**DIABETIC RETINOPATHY DETECTION**

**A PROJECT REPORT**

***Submitted by***

**ANDHI B** **211419104011**

**SOWMIYA K**  **211419104261**

**THARANI EMMANUEL E 211419104326**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

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

**APRIL 2023**

**DIABETIC RETINOPATHY DETECTION**

**A PROJECT REPORT**

***Submitted by***

**ANDHI B** **211419104011**

**SOWMIYA K**  **211419104261**

**THARANI EMMANUEL E 211419104326**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

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

**APRIL 2023**

**PANIMALAR ENGINEERING COLLEGE**

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

**BONAFIDE CERTIFICATE**

Certified that this project report **“DIABETIC RETINOPATHY DETECTION”** is the bonafide work of **“ANDHI B [REGISTER NO: 211419104011], SOWMIYA K [REGISTER NO: 211419104261] and THARANI EMMANUEL E [REGISTER NO: 211419104326]”** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **Dr.L.JABASHEELA,M.E.,Ph.D**  **HEAD OF THE DEPARTMENT**  DEPARTMENT OF CSE,  PANIMALAR ENGINEERING COLLEGE,  NAZARATHPETTAI,  POONAMALLEE,  CHENNAI-600 123 | **SIGNATURE**  **M.MAHESWARI, M.Tech.,**  **SUPERVISOR**  **ASSISTANT PROFESSOR,**  DEPARTMENT OF CSE,  PANIMALAR ENGINEERING COLLEGE,  NAZARATHPETTAI,  POONAMALLEE,  CHENNAI-600 123. |

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**DECLARATION BY THE STUDENT**

We **ANDHI B [REGISTER NO: 211419104011], SOWMIYA K [REGISTER NO: 211419104261] and THARANI EMMANUEL E [REGISTER NO: 211419104326]** hereby declare that this project report titled “**DIABETIC RETINOPATHY DETECTION”**, under the guidance of **M. MAHESWARI, M.Tech.,**  **ASSISTANT PROFESSOR** , is the orginial work done by us and we have not plagiarized or submitted to any other degree in any university by us.

1. **ANDHI B**
2. **SOWMIYA K**
3. **THARANI EMMANUEL E**

**ACKNOWLEDGEMENT**

We would like to 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 express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide**  **Mrs. M. MAHESWARI, M.Tech.,**  and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

**ANDHI B**

**SOWMIYA K**

**THARANI EMMANUEL E**

**ABSTRACT**

Detecting the abnormalities in the human eye is extremely difficult due to the various complexities associated with the process. Retinal images captured by digital cameras can be used to identify the nature of the abnormalities affecting the human eye. Conventional disease identification techniques from retinal images are mostly dependent on manual intervention. Human observation is highly prone to error, the success rate of these techniques is quite low. Diabetic Retinopathy is one such disease of retina which occurs in people suffering from long standing diabetes. It is a multistage progressing disease namely Non Proliferative Diabetic Retinopathy and Proliferative Diabetic Retinopathy. Microaneurysms, haemorrhages and exudates are the abnormal features commonly observed in the retinal image of a person affected by diabetic retinopathy. Image processing techniques are used to pre-process the fundus image, which is followed by segmentation of anomalies. Feature extraction is done and the detected features are used to classify the different stages of diabetic retinopathy that are Normal , Mild , Moderate,Severe and Proliferactive. The classification technique used is Random Forest.

**TABLE OF CONTENTS**

| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
|  | **ABSTRACT** | Vi |
|  | **LIST OF TABLES** | Ix |
|  | **LIST OF FIGURES** | Ix |
|  | **LIST OF ABBREVIATIONS** | X |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 Problem Definition | 2 |
|  | 1.2 Scope of the project | 2 |
| **2.** | **LITERATURE SURVEY** | 4 |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | 3.1 Existing System | 13 |
|  | 3.2 Proposed system | 13 |
|  | 3.3 Feasibility Study | 14 |
|  | 3.4 Software Requirements | 17 |
|  | 3.5 Hardware Requirements | 17 |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1 Data dictionary | 19 |
|  | 4.2 Data Flow Diagram | 20 |
|  | 4.3 UML Diagrams | 22 |
|  |  |  |
| **5.** | **SYSTEM ARCHITECTURE** |  |
|  | 5.1 Module Design | 27 |
|  | 5.2 Algorithms | 32 |
| **6.** | **SYSTEM IMPLEMENTATION** |  |
|  | 6.1 Client-side coding | 37 |
|  | 6.2 Server-side coding | 39 |
| **7.** | **SYSTEM TESTING** |  |
|  | 7.1 Test Cases | 67 |
| **8.** | **CONCLUSION** |  |
|  | 8.1 Conclusion | 70 |
|  | 8.2 Future Enhancements | 70 |
|  | **APPENDICES** |  |
|  | A.1 Sample Screens | 71 |
|  | **REFERENCES** | 74 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TABLE DESCRIPTION** | **PAGE NO.** |
| 7.1 | NORMAL TEST CASE | 69 |
| 7.2 | MODERATE TEST CASE | 69 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG NO.** | **FIGURE DESCRIPTION** | **PAGE NO.** |
| 4.1 | DFD LEVEL 0 | 21 |
| 4.2 | DFD LEVEL 1 | 22 |
| 4.3 | USE-CASE DIAGRAM | 23 |
| 4.4 | CLASS DIAGRAM | 24 |
| 4.5 | ACTIVITY DIAGRAM | 25 |
| 4.6 | SEQUENCE DIAGRAM | 26 |
| 5.1 | SYSTEM ARCHITECTURE DIAGRAM | 28 |
| 5.2 | PREPROCESSING | 30 |
| 5.3 | SEGMENTATION | 31 |
| 5.4 | FEATURE EXTRACTION | 32 |
| 5.5 | CLASSIFICATION | 33 |
| A.1 | MAIN PAGE | 72 |
| A.2 | LOGIN PAGE | 72 |
| A.3 | UPLOADING NON AFFECTED FUNDUS IMAGE | 73 |
| A.4 | DETECTION OF TYPE OF DISEASE(NORMAL) | 73 |
| A.5 | UPLOADING NON AFFECTED FUNDUS IMAGE | 74 |
| A.6 | DETECTION OF TYPE OF DISEASE(PROLIFERACTIVE) | 74 |

**LIST OF ABBREVATIONS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1. | DR | Diabetic Retinopathy |
| 2. | RF | Random Forest |
| 3. | ML | Machine Learning |

1. **INTRODUCTION**
   1. **INTRODUCTION**
   2. **PROBLEM DEFINITION**

Diabetic retinopathy is a serious sight-threatening complication of diabetes. Diabetes interferes with the body's ability to use and store sugar (glucose). The disease is characterized by too much sugar in the blood, which can cause damage throughout the body, including the eyes Diabetes effects the circulatory system of a person, including that of the retina, which leads to DR. The oxygen supply to the visual system is reduced to a huge extent and it causes swellings on the retinal vessels. Also retinal lesions are formed which includes haemorrhages, microaneurysms and exudates. These are the symptoms for the disease, which will not be visible in the initial stages of the disease. Therefore, unless the patient takes regular examination of the disease, it cannot be identified and thus not cured.

* 1. **SCOPE OF THE PROJECT**

Diabetic Retinopathy has become one of the leading causes of blindness, but it can be cured if diagnosed at an early stage, this requires regular eye examinations. Currently the examinations are done by ophthalmologists or optometrists manually. People from low income backgrounds and rural areas may not be able to afford these regular checkups. Setting up mass screening centers with automated systems can eliminate unnecessary visits to the ophthalmologist. This will also eliminate human intervention and thus reduce the cost.

The automated system will grade the images on the level of severity and refer only those patients who need medical attention to the Ophthalmologist. This will also relieve the burden on the doctor who would otherwise have to go through a lot of images which will come from the mass screening camps.

* 1. **LITERATURE SURVEY**

**2. LITERATURE SURVEY**

* + 1. **TITLE:** Internet of things and deep learning enabled diabetic retinopathy diagnosis using retinal Fundus images.

**AUTHOR:** Thangam palaniswamy, Mahendran vellingiri.

**YEAR :** 2023

**DESCRIPTION:**  Internet of Things (IoT) and computer vision technologies find useful in different applications, especially in healthcare. IoT driven healthcare solutions provide intelligent solutions for enabling substantial reduction of expenses and improvisation of healthcare service quality. At the same time, Diabetic Retinopathy (DR) can be described as permanent blindness and eyesight damage because of the diabetic condition in humans. Accurate and early detection of DR could decrease the loss of damage.

**DISADVANTAGES:** It is difficult in image acquisition and limited user application then highly accurate and difficult to spoof it is highly dependable.

* + 1. **TITLE:**  Graph Adversarial Transfer Learning For Diabetic Retinopathy Classification.

**AUTHOR:** Jingbottu,Huan Wang,YeLu.

**YEAR:** 2022

**DESCRIPTION:** Diabetic retinopathy (DR) is an essential factor that has caused vision loss and even blindness in middle-aged and older adults. A system that can automatically perform DR diagnosis can help ophthalmologists save a lot of tedious work, such as DR grading or lesion detection. At the same time, patients can find their diseases earlier and perform the correct treatment. However, most of the existing methods require many DR annotations to train the model, and the DR data will vary to different degrees due to various shooting tools. The above problems lead to the inefficient use of existing data in the experiment, limiting actual deployment. Train the model by designing two classifiers for adversarial purpose to improve the robustness of model and also to improve classification performance of model.

**DISADVANTAGES:** limited on fixed number of points then time and space complexity higher

* + 1. **TITLE:** Deep learning technique for diabetic retinopathy

**AUTHOR:**  Mohammad Z.Atwany

**YEAR:** 2022

**DESCRIPTION:**  Diabetic Retinopathy (DR) is a degenerative disease that impacts the eyes and is a consequence of Diabetes mellitus, where high blood glucose levels induce lesions on the eye retina. Diabetic Retinopathy is regarded as the leading cause of blindness for diabetic patients, especially the working-age population in developing nations. Treatment involves sustaining the patient’s current grade of vision since the disease is irreversible. Early detection of Diabetic Retinopathy is crucial in order to sustain the patient’s vision effectively. The main issue involved with DR detection is that the manual diagnosis process is very time, money, and effort consuming and involves an ophthalmologist’s examination of eye retinal fundus images

**DISADVANTAGES:** DR grading and classification methodologies from a qualitative point of viewSometime it will not give accurate answer no correlation can be deduced due to varying data distributions. A common issue the papers generally fail to highlight is the extreme amount of time needed to train new models every time.

* + 1. **TITLE:** Multi-stream deep neural networks for diabetic retinopathy severity classification under a boosting framework.

**AUTHOR:** Hamza Mustafa,Syed Farooq ali

**YEAR:** 2022

**DESCRIPTION:** Diabetic Retinopathy (DR) is an eye disorder in patients with diabetes. Detection of DR presence and its complications using fundus images at an early stage helps prevent its progression to the advanced levels. In the recent years, several well-designed Convolutional Neural Networks (CNN) have been proposed to detect the presence of DR with the help of publicly available datasets. However, these existing CNN-based classifiers focus on utilizing different architectural settings to improve the performance of detection task only i.e. presence or absence of DR

**DISADVANTAGES:**  Scarifies intelligibility and interpretability.

* + 1. **TITLE:** A Deep Learning Approach for the Detection of Neovascularization in Fundus Images Using Transfer Learning.

**AUTHOR:** Soo siang teoh,Haidi Ibrahim,Michel chi Seng tang.

**YEAR:** 2022

**DESCRIPTION:** We demonstrated that the proposed method outperforms each individual CNN for neovascularization detection. It also shows better performance compared to another method that utilized deep learning models for feature extraction and Support Vector Machine (SVM) for classification. Patients with diabetes are at risk of developing a retinal disorder called Proliferative Diabetic Retinopathy (PDR). One of the main characteristics of PDR is the development of neovascularization, a condition in which abnormal blood vessels are formed on the retina. This condition can cause blindness if it is not detected and treated early. Numerous studies have proposed different image processing techniques for detecting neovascularization in fundus images.

**DISADVANTAGES:** Abnormal retinal neovascularization could lead to many complications including retinal detachment, haemorrhage and glaucoma.Retinopathy is a relatively common complication of T2DM and is associated with genetic and environmental factors.

* + 1. **TITLE:** Hand gesture re cognition based on a Harris hawks optimized convolution neural network

**AUTHOR:** Thippa Reddy,Guatam sirinivas, koppu,Praveen Kumar reddy

**YEAR:** 2022

**DESCRIPTION:** Hand gesture images. To tune the hyper-parameters of the CNN, a recently developed metaheuristic algorithm, namely, the Harris Hawks Optimization (HHO) algorithm, is used.

**DISADVANTAGES:**  They are neither self-revealing nor self-explanatory. A named button on a toolbar has an explicit purpose and is also easy to find, gestures, however, may be arbitrary and are usually more difficult to discover.

* + 1. **TITLE:** Classification of Eye Diseases in Fundus Images

**AUTHOR:** Omar bernabe, Elena Acevedo.

**YEAR:** 2022

**DESCRIPTION:** Image processing, particularly Deep Learning techniques, is significant support for detecting and diagnosing the latter eye conditions and Glaucoma and Diabetic Retinopathy.Convolutional Neural Networks (CNN) are algorithms from Deep Learning and are very useful.

**DISADVANTAGES:**  Include distortion of images caused by the spherical shape of the globe, eyelash artifacts, false color representation of fundus findings, continued inability to view the far peripheral retina, and much higher equipment costs.

* + 1. **TITLE:** Classification of Diabetic retinopathy severity based on GCA attention mechanism.

**AUTHOR:** Binuha yang,Tongyanli

**YEAR:** 2021

**DESCRIPTION:** GCA mechanism is proposed to update the attention weight of the different channels of feature map with model training process. Diabetic retinopathy (DR) is one of the major complications caused by diabetes and can lead to severe vision loss or even complete blindness if not diagnosed and treated in a timely manner. In this paper, a new feature map global channel attention mechanism (GCA) is proposed to solve the problem of the early detection of DR.

**DISADVANTAGES:** It causes following a temporal artery biospy are uncommon and test is not seen as dangerous.

* + 1. **TITLE:** Diabetic retinopathy early detection based on multifracted geometry analysis for octa macular images using support vector machine.

**AUTHOR:**  Mohammad M.Abdelsalam,M.A.Zahran

**YEAR:** 2021

**DESCRIPTION:** Diabetic Retinopathy (DR) is a complication of diabetes that affects the eyes. It is caused by blood vessel damage of the light-sensitive tissue at the back of the retina. Neovascularization are emerged and the small blood vessels are blocked The classification of DR using multifracted geometry and lacunarity parameters .

