INDIAN INSTITUTE OF INFORMATION TECHNOLOGY ALLAHABAD

(A Centre of Excellence in Information Technology) Deoghat, Jhalwa, Allahabad – 211 012 (U.P.),
India Ph. 0532-2922008, 2922216, Fax: 0532-2430006,
Web: www.iiita.ac.in, E-mail: dr.e@iiita.ac.in

Semester VII – End Semester Project Report

Project Title

Real Time Emotion Classification using User's Facial Expression

Under the Guidance of:

Prof. U.S. Tiwari

Submission By:
Vaibhav Srivastava (IIT2013027)
Himanshu Tuteja (IIT2013038)
Anirudh Gupta (IIT2013117)

CERTIFICATION

This is to certify that the group of below mentioned students has successfully completed the project work titled **"Real Time Emotion Classification using User's Facial Expression"** as the VII semester mini-project prescribed by the Indian Institute of Information Technology, Allahabad.

This project is the record of authentic work carried out during the academic year (Aug – Dec) 2016.

Vaibhav Srivastava (IIT2013027)
Himanshu Tuteja (IIT2013038)
Anirudh Gupta (IIT2013117)

ACKNOWLEDGEMENT

The present work took a lot of our efforts. However, the work carried out would not have been possible without the support and effort of many people around. We would like to thank all of them for their throughout help and support.

We are highly indebted to **Prof. U.S. Tiwari** for his expert guidance and throughout during the entire course of the project. We would like to thank him for providing the required information and assistance to complete the project.

The project would not have been possible without the kind cooperation and support **Mr. Sudhakar Mishra**. Our special thanks also goes to our colleague and friends in carrying out the task and also to the people who have willingly helped us out with their abilities.

Table of Content

1. Abstract

Facial behaviour is one of the most important cues for sensing human emotion and intentions among people. As computing becomes more human centred, an automatic system for accurate facial expression analysis is relevant in emerging fields such as interactive games (for instance, the games played using Microsoft Kinect), online education, entertainment, autonomous driving, analysis of viewer reaction to advertisements, etc. For example, the reactions of gamers could be used as feedback to improve the gaming experience on systems like the Microsoft Kinect. Similarly, analysis of facial expressions of drivers would help in determining their stress level and such information could be used to alert drivers if they are stressed and in a state unsafe for driving. With these applications in mind, this report describes our attempt to learn facial expressions in real time using deep learning. We also contrast our deep learning approach with conventional "shallow" learning based approaches and show that a convolutional neural network is far more effective at learning representations of facial expression data.

2. Project Title

REAL TIME EMOTION CLASSIFICATION USING USER'S FACIAL EXPRESSION

3. Introduction

3.1 The Problem

Human interaction is carried out mainly using speech, but they can also employ body gestures to display their emotions. And the display of human emotions is generally done using facial expressions. This non-verbal communication can also tell much to the person with whom we are interacting. Although humans recognize emotions of each other virtually with ease and without delay, but implementing such a problem in a machine could be really a hard task.

One of a huge challenge is developing computers that can effectively simulate human interaction. A lot of research has been done in the past years in this field. This motivates us to develop a system which can efficiently determine the mental state of the user by studying the facial expressions.

3.2 The Solution

We propose solution in this report for the problem in hand. We propose a system which classifies the emotion of the person using various machine learning techniques in real time. Different techniques have different evaluating parameters and different accuracies. Our goal is to find the best suitable method that can be used as the solution to the above problem.
Any learning algorithm that uses features extracted from data will always be upper-bounded in its accuracy by the expressiveness of the features. As a result, a motivation for deep learning based approaches is that learning algorithms can learn, or design features much better than humans can.

4. Project Objectives

The objectives that are to be achieved are-

a) Classifying the emotion of the person in real time.

b) Using effective machine learning algorithms to achieve the above mentioned task.

c) Comparing the accuracies obtained by different methodologies and choosing the best one.

5. Scope of Project Work

The scope of this project includes-

a) Developing interactive games which can use the user's emotion to keep his interest in the game. Also, the reaction can be used as the feedback for the developer and can help him in improving the gameplay.

b) Analysis of viewer reaction to advertisements.

c) E-Commerce sites can use this tool to study the user's interest on their website and can improve their services.

d) It has been observed that the mental status of a person has an effect on his/her driving which sometimes cause serious accidents. With the proper use of this system we can help in avoiding such accidents.

6. Literature Survey

The basic task required to accomplish the solution of online facial feature recognition system is understanding the concepts of machine learning and convolution neural networks (CNN) and comparing their results to obtain the best possible solution, the one with the highest accuracy

6.1. The Database

In this project, we used a dataset provided by FER2013[1] website, which consists of about 32298 well-structured 48 × 48 pixel gray-scale images of faces. The images are processed in such a way that the faces are almost centred and each face occupies about the same amount of space in each image. Each image has to be categorized into one of the seven classes that express different facial emotions. These facial emotions have been categorized as:

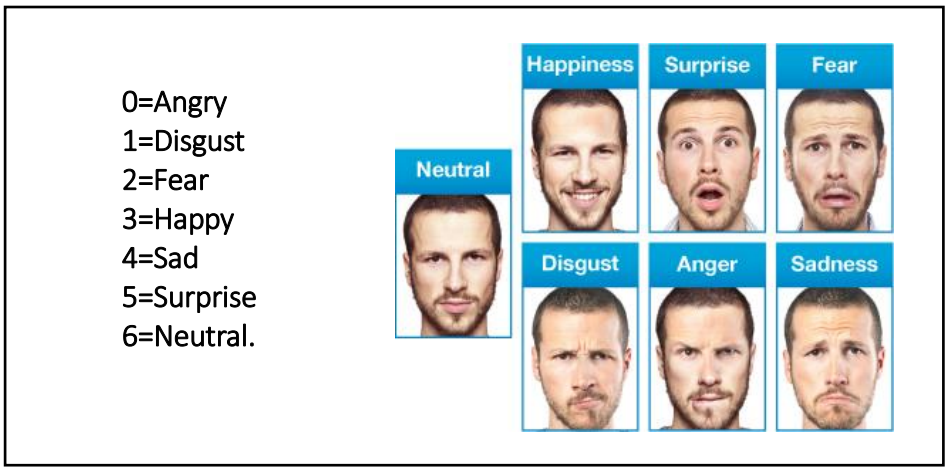
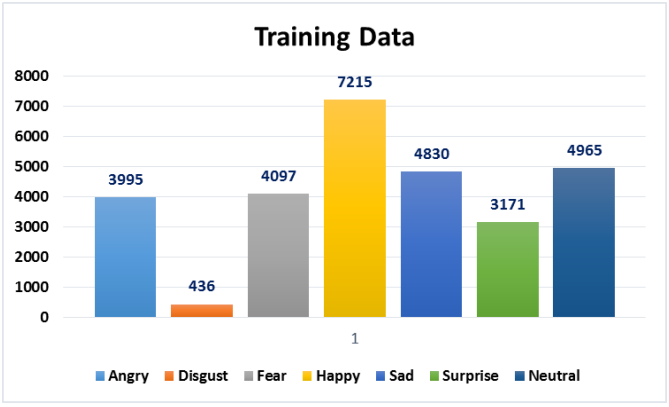
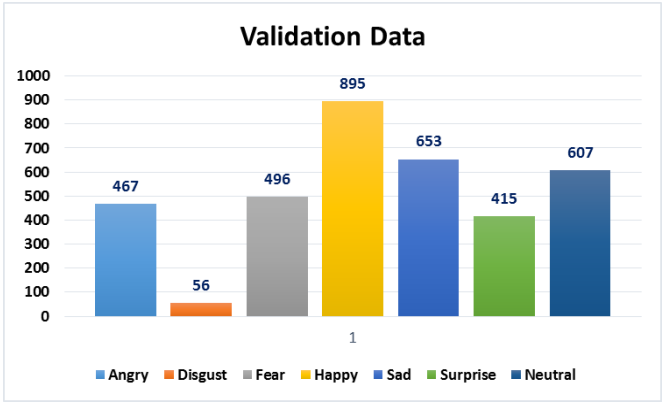


Fig. 1: An example for each facial expression category.





