International Journal of Research in Science and Technology http://www.ijrst.com

(IJRST) 2017, Vol. No. 7, Issue No. III, Jul-Sep **e-ISSN: 2249-0604, p-ISSN: 2454-180X**

EMPLOYABILITY OF MACHINE LEARNING TOOLS IN DESIGNING A PLAY/PAUSE MODEL BASED ON EYE GAZE IN INCREASING THE EFFICACY OF ONLINE VIDEO LEARNING

Aahan Gupta
Student Researcher

1. INTRODUCTION

Online video education is steadily becoming more important. With the rise in use of internet , student tend to study more using the contents available online such as ebooks and video lectures. Online lectures are currently conceivable with the approach of video correspondences. The improvement of the online address makes it conceivable with the goal that the instructor and understudy never again must be in a similar region to educate and learn, individually. Students get a chance to learn from the best professors

2. LITERATURE SURVEY

Eye detection and eye tracking are not a new area of research. Various researches have been done and many libraries exist for the same. Similarly , various works have been done in machine learning . A thorough literature survey was the backbone of our project and was crucial in deciding which technique should we use in our project.

Depiction of the standard algorithm

1. k initial "means" (in this case k=3) are ‘haphazardly produced’ within the data domain (shown in color).

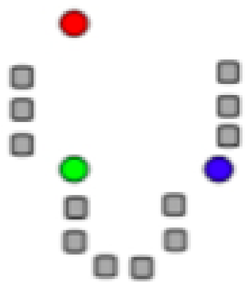


Fig. 2.2

2. k clusters are made by connecting each observation with the closest mean. The partitions here represent the Voronoi diagram generated by the means.

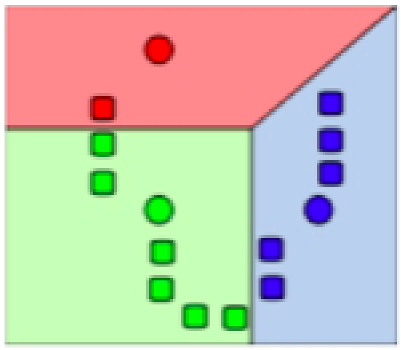
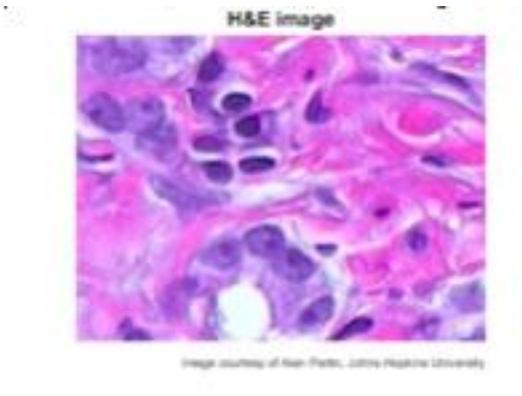


Fig. 2.3

3. The new mean is the centroid of each k cluster.



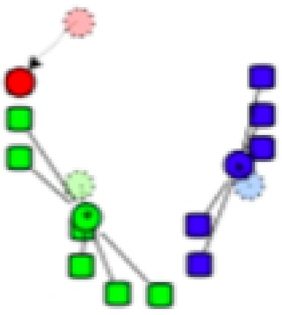


Fig. 2.4

4. Steps 2, 3 are recurred till the time convergence is attained.

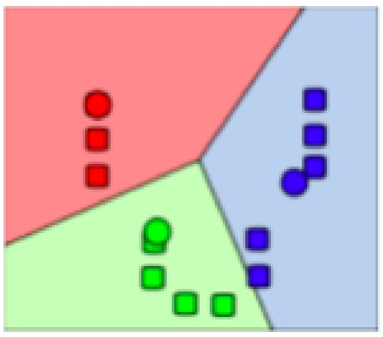


Fig. 2.5

Since it is a heuristic computation, there is absolutely no qualification that it'll combine to the worldwide ideal, and the results may rely after the underlying teams. As the computation is normally quick, it is basic to perform it various circumstances with various start conditions. Regardless, in the most pessimistic circumstance, k-means can be decrease back again to unite: specifically it's been confirmed that there can be found certain point models, even in 2 measurements, which k-implies will take exponential time, that is 2Ω(n), to converge.

