DIABETIC RETINOPATHY DETECTION USING DEEP LEARNING TECHNIQUES

A PROJECT REPORT

Submitted by

CHARULATHA.K [REGISTER NO: 211417104042]
JAYASRI.K [REGISTER NO: 211417104092]
KAMATHAMPALLI JAYASRI REDDY
[REGISTER NO:211417104105]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

AUGUST 2021

BONAFIDE CERTIFICATE

Certified that this project report **"DIABETIC RETINOPATHY DETECTION USING DEEP LEARNING TECHNIQUES** "is the bonafide work of "**CHARULATHA (211417104042) JAYASRI.K (211417104092) KAMATHAMPALLI JAYASRI REDDY (211417104105)"** who carried out the project work under my supervision.

SIGNATURE

Dr.S.MURUGAVALLI,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mrs.K.KIRUTHIKA
SUPERVISOR
ASSISTANT PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on 05/08/2021

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Thiru.C.SAKTHIKUMAR,M.E.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Mrs.K.KIRUTHIKA.,** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

NAME OF THE STUDENTS

CHARULATHA.K
JAYASRI.K
KAMATHAMPALLI JAYASRIREDDY

ABSTRACT:

Diabetic Retinopathy is a complication of diabetes that is caused due to the changes in the blood vessels of the retina and is one of the leading causes of blindness in the developed world. Up to the present, Diabetic Retinopathy is still screened manually by ophthalmologist which is a time consuming process and hence this paper aims at automatic diagnosis of the disease into its different stages using deep learning. In our approach, we trained a Deep Convolutional Neural Network model on a large dataset consisting of around 35,000 images to automatically diagnose and thereby classify high resolution fundus images of the retina into five stages based on their severity. Within this paper, an application system is built which takes the input parameters as the patient's details along with the fundus image of the eye. A trained deep convolutional neural network model will further extract the features of the fundus images and later with the help of the activation functions like relu and softmax along with optimizer like Adam an output is obtained. The output obtained from the Convolutional Neural Network (CNN) model and the patient details will collectively make a standardized report

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- ✓ CNN – Convolutional neural network
- ✓ DR – Diabetic Retinopathy
- ✓ ReLU - Rectified Linear Unit

1. INTRODUCTION

Diabetic retinopathy is an ailment that occurs in people suffering from diabetes causing progressive damage to the retina, the light-sensitive lining at the back of the eye. Diabetic retinopathy is a serious sight-menacing complication of diabetes. The ability of the body to store and use sugar is intervened by diabetes. The disease is characterized by excess of sugar levels in the blood, which can cause damage throughout the body as well as the eyes. Over a period of time, diabetes can cause damage to the blood vessels in the retina. Diabetic retinopathy is a condition that occurs when blood and other fluids start leaking from these tiny blood vessels causing the retinal tissue to swell. This results in cloudy or blurred vision. This condition usually affects both eyes. The longer a person has diabetes, the more is the probability that they will develop diabetic retinopathy. If left untreated, diabetic retinopathy can lead to blindness. Diabetic people, who experience long periods of high blood sugar, have a tendency of fluid accumulation in the lens which in turn changes the curvature of the lens, leading to blurred vision. However, once blood sugar levels are under control, blurred vision will improve. Patients suffering from diabetes, who have controlled blood sugar levels will have a slower onset and progression of diabetic retinopathy. The early stages of diabetic retinopathy have no visual symptoms. Hence it is recommended that everyone with diabetes must have a comprehensive dilated eye examination once a year. Detecting and treating the ailment at an earlier stage can limit the potential for significant vision loss from diabetic retinopathy.

The medication of diabetic retinopathy varies depending on the severity of the disease. In order to seal the leaking blood vessels or to discourage other blood vessels from leaking laser surgeries can play an important role. In order to stop the formation of new blood vessels your optometrist might need to inject medications into the eye. People with severe diabetic retinopathy might need a surgical procedure to remove and replace the gel-like fluid in the back of the eye, called the vitreous. Surgical procedures may also be needed to repair a retinal detachment. Retinal detachment is a separation of the light-receiving lining in the back of the eye. The diabetic people can help prevent or slow the growth of diabetic retinopathy by taking

prescribed medication, sticking to controlled diet, exercising on a regular basis, controlling high blood pressure, avoiding alcohol and smoking.

Diabetic Retinopathy comprises of certain stages which help us to distinguish the severity of the disease and accordingly come up with the prevention methods. The stages of diabetic retinopathy include:

- No DR: In this stage the eye is not affected with the disease. i.e. eye is healthy
- Mild DR: In the first stage, mild no proliferative, there will be balloon-like swelling in small areas of the blood vessels in the retina.
- Moderate DR: In the second stage, known as moderate no proliferative retinopathy, someof the blood vessels in the retina will become blocked.
- Severe DR: The third stage, severe no proliferative retinopathy brings with it more blocked blood vessels, which leads to areas of the retina no longer receiving adequate blood flow. Without proper blood flow, the retina can't grow new blood vessels to replace the damaged ones.
- Proliferative DR: The fourth and final stage is known as proliferative retinopathy. This is the advanced stage of the disease. Additional new blood vessels will begin to grow in the retina, but they will be fragile and abnormal. Because of this, they can leak blood which will lead to vision loss and possibly blindness.

Convolution neural network is a subset of deep learning neural network. It is mainly used for image classification and image analysis. The goal behind CNN is to mimic how human brain analyzes the image. Convolution neural network is comprised of one or more convolutional layers and then followed by one or more fully connected layers.

The CNN consist of input, hidden and output layer. The input layer basically consists of arrays of pixels. The hidden layer is the most important layer as it plays the main role in image computation. Hidden layer comprises of activation functions and biases. The output layer helps us to determine the class score. The benefit of CNN's is that they are easier to train with providing high accuracies.

1.1 Overview

The main difficulty faced by DR affected patients is that they are unaware of the disease until the changes in the retina have progressed to a level that treatment will in turn tend to be less effective. Automated screening techniques for the detection purpose have great significance in saving cost, time and labor. The screening of diabetic patients for the development of diabetic retinopathy can reduce the risk of blindness by 50%.With the increase in the rate of the patients affected by the disease there is all the more need for automated systems to take the charge since the number of ophthalmologists is also not sufficient to cope with all patients, especially in rural areas or if the workload of local ophthalmologists is substantial.

Therefore, automated early detection could limit the severity of the disease and assist ophthalmologists in investigating and treating the disease more efficiently. There may exist different kind of abnormal lesions caused by diabetic retinopathy the most frequent being exudates, hemorrhage, micro aneurysm.

1.2 Problem Definition

The major challenge was about the computational complexity that the laptops that we have with us could handle. The program had to deal with several thousands of neurons. Initially while working with a laptop that was running on an Intel Celeron n3050 processor which had a clock speed of 1.6 GHz, it was not possible to train more that 37 images. Mat lab (software) crashed every single time we tried to add a new image for training. Then we moved to using a laptop that had a dual core Intel i3 processor which had a clock frequency of 3.4GHz. With this we were able to train more images but it took approximately 20 to 25 minutes for training each new image. Finally we moved to the systems running on quad core Intel i5 processor which took only less than two minutes to train with an image.

2. LITERATURE REVIEW

2.1 Traditional DR Detection approach

Chandrashekar [1] proposed a work using morphological methods for the retinal vessels extraction on the retinal fundus images. Jaspreet and Sinha [2] proposed a segmentation method that uses morphological filters for segmenting blood vessels. No effective improvement in the performance on increasing the filter bank counts; instead it increases the convolution operation, which is a time-consuming task. Hussain et al. [3] proposed a method that uses adaptive thresholding for exudates detection and eliminates artifacts from the exudates; the retinal structures are used in classification. The proposed method did not cover all the DR signs; it needs to be explored. Jiang and Mojon [4] proposed a method, adaptive thresholding on verification-based multi-threshold probing approach. With global thresholding, the blood vessels cannot be segmented due to the image gradients. So, image probing with various threshold values were used to extract the threshold image. Sanchez et al.[5] proposed a mixture model that separates the exudates from the image background, and edge detection methods are used to separate hard and soft exudates.

Jonathan et al. [6] used multiple classifiers for the retinal image classification. To distinguish the blood vessels, micro aneurysms, and exudates, segmentation was carried out on fundus images. The segmented region, textural features, and other features obtained from GLCM were given to the classifiers to classify the normal and abnormal images. The detection system has achieved 92% on normal images and 91% on abnormal images. Gerald Liew [7] used a statistical approach that showed the association of retinal vascular signs and highlighted both qualitative and quantitative evaluation of retinal vasculature. This proposed system needs trained assistance in the identification of blood vessels. Reza et al. [8] used a marker controlled based watershed segmentation method, which detects the exudates and opticdisk. Before that, average filtering and contrast enhancement has been applied to remove artifacts. Hoover & Goldbaum [9] proposed a fuzzy-based voting system that detects the optic disk location where many feature elements overlap. Li and Chutatape [10] used an active shape model, which is a model-based approach to extract the vasculature parts based on the Optic disk location. The extracted data was used to discover the central macular region.

Gardener et al. [11] used an Artificial Neural Network (ANN) to find different features such as exudates, blood vessels, and hemorrhages with 93.1%, 91.7%, and 73.8% of accuracy. Yun et al. [12] proposed an automatic diabetic retinopathy classification system that classifies mild, moderate, and severe non-proliferative diabetic retinopathy and proliferative diabetic retinopathy and achieved an accuracy of 72% which extracts six features. In general, the traditional approach for DR detection and classification can be made, as shown in Fig -3. Initially, the data can be collected, preprocessed, image segmentation can be performed to segment the essential part, image features are extracted manually, and classification can be done separately using binary or multi-class classifiers.

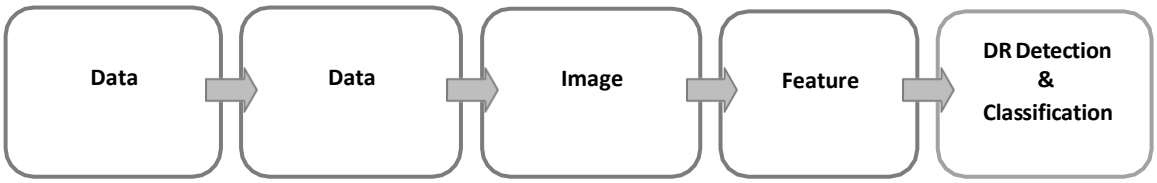


Fig -1: DR detection and classification process using traditional method

2.1 Deep Learning based DR Detection Approach

Deep Learning (DL) has become a robust tool that acts as a part of machine learning andan optimal technique which can replace machine learning in most areas. Deep Learning modelhas a hierarchical based architecture which consists of a multilayered structure. Predominantly, in medical image analysis, DL plays a unique role in medical image classification, localization, segmenting essential data, and detection. Recently, in Diabetic Retinopathy disease detection and classification, DL provides impressive results with various methods. Some of the DL-based methods are Convolutional Neural Networks (CNN), Deep Boltzmann Machines (DBM), Auto encoders, Deep Neural Networks (DNN), Recurrent Neural Networks (RNN), Deep Belief Networks (DBN), Generative Adversarial Networks (GAN) and sparse coding. An increase in the number of training data increases the model performance since many low, and high-level features are automatically extracted and learned from the training data.

Convolutional Neural Networks (CNNs) are universally used by many researchers in medical image analysis than other DL-based methods. The CNN architecture consists of three common layers: convolutional layers, pooling layers, and fully connected layers. According tothe researcher's vision and requirement, CNNs size, the number of layers, and filter count vary.In the convolutional layers, various filters combine to extract the image features to produce feature maps. Next, in the pooling layers, those feature maps dimensions are reduced mostly using the average or max-pooling technique. After pooling layers, fully connected layers are used, which describes the overall image feature set. Finally, the classification task is accomplished by one of the two activation functions, sigmoid for binary classification and softmax for multi-classification, respectively. Some of the pre-trained CNN architectures available on the Image Net dataset are LeNet, Alex Net, Google Net, VGGNet, and ResNet. These pre-trained architectures are used to speed up the training process with transfer learning strategies such as fine-tuning some of the middle layers or fully connected layers, or in some cases, the whole pre-trained model is trained.

In general, the DL-based DR detection and classification can be done, as shown in Fig -4. Initially, the data can be collected, preprocessed to enhance the image quality, augmentation can be done if necessary when the image samples are less. Then, it can be fed into the DL modelto extract image features and classify them according to the severity levels.

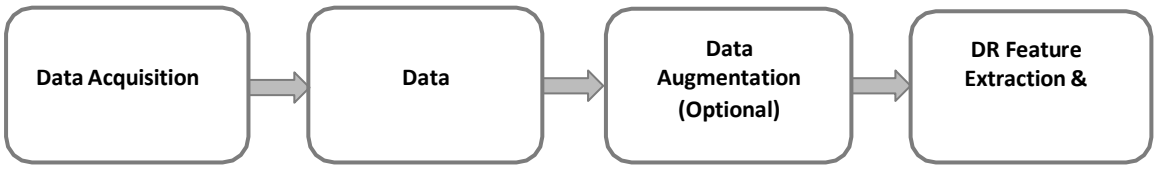


Fig -2: DR detection and classification process using DL-based method

Many DL-based automatic diabetic retinopathy detection systems were developed, andthe classification method differs from researcher to researcher. Some of the classification methods used in literature are Binary and multi-level classifications, Vessels-based classification, and Lesion-based classification.

2.2 Binary and multi-level classification

In diabetic retinopathy disease detection, various studies have been made on different classification methods. In binary classification, the diabetic retinopathy disease has been classified into two classes only. As per studies, the two classes can be normal image/DR image and Referable DR/Non-referable DR. In multi-level classification, the diabetic retinopathy disease has been classified into many classes as per researchers' vision.

2.2.1 Binary classification

Quellec et al. [13] proposed an automated detection method where three CNNs (Alex Net and two other networks) were used to detect micro aneurysms, hemorrhages, soft and hard exudates from three different datasets, Kaggle, DiaretDB1and E- ophtha (private). The fundus images were resized, cropped, normalized, augmented, and the Gaussian filter was applied in the preprocessing phase. The disease was classified into two classes, referable and non-referable DR, and produced a ROC value of 0.954 and 0.949 in Kaggle and E-ophtha, respectively.

Jiang et al. [14] proposed a model where three pre-trained CNNs (Inception-v3, ResNet152, and Inception-ResNet-v2) were used to classify the dataset as referable diabetic retinopathy or non-referable diabetic retinopathy. Before CNN training, the images were resized, enhanced and augmented, and then the models were integrated using Adaboost technique. Further to update the network weights, Adam optimizer was used and the system achieved 88.21% accuracy and AUC value of 0.946.