6.2. Python

Python is a high-level, interpreted, dynamic and general purpose language which allows a greater readability to the user, as its syntax allows user to use fewer lines to express the concepts. Also it provides constructs to enable programs a small as well as a large scale. Python supports object oriented, procedural, functional and imperative programming paradigms. It also has a very large and comprehensive standard library.

6.3. Keras[2]

Keras is a high-level library of neural networks, written in Python and is capable of running on top of either Theano or TensorFlow. It enables fast experimentation. Being able to achieve result from an idea with the least possible delay is key to doing good research.

6.4. Haar-cascade Detector[3]

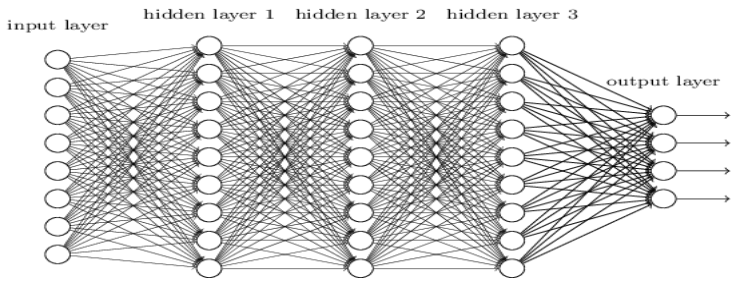
Face detection includes determining the location in which the face is present in an image. It only detects the facial features and hence detects the face and ignores all the other objects like trees, cars etc. It is a machine learning based approach where a lot of positive and negative images are

used to train the classifier. Then the classifier is used to detect faces in other test images. In this method, instead of applying all the features at once, we apply feature in stages. If a set of features is applied and the result crosses the threshold, then further tests will be carried out. However if it fails in a test, the further tests won't be performed in that region.

6.5. Convolution Neural Network[4][5][6]

Convolutional neural networks (CNNs) are widely used in the tasks of pattern and image recognition because of their superiority over other techniques. Convolutional Neural Networks are similar to simple Neural Networks: they are made up of neurons with learnable weights and biases. Each neuron admits some inputs, performs a dot product operation and optionally follows it with a non-linearity. From the raw image pixels on one end to the scores of classes at the other, the entire network expresses a single differential score function. And they still have a loss function (e.g. Softmax) on the last (fully-connected) layer and all the tricks learned for developing regular Neural Networks are still applicable.

The basic need of CNN aroused for image recognitions problems as in case of images the no of parameters in input layer become large and in order to make the recognizing system efficient the number of hidden layers in the neural network are also large, due to which the effect to the weights of initial hidden layers is not much during back propagation. This increases the number of iterations needed to adjust the weights in order to obtain good accuracy from the system, thereby increasing the computation power.



For example an input image of size 300 x 300 pixels will require 90000 neurons in the input layer. Adjusting weights for such large number of input neurons will require large number of hidden layers and also good accuracy can't be guaranteed.

CNN take into consideration spatial information in an image. It extracts the important features and trains the system accordingly. Given below is the basic architecture overview of CNN:

- **Input Layer** :-

  This layer holds the raw pixel values of the image e.g. in this case an image of width and height 48 and 48, and with the three color channels which are R, G and B it would have dimensions 48 x 48 x 3.

- **CONV Layer** :-

  In this layer, each neuron performs a dot product between their weights and their local receptive fields. This may result in volume such as [48x48x12] if we have decided to employ 12 filters.

- **RELU Layer** :-

  This layer performs an elementwise activation function, such as the **max(0,x)** which is used to do the thresholding at zero. This removes negative intensities while keeping the

volume unchanged [48x48x12].

- **POOL layer** :-

  This layer will do a downsampling along the spatial dimensions (width, height), thus resulting a volume as [24x24x12] in case if we are using a 2 x 2 pooling filter. Eg. A 2 x 2 max pool filter.

- **FC Layer**

  This fully-connected layer computes the class scores, resulting in a volume of size [1x1x7], where each number corresponds to a particular class score, among the 7 possible categories. As the name implies, each neuron in this layer will be connected to all the neurons in the previous volume.

The filters in the CONV layer are simple edge detection filters at different angles (horizontal, vertical, +45 degree i.e. diagonals, etc.) as the edges carry the useful information in the image. We can multiple combinations of CONV, RELU and POOL layer in between the Input and the FC layer to try out different architectures in order to obtain best possible results.

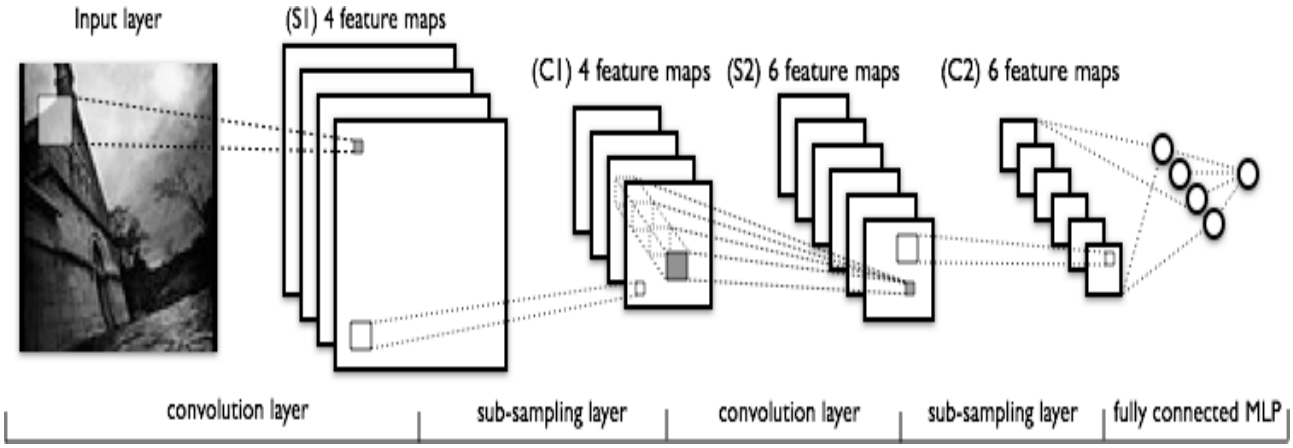
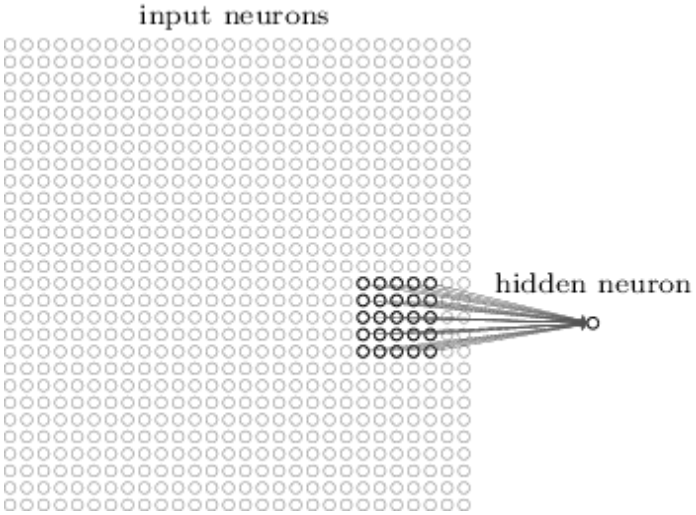