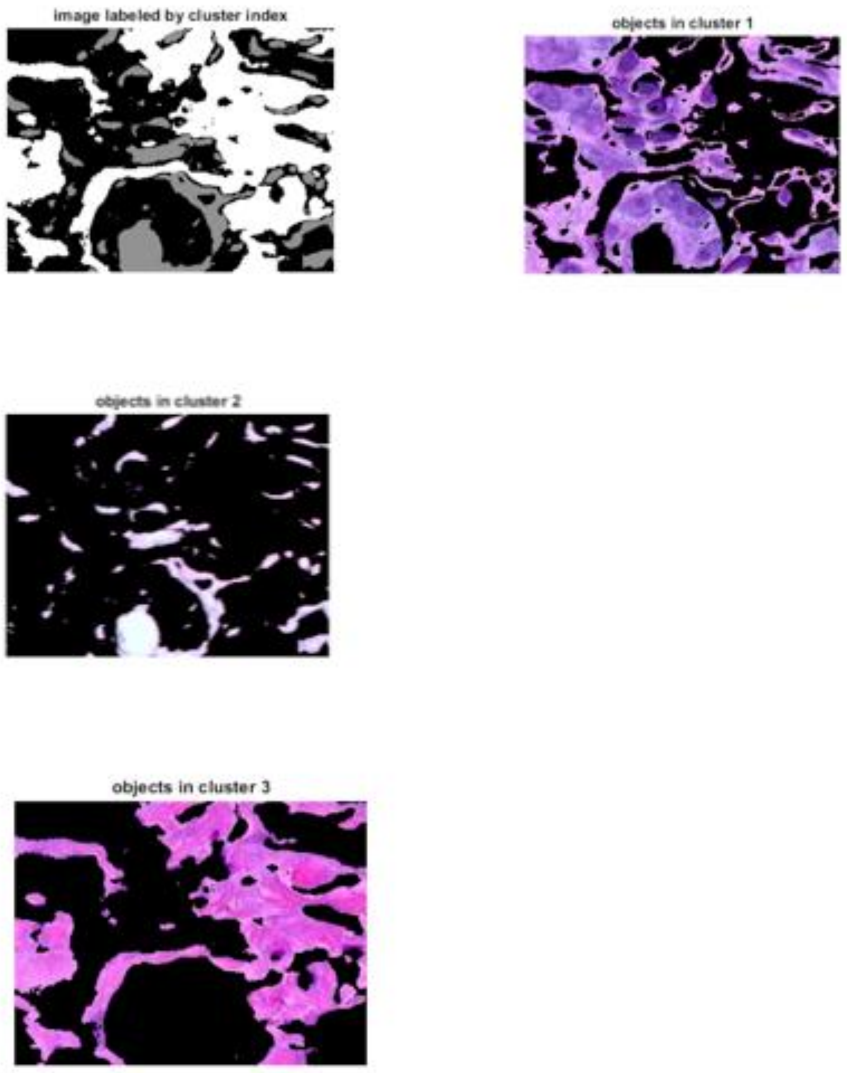


Fig. 2.6

How I planned to use k-clustering :

For detecting eye gaze we planned to use the separation between the sclera and the pupil.
For this eye was clustered into 3 parts sclera cluster, pupil cluster and skin cluster on the basis of colour.

Direction could be divided into 3 wide classes:

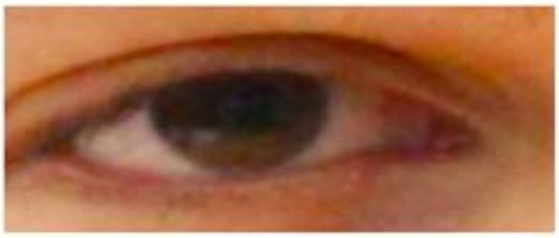
1. Front : If the separation between the sclera centroid and pupil centroid is minimal.

2. Left : if the separation between the sclera centroid and pupil centroid is large and pupil centroid lies to left.

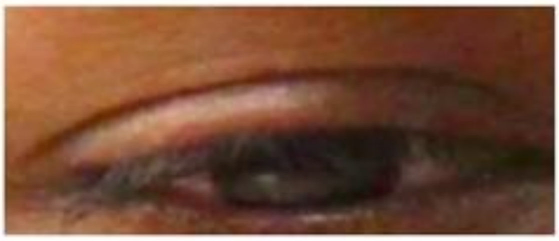
3. Right: if the separation between the sclera centroid and pupil centroid is large and pupil centroid lies to right.

4. Up: if the separation between the sclera centroid and pupil centroid is small and pupil centroid lies to upwards region.

Favourable classes :

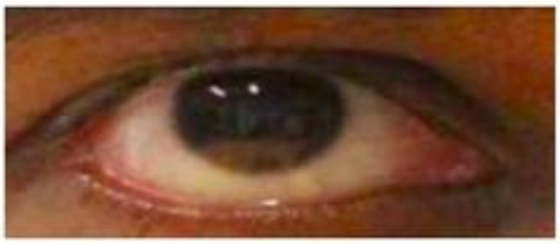


1) Front (**Fig. 2.11**)



2) Down (**Fig. 2.12**)

Unfavourable classes :



1) Up (**Fig. 2.13**)

2) Right (**Fig. 2.14**)

3) Left (**Fig. 2.15**)

Function in Matlab :

[idx,C,sumd,D] = kmeans(X,k);

is the matlab function for k-means clustering.

Problems Faced while Using K-clustering :

The cluster formed were not so accurate because of the following reasons :

1. The feature used for making clusters was the pixel colour and due to the image being through the webcam the image was in a very low resolution and its quality wasn't upto the mark.
2. Many regions of the image were dark thus leading to wrong clusters being formed.