Zago et al. [15] proposed a method where two CNNs (pre-trained VGG16 and a CNN) were used to detect Diabetic Retinopathy or non-diabetic retinopathy images based on the red lesion patches probability. This model was trained on the DIARETDB1 dataset, and it was tested on the few datasets: IDRiD, Messidor, Messidor-2, DDR, DIARETDB0, and Kaggle. The model achieved the best results on the Messidor dataset with a sensitivity value of 0.94 and an AUC value of0.912.

2.2.2 Multi-level classification

Pratt et al. [16] proposed a method where a CNN was used with ten convolutional layers, eight max-pooling layers, three fully connected layers, and a softmax classifier was used to classify the Kaggle dataset images into five classes according to the DR severity levels. During the preprocessing phase, the images are color normalized, resized, and to reduce overfitting, L2 regularization and dropout techniques were used. The system produced a specificity of 95%, an accuracy of 75%, and a sensitivity of 30%.

Gulshan et al. [17] proposed a method where 10 CNNs (pre-trained Inception-v3) were trained to detect Diabetic macular edema (DME) and Diabetic retinopathy. Eyepacs-1 and Messidor-2 datasets were used to test the CNN model. The dataset images were initially normalized, resized and fed into the CNN model to classify the images into referable DME, moderate/worse DR, severe/worse DR, or fully gradable DR. The model produced a specificityof 93% in two of the datasets taken and sensitivity of 97.5% and 96.1% in eyepacs-1 and Messidor-2 datasets respectively.

Table 1: Merits and Demerits of Multi-level DR classification methods

DL Technique	Dataset	Merits	Demerits
CNN [19]	Messidor-2(1748 images)	o IDX-DR device was integrated with a CNN for DR detection and classification	○ They considered mild DR images as no DR ○ The 5 DR severity levels were not examined
1. Back propagation Neural Network (BPNN) 2. Deep Neural Network (DNN) 3. CNN (pre-trained VGG16) [20]	Kaggle (2000 images)	o The dataset images were preprocessed by detecting edges, applying median filter, and binary conversion and so on o The result shows DNN surpasses the CNN and BPNN	○ For network training, they have not used many images. So, the model could only learn few features ○ They have used only one dataset to assess their work
CNN (Alex Net, ResNet, Google Net,VGGNet) [21]	Kaggle (35,126 images)	o Transfer learning were used which ultimately reduces the training time o The last FC layer and hyperparameters alonewas tuned o VGGNet achieved best results	○ They have used only one dataset to assess their work ○ No DR lesions were detected
CNN(pre-trained AlexNet) [22]	1. Training - Kaggle 2. Testing – IDRiD	o The Alex Net and the handcrafted features are integrated	○ They have used only one dataset to assess their work ○ No DR lesions were detected
R-FCN [23]	1. Messidor 2. Private dataset	o Feature pyramid network and 5 region proposal networks has been added to modify a R-FCN method	○ They have used only one public dataset to assess their work ○ No exudates detected, only HM and MA are detected

3. SYSTEM AND REQUIREMENT ANALYSIS

3.1. SYSTEM ANALYSIS

3.1.1 Existing System

Besides a binocular model for the five class DR detection task is also trained and evaluated to further prove the effectiveness of the binocular design. The result shows that, on a 10% validation set, the binocular model achieves a kappa score of 0.829 which is higher than that of existing non ensemble model. Finally the comparison between confusion matrices obtained through models with paired and unpaired inputs is performed and it demonstrates that the binocular architecture does improve the classification performance.

3.1.2 Proposed System

For a deep learning model, the most important parts that should be focused on are data set, network architecture and training method. Before being used to train our model, fundus images data set obtained from public resources is preprocessed and augmented. The model accepts two fundus images corresponding to the left eye and right eye as inputs and then transmits them into the Siamese-like blocks. The information from two eyes is gathered into the fully-connected layer and finally the model will output the diagnosis result of each eye respectively.

3.1.3 Hardware Requirements

Hard Disk	:	500GB and Above
RAM	:	4GB and Above
Processor	:	I3 and Above

3.1.4 Software Requirements

Operating System	:	Windows 7 , 8, 10 (64 bit)
Software	:	Python 3.7
Tools	:	Anaconda (Jupyter Note Book IDE)

3.2 REQUIREMENT ANALYSIS

3.2.1PERFORMANCE REQUIREMENT

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the web service;
- Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module

3.2.2. SAFETY REQUIREMENTS

1. The software may be safety-critical. If so, there are issues associated with its integrity level

2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.

3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.

4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.

5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.

6. Systems with different requirements for safety levels must be separated.

7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

4. IMPLEMENTATION CONSTRAINS AND SYSTEM DESIGN

4.1 IMPLEMENTATION CONSTRAINS

4.1.1 Constraints in Analysis

- Constraints as Informal Text
- Constraints as Operational Restrictions
- Constraints Integrated in Existing Model Concepts
- Constraints as a Separate Concept
- Constraints Implied by the Model Structure

4.1.2 Constraints in Design

- Determination of the Involved Classes
- Determination of the Involved Objects
- Determination of the Involved Actions
- Determination of the Require Clauses
- Global actions and Constraint Realization

4.1.3 Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat

relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

4.2 SYSTEM DESIGN

4.2.1 UML diagram

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

4.2.1.1. Usecase Diagram

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.
Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system