Fig. Basic CNN Architecture

The three basic ideas used in Convolutional Neural Networks are as: ***Local receptive fields***, ***pooling*** and ***shared weights***. They are described as:-



- **Local Receptive Fields:** In a convolution neural network each unit in a hidden layer is only connected to a small number of units in the previous layer. For instance the very first hidden layer will only be attached to a small localized region of the input image. This region is called the receptive field.
- **Shared Weights:** When we take the dot product between a filter and a patch of an image, we are measuring the correlation between the filter and the patch. The larger the value of the dot product, the more alike the filter and the patch look. Because we want to capture the same feature at different places in an image.

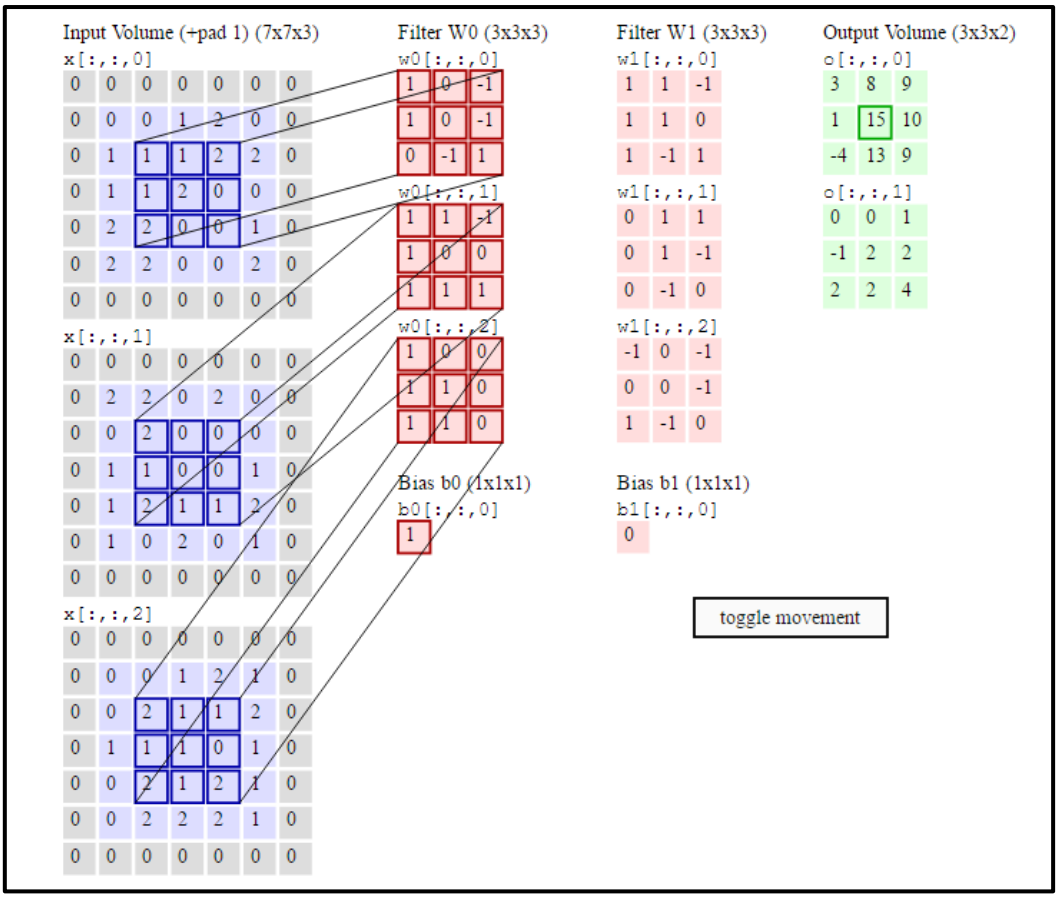


Fig: The spatial filters are applied on the input image (CONV layer)

7. Other Related Work

7.1 Learning facial expressions from an Image[7]

This paper aims in classifying the facial expression using SVM, SVM with Gabor filter and CNN. The results of this paper show that the CNN method outperforms the other approaches and obtains accuracy of 55%.

7.2 Convolutional Neural Networks for Facial Expression Recognition [8]

This paper developed various CNNs for a facial expression recognition problem and evaluated their performances using different post-processing and visualization techniques. The results demonstrated that deep CNNs are capable of learning facial characteristics and improving facial emotion detection. Also, the hybrid feature sets did not help in improving the model accuracy, which means that the convolutional networks can intrinsically learn the key facial features by using only raw pixel data.

8. Methodology

The first step to recognise the emotion involves the requirement of a model that can be used for classification of the emotion. We have used CNN model for this purpose. Below is the flow chart for the solution of the given problem.

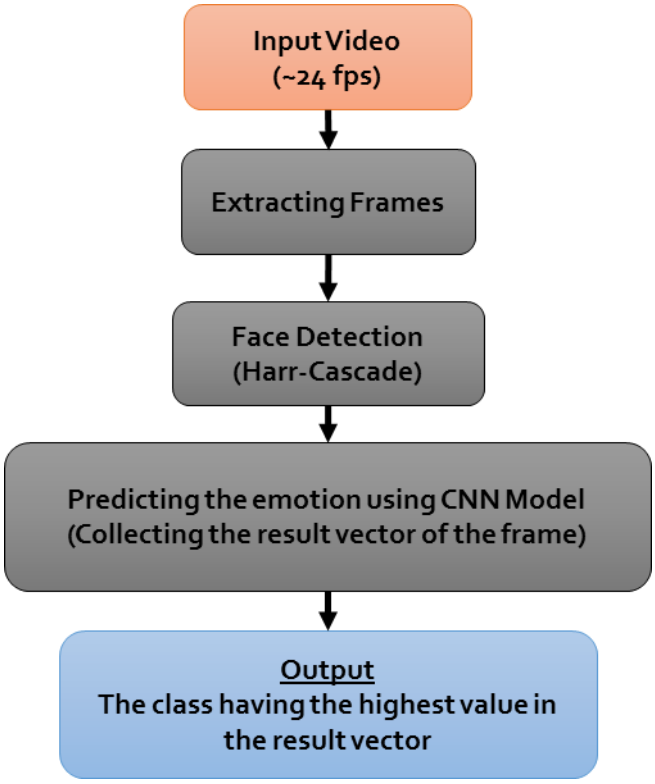


Fig: Flow chart of the Methodology

8.1 CNN Models

Deep learning is a popular technique used in computer vision. We chose convolutional neural network (CNN) layers as building blocks to create our model architecture. CNNs are known to imitate how the human brain works when analysing visuals.