**DISADVANTAGES:**  OCTA are it’s limited field of view inability to view leakage inability to detect blood flow They are neither self-revealing nor self-explanatory

* + 1. **TITLE:** Automated microaneursysms detection in Retinal images using radon transform and supervised learning application to mass screening of diabetic retinopathy.

**AUTHOR:**  Meysam Tavakoli,Alireza, Mehdizadeh,Tim ellis

**YEAR:** 2021

**DESCRIPTION:** Detection of red lesions in color retinal images is a critical step to prevent the development of vision loss and blindness associated with diabetic retinopathy (DR). Microaneurysms (MAs) are the most frequently observed and are usually the first lesions to appear as a consequence of DR. Therefore, their detection is necessary for mass screening of DR. However, detecting these lesions is a challenging task because of the low image contrast, and the wide variation of imaging conditions.

**DISADVANTAGES:**  These is sometimes slower to detect so it cause any type vascular disease.

* + 1. **TITLE:** Hybrid CNN-SVD based prominent features extraction and selection for grading Diabetic

**AUTHOR:**  MD Nahiduzzaman,MD.robiulislam

**YEAR:** 2021

**DESCRIPTION:** An early-stage diagnosis is therefore beneficial to prevent diabetes patients from losing their sight. This study introduced a novel method to detect DR for binary class and multiclass classification based on the APTOS-2019 blindness detection and Messidor-2 datasets. First, DR images have been pre-processed using Ben Graham’s approach.

**DISADVANTAGES:** They fail to encode the position and orientation of object.

1. **TITLE:** International diabetes federation

**AUTHOR:** D.Atlas

**YEAR:** 2021

**DESCRIPTION:** The aim is to provide diabetes centres and their personnel with the essential medical equipment to screen people living with diabetes for DR to help manage and prevent this common and costly complication.The project also intends to improve data collection and increase the available data on diabetic eye disease worldwide.

**DISADVANTAGES:** The progression of diabetic retinopathy can get worse, and when you have the mild form of nonproliferative diabetic retinopathy, we estimate about a 6% chance of developing into the proliferative form within 12 months. With moderate nonproliferative diabetic retinopathy, it&#39;s about 20 to 40%.

1. **TITLE:** Identification of diabetic retinopathy through machine learning

**AUTHOR:** M. Bader Alazzam, F. Alassery, and A. Almulihi

**YEAR:** 2021

**DESCRIPTION:** The RBM model of machine learning automatic disease detection performed well in terms of diagnostic accuracy, sensitivity, and application in diabetic retinopathy screening.

**DISADVANTAGES:** SVM algorithm is not suitable for large data sets. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.

1. **TITLE:**  Multicenter, head-to-head, real-world validation study of seven automated artificial intelligence diabetic retinopathy screening systems

**AUTHOR:** A. Y. Lee, R. T. Yanagihara, C. S. Lee .

**YEAR:** 2021

**DESCRIPTION:** Several others are under consideration while in clinical use in other countries, but their real-world performance has not been evaluated systematically. We compared the performance of seven automated AI-based DR screening algorithms (including one FDA-approved algorithm) against human graders when analyzing real-world retinal imaging data.

**DISADVANTAGES:** The disadvantages of multicenter trials relate primarily to cost and to logistical aspects of coordinating research across multiple units

1. **TITLE:** Early detection of diabetic retinopathy using pca-firefly based deep learning model.

**AUTHOR :**T. R. Gadekallu, N. Khare, S. Bhattacharya

**YEAR:** 2020

**DESCRIPTION:** (PCA) is used to extract the most significant features in the dataset. Further, Firefly algorithm is implemented for dimensionality reduction. This reduced dataset is fed into a Deep Neural Network Model for classification. The results generated from the model is evaluated against the prevalent machine learning models and the results justify the superiority of the proposed model in terms of Accuracy, Precision, Recall, Sensitivity and Specificity.

**DISADVANTAGES:**  Such data inputs could produce results that are very much off the correct projection of the data . PCA presents limitations when it comes to interpretability. Since we are transforming the data, features lose their original meaning.

* 1. **SYSTEM ANALYSIS**

**3.SYSTEM ANALYSIS**

* 1. **EXISTING SYSTEM**

All the existing systems are concentrating on segmenting the primary anomalies responsible for initial stage of the disease. Their proposed method is also database dependent and would work only for that specific database. There is no single approach or method for segmenting all the anomalies. Building an optimal feature set is needed as this disease is vision threatening and high benchmarking results need to be achieved. Many classification techniques are proposed with different and novel approaches of pre- processing. All these surveyed papers achieve an average of 85% accuracy which still need to be improved. Researches throughout the world are trying to propose better classification algorithms by building significant feature set. Limitations of the existing systems are defects that arise due to DR compliances which cannot be extracted using single method. Hence there is a need to developed system will overcome and able extract all features using single platform.

* 1. **PROPOSED SYSTEM**

The Proposed system is a multistage classifier of Diabetic Retinopathy. This system overcomes the drawbacks of the existing system by classifying the disease in to four stages, namely, Normal, Mild, Moderate, Sever and Proliferactive. This multistage classification is important because the disease itself progresses in multiple stages. The reoccurrence of the disease depends on the stage in which the treatment is provide.

The pre-processing part of both the existing and proposed system remain similar, the difference comes in the segmentation and feature extraction stages. Existing system only segmented anomalies like microaneurysms, the problem with this is this anomaly

* 1. **FEASIBILITY STUDY**

A feasibility study is carried out to select the best system that meets performance requirements. The main aim of the feasibility study activity is to determine whether it would be financially and technically feasible to develop the product. The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behavior of the system. The key objective of the feasibility study is to weigh up three types of feasibility. They are:

Operational Feasibility Technical Feasibility Economic Feasibility

**Operational Feasibility:**

This is mainly related to human organizational and political aspects.

The points to be considered are:

What new skills will be required? Do the existing staff members have these skills? If not, can they be trained in due course of time?

What changes will be brought with the system?

The proposed system will provide an automated system for the detection of the disease.

**Technical Feasibility:**

Technical feasibility analysis makes a comparison between the level of technology available and that is needed for the development of the project. The level of technology

consists of the factors like software tools, machine environment, and platform developed and so on. It is also concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may vary considerably.

**Economic Feasibility:**

This is the most important part of the project because the terms and conditions for implementing the project have to be economically feasible. The risk of finance does not exist as the existing hardware is sufficient and the software is free of cost. The system is economically feasible.

**Hardware Interface:**

Describes the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, communication protocols to be use

**Software Interface:**

Describes the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications.

**Objective of Software Project Planning**

The objective of software project planning is to provide a frame work that enables the reasonable estimation of resources, cost and schedule.

**Software Scope**

The first activity in software project planning is the determination of software scope. A software project scope must be unambiguous and understandable at the

management and technical levels. The software scope means the actual operation that is going to be carried out by the software and its plus points and limitations.

**Resources**

The second task of software planning is the estimation of resources required. Each resource is specified with the following characteristic.

**3.4 SOFTWARE REQUIREMENTS**

Tools : Anaconda IDE

Operating System : Windows 7 and above

**3.5 HARDWARE REQUIREMENTS**

Processor : Intel Core i5

Speed : 2.4GHz

RAM : 4 GB

Hard Disk : 20 GB

**4.SYSTEM DESIGN**

**4.SYSTEM DESIGN**

* 1. **DATA DICTIONARY**

This is normally represented as the data about data. It is also termed as metadata some times which gives the data about the data stored in the database. It defines each data term encountered during the analysis and design of a new system. Data elements can describe files or the processes. Following are some rules, which defines the construction of data dictionary entries:

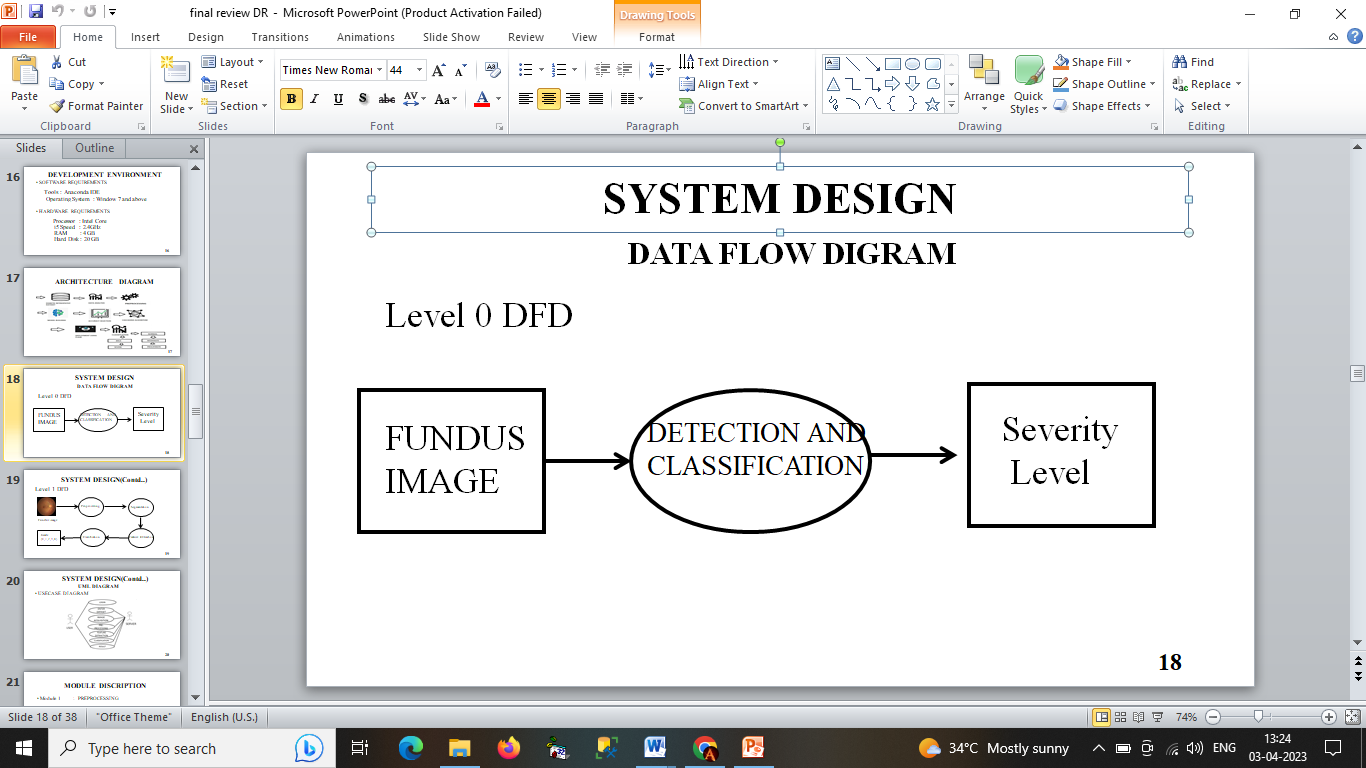
* + 1. Words should be defined to understand for what they need and not the variable need by which they may be described in the program.
    2. Each word must be unique. We cannot have two definition of the same client.
    3. Aliases or synonyms are allowed when two or more enters shows the same meaning. For example, a vendor number may also be called as customer number.
    4. A self-defining word should not be decomposed. It means that the reduction of any information in to subpart should be done only if it is really required that is it is not easy to understand directly.

Data dictionary includes information such as the number of records in file, the frequency a process will run, security factor like pass word which user must enter to get excess to the information.

* 1. **DATAFLOW DIAGRAMS**

A Data Flow Diagram (DFD) is a graphical representation of the ―flow‖ of data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of the data processing. The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of the data between those parts.

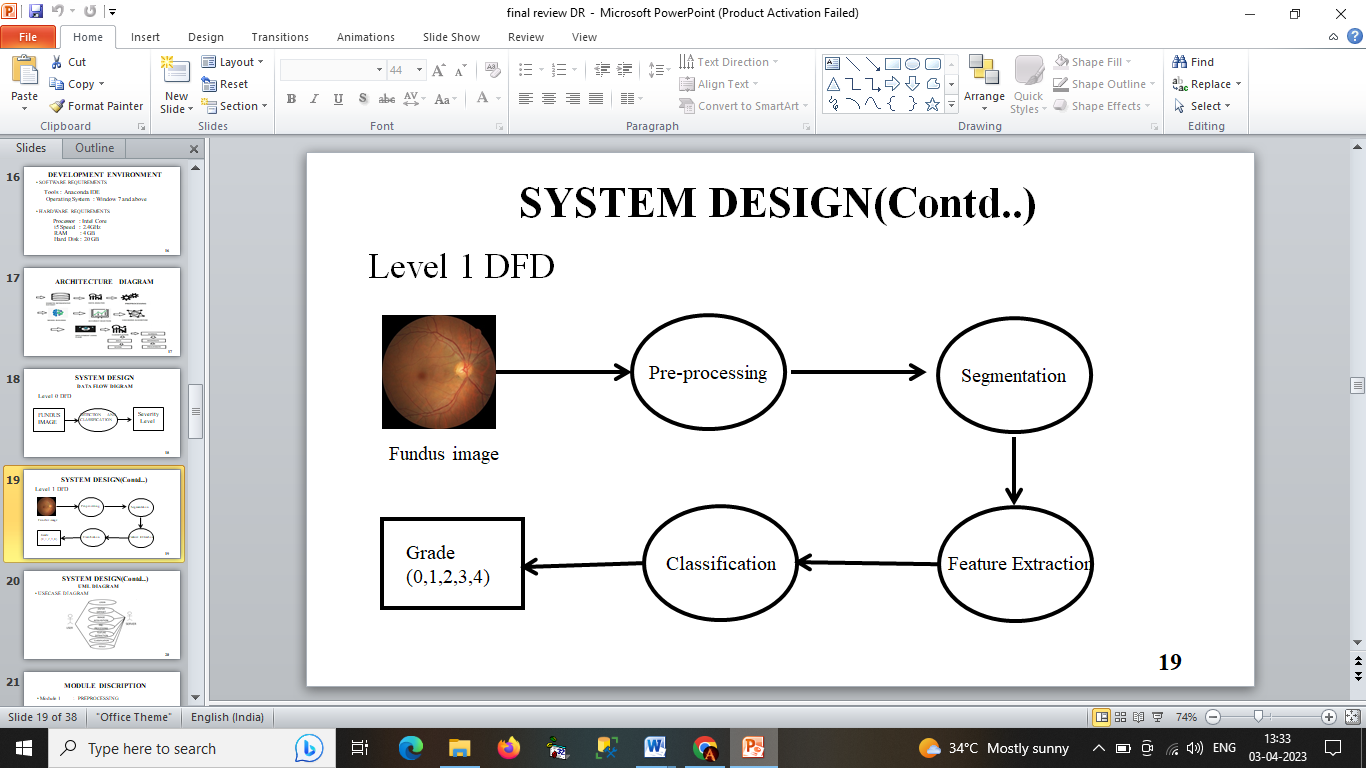
**LEVEL 0:**

****

**FIG NO. 4.1 DFD LEVEL 0**

The Level 0 DFD shows how the system is divided into 'sub-systems' (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole.  It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.