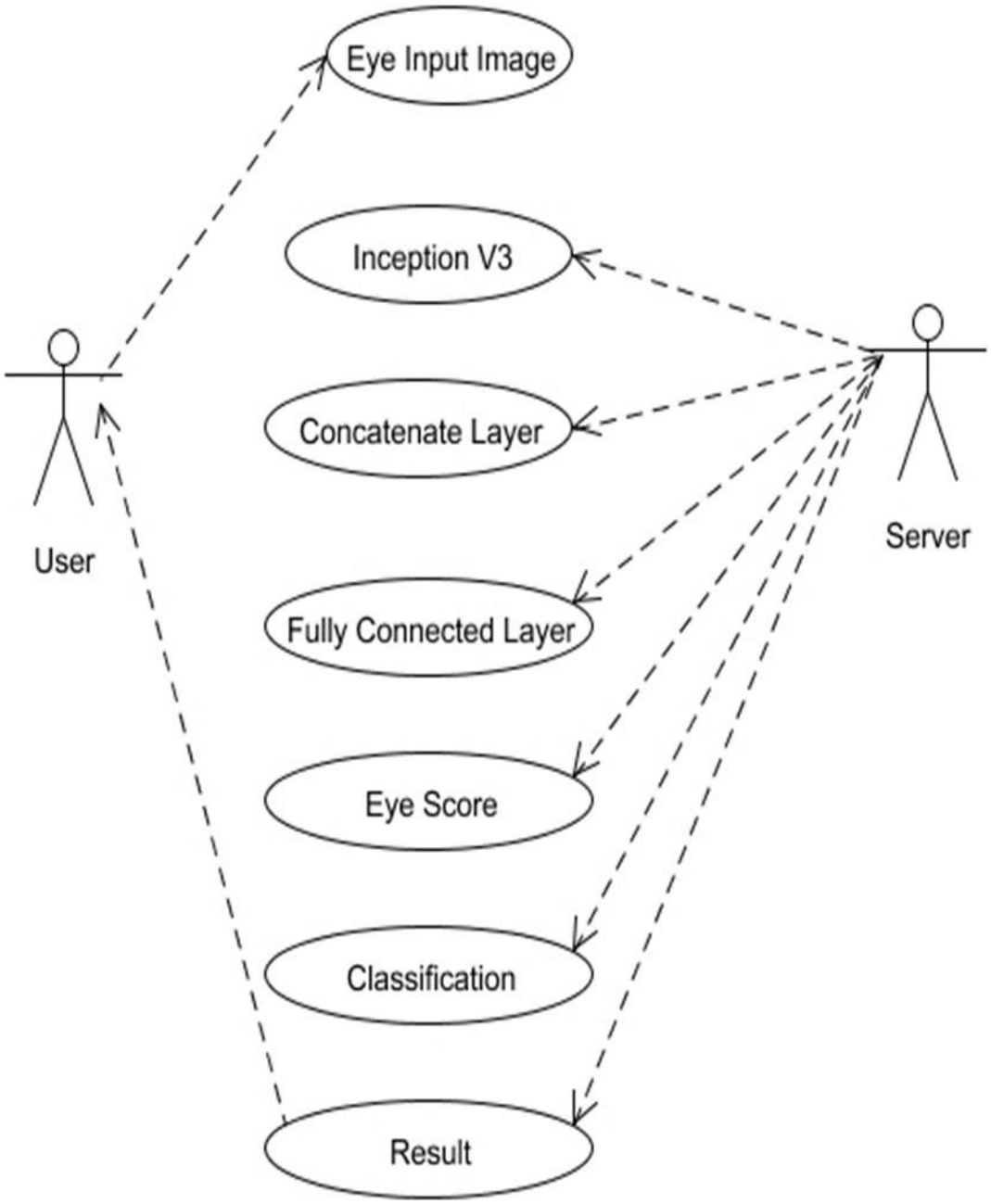


Fig .4.1

4.2.1.2 Activity Diagram

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow

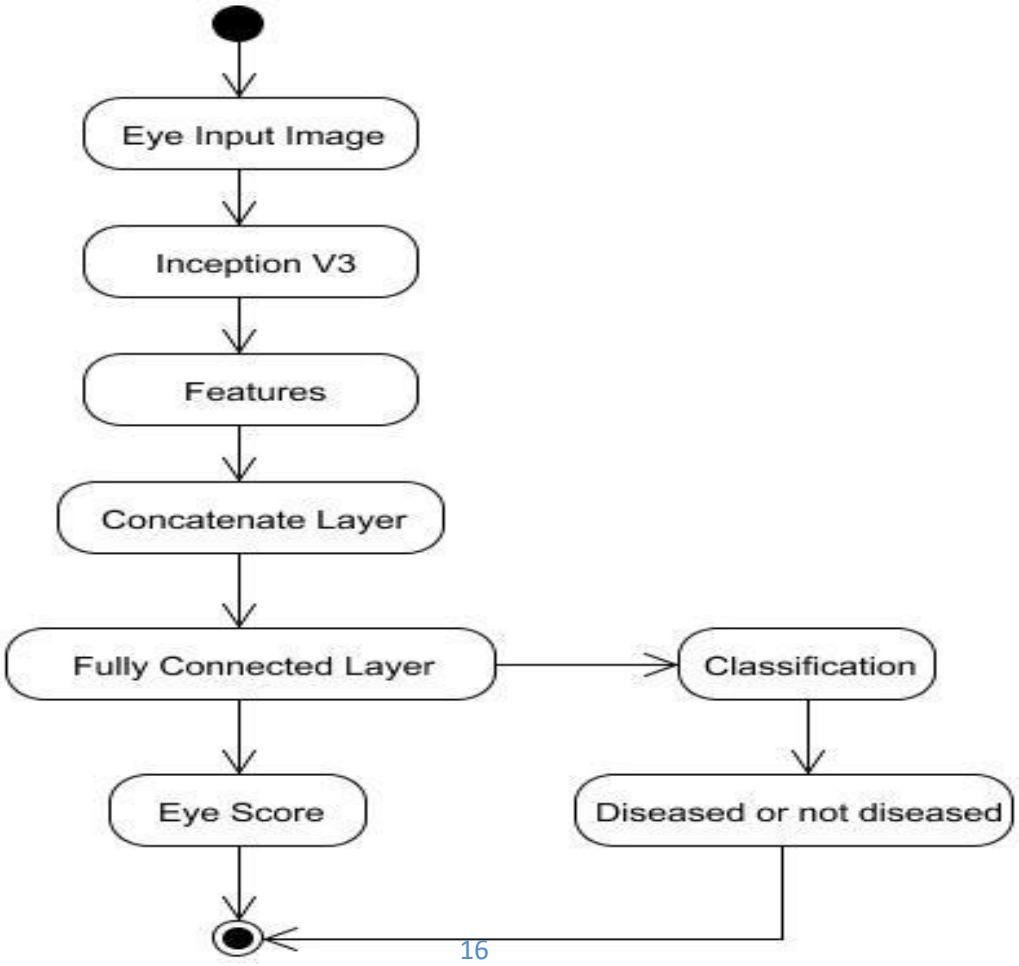


Fig 4.2

4.2.1.3 Sequence Diagram

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram

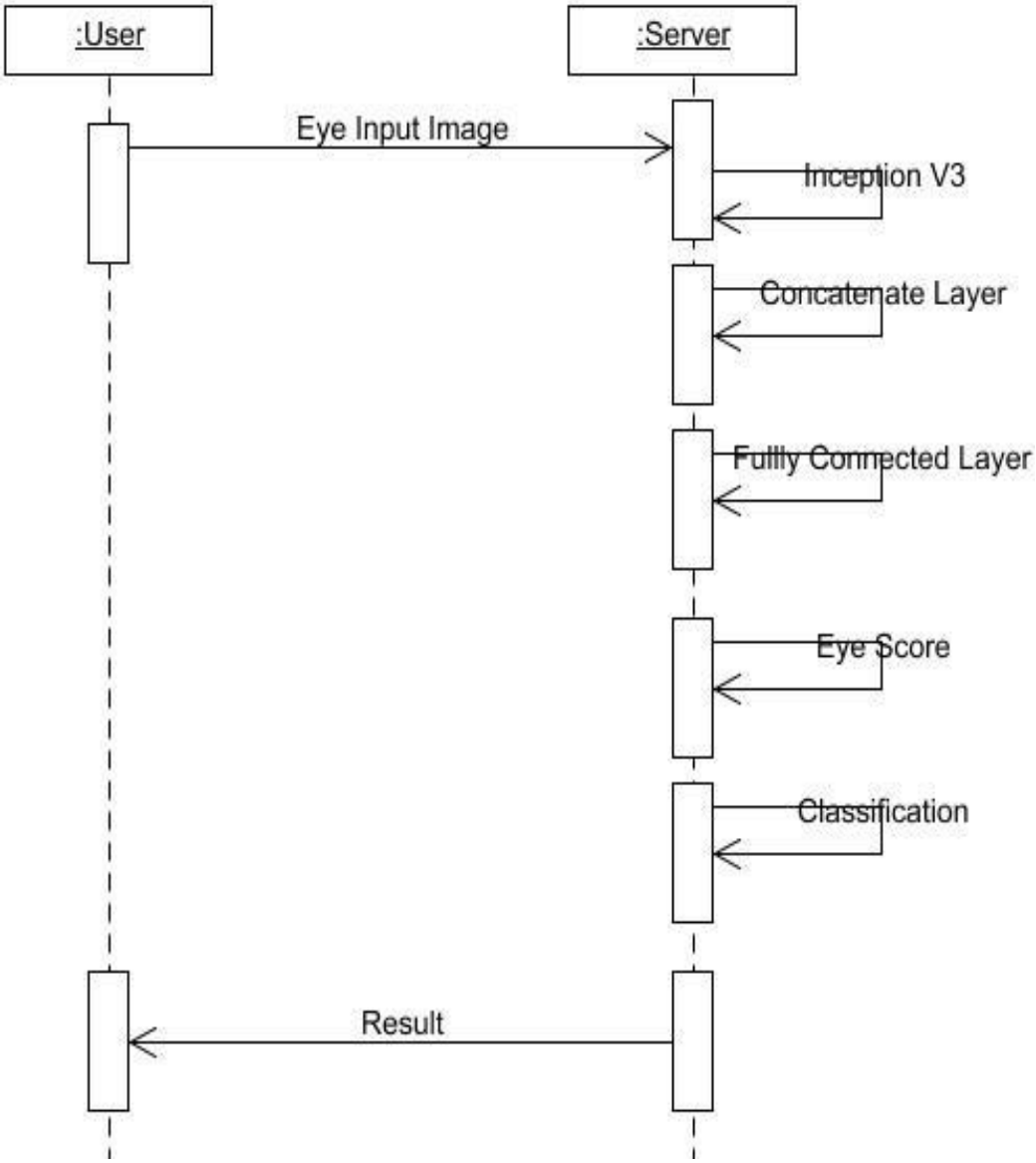


Fig 4.3

4.2.1.4 Collaboration Diagram

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

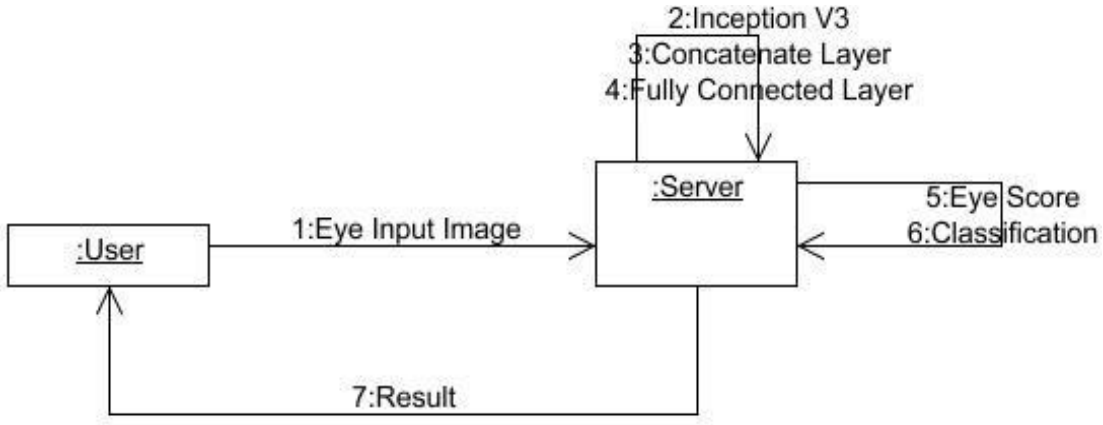


Fig 4.4

4.2.1.5 Block Diagram

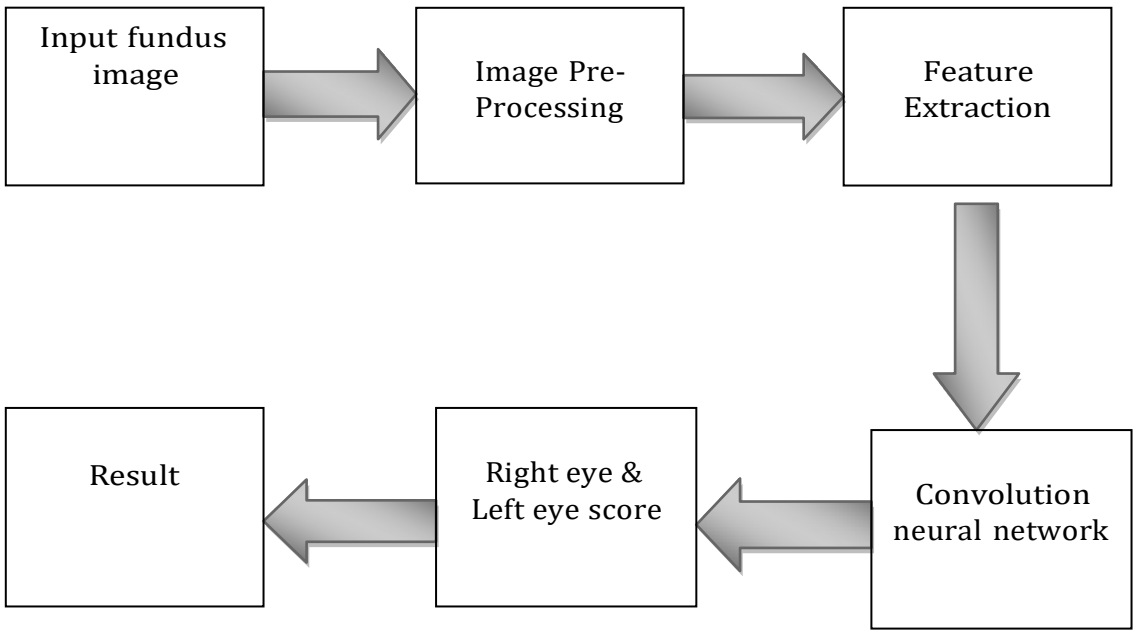


fig 4.5

5. SYSTEM ARCHITECTURE

5.1 ARCHITECTURE OVERVIEW

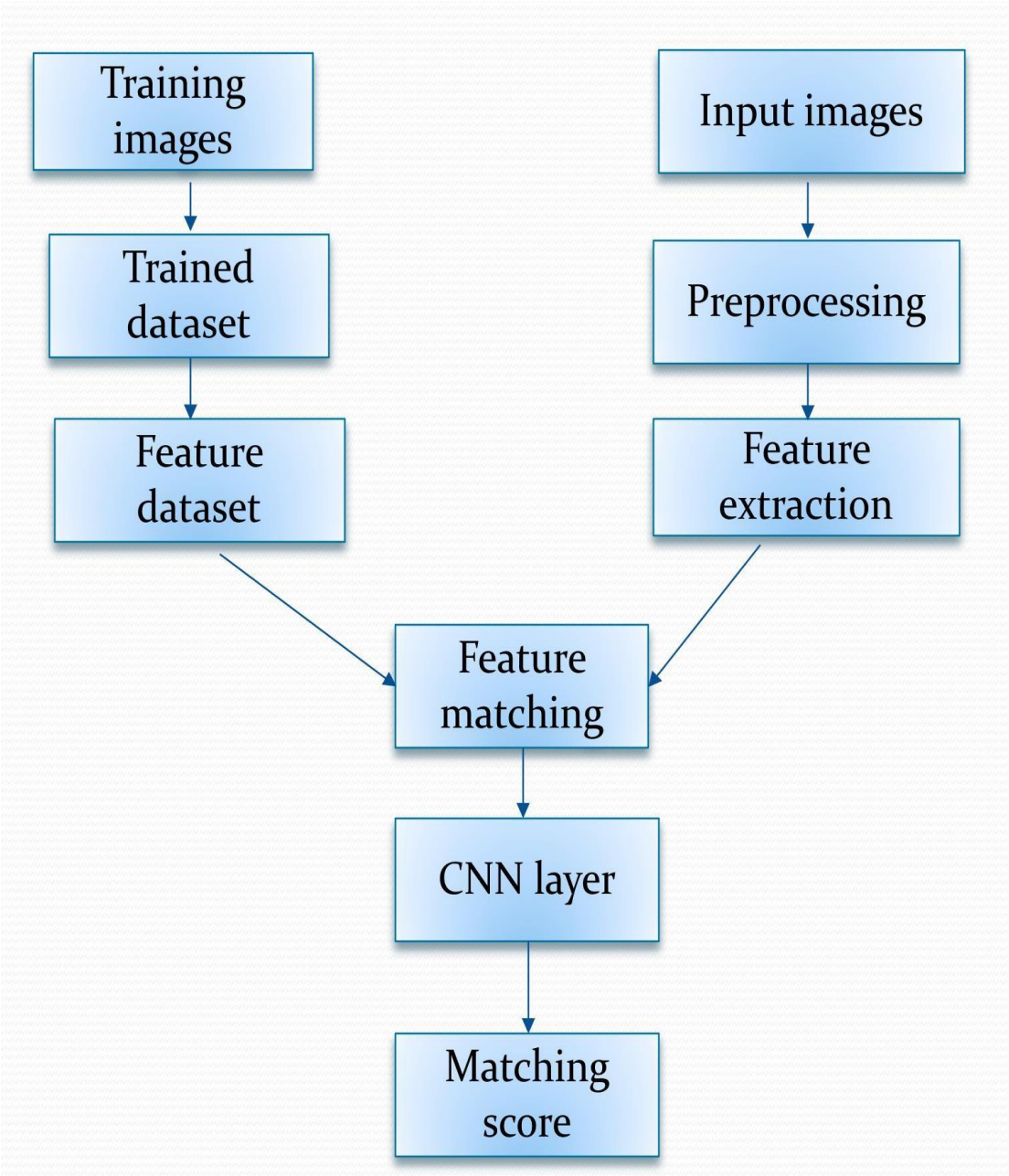


Fig 5.1

5.2 MODULE DESIGN SPECIFICATION

- ✓ Image Pre-Processing
- ✓ Convolution Neural network
- ✓ Prediction and Matching Score

Image pre-processing

Dataset is a primary and significant part that needs to be dealt with for a learning application. Collecting the healthy and affected eye images through OCT [Optical Coherence Tomography] scan. This model accepts two funds images corresponding to the left eye and right eye as inputs. Involving the preprocessing step we have to convert the RGB images to grey scale and resize the image from the collected image.

In training phase, the input of the CNN algorithm will be the preprocessed training data. As the input goes iteratively through each layer (convolutional, pooling, and fully connected), it will identify the features in the images. The layers in the algorithm will find the best features which are needed for classification using feature maps. The algorithm will first classify whether the patient is suffering from disease or not. If the patient is suffering from the disease, then it will check the severity level of his disease. The output for this will be the patient's severity level

The funds photographs in our dataset have large variation, such as discrepant brightness or resolution, since most of them are obtained with different equipment in different environment. Basically, the model accepts two funds images corresponding to the left eye and right eye as inputs and then transmits them into the Siamese-like blocks.

Convolution Neural network

Convolution neural network is a subset of deep learning neural network. It is mainly used for image classification and image analysis. The goal behind CNN is to do how human brain analyzes the image. Convolution neural network is comprised of one or more convolutional layers and then followed by one or more fully connected layers. The convolution layer of CNN will extract features from the source image. The CNN consist of input, hidden and output layer.

The input layer basically consists of arrays of pixels. The hidden layer is the most important layer as it plays the main role in image computation. Hidden layer comprises of activation functions and biases. The output layer helps us to determine the class score. A Convolutional Neural Network can takes image as an input , assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. It Splitting the images into testing set and training set.Training set has larger number of dataset then the training set.

Transfer learning method is a widely used training method of convolution neural network. By loading the weights of Inception blocks pre-trained on Image Net data set, the model will has a better weights initialization before starting the gradient optimization. Moreover, considering the huge difference between the fundus images data set and Image Net dataset, none of layers in weight-sharing Inception blocks are frozen.

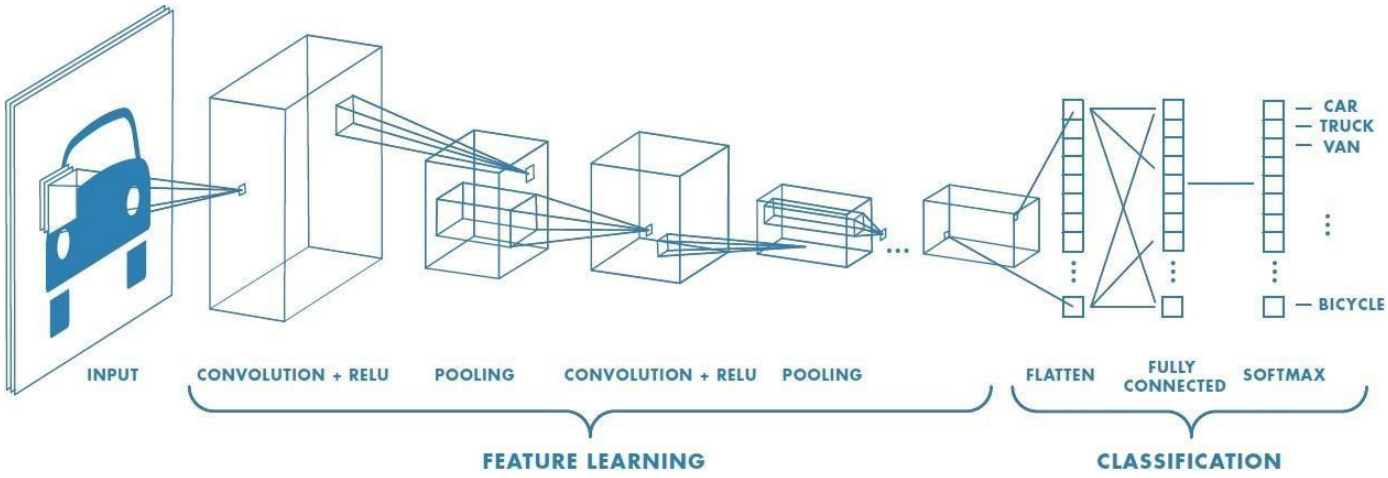


Fig 5.2.1

Prediction and Matching Score

CNN accepts the image as an input and it will test the input images and predict that the patient is suffering from the disease or not. If the patient is suffering from the disease, then it will check the severity level of his disease. The output for this will be the patient's severity level. The confusion matrices of prediction results of left eye, right eye, and both eyes together. The prediction results of the left eye and the right eye have very similar distribution patterns, indicating that the data partition method preserves the original image categories distribution of left eyes and right eyes.

5.3 PROGRAM DESIGN LANGUAGE

Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- ✓ web development (server-side),
- ✓ software development,
- ✓ mathematics
- ✓ System scripting.

What can Python do?

- ✓ Python can be used on a server to create web applications.
- ✓ Python can be used alongside software to create workflows.
- ✓ Python can connect to database systems. It can also read and modify files.
- ✓ Python can be used to handle big data and perform complex mathematics.
- ✓ Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- ✓ Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- ✓ Python has a simple syntax similar to the English language.
- ✓ Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- ✓ Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- ✓ Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- ✓ The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- ✓ Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.
- ✓ Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are

2.7.15 and 3.6.5. However, an official End Of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained.

