

Detecting Diabetic Retinopathy Category From Retinal Images Using Deep Convolution Neural Networks

Submitted in partial fulfillment of the requirements
of the degree of

B.E. Information Technology

By

Akash Dabhi	44
Nachiket Makwana	67
Archit Masurkar	68
Sarvesh Narkar	70

Supervisor:

Dr. Vaishali Jadhav
Assistant Professor



Department of Information Technology
St. Francis Institute of Technology
(Engineering College)

University of Mumbai
2017-2018

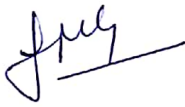
CERTIFICATE

This is to certify that the project entitled **Detecting Diabetic Retinopathy Category From Retinal Images Using Deep Convolution Neural Networks** is a bonafide work of **Akash Dabhi** (Roll No.44), **"Nachiket Makwana"** (Roll No.67), **"Archit Masurkar"** (Roll No.68), **"Sarvesh Narkar"** (Roll No.70) submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Information Technology.



24/4/18

Dr. Vaishali Jadhav
Supervisor/Guide



Dr. Joanne Gomes
Head Of Department

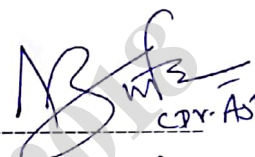


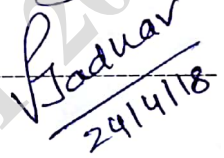
Dr. Sincy George
Principal

Project Report Approval for B.E.

This project report entitled *Detecting Diabetic Retinopathy Category From Retinal Images Using Deep Convolution Neural Networks* by Akash Dabhi, Nachiket Makwana, Archit Masurkar, Sarvesh Narkar is approved for the degree of *B.E. in Information Technology*.

Examiners

1. 
Dr. Ashfaq

2. 
P. Jadhav
24/4/18

Date: 24/04/2018

Place: BORIVALI (W)

Declaration

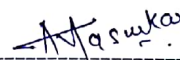
I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.




Akash Dabhi(44)



Nachiket Makwana(67)



Archit Masurkar(68)



Sarvesh Narkar(70)

Date: 24/04/2018

Abstract

Diabetic retinopathy is a condition that occurs in people who have diabetes. It causes progressive damage to the retina, the light-sensitive lining at the back of the eye. Diabetic retinopathy is a serious sight-threatening complication of diabetes. Our project is Determining the Diabetic Retinopathy Category from Retinal Images using Deep Convolution Neural Networks. Our proposed system can be used for early detection and diagnosis of Diabetic Retinopathy and prove beneficial, for those affected. Our objective is to create a model that would take a pre-processed retina image as input and output whether or not it has stage 4 diabetic retinopathy. Currently, diagnosing DR is a slow and arduous process that requires trained doctors to analyse colour photographs of retinas. This model will facilitate the removal of the ambiguous diagnoses done by the ophthalmologists. This task will be done by using one of the pre-trained models- Inception V3 model which is a general object classification model to be used specifically for our task. This will be done using transfer learning and retraining the top layers of the model.

Contents

1	Introduction	1
1.1	Description	1
1.2	Problem Formulation	1
1.3	Motivation	2
1.4	Proposed Solution	2
1.5	Scope of the Project	2
2	Review of Literature	4
2.1	Diabetic Retinopathy	4
2.2	Neural Networks and Deep Learning	5
2.3	Convolution Neural Networks	6
2.4	Transfer Learning	6
3	System Analysis	8
3.1	Functional Requirements	8
3.2	Non Functional Requirements	8
3.3	Specific Requirements	9
3.4	Use - Case Diagram	10
4	Analysis Modeling	13
4.1	Flowchart	13
4.2	Activity Diagram	14
4.3	Sequence Diagram	15
4.4	Functional Modeling (Data Flow Diagram)	16
4.5	Timeline Charts	21
5	Design	24
5.1	Architectural Design	24
5.2	User Interface Design	25
6	Implementation	28
6.1	Algorithms Used	28
6.2	Working of the Project	32
7	Testing	38
7.1	Test Cases	38
7.2	Type of Testing Used	46
8	Results and Discussions	48
8.1	Results	48
8.2	Discussion	51

9 Conclusion and Future Scope	52
9.1 Conclusion	52
9.2 Future Scope	52
10 Literature Cited	54

SEITT LIBRARY 2018

List of Figures

2.1	A simple Artificial Neural Network	5
2.2	Working of a Convolution Neural Network	6
3.1	Use case diagram for DR Category Detection System	10
4.1	Flowchart of DR Category Detection System	13
4.2	Activity diagram for DR Category Detection System	14
4.3	Sequence diagram for DR Category Detection System	15
4.4	Context level diagram for DR Category Detection System - Level 0 DFD . .	16
4.5	Training the Deep Learning Model to assist in detecting DR Category - Level 1 DFD	16
4.6	Testing of the DR Category of single image - Level 1 DFD	17
4.7	Image acquisition process - Level 2 DFD	18
4.8	Image pre-processing process - Level 2 DFD	18
4.9	Training the Deep Learning Model using massive image dataset - Level 2 DFD	19
4.10	Classification result for single image - Level 2 DFD	20
4.11	Timeline chart for sem 7	21
4.12	Timeline chart for sem 8	22
5.1	Architecture of DR Grading System	24
5.2	Web page before uploading the retinal image	25
5.3	Analyzing the retinal image	26
5.4	Displaying the DR Category	27
6.1	Inception block	28
6.2	Top 2 blocks of Inception V3	31
7.1	Testing the system with CSV file	40
7.2	Testing the system with external images	41
7.3	Testing the system with retinal images of animals	42
7.4	Testing the system with artificially generated images	43
7.5	Testing the system with blur retinal images	44
7.6	Graph of time taken for batch prediction	45
8.1	Graph of accuracy	50
8.2	Graph of loss	50

List of Tables

3.1	Use case 1 - Capture image	10
3.2	Use case 2 - Upload image	11
3.3	Use case 3 - Login to portal	11
3.4	Use case 4 - Pre-process image	11
3.5	Use case 5 - Determine DR Category	12
3.6	Use case 6 - Display DR Category	12
3.7	Use case 7 - Generate report	12
7.1	Results of test cases	38
7.2	Time taken for batch prediction	45
8.1	Summary of training	48

SEIT LIBRARY 2018

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
i	BMP	Bitmap Image File
ii	CNNs	Convolution Neural Networks
iii	CSV	Comma Separated Values
iv	CUDA	Compute Unified Device Architecture
v	DFD	Data Flow Diagram
vi	DR	Diabetic Retinopathy
vii	GPU	Graphics Processing Unit
viii	ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ix	JPG	Joint Photographic Experts Group
x	K-NNs	K-Nearest Neighbours
xi	LSVRC	Large Scale Visual Recognition Challenge
xii	ML	Machine Learning
xiii	NN	Neural Network
xiv	OS	Operating System
xv	PNG	Portable Network Graphics
xvi	PPM	Portable PixMap
xvii	RAM	Random Access Memory 2
xviii	SVM	Support Vector Machine
xix	TIFF	Tagged Image File Format
xx	VRAM	Video Random Access Memory

Chapter 1

Introduction

Diabetic retinopathy (DR) is a condition that occurs in people who have diabetes. It causes progressive damage to the retina, the light-sensitive lining at the back of the eye. Diabetic retinopathy is a serious sight-threatening complication of diabetes. Our project is Determining the Diabetic Retinopathy Category from Retinal Images using Deep Convolution Neural Networks (CNNs). Our proposed system can be used for early detection and diagnosis of Diabetic Retinopathy and prove beneficial, for those affected.

1.1 Description

Our objective is to create a model that would take a pre-processed retina image as input and output whether or not it has stage 4 diabetic retinopathy. Currently, diagnosing DR is a slow and arduous process that requires trained doctors to analyse colour photographs of retinas. This model will facilitate the removal of the ambiguous diagnoses done by the ophthalmologists.

1.2 Problem Formulation

DR is the leading cause of blindness for people aged 20 to 64 years. It is estimated to affect about 93 million people globally, though only half are aware of it. If caught early enough, vision degeneration can be slowed if not stopped, but this is often difficult because symptoms may appear too late to provide effective treatment. In India there is a shortage of 1,27,000 eye doctors, due to which 45% of patients suffer vision loss before diagnosis. The number of ophthalmologists in India is 11,441 per million, whereas our population is 1.3 billion. Even when available, ophthalmologists in India are inconsistent. Hence a dire need for an automated system that would exhibit more precision in diagnosis has risen.

1.3 Motivation

Over time, diabetes damages the blood vessels in the retina. Diabetic retinopathy occurs when these tiny blood vessels leak blood and other fluids. This causes the retinal tissue to swell, resulting in cloudy or blurred vision. The main motivation of this project is to provide early and correct diagnosis for patients who are victims of DR. Since, early detection leads to increase chance of curing, it is very important that the detection should be done at initial stages only. Due to human errors and lack of training of Ophthalmologists, it isn't always possible to detect and diagnose the disease easily. So, to improve diagnostic accuracy and patients chances of getting cured at an earlier stage, our system provides the best possible solution. Also the fundus camera used for taking retinal images is a heavy equipment and so implies its high cost. Hence a solution to developing an inexpensive solution to this is also considered in our system.

1.4 Proposed Solution

One of our objectives is to create a deep learning model that would take a pre-processed retinal image as input and output whether or not it has stage 4 diabetic retinopathy. This model will facilitate the removal of the ambiguous diagnoses done by the ophthalmologists. The second objective of our project is to make an affordable and light weight hand- held fundus camera. This camera will be made as an extension to the smart phone to which our lens will be mounted just as any other camera accessory. The images will be captured in the smartphone using the fundus lens extension and then processed and supplied to the deep learning model for category classification result (Categories 0:No DR, 1: Mild DR, 2: Moderate DR, 3:Severe DR, 4: Proliferative DR).

1.5 Scope of the Project

The aim of this project is to provide early detection and diagnosis of Diabetic Retinopathy. We understood the existing system for the detection of DR and what are the drawbacks of this system. This project aims to conquer these problems with its implementation. With the inexpensive camera extension, it will facilitate the easy acquisition of patients retinal images with a simple smart phone helping individual Ophthalmologists to incorporate this service in their clinical services. These images fed into the Deep learning model will provide with the category of DR in the patients eye.

This would enable the faster and accurate prediction and diagnosis of patients condition. Early detection will help prevent DR from reaching its irreversible state of permanent blindness.

SEITT LIBRARY 2018

Chapter 2

Review of Literature

This chapter provides the review on Image Processing, Computer Vision, Open CV, Python, Machine learning, Artificial Neural Networks, Convolution Neural Networks, Transfer Learning, Inception V3 model.

For the development of a system that is able to recognize diabetic retinopathy category through retinal images, previous research on the way retinal images have been analysed needs to be reviewed.

2.1 Diabetic Retinopathy

Diabetic retinopathy is caused by damage to the blood vessels in the tissue at the back of the eye (retina). Poorly controlled blood sugar is a risk factor. Early symptoms include floaters, blurriness, dark areas of vision and difficulty perceiving colours. It affects up to 80 percent of people who have had diabetes for 20 years or more [1]. At least 90% of new cases could be reduced if there were proper treatment and monitoring of the eyes. The longer a person has diabetes, the higher his or her chances of developing diabetic retinopathy. Each year in the United States, diabetic retinopathy accounts for 12% of all new cases of blindness. It is also the leading cause of blindness for people aged 20 to 64 years. [2]

Varun Gulshan and Lily Peng proposed a comprehensive method of automated DR category grading from retinal images using deep learning algorithms. They have emphasized the need of this project and how deep learning will be beneficial because of their superior image recognition abilities. They have trained 26 CNN layers over the final layer of the Inception V3 model and achieved a classification accuracy of around 97%. [3]

2.2 Neural Networks and Deep Learning

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is inspired by the structure and functional aspects of biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modelling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables. The goal of the neural network is to solve problems in the same way that the human brain would, although several neural networks are more abstract [4]. Figure 2.1 below shows a simple Artificial Neural Network.

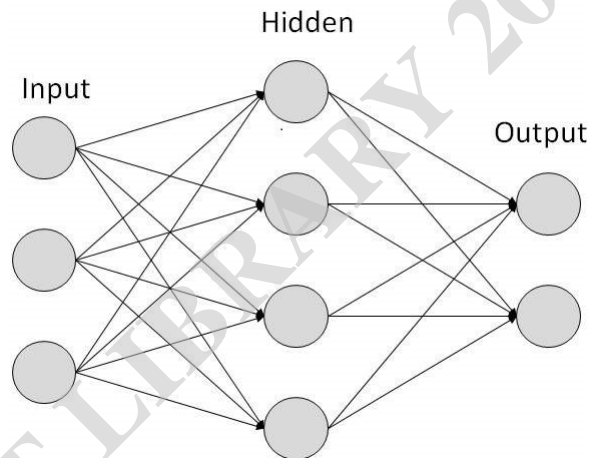


Figure 2.1: A simple Artificial Neural Network [4]

Deep learning is a family of computational methods that allow an algorithm to program itself by learning from a large set of examples that demonstrate the desired behavior, removing the need to specify rules explicitly. Application of these methods to medical imaging requires further assessment and validation. [3]

2.3 Convolution Neural Networks

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply [2]. Figure 2.2 below shows the working of CNN.