MultiClass Classification :

The next idea for eye gaze detection was to use MultiClass Classification. Eyes were divided into classes namely :

1.up
2.down
3.left
4.right
5.front
If the test input belonged to class 2 or 5 then the video was supposed to play else it was supposed to be in pause state.

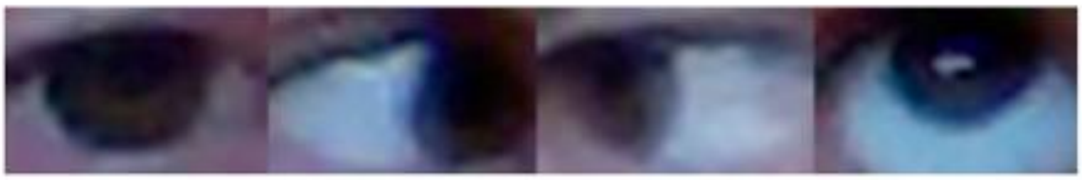


Fig. 2.7: One example of each labeling taken from the training set

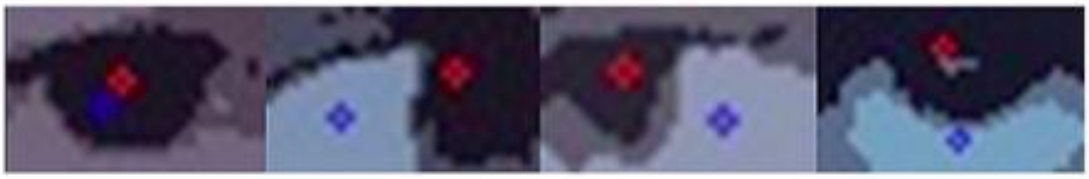
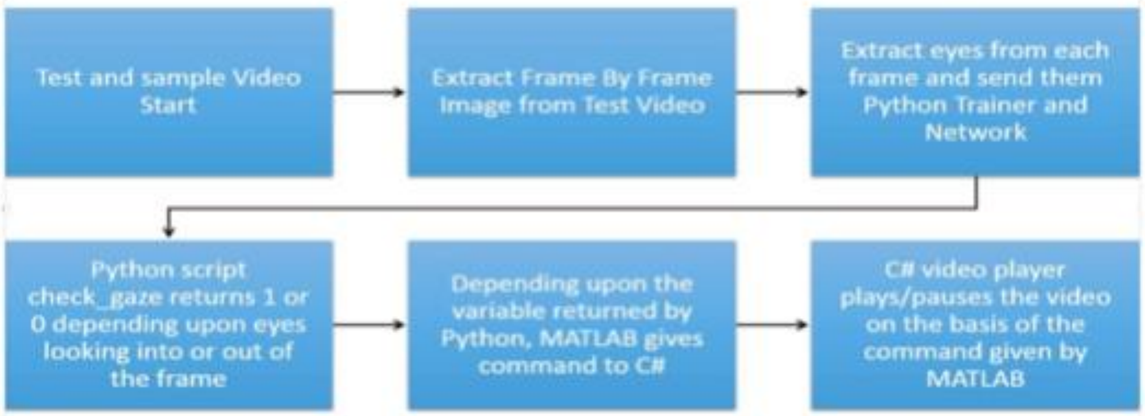


Fig. 2.8: Eye images having undergone 3-means clustering with type-1 features marked

3. METHODOLOGY

The actual implementation of the project was done in the following manner:

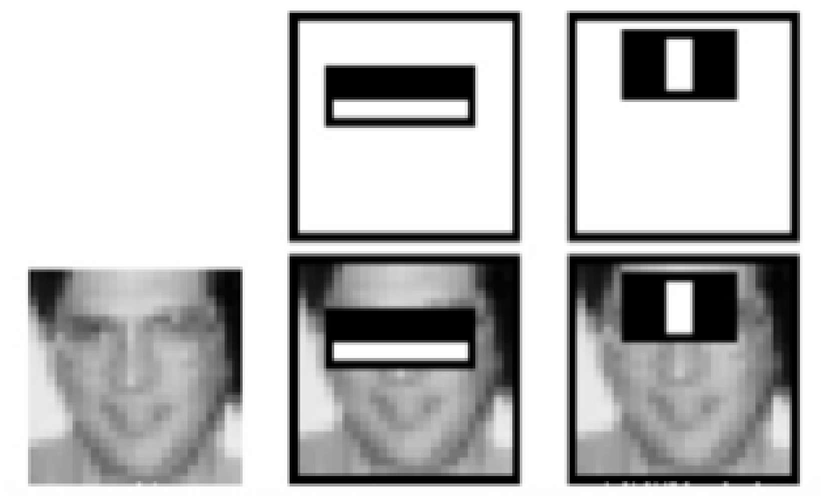
Flowchart depicting the overall process flow of the the program (Fig.3.1)


Test and sample Video Start
Extract Frame By Frame Image from Test Video
Extract eyes from each frame and send them Python Trainer and Network
Python script check_gaze returns 1 or 0 depending upon eyes looking into or out of the frame
Depending upon the variable returned by Python, MATLAB gives command to C#
C# video player plays/pauses the video on the basis of the command given by MATLAB

Face Detection and Extraction of facial features

The face parts are detected using the function buildDetector().