- ✓ Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.
- ✓ It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- ✓ Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- ✓ Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- ✓ Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python is Interpreted

- ✓ Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can

be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

- ✓ This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.
- ✓ One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
- ✓ In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.
- ✓ For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.
- ✓ Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.
- ✓ Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

Convolutional Neural Networks (CNN) Introduction:

Convolutional neural networks (CNN) sounds like a weird combination of biology and math with a little CS sprinkled in, but these networks have been some of the most influential innovations in the field of computer vision. 2012 was the first year that neural nets grew to prominence as Alex Krizhevsky used them to win that year's ImageNet competition (basically, the annual Olympics of computer vision), dropping the classification error record from 26% to 15%, an astounding improvement at the time. Ever since then, a host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

However, the classic, and arguably most popular, use case of these networks is for image processing. Within image processing, let's take a look at how to use these CNNs for image classification.

The Problem Space

Image classification is the task of taking an input image and outputting a class (a cat, dog, etc) or a probability of classes that best describes the image. For humans, this task of recognition is one of the first skills we learn from the moment we are born and is one that comes naturally and effortlessly as adults. Without even thinking twice, we're able to quickly and seamlessly identify the environment we are in as well as the objects that surround us. When we see an image or just when we look at the world around us, most of the time we are able to

immediately characterize the scene and give each object a label, all without even consciously noticing. These skills of being able to quickly recognize patterns, generalize from prior knowledge, and adapt to different image environments are ones that we do not share with our fellow machines.

Fig 5.3.1

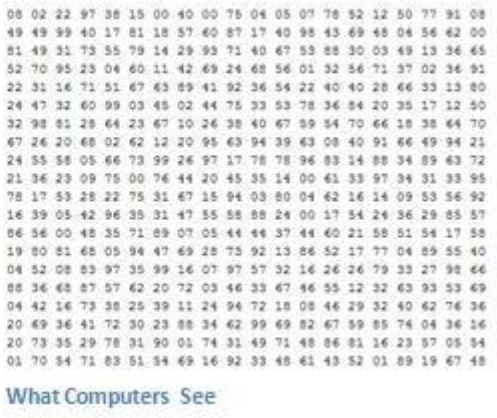


Fig 5.3.2

Inputs and Outputs

When a computer sees an image (takes an image as input), it will see an array of pixel values. Depending on the resolution and size of the image, it will see a 32 x 32 x 3 array of numbers (The 3 refers to RGB values). Just to drive home the point, let's say we have a color image in JPG form and its size is 480 x 480. The representative array will be 480 x 480 x 3. Each of these numbers is given a value from 0 to 255 which describe the pixel intensity at that point. These numbers, while meaningless to us when we perform image classification, are the only inputs available to the computer. The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class (.80 for cat, .15 for dog, .05 for bird, etc).

What We Want the Computer to do

Now that we know the problem as well as the inputs and outputs, let's think about how to approach this. What we want the computer to do is to be able to differentiate between all the images it's given and figure out the unique features that make a dog a dog or that make a cat a cat. This is the process that goes on in our minds subconsciously as well. When we look at a picture of a dog, we can classify it as such if the picture has identifiable features such as paws or 4 legs. In a similar way, the computer is able perform image classification by looking for low level features such as edges and curves, and then building up to more abstract concepts through a series of convolutional layers. This is a general overview of what a CNN does. Let's get into the specifics

Biological Connection

But first, a little background. When you first heard of the term convolutional neural networks, you may have thought of something related to neuroscience or biology, and you would be right. Sort of. CNNs do take a biological inspiration from the visual cortex. The visual cortex has small regions of cells that are sensitive to specific regions of the visual field. This idea was expanded upon by a fascinating experiment by Hubel and Wiesel in 1962 (Video) where they showed that some individual neuronal cells in the brain responded (or fired) only in the presence of edges of a certain orientation. For example, some neurons fired when exposed to vertical edges and some when shown horizontal or diagonal edges.

Hubel and Wiesel found out that all of these neurons were organized in a columnar architecture and that together, they were able to produce visual perception. This idea of specialized components inside of a system having specific tasks (the neuronal cells in the visual cortex

looking for specific characteristics) is one that machines use as well, and is the basis behind CNNs.

Structure

Back to the specifics. A more detailed overview of what CNNs do would be that you take the image, pass it through a series of convolutional, nonlinear, pooling (down sampling), and fully connected layers, and get an output. As we said earlier, the output can be a single class or a probability of classes that best describes the image. Now, the hard part understands what each of these layers do. So let's get into the most important one.

First Layer – Math Part

The first layer in a CNN is always a Convolutional Layer. First thing to make sure you remember is what the input to this conv (I'll be using that abbreviation a lot) layer is. Like we mentioned before, the input is a 32 x 32 x 3 array of pixel values. Now, the best way to explain a conv layer is to imagine a flashlight that is shining over the top left of the image. Let's say that the light this flashlight shines covers a 5 x 5 area. And now, let's imagine this flashlight sliding across all the areas of the input image. In machine learning terms, this flashlight is called a filter (or sometimes referred to as a neuron or a kernel) and the region that it is shining over is called the receptive field. Now this filter is also an array of numbers (the numbers are called weights or parameters). A very important note is that the depth of this filter has to be the same as the depth of the input (this makes sure that the math works out), so the dimension of this filter is 5 x 5 x 3. Now, let's take the first position the filter is in for example. It would be the top left corner. As the filter is sliding, or convolving, around the input image, it is multiplying the values in the filter with the original pixel values of the image (aka computing element wise

multiplications). These multiplications are all summed up (mathematically speaking, this would be 75 multiplications in total). So now you have a single number. Remember, this number is just representative of when the filter is at the top left of the image. Now, we repeat this process for every location on the input volume. (Next step would be moving the filter to the right by 1 unit, then right again by 1, and so on). Every unique location on the input volume produces a number. After sliding the filter over all the locations, you will find out that what you're left with is a 28 x 28 x 1 array of numbers, which we call an activation map or feature map. The reason you get a 28 x 28 array is that there are 784 different locations that a 5 x 5 filter can fit on a 32 x 32 input image. These 784 numbers are mapped to a 28 x 28 array.

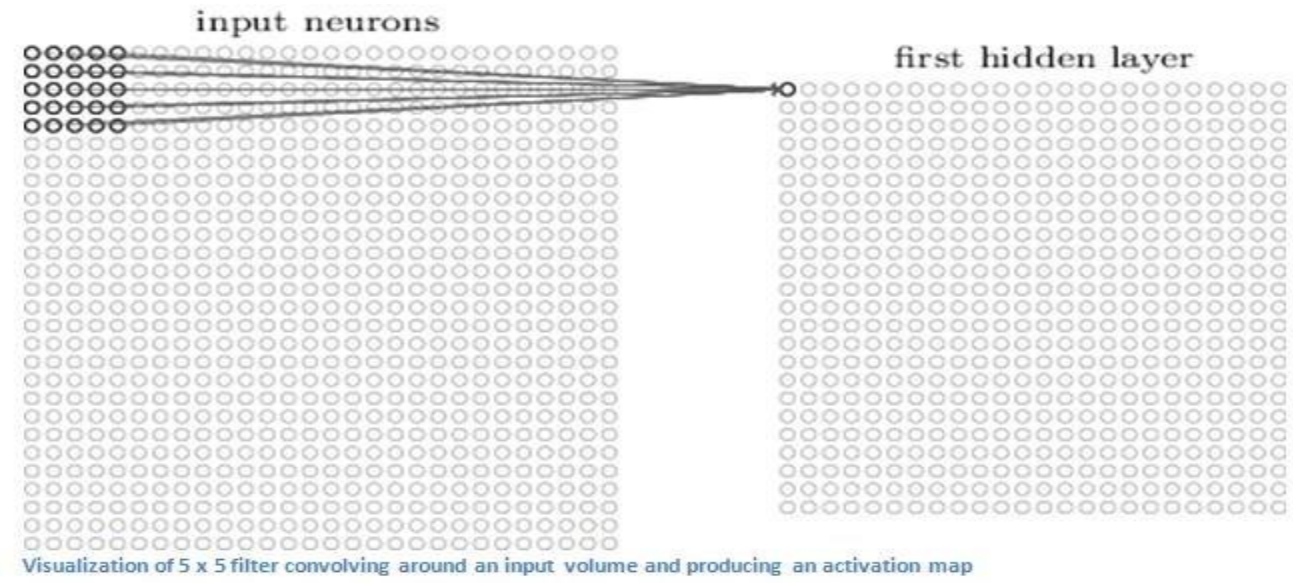


Fig 3.2.3

Let's say now we use two 5 x 5 x 3 filters instead of one. Then our output volume would be 28 x 28 x 2. By using more filters, we are able to preserve the spatial dimensions better. Mathematically, this is what's going on in a convolutional layer.

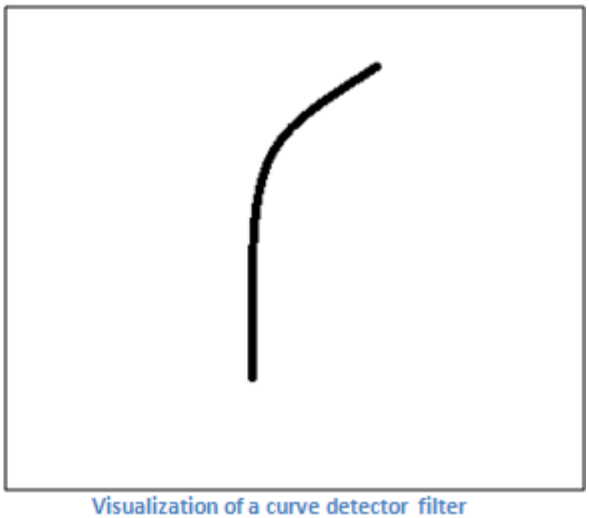
First Layer – High Level Perspective

However, let's talk about what this convolution is actually doing from a high level. Each of these filters can be thought of as **feature identifiers**. When I say features, I'm talking about things like straight edges, simple colors, and curves. Think about the simplest characteristics that all images have in common with each other. Let's say our first filter is 7 x 7 x 3 and is going to be a curve detector. (In this section, let's ignore the fact that the filter is 3 units deep and only consider the top depth slice of the filter and the image, for simplicity.)As a curve detector, the filter will have a pixel structure in which there will be higher numerical values along the area that is a shape of a curve (Remember, these filters that we're talking about as just numbers!).

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

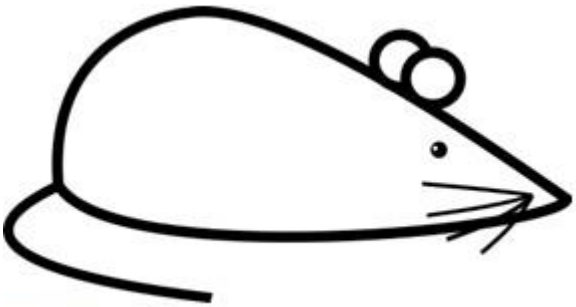
Fig 3.2.4



Visualization of a curve detector filter

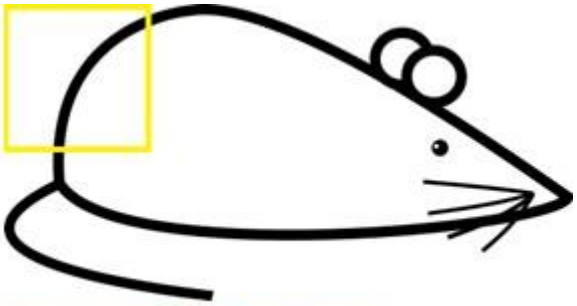
Fig 3.2.5

Now, let's go back to visualizing this mathematically. When we have this filter at the top left corner of the input volume, it is computing multiplications between the filter and pixel values at that region. Now let's take an example of an image that we want to classify, and let's put our filter at the top left corner.



Original image

Fig 3.2.6



Visualization of the filter on the image

Fig 3.2.7

Remember, what we have to do is multiply the values in the filter with the original pixel values of the image.

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Fig 3.2.8

Basically, in the input image, if there is a shape that generally resembles the curve that this filter is representing, then all of the multiplications summed together will result in a large value! Now let's see what happens when we move our filter.

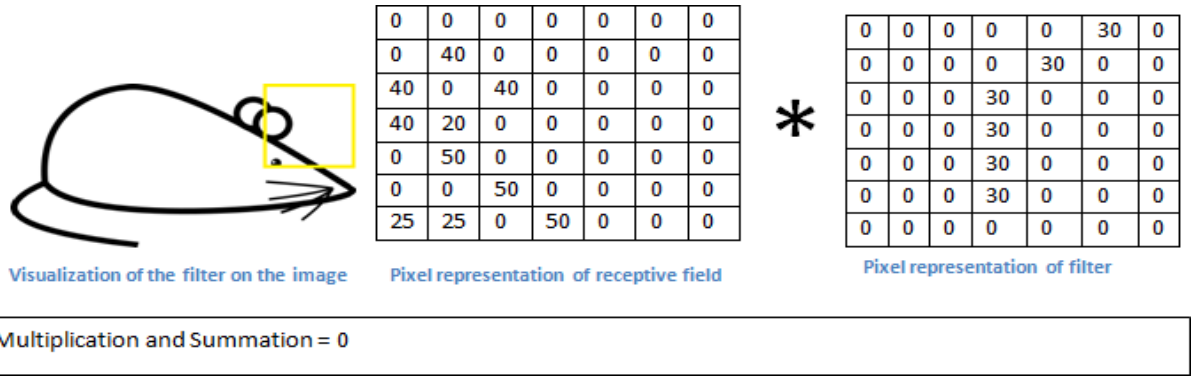


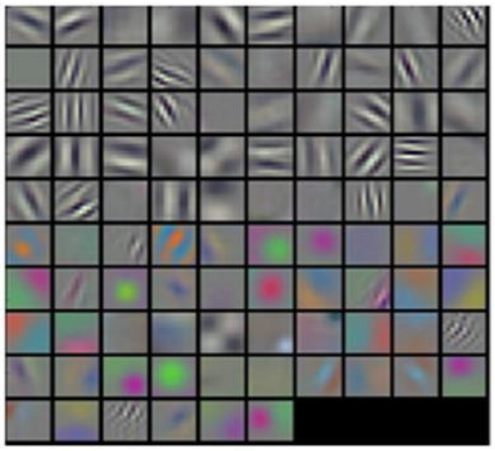
Fig 3.2.9

The value is much lower! This is because there wasn't anything in the image section that responded to the curve detector filter. Remember, the output of this conv layer is an activation map. So, in the simple case of a one filter convolution (and if that filter is a curve detector), the activation map will show the areas in which there at mostly likely to be curves in the picture. In this example, the top left value of our 26 x 26 x 1 activation map (26 because of the 7x7 filter instead of 5x5) will be 6600. This high value means that it is likely that there is some sort of curve in the input volume that caused the filter to activate. The top right value in our activation map will be 0 because there wasn't anything in the input volume that caused the filter to activate (or more simply said, there wasn't a curve in that region of the original image). Remember, this is just for one filter. This is just a filter that is going to detect lines that curve outward and to the right. We can have other filters for lines that curve to the left or for straight edges. The more filters, the greater the depth of the activation map, and the more information we have about the input volume.