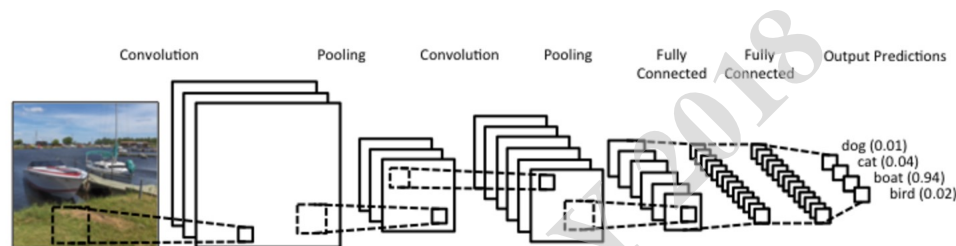


Figure 2.2: Working of a Convolution Neural Network [5]

2.4 Transfer Learning

Transfer learning is the process of taking a pre-trained model (the weights and parameters of a network that has been trained on a large dataset by somebody else) and fine-tuning the model with your own dataset. The idea is that this pre-trained model will act as a feature extractor. You will remove the last layer of the network and replace it with your own classifier (depending on what your problem space is). You then freeze the weights of all the other layers and train the network normally (Freezing the layers means not changing the weights during gradient descent/optimization).

In 2010, the introduction of the yearly ImageNet challenge [6, 7] boosted the research on image classification and the belonging gigantic set of labelled data is often used in publications ever since. In a later work of Krizhevsky, a network with 5 convolutional, 3 max pooling, and 3 fully connected layers is trained with 1.2 million high resolution images from the ImageNet LSVRC-2010 contest.

After implementing techniques to reduce overfitting, the results are promising compared to previous state-of-the-art models. Furthermore, experiments are done with lowering the network size, stating that the number of layers can be significantly reduced while the performance drops only a little. [8]

ImageNet is a dataset that contains 14 million images with over 1,000 classes. When we think about the lower layers of the network, we know that they will detect features like edges and curves. Now, unless you have a very unique problem space and dataset, your network is going to need to detect curves and edges as well. Rather than training the whole network through a random initialization of weights, the frozen weights of the pre-trained model can be used and the more important layers i.e. ones that are higher up can be focused for training.

Christian Szegedy and Wei Liu described the Inception architecture that achieves the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of Inception architecture is the improved utilization of the computing resources inside the network. By a carefully crafted design, the depth and width of the network have been increased while keeping the computational budget constant. [9]

Chapter 3

System Analysis

3.1 Functional Requirements

- The system requires retinal images of patients.
- The system will apply pre-processing techniques such as noise removal, image resizing, mapping to average grey level and reshaping image as desired for input to the deep learning model.
- Feature selection, extraction and classification will be done by the deep learning model.
- The system will detect whether the patient has DR or not.
- If the patient has DR, the system will classify the retinal image based on the vectors obtained from feature extraction process.

3.2 Non Functional Requirements

- Performance accuracy for the system application in the medical field, the result should be accurate for the complete dataset and the new images which will be added.
- Sensitivity and specificity are statistical measures of the performance of a binary classification test, also known in statistics as classification function. The sensitivity and specificity should be high as it forms a part of determining the accuracy of the system.
- Sensitivity (true positive rate or recall) measures the proportion of positives that are correctly identified as such.
- Specificity (true negative rate) measures the proportion of negatives that are correctly identified as such.
- The data of each patient must remain safe, the images must be safeguarded against illegal usage.
- Only the Ophthalmologist or examiner must have the right to access the camera extension gear and also for the input to the model, also the report should be delivered to him first.

3.3 Specific Requirements

Software Requirements

- Windows OS/ Linux OS
- Python
- Octave
- Deep learning libraries (TensorFlow, Theano, Keras, SciPy, OpenCV, PIL)
- Browser

Hardware Requirements

- A personal computer with features such as: Intel i5 – 6th gen processor, 8 Gb RAM memory, 4 Gb Graphic Memory and GPU support.

Requirements for the camera extension prototype:

Material required:

- 3 mm Cardboard pieces
- PVC pipes for lens cavity and optical tube
- Glue
- Electrical insulation tape and sticky tape
- Black chart paper
- Scissors, ruler, cutter
- A condensing lens (20 – 28D)
- A smartphone (Its camera)

3.4 Use – Case Diagram

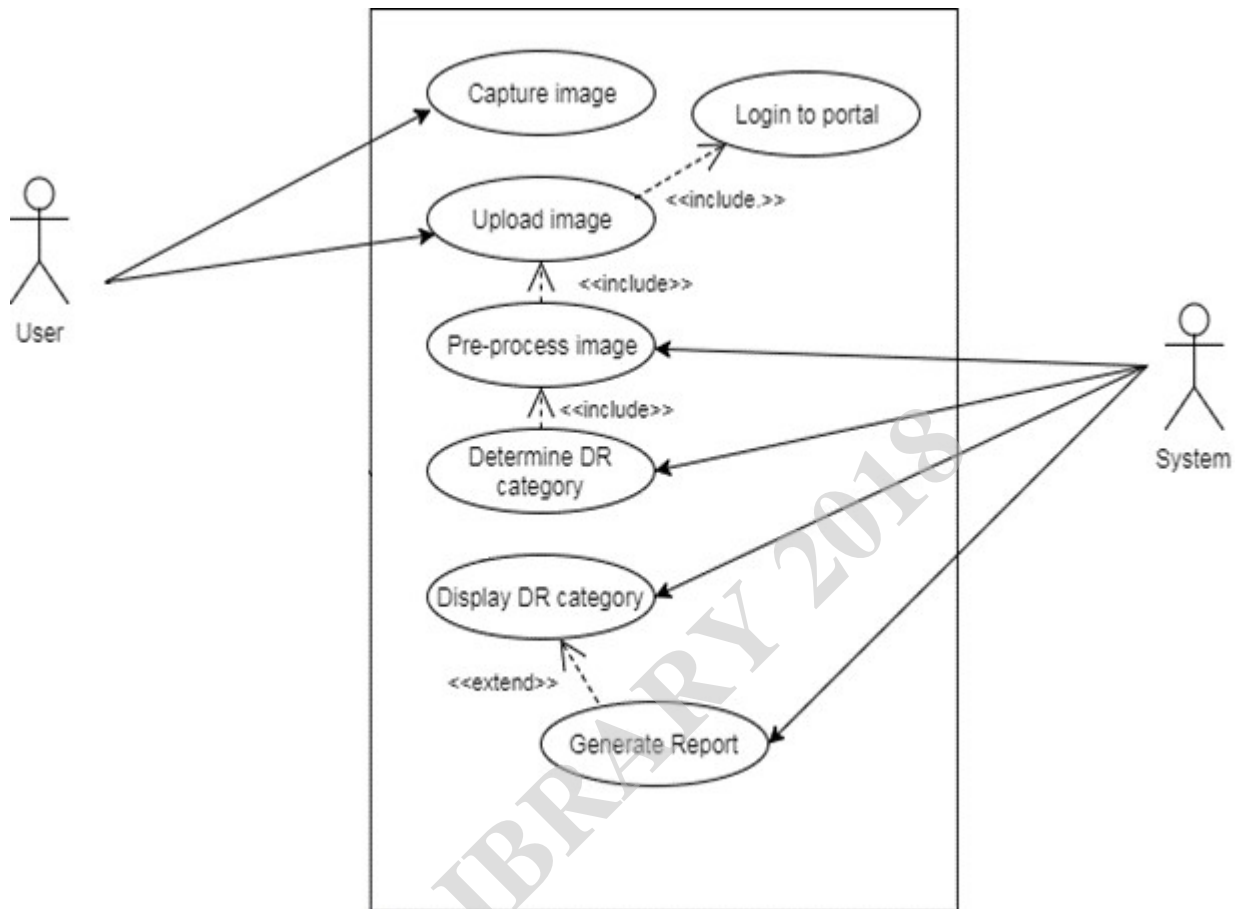


Figure 3.1: Use Case Diagram for DR Category Detection System

Figure 3.1 above shows the use case diagram for DR category detection system. A USER is any person who wishes a DR category to be generated from the input retinal image. A SYSTEM is a trained model which accepts the retinal image as an input and gives the DR category of the image.

Table 3.1: Use case 1 - Capture image

Use case	Capture Image
Use Case ID	UC01
Actor	User
Description	User needs to capture image which is to be uploaded on portal.

Table 3.2: Use case 2 - Upload image

Use case	Upload Image
Use Case ID	UC02
Actor	User
Description	User uploads the captured image on the portal.

Table 3.3: Use case 3 - Login to portal

Use case	Login to Portal
Use Case ID	UC03
Actor	User
Description	User needs to login to the portal using his/her credentials.

Table 3.4: Use case 4 - Pre-process image

Use case	Pre-process Image
Use Case ID	UC04
Actor	System
Description	The system pre-processes the image uploaded by the user.

Table 3.5: Use case 5 - Determine DR Category

Use case	Determine DR Category
Use Case ID	UC05
Actor	System
Description	The system determines DR category from the trained model.

Table 3.6: Use case 6 - Display DR Category

Use case	Display DR Category
Use Case ID	UC06
Actor	System
Description	The DR category is displayed on the portal.

Table 3.7: Use case 7 - Generate report

Use case	Generate Report
Use Case ID	UC07
Actor	System
Description	A pdf report of the determined category is generated.

Description: Tables 3.1 to 3.7 above show each use case along with their actor and description. As shown in Fig 3.1, the user captures the image using the Special handheld camera. He has to login to the web portal designed to upload the images which returns the result of the DR category. Once the user logs in, he has to upload the image on the portal. This image undergoes the pre-processing which is done by the system (backend). The system predicts the DR category of the uploaded retina image based on the training done on the deep learning model. The system gets the predicted output of the deep learning model for the uploaded model and generated a report which is displayed to the user.

Chapter 4

Analysis Modeling

The tools and charts required to demonstrate the work flow of our project are documented as follows.

4.1 Flowchart

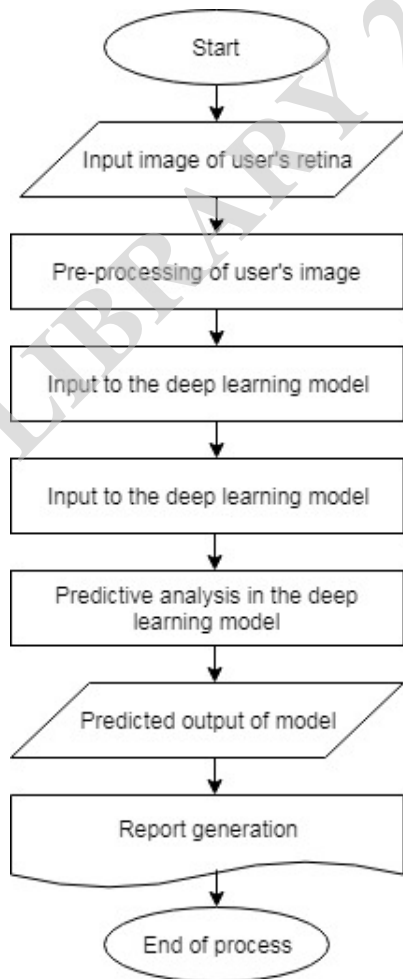


Figure 4.1: A Flowchart defining the flow of processes in the DR Category Detection System

Description: As shown in the above Figure 4.1, this flowchart defines the flow of processes that take place in the procedure of detecting Diabetic Retinopathy category from the uploaded image.

4.2 Activity Diagram

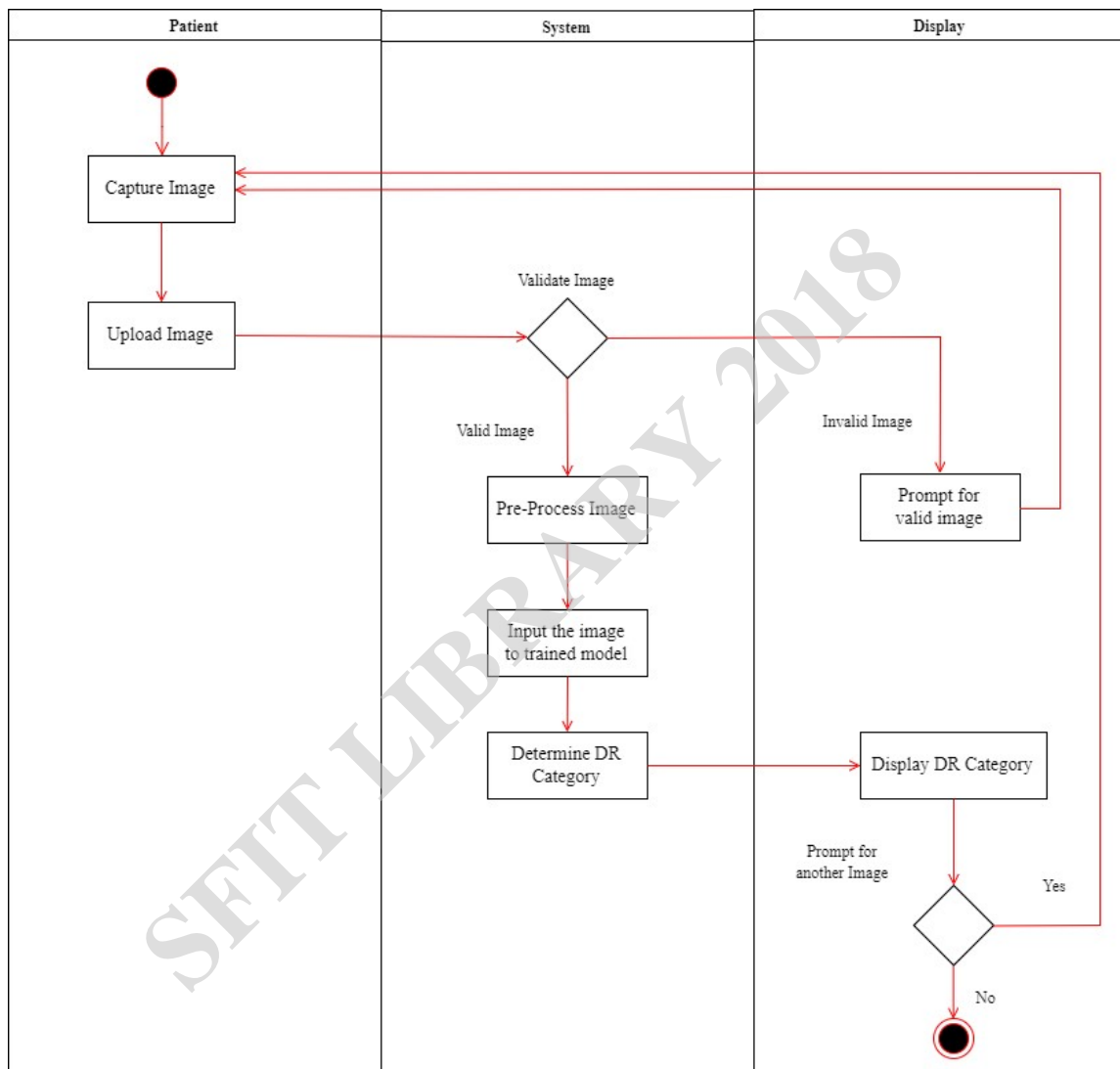


Figure 4.2: Activity Diagram for DR Category Detection System

Description: As shown in the above Figure 4.2, the activity diagram shown above is divided into three partitions 1.) Patient, 2.) System and 3.) Display. The patient Captures the image and Uploads image. This image is checked against similar examples in order to Validate whether the uploaded image is a valid Retinal Image. If the image is a valid one, then it goes into pre-processing stage

and it is given as an input to the Trained model. This activity outputs the DR category. If the image is not a valid one, then the prompt displays a message to upload a valid image. The prompt displays the DR category. Once the output is displayed, the system becomes ready for another image to analyze.