The input to this function are thresholdFaces , thresholdParts and stdsize. This outputs a detector . This function uses vision.CascadeObjectDetector() to detect the face parts. vision.CascadeObjectDetector() is an inbuilt function in the in the Image Processing Toolbox which is implemented using the Haar-like features.



Haar-like features superimposed on a human face (Fig.3.2)

Every human face share some comparable properties. This information is utilized to build certain highlights known as Haar Features.

The features that are like a human face are:

- The eyes area is darker than the upper-cheeks.
- The nose connect locale is brighter than the eyes.

That is valuable space learning:

- Location - Size: eyes and nose connect area

- Value: darker/brighter

Fig .3.3

The four highlights connected in this calculation are connected onto a face and appeared on the left. Rectangle features:

- Value = Σ (pixels in black area) - Σ (pixels in white area)
- Three sorts: two-, three-, four-rectangles, Viola and Jones utilized two-rectangle highlights

- For instance: the distinction in splendor between the white and dark rectangles over a particular region

- Each component is identified with an uncommon area in the sub-window

Notwithstanding, with the utilization of a picture portrayal called the necessary picture, rectangular highlights can be assessed in steady time, which gives them an extensive speed advantage over their more modern relatives. Since each rectangular territory in a component is constantly contiguous no less than one other rectangle, it takes after that any two-rectangle highlight can be processed in six exhibit references, any three-rectangle include in eight,and any four-rectangle include in only ten.

The vital picture at area (x,y), is the whole of the pixels above and to one side of (x,y), comprehensive.

The detector built above is sent to the function detectFaceParts() which with help of the detector and an inbuilt step() function makes bounding boxes around the faces and the face parts ie nose , mouth , right eye and left eye and returns the image with the bounded boxes.

We crop out the left and the right eye , obtained by the above generated boxes and detect the gaze of individual eyes.

The dataset that we used for the extraction purposes was the **Columbia Gaze Dataset**.

Creating Test File For Neural Training :

The eye extracted via matlab is of dimension 470*250 pixels.This image is then resize into an image such that the cropped portion starts from the following dimensions :
Initial Row : 80
Initial Column : 115
Width : 250
Height : 110
The cropped image was then resized to size 20*45 and then converted it to grayscale.

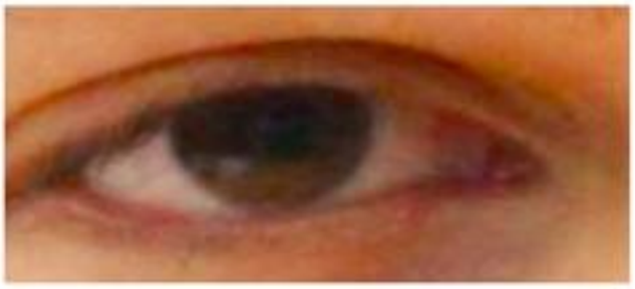


Fig.3.4

turned to grayscale

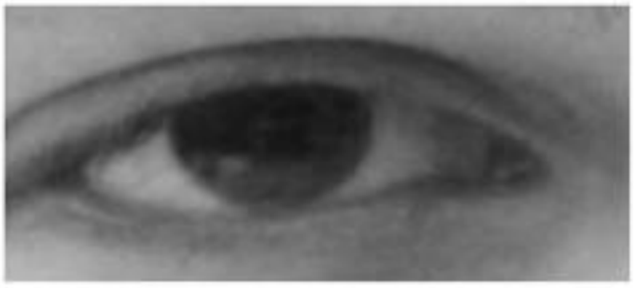


Fig.3.5

The grayscale image was then written out pixel by pixel in a text file where each row represents a single eye and at the end of the row is a variable that tells the image belongs to which class.This file is then used for training the neural network.

Training Using Pybrain

Pybrain is a Neural Network Library of Python.

Its will most likely offer adjustable, easy to-utilize but still capable computations for Machine Learning Jobs and a variety of predefined situations to check and consider your algorithms.PyBrain is brief for Python-Based Encouragement Learning, Artificial Intelligence and Neural Network Library.

While there are a couple of machine learning libraries out there, PyBrain plans to be a simple to-utilize measured library that can be utilized by passage level understudies yet at the same time offers the adaptability and calculations for best in class examine.

What PyBrain can do

PyBrain, as it's composed out name as of now recommends, contains calculations for neural systems, for fortification learning (and the mix of the two), for unsupervised learning, and development. Since the greater part of the present issues manage nonstop state and activity spaces, work approximators (like neural systems) must be utilized to adapt to the huge dimensionality.