A typical architecture of a convolutional neural network will contain an input layer, some convolutional layers, some dense layers (aka. fully-connected layers), and an output layer. These are linearly stacked layers ordered in sequence. In Keras, the model is created as Sequential() and more layers are added to build architecture.

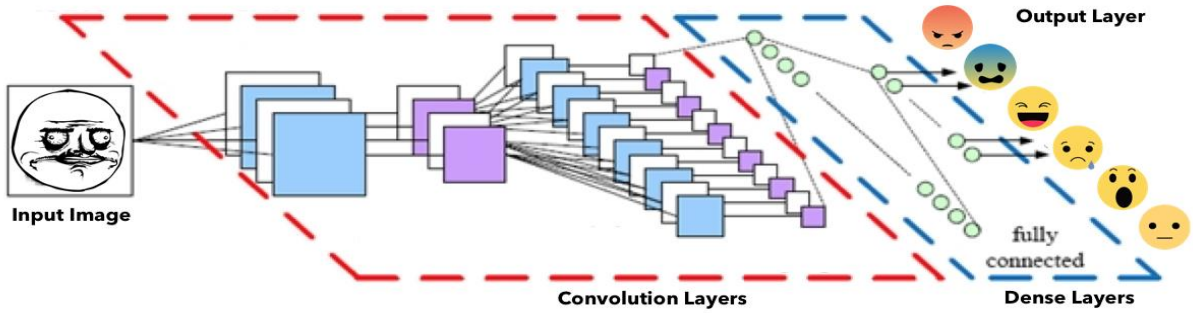


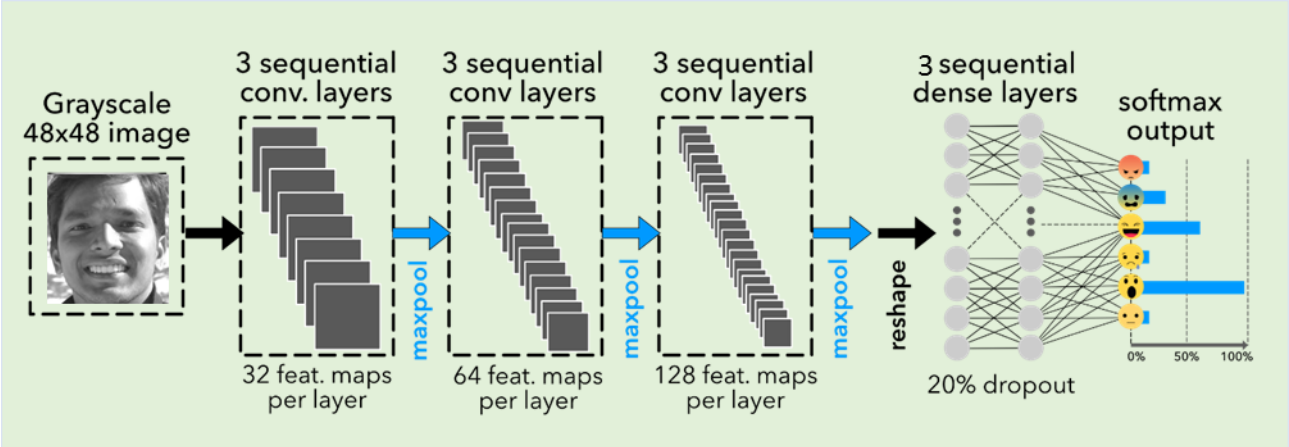
Fig: Basic CNN model for the classifying the emotions

We worked with 3 different models and compared their results to find the best model to solve the stated problem. The models are:

a. **CNN Model(C1)**
b. **Improved CNN (C2)**
c. **CNN + ORB**

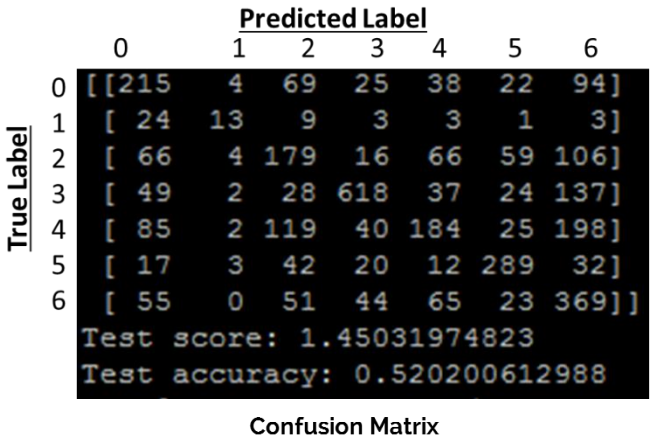
8.1.1 The CNN Model (C1)

This model consists of total 9 convolutional layers consisting of 3 layers of 32 3x3 filter, 3 layers of 64 3x3 filters and 3 layers of 128 3x3 filters, followed by a dropout of 20% and 3 dense layers 64, 64 and 7 neurons each. The total parameters (neurons) used are 779783. The accuracy obtained on FER2013 validation set is **52.02%**



Layer (type)	Output Shape	Param #	Connected to
convolution2d_1 (Convolution2D)	(None, 32, 48, 48)	320	convolution2d_input_1[0][0]
convolution2d_2 (Convolution2D)	(None, 32, 48, 48)	9248	convolution2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 32, 48, 48)	9248	convolution2d_2[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 32, 24, 24)	0	convolution2d_3[0][0]
convolution2d_4 (Convolution2D)	(None, 64, 24, 24)	18496	maxpooling2d_1[0][0]
convolution2d_5 (Convolution2D)	(None, 64, 24, 24)	36928	convolution2d_4[0][0]
convolution2d_6 (Convolution2D)	(None, 64, 24, 24)	36928	convolution2d_5[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 64, 12, 12)	0	convolution2d_6[0][0]
convolution2d_7 (Convolution2D)	(None, 128, 12, 12)	73856	maxpooling2d_2[0][0]
convolution2d_8 (Convolution2D)	(None, 128, 12, 12)	147584	convolution2d_7[0][0]
convolution2d_9 (Convolution2D)	(None, 128, 12, 12)	147584	convolution2d_8[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 128, 6, 6)	0	convolution2d_9[0][0]
flatten_1 (Flatten)	(None, 4608)	0	maxpooling2d_3[0][0]
dense_1 (Dense)	(None, 64)	294976	flatten_1[0][0]
dense_2 (Dense)	(None, 64)	4160	dense_1[0][0]
dense_3 (Dense)	(None, 7)	455	dense_2[0][0]

