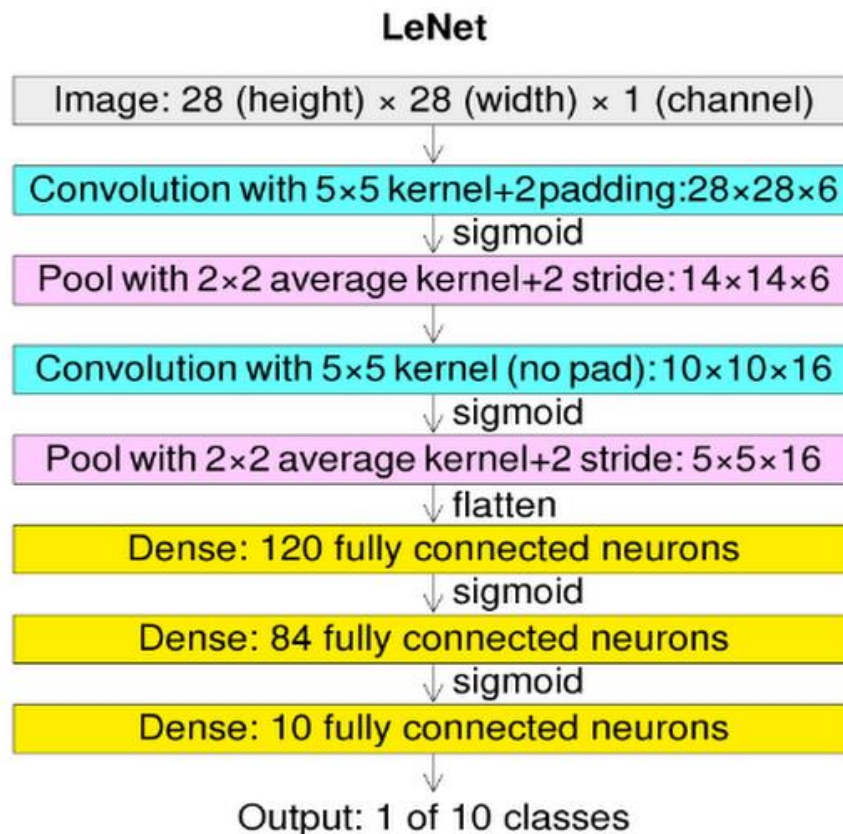


Fig4.17 Le Net Architecture

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.

4.19 MODULE DESCRIPTION:

4.19.1 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.

CNV

Trained data for CNV:

```
===== Images in: dataset/train/CNV
images_count: 400
min_width: 512
max_width: 768
min_height: 496
max_height: 496
```

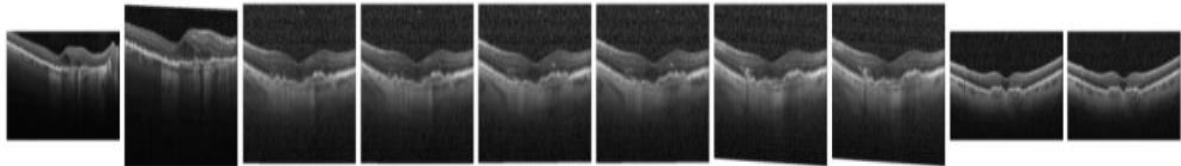


Fig.4.18 CNV

DME

Trained data for DME:

```
===== Images in: dataset/train/DME
images_count: 400
min_width: 512
max_width: 512
min_height: 496
max_height: 512
```

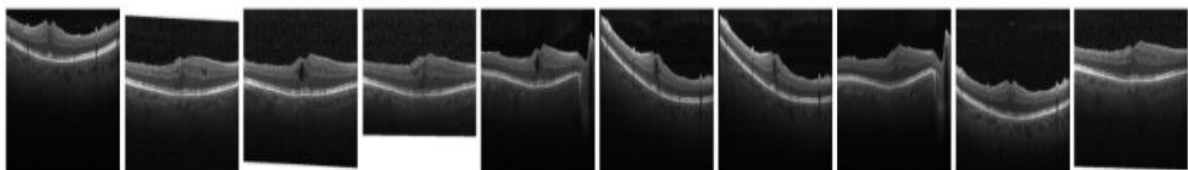


Fig.4.19 DME

DRUSEN

Trained data for DRUSEN:

```
===== Images in: dataset/train/DRUSEN
images_count: 400
min_width: 512
max_width: 512
min_height: 496
max_height: 496
```