We used Pybrain to implement Neural Networks for MultiClass Classification to detect Eye Gaze.
Eye Gaze was basically divided into two broad classes :

1) Into the video

2) Out of the video

Functions From Pybrain :

1) pybrain.datasets.classification.ClassificationDataSet(*inp*, *target=1*, *nb_classes=0*)

inp refers to number of features to classify the sample. target refers to number of output variables. nb_classes refers to the classes in which sample has to be classified.
2) _convertToOneOfMany()

Proselytes the objective classes to a 1-of-k portrayal, holding the old focuses as a field class.

To supply particular limits, set the limits parameter, which comprises of target esteems for non-enrollment and participation.

3) fnn.activate(sample)

This is used to activate the network 'fnn' over the input named 'sample' and the value returned is the target class for the sample.

4) alldata.splitWithProportion(p) :

This function splits the DataSet 'alldata' into two sets one for training and one for validation so that one set has p% of the values and other has (100-p)% values.

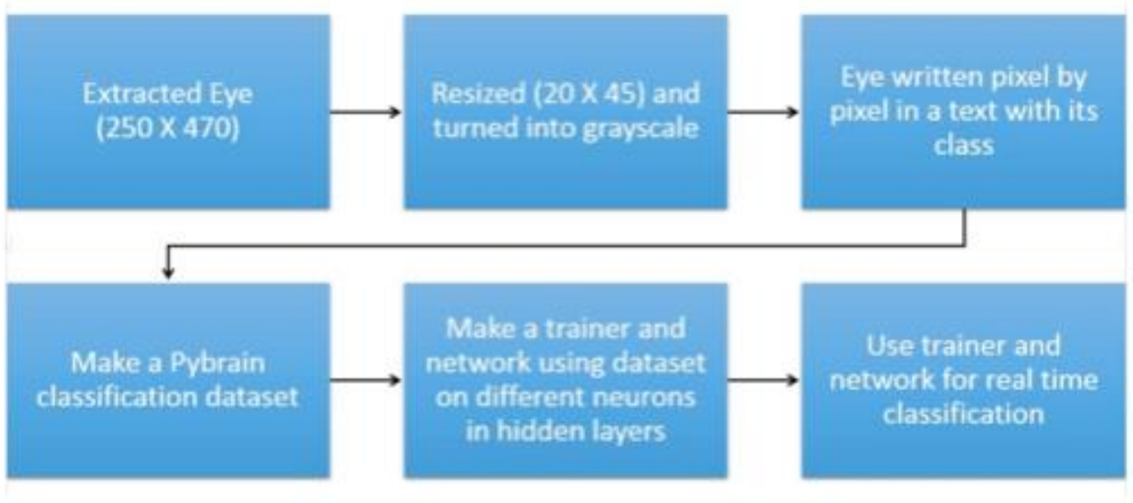
5) buildNetwork(trndata.indim, param1,param2 ,trndata.outdim, outclass=SoftmaxLayer)

The function buildNetwork is used to build a network on the input DataSet with trndata.indim being the number of input features, param1 being number of neurons in hidden layer 1,param2 being number of neuron in hidden layer 2 and trndata.outdim being number of output variables.

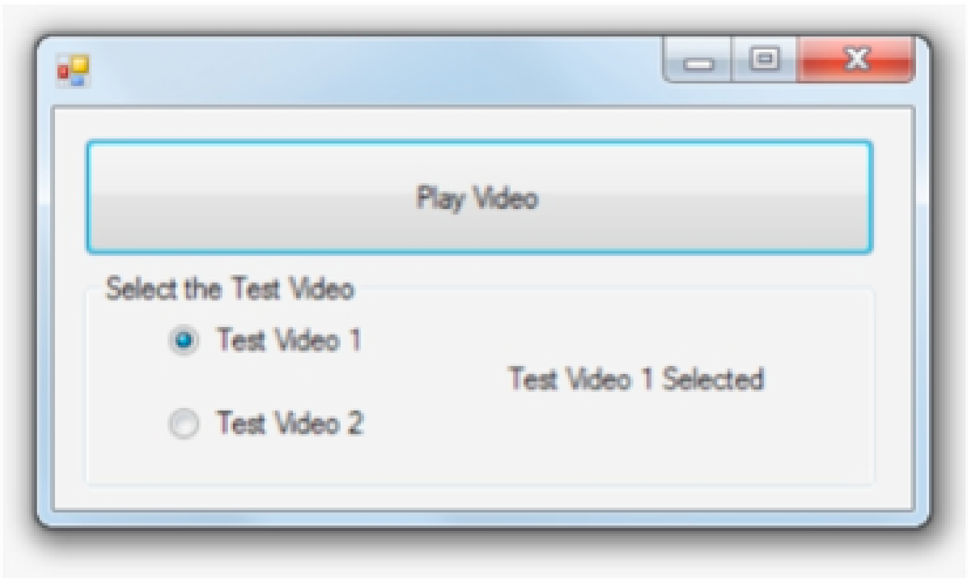
outclass=SoftmaxLayer tells that sum of probability of a sample belonging to a particular output class equals 1.