**LEVEL 1:**

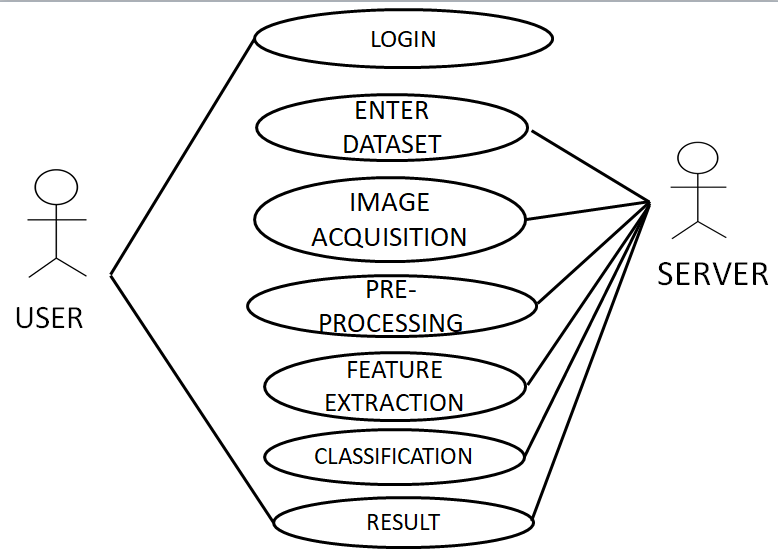
****

**FIG NO. 4.2 DFD LEVEL 1**

Figure 4.2 shows the level 1 DFD. The fundus image of the eye is given as the input, under goes the series of process those are Pre-processing, Segmentation, Feature extraction ,Classification and finally we get the classified output . the single process node from the context diagram is broken down into sub-processes. As these processes are added, the diagram will need additional data flows and data stores to link them together.

* 1. **UML DIAGRAMS**

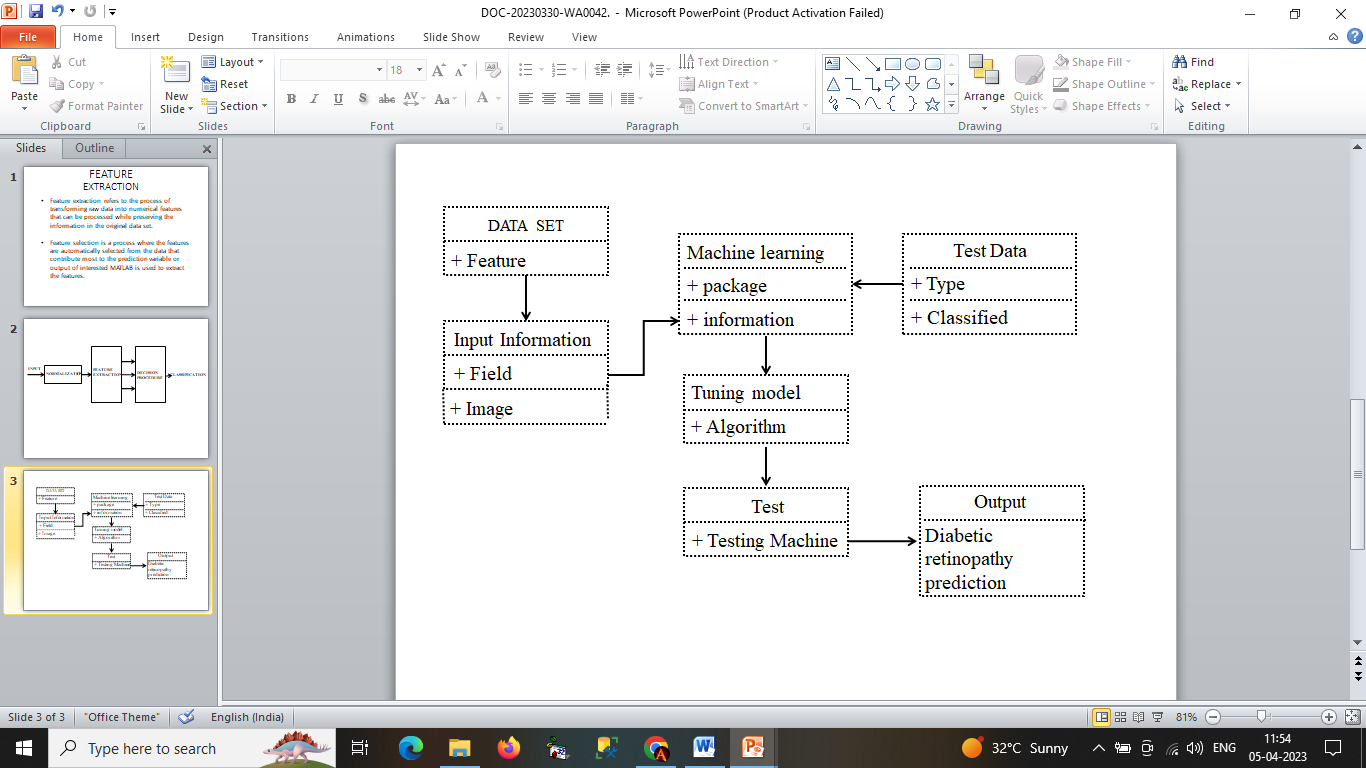
**4.3.1 USE-CASE DIAGRAM:**

****

**FIG NO. 4.3 USE-CASE DIAGRAM**

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

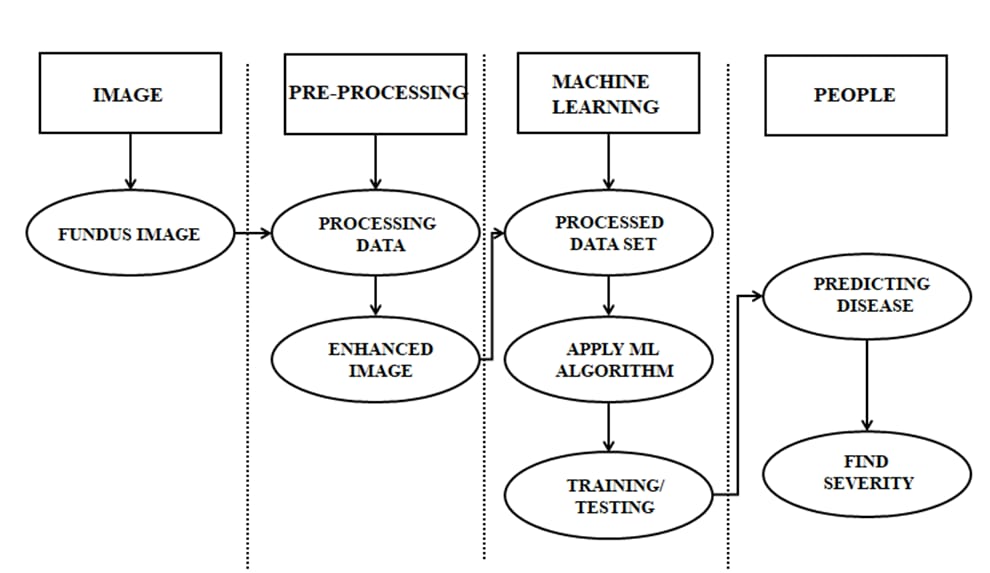
* + 1. **CLASS DIAGRAM**

****

**FIG NO. 4.4 CLASS DIAGRAM**

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling.

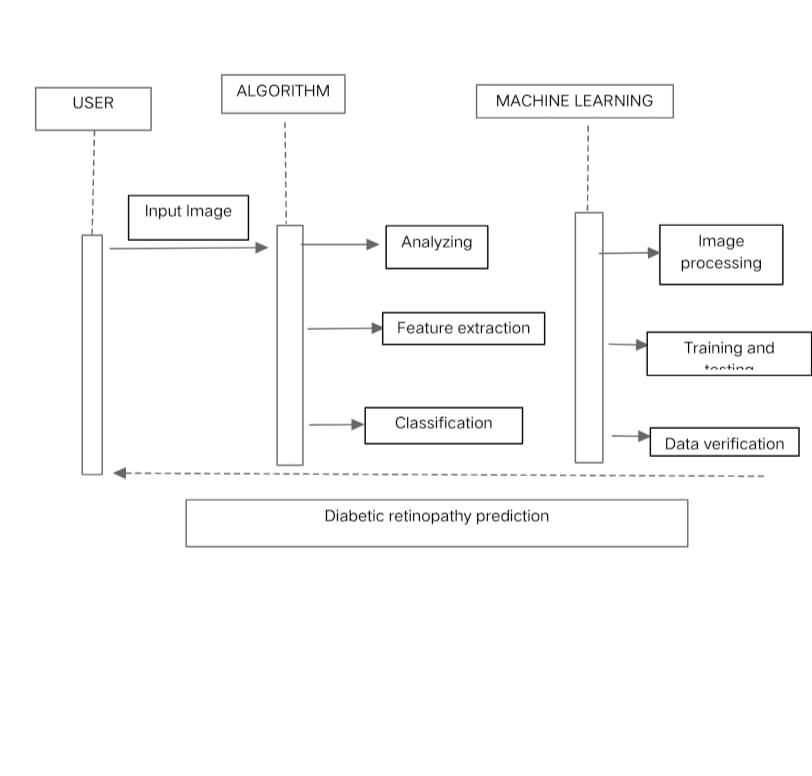
**4.3.3. ACTIVITY DIAGRAM**

****

**FIG NO. 4.5 ACTIVITY DIAGRAM**

Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

**4.3.4.SEQUENCE DIAGRAM**

****

**FIG NO. 4.6 SEQUENCE DIAGRAM**

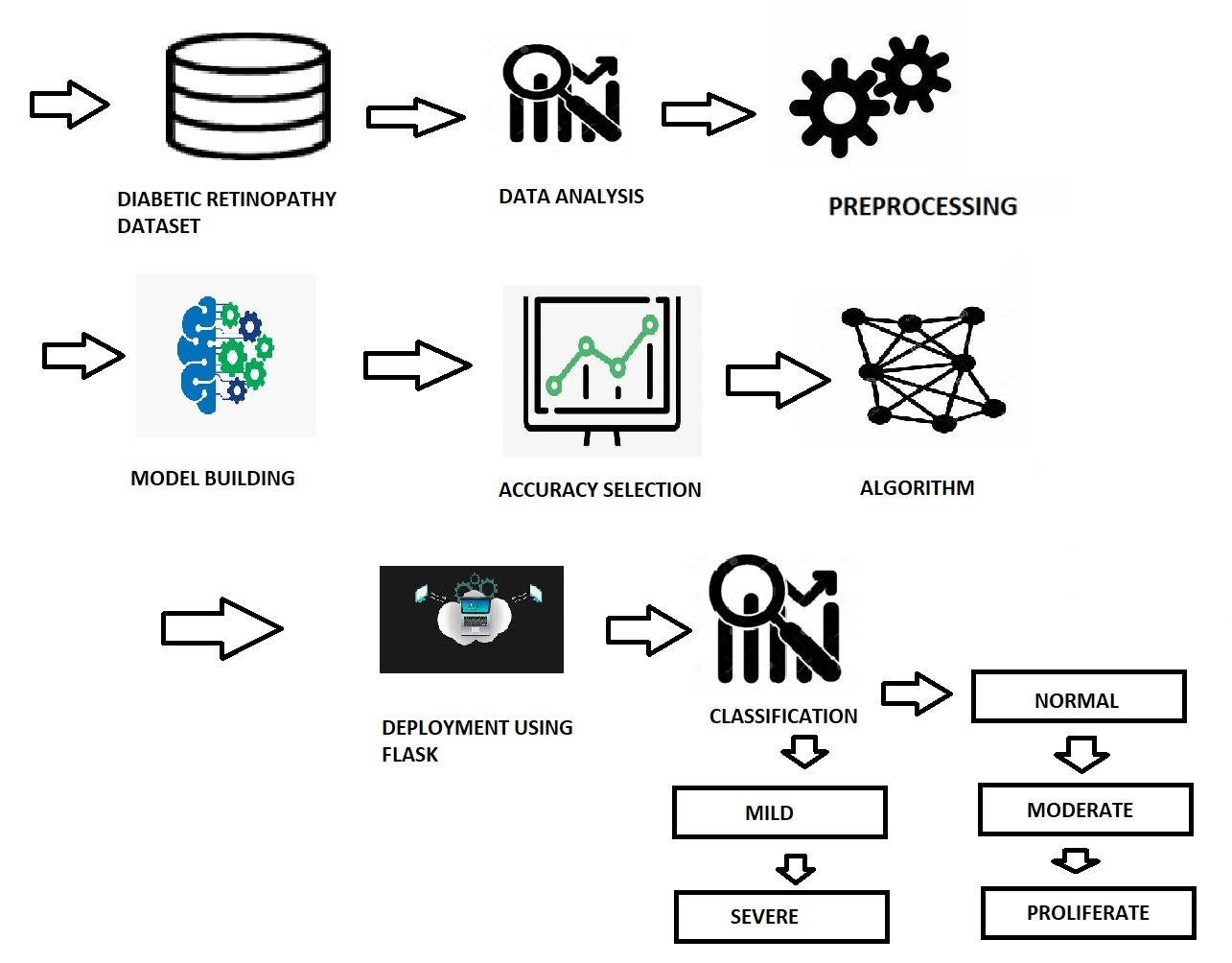
A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

**5.SYSTEM ARCHITECTURE**

**5. SYSTEM ARCHITECTURE**

**5.1MODULE DESIGN**

**SYSTEM ARCHITECTURE**

****

**FIG NO. 5.1 SYSTEM ARCHITECTURE DIAGRAM**

Countering the spam comment phenomenon has become one of the most important challenges for YouTube. This demands a holistic approach to analyzing heterogeneous data and storing the results. The major contribution of this paper is the proposition of an innovative distributed architecture to tackle the above- mentioned problems.

