**MAJOR PROJECT REPORT**

**On**

# Detection of Diabetic Retinopathy using Support Vector Machine

***Submitted in partial fulfilment of the requirements***

***for the award of degree***

***of***

**Bachelor of Technology**

**In**

**Computer Science & Engineering**

**Submitted To:**
**Dr. Priyanka Vashisht**

**Submitted by:**
**Akash Rai**

**Pragya Yadav**

**Prashant Goyal**

**Yash Mendiratta**

**Department of Computer Science & Engineering**

**Amity School of Engineering and Technology**

**Guru Gobind Singh Indraprastha University, New Delhi**

# CERTIFICATE

This is to certify that the project entitled "**Detection of Diabetic Retinopathy Detection using Support Vector Machine**" done by

| **Name:** | **Enrollment No.** |
|---|---|
| 1. Akash Rai | 00310402715 |
| 2. Pragya Yadav | 40810402715 |
| 3. Prashant Goyal | 41410402715 |
| 4. Yash Mendiratta | 40110402715 |

is an authentic work carried out by them under my guidance at ***AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY, NEW DELHI***. The matter embodied in this project report has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

**Project Guide**

Dr. Priyanka Vashisht

Asst. Professor,
Dept. of CSE & IT

# DECLARATION

The Submission to the Department of Computer Science and Engineering, Amity School of Engineering and Technology, Delhi, submitted by the authors for the purpose of obtaining the degree of Bachelor of Science in Computer Science. We here by announce that the results of this thesis are entirely based on our research. Resources taken from any research conducted by other researchers are mentioned through reference. This thesis either in whole or in part, hasn’t been previously submitted for any degree.

| | |
|---|---|
| **Akash Rai** (00310402715) | **Pragya Yadav** (40810402715) |
| **Yash Mendiratta** (40110402715) | **Prashant Goyal** (41410402715) |

# ACKNOWLEDGEMENT

For the accomplishment of this project, we would like to take this opportunity to express gratitude to the people who have been a part of this project right from its inception and helped and supported me. It is to them we owe our deepest gratitude.

We express our sincere gratitude to **Prof. Dr. Rekha Aggarwal (Director)**, Amity School of Engineering and Technology, and **Dr. Pinki Nayak, Head of Department (Department of CS/IT)**, for their constant encouragement and guidance.

We are extremely thankful to **Dr. Priyanka Vashisht,**Assistant Professor for sharing expertise, sincere and valuable guidance and encouragement extended towards us.

# ABSTRACT

Diabetic Retinopathy (DR) is a medical condition of the eye that is caused by diabetes. It is one of the leading causes for blindness in the working age population. It's caused by damage to the blood vessels of the light-sensitive tissue at the back of the eye (retina). Eventually, it can cause blindness.

The objective of our thesis is to detect  the presence of diabetic retinopathy by applying ensemble of machine learning classifying algorithms on features extracted from output of different retinal image. It will give us accuracy of which algorithm will be suitable and more accurate for prediction of the disease. Decision making for predicting the presence of diabetic retinopathy is performed using Support Vector Machine and K- nearest Neighbour algorithms.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

|  |  |
|---|---|
| DR– | Diabetic Retinopathy |
| MA – | Micro Aneurysms |
| FSH– | Flame Shaped Haemorrhages. |
| CNN- | Convolution Neural Networks |
| VTDR- | Vision-Threatening Diabetic Retinopathy |

# CHAPTER 1:
# INTRODUCTION

Diabetes is a chronic and organ disease that occurs when the pancreas does not secrete enough insulin or the body is unable to process it properly. Overtime, diabetes affects the circular system, including that of the retina. Diabetes retinopathy (DR) is a medical condition where the retina is damaged because of fluid leaks from blood vessels into the retina. It is one of the most common diabetic eye diseases and a leading cause of blindness. Nearly 415 million diabetic patients are at risk of having blindness because of diabetics. It occurs when diabetes damages the tiny blood vessels inside the retina, the light sensitive tissue at the back of the eye. This tiny blood vessel will leak blood and fluid on the retina forms features such as micro-aneurysms, haemorrhages, hard exudates, cotton wool spots or venous loops. Diabetic retinopathy can be classified as non – proliferative diabetic retinopathy (NPDR) and proliferative diabetic retinopathy (PDR) . Depending on the presence of features on the retina, the stages of DR can be identified. In the NPDR stage, the disease can advance from mild , moderate to severe stage with various levels of feature scentless growth of new blood vessels . PDR is the advanced stage where the fluids sent by the retina for nourishment trigger the growth of new blood vessels. They grow along the retina and over the surface of the clear, vitreous gel that fills the inside of the eye. If they leak blood, severe vision loss and even blindness can result.

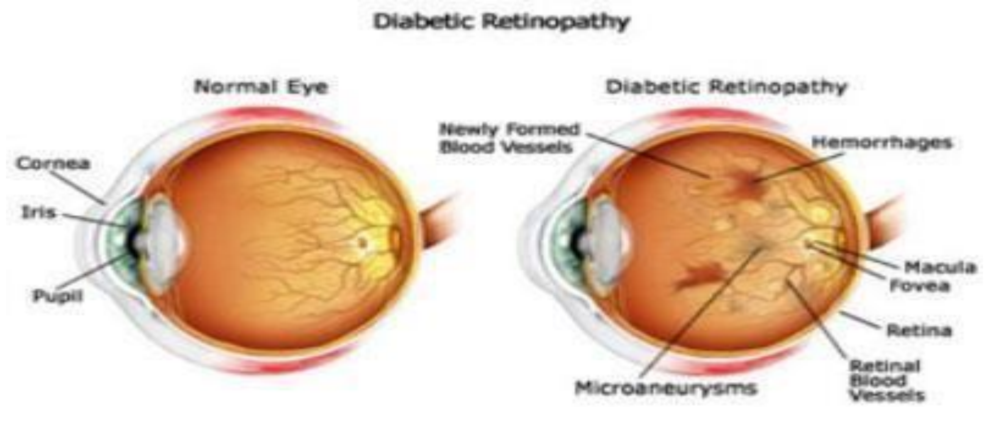


Fig. 1 Normal Eye vs Affected Eye

## 1.1 Motivation

As are search group, we wanted to do our undergraduate thesis on are search that will assist a huge amount of people in their healthy lives. The number of people with diabetic retinopathy is growing higher day by day. It is estimated that the number will grow from 126.6 million to 191.0 million by 2030 and the number with vision -threatening diabetic retinopathy (VTDR) will increase from 37.3 million to 56.3 million, if any proper action is not taken. Despite growing evidence, the effectiveness of routine DR screening and early treatment, it is frequently leading to poor visual functioning and represents the leading cause of blindness. Most of the time it has been neglected in health care and in many low-income countries because of inadequate medical service. While researching about these factors we get motivated to work with this topic. As there are insufficient ways to detect about diabetic retinopathy, we will build a system which we prediction about diabetic retinopathy. Thus, we decided to use Machine Learning Algorithms for the prediction of this disease.

## 1.2 Objectives and Goals

This thesis mainly focuses on the prediction of diabetic retinopathy and analysis performed of different algorithm for the prediction. Machine learning algorithms such as KNN can be trained by providing training datasets to the mind then these algorithms can predict the data by comparing the provided data with the training datasets.

Our objective is to train our algorithm by providing training dataset to it and our goal is to detect diabetic retinopathy using different types of classification algorithms. The main objective of the project is to unambiguously define a database and a testing protocol which can be used to benchmark diabetic retinopathy detection methods. By using the database and the defined testing protocol, the results between different methods can be compared.

## 1.3 Thesis Organization

Chapter 1 is the **INTRODUCTION** of the thesis. The motivation and objective & Goals of the thesis are described here.

Chapter 2is **INTRODUCTION TO ANACONDA**. This chapter contains information about anaconda and various other software like spyder, jupyter.

Chapter 3 is **LITERATURE REVIEW.** This chapter consists of Literature Review.which indicates our information collection repository. This chapter also consists of "Related Works and research" which indicates to the real life works and researches done by others, which are related to our thesis work in many ways.

Chapter 4 is **PROPOSED MODEL FOR PREDICTION**. Which consists of PROPOSED METHODLOGY.

Chapter 5 is **IMPLEMENTATION.** Which consists of Data collection, Data Description , Visualization and Split Data.

Chapter 6 is **IMAGE PROCESSING**. Which consists of Techniques used in model.

Chapter 7 is **Hardware AND GPU** specification.

Chapter 8 is **Experimental Result & Analysis** where we have shown the machine learning .Algorithms, which model gives maximum accuracy, which has better prediction. For analysing, we have used histograms, plots and different kind of comparison graphs and so on.

Chapter 9 is **CONCLUSION** which consists of Conclusion Remarks and Future Works.

# CHAPTER 2:

# INTRODUCTION TO ANACONDA

Anaconda is a fairly sophisticated installer. It supports installation from local and remote sources such as CDs and DVDs, images stored on a hard drive, NFS, HTTP, and FTP. Installation can be scripted with kickstart to provide a fully unattended installation that can be duplicated on scores of machines. It can also be run over VNC on headless machines. A variety of advanced storage devices including LVM, RAID, iSCSI, and multipath are supported from the partitioning program. Anaconda provides advanced debugging features such as remote logging, access to the python interactive debugger, and remote saving of exception dumps.

## Steps to Download Anaconda 3.6

1. Download anaconda 3.6
2. Based on your Operating System, click on Windows or Mac.
3. Download Python 3.6 version for your 64/32 bits OS.
4. Right click on "My computer" and goto Properties.

## Steps to Configure Anaconda 3.6:

1. Search for anaconda command prompt and open it.
2. Type in the following commands to update conda and jupyter

#update conda :conda update conda

#update jupyter :conda update jupyter

Type the following commands to install the inbuilt packages:
#install numpy :conda install numpy

#install seaborn :conda install seaborn

#install matplotlib :conda install matplotlib

## 2.1 Scikit- Image

scikit-image
image processing in python

Fig. 2.1: Scikit Image

Scikit-image is a collection of algorithms for image processing. It is available free of charge and free of restriction. We pride ourselves on high-quality, peer-reviewed code, written by an active community of volunteers.