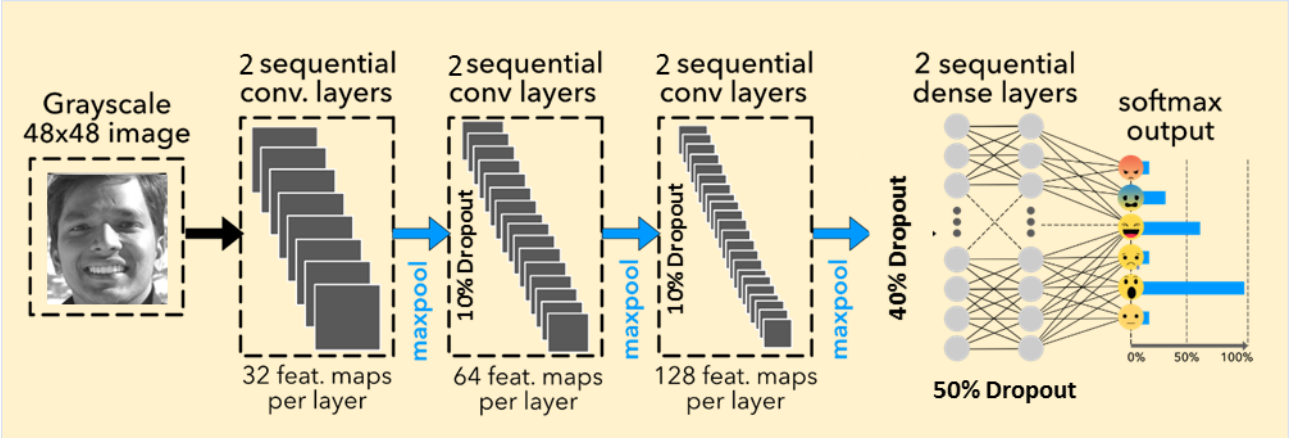
Total params: 779783



Confusion Matrix

8.1.2 The Improved CNN Model (C2)

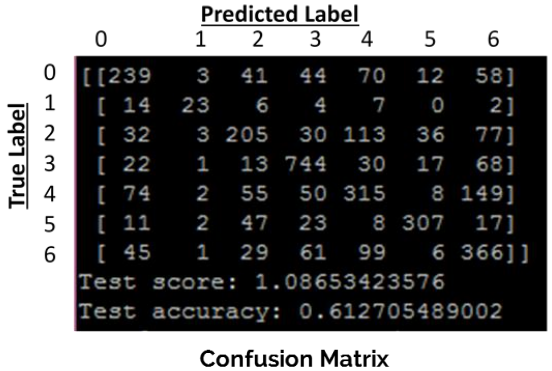
This model consists of total 6 convolutional layers consisting of 2 layers of 32 3x3 filter, 2 layers of 64 3x3 filters and 2 layers of 128 3x3 filters, with Maxpool of 2x2 & dropout of 10% after each layer followed by a dropout of 40% and 2 dense layers with 2048 neurons and 50% dropout. The total parameters (neurons) used are 7394247.The accuracy obtained on FER2013 validation set is **61.27%.**



8.1.3 Model: CNN + ORB

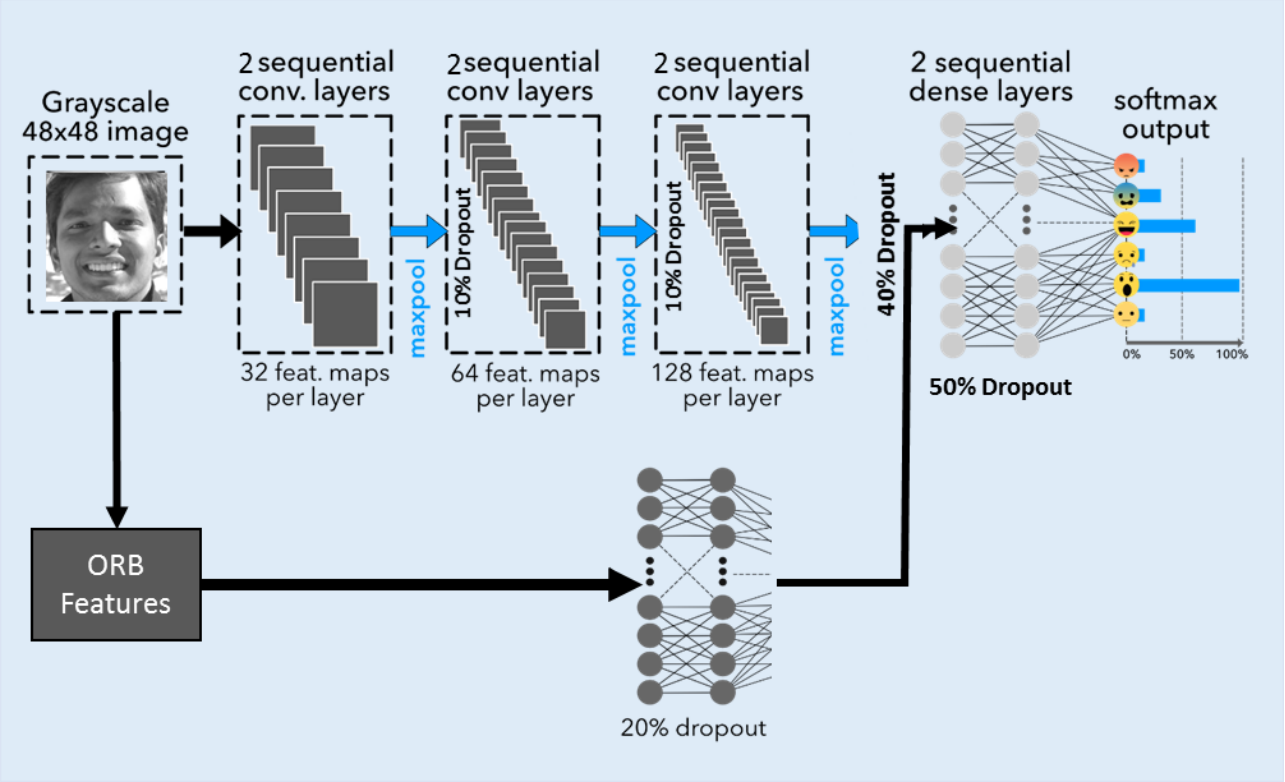
Layer (type)	Output Shape	Param #	Connected to
convolution2d_1 (Convolution2D)	(None, 3, 50, 32)	13856	convolution2d_input_1[0][0]
convolution2d_2 (Convolution2D)	(None, 5, 52, 32)	9248	convolution2d_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 2, 26, 32)	0	convolution2d_2[0][0]
dropout_1 (Dropout)	(None, 2, 26, 32)	0	maxpooling2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 4, 28, 64)	18496	dropout_1[0][0]
convolution2d_4 (Convolution2D)	(None, 6, 30, 64)	36928	convolution2d_3[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 3, 15, 64)	0	convolution2d_4[0][0]
dropout_2 (Dropout)	(None, 3, 15, 64)	0	maxpooling2d_2[0][0]
convolution2d_5 (Convolution2D)	(None, 5, 17, 128)	73856	dropout_2[0][0]
convolution2d_6 (Convolution2D)	(None, 7, 19, 128)	147584	convolution2d_5[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 3, 9, 128)	0	convolution2d_6[0][0]
dropout_3 (Dropout)	(None, 3, 9, 128)	0	maxpooling2d_3[0][0]
flatten_1 (Flatten)	(None, 3456)	0	dropout_3[0][0]
dense_1 (Dense)	(None, 2048)	7079936	flatten_1[0][0]
dropout_4 (Dropout)	(None, 2048)	0	dense_1[0][0]
dense_2 (Dense)	(None, 7)	14343	dropout_4[0][0]