**MODULE DESCRIPTION**

There are four modules in this methodology. They are:

Preprocessing

Segmentation

Feature Extraction

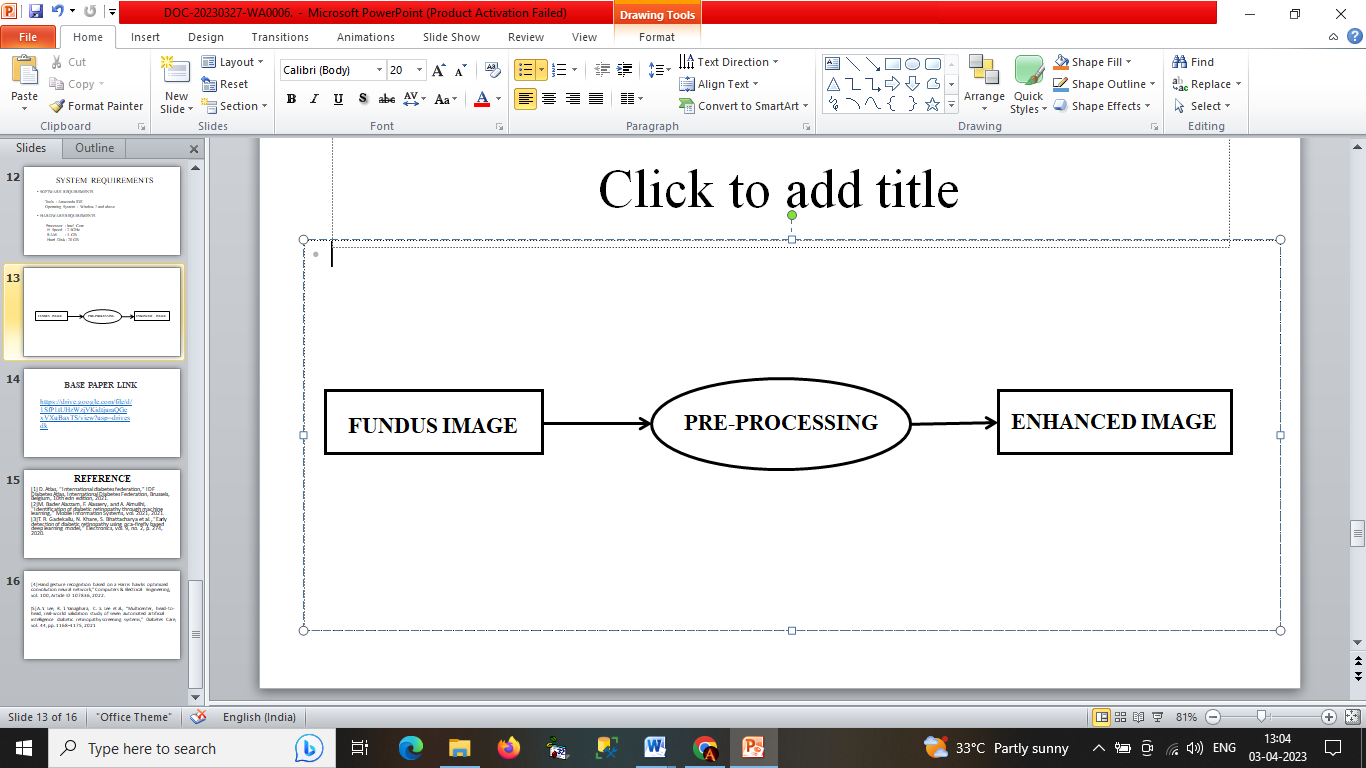
Classifications

1. **PREPROCESSING**

Preprocessing is a process of preparing the raw data and making it

suitable for a machine learning model Patient movement, poor focus, bad positioning, reflections, inadequate illumination can cause a significant proportion of images to be of such poor quality as to interfere with analysis. In approximately 10% of the retinal images, artifacts are significant enough to impede human grading. Preprocessing of such images can ensure adequate level of success in the automated abnormality detection. In the retinal images there can be variations caused by the factors including differences in cameras, illumination, acquisition angle and retinal pigmentation. First step in the preprocessing is to attenuate such image variations by normalizing the color of the original retinal image against a reference image. Few of the retinal images acquired using standard clinical protocols often exhibit low contrast. Also, retinal images typically havea higher contrast in the center of the image with reduced contrast moving

outward from the center. For such images, a local contrast enhancement method is applied as a second preprocessing step.

**

*.*

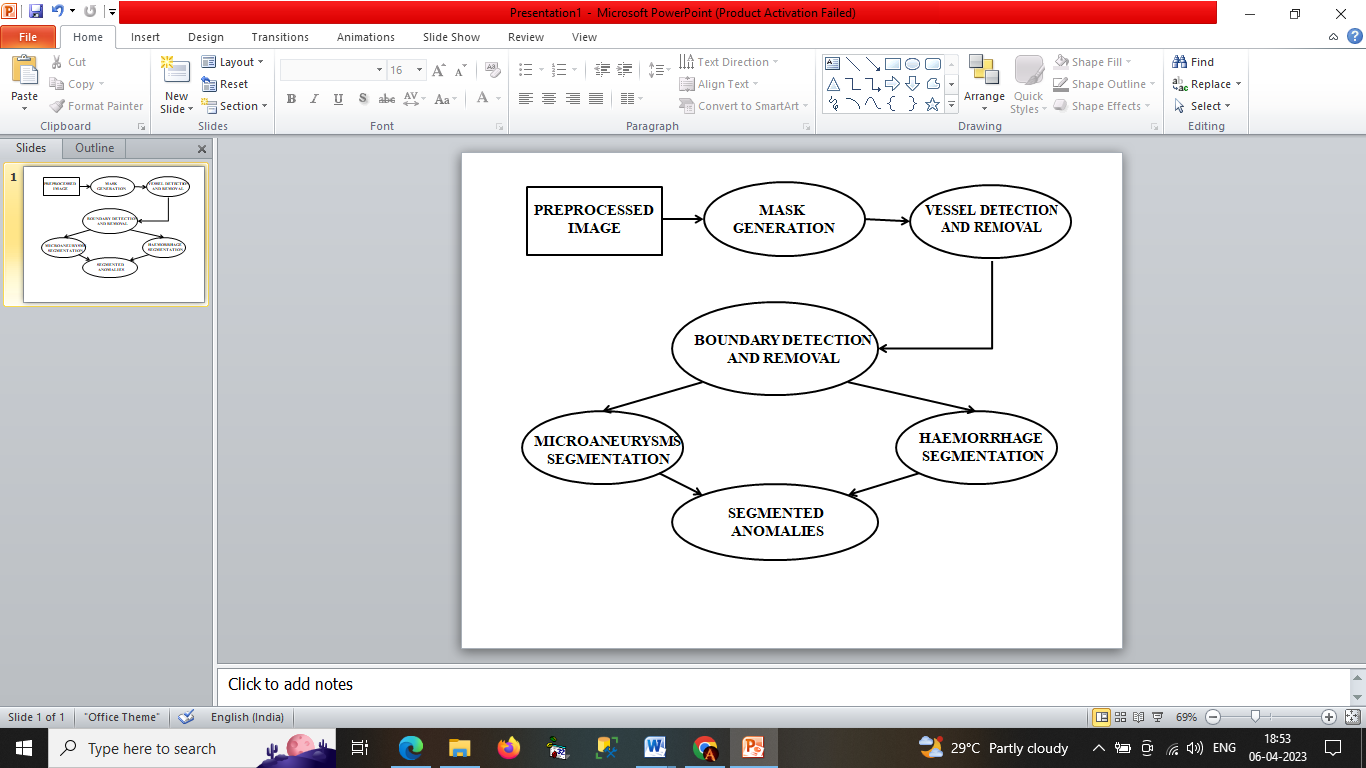
**FIGURE 5.2 PREPROCESSING**

Preprocessing is a process of preparing the raw data and making itsuitable for a machine learning model Preprocessing of such images can ensure adequate level of success in the automated abnormality detection. In the retinal images there can be variations caused by the factors including differences in cameras, illumination, acquisition angle and retinal pigmentation.

1. **SEGMENTATION**

Involves the partitioning of an image or volume into Involves the partitioning of an image or volume into distinct (usually) non-overlapping regions in a meaningful way.Segmentation-

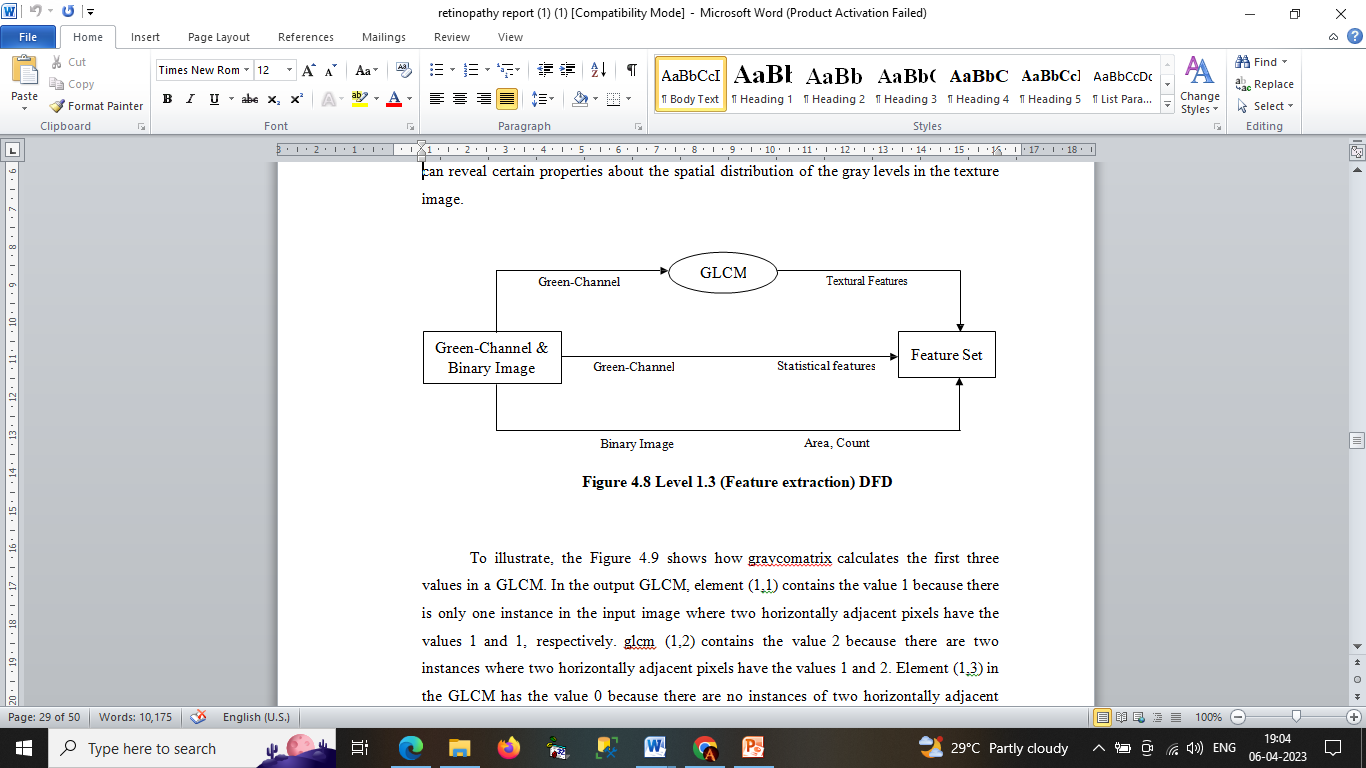
* Identifies separate objects within an image Identifies separate objects within an image.
* Finds regions of connected pixels with similar properties.
* Finds boundaries between regions.
* Removes unwanted regions.



**FIGURE 5.3 SEGMENTATION**

1. **FEATURE EXTRACTION**

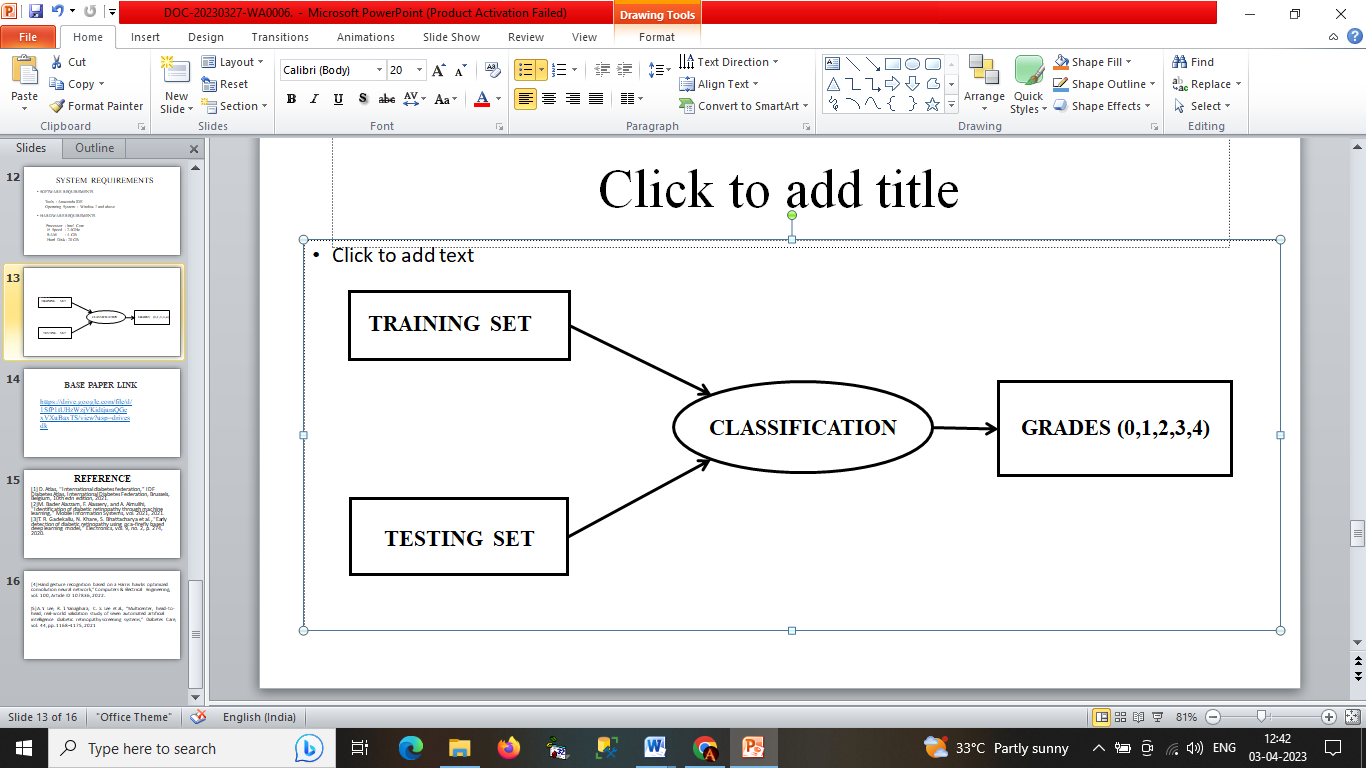
Feature extraction a type of dimensionality reduction that efficiently represents interesting parts of an image as a compact feature vector. This approach is useful when image sizes are large and a reduced feature representation is required to quickly complete tasks such as image matching, classification and retrieval. Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set

****

**FIG NO. 5.4 FEATURE EXTRACTION**

1. **CLASSIFICATIONS**

Image classification analyzes the numerical properties of various image features and organizes data into categories Classification algorithms typically employ two phases of processing: training and testing. In the initial training phase, characteristic properties of typical image features are isolated and, based on these, a unique description of each classification category, i.e. training class, is created. In subsequent testing phase, these feature-space partitions are used to classify image feature



**FIG NO. 5.5 CLASSIFICATION**

* 1. **ALGORITHMS**

There are three algorithms used in this method. They are:

* + 1. Proposed Algorithm
    2. Otsu Thresholding Algorithm
    3. Random Forest Algorithm

1. **PROPOSED ALGORITHM**

**Input:** Fundus image

**Output:** Grade of severity (0, 1, 2, 3, 4)

Step 1: Input fundus image is retrieved from the test set

Step 2: Green channel of the image is extracted

Step 3: The image is then passed through median filter

Step 4: CLAHE is applied to the output of previous step

Step 5: The image is then resized to a standard size of 576\*720

Step 6: Morphological operations are applied to extract microaneurysms and Haemorrhages

Step 7: Area and count of these anomalies are extracted as features. Statistical and GLCM features are also extracted

Step 8: The feature set extracted is then provided to the Random forest classifier for classification of severity levels.