4.3 Sequence Diagram

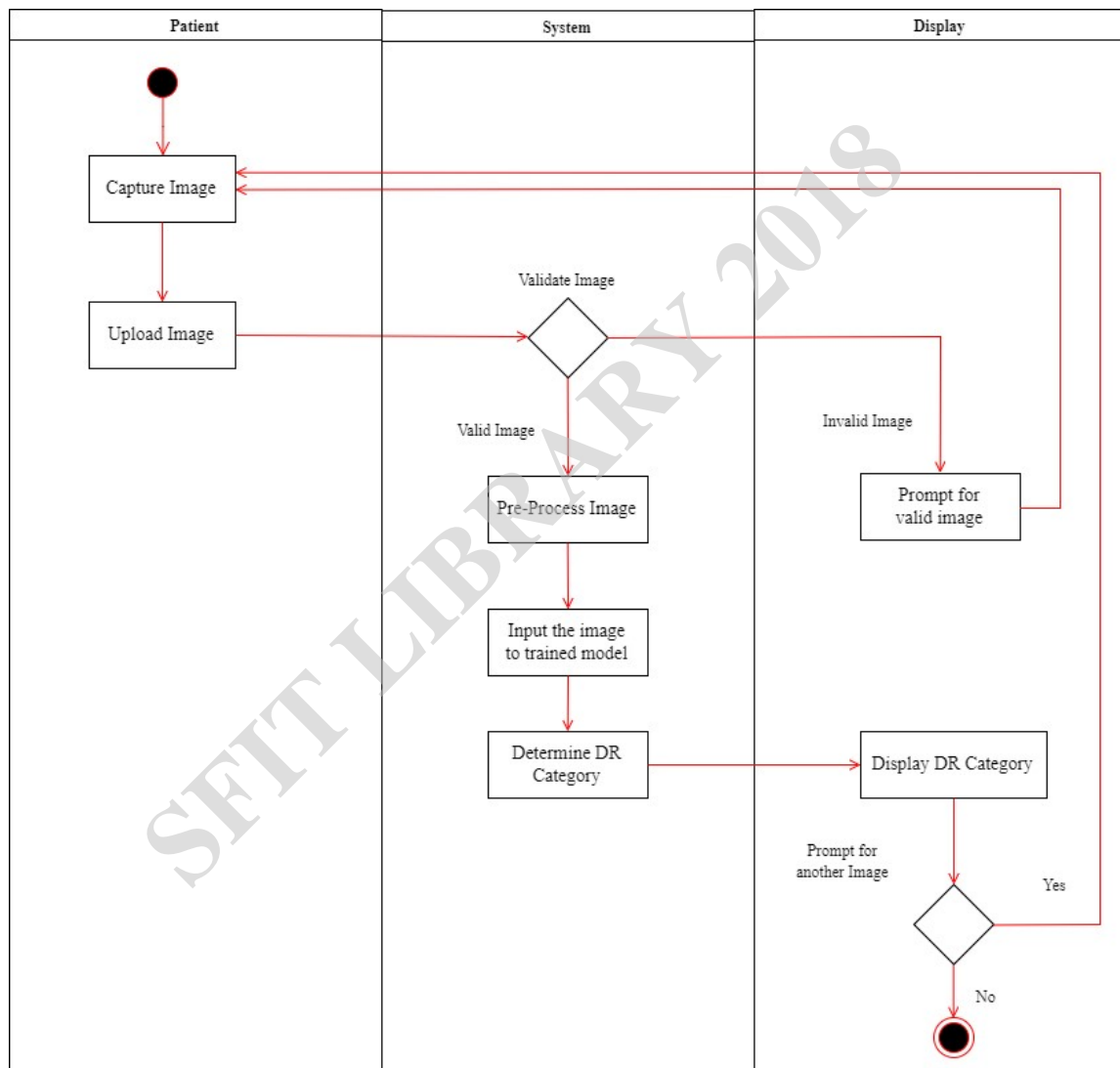


Figure 4.3: Sequence Diagram

Description: As shown in the above Figure 4.3, the patient captures the image using the Camera module. This image is passed on to the system where it is pre-processed and is sent to classification model for determining the DR category. This category is displayed as a report to the patient.

4.4 Functional Modeling (Data Flow Diagram)

Context Level Diagram Level 0 DFD

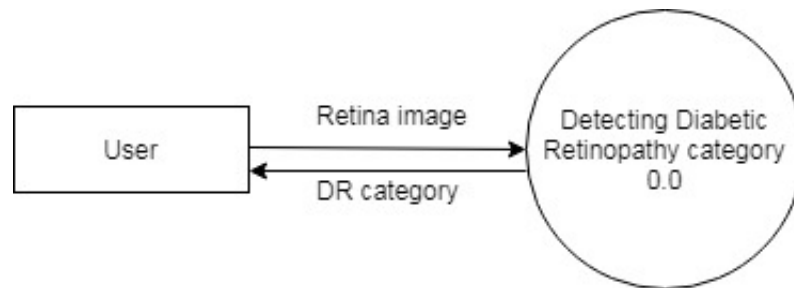


Figure 4.4: Context Level Diagram for Diabetic Retinopathy Detection System - Level 0

Description: As shown in the above Figure 4.4, the user uploads the image of his retina using the camera. The Detecting Diabetic Retinopathy Category process receives the retinal image. This process then displays the DR category to the user.

Level 1 DFD

Training

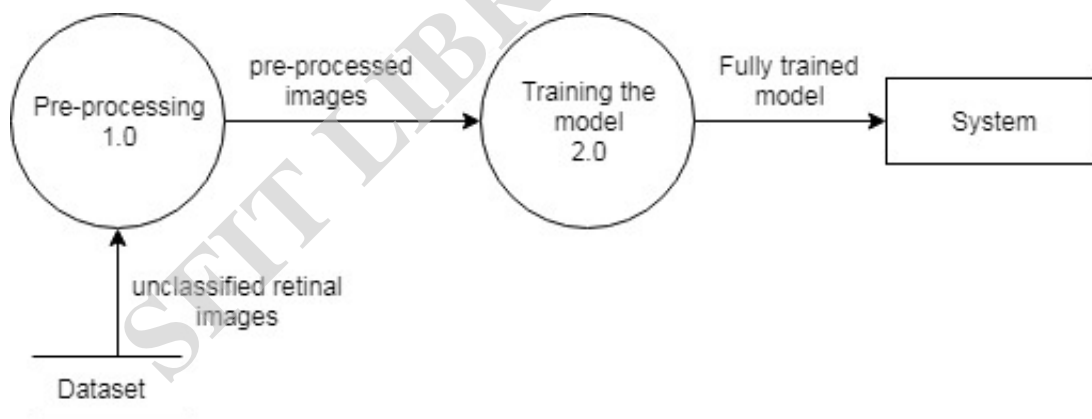


Figure 4.5: Training of the Deep Learning Model to assist in detecting DR Category - Level 1

Description: As shown in the above Figure 4.5, the dataset containing several images of Retina as well as their classification labels are fed to the Pre-processing process. This process gives out the pre-processed images which are then fed to the Training the model process. This trains the model and makes the system ready to detect the DR category for new images.

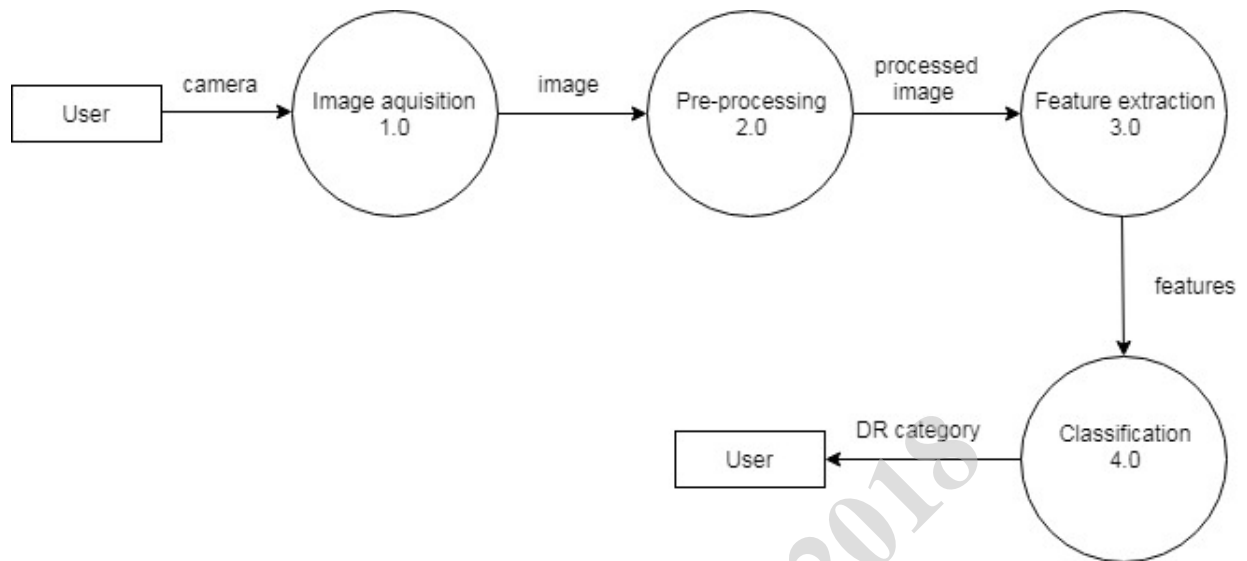
Testing

Figure 4.6: Testing of the DR Category of single image uploaded by user - Level 1

Description: As shown in the above Figure 4.6, the user uses the handheld camera to capture the Retina image for the Image Acquisition process. This process supplies the image to the Pre-processing process which carries out the required filtration on the image which is then supplied to the trained model for Feature Extraction process. This process extracts the features based on the training of the model. These features are supplied to the classification model to determine the DR category.

Level 2 DFD

Image Acquisition

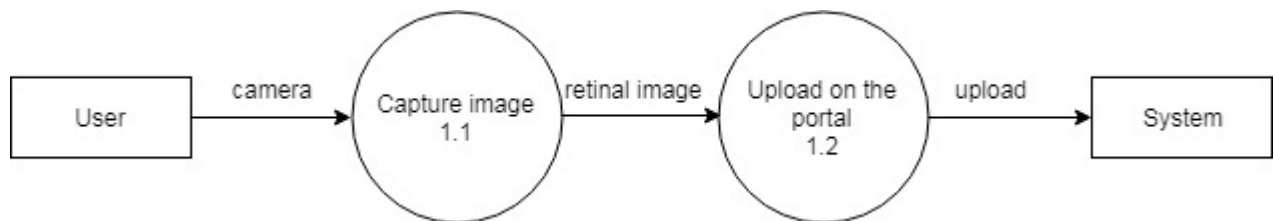


Figure 4.7: Image Acquisition process in implementation - Level 2

Description: As shown in the above Figure 4.7, the image is captured by the user using the camera. This is the Capture image process. This captured image of the retina of the patient. This image is to be uploaded on the web portal for further analysis.

Image Pre-processing



Figure 4.8: Image Pre-processing process - Level 2

Description: As shown in the above Figure 4.8, the system supplies the uploaded image for Pre-processing. The pre-processing of the image involves the following steps. The image being of higher resolution is resized into a 256*256 pixel image. This resized image is supplied to the Clipping the edges of image process to remove the extra edges and give the image a perfect shape. This clipped image is then given as input to the Mapping image to 50% grey image process which is done in order to remove the camera glare if found. This pre-processed image is then supplied to the system for further analysis.