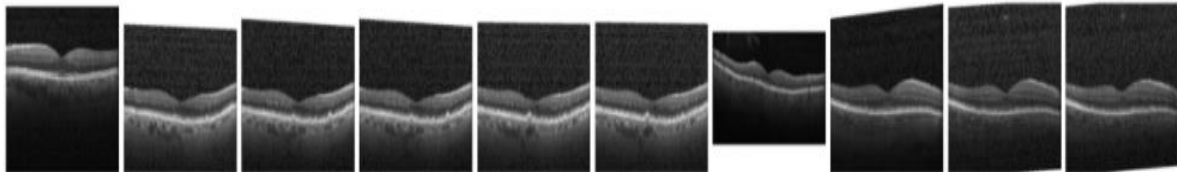


Fig.4.20 DRUCEN

NORMAL

Trained data for NORMAL:

```
===== Images in: dataset/train/NORMAL
images_count: 400
min_width: 512
max_width: 512
min_height: 496
max_height: 512
```

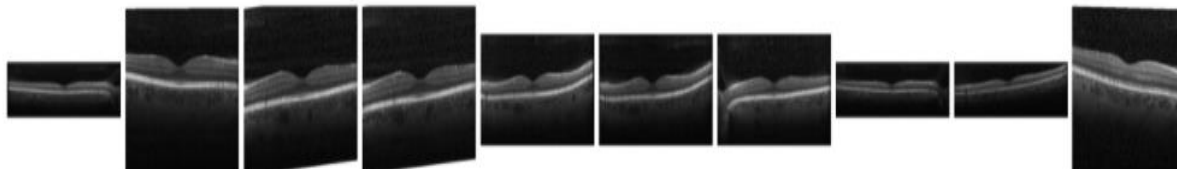


Fig.4.21 NORMAL

4.19.2 TO TRAIN THE MODULE BY GIVEN IMAGE DATASET:

To train our dataset using classifier and fit generator function also we make training steps per epoch's then total number of epochs, validation data and validation steps using this data we can train our dataset.

```

Model: "sequential"
Layer (type)                Output Shape                Param #
-----
conv2d (Conv2D)              (None, 48, 48, 96)         34944
activation (Activation)       (None, 48, 48, 96)         0
max_pooling2d (MaxPooling2D) (None, 24, 24, 96)         0
conv2d_1 (Conv2D)             (None, 14, 14, 256)        2973952
activation_1 (Activation)     (None, 14, 14, 256)        0
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 256)         0
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 256)         0
flatten (Flatten)            (None, 2304)               0
dense (Dense)                 (None, 4096)               9441280
activation_2 (Activation)     (None, 4096)               0
dropout (Dropout)            (None, 4096)               0
dense_1 (Dense)               (None, 4096)               16781312
activation_3 (Activation)     (None, 4096)               0
dropout_1 (Dropout)          (None, 4096)               0
dense_2 (Dense)               (None, 1000)               4097000
activation_4 (Activation)     (None, 1000)               0
dropout_2 (Dropout)          (None, 1000)               0
dense_3 (Dense)               (None, 4)                  4004
activation_5 (Activation)     (None, 4)                  0
=====
Total params: 33,332,492
Trainable params: 33,332,492
Non-trainable params: 0

```

Fig.4.22 Training Data

4.19.3 WORKING PROCESS OF LAYERS IN CNN MODEL:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

Input Layer:

Input layer in CNN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension $28 \times 28 = 784$, it need to convert it into 784×1 before feeding into input.