6) pybrain.supervised.trainers.**BackpropTrainer**(*module*, *dataset=None*, *learningrate=0.01*, *lrdecay=1.0*, *momentum=0.0*, *verbose=False*,*batchlearning=False*, *weightdecay=0.0*)

Discipline that prepares the variables of a component according to a directed dataset (possibly consecutive) by backpropagating the blunders (through time).


Extracted Eye (250 X 470)
Resized (20 X 45) and turned into grayscale
Eye written pixel by pixel in a text with its class
Make a Pybrain classification dataset
Make a trainer and network using dataset on different neurons in hidden layers
Use trainer and network for real time classification

Process flow of the usage of the Pybrain library for classification (Fig.3.6)



The UI of the executable that starts the process (Fig.3.7)

The training rate provides proportion which parameters are modified into the span of the angle. The training rate diminishes by lrdecay, which is useful to raise the learning rate after every planning step. The variables are likewise well balanced regarding power, which is the percentage where the position of the previous timestep is implemented.

Regarding batchlearning is defined, the variables are rejuvenated just toward the final of every get older. Default is False.Weightdecay pertains to the weightdecay rate, where 0 is not a weight rot by any stretch out of the creativity.

Making a UI and video player in C# .NET

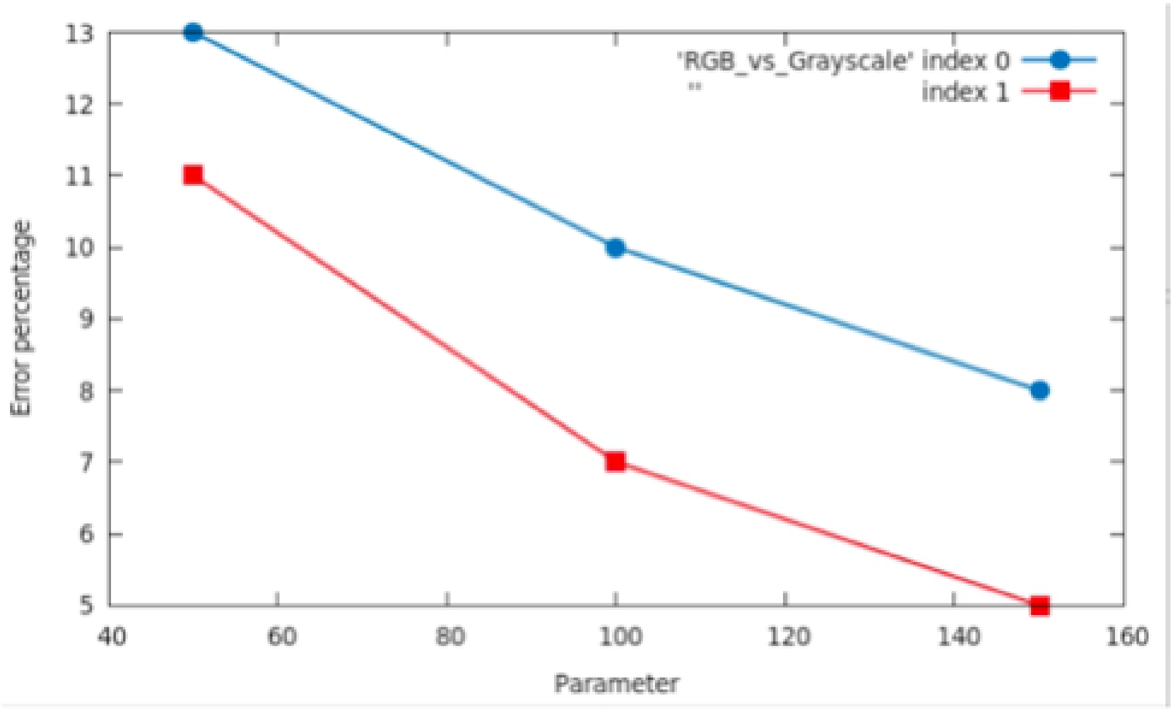
The execution starts using an executable, whose starting window is as follows:

The two radio-buttons correspond to the two test videos that we have prepared using which the testing will be done. Selecting the first radio-button will make the the testing process to use the first video and selecting the second one will make the testing process use another testing video.
The “Play Video” button calls matlab via command-line and passes to it, the name of the matlab code file which does all the processing, the test video file name and the sample video filename as arguments.
The test video and sample video files are in the same folder as the the matlab code file.
The executable was coded using C# and .NET.

The video player was also coded in C# .NET. It uses the VLC ActiveX plugin for the playback. The whole code concerning the playback resides in a .dll file (matlab_test.dll) which is loaded by MATLAB while processing and then the appropriate functions are called from the MATLAB script for playing and pausing the video according to the processing results.

4.ANALYSIS AND RESULT

Error Estimation



Marking Scheme :

1) The Red Marking represent Error Percentage for GrayScale Dataset.

2) The Blue Marking represent Error Percentage for RGB Dataset.