Training

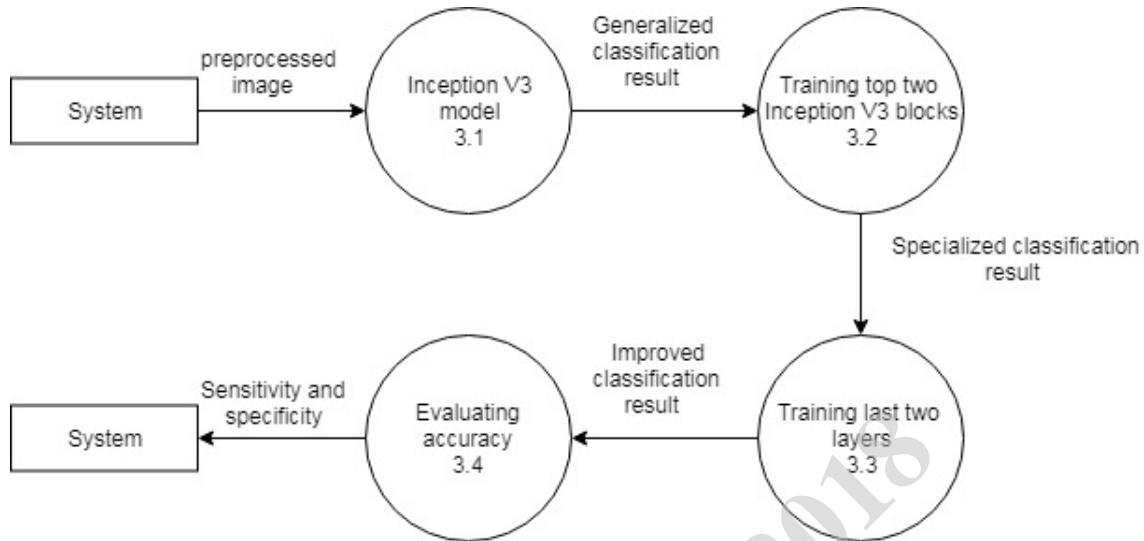


Figure 4.9: Training the Deep Learning Model using massive image dataset - Level 2

Description: As shown in the above Figure 4.9, in this whole process, the whole dataset is used for training the deep learning model. This activity is a totally disparate activity from the rest of the process. This activity is actually the predecessor to all the activities. The pre-processing in the final analysis of single patient image as well as for applying on the whole dataset is the same. So here, the whole dataset is pre-processed as specified earlier. These pre-processed images are supplied as input to the Inception V3 model which is a pre-trained model. The concept of transfer learning comes into picture here which does not require any developer to build and train Convolution Neural Networks from scratch which is an expensive process in terms of computation resources and time. The output of the Inception V3 model is the features which are extracted by the Inception model which are in accordance to the classification results of a generalized dataset. To make it applicable for our task, the top two blocks of the Inception model are trained as per our specific task. This gives the specialized classification results which are supplied to two more Neural network layers in order to enhance accuracy of the model. The final stage in the training procedure is evaluating the accuracy of the trained model by calculating parameters such as Sensitivity and Specificity which are stored into the system.

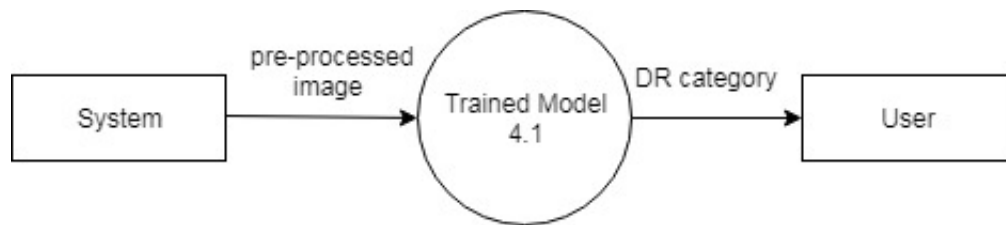
Classification Result

Figure 4.10: Classification Result for single image displayed to patient - Level 2

Description: As shown in the above Figure 4.10, the image captured is supplied to the portal, where the image is passed by the system to the deep learning model to test along with the training done on it. This model gives the output as the DR category of the patients image.

4.5 Timeline Charts

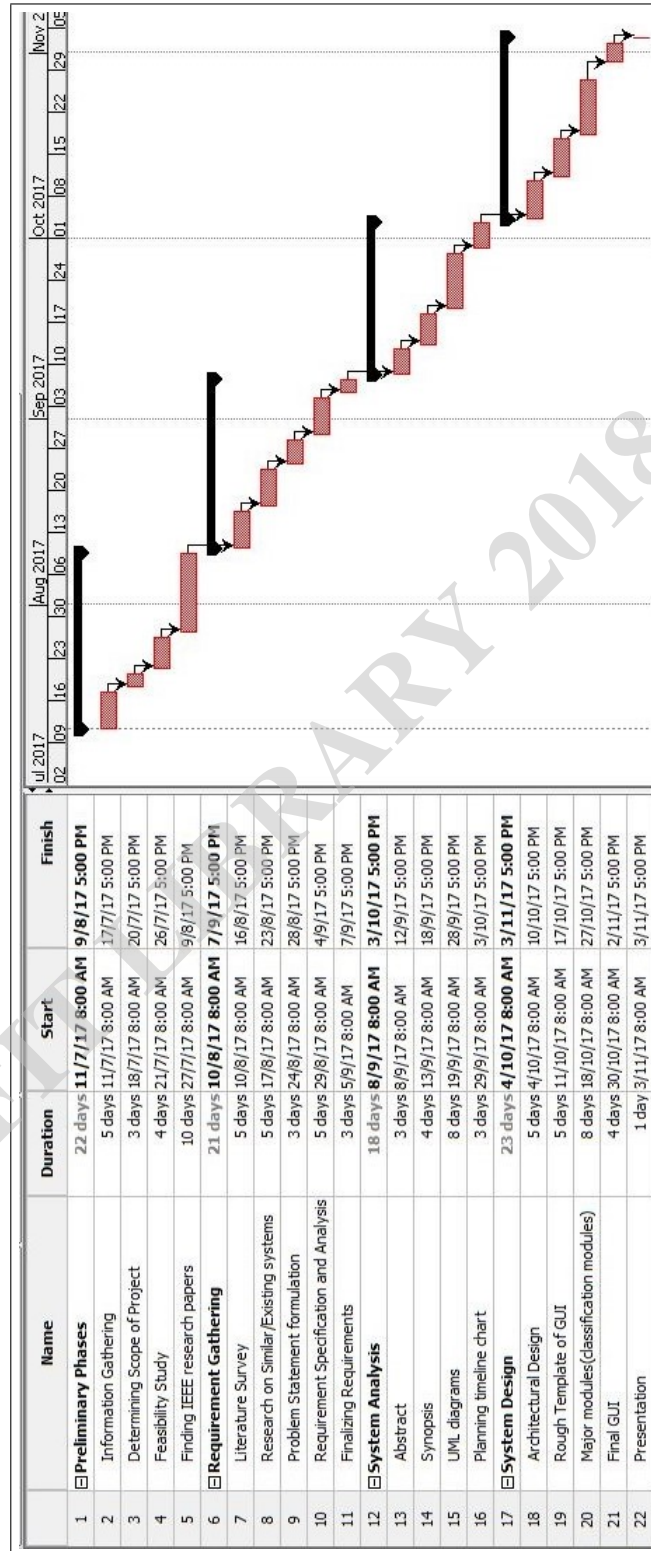


Figure 4.11: Timeline Chart for Sem 7

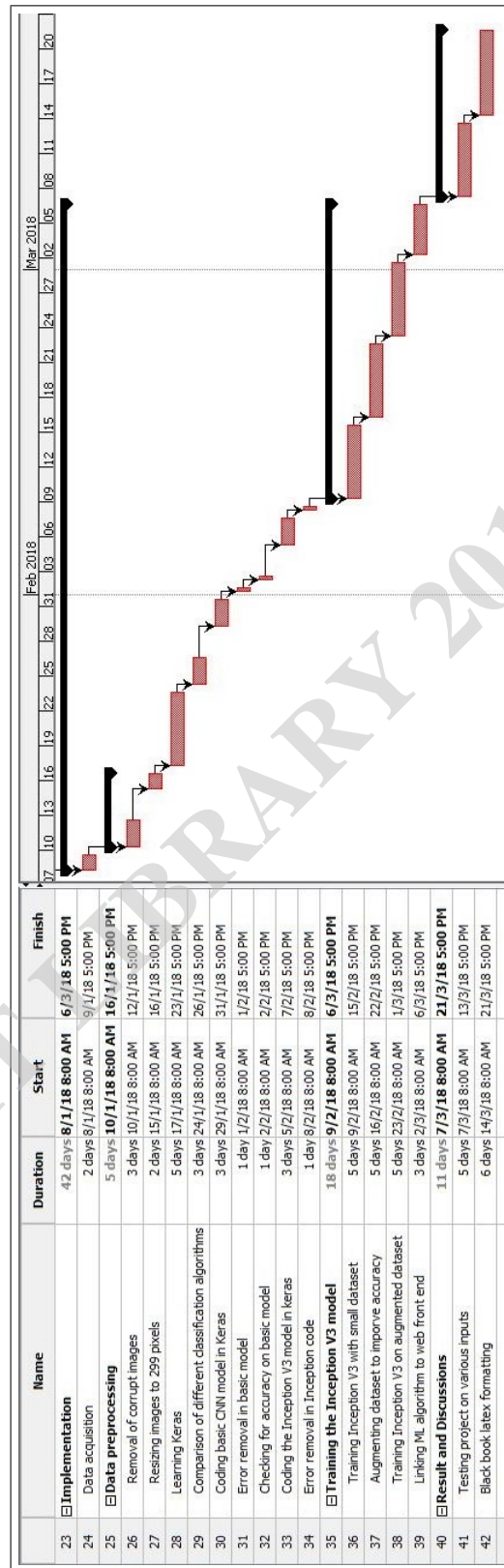


Figure 4.12: Timeline Chart for Sem 8

Description: In the above figures we have jotted down our sequential plan of execution of the project proceedings in a graphical way using the gantt charts. The semester 7 chart, i.e Figure 4.11, includes activities such as paper finding, review of literature, UML diagrams and system analysis. These activities correspond to the project charter to be made before the implementation starts. In the second phase i.e. semester 8, the focus was on implementation of the project plan as shown in the Figure 4.12. The implementation includes various activities such as data collection, data pre-processing, coding a small CNN and reviewing it, augmenting the data and training the large inception V3 architecture for certain number of iterations until the desired accuracy was achieved.

SEITT LIBRARY 2018

Chapter 5

Design

Here we present the design aspect of our project such as the system architecture and the User Interface Design.

5.1 Architectural Design

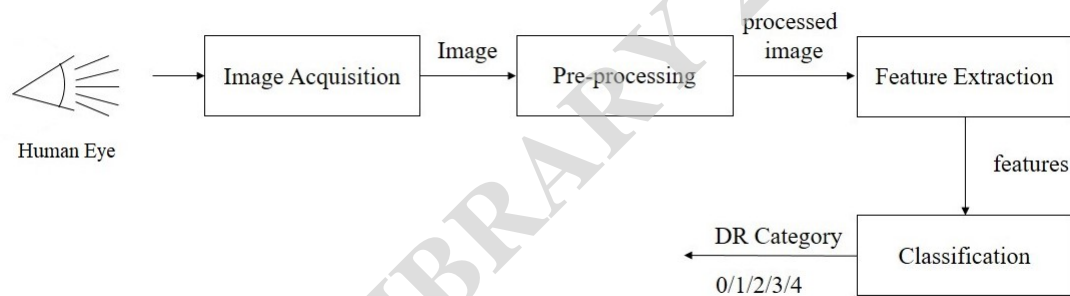


Figure 5.1: Architecture of DR Grading System

Description: As shown in above Figure 5.1, the patient is prompted to upload an image of his retina for diagnosis using a camera which will get pre-processed and sent as an input to the deep learning model for feature extraction. Once this is done, the deep learning model will output the category of DR in this particular uploaded image based on the training done in the earlier phase of the project.

5.2 User Interface Design

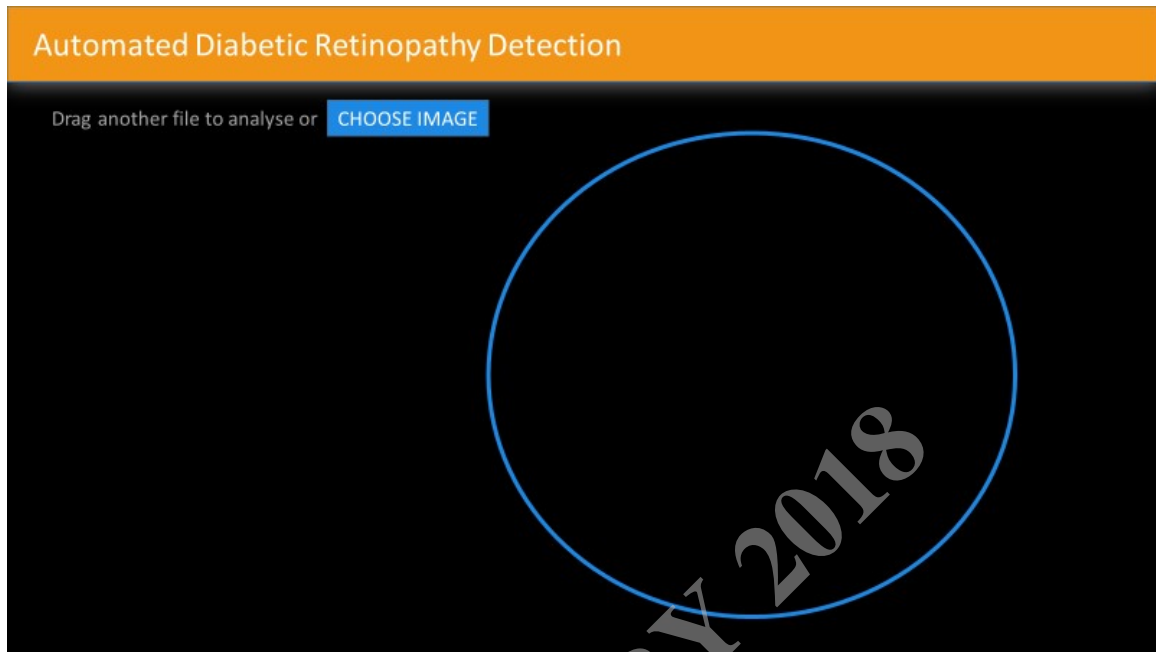


Figure 5.2: Web page before uploading the retinal image

Description: As per the above Figure 5.2, this is our webpage which has a button for uploading the retinal image whose Diabetic Retinopathy category needs to be checked.