Convo Layer:

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then the filter over the next receptive field of the same input image by a Stride and do the same operation again. It will repeat the same process again and again until it goes through the whole image. The output will be the input for the next layer.

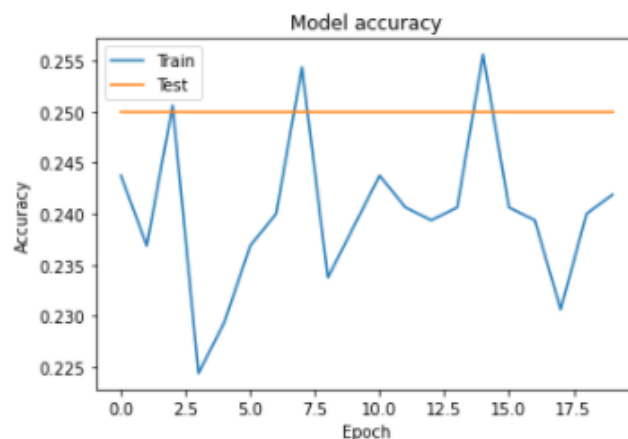


Fig4.23 CNN Model Trained Dataset Accuracy

Pooling Layer:

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive. So, the max pooling is only way to reduce the spatial volume of input image. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4×4 dimension input is reducing to 2×2 dimensions.

Fully Connected Layer (FC):

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

Softmax / Logistic Layer:

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

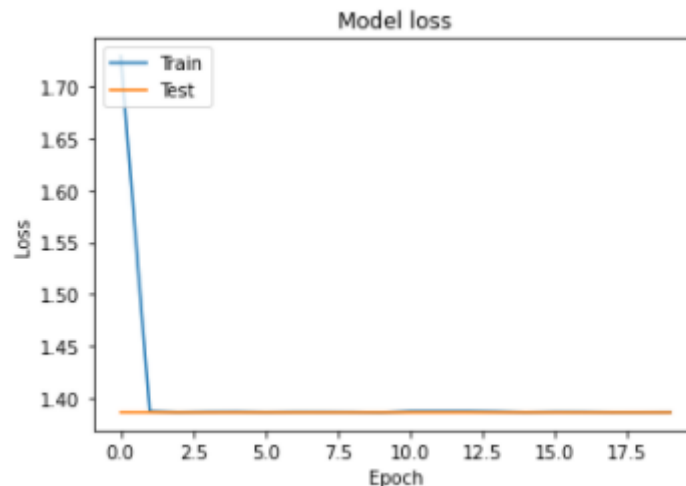


Fig.4.24 CNN Model Trained Dataset Loss Values

Output Layer:

Output layer contains the label which is in the form of one-hot encoded. Now you have a good understanding of CNN.

Libraries Required:

1. **TensorFlow**: Just to use the tensor board to compare the loss and adam curve our result data or obtained log.
2. **keras**: To pre-process the image dataset.
3. **matplotlib**: To display the result of our predictive outcome.
4. **OS**: To access the file system to read the image from the train and test directory from our machines.

4.20 DEPLOY

4.20.1 Deploying the model in Django Framework and predicting output

In this module the trained deep learning model is converted into hierarchical data format file (.h5 file) which is then deployed in our Django framework for providing better user interface and predicting the output whether the given OCT image is CNV / DME / DRUSEN / NORMAL.

4.20.2 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.

4.20.3 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.

`<!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.



Fig4.25 Basic Construction of HTML Page

4.20.4 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:

1. Cascading: Falling of Styles
2. Style: Adding designs/Styling our HTML tags
3. Sheets: Writing our style in different documents

CHAPTER 5

RESULT AND FUTURE WORK

5.1 RESULT:

The Below image shows the interface of the project:

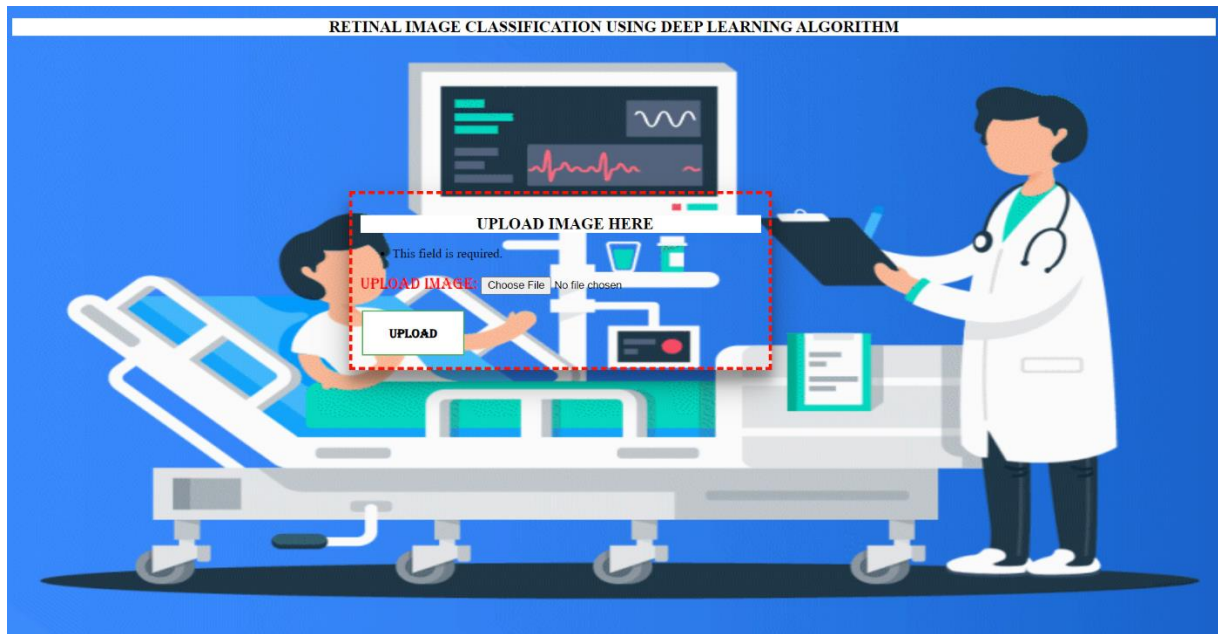


Fig.5.1 User Interface

The below image is showing Uploading the User Retinal Image:

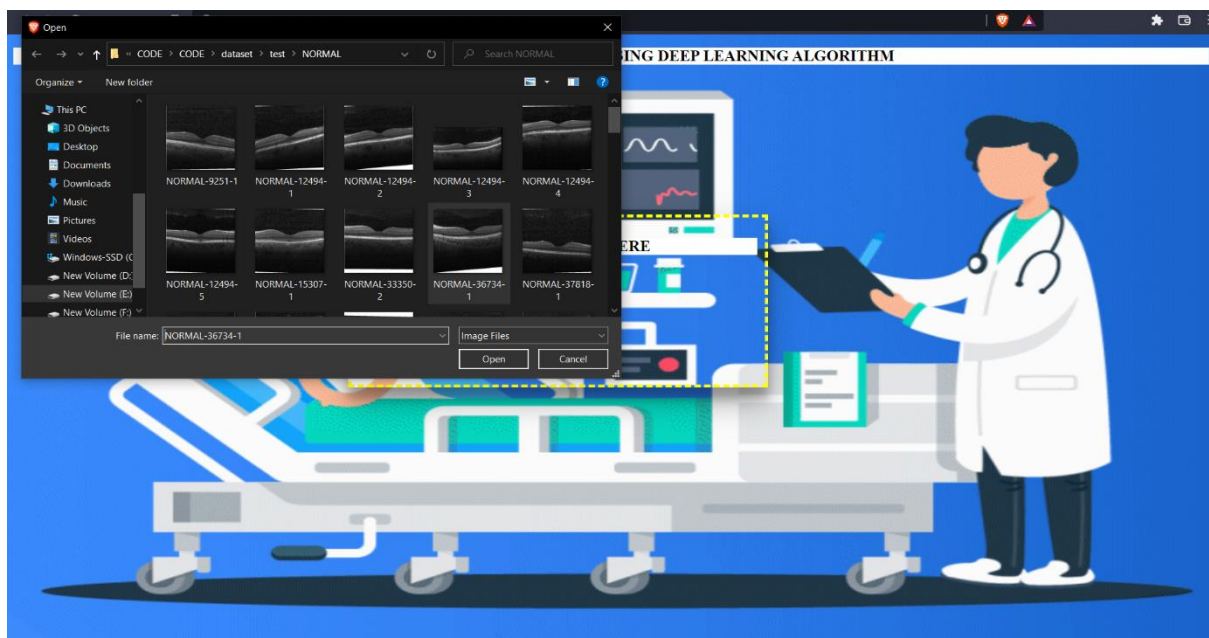


Fig.5.2 Uploading the User Data

The below image shows the Final Retinal image of the User:



Fig.5.3 Final Deployment and Result

5.2 Future Work:

1. Medical department wants to automate the detecting of retinal disease from eligibility process (real time).
2. To optimize the work to implement in Artificial Intelligence environment.

CHAPTER 6

CONCLUSION:

6.1 Summary:

It focused how image from given dataset (trained dataset) and past data set used to predict the pattern of retinal diseases using CNN model. This brings some of the following insights about retinal disease prediction. The major benefit of the CNN classification framework is the ability to classify images automatically. The eye diseases mainly contribute to blindness and often can't be remedied because the patients are diagnosed too late with the diseases. In this study, we have discussed the overview of methodologies for detecting the abnormalities in retinal images which includes collection of retinal image data set, preprocessing techniques, feature extraction techniques and classification schemes.

6.2 APPENDIX:

To build a model for training and testing:

Sample Code:

Module 1:

```
import os

import numpy as np # linear algebra
import matplotlib.pyplot as plt

# DL framework - tensorflow, keras a backend

import tensorflow as tf
import tensorflow.keras.backend as K

from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout,
BatchNormalization
from tensorflow.keras.layers import Conv2D, SeparableConv2D, LeakyReLU,
Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
EarlyStopping
from IPython.display import display
from os import listdir
```

```

from os.path import isfile, join
from PIL import Image
import glob
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
import warnings
warnings.filterwarnings('ignore')
dir_name_train_NORMAL = 'dataset/train/NORMAL'
dir_name_train_CNV = 'dataset/train/CNV'
dir_name_train_DME = 'dataset/train/DME'
dir_name_train_DRUSEN = 'dataset/train/DRUSEN'
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=(35, 10))
    for idx, img_path in enumerate(item_files):
        plt.subplot(2, n, idx+1)
        img = plt.imread(img_path)
        plt.imshow(img, cmap='gray')
        plt.axis('off')
    plt.tight_layout()
def Images_details_Print_data(data, path):
    print(" ===== Images in: ", path)
    for k, v in data.items():
        print("%s:\t%s" % (k, v))
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 # No image will be bigger than that
    data['max_width'] = 0
    data['min_height'] = 10**100 # No image will be bigger than that

```

```

data['max_height'] = 0
for f in files:
    im = Image.open(f)
    width, height = im.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'])
    Images_details_Print_data(data, path)
print("")
print("Trained data for CNV:")
print("")
Images_details(dir_name_train_CNV)
print("")
plot_images(dir_name_train_CNV, 10)
print("")
print("Trained data for DME:")
print("")
Images_details(dir_name_train_DME)
print("")
plot_images(dir_name_train_DME, 10)
print("")
print("Trained data for DRUSEN:")
print("")
Images_details(dir_name_train_DRUSEN)
print("")
plot_images(dir_name_train_DRUSEN, 10)
print("")
print("Trained data for NORMAL:")
print("")
Images_details(dir_name_train_NORMAL)
print("")
plot_images(dir_name_train_NORMAL, 10)
Classifier=Sequential()