Disclaimer:

The filter I described in this section was simplistic for the main purpose of describing the math that goes on during a convolution. In the picture below, you'll see some examples of actual

visualizations of the filters of the first conv layer of a trained network. Nonetheless, the main argument remains the same. The filters on the first layer convolve around the input image and "activate" (or compute high values) when the specific feature it is looking for is in the input volume.



Visualizations of filters

Fig 3.2.10

Going Deeper Through the Network

Now in traditional convolutional neural network architecture, there are other layers that are interspersed between these conv layers. I'd strongly encourage those interested to read up on them and understand their function and effects, but in a general sense, they provide nonlinearities and preservation of dimension that help to improve the robustness of the network and control over fitting. A classic CNN architecture would look like this.

Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool ->Fully Connected

The last layer, however, is an important one and one that we will go into later on. Let's just take a step back and review what we've learned so far. We talked about what the filters in

the first conv layer are designed to detect. They detect low level features such as edges and curves. As one would imagine, in order predicting whether an image is a type of object, we need the network to be able to recognize higher level features such as hands or paws or ears. So let's think about what the output of the network is after the first conv layer. It would be a 28 x 28 x 3 volume (assuming we use three 5 x 5 x 3 filters). When we go through another conv layer, the output of the first conv layer becomes the input of the 2nd conv layer. Now, this is a little bit harder to visualize. When we were talking about the first layer, the input was just the original image. However, when we're talking about the 2nd conv layer, the input is the activation map(s) that result from the first layer. So each layer of the input is basically describing the locations in the original image for where certain low level features appear. Now when you apply a set of filters on top of that (pass it through the 2nd conv layer), the output will be activations that represent higher level features. Types of these features could be semicircles (combination of a curve and straight edge) or squares (combination of several straight edges). As you go through the network and go through more conv layers, you get activation maps that represent more and more complex features. By the end of the network, you may have some filters that activate when there is handwriting in the image, filters that activate when they see pink objects, etc. If you want more information about visualizing filters in ConvNets, Matt Zeiler and Rob Fergus had an excellent research paper discussing the topic. Jason Yosinski also has a video on YouTube that provides a great visual representation. Another interesting thing to note is that as you go deeper into the network, the filters begin to have a larger and larger receptive field, which means that they are able to consider information from a larger area of the original input volume (another way of putting it is that they are more responsive to a larger region of pixel space).

Fully Connected Layer

Now that we can detect these high level features, the icing on the cake is attaching a **fully connected layer** to the end of the network. This layer basically takes an input volume (whatever the output is of the conv or ReLU or pool layer preceding it) and outputs an N dimensional vector where N is the number of classes that the program has to choose from. For example, if you wanted a digit classification program, N would be 10 since there are 10 digits. Each number in this N dimensional vector represents the probability of a certain class. For example, if the resulting vector for a digit classification program is [0 .1 .1 .75 0 0 0 0 0 .05], then this represents a 10% probability that the image is a 1, a 10% probability that the image is a 2, a 75% probability that the image is a 3, and a 5% probability that the image is a 9 (Side note: There are other ways that you can represent the output, but I am just showing the softmax approach). The way this fully connected layer works is that it looks at the output of the previous layer (which as we remember should represent the activation maps of high level features) and determines which features most correlate to a particular class. For example, if the program is predicting that some image is a dog, it will have high values in the activation maps that represent high level features like a paw or 4 legs, etc. Similarly, if the program is predicting that some image is a bird, it will have high values in the activation maps that represent high level features like wings or a beak, etc. Basically, a FC layer looks at what high level features most strongly correlate to a particular class and has particular weights so that when you compute the products between the weights and the previous layer, you get the correct probabilities for the different classes.

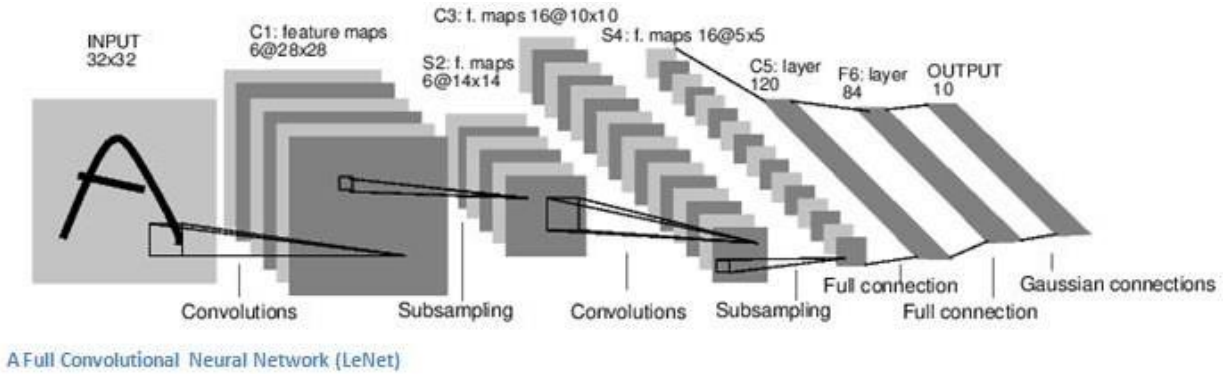


Fig 3.2.11

Training (AKA: What Makes this Stuff Work)

Now, this is the one aspect of neural networks that I purposely haven't mentioned yet and it is probably the most important part. There may be a lot of questions you had while reading. How do the filters in the first conv layer know to look for edges and curves? How does the fully connected layer know what activation maps to look at? How do the filters in each layer know what values to have? The way the computer is able to adjust its filter values (or weights) is through a training process called **backpropagation**.

Before we get into backpropagation, we must first take a step back and talk about what a neural network needs in order to work. At the moment we all were born, our minds were fresh. We didn't know what a cat or dog or bird was. In a similar sort of way, before the CNN starts, the weights or filter values are randomized. The filters don't know to look for edges and curves. The filters in the higher layers don't know to look for paws and beaks. As we grew older however, our parents and teachers showed us different pictures and images and gave us a corresponding label.

This idea of being given an image and a label is the training process that CNNs go through. Before getting too into it, let's just say that we have a training set that has thousands of images of dogs, cats, and birds and each of the images has a label of what animal that picture is. Back to backprop.

So backpropagation can be separated into 4 distinct sections, the forward pass, the loss function, the backward pass, and the weight update. During the **forward pass**, you take a training image which as we remember is a 32 x 32 x 3 array of numbers and pass it through the whole network. On our first training example, since all of the weights or filter values were randomly initialized, the output will probably be something like [.1 .1 .1 .1 .1 .1 .1 .1 .1 .1], basically an output that doesn't give preference to any number in particular. The network, with its current weights, isn't able to look for those low level features or thus isn't able to make any reasonable conclusion about what the classification might be. This goes to the **loss function** part of backpropagation. Remember that what we are using right now is training data. This data has both an image and a label. Let's say for example that the first training image inputted was a 3. The label for the image would be [0 0 0 1 0 0 0 0 0 0]. A loss function can be defined in many different ways but a common one is MSE (mean squared error), which is ½ times (actual - predicted) squared.

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Let's say the variable L is equal to that value. As you can imagine, the loss will be extremely high for the first couple of training images. Now, let's just think about this intuitively. We want to get to a point where the predicted label (output of the ConvNet) is the same as the

training label (This means that our network got its prediction right).In order to get there, we want to minimize the amount of loss we have. Visualizing this as just an optimization problem in calculus, we want to find out which inputs (weights in our case) most directly contributed to the loss (or error) of the network.

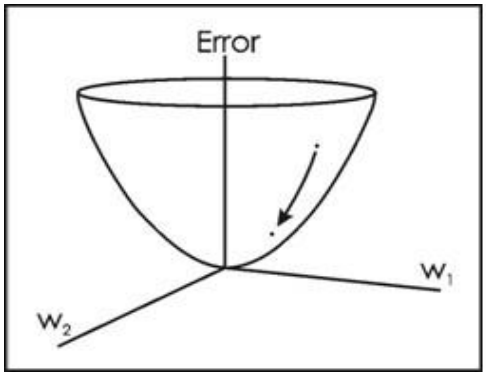


Fig 3.2.12

One way of visualizing this idea of minimizing the loss is to consider a 3-D graph where the weights of the neural net (there are obviously more than 2 weights, but let's go for simplicity) are the independent variables and the dependent variable is to loss. The task of minimizing the loss involves trying, we want to get to the lowest point in our bowl shaped object. To do this we have to take a derivation of the loss (visual terms: calculate the slope in every direction) with respect to the weights

This is the mathematical equivalent of a **dL/dW** where W are the weights at a particular layer. Now, what we want to do is perform a **backward pass** through the network, which is determining which weights contributed most to the loss and finding ways to adjust them so that the loss decreases. Once we compute this derivative, we then go to the last step which is the **weight update**. This is where we take all the weights of the filters and update them so that they change in the opposite direction of the gradient.

$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
w_i = Initial Weight
η = Learning Rate

The **learning rate** is a parameter that is chosen by the programmer. A high learning rate means that bigger steps are taken in the weight updates and thus, it may take less time for the model to converge on an optimal set of weights. However, a learning rate that is too high couldresult in jumps that are too large and not precise enough to reach the optimal point.

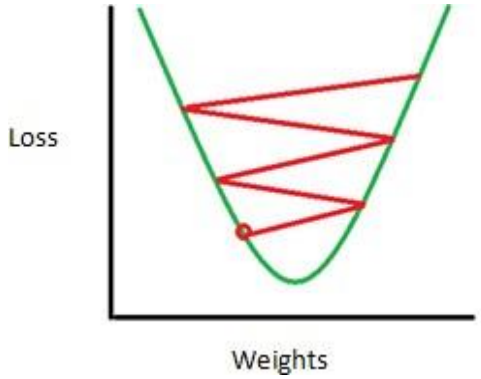


Fig 3.2.13

Consequence of a high learning rate where the jumps are too large and we are not able to minimize the loss

The process of forward pass, loss function, backward pass, and parameter update is one training iteration. The program will repeat this process for a fixed number of iterations for each set of training images (commonly called a batch). Once you finish the parameter update on the last training example, hopefully the network should be trained well enough so that the weights of the layers are tuned correctly.

Testing

Finally, to see whether or not our CNN works, we have a different set of images and

labels (can't double dip between training and test!) and pass the images through the CNN. We compare the outputs to the ground truth and see if our network works!

How Companies Use CNNs

Data, data, data. The companies that have lots of this magic 4 letter word are the onesthat have an inherent advantage over the rest of the competition. The more training data that you can give to a network, the more training iterations you can make, the more weight updates you can make, and the better tuned to the network is when it goes to production. Facebook (and Instagram) can use all the photos of the billion users it currently has, Pinterest can use information of the 50 billion pins that are on its site, Google can use search data, and Amazon can use data from the millions of products that are bought every day. And now you know the magic behind how they use it.

Disclaimer

While this post should be a good start to understanding CNNs, it is by no means a comprehensive overview. Things not discussed in this post include the nonlinear and pooling layers as well as hyperparameters of the network such as filter sizes, stride, and padding. Topics like network architecture, batch normalization, vanishing gradients, dropout, initialization techniques, non-convex optimization, biases, and choices of loss functions, data augmentation, regularization methods, computational considerations, modifications of backpropagation, and more were also not discussed (yet).

Methodology

This section is divided in two major parts that is image processing of data and training that neural network with the processed image. The images are taken from Kaggle, where all the images are in one zip file. The images have been further classified in different levels of Diabeticretinopathy for the neural network

Image classification:

A python program has been written that takes labeled class name image input from a excel sheet and copies that image to the folder that have same class name to which that image belongs. Fig 1: Snippet of train file. In the above figure there is an image table containing image name and level. The table indicates that the image belongs to which level of diabetic retinopathy the python program classifies image.

Image resizes:

All the fundus images taken for training have 4928x3264 resolution but will be resized down to 1024x720 resolution. Python's opencv2 library has been used to resize the image, setting quality parameter as 95% Fundus image: Fundus photography involves capturing a photograph of the back of the eye i.e. fundus. Specialized fundus cameras that consist of an intricate microscope attached to a flash enabled camera are used in fundus photography. The main structures that can be visualized on a fundus photo are the central and peripheral retina, optic disc and macula. Fundus photography can be performed with colored filters, or with specialized dyes including fluoresces in and indocyanine green following figure shows the

fundus image of the eye. This image is further processed to check whether it is a fundus image ornot. The proposed system operates as a web app where the user can upload any kind of image and can fool the neural network. Hence to avoid this dilemma a filter has to be applied that will only allow fundus image to be passed to neural network. The filter basically uses opencv2 python library which uses Scale-Invariant Feature Transform (SIFT) algorithm to compute the key points and the key point description of the image. Key point descriptor is a 16x16 matrix that describes the neighborhood around the key point

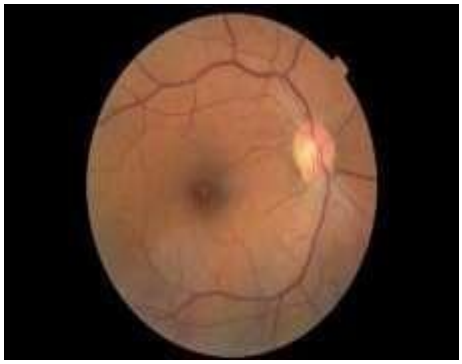


Fig-4: Fundus Image of eye.

On acquisition of key points and the key point descriptor of user entered image as well as the predefined fundus image, the next step is the comparison of those key points and key point descriptor using FLANN based matcher. Fast Library for Approximate Nearest Neighbors (FLANN) uses the key point neighborhood to find the match between images.