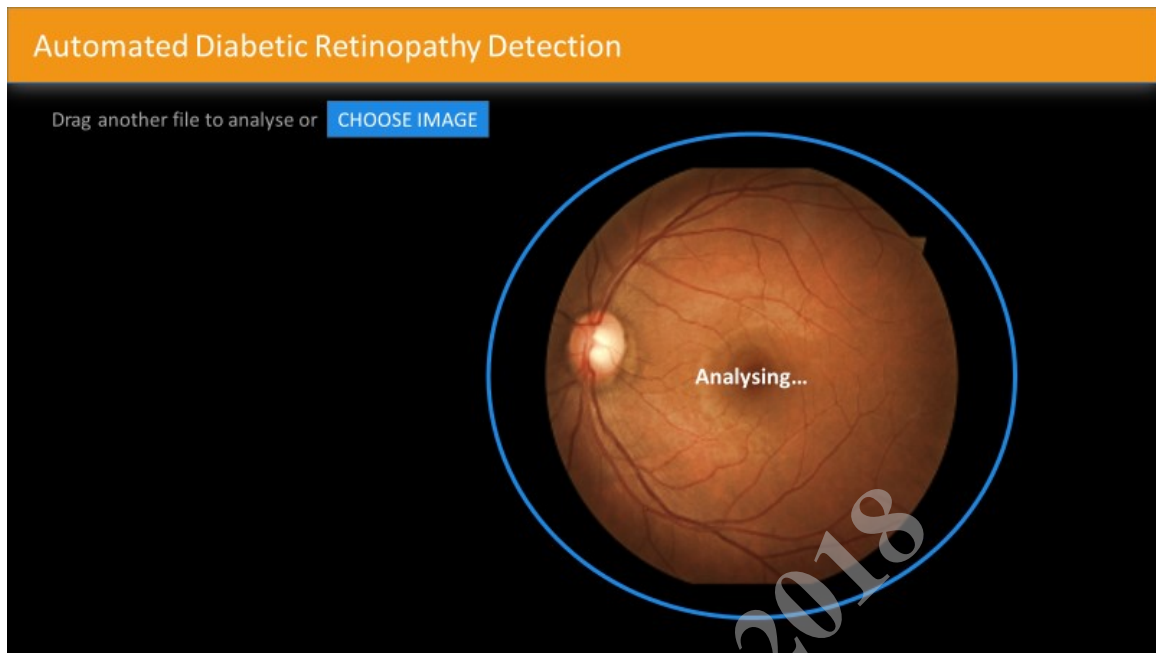


Figure 5.3: Analyzing the retinal image

Description: As per the above Figure 5.3, after uploading the retinal image the trained model tests the image with respect to the features for checking the Diabetic Retinopathy category.

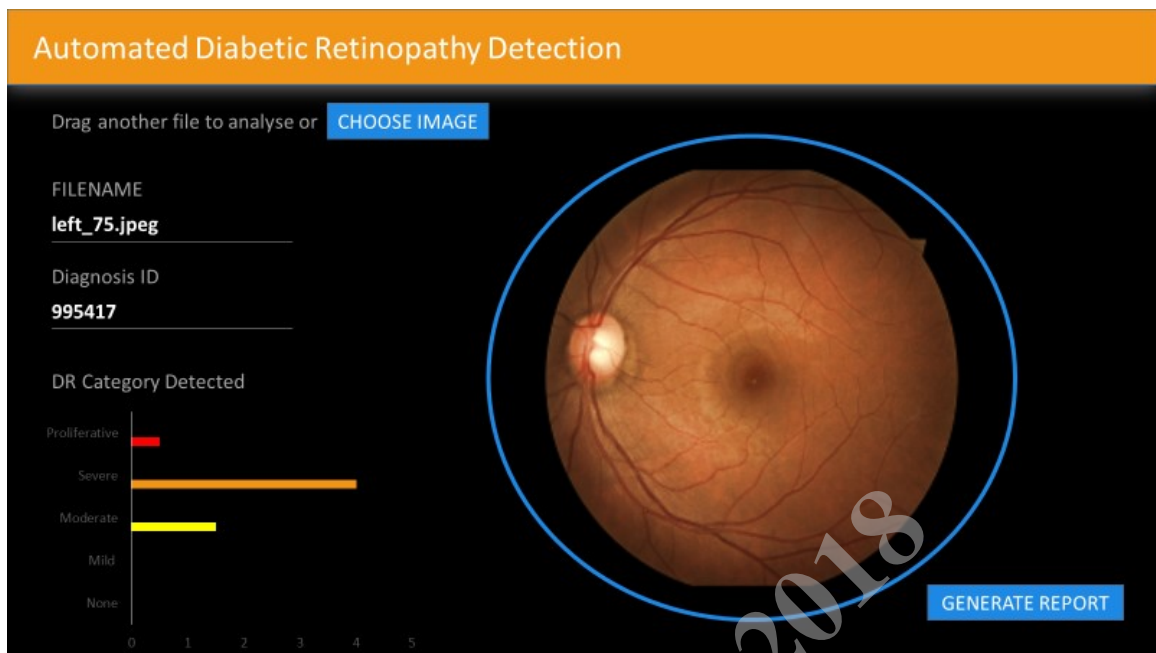


Figure 5.4: Displaying the DR Category

Description: As per the above Figure 5.4, after the model has tested the retinal image, a graph consisting of all the DR categories along with their corresponding confidence levels is displayed.

Chapter 6

Implementation

6.1 Algorithms Used

6.1.1 Comparison of Classification Algorithms

Initially we used some of the famous supervised learning algorithms such as K-Nearest neighbours (K-NNs) and Support Vector Machines (SVMs). For our classification task these algorithms failed miserably in classifying the images in our dataset. We found an accuracy of 25% for KNN and 24% for SVMs. This is because the feature extraction capabilities of KNNs and SVMs is lower as compared to CNNs. We can justify this because we found an accuracy of around 66% in the initial stages of our research when we trained only the final layer of the Inception V3 architecture. The diagram of the inception block is given in the below Figure 6.1.

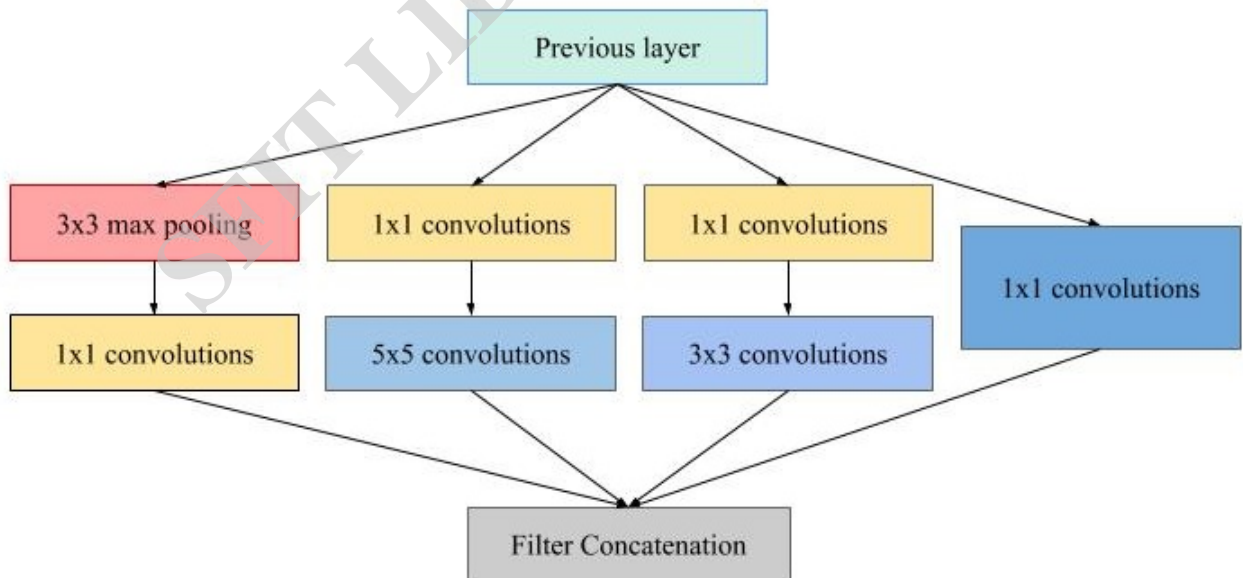


Figure 6.1: Inception Block

Many such inception blocks are stacked over each other to make a deep network which is called as GoogLeNet but is generally referred as Inception V3 model.

6.1.2 Inception V3 model - A Deep Convolutional Neural Network

As previously mentioned, we have used the Inception V3 model and retrained it for our classification task. The understanding of doing this was derived from the concept of Transfer learning. It is a widely used concept by many deep learning researchers. Transfer learning must be used when there is a lot of data for the problem we are transferring learning from and usually relatively less data for the problem we are transferring learning to. In our case, we are transferring the learning of Inception V3, which is trained on 1.2 million images of the ImageNet challenge, to our classification task which consists of 25000 retinal images. The underlying logic behind this is that the low level features such as random shapes, edges, lines etc. contained in the weights of the lower layers of the Inception V3 are very helpful for any custom image classification task. In our implementation, we have removed the top layer of Inception V3 which is a softmax layer for the default 1000 category classification of ImageNet. Instead, we have added a Global Average Pooling layer, a fully connected layer of 1024 neurons and a final fully connected layer with 5 neurons and a softmax activation function which maps the outputs for our classification categories. The softmax activation function is the function which reports the probabilities of the input being of a particular category. It is the standard function used in case of multi-class classification problems. Its equation is given as follows:

$$\begin{aligned}
 &\text{If } z^{[L]} = w^{[L]} * a^{[L-1]} + b^{[L]} \\
 &\text{Then} \\
 &t = e^{(z^{[L]})} \\
 &a^{[L]} = \frac{t_j}{\sum_{j=1}^n (t_j)} , \text{ Softmax function equation} \quad \text{----- (1)}
 \end{aligned}$$

Where L = layer number,

$z^{[L]}$ = Intermediate value at layer L,

$w^{[L]}$ = Weights of layer L,

$a^{[L-1]}$ = Activations of previous layer(outputs of layer L-1, inputs to layer L),

$b^{[L]}$ = Bias value of layer L

These layers are added over the Inception V3 model in a sequence. We have frozen the weights of the lower layer layers of the Inception V3 architecture and retrained only the top two blocks (layer 249 to layer 313). Initially we trained only the top 3 layers which we have added for a few epochs and then we go on to training the top two blocks along with the 3 final layers for more epochs. The optimizer used in our training of the model was adam optimizer with a learning rate of 0.001. The batch size set is of 64. Rest of the parameters of adam were kept default. The important considerations in the input to the Inception network is that we cannot supply an image of size less than 139 pixels height and width. The image size goes on decreasing as the convolution operations are applied which is governed by the formula :

$$n_h^{[l]} = \left(\frac{n_h^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right), \quad n_w^{[l]} = \left(\frac{n_w^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right) \quad \text{-----}(2)$$

Output Image dimensions = (n_h, n_w, n_c)

Where n_h = Height of image (number of pixels),

n_w = Width of image (number of pixels),

$[l]$ = Layer number,

p = Padding applied to image,

f = Number of filters applied on image in a convolution layer,

s = Strides applied in a convolution layer

n_c = Number of channels, also referred to as depth

If the image size in a layer decreases upto a limit where the image height and width become less than the filter size of a convolution layer, then mathematically, this operation is not possible. This causes the training to stop as an error occurs. The optimum sizes that can be supplied as input are 299/256/224/192/139 pixels. Increasing the image size to a maximum 299 can give a good accuracy but trains slowly as the input image is large enough. Figure 6.2 below shows the Top 2 blocks of Inception V3.