1. **OTSU THRESHOLDING ALGORITHM**

Otsu thresholding divides image into Foreground and Background Pixels, thus assigning Pixels nearer to the black level as 0 and white level as 1, converting image to binary. The thresholding identifies minimum variance between these pixels to aptly identify them.

Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

Otsu thresholding algorithm has the following steps:

**Input:** Pre-processed image

**Output:** Binary image

Step 1: Read a gray scale image.

Step 2: Calculate image histogram.

Step 3: Select a threshold and referred as t,

* 1. Calculate foreground variance.
  2. Calculate background variance.

Step 4: Calculate Within-Class variance.

Step 5: Repeat steps 3 and 4 for all possible threshold value.

Step 6: Final global threshold, T = threshold in MIN (Within-class variance)

Step 7: Binarize Image = gray scale image > T

1. **RANDOM FOREST ALGORITHM**

Random Forests algorithm is one of the best among classification algorithms - able to classify large amounts of data with accuracy. Random Forests are an ensemble learning method (also thought of as a form of nearest neighbor predictor) for classification and regression that construct a number of decision trees at training time and outputting the class that is the mode of the classes output by individual trees.

Random Forests are a combination of tree predictors where each tree depends on the values of a random vector sampled independently with the same distribution for all trees in the forest. The basic principle is that a group of “weak learners” can come together to form a “strong learner”. Random Forests are a wonderful tool for making predictions considering they do not over fit because of the law of large numbers. Introducing the right kind of randomness makes them accurate classifiers and regressors. Single decision trees often have high variance or high bias. Random Forests attempts to mitigate the problems of high variance and high bias by averaging to find a natural balance between the two extremes. Considering that Random Forests have few parameters to tune and can be used simply with default parameter settings, they are a simple tool to use without having a model or to produce a reasonable model fast and efficiently.

Random Forests grows many classification trees. Each tree is grown as follows:

**Input:** Feature set

**Output:** Grade of severity

Step 1: If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.

Step 2: If there are M input variables, a number mM is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held.

Step 3: Constant during the forest growing.

Step 4: Each tree is grown to the largest extent possible. There is no pruning.

* 1. SYSTEM IMPLEMENTATION
  2. **CLIENT SIDE CODING**

from flask import Flask, render\_template, request, url\_for, Markup, jsonify import pickle

import pandas as pd import numpy as np

from sklearn.preprocessing import MinMaxScaler from werkzeug.utils import secure\_filename import pickle

from flask import \* import os

from werkzeug.utils import secure\_filename import label\_image

def load\_image(image):

text = label\_image.main(image) return text

app = Flask( name ) #Initialize the flask App

@app.route('/')

@app.route('/first') def first():

return render\_template('first.html')

@app.route('/login') def login():

return render\_template('login.html') @app.route('/chart')

def chart():

return render\_template('chart.html')

@app.route('/index') def index():

return render\_template('index.html')

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

if request.method == 'POST':

# Get the file from post request f = request.files['file']

file\_path = secure\_filename(f.filename) f.save(file\_path)

# Make prediction

result = load\_image(file\_path) result = result.title()

d = {"Normal":"✓(NO RISK)",'Mild':"❌(15% RISK)","Moderate":"❌(45%

RISK)","Severe":"❌(75% RISK)","Proliferative":"❌(95% RISK)"} result = result+d[result]

#result = [result] print(result) os.remove(file\_path) return result

return None

if name == ' main ': app.run()

* 1. **SERVER SIDE CODING**

from future import absolute\_import from future import division

from future import print\_function import argparse

import collections

from datetime import datetime import hashlib

import os.path import random import re import sys import tarfile

import numpy as np

from six.moves import urllib import tensorflow.compat.v1 as tf

from tensorflow.python.framework import graph\_util from tensorflow.python.framework import tensor\_shape from tensorflow.python.platform import gfile

from tensorflow.python.util import compat if not gfile.Exists(image\_dir):

tf.logging.error("Image directory '" + image\_dir + "' not found.") return None

result = collections.OrderedDict()

sub\_dirs = [ os.path.join(image\_dir,item)

for item in gfile.ListDirectory(image\_dir)] sub\_dirs = sorted(item for item in sub\_dirs

if gfile.IsDirectory(item)) for sub\_dir in sub\_dirs:

extensions = ['jpg', 'jpeg', 'JPG', 'JPEG'] file\_list = []

dir\_name = os.path.basename(sub\_dir) if dir\_name == image\_dir:

continue

tf.logging.info("Looking for images in '" + dir\_name + "'") for extension in extensions:

file\_glob = os.path.join(image\_dir, dir\_name, '\*.' + extension) file\_list.extend(gfile.Glob(file\_glob))

if not file\_list:

tf.logging.warning('No files found') continue

if len(file\_list) < 20:

tf.logging.warning(

'WARNING: Folder has less than 20 images, which may cause issues.') elif len(file\_list) > MAX\_NUM\_IMAGES\_PER\_CLASS: tf.logging.warning(

'WARNING: Folder {} has more than {} images. Some images will '

'never be selected.'.format(dir\_name, MAX\_NUM\_IMAGES\_PER\_CLASS)) label\_name = re.sub(r'[^a-z0-9]+', ' ', dir\_name.lower())

training\_images = []

testing\_images = [] validation\_images = [] for file\_name in file\_list:

base\_name = os.path.basename(file\_name) hash\_name = re.sub(r'\_nohash\_.\*$', '', file\_name)

hash\_name\_hashed = hashlib.sha1(compat.as\_bytes(hash\_name)).hexdigest() percentage\_hash = ((int(hash\_name\_hashed, 16) %

(MAX\_NUM\_IMAGES\_PER\_CLASS + 1)) \* (100.0 / MAX\_NUM\_IMAGES\_PER\_CLASS))

if percentage\_hash < validation\_percentage: validation\_images.append(base\_name)

elif percentage\_hash < (testing\_percentage + validation\_percentage): testing\_images.append(base\_name)

else: training\_images.append(base\_name)

result[label\_name] = { 'dir': dir\_name,

'training': training\_images, 'testing': testing\_images, 'validation': validation\_images,

}

return result

def get\_image\_path(image\_lists, label\_name, index, image\_dir, category): if label\_name not in image\_lists:

tf.logging.fatal('Label does not exist %s.', label\_name) label\_lists = image\_lists[label\_name]

if category not in label\_lists:

tf.logging.fatal('Category does not exist %s.', category) category\_list = label\_lists[category]

if not category\_list:

tf.logging.fatal('Label %s has no images in the category %s.', label\_name, category)

mod\_index = index % len(category\_list) base\_name = category\_list[mod\_index] sub\_dir = label\_lists['dir']

full\_path = os.path.join(image\_dir, sub\_dir, base\_name) return full\_path

def get\_bottleneck\_path(image\_lists, label\_name, index, bottleneck\_dir, category, architecture):

return get\_image\_path(image\_lists, label\_name, index, bottleneck\_dir, category) + '\_' + architecture + '.txt'

def create\_model\_graph(model\_info):

with tf.Graph().as\_default() as graph:

model\_path = os.path.join(FLAGS.model\_dir, model\_info['model\_file\_name']) with gfile.FastGFile(model\_path, 'rb') as f:

graph\_def = tf.GraphDef() graph\_def.ParseFromString(f.read())

bottleneck\_tensor, resized\_input\_tensor = (tf.import\_graph\_def( graph\_def,

name='', return\_elements=[

model\_info['bottleneck\_tensor\_name'], model\_info['resized\_input\_tensor\_name'],

]))

return graph, bottleneck\_tensor, resized\_input\_tensor

def run\_bottleneck\_on\_image(sess, image\_data, image\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor):

resized\_input\_values = sess.run(decoded\_image\_tensor,

{image\_data\_tensor: image\_data}) bottleneck\_values = sess.run(bottleneck\_tensor,

{resized\_input\_tensor: resized\_input\_values}) bottleneck\_values = np.squeeze(bottleneck\_values)

return bottleneck\_values

def maybe\_download\_and\_extract(data\_url): dest\_directory = FLAGS.model\_dir

if not os.path.exists(dest\_directory): os.makedirs(dest\_directory) filename = data\_url.split('/')[-1]

filepath = os.path.join(dest\_directory, filename) if not os.path.exists(filepath):

def \_progress(count, block\_size, total\_size): sys.stdout.write('\r>> Downloading %s %.1f%%' %

(filename,

float(count \* block\_size) / float(total\_size) \* 100.0)) sys.stdout.flush()

filepath, \_ = urllib.request.urlretrieve(data\_url, filepath, \_progress) print()

statinfo = os.stat(filepath)

tf.logging.info('Successfully downloaded', filename, statinfo.st\_size, 'bytes.')

tarfile.open(filepath, 'r:gz').extractall(dest\_directory) def ensure\_dir\_exists(dir\_name):

if not os.path.exists(dir\_name): os.makedirs(dir\_name)

bottleneck\_path\_2\_bottleneck\_values = {}

def create\_bottleneck\_file(bottleneck\_path, image\_lists, label\_name, index, image\_dir, category, sess, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor):

tf.logging.info('Creating bottleneck at ' + bottleneck\_path) image\_path = get\_image\_path(image\_lists, label\_name, index,

image\_dir, category) if not gfile.Exists(image\_path):

tf.logging.fatal('File does not exist %s', image\_path) image\_data = gfile.FastGFile(image\_path, 'rb').read() try:

bottleneck\_values = run\_bottleneck\_on\_image(

sess, image\_data, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor)

except Exception as e:

raise RuntimeError('Error during processing file %s (%s)' % (image\_path,

str(e))) bottleneck\_string = ','.join(str(x) for x in bottleneck\_values) with open(bottleneck\_path, 'w') as bottleneck\_file: bottleneck\_file.write(bottleneck\_string)

def get\_or\_create\_bottleneck(sess, image\_lists, label\_name, index, image\_dir, category, bottleneck\_dir, jpeg\_data\_tensor,

decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor, architecture):

label\_lists = image\_lists[label\_name] sub\_dir = label\_lists['dir']

sub\_dir\_path = os.path.join(bottleneck\_dir, sub\_dir) ensure\_dir\_exists(sub\_dir\_path)

bottleneck\_path = get\_bottleneck\_path(image\_lists, label\_name, index,

bottleneck\_dir, category, architecture) if not os.path.exists(bottleneck\_path):

create\_bottleneck\_file(bottleneck\_path, image\_lists, label\_name, index, image\_dir, category, sess, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor)

with open(bottleneck\_path, 'r') as bottleneck\_file: bottleneck\_string = bottleneck\_file.read() did\_hit\_error = False

try:

bottleneck\_values = [float(x) for x in bottleneck\_string.split(',')] except ValueError:

tf.logging.warning('Invalid float found, recreating bottleneck') did\_hit\_error = True

if did\_hit\_error:

create\_bottleneck\_file(bottleneck\_path, image\_lists, label\_name, index, image\_dir, category, sess, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor)

with open(bottleneck\_path, 'r') as bottleneck\_file:

bottleneck\_string = bottleneck\_file.read()

bottleneck\_values = [float(x) for x in bottleneck\_string.split(',')] return bottleneck\_values

def cache\_bottlenecks(sess, image\_lists, image\_dir, bottleneck\_dir, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor, architecture):

how\_many\_bottlenecks = 0 ensure\_dir\_exists(bottleneck\_dir)

for label\_name, label\_lists in image\_lists.items():

for category in ['training', 'testing', 'validation']:

category\_list = label\_lists[category]

for index, unused\_base\_name in enumerate(category\_list): get\_or\_create\_bottleneck(

sess, image\_lists, label\_name, index, image\_dir, category, bottleneck\_dir, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor, architecture)

how\_many\_bottlenecks += 1

if how\_many\_bottlenecks % 100 == 0:

tf.logging.info(

str(how\_many\_bottlenecks) + ' bottleneck files created.')