## 2.2 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

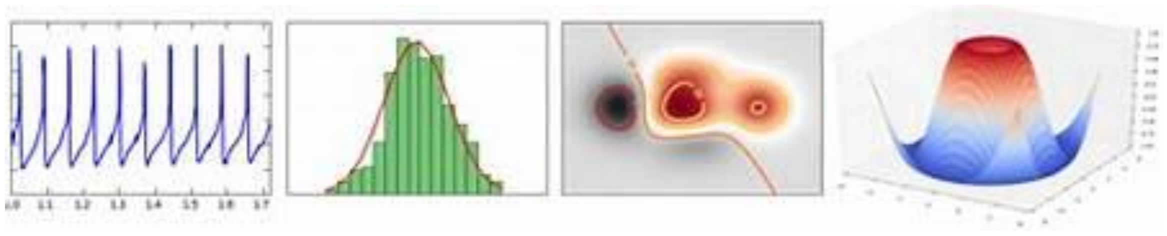


Fig. 2.2: Various Graphs in Matplotlib

## 2.3. Numpy

Fig. 2.3 :Numpy Logo

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allowsNumPy to seamlessly and speedily integrate with a wide variety of databases.

## 2.4 Scikit-Learn :

Fig. 2.4 :Scikit –learn logo

- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

## 2.5 Ide Used For Python: Spyder (Python 3.6)

Fig. 2.5: Spyder Logo

Spyder is an open source cross-platform integrated development environment for scientific programming in the Python language. Spyder integrates NumPy, SciPy, Matplotlib and IPython, as well as another open source software.

In order to use spyder IDE, we need to first download the anaconda distribution which contains spyder and also other pack.

## 2.6 Tensorflow

Fig. 2.6: TensorFlow logo

TensorFlow is an open source software library for numerical computation using data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture enables you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. TensorFlow also includes Tensorboard,a data visualization toolkit.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization for the purposes of conducting machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.

TensorFlow provides stable Python API and C APIs as well as without API backwards compatibility guarantee like C++, Go, Java, JavaScript and Swift.

# CHAPTER 3:

# LITERATURE REVIEW

This chapter contains literature review related with supervised learning model, classification algorithms like CNN. This chapter also refers the related works and research. Besides it will give information about our research activity.

## 3.1 Machine Learning

Machine learning, a branch of artificial intelligence, concerns the construction and study of systems that can learn from data. Machine learning algorithms use computational methods to "learn" information directly from data without relying on a predetermined equation as a model. The algorithms adaptively improve their performance as the number of samples available for learning increases. Tom M. Mitchell provided a widely quoted and more formal definition: A computer program is said to learn from experience with respect to some class of tasks and performance measure, if its performance at tasks in, as measured by P, improves with experience.

The core of machine learning deals with representation and generalization. Representing the data instances and functions evaluated on these instances are part of all machine of learning systems. Generalization is the ability of a machine learning system to perform accurately on new, unseen data instances after having experience data learning instance. The training examples come from some generally unknown probability distribution and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases. The performance of generalization is usually evaluated with respect to the ability to reproduce known knowledge from newer examples. There are different types of machine learning, but the two main ones are:

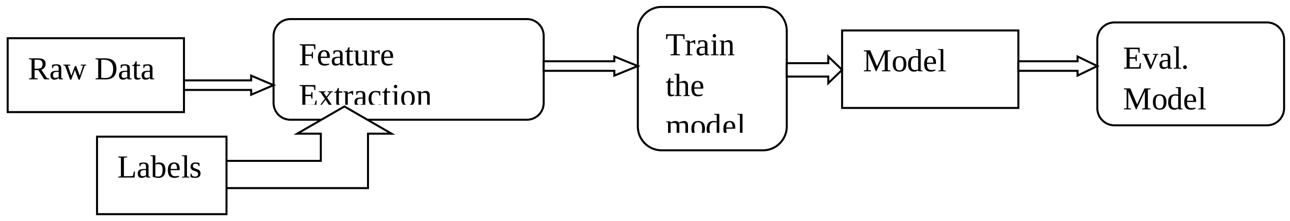
- Supervised Learning
- Unsupervised Learning

## 3.2 Supervised Learning Model

Supervised learning is the machine learning task of inferring a function from supervised training data. Training data for supervised learning includes a set of examples with paired input subjects and desired output. A supervised learning algorithm analyses the training data and produces an inferred function, which is called classifier or aggression function. The function should predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data to unseen situations in a reasonable way.

A simple analogy to supervised learning is the relationship between a student and a teacher. Initially the teacher teaches the student about a particular topic. Teaching the student the concepts of the topic and then giving answers to many questions regarding the topic. Then the teacher sets an exam paper for the student to take, where the student answers newer questions.

Training



Prediction

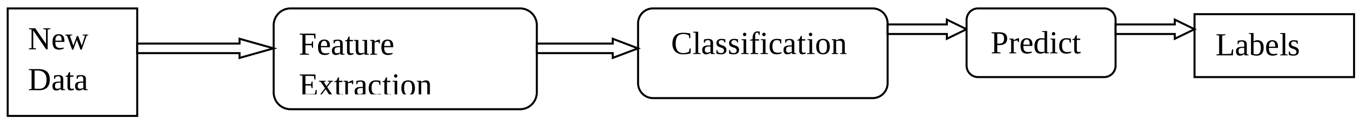


Fig 3.1 :Flow-diagram: Work flow of Supervised Learning Model

## 3.3 Algorithms :

Since there are so many algorithms for machine learning, it is not possible to use all of them for analysis. For this research paper, we will be using four of them Support Vector Machine (SVM)

### 3.3.1 Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data(supervised learning), the algorithm outputs an optimal hyperplane which categorizes new example

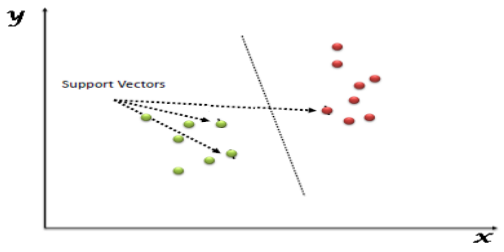


Fig : 3.2 SVM hyperplanes

#### 3.3.1.1 WORKING

As we know that we got accustomed to the process of segregating the two classes with a hyper-plane.

- Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle. In this scenario, hyper-plane "B" has excellently performed this job.

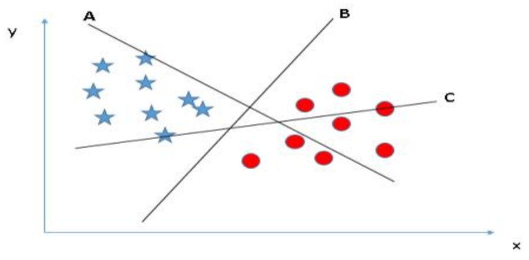


Fig: 3.3 classification using hyperplanes

- Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.

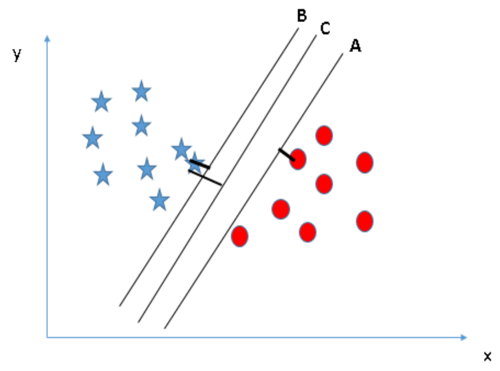
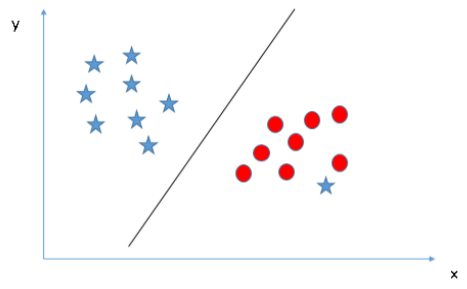


Fig: 3.4 Margins

- Can we classify two classes (Scenario-3)?: Here, one star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.



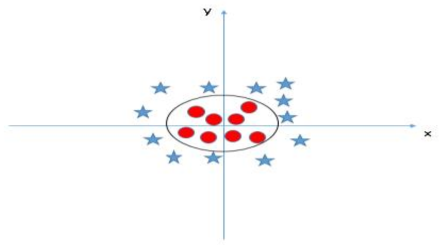


Fig: 3.5Non Robust to outliers

Fig: 3.6 Robust to outliers

- Find the hyper-plane to segregate to classes (Scenario-4): In the scenario below, we can't have linear hyper-plane between the two classes. Yes, SVM can solve this method by using **kernels.** i.e. it converts not separable problem to separable problem, these functions are called kernels.

### 3.3.2 Neural Network

The term neural network was traditionally used to refer to a network or circuit of neurons. The modern usage of the term often refers to artificial neural networks, which are composed of artificial neurons or nodes. Thus, the term may refer to either biological neural networks, made up of real biological neurons, or artificial neural networks, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modelled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.

Unlike von Neumann model computations, artificial neural networks do not separate memory and processing and operate via the flow of signals through the net connections, somewhat akin to biological networks.

These artificial networks may be used for predictive modelling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

## 3.4 Related Works & Research

Diabetic retinopathy is the leading cause of blindness in the working- age population of the developed world. Since 1982, the quantification of diabetic retinopathy and detection of features such a s exudate sand blood vessel on fundus images were studied. A lot of work has been done in this field. Before starting implementation of main task, we go through similar paper to know about the whole system such as what are the things, we need to consider in order to detect diabetic retinopathy. Akara Matthew N. Dailey has proposed a “Machine learning approach to automatic exudate detection in retinal images from diabetic patients” . In their paper they presented a series of experiments on features election and exudates classification using K-nearest Neighbour (KNN) and support vector machine (SVM) classifiers.

Rajendra Acharya U.,E.Y.K.Ng , Kwan-HoongNgandJasjit S . Suri introduced algorithms for the automated detection of diabetic retinopathy using digital fundus images where they improved an algorithm used for extraction of some features from digital fundus images. Moreover, Varun G. and Lily P. has used deep learning for detection of diabetic retinopathy.