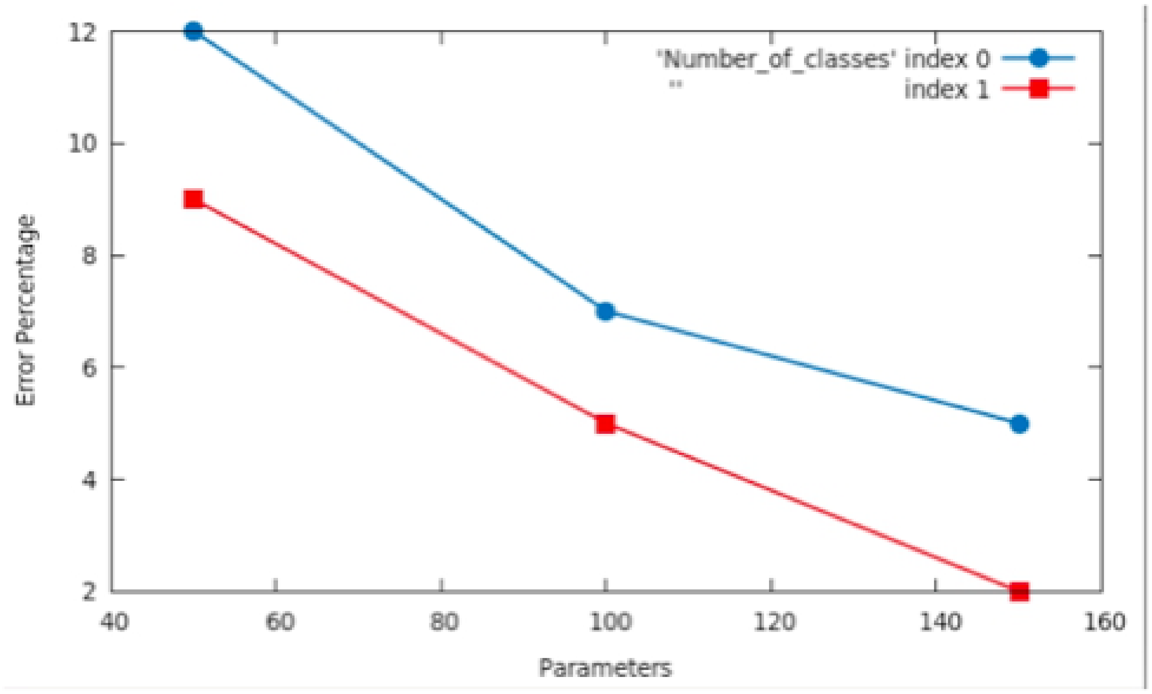
The above graph is plotted between the error percentage obtained on a dataset of

414 RGB images and dataset of 414 grayscale images vs Parameter i.e. the maximum number of neurons in hidden layer 1 and 2. Both the datasets contain 272 negative and 142 positive eye samples.

We observe that for a particular parameter value , the error percentage in GrayScale dataset is lower than that in the RGB dataset.

The main reason for this is the number of features, we are using an image of size 20*45 in which each pixel represents a feature. So for a GrayScale image we have

900 features and for an RGB image we have 900 features for each of the Red,Green and Blue channels, in all that makes 2700 features. More number of features makes the generalization of hypothesis difficult.
Although Error Percentage decreases for both the graphs with increase in number of parameters.



Marking Scheme :
1) The Red Marking represent Error Percentage for 5 output classes namely

'Front' ,'Down' , 'Left' , 'Right' , 'Up' .

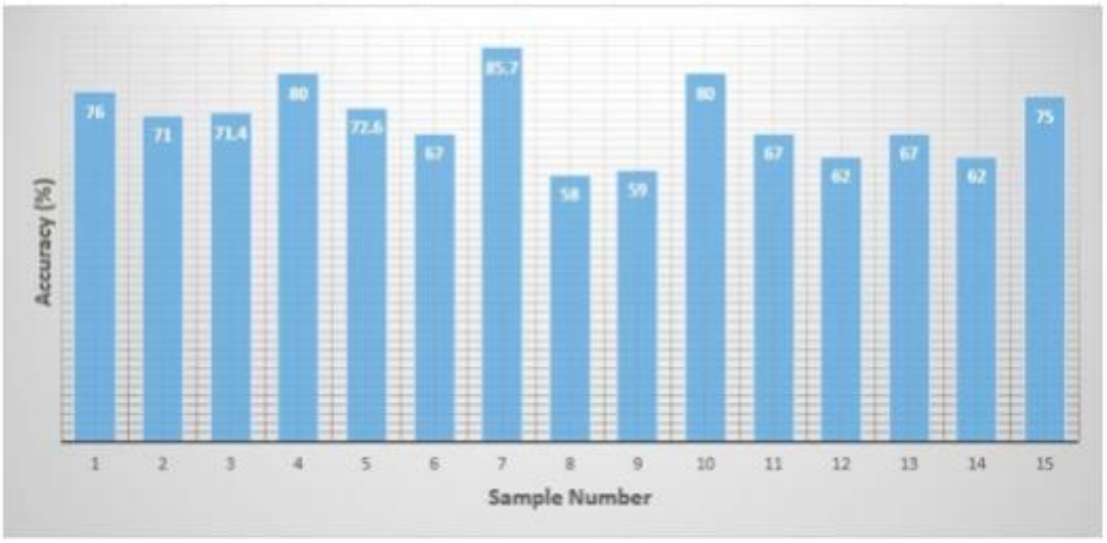
2) The Blue Marking represent Error Percentage for 2 output classes

namely 'Into the Frame' , 'Out of Frame' .