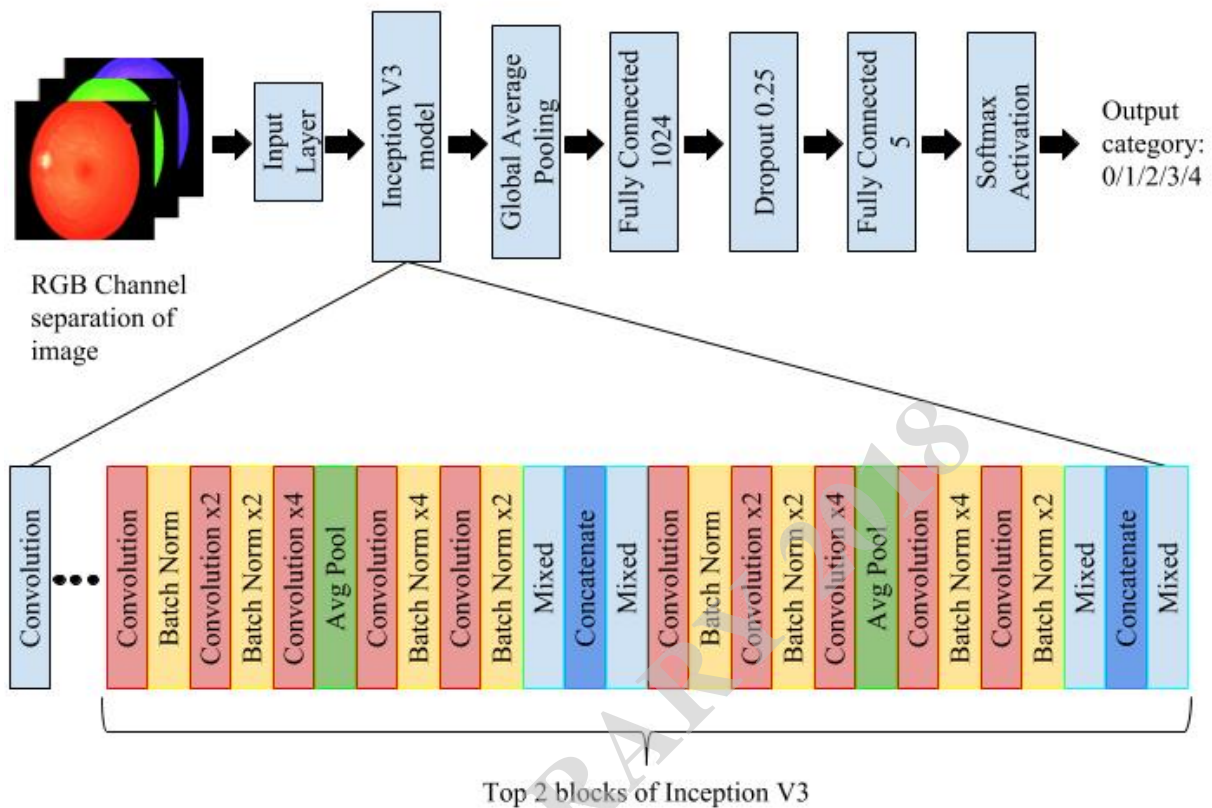


Figure 6.2: Top 2 Blocks of Inception V3

The training has been done on two GPUs - Nvidia GeForce 940MX with VRAM 4GB and Nvidia GTX 1060 with VRAM 6GB. The training time varied drastically on both the GPUs. It took us 10 hours to train on Nvidia 940MX as opposed to just 1.5 hours for the same task on the GTX 1060 owing to the large number of CUDA cores in the later one and increased capacity of the VRAM to hold the model parameters. Using a faster GPU with a larger VRAM is beneficial in terms of performance as well as accuracy. The memory limitations can hamper the models performance and may cause a decrease in the accuracy because it may dump some values out of the memory. This is done in order to keep the model training going.

6.2 Working of the project

6.2.1 Code

```
from keras.applications.inception_v3 import InceptionV3
from keras.preprocessing import image
from keras.models import Model, load_model
from keras import optimizers
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras import regularizers
import keras.backend as K
K.set_image_data_format('channels_last')
K.set_learning_phase(1)
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.inception_v3 import preprocess_input
from sklearn.metrics import roc_auc_score
from tkinter import filedialog
from tkinter import *
import numpy as np
from keras.initializers import glorot_uniform
from keras.utils import to_categorical
from keras.utils.vis_utils import model_to_dot
from keras.utils import plot_model
import time
import matplotlib.pyplot as plt

tic = time.time()
size = 299
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
```

```

# asking user if the model is already trained or not,
# If 'n' then the training starts, if 'Y' then pretrained model is loaded
a = input("Already trained the model? Y/n: ")

if a == 'n':
    # preprocessing and real time data augmentation of training set images
    train_data = ImageDataGenerator(rescale = 1./255, horizontal_flip = True,
        vertical_flip = True,samplewise_std_normalization=True,
        shear_range = 0.2, zoom_range = 0.1)

    # preprocessing and real time data augmentation of test set images
    test_data = ImageDataGenerator(rescale = 1./255, horizontal_flip = True,
        vertical_flip = True,samplewise_std_normalization=True,
        shear_range = 0.2, zoom_range = 0.1)

    # Creates a DirectoryIterator object for getting images from the
    # directory specified with images in sub directories 0,1,2,3,4 for
    # train set
    X_train = train_data.flow_from_directory( 'C:/Users/nachiket/Desktop/
        SEM_8/BE_project/final_dataset/Final_dataset_train',
        target_size = (size, size),batch_size = 64,
        class_mode = 'categorical')
    label_map = (X_train.class_indices)
    print(label_map)

    # Creates a DirectoryIterator object for getting images from the
    # directory specified with images in sub directories 0,1,2,3,4 for
    # train set
    X_test = test_data.flow_from_directory('C:/Users/nachiket/Desktop/
        SEM_8/BE_project/final_dataset/Final_dataset_test',

```

```
target_size = (size, size), batch_size = 64,
class_mode = 'categorical')
label_map1 = (X_test.class_indices)
print(label_map1)

input_shape = (size, size, 3)
# create the base pre-trained model
base_model = InceptionV3(weights='imagenet', include_top=False)

# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x = Dense(1024, activation='relu',
        kernel_regularizer=regularizers.l2(0.01),
        name = 'FC_sftm')(x)
x = Dropout(0.25)(x)

# and a logistic layer -- we have 5 classes
predictions = Dense(5, activation='softmax')(x)

# this is the model we will train
model = Model(inputs=base_model.input, outputs=predictions)

# first: train only the top layers (which were randomly initialized)
# i.e. freeze all convolutional InceptionV3 layers
for layer in base_model.layers:
    layer.trainable = False

# compile the model (should be done *after* setting layers to
```

```

# non-trainable)
model.compile(optimizer=optimizers.Adam(lr=0.001,
    beta_1=0.9, beta_2=0.999), loss='categorical_crossentropy',
    metrics = ['accuracy'])

# train the model on the new data for a few epochs
#steps per epoch = number of samples/batch_size
hist1 = model.fit_generator(X_train,steps_per_epoch = (21667/64),
    epochs = 4,validation_data = X_test,validation_steps = 40,
    initial_epoch = 4)

# at this point, the top layers are well trained and we can start
# fine-tuning convolutional layers from inception V3. We will freeze
# the bottom N layers and train the remaining top layers.

# let's visualize layer names and layer indices to see how many layers
# we should freeze:
for i, layer in enumerate(base_model.layers):
    print(i, layer.name)

# we chose to train the top 2 inception blocks, i.e. we will freeze
# the first 249 layers and unfreeze the rest:
for layer in model.layers[:249]:
    layer.trainable = False
for layer in model.layers[249:]:
    layer.trainable = True

# we need to recompile the model for these modifications to take effect
#from keras.optimizers import ADAM
model.compile(optimizer=optimizers.Adam(lr=0.001, beta_1=0.9,

```

```

        beta_2=0.999), loss='categorical_crossentropy',
        metrics = ['accuracy'])

# we train our model again (this time fine-tuning the top 2
# inception blocks alongside the top Dense layers
hist2 = model.fit_generator(X_train,
                            steps_per_epoch = (21667/64), epochs = 10,
                            validation_data = X_test,
                            validation_steps = 40, initial_epoch = 2,
                            callbacks = [keras.callbacks.TensorBoard
                            (log_dir = 'C:/Users/nachiket/Desktop/SEM_8/
                            BE_project/logging', histogram_freq = 2, batch_size = 64,
                            write_graph = True, write_grads = True,
                            write_images = True)])

model.save('tp3.h5')

def plot_training(history):
    acc = history.history['acc']
    val_acc = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(len(acc))

    plt.plot(epochs, acc, 'r.')
    plt.plot(epochs, val_acc, 'r')
    plt.title('Training and validation accuracy')

    plt.figure()
    plt.plot(epochs, loss, 'r.')

```

```

plt.plot(epochs, val_loss, 'r-')
plt.title('Training and validation loss')
plt.show()

plot_training(hist2)

elif a == 'Y':
    # loading a saved model
    model = load_model('Inception_retrained.h5')
    model.compile(loss = 'categorical_crossentropy', optimizer = 'adam',
    metrics = ['accuracy'])

    print('Weights and model loaded')

# preprocessing and real time data augmentation of test set images
test_data = ImageDataGenerator(rescale = 1./255,
    samplewise_std_normalization=True)
# Creates a DirectoryIterator object for getting images from the directory
# specified with images in sub directories 0,1,2,3,4 for train set
X_pred = test_data.flow_from_directory('C:/Users/nachiket/Desktop/SEM_8/
    BE_project/Codes/predict', shuffle = False, target_size = (size, size),
    batch_size = 64, class_mode = None)
op = model.predict_generator(X_pred)

toc = time.time()
print("Time taken= " + str((toc-tic)) + "s")

```

Chapter 7

Testing

7.1 Test Cases

The results of the test cases have been summarized in the below table 7.1 followed by a detailed explanation of each test case.

Table 7.1: Results of test cases

Test case ID	Test Case(Actions performed)	Expected result	Actual Outcome	Result
1	Supplying CSV file format for prediction rather than image file formats such as PNG, TIFF, JPG, BITMAP	No output	Empty list	Pass
2	Supplying images for prediction on images (general purpose objects) for which it is not trained.(retinal images)	Low confidence levels	Low confidence levels	Pass
3	Testing with retinal images of animals	Low confidence levels	Low confidence levels	Pass

Test case ID	Test Case(Actions performed)	Expected result	Actual Outcome	Result
4	Using artificially generated retinal images using paint.exe to fool the system pertaining to close resemblance to training set images	Low confidence levels	Low confidence levels	Pass
5	Testing on blur test set images	Low confidence levels	Ambiguous results	Fail
6	Time consumption stress testing	Result display time proportional to number of images provided without latency	Correct outputs within expected time limit for output display	Pass

7.1.1 Supplying other file formats for prediction rather than image file formats

We choose to test the model with a CSV file. The CSV file used here is just random file containing some numbers which is totally irrelevant to the project.



The screenshot shows a Jupyter Notebook interface. At the top, there is a toolbar with buttons for undo, redo, and insert. Below the toolbar is a table with 10 columns labeled A through J and 8 rows. The first two columns (A and B) contain data, while the rest are empty. Below the table, there is a code cell with the following text:

```
Found 0 images belonging to 4 classes.
[]
Time taken= 76.17048048973083s
>>>
```

	A	B	C	D	E	F	G	H	I	J
1	1	100								
2	2	200								
3	3	300								
4	4	400								
5	5	500								
6	6	600								
7	7	700								
8	8	800								

Figure 7.1: Testing the system with CSV file format

Description: The Keras API has a function `flow_from_directory()`, which in literal terms means that the data (images in our case) flows from the subdirectories to the program execution in the GPU. This function takes in the directory as an argument in which our subdirectories lie. It should contain one subdirectory per class. Any PNG, JPG, BMP, PPM or TIFF images inside each of the subdirectories directory tree will be included in the generator. But as we can see from the above Figure 7.1 that by placing a .csv file in 4 sub directories, the function does not detect this file format as legal and reports no images detected. Eventually, it returns an empty array. The user must be careful to provide only image file formats such as PNG, JPG, BMP and TIFF to the machine learning model.

7.1.2 Supplying such images for prediction on which it wasnt trained

Just for the sake of some harmless testing, we supply images of some general purpose objects such as cats, dogs, birds, truck etc. and see what prediction it gives us.

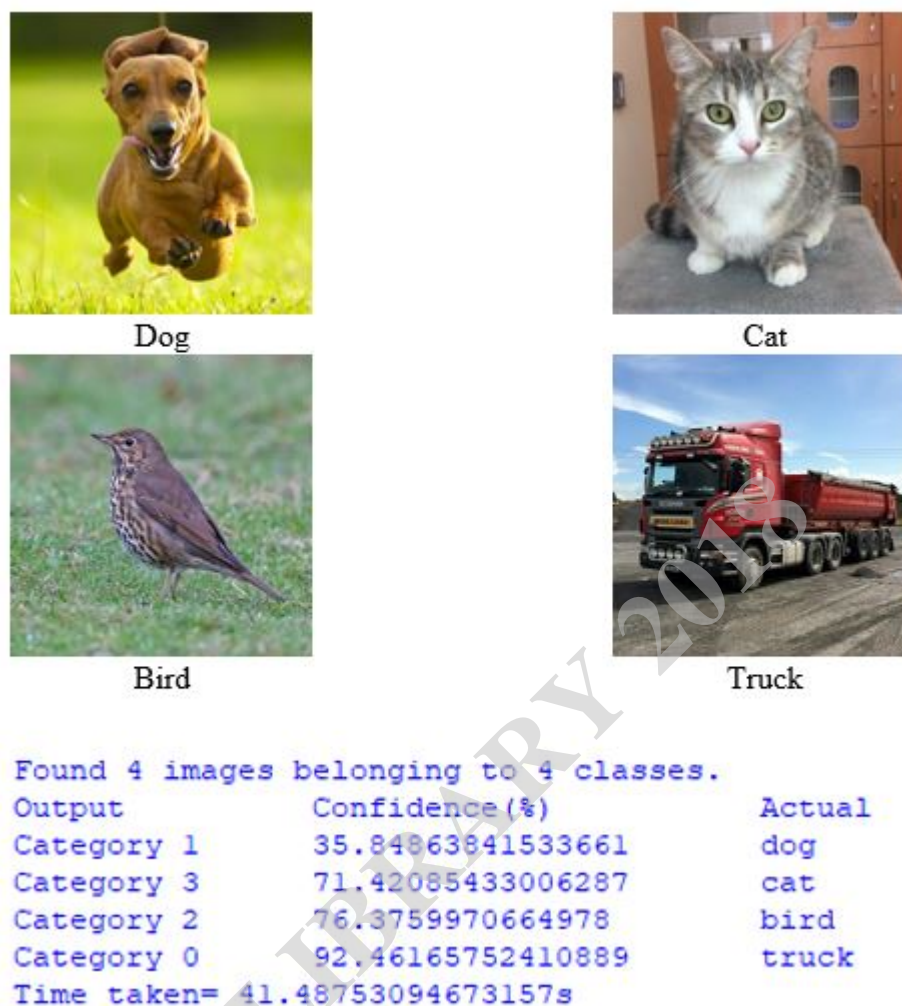


Figure 7.2: Testing the system with external images

Description: As shown in the above Figure 7.2, the images supplied to our model for prediction were of a dog, cat, bird and a truck. As seen in the output, the model tries to predict the category of each of these images for a DR category. But the confidence level (probability) of each prediction is very low. For correct predictions on actual retinal images, our model gives us a confidence level of above 97 percentage and hence, these results are undesirable and can be discarded straight away. In this the model has proven to be successful to not recognize images which are not retinal images in the first place.