Total params: 7394247



Confusion Matrix

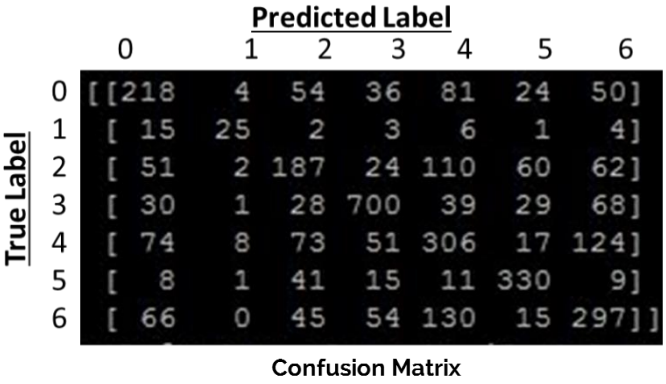
This model includes the CNN model C2 along with the ORB features of the image. The ORB features of the image are merged with CNN parameters forming a larger set of inputs which are then given to the neural network (2048 neurons) to train the model. The total parameters (neurons) used are 36482791.The accuracy obtained on FER2013 validation set is **57.48%.**



Layer (type)	Output Shape	Param #	Connected to
convolution2d_1 (Convolution2D)	(None, 32, 50, 50)	320	convolution2d_input_1[0][0]
convolution2d_2 (Convolution2D)	(None, 32, 52, 52)	9248	convolution2d_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 32, 26, 26)	0	convolution2d_2[0][0]
dropout_1 (Dropout)	(None, 32, 26, 26)	0	maxpooling2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 64, 28, 28)	18496	dropout_1[0][0]
convolution2d_4 (Convolution2D)	(None, 64, 30, 30)	36928	convolution2d_3[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 64, 15, 15)	0	convolution2d_4[0][0]
dropout_2 (Dropout)	(None, 64, 15, 15)	0	maxpooling2d_2[0][0]
convolution2d_5 (Convolution2D)	(None, 128, 17, 17)	73856	dropout_2[0][0]
convolution2d_6 (Convolution2D)	(None, 128, 19, 19)	147584	convolution2d_5[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 128, 9, 9)	0	convolution2d_6[0][0]
dropout_3 (Dropout)	(None, 128, 9, 9)	0	maxpooling2d_3[0][0]
flatten_1 (Flatten)	(None, 10368)	0	dropout_3[0][0]
dense_1 (Dense)	(None, 4096)	6557696	dense_input_1[0][0]
dropout_4 (Dropout)	(None, 4096)	0	dense_1[0][0]
dense_2 (Dense)	(None, 2048)	29624320	merge_1[0][0]
dropout_5 (Dropout)	(None, 2048)	0	dense_2[0][0]
dense_3 (Dense)	(None, 7)	14343	dropout_5[0][0]

Total params: 36482791

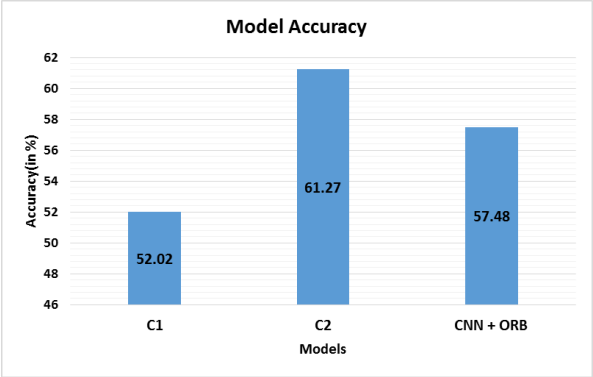
Training....

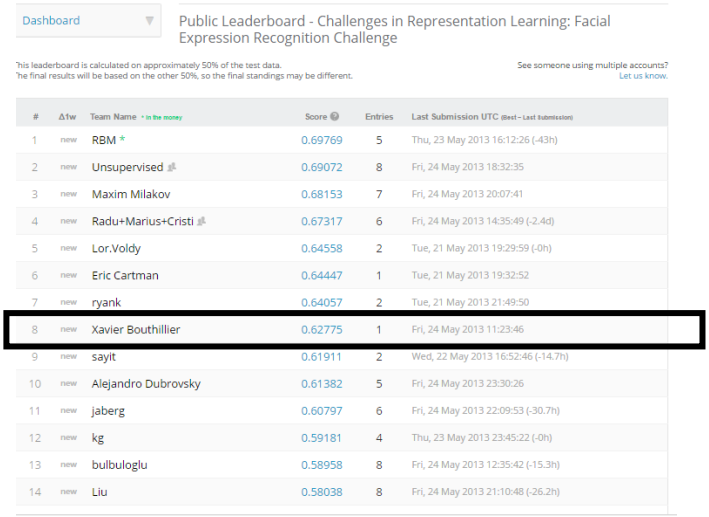


Confusion Matrix

8.2 Choosing the Best Model

Out of the 3 above discussed models, model 2 (C2) performed the best with the accuracy of **61.27%.**A worldwide online challenge was conducted on renowned website kaggle.com[7] for facial expression recognition using the same data (FER2013). We stand at **8th** international position with our model. Using the C2 model we have developed python applications for classifying the emotions on video data.




Dashboard

Public Leaderboard - Challenges in Representation Learning: Facial Expression Recognition Challenge

This leaderboard is calculated on approximately 50% of the test data.
The final results will be based on the other 50%, so the final standings may be different.

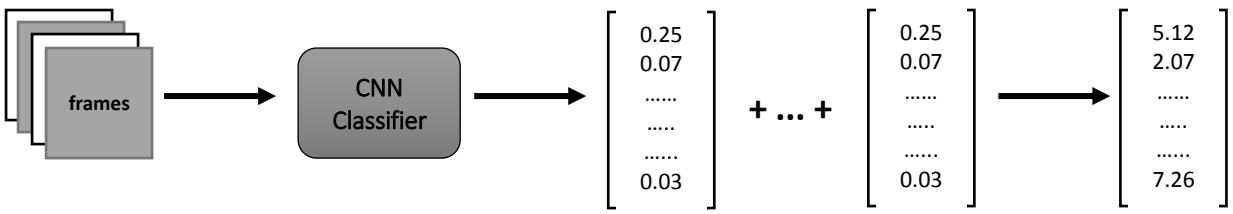
See someone using multiple accounts? Let us know.

#	Δ1w	Team Name * In the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	new	RBM *	0.69769	5	Thu, 23 May 2013 16:12:26 (-43h)
2	new	Unsupervised	0.69072	8	Fri, 24 May 2013 18:32:35
3	new	Maxim Milakov	0.68153	7	Fri, 24 May 2013 20:07:41
4	new	Radu+Marius+Cristi	0.67317	6	Fri, 24 May 2013 14:35:49 (-2.4d)
5	new	Lor.Voldy	0.64558	2	Tue, 21 May 2013 19:29:59 (-0h)
6	new	Eric Cartman	0.64447	1	Tue, 21 May 2013 19:32:52
7	new	ryank	0.64057	2	Tue, 21 May 2013 21:49:50
8	new	Xavier Bouthillier	0.62775	1	Fri, 24 May 2013 11:23:46
9	new	sayit	0.61911	2	Wed, 22 May 2013 16:52:46 (-14.7h)
10	new	Alejandro Dubrovsky	0.61382	5	Fri, 24 May 2013 23:30:26
11	new	jaberg	0.60797	6	Fri, 24 May 2013 22:09:53 (-30.7h)
12	new	kg	0.59181	4	Thu, 23 May 2013 23:45:22 (-0h)
13	new	bulbuloglu	0.58958	8	Fri, 24 May 2013 12:35:42 (-15.3h)
14	new	Liu	0.58038	8	Fri, 24 May 2013 21:10:48 (-26.2h)

Kaggle Position(8th) with 61.27% accuracy

8.3 Classifying the Emotion

Once the weights of various layers of the CNN are adjusted according to the training data set, the next task is to use these weights and the CNN to predict the emotion of the given input. For the input we can get a real time webcam feed or any other video file. Since the video file consists of many frames, we can either predict the emotion of the person in each frame or we can predict the emotion at an interval of **k frames,** say 24. The output vector of the classifier is a 7 x 1 vector consisting the probabilities for each of the 7 given emotions. For the first case we can predict the emotion with the highest probability and for the second case we can add the result vector (probabilities) of the k frames and predict the emotion having the highest probability value.