In “Diagnosis of Diabetic Retinopathy using Machine Learning” research paper S.Gupta and K. A M tried to detect retinal micro-aneurysms and exudates retinal funds from images. After preprocessing, morphological operations are performed to find the feature and the features are get extracted such a GLCM and splat for classification. They achieved the sensitivity and specificity of 87% and 100% respectively with accuracy of 86%.

Tiago T.G. in his paper “Machine Learning on the Diabetic Retinopathy Debrecen Dataset” has used R language for predicting diabetic retinopathy. He used a dataset in which the features were extracted from images of the eye of a diabetic patient. In his work he used eight different classification algorithms and also shown some comparisons. He achieved 78% accuracy from his work.

Those are some related paper of our topic from where we took knowledge and idea to develop version. In our work we will use different machine learning classification algorithms to classify diabetic retinopathy.

# CHAPTER 4:

# PROPOSED MODEL FOR PREDICTION

This chapter contains proposed model, dataset collection, description, data visualization and also classifying algorithms that are used for analysis performance.

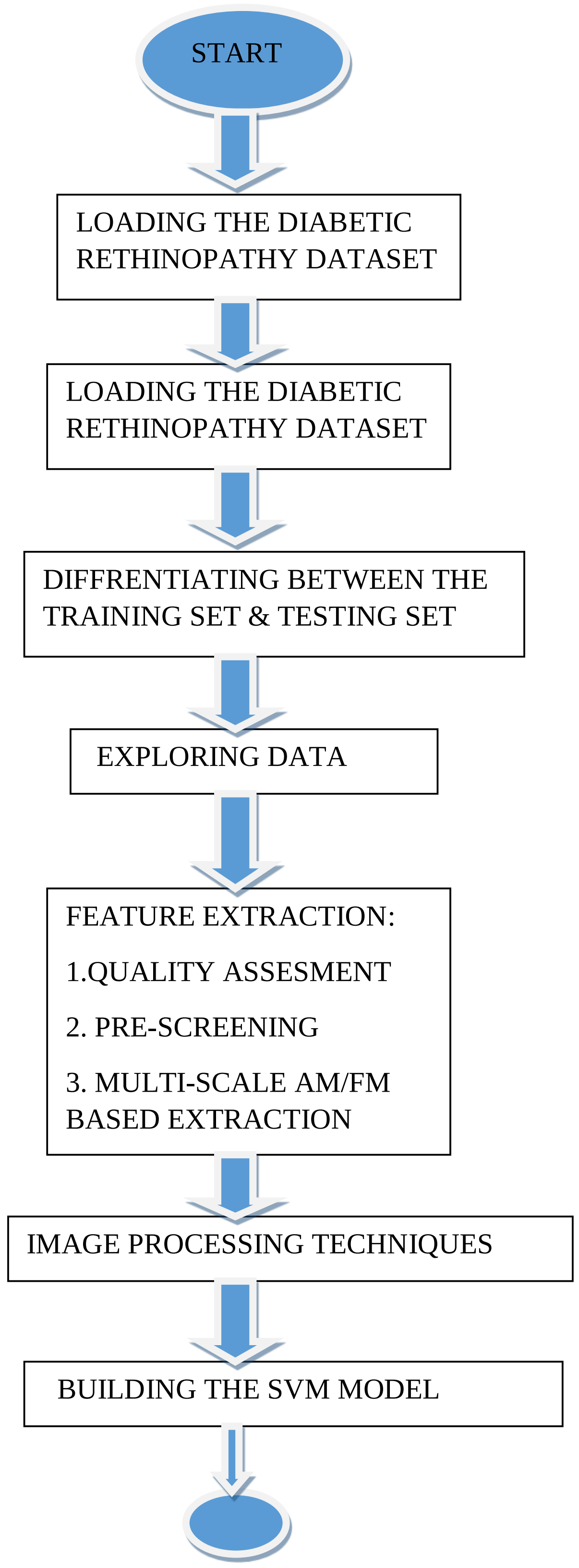
## 4.1 Proposed Model

Our First phase is data acquisition. We have collected our dataset from Kaggle. The dataset contains features to predict whether an image have signs of diabetic retinopathy or not. Then features and labels of the dataset are identified.

After that the dataset is divided into two sets, one for training where most of the data is used and the other one is testing. In training set algorithms has been fitted for the analysis performance of the model.

The algorithms we used are convolution neural networks. After the system has done learning from training datasets, newer data is provided without outputs.  The final model generates the output using the knowledge it gained from the data on which it was trained.

In final phase we get the accuracy of each algorithm and get to know which particular algorithm will give us more accurate results for the prediction of diabetic retinopathy.


START
LOADING THE DIABETIC RETHINOPATHY DATASET
LOADING THE DIABETIC RETHINOPATHY DATASET
DIFFRENTIATING BETWEEN THE TRAINING SET & TESTING SET
EXPLORING DATA
FEATURE EXTRACTION:
1.QUALITY ASSESMENT
2. PRE-SCREENING
3. MULTI-SCALE AM/FM BASED EXTRACTION
IMAGE PROCESSING TECHNIQUES
BUILDING THE SVM MODEL

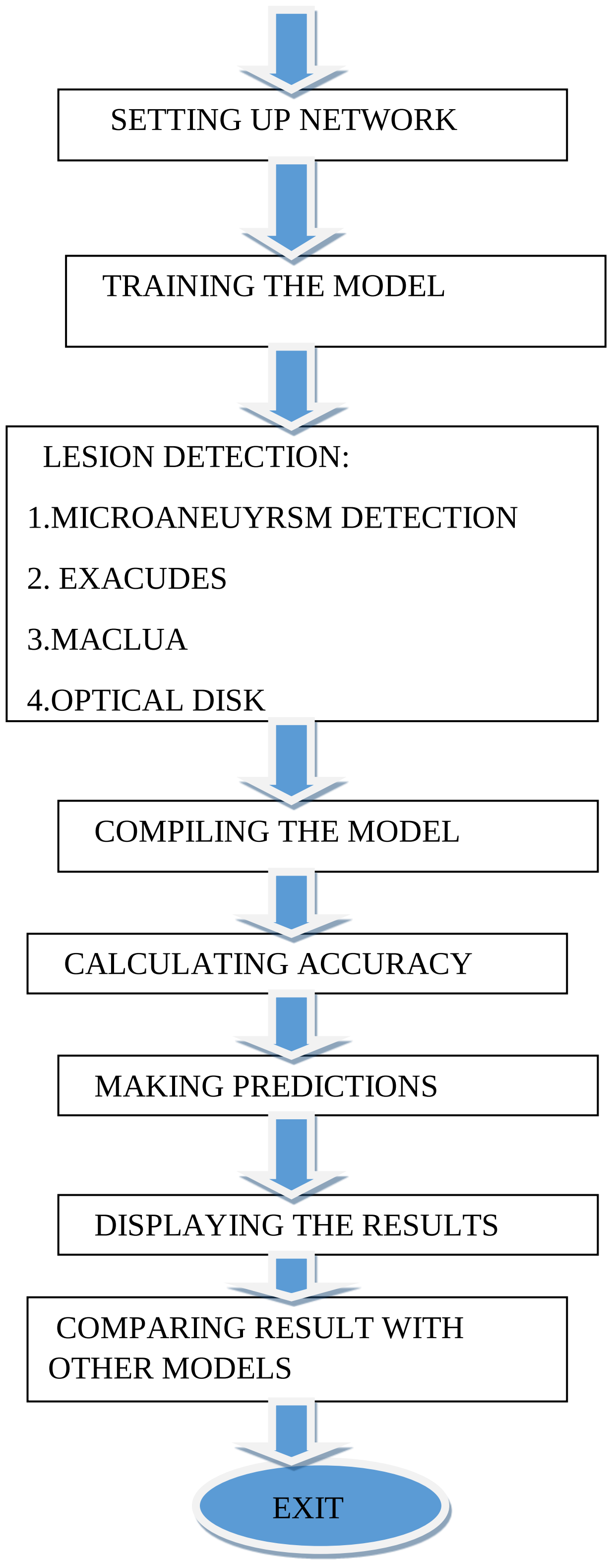


Fig 4: Flow-diagram: Algorithm for Proposed Methodology

# CHAPTER 5:

# IMPLEMENTATION

## 5.1 Data Collection

In our project we have use dataset that is obtained from Kaggle. This dataset contains images that can predict signs of diabetic retinopathy or not. All features represent either a detected vision descriptive feature of an anatomical part or an image-level descriptor. The Kaggle database has been assisted diagnoses of diabetic retinopathy. We have seen different kind of dataset using it hub and other websites which was used for different kind of projects based on diabetic retinopathy. As we wanted to work with detection of diabetic retinopathy, this dataset will be appropriate for our work as it has different type of features.

## 5.2 Data Description

Our dataset contains different types of features that is extracted from the Kaggle image set. This dataset is used to predict where an image contains signs of diabetic retinopathy or not. The value here represents different point of retina of diabetic patients.

The dataset has following features:

1. Dataset is generated by Eyepacs and Available at Kaggle.
2. http://www.kaggle.com/c/diabetic-retinopathy-detection/data
3. Consists of Images with different shades different camera
4. Score by trained professional.

## 5.3 Data Visualization

Another important feature in the data distribution is the skewness of each class. Data visualization helps to see how the data looks like and also what kind of data correlation we have. The dataset distribution of each feature. This is a histogram. A histogram is an accurate graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. Histograms are a great way to get to know your data. They allow you to easily see where a large and a little amount of the data can be found. In short, the histogram consists of any x-axis and y-axis, where they-axis shows how frequently the values on the x-axis occur in the data.

## 5.4 Split Dataset

Separating data into training and testing sets is an important part of evaluating data mining models. Typically, when separating a dataset into two parts, most of the data is used for training, and a smaller portion of the data is used for testing. We have also split our dataset into two sets. One is for training and another for testing. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. After the model has been processed by using the training set, we have tested the model by making predictions against the test set because the data in the testing set already contains known values for the attribute that we want to predict, it is easy to determine whether the model's guesses are correct or not. In addition, we have used 80% of our data for training and 20% for testing.