def get\_random\_cached\_bottlenecks(sess, image\_lists, how\_many, category, bottleneck\_dir, image\_dir, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor, architecture):

class\_count = len(image\_lists.keys()) bottlenecks = []

ground\_truths = [] filenames = []

if how\_many >= 0:

for unused\_i in range(how\_many):

label\_index = random.randrange(class\_count) label\_name = list(image\_lists.keys())[label\_index]

image\_index = random.randrange(MAX\_NUM\_IMAGES\_PER\_CLASS + 1) image\_name = get\_image\_path(image\_lists, label\_name, image\_index,

image\_dir, category) bottleneck = get\_or\_create\_bottleneck(

sess, image\_lists, label\_name, image\_index, image\_dir, category, bottleneck\_dir, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor, architecture)

ground\_truth = np.zeros(class\_count, dtype=np.float32) ground\_truth[label\_index] = 1.0 bottlenecks.append(bottleneck) ground\_truths.append(ground\_truth) filenames.append(image\_name)

else:

for label\_index, label\_name in enumerate(image\_lists.keys()): for image\_index, image\_name in enumerate(

image\_lists[label\_name][category]):

image\_name = get\_image\_path(image\_lists, label\_name, image\_index, image\_dir, category)

bottleneck = get\_or\_create\_bottleneck(

sess, image\_lists, label\_name, image\_index, image\_dir, category, bottleneck\_dir, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_input\_tensor, bottleneck\_tensor, architecture)

ground\_truth = np.zeros(class\_count, dtype=np.float32) ground\_truth[label\_index] = 1.0 bottlenecks.append(bottleneck) ground\_truths.append(ground\_truth) filenames.append(image\_name)

return bottlenecks, ground\_truths, filenames def get\_random\_distorted\_bottlenecks(

sess, image\_lists, how\_many, category, image\_dir, input\_jpeg\_tensor, distorted\_image, resized\_input\_tensor, bottleneck\_tensor):

class\_count = len(image\_lists.keys()) bottlenecks = []

ground\_truths = []

for unused\_i in range(how\_many):

label\_index = random.randrange(class\_count) label\_name = list(image\_lists.keys())[label\_index]

image\_index = random.randrange(MAX\_NUM\_IMAGES\_PER\_CLASS + 1) image\_path = get\_image\_path(image\_lists, label\_name, image\_index, image\_dir,

category)

if not gfile.Exists(image\_path):

tf.logging.fatal('File does not exist %s', image\_path) jpeg\_data = gfile.FastGFile(image\_path, 'rb').read() distorted\_image\_data = sess.run(distorted\_image,

{input\_jpeg\_tensor: jpeg\_data})

bottleneck\_values = sess.run(bottleneck\_tensor,

{resized\_input\_tensor: distorted\_image\_data}) bottleneck\_values = np.squeeze(bottleneck\_values) ground\_truth = np.zeros(class\_count, dtype=np.float32) ground\_truth[label\_index] = 1.0 bottlenecks.append(bottleneck\_values) ground\_truths.append(ground\_truth)

return bottlenecks, ground\_truths

def should\_distort\_images(flip\_left\_right, random\_crop, random\_scale, random\_brightness):

return (flip\_left\_right or (random\_crop != 0) or (random\_scale != 0) or (random\_brightness != 0))

def add\_input\_distortions(flip\_left\_right, random\_crop, random\_scale, random\_brightness, input\_width, input\_height, input\_depth, input\_mean, input\_std):

jpeg\_data = tf.placeholder(tf.string, name='DistortJPGInput') decoded\_image = tf.image.decode\_jpeg(jpeg\_data, channels=input\_depth) decoded\_image\_as\_float = tf.cast(decoded\_image, dtype=tf.float32) decoded\_image\_4d = tf.expand\_dims(decoded\_image\_as\_float, 0) margin\_scale = 1.0 + (random\_crop / 100.0)

resize\_scale = 1.0 + (random\_scale / 100.0) margin\_scale\_value = tf.constant(margin\_scale) resize\_scale\_value = tf.random\_uniform(tensor\_shape.scalar(),

minval=1.0, maxval=resize\_scale)

scale\_value = tf.multiply(margin\_scale\_value, resize\_scale\_value) precrop\_width = tf.multiply(scale\_value, input\_width)

precrop\_height = tf.multiply(scale\_value, input\_height) precrop\_shape = tf.stack([precrop\_height, precrop\_width]) precrop\_shape\_as\_int = tf.cast(precrop\_shape, dtype=tf.int32) precropped\_image = tf.image.resize\_bilinear(decoded\_image\_4d,

precrop\_shape\_as\_int)

precropped\_image\_3d = tf.squeeze(precropped\_image, squeeze\_dims=[0]) cropped\_image = tf.random\_crop(precropped\_image\_3d,

[input\_height, input\_width, input\_depth])

if flip\_left\_right:

flipped\_image = tf.image.random\_flip\_left\_right(cropped\_image) else:

flipped\_image = cropped\_image

brightness\_min = 1.0 - (random\_brightness / 100.0) brightness\_max = 1.0 + (random\_brightness / 100.0) brightness\_value = tf.random\_uniform(tensor\_shape.scalar(),

minval=brightness\_min, maxval=brightness\_max)

brightened\_image = tf.multiply(flipped\_image, brightness\_value) offset\_image = tf.subtract(brightened\_image, input\_mean) mul\_image = tf.multiply(offset\_image, 1.0 / input\_std)

distort\_result = tf.expand\_dims(mul\_image, 0, name='DistortResult') return jpeg\_data, distort\_result

def variable\_summaries(var):

"""Attach a lot of summaries to a Tensor (for TensorBoard visualization).""" with tf.name\_scope('summaries'):

mean = tf.reduce\_mean(var) tf.summary.scalar('mean', mean)

with tf.name\_scope('stddev'):

stddev = tf.sqrt(tf.reduce\_mean(tf.square(var - mean))) tf.summary.scalar('stddev', stddev) tf.summary.scalar('max', tf.reduce\_max(var)) tf.summary.scalar('min', tf.reduce\_min(var)) tf.summary.histogram('histogram', var)

def add\_final\_training\_ops(class\_count, final\_tensor\_name, bottleneck\_tensor, bottleneck\_tensor\_size):

with tf.name\_scope('input'):

bottleneck\_input = tf.placeholder\_with\_default( bottleneck\_tensor,

shape=[None, bottleneck\_tensor\_size], name='BottleneckInputPlaceholder')

ground\_truth\_input = tf.placeholder(tf.float32,

[None, class\_count], name='GroundTruthInput') layer\_name = 'final\_training\_ops'

with tf.name\_scope(layer\_name):

with tf.name\_scope('weights'):

initial\_value = tf.truncated\_normal( [bottleneck\_tensor\_ize, class\_count], stddev=0.001)

layer\_weights = tf.Variable(initial\_value, name='final\_weights') variable\_summaries(layer\_weights)

with tf.name\_scope('biases'):

layer\_biases = tf.Variable(tf.zeros([class\_count]), name='final\_biases') variable\_summaries(layer\_biases)

with tf.name\_scope('Wx\_plus\_b'):

logits = tf.matmul(bottleneck\_input, layer\_weights) + layer\_biases

tf.summary.histogram('pre\_activations', logits) final\_tensor = tf.nn.softmax(logits, name=final\_tensor\_name) tf.summary.histogram('activations', final\_tensor)

with tf.name\_scope('cross\_entropy'):

cross\_entropy = tf.nn.softmax\_cross\_entropy\_with\_logits( labels=ground\_truth\_input, logits=logits)

with tf.name\_scope('total'):

cross\_entropy\_mean = tf.reduce\_mean(cross\_entropy) tf.summary.scalar('cross\_entropy', cross\_entropy\_mean) with tf.name\_scope('train'):

optimizer = tf.train.GradientDescentOptimizer(FLAGS.learning\_rate) train\_step = optimizer.minimize(cross\_entropy\_mean)

return (train\_step, cross\_entropy\_mean, bottleneck\_input, ground\_truth\_input, final\_tensor)

def add\_evaluation\_step(result\_tensor, ground\_truth\_tensor): with tf.name\_scope('accuracy'):

with tf.name\_scope('correct\_prediction'):

prediction = tf.argmax(result\_tensor, 1) correct\_prediction = tf.equal(

prediction, tf.argmax(ground\_truth\_tensor, 1)) with tf.name\_scope('accuracy'):

evaluation\_step = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32)) tf.summary.scalar('accuracy', evaluation\_step)

return evaluation\_step, prediction

def save\_graph\_to\_file(sess, graph, graph\_file\_name): output\_graph\_def = graph\_util.convert\_variables\_to\_constants( sess, graph.as\_graph\_def(), [FLAGS.final\_tensor\_name])

with gfile.FastGFile(graph\_file\_name, 'wb') as f: f.write(output\_graph\_def.SerializeToString()) return

def prepare\_file\_system():

if tf.gfile.Exists(FLAGS.summaries\_dir): tf.gfile.DeleteRecursively(FLAGS.summaries\_dir) tf.gfile.MakeDirs(FLAGS.summaries\_dir)

if FLAGS.intermediate\_store\_frequency > 0: ensure\_dir\_exists(FLAGS.intermediate\_output\_graphs\_dir) return

def create\_model\_info(architecture):

architecture = architecture.lower() if architecture == 'inception\_v3'

bottleneck\_tensor\_name = 'pool\_3/\_reshape:0' bottleneck\_tensor\_size = 2048

input\_width = 299

input\_height = 299

input\_depth = 3 resized\_input\_tensor\_name = 'Mul:0'

model\_file\_name = 'classify\_image\_graph\_def.pb' input\_mean = 128

input\_std = 128

elif architecture.startswith('mobilenet\_'):

parts = architecture.split('\_')

if len(parts) != 3 and len(parts) != 4:

tf.logging.error("Couldn't understand architecture name '%s'", architecture) return None

version\_string = parts[1]

if (version\_string != '1.0' and version\_string != '0.75' and version\_string != '0.50' and version\_string != '0.25'): tf.logging.error(

version\_string, architecture) return None

size\_string = parts[2]

if (size\_string != '224' and size\_string != '192' and size\_string != '160' and size\_string != '128'): tf.logging.error(

size\_string, architecture) return None

if len(parts) == 3:

is\_quantized = False else:

if parts[3] != 'quantized':

tf.logging.error(

"Couldn't understand architecture suffix '%s' for '%s'", parts[3], architecture)

return None is\_quantized = True

data\_url = ['http://download.tensorflow.org/models/mobilenet\_v1\_'](http://download.tensorflow.org/models/mobilenet_v1_%27) data\_url += version\_string + '\_' + size\_string + '\_frozen.tgz' bottleneck\_tensor\_name = 'MobilenetV1/Predictions/Reshape:0' bottleneck\_tensor\_size = 1001

input\_width = int(size\_string) input\_height = int(size\_string)

input\_depth = 3 resized\_input\_tensor\_name = 'input:0' if is\_quantized:

model\_base\_name = 'quantized\_graph.pb' else:

model\_base\_name = 'frozen\_graph.pb'

model\_dir\_name = 'mobilenet\_v1\_' + version\_string + '\_' + size\_string model\_file\_name = os.path.join(model\_dir\_name, model\_base\_name) input\_mean = 127.5

input\_std = 127.5 else:

tf.logging.error("Couldn't understand architecture name '%s'", architecture) raise ValueError('Unknown architecture', architecture)

return {

'data\_url': data\_url,

'bottleneck\_tensor\_name': bottleneck\_tensor\_name, 'bottleneck\_tensor\_size': bottleneck\_tensor\_size, 'input\_width': input\_width,

'input\_height': input\_height, 'input\_depth': input\_depth,

'resized\_input\_tensor\_name': resized\_input\_tensor\_name, 'model\_file\_name': model\_file\_name,

'input\_mean': input\_mean, 'input\_std': input\_std,

}

def main(\_): tf.logging.set\_verbosity(tf.logging.INFO)

prepare\_file\_system()

model\_info = create\_model\_info(FLAGS.architecture) if not model\_info:

tf.logging.error('Did not recognize architecture flag') return -1 maybe\_download\_and\_extract(model\_info['data\_url']) graph, bottleneck\_tensor, resized\_image\_tensor = (

create\_model\_graph(model\_info))

image\_lists = create\_image\_lists(FLAGS.image\_dir, FLAGS.testing\_percentage, FLAGS.validation\_percentage)

class\_count = len(image\_lists.keys()) if class\_count == 0:

tf.logging.error('No valid folders of images found at ' + FLAGS.image\_dir) return -1

if class\_count == 1:

tf.logging.error('Only one valid folder of images found at ' + FLAGS.image\_dir +

' - multiple classes are needed for classification.')

return -1

do\_distort\_images = should\_distort\_images(

FLAGS.flip\_left\_right, FLAGS.random\_crop, FLAGS.random\_scale, FLAGS.random\_brightness)

with tf.Session(graph=graph) as sess:

jpeg\_data\_tensor, decoded\_image\_tensor = add\_jpeg\_decoding( model\_info['input\_width'], model\_info['input\_height'], model\_info['input\_depth'], model\_info['input\_mean'], model\_info['input\_std'])

if do\_distort\_images: (distorted\_jpeg\_data\_tensor,

distorted\_image\_tensor) = add\_input\_distortions( FLAGS.flip\_left\_right, FLAGS.random\_crop, FLAGS.random\_scale, FLAGS.random\_brightness, model\_info['input\_width'], model\_info['input\_height'], model\_info['input\_depth'], model\_info['input\_mean'], model\_info['input\_std'])

else:

cache\_bottlenecks(sess, image\_lists, FLAGS.image\_dir, FLAGS.bottleneck\_dir, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_image\_tensor, bottleneck\_tensor, FLAGS.architecture)

evaluation\_step, prediction = add\_evaluation\_step( final\_tensor, ground\_truth\_input)

merged = tf.summary.merge\_all()

train\_writer = tf.summary.FileWriter(FLAGS.summaries\_dir + '/train',

sess.graph) validation\_writer = tf.summary.FileWriter(

FLAGS.summaries\_dir + '/validation') init = tf.global\_variables\_initializer() sess.run(init)

for i in range(FLAGS.how\_many\_training\_steps): if do\_distort\_images:

(train\_bottlenecks,

train\_ground\_truth) = get\_random\_distorted\_bottlenecks( sess, image\_lists, FLAGS.train\_batch\_size, 'training', FLAGS.image\_dir, distorted\_jpeg\_data\_tensor,

distorted\_image\_tensor, resized\_image\_tensor, bottleneck\_tensor) else:

(train\_bottlenecks,

train\_ground\_truth, \_) = get\_random\_cached\_bottlenecks( sess, image\_lists, FLAGS.train\_batch\_size, 'training',

FLAGS.bottleneck\_dir, FLAGS.image\_dir, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_image\_tensor, bottleneck\_tensor, FLAGS.architecture)

train\_summary, \_ = sess.run( [merged, train\_step],

feed\_dict={bottleneck\_input: train\_bottlenecks, ground\_truth\_input: train\_ground\_truth})

train\_writer.add\_summary(train\_summary, i)

is\_last\_step = (i + 1 == FLAGS.how\_many\_training\_steps) if (i % FLAGS.eval\_step\_interval) == 0 or is\_last\_step: train\_accuracy, cross\_entropy\_value = sess.run(