Result vector of frames

The first step is to extract the frames from the input video and obtain the face of the person using the harr cascade detector. This image of face is then feeded to the CNN architecture to obtain the probabilities for different classes of emotions. Since training the CNN requires high Graphics Processing Unit, we have trained the model on GPU and stored the model and the weights. These weights are later loaded on developed local machine application for predicting the emotion.

```
new_model_1.py - C:\Users\Annu\Documents\Battlefield 4\new_model_1.py (2.7.1...
File  Edit  Format  Run  Options  Window  Help
# model architecture:
model = Sequential()
model.add(Convolution2D(32, 3, 3, border_mode='full', input_shape=(1, 48, 48), a
model.add(Convolution2D(32, 3, 3, border_mode='full', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Convolution2D(64, 3, 3, border_mode='full', activation='relu'))
model.add(Convolution2D(64, 3, 3, border_mode='full', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Convolution2D(128, 3, 3, border_mode='full', activation='relu'))
model.add(Convolution2D(128, 3, 3, border_mode='full', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(2048, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes,activation='softmax'))
model.summary()


# optimizer:
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.save('model_221.h5')
#model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accu
print ('Training....')
model.fit(X_train, Y_train, nb_epoch=nb_epoch, batch_size=batch_size, shuffle=Fa

####################

#model.load_weights('my_model_weights_20_11.h5')
model.save_weights('my_model_221.h5')
score = model.evaluate(X_test, Y_test, show_accuracy=True, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

```
*webcam_video_test.py - C:\Users\Annu\Desktop\Sem VII Project\facial_exp\web...
File  Edit  Format  Run  Options  Window  Help
################################# LOAD THE MODEL
K.set_image_dim_ordering('th')

model= load_model('model_221.h5')
print ('Model Loaded....')

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accur
model.load_weights('my_model_221.h5')
print ('Weights Loaded....')

##################WEBCAM DATA

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
ans = []
time = []
t = 0
p = 0
i = 0
c = 0
x_test = np.empty([1,1,48,48])
while(True):
    ret,frame = cap.read()
    #gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = frame;
    cv2.imshow('img',frame)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        i1 = frame[y-p:y+h+p,x-p:x+w+p]
        pts1 = np.float32([[x-p,y-p],[x+w+p,y-p],[x-p,y+p+h],[x+w+p,y+h+p]])
        pts2 = np.float32([[0,0],[255,0],[0,255], [255,255]])

        M = cv2.getPerspectiveTransform(pts1,pts2)
        img = cv2.warpPerspective(frame,M,(256,256))
        input_img = cv2.resize(img, (48,48))