# CHAPTER 6:

# IMAGE PROCESSING

## 6.1 Loading An Image In Python

1. Importing the packages like matplotlib, skimage and numpy.
2. From skimage package importing a module called **“io”** with which we can read images.
3. Io.imread will read the image and store it into **“image”** variable.
4. Plt.imshow will display the image read earlier.

## 6.2 Converting an image into Green Channel

There are different ways to convert an image into green channel.

1. Here we are using the PILLOW package using which we can quickly and very easily convert an Importing PIL package.
2. Declaring a variable img which will store the image to be convert into green channel.
3. Img.save will save the image into the path by replacing the older image.
4. Plt.imshow will display the rgbimage into a green channel image.

## 6.3 Different Ways to Convert an Image into Green channel

There are various methods/ways available to convert an RGB image into green channel.
The following methods were given below:

### 6.3.1 Using Rgb2green

(RGB) converts the truecolor image RGB to the grayscale intensity image I. The rgb2green function converts RGB images to green channel by eliminating the hue and saturation information while retaining the luminance. If you have Parallel Computing Toolbox™ installed, rgb2green can perform this conversion on a GPU. I = rgb2green(RGB) newmap = rgb2green(map)

## 6.3.2 Using As-Green=”True”

By enabling the as-green  function to true. The image will be plotted in green channel.

## 6.3.3 Using Pillow

Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. The latest version of PIL is 1.1.7, was released in September 2009 and supports Python 1.5.2–

2.7, with Python 3 support to be released "later".

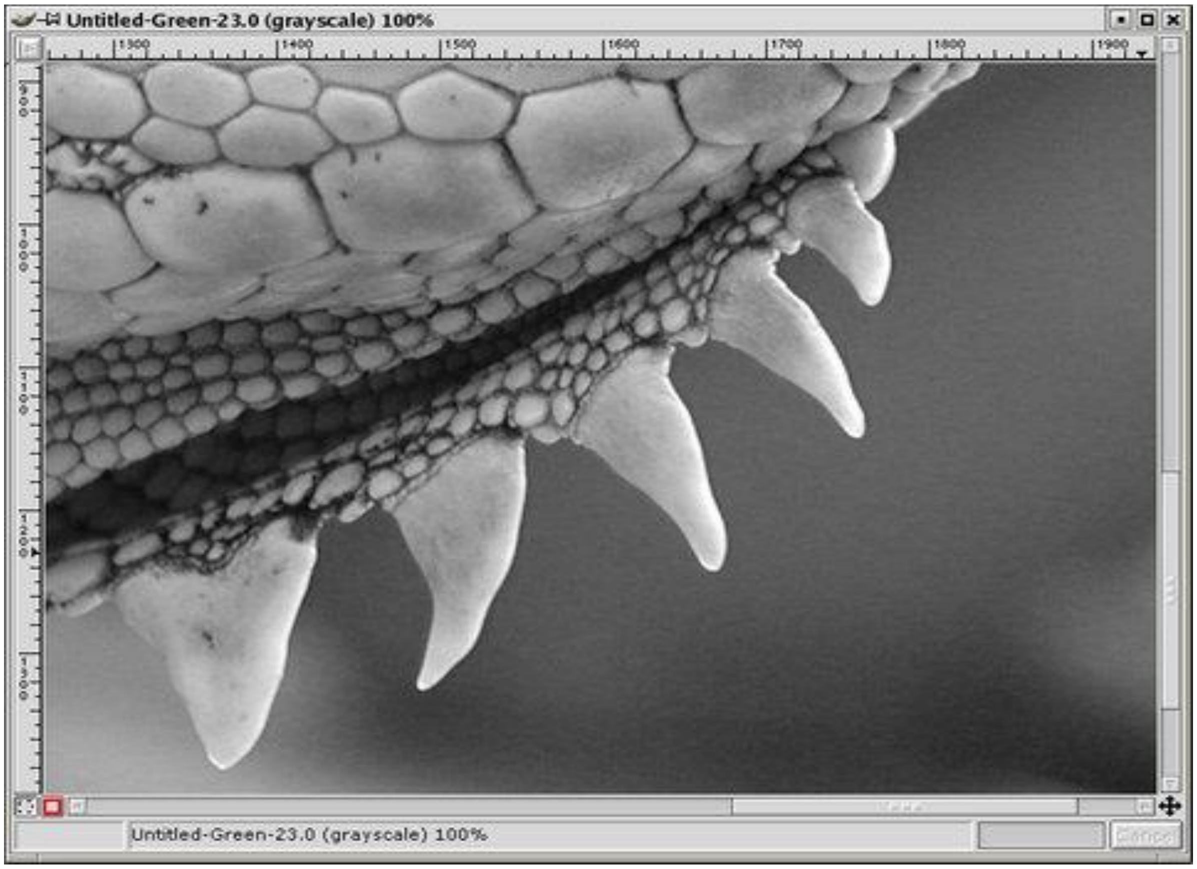

Untitled-Green-23.0 (grayscale) 100%
Untitled-Green-23.0 (grayscale) 100%

Fig. 6.1:Differents ways to convert into Green channel

## 6.4 Loading a Collection of  Images

In order to load image collection in spyder, we follow the following code:

1. Importing the packages like matplotlib, skimage and numpy.

2.From skimage package importing a module called **“io”** with which we can read images.

3.Io.imread_collection will read the images and store them into **“data1”** variable.

4. Plt.subplots will plot the images read earlier.

## 6.5 Loading a Collection of Retinal Images

In order to load an image collection in spyder, we follow the following code:

1.Importing the packages like matplotlib, skimage and numpy.

2.From skimage package importing a module called **“io”** with which we can read images.

3.Io.imread_collection will read the image and store it into **“data1”** variable.

4.Plt.subplots will plot  the images read earlier having the same extension

## 6.6 Creating a Green channel Images Dataset and Storing  Into a Separate Folder

In order to load an image collection in spyder, we follow the following code:

1.Importing the packages like matplotlib, skimage and numpy.

2.Applying For loop in order to read all the images in the directory mentioned above.

3.Rgb2grey will convert the images in the directory to gray.

4.Io.imread will read the gray images and io.imsave them into the path defined.

## 6.7 Contrast Enhancement

Image enhancement techniques have been widely used in many applications of image processing where the subjective quality of images is important for human interpretation. Contrast is an important factor in any subjective evaluation of image quality. Contrast is created by the difference in luminance reflected from two adjacent surfaces.

In other words, contrast is the difference in visual properties that makes an object distinguishable from other objects and the background.

## 6.8 The Reasons to Use Adaptive Equalisation

1. The image looks brighter
2.The image would be a good model to work on.
3.The contrast is better than the other techniques.
4.The contents of the image are more visible in this mechanism

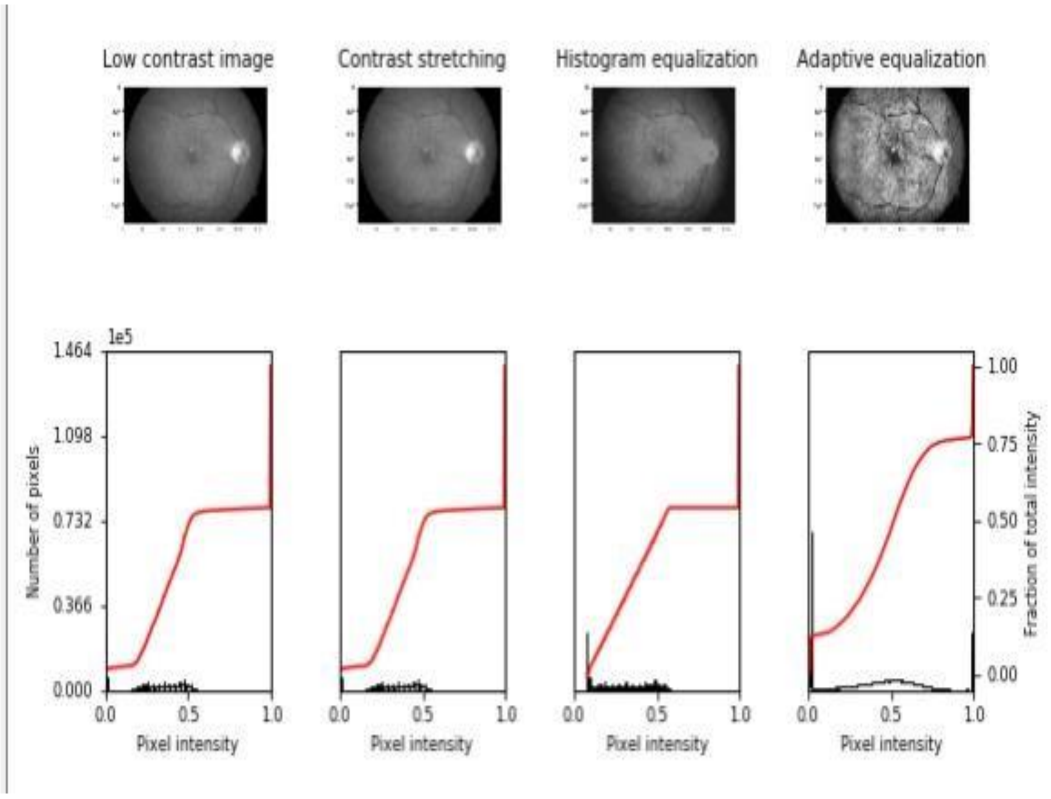


Fig. 6.2 : Different methods of Contrast Enhancement

# CHAPTER 7:

# HARDWARE & CPU SPECIFCATION

## 7.1 Hardware Specification

### CPU:

| **Name** | **AMD FX(tm)-8300** |
|---|---|
| Cores | 8 |
| Clock Speed(mhz) | 3300 |
| Typical TDP | 95W |
| Socket | Socket AM3+ |
| Micro Architecture | Piledriver |
| Platform | Anaconda |
| Processor core | Vishera |
| Core Stepping | OR-C0 |
| CPUID | 600F20 |

Table 7.1: CPU Specification

### Memory:

| | |
|---|---|
| Physical memory | 16GB |
| GPU | NVIDIA GeForce GT 620 |

Table 7.1.1: GPU Specification

## Software:

| **Name** | **Type** | **Version** | **Architecture** |
|---|---|---|---|
| Anaconda | Python distributer | Anaconda2.4.2.0 Python 3.5 | 64bit (x86) |
| Spyder | Python IDE | 2016.2.3Build#PC 162.1967.10. | 64 bit (x86) |
| Jupyter | Python package | 0.16.1 | 64 bit (x86) |