```

```

Classifier.add(Convolution2D(32,(3,3),input_shape=(200,200,3),activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(38, activation='relu'))
Classifier.add(Dense(4, activation='softmax'))
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
training_set=train_datagen.flow_from_directory('dataset/train',target_size=(200,200),batch_size=32,class_mode='categorical')
img_dims = 150
epochs = 2
batch_size = 32
##### Fitting the model
history = Classifier.fit_generator(
    training_set, steps_per_epoch=training_set.samples / batch_size,
    epochs=epochs,
    validation_data=test_set,validation_steps=test_set.samples / batch_size)
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()
    # Plot training & validation loss values
    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()
```

REFERENCE:

- [1]. Hernandez-Matas, Carlos, Antonis A. Argyros, and Xenophon Zabulis. "Retinal image preprocessing, enhancement, and registration." *Computational Retinal Image Analysis* (2019): 59-77.
- [2] Sarki, Rubina, Khandakar Ahmed, Hua Wang, and Yanchun Zhang. "Automatic detection of diabetic eye disease through deep learning using fundus images: A survey." *IEEE Access* 8 (2020): 151133-151149.
- [3] Badar, Maryam, Muhammad Haris, and Anam Fatima. "Application of deep learning for retinal image analysis: A review." *Computer Science Review* 35 (2020): 100203.
- [4] Dutta, Suvajit, B. C. Manideep, Syed Muzamil Basha, Ronnie D. Caytiles, and N. C. S. N. Iyengar. "Classification of diabetic retinopathy images by using deep learning models." *International Journal of Grid and Distributed Computing* 11, no. 1 (2018): 89-106.
- [5] Hernandez-Matas, Carlos, Antonis A. Argyros, and Xenophon Zabulis. "Retinal image preprocessing, enhancement, and registration." *Computational Retinal Image Analysis* (2019): 59-77.
- [6] X. Li, X. Hu, L. Yu, L. Zhu, C.-W. Fu, and P.-A. Heng, "CA-Net: Cross-disease attention network for joint diabetic retinopathy and diabetic macular edema grading," *IEEE Trans. Med. Image.*, vol. 39, no. 5, pp. 1483–1493, May 2020.
- [7] L. Fang, C. Wang, S. Li, H. Rabbani, X. Chen, and Z. Liu, "Attention to lesion: Lesion-aware convolutional neural network for retinal optical coherence tomography image classification," *IEEE Trans. Med. Image.*, vol. 38, no. 8, pp. 1959–1970, Aug. 2019.
- [8] N. E. M. Khalifa, M. Lowey, M. H. N. Taha, and H. N. E. T. Mohamed, "Deep transfer learning models for medical diabetic retinopathy detection," *Acta Inf. Medica*, vol. 27, no. 5, p. 327, 2019.

- [9] Y. Zhu, C. Zhao, H. Guo, J. Wang, X. Zhao, and H. Lu, "Attention Couple Net: Fully convolutional attention coupling network for object detection," *IEEE Trans. Image Process.*, vol. 28, no. 1, pp. 113–126, Jan. 2019.
- [10] C. Li et al., "multi-view mammographic density classification by dilated and attention- guided residual learning," *IEEE/ACM Trans. Compute. Biol. Bionic.*, early access, Feb. 3, 2020.
- [11] S. Phan, S. Satoh, Y. Yoda, K. Kashiwagi, and T. Oshika, "Evaluation of deep convolutional neural networks for glaucoma detection," *Jon. J. Ophthalmic.*, vol. 63, pp. 276–283, Feb. 2019.
- [12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020
- [13] R. Rasti, H. Rabbani, A. Mehridehnavi, and F. Hajizadeh, "Macular OCT classification using a multi-scale convolutional neural network ensemble," *IEEE Trans. Med. Image.*, vol. 37, no. 4, pp. 1024–1034, Apr. 2018.
- [14] . V. Das, S. Dandapat, and P. K. Bora, "A data-efficient approach for automated classification of OCT images using generative adversarial network," *IEEE Sensors Lett.*, vol. 4, no. 1, pp. 1–4, Jan. 2020
- [15] . Y. Rong et al., "Surrogate-assisted retinal OCT image classification based on convolutional neural networks," *IEEE J. Biomed. Health Informant.*, vol. 23, no. 1, pp. 253–263, Jan. 2019.
- [16] L. Huang, X. He, L. Fang, H. Rabbani, and X. Chen, "Automatic classification of retinal optical coherence tomography images with layer guided convolutional neural network," *IEEE Signal Process. Lett.*, vol. 26, no. 7, pp. 1026–1030, Jul. 2019.
- [17] L. Huang, X. He, L. Fang, H. Rabbani, and X. Chen, "Automatic classification of retinal optical coherence tomography images with layer guided convolutional neural network," *IEEE Signal Process. Lett.*, vol. 26, no. 7, pp. 1026–1030, Jul. 2019.
- [18] C. Li et al., "multi-view mammographic density classification by dilated and attention- guided residual learning," *IEEE/ACM Trans. Compute. Biol. Bionic.*, early access, Feb. 3, 2020.