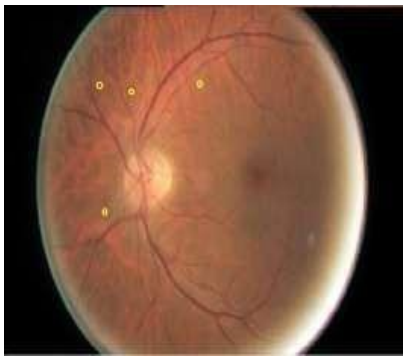


Fig-5: Key points of the image

The key point descriptor has been passed to this algorithm, as it stores key point neighborhood in 16x16 matrixes. The FLANN's knnmatcher method is used to find matches between two images which will take good ratio test to find best matches. If the number of matches are 0 then image is not a fundus image whereas if the number of matches are greater than 0 the image is a fundus image. On confirmation, that the image is a fundus image it is then passed to the neural network. The architecture of the neural network is represented as follows: Layer Name Layer Specification Layer Activation Function Input Layer 64x64x3 Null Convolution2D-1 32x(3x3) ReLU MaxPooling2D-1 2x2 Null Convolution2D-2 32x(3x3) ReLU MaxPooling2D-1 2x2 Null Flatten Null Dense-1 128 units ReLU Dropout-1 20% Null Dense-2 128 units ReLU Dropout-2 20% Null Output 5 units Softmax

Table 2 layer Activities

Layer Name	**Layer Specification**	**Layer Activation Function**
Input Layer	64x64x3	Null
Convolution2D-1	32x(3x3)	ReLU
MaxPooling2D-1	2x2	Null
Convolution2D-2	32x(3x3)	ReLU
MaxPooling2D-1	2x2	Null
Flatten	Null	Null
Dense-1	128 units	ReLU
Dropout-1	20%	Null
Dense-2	128 units	ReLU
Dropout-2	20%	Null
Output	5 units	Softmax

- **Convolutional Layer**

  The first layer is convolutional layer, this layer performs heavy computation which make further job easy. This layer works as an input layer taking 128x128x3(i.e. 128 pixels width and height, and 3 because images have depth 3, the color channels) as the input size of the image. Filters of 3x3 matrixes will slide over all the spatial locations. The convolution layer comprises a set of independent filters. Each filter is independently convolved with the image resulting in 32 feature maps.

- **Max-pooling Layer**

  In Max-pooling layer highest weighted feature is extracted, this is achieved by converting above 3x3 matrixes in more compressed matrix. The above 3x3 matrix is the converted into 2x2 matrixes which involves only the highest weighted feature that is present in 3x3 matrixes.

- **Flatten Layer**

  Flatten layer converts the matrix of image into one single dimension array which acts as an input to the dense layer.

- **Dropout Layer**

  The dropout layer performs inexpensive and powerful operation that highly improve generalization abilities of the neural network. This method involves randomly removing and restoring neurons during the training with a probability determined by the hyperparameters called dropout rate.

- **Dense Layer**

  The fully connected layer has its neurons connected to all neurons in the previous layer. These layers are used as last elements of deep neural classifier, which are feed by the features extracted by the successive convolutional layers.

- **Output layer**

  The last layer that produces the output of the network is a softmax layer or sigmoid neuron, depending on the solving task - binary or multiclass classification.

Activation Functions

1. ReLU: Rectified Linear Unit (ReLU) is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning.

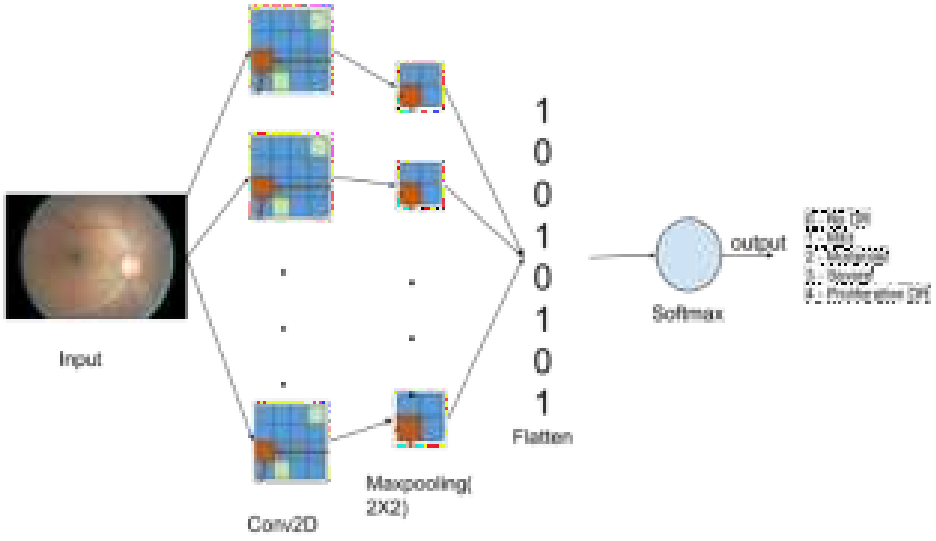


Fig-6: Neural Network Architecture

As you can see in the above figure, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero. 2. Softmax: The softmax function is a more generalized logistic activation function which is used for multiclass classification.

$$f(x) = x^+ = \max(0, x)$$

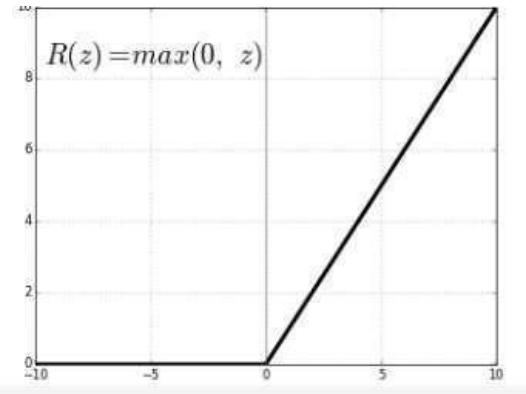


Fig-7: Graphical representation of ReLU.

Softmax:

The softmax function is a more generalized logistic activation function which is used for multiclass classification.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

Preprocessing

Initially the original fundus images are resized to a dimension of 336 x 448. Due to the immense information and varying contrast of images obtained from the fundus cameras preprocessing is necessary. Without preprocessing the images suffer from vignetting effects and image distortion. Since the images are obtained from different fundus cameras they will be having non uniform illumination, illumination normalization techniques have to be incorporated.

Green channel extraction

Every image consists of three channels namely red, green and blue. Green channel is extracted from the fundus image. Extraction of green channel provides better contrast between maximum and minimum intensities in an image. It has less noise compared to that of red or blue channels. Blue channel is not normally considered because it provides less contrast and does not contain much information.

Adaptive Histogram Equalization

This image processing technique improves the contrast of the image. It is usually performed on small regions of the image called tiles. This helps in enhancing the edges of the image.

Optic disc removal

Automatic Optic Disc (OD) removal is a vital step because optic disc act as false positives as they greatly resemble exudates in intensity. Firstly Images are converted into grayscale whose intensity values ranges between 0 to 255 and then thresholding is done. The pixels that have intensities lower than the threshold were converted to 0 (black), while pixels above the threshold were converted to 255 (white). The dilate-erode combinations are used to remove unconnected small regions while preserving large unconnected regions.

This provides an initial region from which the segmentation process begins. A disk structuring element of size 30 is applied over the image and an OFF pixel is set to ON if any of the structuring elements overlap ON pixels of the image. The threshold is continuously updated and this process is repeated until only one region remains.

Extraction of region proposals

Region of interest is extracted using 2 methods by means of thresholding and blocking (Pixel based) [14]. Based on intensity differences the region of interests (i.e., those segments which contain lesions) are extracted. For further processing only these segments are considered.

Threshold based

Regions are extracted from the preprocessed image by thresholding the intensity values of red lesions and exudates separately [13]. For thresholding, lab color space is used since it is the most accurate way of representing colors. It is a 3 axis color space in which L represents lightness, a & b represents color dimensions.

Fig-8. Thresholding

Pixel based

Images are divided into blocks equal segments of size 128 x 128 pixels and the blocks which are suspected to have signs of DR are extracted. Only these extracted blocks are considered for training.

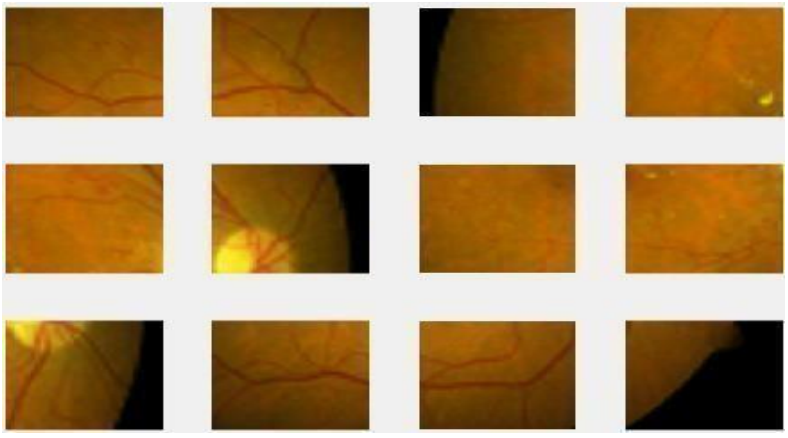


Fig-9 Pixel based blocking

CNN classification

- ✓ The number of input layer neurons is equal to the number of pixels in the input image. Convolutional layer makes use of the convolutional features and computes the product between the image patches and the filter.

- ✓ For the activation layer ReLU (Rectified Linear Unit) can be used. ReLU layer perform a threshold operation to each element of the input where any value less than zero is set to zero. Element wise activation is applied to the output of the convolutional layer.

- ✓ The function of the pooling layer is to down convert the volume so as to make the computation faster and to reduce the memory requirement.

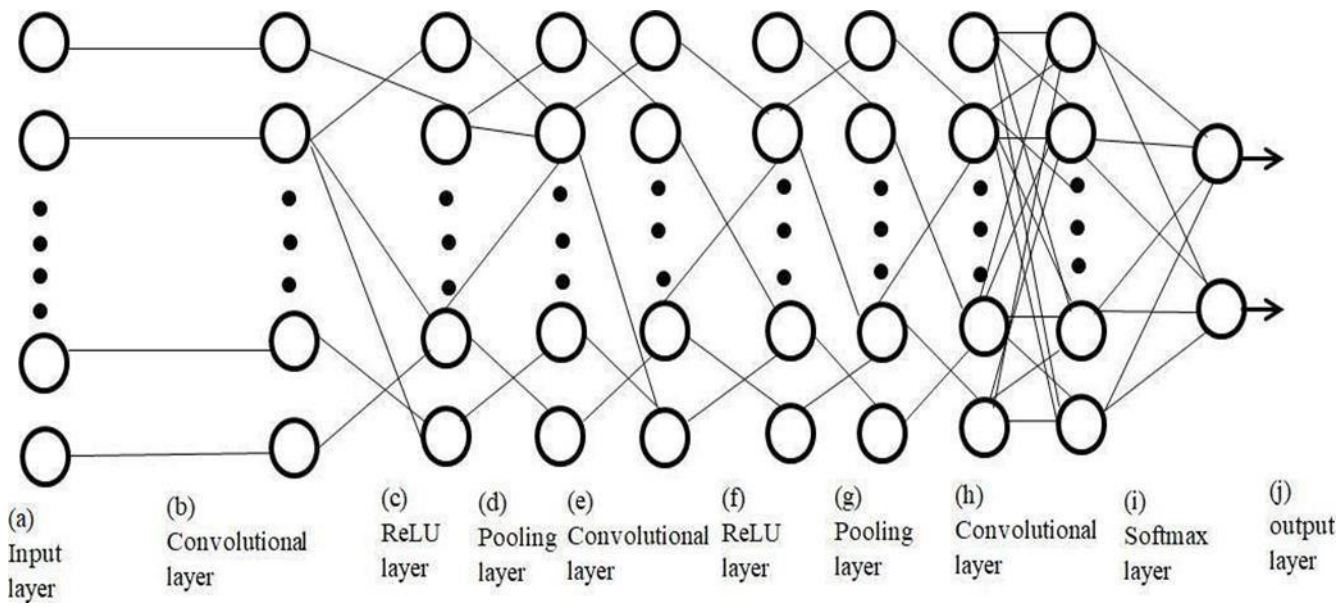
- ✓ The convolutional layer that is present just before the fully connected layer holds the information about the features including the edges, contrast, blobs and shapes. And the fully connected layer contains the aggregated information from the previous layers.

- ✓ A softmax layer is used as the final layer of CNN. It assigns decimal probabilities to each class. The output layer classifies the input images into 2 classes namely normal (without DR) and abnormal (with DR).

- ✓ Fully connected layer takes the output of all the neurons from the former layer. The designed layers are used for training and testing of images.

- ✓ The images for training can be obtained from IDRiD (Indian Diabetic Retinopathy Image Dataset) which provides marked images of Diabetic lesions.

- ✓ In our method we have used 10 layers for R-CNN, trained it on 130 fundus images and tested on 110 images. All the images were classified into two groups i.e., with DR and without DR


(a) Input layer
(b) Convolutional layer
(c) ReLU layer
(d) Pooling layer
(e) Convolutional layer
(f) ReLU layer
(g) Pooling layer
(h) Convolutional layer
(i) Softmax layer
(j) output layer

6. SYSTEM IMPLEMENTATION DETAILS

Technologies used:

Python 3.5.4: Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

Keras library:

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

Flask library:

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

Extensions exist for object-relational mappers, form validation, and upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program. Flask is commonly used with MongoDB, which gives it more control over databases and history. HTML, CSS and JavaScript

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Results:

We have created a webapp for the client where the client can use its username and password credentials to login in our system and use dashboard to interact with our system.