Table 7.2 : Software Specifications

## Operating System:

| | |
|---|---|
| Name | Microsoft Windows 10 Pro |
| Version | 10.0.10586 |
| Build Number | 10586 |
| Systemtype | 64bit |

Table7.2.1:Operating System Details

# CHAPTER 8:

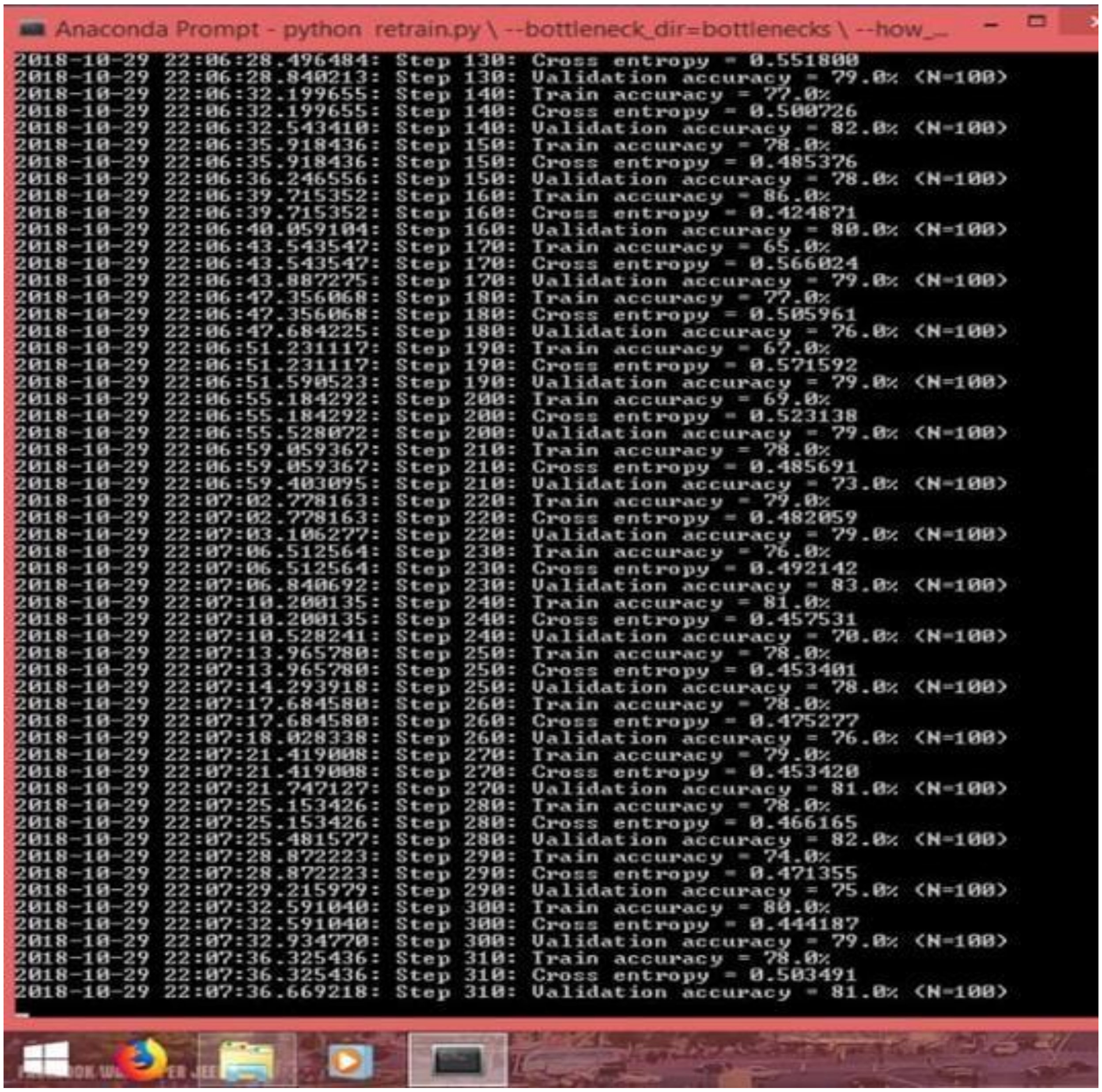
# EXPERIMERIMENTAL RESULT & ANALYSIS

In the previous chapter we have discussed about proposed system and implementation of our thesis. We have demonstrated how we collected our dataset, dataset description, visualization and algorithms we used. Now we discussing about the results we obtained from our experiments upon the implementation of this system. We have divided our dataset into two parts- training and testing dataset. In this chapter we will show the outcome of the training and testing dataset. As mentioned, before we have used four machine learning algorithms. First, we trained our dataset with these algorithms and then we built a model. Then, we tested our testing data set in this model. If the test set accuracy is near to trainset accuracy then we can conclude that we built a good model.

We have total 2100 data of different individual in our dataset. There are 2100 rows and 20 columns in the dataset. After splitting the data into two parts now we have 920 rows for train data and for test data we have 231 rows. When we trained our train data for analysis performance of different algorithms. This is the result we got.

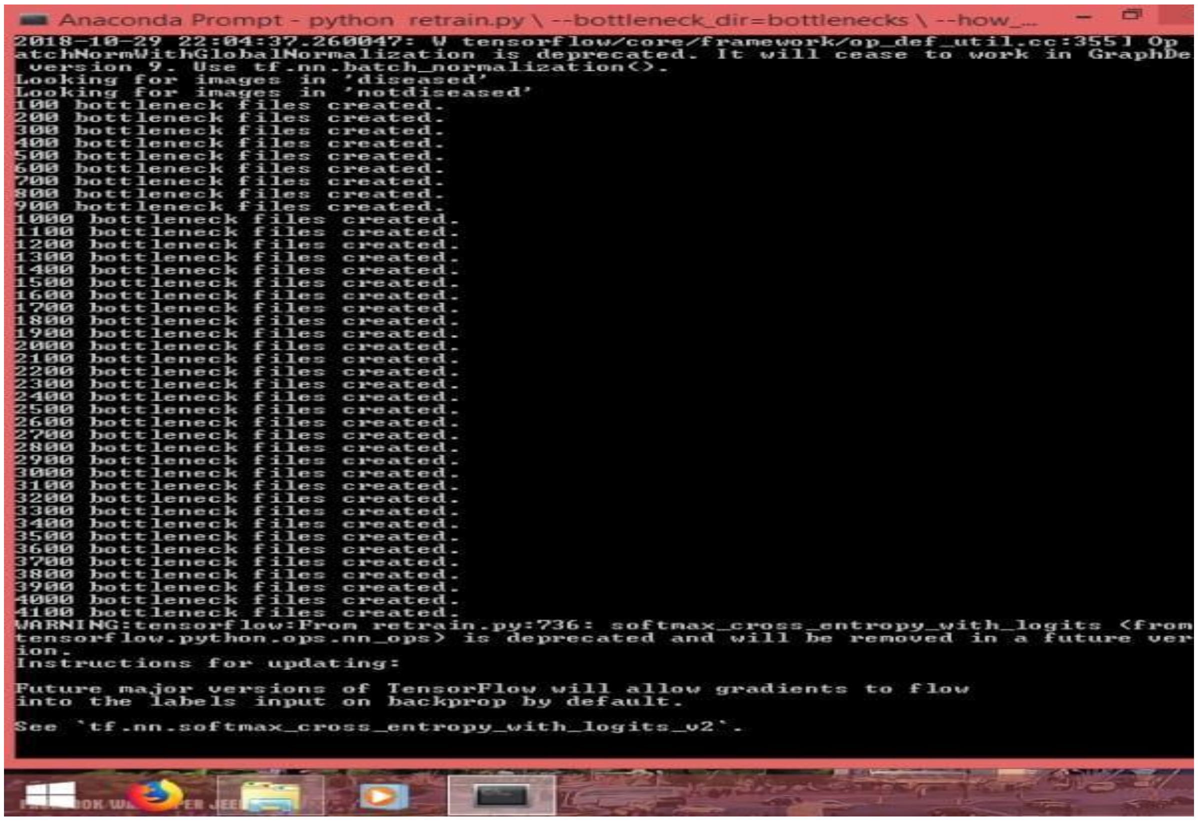
## 8.1 Training Accuracy of SVM Classification method