7.1.3 Testing with retinal images of animals

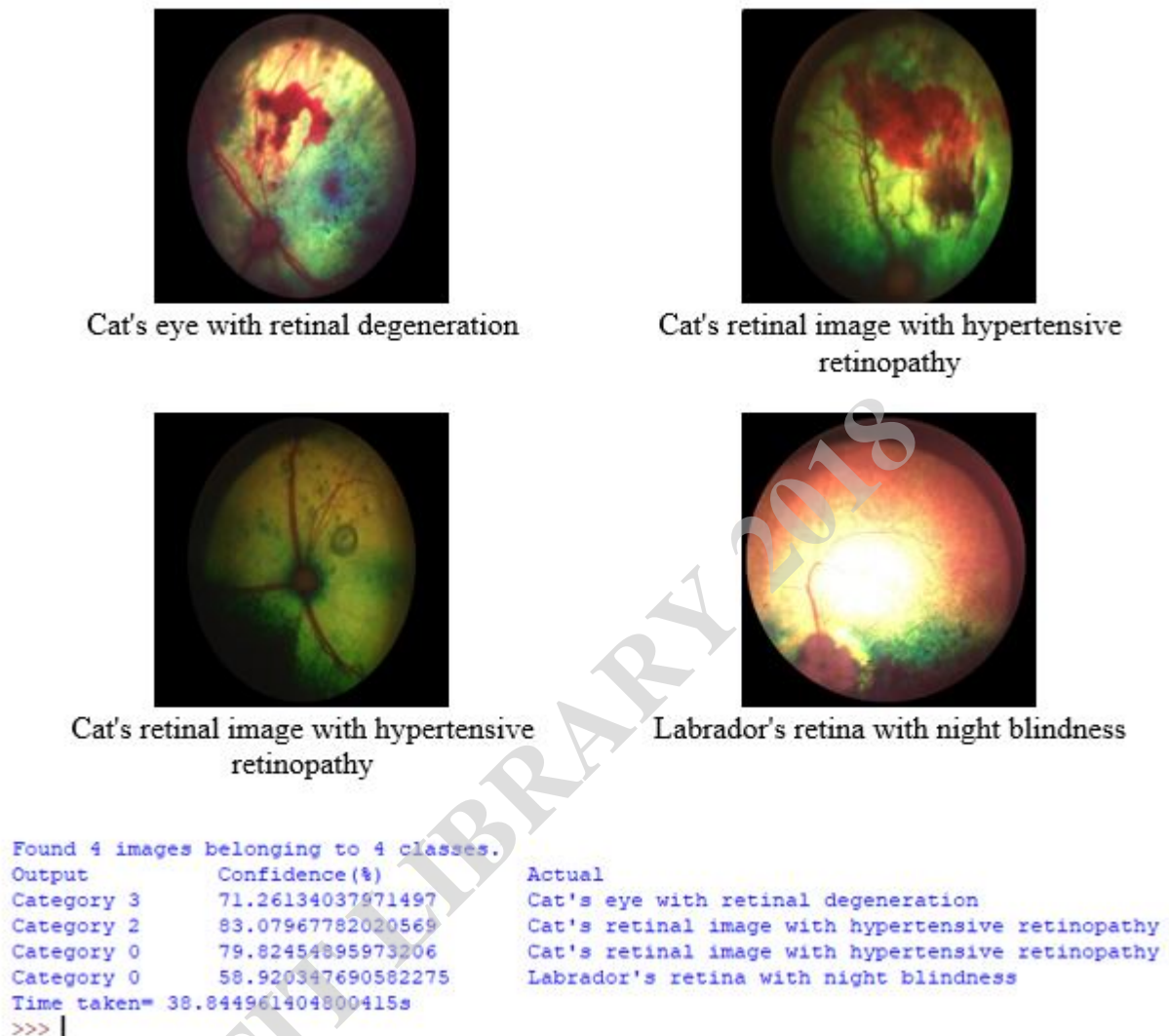


Figure 7.3: Testing the system with retinal images of animals

Description: As shown in the above Figure 7.3, retinal images of four different animals were supplied. In the above case, we can see that on supplying retinal images of animals which hold close resemblance to human retina, the model still fails to give result with a weak confidence level. This proves that the model knows the difference between human retina and animal retina although it wasn't trained to recognize animal retina. The training has been done on 20,000 retinal images of human diabetic patients and it clearly knows to make out the difference between these images. The confidence level are low enough to completely discard these results as being authentic and hence we can say that the model performs good enough in recognizing only human retinal images.

7.1.4 Testing the system with artificially generated images

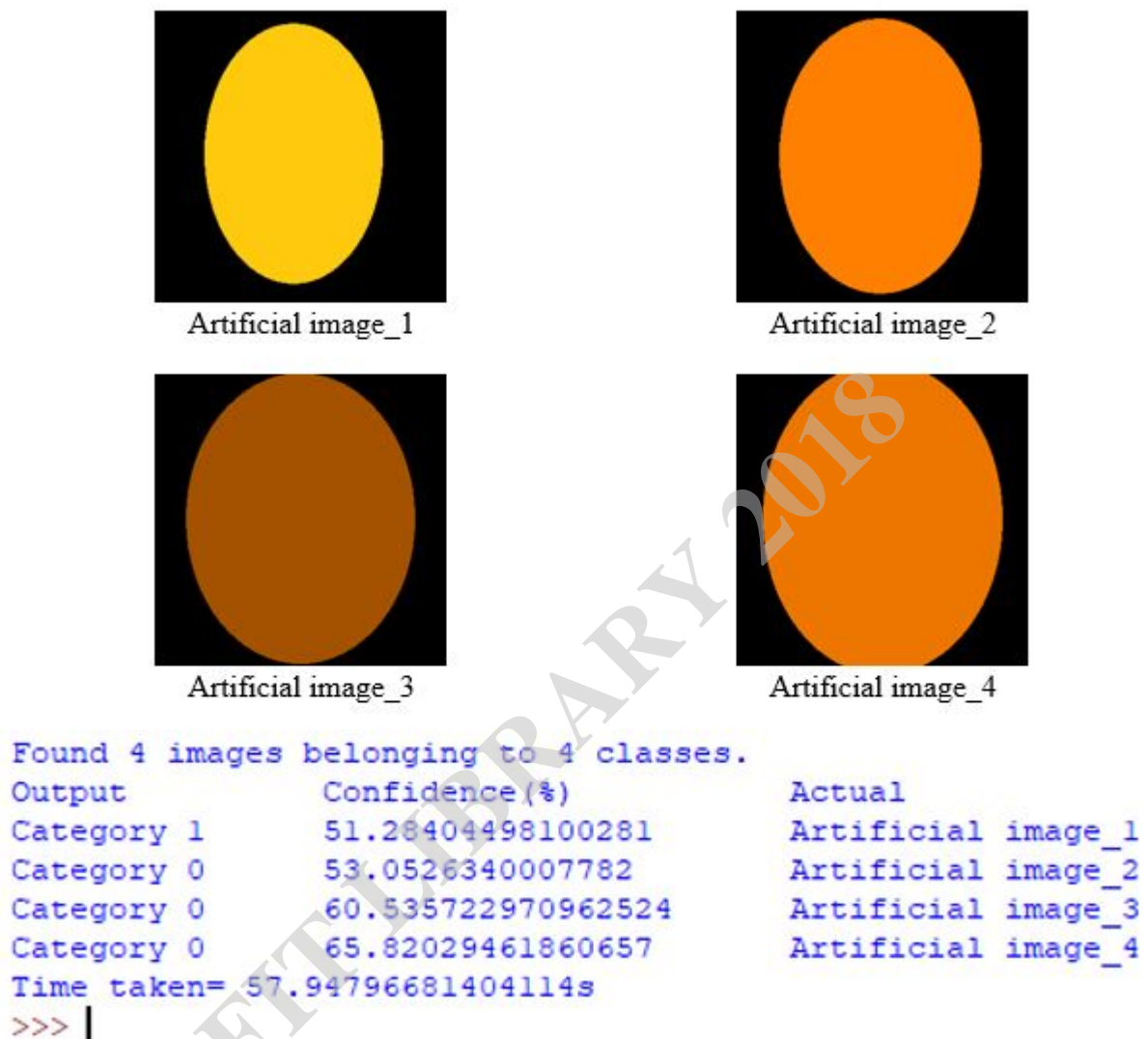


Figure 7.4: Testing the system with artificially generated images

Description: As shown in the above Figure 7.4, the input provided to the model this time were some images created in image editing software like paint. These images look like the images in the dataset which have an orangish or brownish circle at the center of the image with a black background. Since CNNs work on the pixel intensities, these images were thought to be able to fool the system. But, the empirical testing shows that the algorithm does not recognize these images too as being of the retina. With very low confidence percentage, we can say that the model does not identify these images as correct images of a retina and hence the results can be easily discarded.

The model seems to work pretty well on the images it has been trained on and also on images it hasnt been trained on.

7.1.5 Testing the system with blur retinal images

```

Output          Confidence (%)          Actual
Category 1      94.85124349594116        0
Category 0      96.1060643196106         1
Category 0      87.15759515762329        3
Category 4      99.62982535362244        4
Time taken= 36.35705852508545s
>>> |

```

Figure 7.5: Testing the system with blur retinal images

Description: As shown in the above Figure 7.5, blurred retinal images were supplied. Some images from the test dataset were blurred using median blur image processing in order to check robustness of model against blur images. As we can see that the algorithm looks to be confused. This is a common problem faced in many image recognition problems such as self-driving cars, hand-written digit recognition etc. The solution to avoid inaccuracy due to blurring is to add more blurred images to the training dataset, thus preparing the algorithm to face any kind of input.

7.1.6 Time consumption

Table 7.2 below shows the time taken for batch prediction for specified number of images.

Table 7.2: Time taken for batch prediction

Trial number	Number of images	Time taken(seconds)
1	4	34.34
2	20	2.89
3	40	6.25
4	100	16.72
5	200	21.42
6	400	34.92

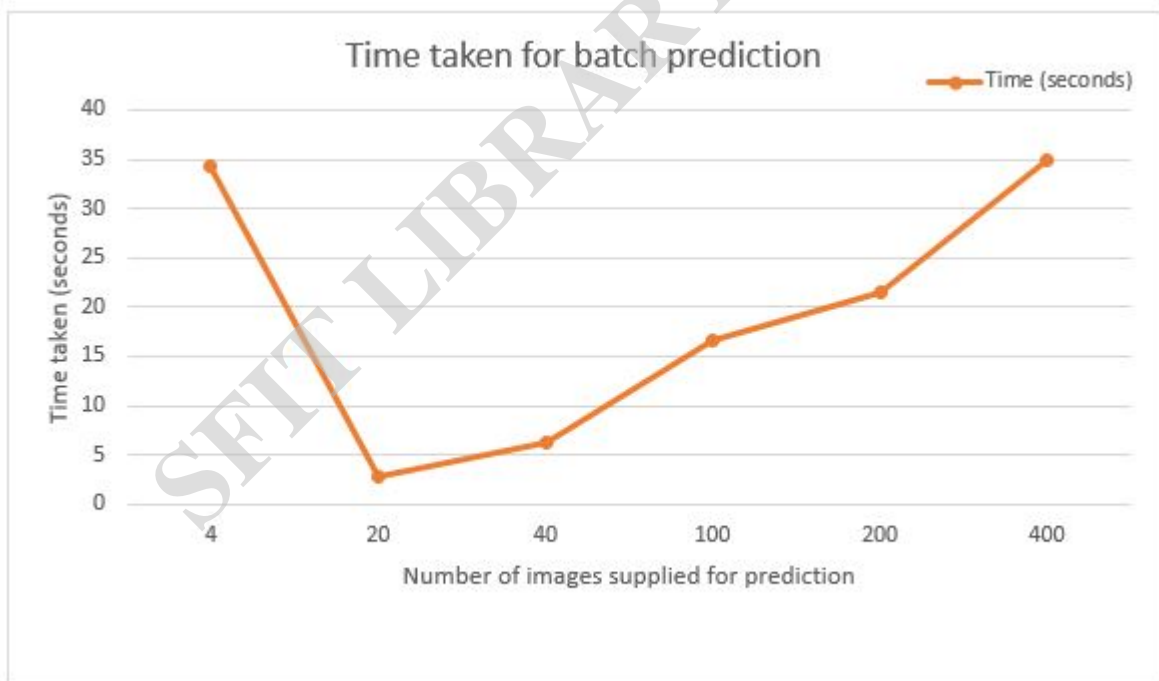


Figure 7.6: Graph of time taken for batch prediction

Description: We test the model to find average time taken to report predictions from limited number of images. We conduct some batch testing instead of supplying individual images and check to see how much time it takes to display the results. The image size ranges between 18 Kb to 26 Kb. The images used in this test case have an average image size of 20.83 Kb each. Figure 7.6 above shows the graph of time taken for batch prediction. These results of time consumption have been obtained by using images from the test set. In the first trial, the trained and saved model is loaded into the GPU's VRAM. This takes a little time because the size of the model is 197 Mb. Once, the model is loaded, the predictions for various batch sizes look to increase linearly. The first trial takes approximately 35 seconds, but once it has been loaded, predictions results are reported in a really short period of time which can be clearly seen in the above table and its graphical representation. This proves the usefulness of this kind of a model which can facilitate lightning fast results even in remote areas with internet access. Although this model hasn't been deployed on the web, we cannot account for the latency in the network, but can conclude that it is really fast in case of result computation.