Fig 11 first page

Test case:

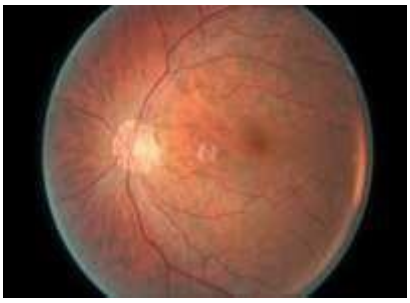


Fig- 12: Fundus image of eye

Figure 6 is test case used for prediction of diabetic retinopathy

Dashboard:

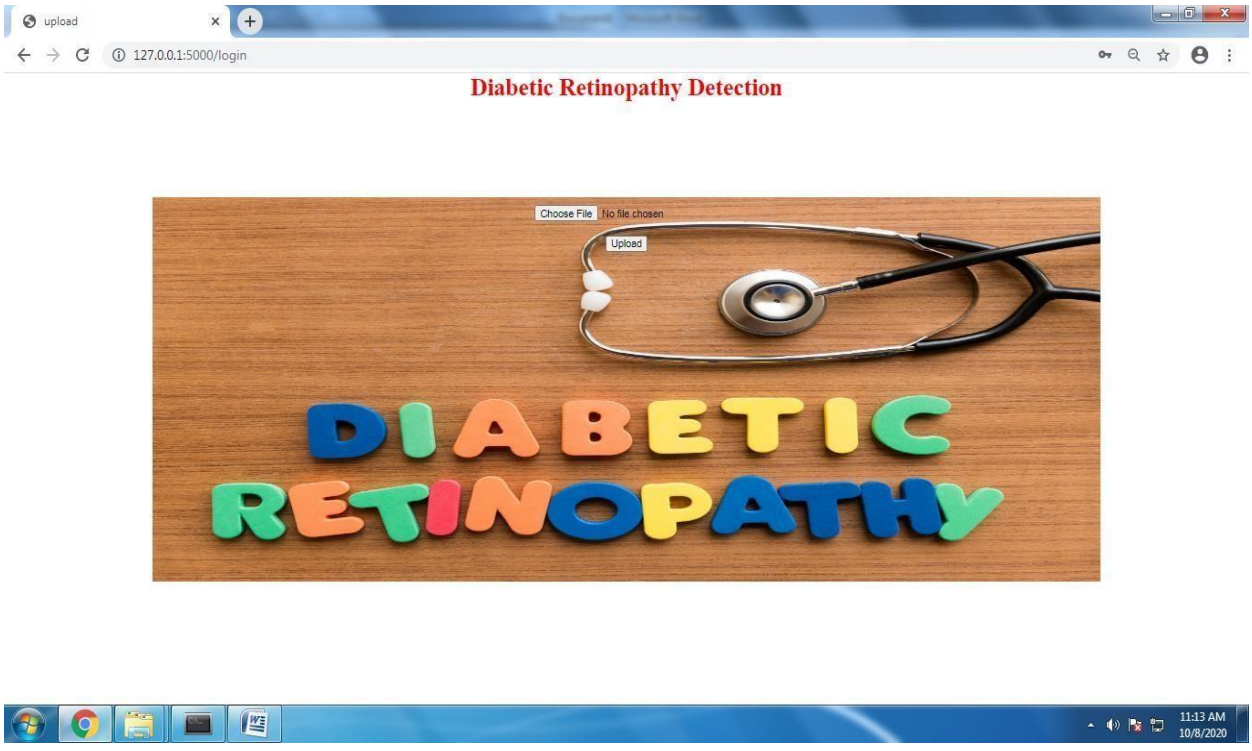


Fig-13: Dashboard Form

Depicts the form where client will fill the patients detail along with its fundus image which can be chosen using browse option. Once all the details are filed this will sent to the server where over neural network will pick the image using patient id

Prediction:

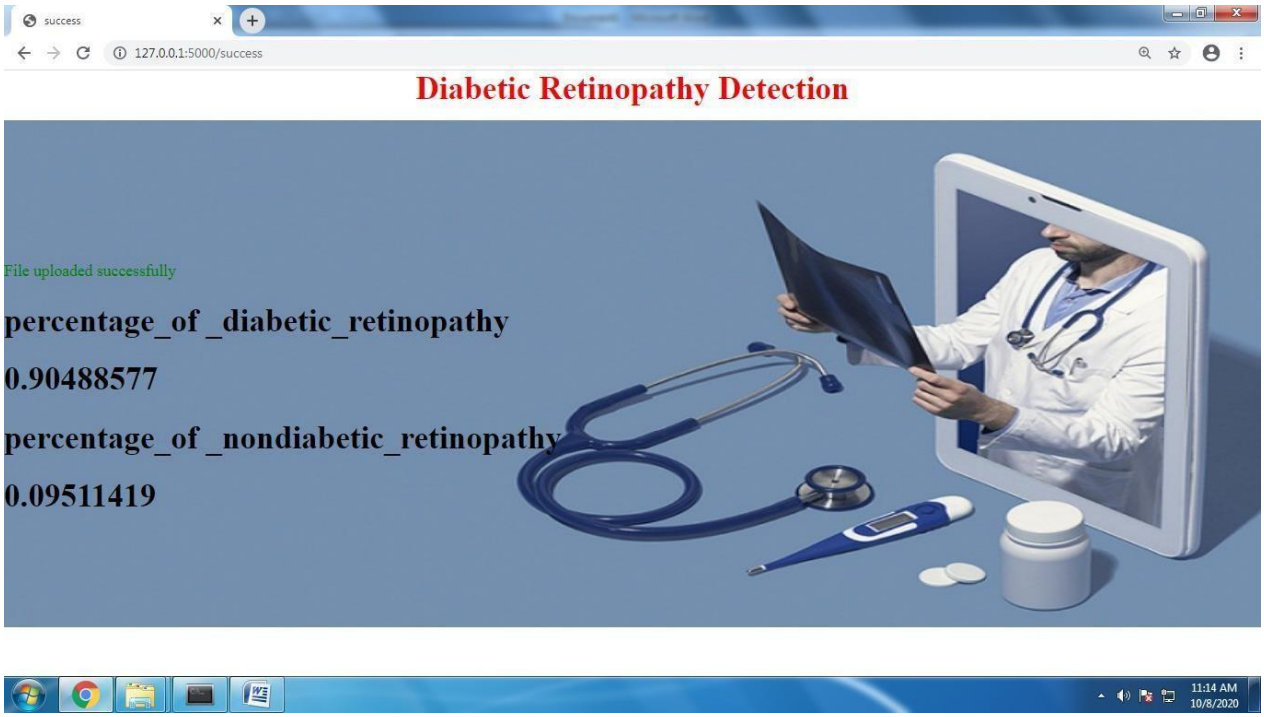


Fig 14 result

Depicts the result that is predicted by neural network. The right side image is the image that is uploaded by client and the result of the neural network is displayed in Output of image. In the above figure we can see that patient's eye is not effected and result is displayed as No Diabetic Retinopathy Detected by our neural network.

PERFORMANCE PARAMETERS

The parameters that are used to evaluate the performance of the proposed model are accuracy, sensitivity, specificity, false positive rate and precision. If P is the number of positive samples (patient with DR), N is the number of negative samples (patient without DR). True

Positive (TP) is the number of patients with DR which is correctly classified. False Positive (FP) is the number of patients without DR which is incorrectly classified. True Negative (TN) is the number of patients without DR which is correctly classified False Negative (FN) is the number of patients with DR which is incorrectly classified. Accuracy = (1) Sensitivity = (2) Specificity = (3) False positive rate = (4) Precision = (5)

EXPERIMENTAL RESULTS

The experiment was done on the images obtained from IDRiD online database along with 50 images collected from the regional hospitals in Kerala to form a total of 240 images. After preprocessing, and candidate region extraction, the images were trained with the CNN network which consisted of 10 layers. The training was done on 130 images for 30 epochs. Testing with the trained network resulted in an accuracy of 93.8 percent.

In this section we evaluate the performance and results RCNN techniques. The performance is evaluated by the various performance parameters like accuracy, sensitivity and speed. Input layer takes in a whole color image in CNN and only suspected blocks in R-CNN. While thresholding the intensity of suspected blocks, the range of L component is taken to be greater than 80, a component to be less than -5 and b component to be greater than 70. First convolution layer contains a 7 x 7 kernel with a stride of 2. ReLU acts as the activation function for the previous layer's output.

6.1 CODING

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand.

Naming conventions

Value conventions

Script and comment procedure

Message box format

Exception and error handling

NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the

conventions. These conventions are as follows:

Class names

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

Member Function and Data Member name

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.
- Proper documentation of flag values.

SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

MESSAGE BOX FORMAT

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

6.1.1 CLIENT-SIDE CODING

```
from flask import  *
app = Flask(__name__)
import tensorflow as tf, sys
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

import os
import sqlite3 as sql
import base64
from flask  import Flask ,render_template,request,jsonify,session
from flask import Flask,abort,render_template,request,redirect,url_for
#from werkzeug import secure_filename
#Flask Libraries
from flask import Flask, redirect, url_for, render_template, request
import requests
import urllib
@app.route('/', methods=['GET', 'POST'])
def home():
    return render_template('index.html')

def validate(username,password):
    con = sql.connect('static/chat.db')
    completion = False
    with con:
        cur = con.cursor()
        cur.execute('SELECT * FROM persons')
        rows = cur.fetchall()
        for row in rows:
            dbuser = row[1]
            dbpass = row[2]
            if dbuser == username:
                completion = (dbpass == password)
    return completion


@app.route('/login', methods=['GET', 'POST'])
def login():
```

```
    error = None
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        completion = validate(username,password)
        if completion == False:
            error = 'invalid Credentials. please try again.'
        else:
            session['username'] = request.form['username']
            return render_template('file_upload_form.html')
    return render_template('file_upload_form.html', error=error)


@app.route('/register', methods = ['GET','POST'])
def register():
    if request.method == 'POST':
        try:
            name = request.form['name']
            username = request.form['username']
            password = request.form['password']
            with sql.connect("static/chat.db") as con:
                cur = con.cursor()
                cur.execute("INSERT INTO persons(name,username,password) VALUES
(?,?,?)",(name,username,password))
                con.commit()
                msg = "Record successfully added"
        except:
            con.rollback()
            msg = "error in insert operation"
        finally:
            return render_template("index.html",msg = msg)
            con.close()
    return render_template('register.html')


@app.route('/list')
def list():
   con = sql.connect("static/chat.db")
   con.row_factory = sql.Row
```

```
    cur = con.cursor()
    cur.execute("select * from persons")

    rows = cur.fetchall();
    return render_template("list.html",rows = rows)


@app.route('/upload')
def upload():
    return render_template("file_upload_form.html")

@app.route('/success', methods = ['POST'])
def success():
    if request.method == 'POST':
        f = request.files['file']
        # change this as you see fit
        #image_path = sys.argv[1]
        image_path =f.filename

        # Read in the image_data
        image_data = tf.gfile.FastGFile(r"C:/Users/Charu/ADR/M3/test/"+str(image_path),
'rb').read()

        # Loads label file, strips off carriage return
        label_lines = [line.rstrip() for line
                       in tf.gfile.GFile("models/retrained_labels.txt")]

        # Unpersists graph from file
        with tf.gfile.FastGFile("models/retrained_graph.pb", 'rb') as f:
            graph_def = tf.GraphDef()
            graph_def.ParseFromString(f.read())
            _ = tf.import_graph_def(graph_def, name='')

        with tf.Session() as sess:
            # Feed the image_data as input to the graph and get first prediction
            softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

            predictions = sess.run(softmax_tensor,
```

```
                {'DecodeJpeg/contents:0': image_data})

            # Sort to show labels of first prediction in order of confidence
            top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]
            print(top_k)
            m=[]
            for node_id in top_k:
                human_string = label_lines[node_id]
                score = predictions[0][node_id]
                for j in human_string,score:
                    print(j)
                    m.append(j)

                #print('%s (score = %.5f)' % (human_string, score))


        return render_template("success.html", list2 = m)

if__name__== '_main_':
    app.run(debug = True)
```

6.1.2 Server side coding

```
#!/usr/bin/env python
# coding: utf-8
# # Import libraries
# In[1]:


import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
import cv2


# In[2]:
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
import itertools


# # Store every image in an array
# In[3]:


import os
data_dir = (r'.\testing_images')
categories = ['no', 'yes']
for i in categories:
    path = os.path.join(data_dir, i)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img))


#
# # Resize each image to same size for fast processing
# In[5]:


img_size = 128
image_array = cv2.resize(img_array, (img_size,img_size))


# In[6]:


dno = cv2.imread('./testing_images/no/15_right.jpeg')
```

```
dyes = cv2.imread('./testing_images/yes/352_left.jpeg')


# In[7]:


plt.rcParams["figure.figsize"] = (5,5)
plt.imshow(dno)
plt.axis('off')


# In[8]:


plt.rcParams["figure.figsize"] = (5,5)
plt.imshow(dyes)
plt.axis('off')


# # Convert each image to grayscale and append into an array
# In[10]:


train_data = []

for i in categories:
    train_path = os.path.join(data_dir,i)
    tag = categories.index(i)
    for img in os.listdir(train_path):
        try:
            image_arr = cv2.imread(os.path.join(train_path , img), cv2.IMREAD_GRAYSCALE)
            new_image_array = cv2.resize(image_arr, (img_size,img_size))
            train_data.append([new_image_array , tag])
        except Exception as e:
            pass


# # Split the features and target in to X and y
# In[11]:
```

```
X = []
y = []
for i,j in train_data:
    X.append(i)
    y.append(j)
X = np.array(X).reshape(-1,img_size,img_size)
print(X.shape)
X = X/255.0
X = X.reshape(-1,128,128,1)


# # One-Hot encode the target column
# In[12]:


from keras.utils.np_utils import to_categorical

y_enc = to_categorical(y, num_classes = 4)


# In[13]:


X_train , X_test, y_train, y_test = train_test_split(X , y_enc , test_size = 0.1, random_state = 42)
X_train , X_val, y_train, y_val = train_test_split(X_train , y_train , test_size = 0.1, random_state
= 42)


# In[14]:


from sklearn.metrics import confusion_matrix
import itertools

from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
```

```
from keras.optimizers import RMSprop,Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from keras.callbacks import EarlyStopping

model = Sequential()


model.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same', activation ='relu', input_shape = (128,128,1)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(filters = 128, kernel_size = (2,2),padding = 'Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.2))


model.add(Conv2D(filters = 256, kernel_size = (2,2),padding = 'Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(1024, activation = "relu"))
model.add(Dropout(0.5))

model.add(Dense(4, activation = "softmax"))

optimizer = Adam(lr=0.001, beta_1=0.9, beta_2=0.999)
```

```
model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accuracy"])

epochs = 20

es = EarlyStopping(
   monitor='val_acc',
   mode='max',
   patience = 3
)

batch_size = 16
imggen = ImageDataGenerator(
       featurewise_center=False,
       samplewise_center=False,
       featurewise_std_normalization=False,
       samplewise_std_normalization=False,
       zca_whitening=False,
       rotation_range=0,
       zoom_range = 0,
       width_shift_range=0,
       height_shift_range=0,
       horizontal_flip=True,
       vertical_flip=False)


# In[15]:


from keras.callbacks import ModelCheckpoint

checkpoint =  ModelCheckpoint("models/model_weights.h5", monitor='val_acc',  verbose=1,
save_best_only=True, mode='max')
callbacks_list = [checkpoint]

imggen.fit(X_train)
history = model.fit_generator(imggen.flow(X_train,y_train,batch_size = batch_size),
                        epochs = epochs, validation_data = (X_val,y_val),
                        steps_per_epoch = X_train.shape[0] // batch_size,
                        callbacks=callbacks_list)
```

```
# In[16]:


# serialize model structure to JSON
model_json = model.to_json()
with open(r".\models\model.json", "w") as json_file:
    json_file.write(model_json)


# In[17]:


os.listdir(r"testing_images")


# In[18]:


import keras
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers.normalization import BatchNormalization


# In[19]:


# Initializing the CNN
classifier = Sequential()

# Convolution Step 1
classifier.add(Convolution2D(96, 11, strides = (4, 4), padding = 'valid', input_shape=(224, 224,
3), activation = 'relu'))
```

```
# Max Pooling Step 1
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
classifier.add(BatchNormalization())

# Convolution Step 2
classifier.add(Convolution2D(256, 11, strides = (1, 1), padding='valid', activation = 'relu'))

# Max Pooling Step 2
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding='valid'))
classifier.add(BatchNormalization())

# Convolution Step 3
classifier.add(Convolution2D(384, 3, strides = (1, 1), padding='valid', activation = 'relu'))
classifier.add(BatchNormalization())

# Convolution Step 4
classifier.add(Convolution2D(384, 3, strides = (1, 1), padding='valid', activation = 'relu'))
classifier.add(BatchNormalization())

# Convolution Step 5
classifier.add(Convolution2D(256, 3, strides=(1,1), padding='valid', activation = 'relu'))

# Max Pooling Step 3
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
classifier.add(BatchNormalization())

# Flattening Step
classifier.add(Flatten())

# Full Connection Step
classifier.add(Dense(units = 4096, activation = 'relu'))
classifier.add(Dropout(0.4))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 4096, activation = 'relu'))
classifier.add(Dropout(0.4))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 1000, activation = 'relu'))
classifier.add(Dropout(0.2))
```

```
classifier.add(BatchNormalization())
classifier.add(Dense(units = 38, activation = 'softmax'))
classifier.summary()



# In[20]:



# change this as you see fit
#image_path = sys.argv[1]
image_path ='testing_images/yes/328_right.jpeg'

# Read in the image_data
image_data = tf.gfile.FastGFile(image_path, 'rb').read()

# Loads label file, strips off carriage return
label_lines = [line.rstrip() for line
                in tf.gfile.GFile("models/retrained_labels.txt")]

# Unpersists graph from file
with tf.gfile.FastGFile("models/retrained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    _ = tf.import_graph_def(graph_def, name='')

with tf.Session() as sess:
    # Feed the image_data as input to the graph and get first prediction
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

    predictions = sess.run(softmax_tensor,             {'DecodeJpeg/contents:0': image_data})

    # Sort to show labels of first prediction in order of confidence
    top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]

    for node_id in top_k:
        human_string = label_lines[node_id]
        score = predictions[0][node_id]
        print('%s (score = %.5f)' % (human_string, score))
```