The above graph is plotted between the error percentage obtained on a dataset of

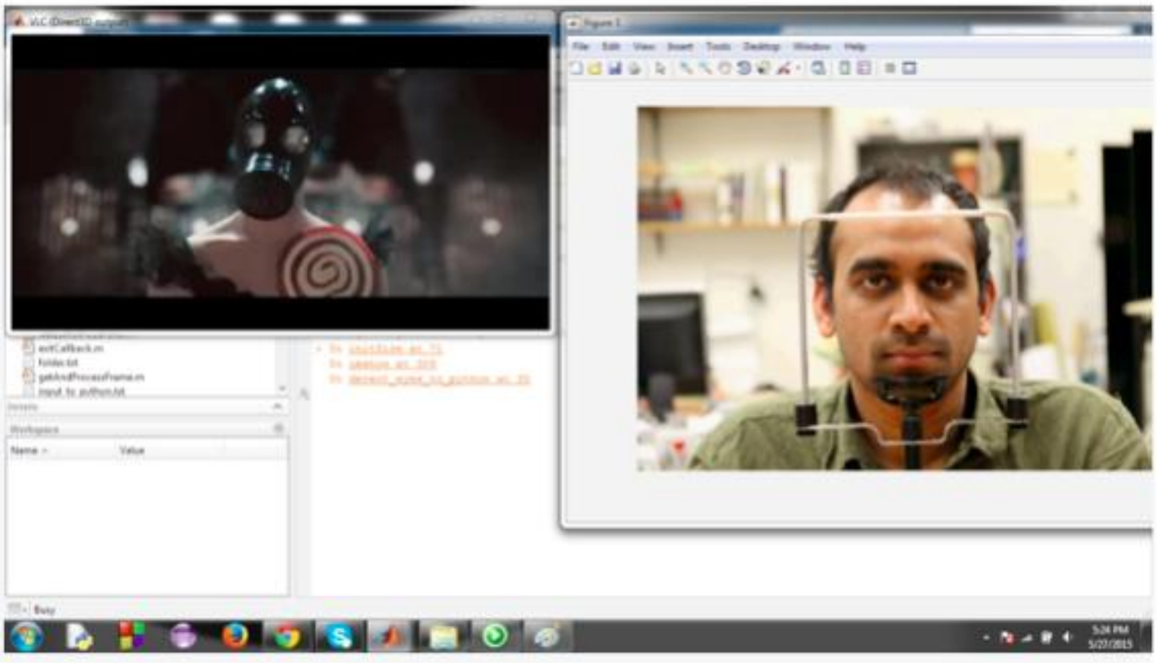
414 grayscale images having 5 output classes and having 2 output classes vs Parameter i.e. the maximum number of neurons in hidden layer 1 and 2. Both the dataset contain 272 negative and 142 positive eye samples.

We see that for a particular parameter value , the error percentage in 2 output classes dataset is lower than that in the 5 output classes dataset.
The main reason for this is the number output classes , when the number of output classes are more, hypothesis tends to face difficulty to decide the parameter vector for each class. More number of classes makes the generalization of hypothesis difficult.Although Error Percentage decreases for both the graphs with increase in number of parameters.



Percentage accuracy for each of the Person's sample videos (Fig.4.3) The final code is run on 15 folders, each containing video of different persons. Each video focuses on different eye gazes of a single person.The Accuracy obtained from each video is mentioned in the bar graph with the appropriate sample number. This Analysis shows that the average accuracy for different faces is 70.3 % .

Final Application



Developed a desktop application that play/pause the video based on eye gaze. The application can be extended to real time so that it can be used in online video lectures and mobile phones.

5.LIMITATIONS AND FURTHER SCOPE

- Incorporating real time video capture

We are using self made videos for testing, this can be extended to real time video capture via webcam.

-Using better and faster methods for gaze detection

We have used the pybrain library which is an Artificial Neural Network Library of Python for the detection, which is slower but easier to use , other better and faster libraries like pylearn2 , theanet etc. can be used to incorporate real time working.

-High success rate on Low quality and lowly lit videos

The present working system uses high quality and well lit images for the processing. This can be extended to work on medium lit and low quality images like we get from a normal webcam.

-Incorporating Rotated Faces

The present working system considers only straight faces, tilted faces need to be incorporated.

-Assembling the whole working system into a single package

Since we are using a lot of different methods for our present system, it uses Python, MATLAB and C# .NET. It can be modified and assembled into a single package, so that it's easier to run and light on system resources

-Integration of all components in a single UI

In the present system, there are different UIs for starting the processing, displaying the video and displaying the current frame being tested,this all can be integrated into a single UI for ease of use and aesthetics.