7.2 Type of Testing Used

An ample of software testing paradigms exist for standard application testing, but only some of them apply to machine learning applications. This is because Machine-learning applications are new in the market. Such ML applications testing is conducted only after deployment for public use. Modifications are implemented as needed. We still provide some testing strategies we have used in our application.

7.2.1 Black Box Testing

Black Box Testing, also known as Behavioral Testing is a software testing method in which the tester does not know the internal structure/design/implementation of the item being tested. These tests can be functional or non-functional, though usually functional. Here we use the techniques such as Equivalence Partitioning and Cause-Effect Graph.

Equivalence Testing: A software test design technique involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data. This testing has been carried out in the training phase and its description is in the results section. We faced the problem of imbalanced dataset since each of our classes did not have the same number of images previously. This problem was overcome by using data augmentation due to which there has been a class equalization of our dataset and the algorithm performance increased.

Cause-Effect Graph: We have used this technique in the test case number 2, 3, 4, 5. The test case 2 enlightens the effect of no desirable confidence levels which was caused by providing certain images to the algorithm for prediction for it wasn't trained. In test case 3, the effect of not having high confidence levels was because of providing retinal images of animals with various symptoms. The algorithm is robust against such discrepancies and shows good performance. In test case 4 the algorithm again produces bad results which are discarded because artificially generated images (resembling the retina) we supplied to it for prediction. The algorithm seems to recognize that these are not legal images and passes in this test too. Finally, in test case 5, some of the test set images were blurred due to which poor results were produced for prediction. This suggests that the algorithm performance must be improved and made tolerant towards blur images because in real world deployment, it might face such cases and still must perform perfectly. Also in test case 1, we have provided a different path of directory where excel files reside, rather than image files which must be the default input to our algorithm. In this case, the algorithm does not provide any output which is a desired outcome for other file formats.

7.2.2 Stress Testing

As per test case 6, batches of various image quantities were supplied to the algorithm to check for any latency in the computational process. In our case, the image batches were supplied back to back without stopping, in order to check if such stress testing slows down the GPU. But it passed and shows a linear relationship of time versus the batch size which displays normal functioning of the algorithm.

Chapter 8

Results and Discussions

8.1 Results

The summary of our training can be found in the below table 8.1.

Table 8.1: Summary of Training

Sr.no.	Number of images per category					Channel mode	Num - ber of epochs	Train- ing ac- curacy	Valida - tion accu- racy
	0	1	2	3	4				
1.	5855	4964	5844	4311	3163	Gray- scale	20	90%	30%
2.	5855	4964	5844	4311	3163	Color	30	90.2%	82.96%
3.	5855	4964	-	4311	3163	Color	20	94%	92%

Each row in the above table corresponds to various approaches that were tried out during the training. In the first approach, all the coloured (3 channel) images in the dataset were converted to grayscale images thereby reducing the number of channels to 1. The sole reason for using grayscale images was to reduce the computations required for training and hence reduce the time required for training the model. After training the model for 20 epochs, the training accuracy achieved was 90%, however the validation accuracy was just 30%. It was observed that the validation accuracy saturated to 30% after 15 epochs. This model failed because while converting the images to grayscale the important features of the images were lost. Thus, it was inferred that using single channel images did not help in achieving the desired accuracy and hence colored images need to be used for further training.

In the second model training, we used color images. This time, we did not train the upper 3 layers for a few epochs, but started training them along with the top two blocks, hence the low accuracy and we can see that the network is overfitting upto some extent. Also, in this training run, we faced the issue that the bias towards category 2 was larger than other categories and the prediction defaulted to category 2 for any category image provided. To mend this, we conducted another training run without the images for category 2, i.e. a classification for only 4 categories. This time we achieved a higher accuracy and the predictions given by the model were almost correct in each case. We can say that the category 2 images are pretty much same as any other category and due to the dearth of significant features present in them, we might be able to say that the previous model got confused and gave us some wrong predictions.

Figure 8.1 below shows the graph of training and validation accuracy and Figure 8.2 below shows the graph of training loss and validation loss for the second approach.

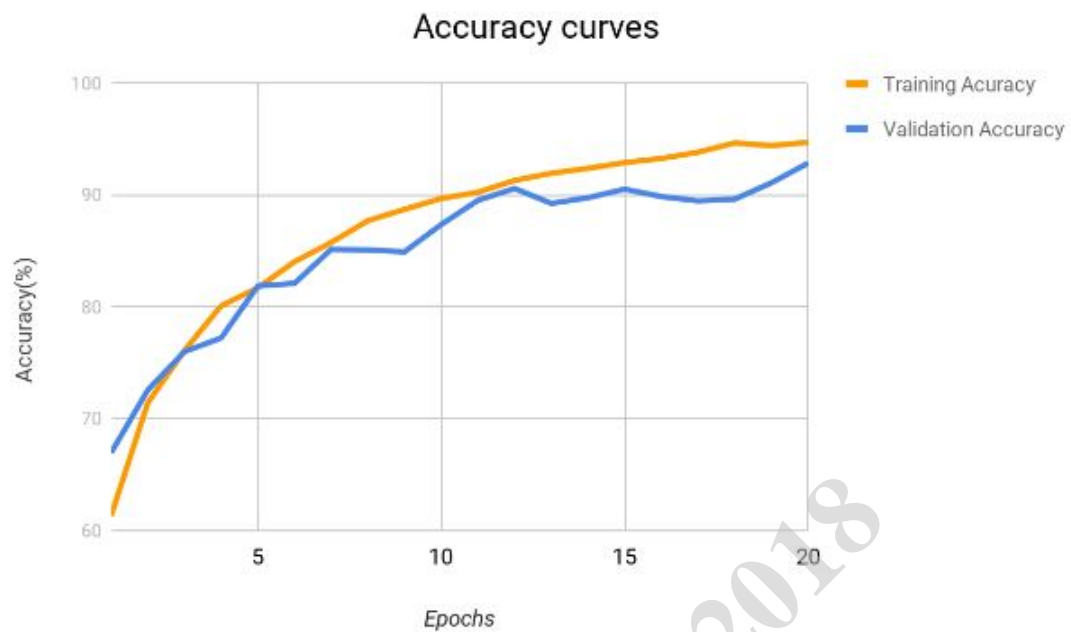


Figure 8.1: Graph of Accuracy

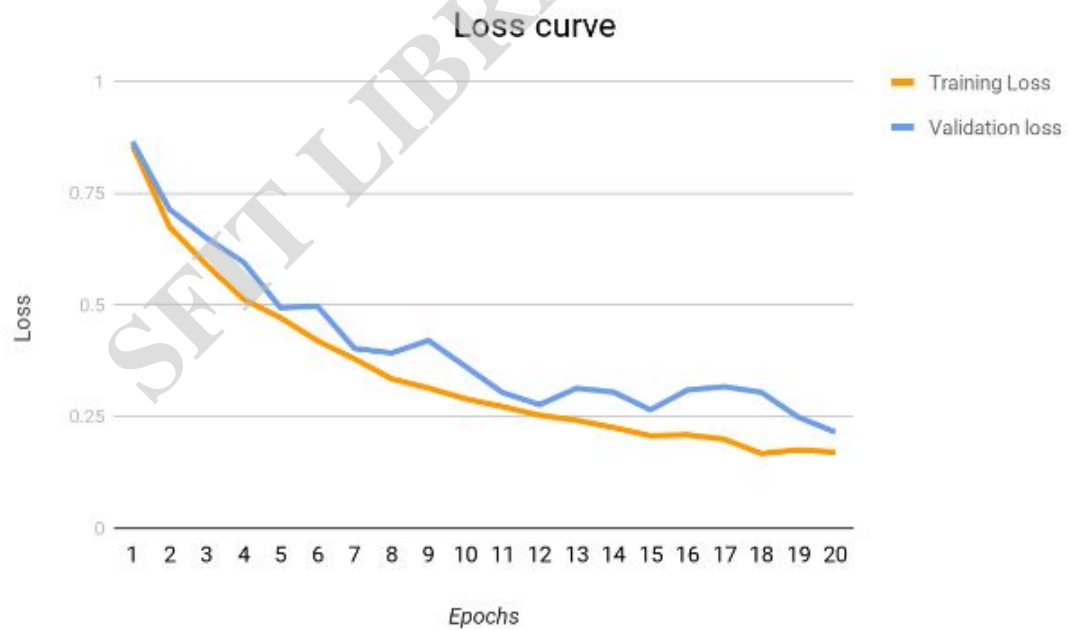


Figure 8.2: Graph of Loss

8.2 Discussion

This project has been done using Inception V3 as our base model. We have tried multiple approaches using the same base model. The further work that can be done is to use another such base model. There are multiple pretrained models available such as ResNet, DenseNet, HighwayNet, MobileNet etc. Once a conclusion is found that which of our above mentioned approaches work and gives perfect results, one must try training the ResNet model. It is again, a state of the art network which gives a tough competition to Inception in the case of ImageNet large scale visual recognition challenge. Although the ResNet takes more time to train since it is deeper than Inception V3, the results must be compared in order to check which model fits the best for this particular retinal image classification task.

Chapter 9

Conclusion and Future Scope

9.1 Conclusion

Diabetic Retinopathy is a rapidly growing cause of blindness all over the world. The estimate of affected population is predicted to rise to from 31 million to 79 million in the next decade. This project is an aid to the ophthalmologists in the early detection and prevention of Diabetic Retinopathy. The screening and diagnosis facilities in India are limited to urban areas and there is a dearth of such resources in remote rural areas. We have developed this deep learning model which has been successful in achieving state of the art accuracy in image classification which are comparable to a clinical setup. The aim of this project is provide convenient access to rural areas for screening purposes where access to trained ophthalmologists is limited.

These accurate results were obtained due to transfer learning and data augmentation. Data augmentation helped us tackle the problem of class imbalance which is a crucial factor while training CNNs. Also transfer learning done using the Inception V3 architecture has proved fruitful because of its virtue of generalization of low level features.

9.2 Future Scope

Due to less number of images of category 3 and 4, our network is biased towards category 2. This is the reason we have not included category 2 in our final model. So our main aim in the future will be to collect images of category 3 and 4 in order to equalize the dataset for all categories. Secondly, we will try to collect real-life data from hospitals all over India in order to study any changes in the retina due to different geographical locations. After collecting retinal images, we will train a model from scratch on this data. Finally, alongwith Diabetic Retinopathy we will also try to detect Macular Edema which is one of the major causes for vision loss.

Appendix

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.

Back-Propagation: Back-propagation is a method used in artificial neural networks to calculate the error contribution of each neuron after a batch of data (in image recognition, multiple images) is processed. This is used by an enveloping optimization algorithm to adjust the weight of each neuron, completing the learning process for that case.

Pooling: Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Another example is average pooling, which uses the average value from each of a cluster of neurons at the prior layer.

Fully Connected: Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP).

Weights: CNNs share weights in convolutional layers, which means that the same filter (weights bank) is used for each receptive field in the layer; this reduces memory footprint and improves performance.

References

- [1] Wikipedia contributors. *Diabetic retinopathy* — *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Diabetic_retinopathy&oldid=805819451. [Online]. 2017.
- [2] Andrej Karpathy. *Stanford University CS231n: Convolutional Neural Networks for Visual Recognition*. <http://cs231n.stanford.edu/>. [Online]. Mar. 2017.
- [3] Gulshan V et al. “Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs”. In: *JAMA* 316.22 (2016), pp. 2402–2410. DOI: 10.1001/jama.2016.17216. eprint: /data/journals/jama/935924/joi160132.pdf. URL: +%20http://dx.doi.org/10.1001/jama.2016.17216.
- [4] Mohammad Mohtashim. *Artificial Intelligence Neural Networks*. https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm. [Online]. Sept. 2006.
- [5] Denny Britz. *Understanding Convolution Neural Networks for NLP - WildML*. <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp.html>. [Online]. Nov. 2015.
- [6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.
- [7] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255.
- [8] Geoffrey E Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).
- [9] Christian Szegedy et al. “Going deeper with convolutions”. In: *Cvpr*. 2015.

Acknowledgements

We hereby take the privilege to present our project report on Detecting Diabetic Retinopathy Category from Retinal Images using Deep Convolution Neural Networks. We are very grateful to our Project Supervisor Dr. Vaishali Jadhav for contributing her valuable moments in our Project from her busy and hectic schedule right from the projects inception. Being after us like a true mentor and great academic parent. She has been a constant source of inspiration to us. Her suggestions have greatly contributed for the betterment of our project.

Our special thanks to the Head of Department Dr. Joanne Gomes, staff members and lab assistants for their co-operation.