7. SYSTEM TESTING

7.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

7.1.2 FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

7.1.3 PERFORMANCE TEST

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

7.1.4 STRESS TEST

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

7.1.5 STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

7.2 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.3 TESTING TECHNIQUES / TESTING STRATERGIES

7.3.1 TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the

intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

7.3.1.1 WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases
- Graph matrices Control

7.3.1.2 BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning

- Boundary value analysis
- Comparison testing

7.3.2 SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works "outward" toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

7.3.2.1 INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is "putting them together"- interfacing. There may be the chances of data lost across on another's sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

7.3.2.2 PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

7.3.2.3 SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

7.3.2.4 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-

validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

* The function or performance characteristics confirm to specifications and are accepted.

* A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

7.3.2.5 USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

7.4 TEST REPORT

S.NO	ACTION	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	TEST RESULT
1.	Enter the username, email and password for registration	Username: ABC Password: **** Email id: abc@gmail.com	ABC *** abc@gmail.com	ABC *** abc@gmail.com	Passed
2.	Compare username and password with registered field	Username: ABC Password: ****	It redirects to dashboard form page	It redirects to dashboard form page	Passed
3.	Compare username and password with registered field	Username: Abd Password: ****	Invalid username and password	Invalid username and password	Passed

S.NO	ACTION	INPUT	EXCEPECTED OUTPUT	ACTUAL OUTPUT	TEST RESULT
4.	By clicking choose file test box Upload input image for analysing accuracy range of diabetic retinopathy	xyz.jpeg	Accuracy range of diabetic retinopathy	Accuracy range of diabetic retinopathy	Passed
5.	By clicking choose file test box Upload input image for analysing accuracy range of diabetic retinopathy	xyz	Error page will be display	Error page will be display	Passed
6.	Analysing the accuracy of diabetic retinopathy by giving the name of the image	328_right.jpeg	Accuracy range of diabetic retinopathy	Accuracy range of diabetic retinopathy	Passed
7.	Analysing the accuracy of diabetic retinopathy by giving the name of the image	12.jpeg	Error page will be display	Error page will be diplay	Passed

8. CONCLUSION

On success of our project we can quickly detect Diabetic Retinopathy with high accuracy from our trained neural network and our system will help to reduce the damage cause by diabetic retinopathy at early stage. Our report generation system will give analysis of patient's eye and will help doctors to take quick action .Our system can be further enhanced by training our neural network model on different eye disease so one can get one stop solution for all eye diseases.

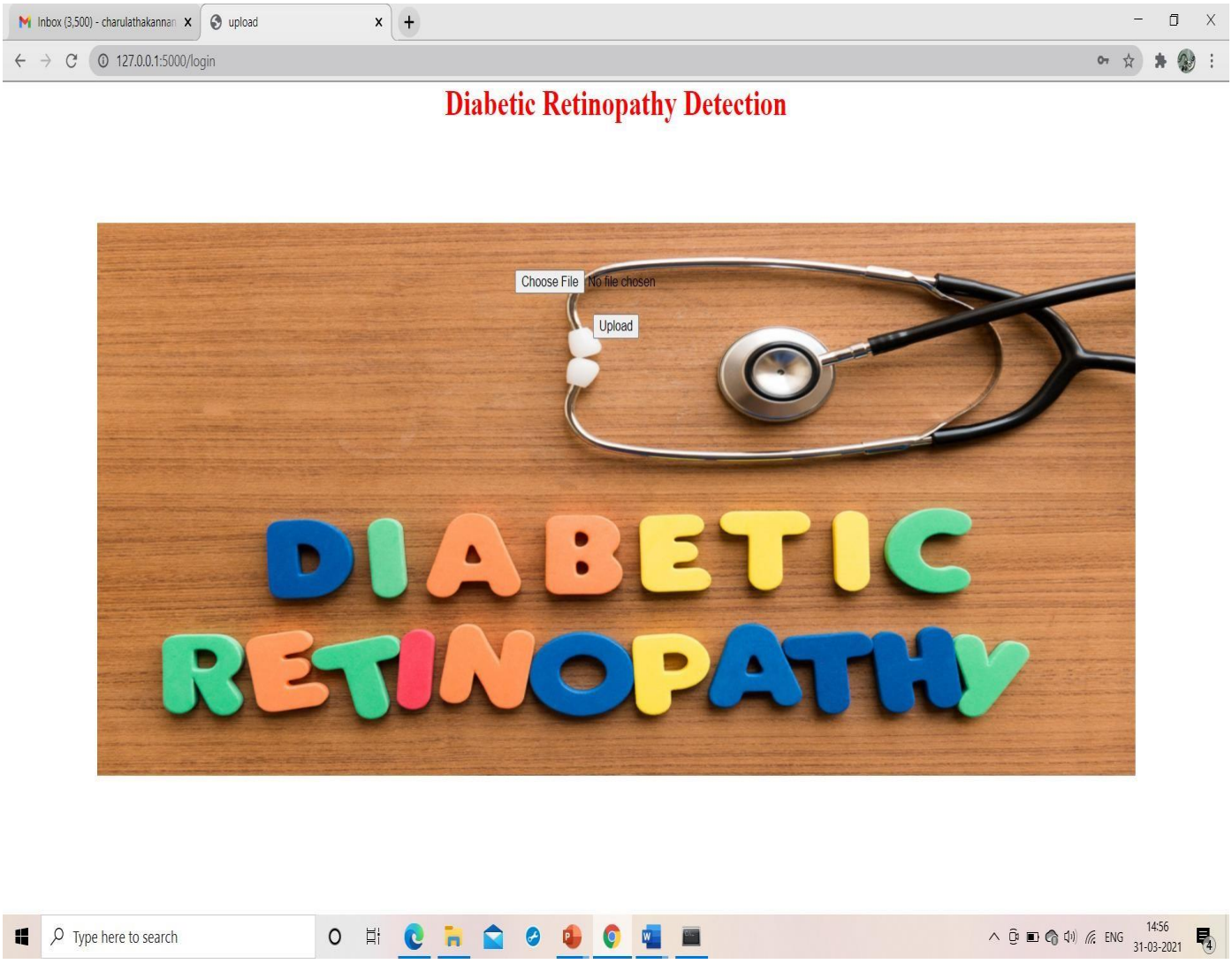
8.1 FUTURE WORK

In this proposed system for the objective of detecting and classification of DR as used 214 fundus images from DIARECT Client-side coding DB1. Detected there lesions namely blood vessels, exudates and micro aneurysms from input images. Then extracted necessary features and classified using ANN classifier. The proposed method performed up to classification sensitivity of 95%, precision of 95% and accuracy 96% respectively. The execution time took about28.064 seconds. As a future work, we can optimize the classifier performance with more images, extracting details features and using different classifier.

APPENDICCES

A.1 SAMPLE SCREENS

first page



Dashboard Form


Inbox (3,500) - charulathakannan
success
127.0.0.1:5000/success
Diabetic Retinopathy Detection
File uploaded successfully
percentage_of _diabetic_retinopathy
0.9048859
percentage_of _nondiabetic_retinopathy
0.09511406
Type here to search
ENG
16:05
31-03-2021

Result

A.2 PUBLICATIONS

DIABETIC RETINOPATHY DETECTION USING DEEP LEARNING TECHNIQUES – Kiruthika K, Charulatha K , Jayasri K , Kamathampalli Jayasri reddy - **www.ijcrt.org (ISSN:2320-28822021) © 2021, Volume.9, Issue 4**

REFERENCES:

[1] K. Istabridis, R. de Figueiredo"Automatic Detection and diagnosis of diabetic retinopathy", IEEE International Conference on Image Processing,12 November 2007.

[2] Eye Smart - What Is Diabetic Retinopathy?, © 2013 American Academy of Ophthalmology

[3] Z. Omar, M. Hanafi, S. Mashohor, M. Muna'im, "Automatic diabetic retinopathy detection and classification system", 7th IEEE International Conference on System Engineering and Technology (ICSET), 1 December 2017.

[4] American Optometric Association, https://www.aoa.org/patients-and-public/eye-andvision-problems/glossary-of-eye-and-visionconditions/diabetic-retinopathy#1

[5] Y. Hatanaka, T. Nakagawa, Y. Hayashi, M. Kakogawa, A. Sawada, K. Kawase, T. Hara and H. Fujita," Improvement of Automatic Hemorrhages Detection Methods using Brightness Correction on Fundus Images," Proc of the International society of optics and photonics on medical imaging, Vol. 6915, 2008.

[6] N. Silberman, K. Ahlrich, R. Fergus and L. Subramanian, "Case for Automated Detection of Diabetic Retinopathy," Proc of the Association for the Advancement of Artificial Intelligence, 2010.

[7] M. García, M. López, D. Álvarez and R. Hornero, "Assessment of four neural network based

classifiers to automatically detect red lesions in retinal images.", Medical Engineering & Physics, Vol. 32, pp. 1085–1093, 2010.

[8] R. Priya and P. Aruna," Review of automated diagnosis of diabetic retinopathy using the support vector machine.", International Journal of Applied Engineering Research, Vol. 1,No. 4,pp. 844-863,2011.

[9] R. Ghosh , K. Ghosh , S. Maitra , "Automatic detection and classification of diabetic retinopathy stages using CNN" , 4th International Conference on Signal Processing and Integrated Networks(SPIN), 28 September 2017

[10] D. Doshi, A. Shenoy, D. Sidhpura, P. Gharpure, "Diabetic retinopathy detection using deep convolutional neural networks", 2016 International Conference on Computing, Analytics and Security Trends (CAST), 01 May 2017.

[11] A. Kesar, N. kaur, P. Singh, "Eye Diabetic Retinopathy by Using Deep Learning", International Research Journal of Engineering and Technology (IRJET).

[12] Y. Kanungo, B. Srinivasan, S. Choudhary, "Detecting diabetic retinopathy using deep learning", 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 15 January 2018

[13] M. Chetoui ,M. Akhloufi ,M. Kardouchi, "Diabetic Retinopathy Detection Using Machine Learning and Texture Features", IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), 30 August 2018.

[14] S. Yu , D. Xiao , Y. Kanagasima, "Exudate detection for diabetic retinopathy with convolutional neural networks", 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 14 September 2017.

[15] S. Albawi , T. Mohammed, S. Al-Zawi,"Understanding of a convolutional neural network",

International Conference on Engineering and Technology (ICET),8 March 2018.

[16] T. Bui, N. Maneerat, U. Watchareeruetai, "Detection of cotton wool for diabetic retinopathy analysis using neural network", IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA), 14 December 2017.

[17] Gulshan, V., Peng, L., Coram, M. et al., "Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs", JAMA, 2016, 316(22):2402–2410

[18] Wang, X., Lu, Y., Wang, Y. and Chen WB, "Diabetic retinopathy stage classification using convolutional neural networks"in International Conference on information Reuse and Integration for data science, 2018, p. 465–71

[19] Abramoff, MD., Lou, Y., Erginay, A., et al.,"Improved Automated Detection of Diabetic Retinopathy on a Publicly Available Dataset through Integration of Deep Learning", Invest Ophthalmol Vis Sci., 2016, 57(13):5200-5206

[20] Dutta, S., Manideep, BC., Basha, SM., Caytiles, RD. and Iyengar, NCSN, "Classification of diabetic retinopathy images by using deep learning models", Int J Grid Distr Comput, 2018, 11(1):99–106

[21] Wan, S., Liang, Y., Zhang, Y, "Deep convolutional neural networks for diabetic retinopathy detection by image classification", Comput Electr Eng, 2018, 72:274–82

[22] Harangi, B., Toth, J., Baran, A., and Hajdu A, "Automatic screening of fundus images using a combination of convolutional neural network and hand-crafted features" in 41st annual International Conference of the IEEE Engineering in Medicine and biology society (EMBC), 2019, p. 2699–702

[23] Wang, J., Luo, J., Liu, B., Feng, R., Lu, L. and Zou, H, "Automated diabetic retinopathy

grading and lesion detection based on the modified R-FCN object-detection algorithm", IET Comput. Vis., 2020, 14(1):1–8.

[24] Hua, CH., Huynh-The, T. and Lee S, "Retinal vessel segmentation using round-wise features aggregation on bracket-shaped convolutional neural networks"in Proceedings of the annual International Conference of the IEEE Engineering in Medicine and biology society, EMBS, 2019, p. 36–9

[25] Wu, Y., Xia, Y., Song, Y., Zhang, Y., and Cai W, "NFN+: a novel network followed network for retinal vessel segmentation. Neural Network", 2020, 126:153–62

[26] Oliveira, A., Pereira, S. and Silva, C.A, "Retinal vessel segmentation based on fully convolutional neural networks", Expert Syst. Appl. 112:229–42, 2018

[27] Chudzik, P., Majumdar, S., Caliva, F., Al-Diri, B. and Hunter A, "Microaneurysm detection using fully convolutional neural networks", Comput Methods Progr Biomed, 2018, 158:185–92