For NNET algorithm our training accuracy is 77.61%. SVM stands for Support Vector Machine which is one of the most efficient algorithm. As we are getting the values of our dataset from the retinal images we are using neural networks.



```
Anaconda Prompt - python  retrain.py \ --bottleneck_dir=bottlenecks \ --how_...
2018-10-29 22:06:28.496484: Step 130: Cross entropy = 0.551800
2018-10-29 22:06:28.840213: Step 130: Validation accuracy = 79.0% (N=100)
2018-10-29 22:06:32.199655: Step 140: Train accuracy = 77.0%
2018-10-29 22:06:32.199655: Step 140: Cross entropy = 0.500726
2018-10-29 22:06:32.543410: Step 140: Validation accuracy = 82.0% (N=100)
2018-10-29 22:06:35.918436: Step 150: Train accuracy = 78.0%
2018-10-29 22:06:35.918436: Step 150: Cross entropy = 0.485376
2018-10-29 22:06:36.246556: Step 150: Validation accuracy = 78.0% (N=100)
2018-10-29 22:06:39.715352: Step 160: Train accuracy = 86.0%
2018-10-29 22:06:39.715352: Step 160: Cross entropy = 0.424871
2018-10-29 22:06:40.059104: Step 160: Validation accuracy = 80.0% (N=100)
2018-10-29 22:06:43.543547: Step 170: Train accuracy = 65.0%
2018-10-29 22:06:43.543547: Step 170: Cross entropy = 0.566024
2018-10-29 22:06:43.887275: Step 170: Validation accuracy = 79.0% (N=100)
2018-10-29 22:06:47.356068: Step 180: Train accuracy = 77.0%
2018-10-29 22:06:47.356068: Step 180: Cross entropy = 0.505961
2018-10-29 22:06:47.684225: Step 180: Validation accuracy = 76.0% (N=100)
2018-10-29 22:06:51.231117: Step 190: Train accuracy = 67.0%
2018-10-29 22:06:51.231117: Step 190: Cross entropy = 0.571592
2018-10-29 22:06:51.590523: Step 190: Validation accuracy = 79.0% (N=100)
2018-10-29 22:06:55.184292: Step 200: Train accuracy = 69.0%
2018-10-29 22:06:55.184292: Step 200: Cross entropy = 0.523138
2018-10-29 22:06:55.528072: Step 200: Validation accuracy = 79.0% (N=100)
2018-10-29 22:06:59.059367: Step 210: Train accuracy = 78.0%
2018-10-29 22:06:59.059367: Step 210: Cross entropy = 0.485691
2018-10-29 22:06:59.403095: Step 210: Validation accuracy = 73.0% (N=100)
2018-10-29 22:07:02.778163: Step 220: Train accuracy = 79.0%
2018-10-29 22:07:02.778163: Step 220: Cross entropy = 0.482059
2018-10-29 22:07:03.106277: Step 220: Validation accuracy = 79.0% (N=100)
2018-10-29 22:07:06.512564: Step 230: Train accuracy = 76.0%
2018-10-29 22:07:06.512564: Step 230: Cross entropy = 0.492142
2018-10-29 22:07:06.840692: Step 230: Validation accuracy = 83.0% (N=100)
2018-10-29 22:07:10.200135: Step 240: Train accuracy = 81.0%
2018-10-29 22:07:10.200135: Step 240: Cross entropy = 0.457531
2018-10-29 22:07:10.528241: Step 240: Validation accuracy = 70.0% (N=100)
2018-10-29 22:07:13.965780: Step 250: Train accuracy = 78.0%
2018-10-29 22:07:13.965780: Step 250: Cross entropy = 0.453401
2018-10-29 22:07:14.293918: Step 250: Validation accuracy = 78.0% (N=100)
2018-10-29 22:07:17.684580: Step 260: Train accuracy = 78.0%
2018-10-29 22:07:17.684580: Step 260: Cross entropy = 0.475277
2018-10-29 22:07:18.028338: Step 260: Validation accuracy = 76.0% (N=100)
2018-10-29 22:07:21.419008: Step 270: Train accuracy = 79.0%
2018-10-29 22:07:21.419008: Step 270: Cross entropy = 0.453420
2018-10-29 22:07:21.747127: Step 270: Validation accuracy = 81.0% (N=100)
2018-10-29 22:07:25.153426: Step 280: Train accuracy = 78.0%
2018-10-29 22:07:25.153426: Step 280: Cross entropy = 0.466165
2018-10-29 22:07:25.481577: Step 280: Validation accuracy = 82.0% (N=100)
2018-10-29 22:07:28.872223: Step 290: Train accuracy = 74.0%
2018-10-29 22:07:28.872223: Step 290: Cross entropy = 0.471355
2018-10-29 22:07:29.215979: Step 290: Validation accuracy = 75.0% (N=100)
2018-10-29 22:07:32.591040: Step 300: Train accuracy = 80.0%
2018-10-29 22:07:32.591040: Step 300: Cross entropy = 0.444187
2018-10-29 22:07:32.934770: Step 300: Validation accuracy = 79.0% (N=100)
2018-10-29 22:07:36.325436: Step 310: Train accuracy = 78.0%
2018-10-29 22:07:36.325436: Step 310: Cross entropy = 0.503491
2018-10-29 22:07:36.669218: Step 310: Validation accuracy = 81.0% (N=100)
_
```

Fig.8.1 :  Training the model


Anaconda Prompt - python retrain.py \ --bottleneck_dir=bottlenecks \ --how_...

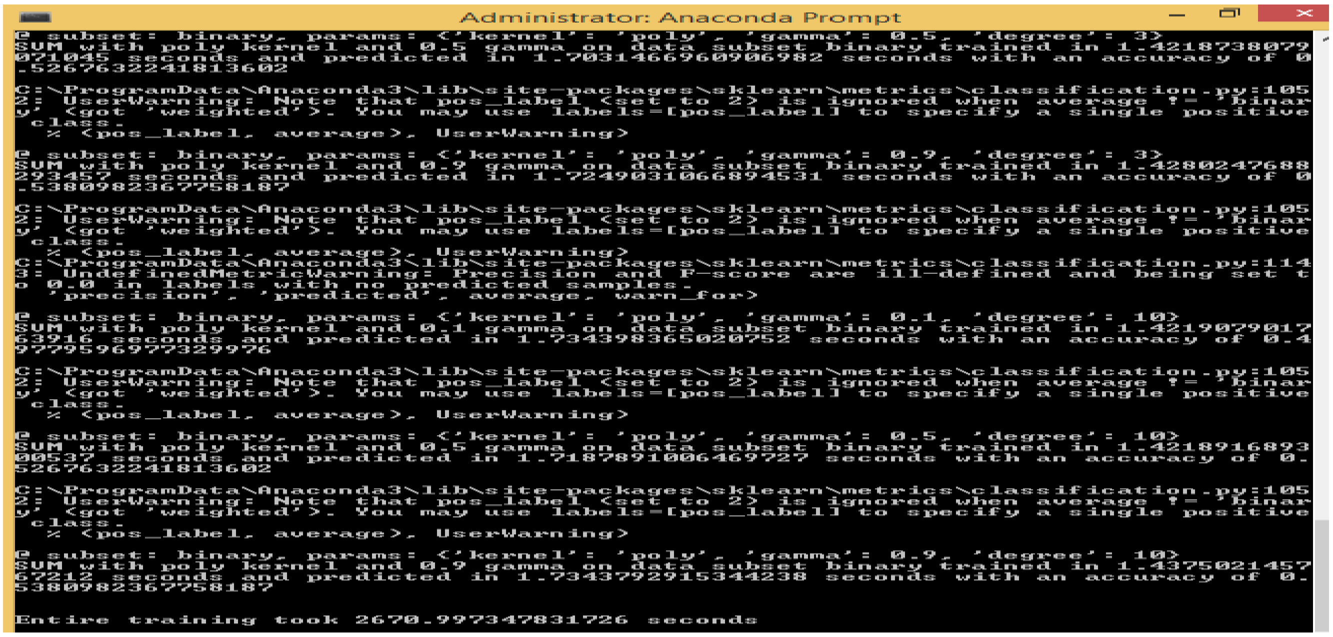
```
2018-10-29 22:04:37.260047: W tensorflow/core/framework/op_def_util.cc:355] Op
atchNormWithGlobalNormalization is deprecated. It will cease to work in GraphDe
 version 9. Use tf.nn.batch_normalization().
Looking for images in 'diseased'
Looking for images in 'notdiseased'
100 bottleneck files created.
200 bottleneck files created.
300 bottleneck files created.
400 bottleneck files created.
500 bottleneck files created.
600 bottleneck files created.
700 bottleneck files created.
800 bottleneck files created.
900 bottleneck files created.
1000 bottleneck files created.
1100 bottleneck files created.
1200 bottleneck files created.
1300 bottleneck files created.
1400 bottleneck files created.
1500 bottleneck files created.
1600 bottleneck files created.
1700 bottleneck files created.
1800 bottleneck files created.
1900 bottleneck files created.
2000 bottleneck files created.
2100 bottleneck files created.
2200 bottleneck files created.
2300 bottleneck files created.
2400 bottleneck files created.
2500 bottleneck files created.
2600 bottleneck files created.
2700 bottleneck files created.
2800 bottleneck files created.
2900 bottleneck files created.
3000 bottleneck files created.
3100 bottleneck files created.
3200 bottleneck files created.
3300 bottleneck files created.
3400 bottleneck files created.
3500 bottleneck files created.
3600 bottleneck files created.
3700 bottleneck files created.
3800 bottleneck files created.
3900 bottleneck files created.
4000 bottleneck files created.
4100 bottleneck files created.
WARNING:tensorflow:From retrain.py:736: softmax_cross_entropy_with_logits (from
tensorflow.python.ops.nn_ops) is deprecated and will be removed in a future ver
ion.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.
```

Fig. 8.2: Retraining the model

From the plot of loss, we can see that the model has comparable performance on both train and test datasets. If these parallel plots start to depart consistently, It might be assign to stop training at an earlier epoch.

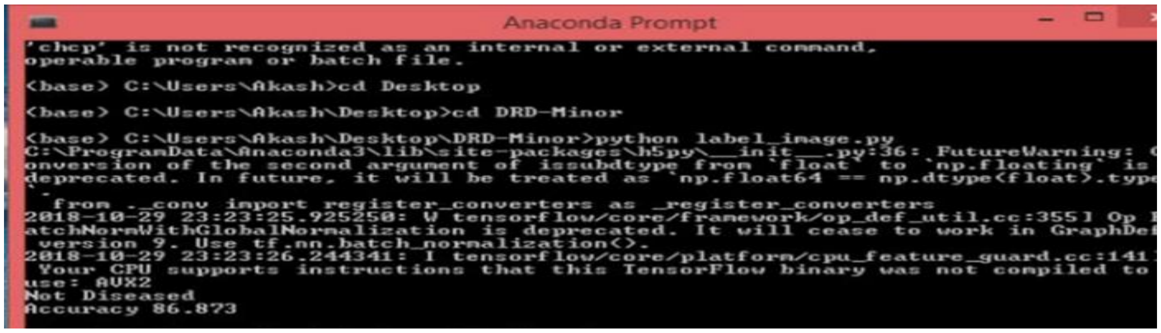
If the lines of train-test loss seem to converge to the same value and are close at the end, then the classifier has high bias. If on the other hand the lines are quite far apart, and then we have a low trainings et error but high validation error, then your classifier has too high variance.

From these we can conclude that our train – test loss model training set loss is low and our test set error is not too high. So, from this it can be said that we have a good train-test accuracy model.