[evaluation\_step, cross\_entropy], feed\_dict={bottleneck\_input: train\_bottlenecks,

ground\_truth\_input: train\_ground\_truth}) tf.logging.info('%s: Step %d: Train accuracy = %.1f%%' %

(datetime.now(), i, train\_accuracy \* 100)) tf.logging.info('%s: Step %d: Cross entropy = %f' %

(datetime.now(), i, cross\_entropy\_value)) validation\_bottlenecks, validation\_ground\_truth, \_ = (

get\_random\_cached\_bottlenecks(

sess, image\_lists, FLAGS.validation\_batch\_size, 'validation', FLAGS.bottleneck\_dir, FLAGS.image\_dir, jpeg\_data\_tensor,

decoded\_image\_tensor, resized\_image\_tensor, bottleneck\_tensor, FLAGS.architecture))

validation\_summary, validation\_accuracy = sess.run( [merged, evaluation\_step], feed\_dict={bottleneck\_input: validation\_bottlenecks,

ground\_truth\_input: validation\_ground\_truth}) validation\_writer.add\_summary(validation\_summary, i) tf.logging.info('%s: Step %d: Validation accuracy = %.1f%% (N=%d)' %

(datetime.now(), i, validation\_accuracy \* 100, len(validation\_bottlenecks)))

intermediate\_frequency = FLAGS.intermediate\_store\_frequency

if (intermediate\_frequency > 0 and (i % intermediate\_frequency == 0) and i > 0):

intermediate\_file\_name = (FLAGS.intermediate\_output\_graphs\_dir + 'intermediate\_' + str(i) + '.pb')

tf.logging.info('Save intermediate result to : ' + intermediate\_file\_name)

save\_graph\_to\_file(sess, graph, intermediate\_file\_name) test\_bottlenecks, test\_ground\_truth, test\_filenames = (

get\_random\_cached\_bottlenecks(

sess, image\_lists, FLAGS.test\_batch\_size, 'testing', FLAGS.bottleneck\_dir, FLAGS.image\_dir, jpeg\_data\_tensor, decoded\_image\_tensor, resized\_image\_tensor, bottleneck\_tensor, FLAGS.architecture))

test\_accuracy, predictions = sess.run( [evaluation\_step, prediction], feed\_dict={bottleneck\_input: test\_bottlenecks,

ground\_truth\_input: test\_ground\_truth}) tf.logging.info('Final test accuracy = %.1f%% (N=%d)' %

(test\_accuracy \* 100, len(test\_bottlenecks))) if FLAGS.print\_misclassified\_test\_images:

tf.logging.info('=== MISCLASSIFIED TEST IMAGES ===')

for i, test\_filename in enumerate(test\_filenames): if predictions[i] != test\_ground\_truth[i].argmax(): tf.logging.info('%70s %s' %

(test\_filename, list(image\_lists.keys())[predictions[i]]))

save\_graph\_to\_file(sess, graph, FLAGS.output\_graph) with gfile.FastGFile(FLAGS.output\_labels, 'w') as f: f.write('\n'.join(image\_lists.keys()) + '\n')

if name == ' main ':

parser = argparse.ArgumentParser() parser.add\_argument(

'--image\_dir', type=str, default='',

help='Path to folders of labeled images.'

)

parser.add\_argument( '--output\_graph', type=str,

default='/tmp/output\_graph.pb', help='Where to save the trained graph.'

)

parser.add\_argument(

'--intermediate\_output\_graphs\_dir', type=str, default='/tmp/intermediate\_graph/',

help='Where to save the intermediate graphs.'

)

parser.add\_argument(

'--intermediate\_store\_frequency', type=int,

default=0, help="""\

How many steps to store intermediate graph. If "0" then will not store.\

"""

)

parser.add\_argument( '--output\_labels', type=str,

default='/tmp/output\_labels.txt',

help='Where to save the trained graph\'s labels.'

)

parser.add\_argument( '--summaries\_dir', type=str,

default='/tmp/retrain\_logs',

help='Where to save summary logs for TensorBoard.'

)

parser.add\_argument(

'--how\_many\_training\_steps', type=int,

default=6000,

help='How many training steps to run before ending.'

)

parser.add\_argument( '--learning\_rate', type=float, default=0.01,

help='How large a learning rate to use when training.'

)

parser.add\_argument(

'--testing\_percentage', type=int,

default=10,

help='What percentage of images to use as a test set.'

)

parser.add\_argument(

'--validation\_percentage', type=int,

default=10,

help='What percentage of images to use as a validation set.'

)

parser.add\_argument(

'--eval\_step\_interval',

type=int, default=10

)

parser.add\_argument( '--train\_batch\_size', type=int, default=100

)

parser.add\_argument( '--test\_batch\_size', type=int,

default=-1

)

parser.add\_argument(

'--validation\_batch\_size', type=int,

default=100

)

parser.add\_argument(

'--print\_misclassified\_test\_images', default=False,

action='store\_true'

)

parser.add\_argument( '--model\_dir',

type=str, default='/tmp/imagenet',

)

parser.add\_argument( '--bottleneck\_dir', type=str,

default='/tmp/bottleneck',

)

parser.add\_argument(

'--final\_tensor\_name', type=str, default='final\_result',

)

parser.add\_argument( '--flip\_left\_right', default=False, action='store\_true'

)

parser.add\_argument( '--random\_crop', type=int, default=0,

)

parser.add\_argument( '--random\_scale', type=int, default=0,

)

parser.add\_argument(

'--random\_brightness', type=int,

default=0,

)

parser.add\_argument( '--architecture', type=str,

default='inception\_v3',

)

FLAGS, unparsed = parser.parse\_known\_args() tf.app.run(main=main, argv=[sys.argv[0]] + unparsed)

* 1. **SYSTEM TESTING**
     1. **SYSTEM TESTING**

**7.1 TEST CASE**

Testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product:

* Meets the requirements that guided its design and development.
* Responds correctly to all kinds of inputs.
* Performs its functions within an acceptable time.
* Can be installed and run in its intended environments.
* Achieves the general result its stakeholder’s desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones.

|  |  |
| --- | --- |
| **Case name** | Best Case |
| **Input** | High Quality Image (.png ) |
| **Actual Output** | Normal (NO RISK) |
| **Expected Output** | Normal (NO RISK) |
| **Remarks** | Pass |

**TABLE 7.1 NORMAL TEST CASE**

|  |  |
| --- | --- |
| **Case name** | Best Case |
| **Input** | High Quality Image (.png ) |
| **Actual Output** | Moderate ❌(45% RISK) |
| **Expected Output** | Moderate ❌(45% RISK) |
| **Remarks** | Pass |

**TABLE 7.2 MODERATE TEST CASE**

* + 1. **CONCLUSION**

**8. CONCLUSION**

* 1. **CONCLUSION**

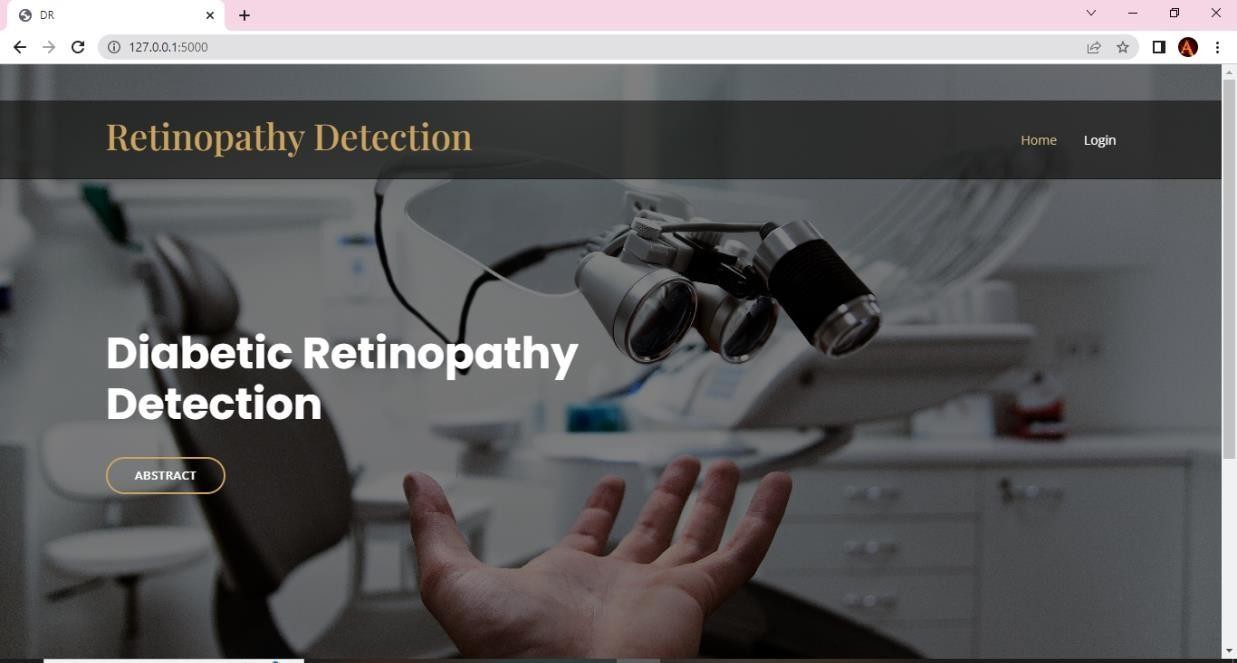
The fast and efficient early detection of Diabetic Retinopathy is only possible if there is an effective method for segmenting the diabetic features in the fundus image. The proposed system presents a fast, effective and robust way of detecting diabetic features in the fundus images which can be used for classification of the images based on the severity of the disease. The retinal images are subjected to gray scale conversion, preprocessing and feature extraction steps. The extracted features are fed to a Random Forest classifier which will classify the images into different severity levels. Thus this Random forest technique has given a successful DR screening method which helps to detect the disease in multiple stagesThe automated system will grade the images on the level of severity and refer only those patients who need medical attention to the Ophthalmologist. This will also relieve the burden on the doctor who would otherwise have to go through a lot of images which will come from the mass screening camps.

* 1. **FUTURE ENHANCEMENTS**

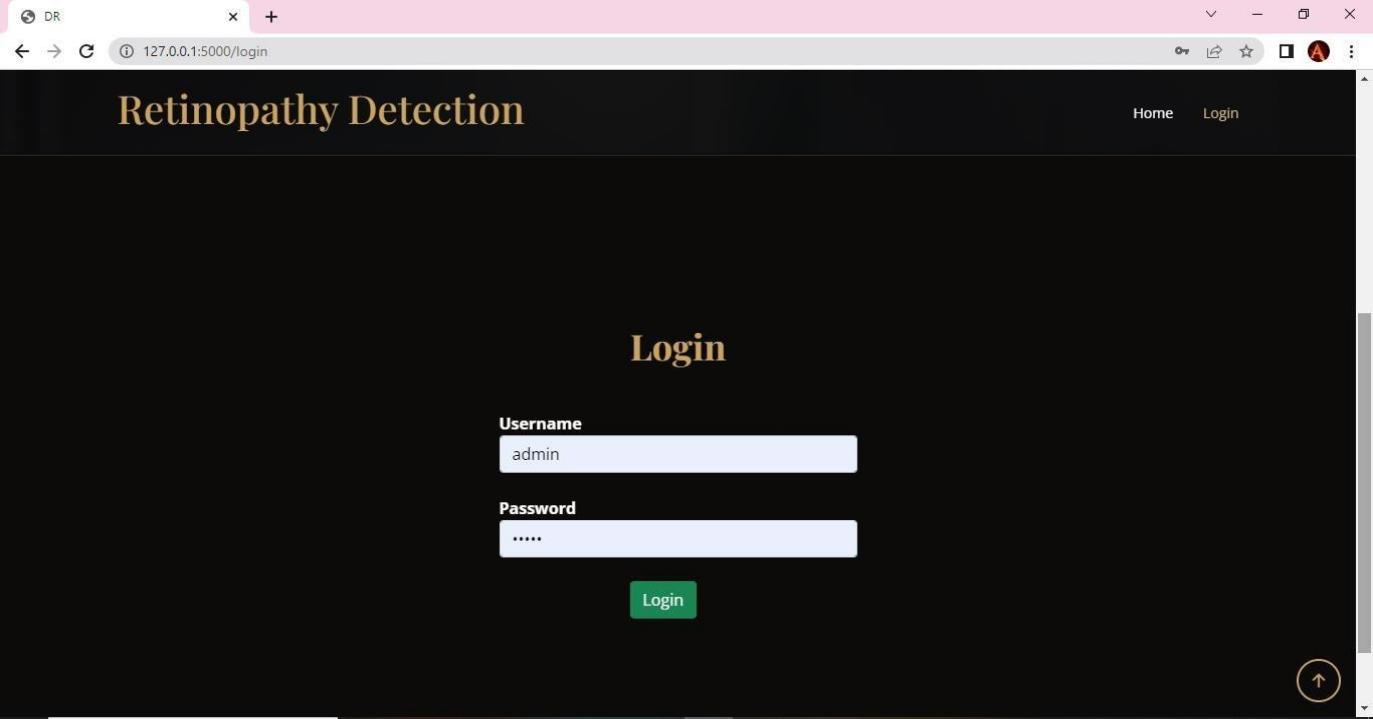
Diabetic retinopathy detection is become simple when creating a mobile application that may useful for all the people around the world who cannot afford money to go hospitals and also detecting the accurate percentage of people affected by this disease is also a future enhancement of this project.