Publication Report:

RE-2022-1604-plag-report

ORIGINALITY REPORT

13%

SIMILARITY INDEX

7%

INTERNET SOURCES

9%

PUBLICATIONS

7%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to Coventry University

Student Paper

2%

2

www.researchgate.net

Internet Source

1%

3

dokumen.pub

Internet Source

1%

4

Carlos Hernandez-Matas, Antonis A. Argyros, Xenophon Zabulis. "Retinal image preprocessing, enhancement, and registration", Elsevier BV, 2019

Publication

1%

5

vuir.vu.edu.au

Internet Source

1%

IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

TITLE OF PAPER/ARTICLE/REPORT, INCLUDING ALL CONTENT IN ANY FORM, FORMAT, OR MEDIA (hereinafter, “the Work”):

Retinal Image Processing Using Neural Network Using with Deep Learning.

COMPLETE LIST OF AUTHORS:

Santhosh Kumar. M, Shiva. M, V. Maria Anu, L. Mary Gladence, S. Revathy, J. Refonna

IEEE PUBLICATION TITLE (Journal, Magazine, Conference, Book):

6th International Conference on Intelligent Computing and Control Systems (ICICCS 2022)

COPYRIGHT TRANSFER

1. The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the “IEEE”) all rights under copyright that may exist in and to: (a) the above Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

CONSENT AND RELEASE

2. In the event the undersigned makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the undersigned, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the “Presentation”). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.

3. In connection with the permission granted in Section 2, the undersigned hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

4. The undersigned hereby warrants that the Work and Presentation (collectively, the “Materials”) are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the undersigned has obtained any necessary permissions. Where necessary, the undersigned has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE.

☒ Please check this box if you do not wish to have video/audio recordings made of your conference presentation.

See reverse side for Retained Rights/Terms and Conditions, and Author Responsibilities.

GENERAL TERMS

- The undersigned represents that he/she has the power and authority to make and execute this assignment.
- The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
- In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall become null and void and all materials embodying the Work submitted to the IEEE will be destroyed.
- For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.

(1) AmRef 28.03.2022
Author/Authorized Agent for Joint Authors **Date**

U.S. GOVERNMENT EMPLOYEE CERTIFICATION (WHERE APPLICABLE)

This will certify that all authors of the Work are U.S. government employees and prepared the Work on a subject within the scope of their official duties. As such, the Work is not subject to U.S. copyright protection.

(2) _____
Authorized Signature **Date**

(Authors who are U.S. government employees should also sign signature line (1) above to enable the IEEE to claim and protect its copyright in international jurisdictions.)

CROWN COPYRIGHT CERTIFICATION (WHERE APPLICABLE)

This will certify that all authors of the Work are employees of the British or British Commonwealth Government and prepared the Work in connection with their official duties. As such, the Work is subject to Crown Copyright and is not assigned to the IEEE as set forth in the first sentence of the Copyright Transfer Section above. The undersigned acknowledges, however, that the IEEE has the right to publish, distribute and reprint the Work in all forms and media.