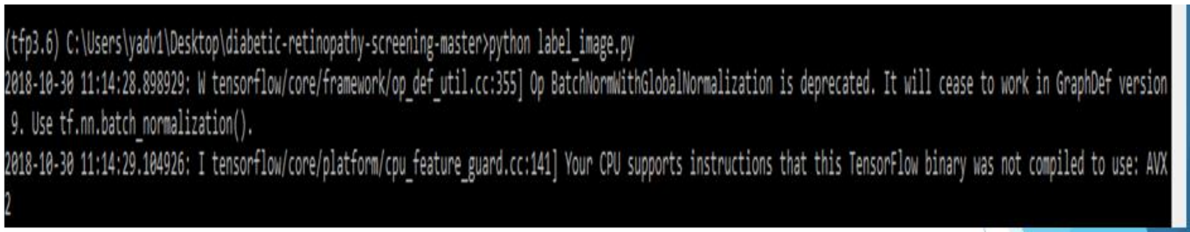
```
@ subset: binary, params: {'kernel': 'poly', 'gamma': 0.5, 'degree': 3}
SVM with poly kernel and 0.5 gamma on data subset binary trained in 1.4218738079
071045 seconds and predicted in 1.7031466960906982 seconds with an accuracy of 0
.5267632241813602

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:105
2: UserWarning: Note that pos_label (set to 2) is ignored when average != 'binar
y' (got 'weighted'). You may use labels=[pos_label] to specify a single positive
 class.
  % (pos_label, average), UserWarning)

@ subset: binary, params: {'kernel': 'poly', 'gamma': 0.9, 'degree': 3}
SVM with poly kernel and 0.9 gamma on data subset binary trained in 1.4280247688
293457 seconds and predicted in 1.7249031066894531 seconds with an accuracy of 0
.5380982367758187

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:105
2: UserWarning: Note that pos_label (set to 2) is ignored when average != 'binar
y' (got 'weighted'). You may use labels=[pos_label] to specify a single positive
 class.
  % (pos_label, average), UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:114
3: UndefinedMetricWarning: Precision and F-score are ill-defined and being set t
o 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)

@ subset: binary, params: {'kernel': 'poly', 'gamma': 0.1, 'degree': 10}
SVM with poly kernel and 0.1 gamma on data subset binary trained in 1.4219079017
63916 seconds and predicted in 1.734398365020752 seconds with an accuracy of 0.4
977959697732997б

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:105
2: UserWarning: Note that pos_label (set to 2) is ignored when average != 'binar
y' (got 'weighted'). You may use labels=[pos_label] to specify a single positive
 class.
  % (pos_label, average), UserWarning)

@ subset: binary, params: {'kernel': 'poly', 'gamma': 0.5, 'degree': 10}
SVM with poly kernel and 0.5 gamma on data subset binary trained in 1.4218916893
00537 seconds and predicted in 1.7187891006469727 seconds with an accuracy of 0.
5267632241813602

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:105
2: UserWarning: Note that pos_label (set to 2) is ignored when average != 'binar
y' (got 'weighted'). You may use labels=[pos_label] to specify a single positive
 class.
  % (pos_label, average), UserWarning)

@ subset: binary, params: {'kernel': 'poly', 'gamma': 0.9, 'degree': 10}
SVM with poly kernel and 0.9 gamma on data subset binary trained in 1.4375021457
67212 seconds and predicted in 1.7343792915344238 seconds with an accuracy of 0.
5380982367758187

Entire training took 2670.997347831726 seconds
```

Fig 8.3 Total Training time



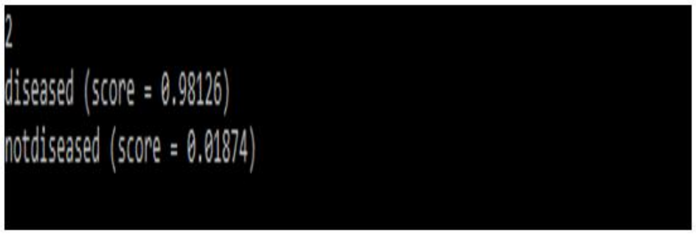
```
'chcp' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\Akash>cd Desktop

(base) C:\Users\Akash\Desktop>cd DRD-Minor

(base) C:\Users\Akash\Desktop\DRD-Minor>python label_image.py
C:\ProgramData\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: C
onversion of the second argument of issubdtype from `float` to `np.floating` is
deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type
`.
  from ._conv import register_converters as _register_converters
2018-10-29 23:23:25.925250: W tensorflow/core/framework/op_def_util.cc:355] Op B
atchNormWithGlobalNormalization is deprecated. It will cease to work in GraphDef
 version 9. Use tf.nn.batch_normalization().
2018-10-29 23:23:26.244341: I tensorflow/core/platform/cpu_feature_guard.cc:141]
 Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2
Not Diseased
Accuracy 86.873
```

Fig 8.4: Prediction of the presence of Disease

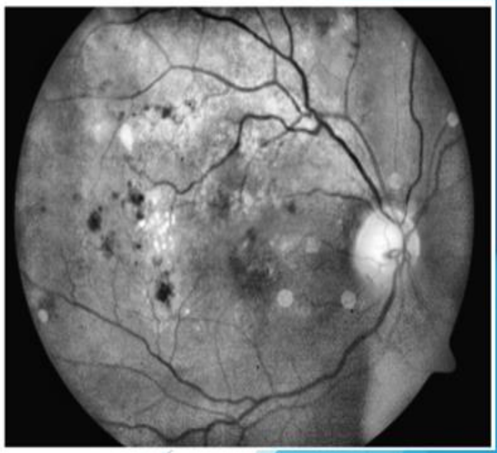


```
(tfp3.6) C:\Users\yadv1\Desktop\diabetic-retinopathy-screening-master>python label_image.py
2018-10-30 11:14:28.898929: W tensorflow/core/framework/op_def_util.cc:355] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in GraphDef version
 9. Use tf.nn.batch_normalization().
2018-10-30 11:14:29.104926: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX
2
```

## RESULT



```
2
diseased (score = 0.98126)
notdiseased (score = 0.01874)
```



# CHAPTER 9:
# CONCLUSION

This chapter contains the difficulties, future works and concluding remarks, which will give the summary of our thesis work and also give the indication of our future plan with our thesis project.

## 9.1 Difficulties and Challenges

There are many difficulties we faced while working. First of all, there's a lot more to do before an algorithm like this can be used widely. For example, as a classifier algorithm SVM was remarkable in percentage but we had to work more on binary classification. Secondly, if we could manage more of our training data, we could train our algorithm more to achieve more accuracy. Furthermore, we also faced some problems while choosing algorithms. It was quite difficult for us to choose some specific machine learning algorithms that would give accurate classification of the disease. In addition, we used simple techniques for feature selection and scaling and possibly we could arrive at better results by introducing more complex techniques for selecting and generating features. We looked at small subspaces for model parameters. Possibility there be other parameter spaces that would yield better performing models. For these few amounts of data in our dataset we faced difficulties in implementing the classification.

## 9.2 Future Work

For any research, there is always room for improvement. Ours is not an exception of that. We have found some areas where this system can be improvised.

1.**Work on more Categories**: This can be improvised with a lot more categorized such as according to ages, genders, background studies, working facilities and soon. As an example, A matured man from the IT background has different eye condition that a matured women from

2.**Work on more classes**: As we working on only two classes whether it is good or bad. In future we are going to add more classes like low, moderate, severe condition. In this way patients can know about their condition more accurate.

3.**Different Algorithms**: CRF (Conditional Random Field), maximum entropy and other probabilistic graphical model can also be used to train our dataset in order to improve the algorithm.

4.**More Analysis:** To achieve more accuracy we could use more dataset. If we use huge amount of dataset, machine will train more and it would give us more accurate prediction and accuracy.

5.**Hardware Implementation**: A hardware product can be the best solution for patient. So, we are looking forward to build a hardware system where we can use our model to implement results on diabetic patients easily. We can then input the data of the patient and wait for the machine to create a new prescription integrated with Doctor's suggestion.

## 9.3 Concluding Remarks

We have tried to construct an ensemble to predict if a patient has diabetic retinopathy using features from retinal photos. After training and testing the model the accuracy we get is quite similar. For both sets NNET is providing higher accuracy rate for predicting DR. Despite the short coming sin reaching good performance results, this work provided a mean to make use and test multiple machine learning algorithms and try to arrive to ensemble models that would outperform mind visual learners. It also allows exploring a little features election, feature generation, parameters election and ensemble selection problems and experiences the constraints in computation time when looking for possible candidate models in high combinatorial spaces, even for a small dataset as the one used. The structure of our research has been built in such a way that with proper dataset and minor alter nation it can work to classify the disease in any number of categories.

# References

[1] Gandhi M. and Dhanasekaran R, Diagnosis of Diabetic Retinopathy Using Morphological Process and SVM Classifier, IEEE International conference on Communication and Signal Processing,2013, India,pp:873-877.

[2] LiT, Meindert N, Reinhardt JM, GarvinMK,Abramoff MD Splat Feature Classification with Application to Retinal Hemorrhage Detection in Fundus Images,IEEE Transactions on Medical Imaging,2013, pp:364-375.

[3] Yau JW, Rogers SL, KawasakiR, Lamoureux EL,Kowalski JW, Bek T,etal.Global prevalence and major risk factors of diabetic retinopathy. Diabetes Care,2012, pp:556–564

[4] Boser B, Guyon I.G,Vapnik V.,″A Training Algorithm for Optimal Margin Classifiers″,Proc. Fifth Ann. Workshop Computational Learning Theory,1992, pp. 144-152.

[5] Mitchell .T , Machine Learning, McGraHill.ISBN0-07-042807-7., McGraw-Hill,Inc.NewYork, NY, USA.Published on March 1, 1997, pp:204-214.