**APPENDICES**

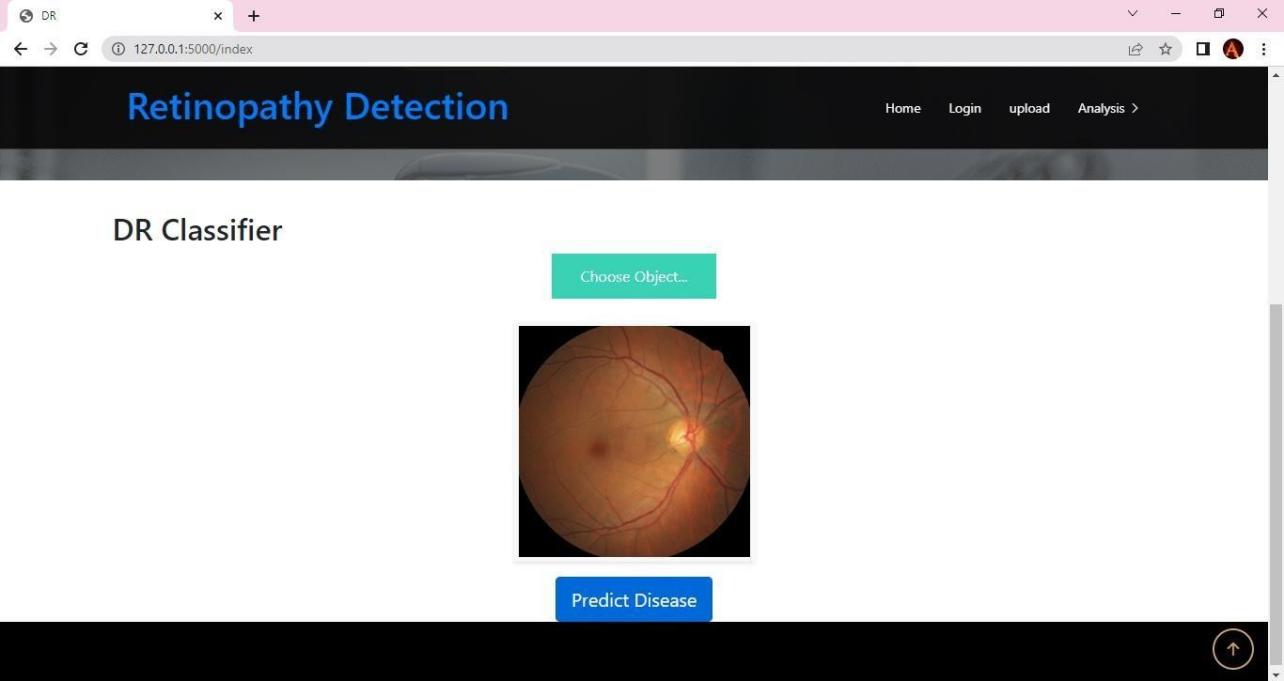
**A.1 SAMPLE SCREENS**



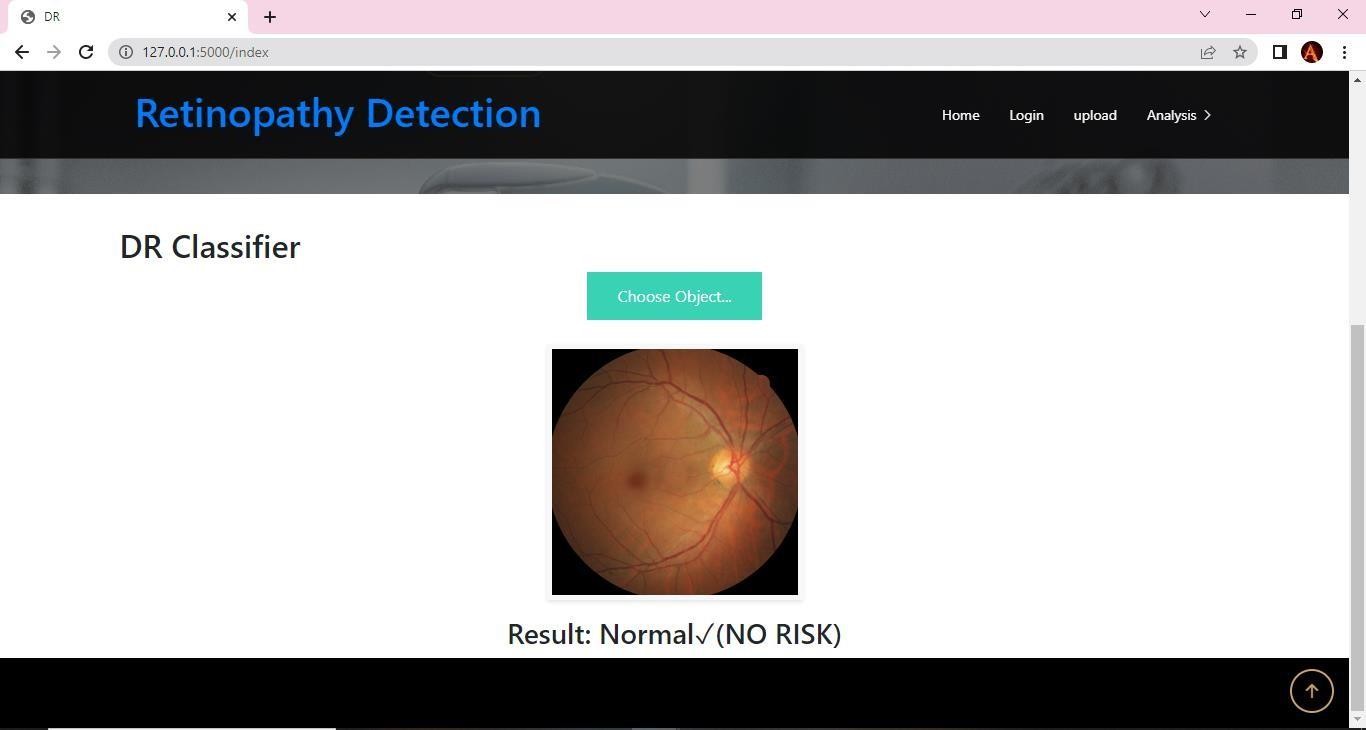
**FIG NO. A.1 MAIN PAGE**



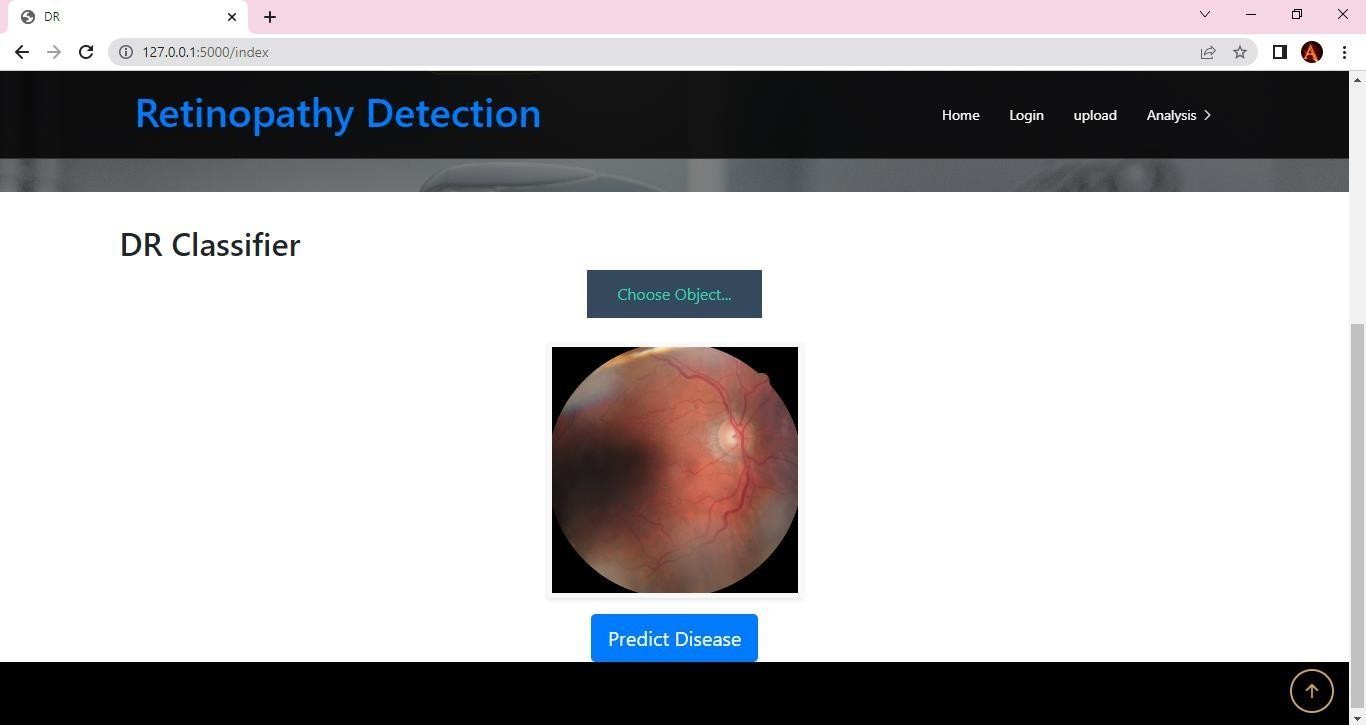
**FIG NO. A.2 LOGIN PAGE**



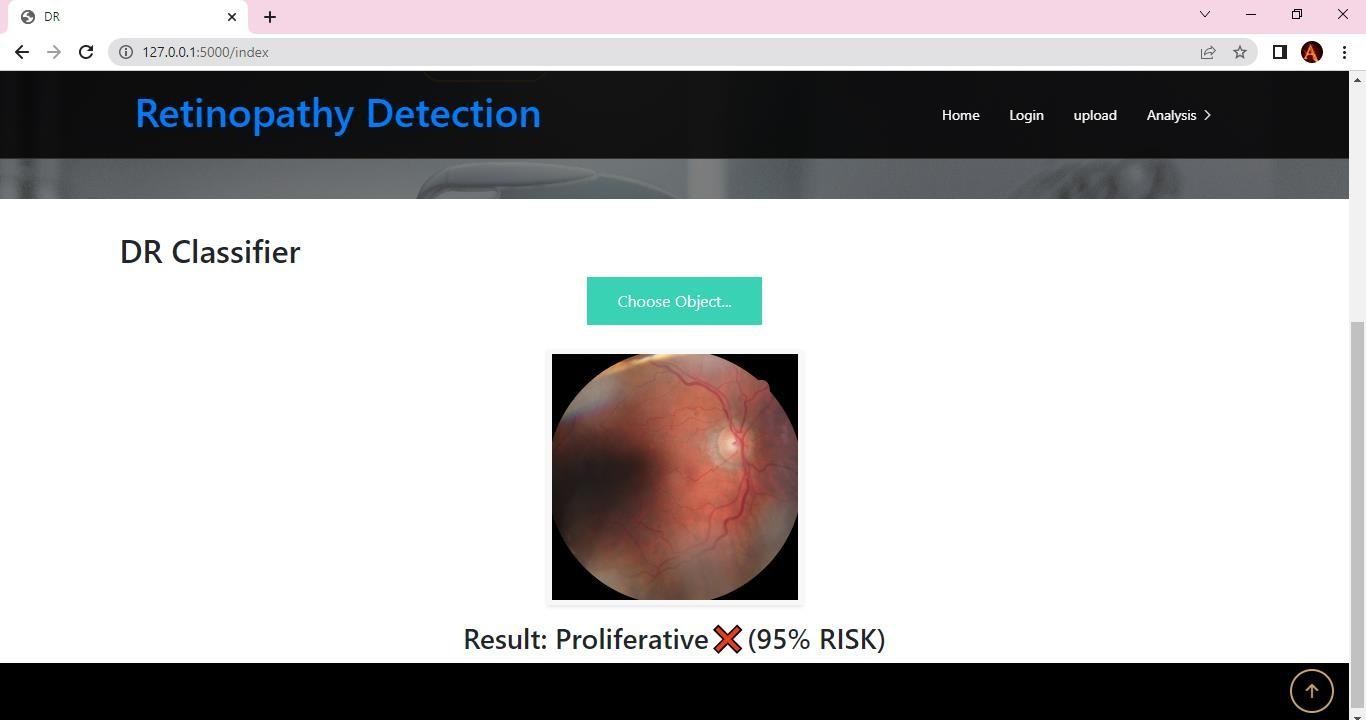
**FIG NO. A.3 UPLOADING NON AFFECTED FUNDUS IMAGE**



**FIG NO. A.4 DETECTION OF TYPE OF DISEASE (NORMAL)**



**FIG NO. A.5 UPLOADING AFFECTED FUNDUS IMAGE**



**FIG NO. A.6 DETECTION OF TYPE OF DISEASE (PROLIFERACTIVE)**

**REFERENCES**

[1]Thangam palaniswamy,Mahendrian vellingiri, “Internet of things and deep learning enabled diabetic retinopathy fundus images” vol.11,2023

[2]Jingbottu, Huan Wang, Yelu, “Graph adversarial transfer learning for diabetic retinopathy classification”,vol:10,2022

[3]Mohammad Z.Atwany, “Deep learning technique for diabetic retinopathy “vol:10,publish IEEE,2022.

[4]Hamza Mustafa,Syed Farooq Ali, “Multi-stream deep neural network for diabetic retinopathy severity classification under a boosting framework” vol:10,2022.

[5]Binhua yang,Tong YanLi “Classification of diabetic retinopathy severity based on GCA attention mechanism “, vol:9,2022

[6] Soo siang teoh,Haidi Ibrahim,Michel chi Seng tang “ A DeepLearning Approach for the Detection of Neovascularization in Fundus Images Using Transfer Learning. “ vol:9,2022.

[7] Omar bernabe, Elena Acevedo. “Classification of Eye Diseases in Fundus ImageS “ vol:10,IEEE,2022.

[8]Thippa Reddy,Guatam sirinivas, koppu,Praveen Kumar reddy “ Hand gesture recognition based on a Harris hawks optimized convolution neural network,” Computers &amp; Electrical Engineering, vol.100, Article ID 107836, 2022.

[9]Mohammad M.Abdelsalam,M.A.Zahran “ Diabetic retinopathy early detection based on multifracted geometry analysis for octa macular images using support vector machine

[10]Meysam tavakoli ,Alireza Mehdizadeh ,Tim ellis “;Automated microaneursysms detection in retinal images using radon transform and supervised learning application to mass screening of diabetic retinopathy “;,vol:9,2021.

[11] .Md Nahiduzzaman,Md.robiul Islam “ Hybrid CNN-SVD based prominent features extraction and selection for grading diabetic retinopathy using extreme learning machine “,vol:9,2021.

[12]D.Atlas, “International diabetes federation,” Brussels, Belgium, 10th edn edition,

2021

[13] M. Bader Alazzam, F. Alassery, and A. Almulihi, “Identification of diabetic retinopathy through machine learning,” Mobile Information Systems, vol. 2021.

[14] A. Y. Lee, R. T. Yanagihara, C. S. Lee et al., “Multicenter, head-to- head, real-world validation study of seven automated artificial intelligence diabetic retinopathy screening systems,” Diabetes Care, vol. 44, pp. 1168–1175, 2021.

[15] T. R. Gadekallu, N. Khare, S. Bhattacharya et al., “Early detection of diabetic retinopathy using pca-firefly based deep learning model,” Electronics, vol. 9, no. 2, p. 274, 2020.