(3) _____
Authorized Signature **Date**

(Authors who are British or British Commonwealth Government employees should also sign line (1) above to indicate their acceptance of all terms other than the copyright transfer.)

rev. 032515

IEEE COPYRIGHT FORM (continued)

RETAINED RIGHTS/TERMS AND CONDITIONS

General

1. Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
2. Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
3. In the case of a Work performed under a U.S. Government contract or grant, the IEEE recognizes that the U.S. Government has royalty-free permission to reproduce all or portions of the Work, and to authorize others to do so, for official U.S. Government purposes only, if the contract/grant so requires.
4. Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use. The IEEE Intellectual Property Rights office must handle all such third-party requests.
5. Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

Author Online Use

6. **Personal Servers.** Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original

IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.

7. Classroom or Internal Training Use. An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.

8. Electronic Preprints. Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the IEEE work with a Digital Object Identifier (DOI) or link to the article abstract in IEEE Xplore, or (2) the accepted version only (not the IEEE-published version), including the IEEE copyright notice and full citation, with a link to the final, published article in IEEE Xplore.

INFORMATION FOR AUTHORS

Author Responsibilities

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html. Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

Author/Employer Rights

If you are employed and prepared the Work on a subject within the scope of your employment, the copyright in the Work belongs to your employer as a work-for-hire. In that case, the IEEE assumes that when you sign this Form, you are authorized to do so by your employer and that your employer has consented to the transfer of copyright, to the representation and warranty of publication rights, and to all other terms and conditions of this Form. If such authorization and consent has not been given to you, an authorized representative of your employer should sign this Form as the Author.

IEEE Copyright Ownership

It is the formal policy of the IEEE to own the copyrights to all copyrightable material in its technical publications and to the individual contributions contained therein, in order to protect the interests of the IEEE, its authors and their employers, and, at the same time, to facilitate the appropriate re-use of this material by others. The IEEE distributes its technical publications throughout the world and does so by various means such as hard copy, microfiche, microfilm, and electronic media. It also abstracts and may translate its publications, and articles contained therein, for inclusion in various compendiums, collective works, databases and similar publications.

THIS FORM MUST ACCOMPANY THE SUBMISSION OF THE AUTHOR'S MANUSCRIPT.

Questions about the submission of the form or manuscript must be sent to the publication's editor.

Please direct all questions about IEEE copyright policy to:

IEEE Intellectual Property Rights Office, copyrights@ieee.org, +1-732-562-3966 (telephone)

Acceptance Letter of the IEEE Conference:

**6th International Conference on
Intelligent Computing and Control Systems**
ICICCS 2022 | May 25-27, 2022
iciccs.com/2022/ | iccs.conf19@gmail.com

Letter of Acceptance

TO

Santhosh Kumar M, Maria Anu V, Shiva M
Dept. of IT, Sathyabama Institute of Science and Technology.

Paper ID- ICICCS083

Subject: Acceptance to the International Conference on Intelligent Computing and Control Systems [ICICCS 2022], 25-27, May 2022, Madurai, India.

Dear Author,

We are happy to inform you that your paper titled **"Retinal Image Processing Using Neural Networks with Deep Learning"** has been accepted for the oral presentation at the International Conference on Intelligent Computing and Control Systems [ICICCS 2022] to be held on **25 – 27, May 2022** at Vaigai College of Engineering, Madurai, Tamil Nadu, India.

With the evident of its previous publications, ICICCS 2022 is also dedicated for the publication in **IEEE Xplore Digital library**. As a result of the peer review process, the technical conference program committee is pleased to inform you that your paper is shortlisted and accepted for the presentation in conference event and formally accepted for inclusion in IEEE Xplore. We appreciate if you could submit the final version of manuscript at your earliest convenience, in order to ensure a novel and timely publication of your research paper.

Once again, on behalf of the conference committee we extend our warm gratitude to welcome you at our conference.

Yours' Sincerely,



Prof. Ravichandran Subbarayalu,
Conference Chair,
ICICCS 2022.

Proceedings by

**IEEE**

ICICCS Previous Publications
[2021](#) [2020](#) [2019](#) [2018](#) [2017](#)