# Appendix 1

```
import time

import pandas

import pickle

import numpy as np

from functools import partial

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, precision_recall_fscore_support

from sklearn import svm

from xgboost import XGBClassifier

from sklearn.neighbors import KNeighborsClassifier


"""

  See main function at bottom for usage. Returns a data dictionary where the

  keys are 'full', 'limited', and 'binary'. The value for each key is another

  dictionary where the keys of that dictionary are 'X_train', 'X_test',

  'y_train', and 'y_test'. The values for these nested dictionaries are numpy

  matrices/vectors for the properly partitioned training/testing data/labels.

"""

def prepareData():
  # Read in reduced data and make copies that will be mutated.

  full_PCA_data = pickle.load(open('reduced_data.bin', 'rb'), encoding='ISO-
8859-1')

  limited_PCA_data = full_PCA_data

  binary_PCA_data = full_PCA_data



  # Read in labels from CSV for training.

  df = pandas.read_csv('trainLabels.csv')

  temp_labels = df.as_matrix()

  full_labels = df.as_matrix()[:,1]

  limited_labels = full_labels

  binary_labels = full_labels



  # Creates 'limited' and 'binary' data which have fewer '0' class objects so

  # that the number of examples for each class is more balanced...'limited' has

  # a number of '0' class objects equal to the number of class '2' objects

  # while 'binary' has the same number of '0' class objects and additionally

  # removes all '1', '3', and '4' class objects.
```

# Appendix 2

```
# Builds sets of row indices to be included in the final data subsets for
# 'limited' and 'binary'.
i, counter, limited, binary = 0, 0, set(), set()
for i, l in enumerate(temp_labels):
  if counter < 5292 and l[1] == 0:
    limited.add(i)
    binary.add(i)
    counter += 1
  elif l[1] == 2:
    limited.add(i)
    binary.add(i)
  elif l[1] != 0:
    limited.add(i)


# Removes row indices that do not belong in the 'limited' data/label subsets.
i, j = 0, 0
while len(limited_PCA_data) > len(limited):
  if j not in limited:
```

```
  limited_PCA_data = np.delete(limited_PCA_data, i, 0)

      limited_labels = np.delete(limited_labels, i, 0)

      i -= 1

    i += 1

    j += 1



  # Removes row indices that do not belong in the 'binary' data/label subsets.

  i, j = 0, 0

  while len(binary_PCA_data) > len(binary):

    if j not in binary:

      binary_PCA_data = np.delete(binary_PCA_data, i, 0)

      binary_labels = np.delete(binary_labels, i, 0)

      i -= 1

    i += 1

    j += 1



  # 'test_size' and 'seed' for splitting so that splits are the same for each
run.

  seed = 7

  size = 0.2
  # Data and their appropriate labels for each training configuration.

  configurations = [

    ('full', full_PCA_data, full_labels),

    #('limited', limited_PCA_data, limited_labels),

    #('binary', binary_PCA_data, binary_labels)

  ]

  # Initialized data dictionary that is returned by this funciton.

  data = {

    'full': {} #, 'limited': {}, 'binary': {}

  }

  # Populate data dictionary with appropriate data/label subsets.

  for c in configurations:

    X_train, X_test, y_train, y_test = train_test_split(*c[1:], test_size=size,
random_state=seed)

    data[c[0]]['X_train'] = X_train

    data[c[0]]['X_test'] = X_test

    data[c[0]]['y_train'] = y_train

    data[c[0]]['y_test'] = y_test
  # Convert labels from strings to ints.

  for _, v in data.items():

    v['y_train'], v['y_test'] = v['y_train'].astype(int),
v['y_test'].astype(int)



  return data



"""

  See main function at bottom for usage. Takes in a key for the data/label

  subsets and a dictionary of model parameters, then trains and predicts with

  each type of model.

"""
```

# Appendix 3

```
def trainModels(data, params):

  f = open(out_file, 'w')

  def trainKNN(data_subset):

    f.write('\nTraining KNN:'+'\n')



    X_train = data[data_subset]['X_train']

    X_test = data[data_subset]['X_test']

    y_train = data[data_subset]['y_train']

    y_test = data[data_subset]['y_test']



    for p in params['knn']:

      header = "@ subset: {0}, params: {1}".format(data_subset, p)

      f.write('\n'+header+'\n')



      n_neighbors = p['n_neighbors']



      model = KNeighborsClassifier(n_neighbors=n_neighbors)
  start = time.time()

      model.fit(X_train, y_train)

      elapsed_train = time.time() - start



      y_pred = model.predict(X_test).astype(int)

      elapsed_predict = time.time() - start



      accuracy = accuracy_score(y_test, y_pred)

      precision, recall, fscore, support = 
precision_recall_fscore_support(y_test, y_pred, pos_label=2, 
average='weighted')



      print("\n{5}\nKNN with {0} neighbors on data subset {1} trained in {2} 
seconds and predicted in {3} seconds with an accuracy of 
{4}\n".format(n_neighbors, data_subset, elapsed_train, elapsed_predict, 
accuracy, header))



      f.write(str(elapsed_train) + ', ' + str(elapsed_predict) + str(accuracy)+ 
', ' + str(precision)+ ', ' + str(recall )+ ', ' + str(fscore )+ ', ' + 
str(support))



  def trainSVM(data_subset):

    f.write('\nTraining SVM:'+'\n')

    X_train = data[data_subset]['X_train']

    X_test = data[data_subset]['X_test']
```

```
y_train = data[data_subset]['y_train']

    y_test = data[data_subset]['y_test']



    for p in params['svm']:

    header = "@ subset: {0}, params: {1}".format(data_subset, p)

      f.write('\n'+header+'\n')

      kernel = p['kernel']

      gamma = p['gamma']

      model = svm.SVC(kernel=kernel, gamma=gamma)

      start = time.time()

      model.fit(X_train, y_train)



 elapsed_train = time.time() - start

      y_pred = model.predict(X_test).astype(int)

      elapsed_predict = time.time() - start

      accuracy = accuracy_score(y_test, y_pred)

      precision, recall, fscore, support =
precision_recall_fscore_support(y_test, y_pred, pos_label=2,
average='weighted')

      print("\n{5}\nSVM with {0} kernel and {6} gamma on data subset {1}
trained in {2} seconds and predicted in {3} seconds with an accuracy of
{4}\n".format(kernel, data_subset, elapsed_train, elapsed_predict, accuracy,
header, gamma))
```

```
      f.write(str(elapsed_train) + ', ' + str(elapsed_predict) + str(accuracy)+

', ' + str(precision)+ ', ' + str(recall )+ ', ' + str(fscore )+ ', ' +
str(support))

  # Iterate over all the data/label subsets and then train and predict with

  # each type of model.

  for k, v in data.items():

    print("\nTraining {0} subset on KNN classifier\n".format(k))

    trainKNN(k)

    print("\nTraining {0} subset on SVM classifier\n".format(k))

    trainSVM(k)

  f.close()

if __name__ == '__main__':

  # Dictionary of parameters keyed by the type of model. Values are lists of

  # parameter sets.

  params = {

      'knn': [{'n_neighbors': 3}, {'n_neighbors': 5},

        {'n_neighbors': 10}, {'n_neighbors': 25}, {'n_neighbors': 50},
{'n_neighbors':100}],

    'svm': [

      {'kernel': 'rbf', 'gamma': 0.1},

     {'kernel': 'rbf', 'gamma': 1.0},
     {'kernel': 'linear','gamma': 0.1},
{'kernel': 'linear','gamma': 1.0},
     {'kernel': 'poly', 'gamma': 0.1},
      {'kernel': 'poly', 'gamma': 1.0}
    ]
  }
  out_file = 'results.dat'
  start = time.time()
  data = prepareData()
  trainModels(data, params)
  print("\nEntire training took {0} seconds".format(time.time() - start))
```

# APPENDIX 4

```
# change this as you see fit

image_path = 'train/114_right.jpeg'

# Read in the image_data

image_data = tf.gfile.GFile(image_path, 'rb').read()



# Loads label file, strips off carriage return

label_lines = [line.rstrip() for line

                   in tf.gfile.GFile("tf_files/retrained_labels.txt")]

# Unpersists graph from file

with tf.gfile.GFile("tf_files/retrained_graph.pb", 'rb') as f:

    graph_def = tf.GraphDef()

    graph_def.ParseFromString(f.read())

    _ = tf.import_graph_def(graph_def, name='')



with tf.Session() as sess:

    # Feed the image_data as input to the graph and get first prediction

    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

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

    # Sort to show labels of first prediction in order of confidence

    #top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]

    #for node_id in top_k:

    human_string = label_lines[0]

    score = predictions[0][0]

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

human_string = label_lines[1]

score = predictions[0][1]

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

human_string = label_lines[2]

score = predictions[0][2]

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

human_string = label_lines[3]

score = predictions[0][3]

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

#human_string = label_lines[4]

#score = predictions[0][4]

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

# APPENDIX 5

```
import cv2

import numpy as np

def adjust_gamma(image, gamma=1.0):



  table = np.array([((i / 255.0) ** gamma) * 255

       for i in np.arange(0, 256)]).astype("uint8")


   return cv2.LUT(image, table)

def extract_ma(image):

    r,g,b=cv2.split(image)

    comp=255-g

    clahe = cv2.createCLAHE(clipLimit=5.0, tileGridSize=(8,8))

    histe=clahe.apply(comp)

    adjustImage = adjust_gamma(histe,gamma=3)

    comp = 255-adjustImage

    J =  adjust_gamma(comp,gamma=4)

    J = 255-J

    J = adjust_gamma(J,gamma=4)


    K=np.ones((11,11),np.float32)

    L = cv2.filter2D(J,-1,K)


    ret3,thresh2 = cv2.threshold(L,125,255,cv2.THRESH_BINARY|cv2.THRESH_OTSU)

    kernel2=np.ones((9,9),np.uint8)

    tophat = cv2.morphologyEx(thresh2, cv2.MORPH_TOPHAT, kernel2)

    kernel3=np.ones((7,7),np.uint8)
    opening = cv2.morphologyEx(tophat, cv2.MORPH_OPEN, kernel3)
    return opening

if __name__ == "__main__":

    import os, os.path
    from os.path import isfile, join



    path=os.getcwd()
    dirs = os.listdir(path)
    onlyfiles = [f for f in dirs if isfile(join(path, f))]
    onlyimages = [f for f in onlyfiles if f.endswith('.jpeg')]
    i = 0
    for item in onlyimages:
        fullpath = os.path.join(path,item)
 #corrected
        if os.path.isfile(fullpath):
            fundus = cv2.imread(fullpath)
            f, e = os.path.splitext(fullpath)
            bloodvessel = extract_ma(fundus)
            cv2.imwrite(f+'_clahe.jpeg',bloodvessel)
        i+=1
        print("images left: "+str(len(onlyimages)-i))
```