        gray_image = cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
        input_img = np.array(gray_image, dtype='float32')
        input_img = input_img / 255
```

Fig: (Code Snippets) **a)** Training CNN Model code using Keras
b) Loading and using saved model for classifying Webcam data.

8.4 Modules

Using the C2 CNN model we have developed 3 different python applications for predicting the emotions. They are

i. Real Time Webcam Video classification
ii. Classification of facial expression on Video file
iii. Classification of facial expression on Image file

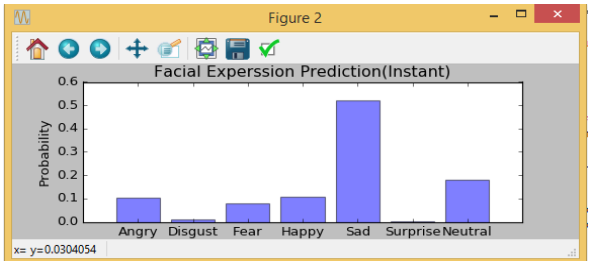


Fig: A bar graph showing the result probabilities of various emotions at an instant

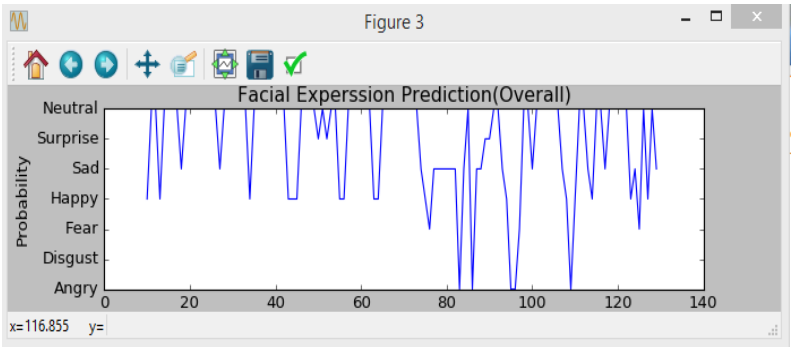
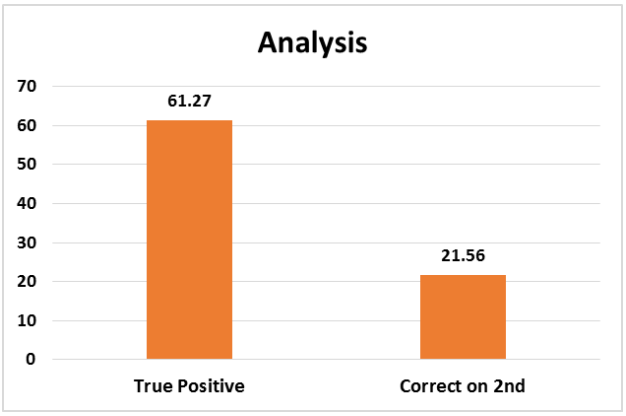
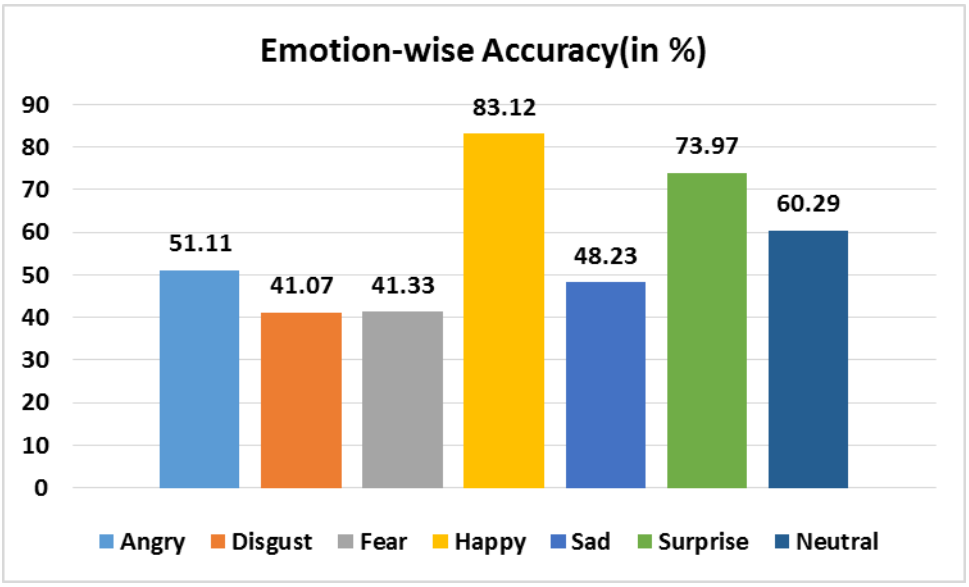


Fig: The real time graph showing the emotions represented in each frame

9. Results & Analysis

As it turns out, the final CNN model had a **validation accuracy of 61.27%**. The emotion-wise accuracy is represented in the graph. On using several other images to test the model developed, we analysed that when the prediction made by model is wrong the correct label (emotion) is often the **second most likely emotion**. This actually makes a lot of sense. Because our expressions usually consist a combination of emotions, and ***only*** using one label to represent an expression can be hard. On testing the validation data, we analysed that **21.56% misclassified images have second most likely emotion as the correct label.** We can see the image with black border(b) is misclassified but the correct label is the second most likely emotion.





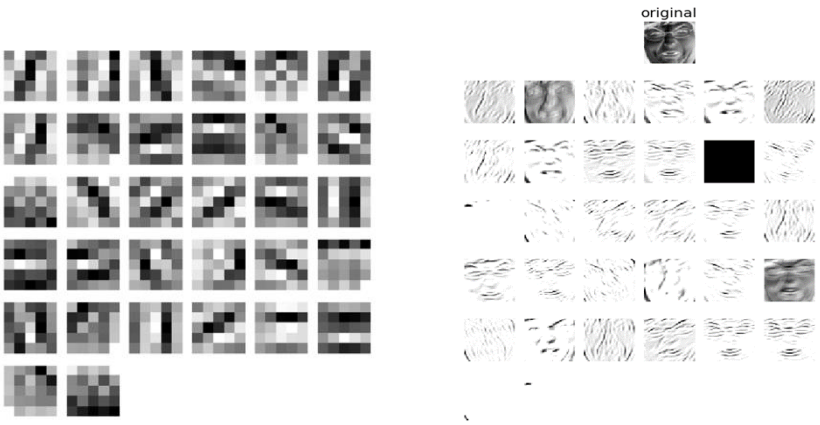
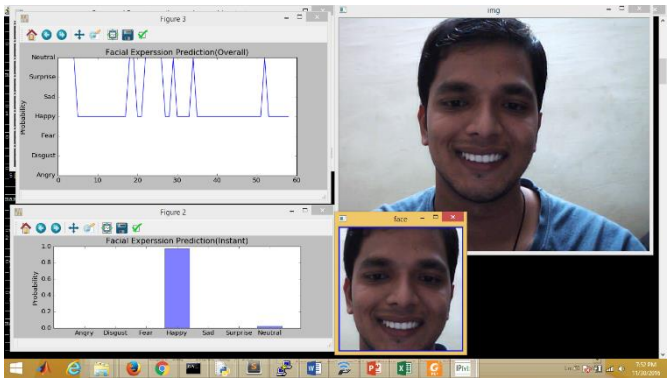
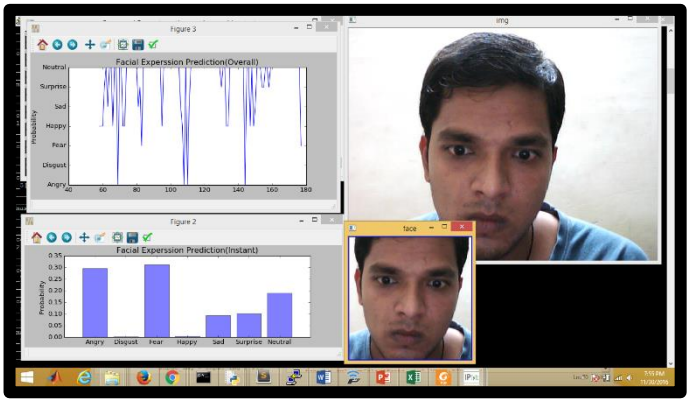
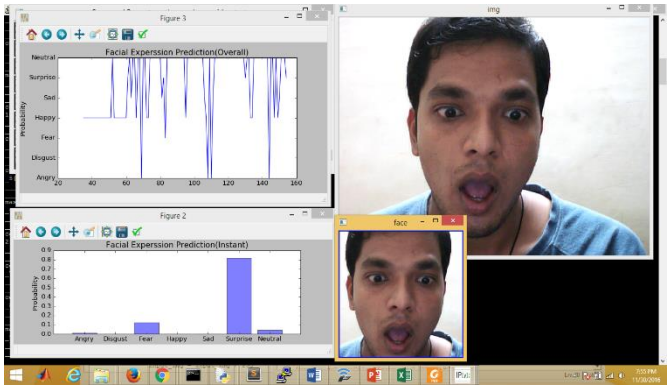
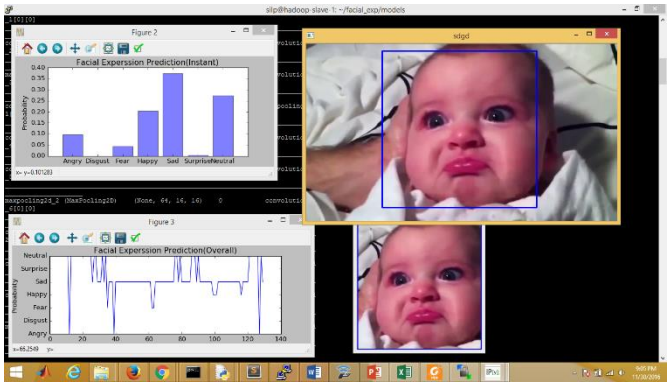


Fig: Layer 1 weights and output Visualisation









9. Technical Requirements

- Nvidia GPU
- CUDA
- Anaconda
- Theano
- Keras
- PyCharm

10. Conclusion

We developed various CNNs for a facial expression recognition problem and evaluated their performances using different post-processing and visualization techniques. The results demonstrated that deep CNNs are capable of learning facial characteristics and improving facial emotion detection. Also, the hybrid feature sets did not help in improving the model accuracy, which means that the convolutional networks can intrinsically learn the key facial features by using only raw pixel data.

11. References

[1] Goodfellow, Ian J., et al. "Challenges in representation learning: A report on three machine learning contests." *International Conference on Neural Information Processing*. Springer Berlin Heidelberg, 2013.

[2] https://github.com/fchollet/**keras**/tree/master/**keras**

[3] Wilson, Phillip Ian, and John Fernandez. "Facial feature detection using Haar classifiers." *Journal of Computing Sciences in Colleges* 21.4 (2006): 127-133.

[4] LeCun, Yann, and Yoshua Bengio. "Convolutional networks for images, speech, and time series." *The handbook of brain theory and neural networks*3361.10 (1995): 1995.

[5] http://cs231n.github.io/convolutional-networks/

[6] Mollahosseini, Ali, David Chan, and Mohammad H. Mahoor. "Going deeper in facial expression recognition using deep neural networks." *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016.

[7]Chudasama, Bhrugurajsinh, Chinmay Duvedi, and Jithin Parayil Thomas. "Learning facial expressions from an image."

[8] Alizadeh, Shima, and Azar Fazel. "Convolutional Neural Networks for Facial Expression Recognition."

[9] http://neuralnetworksanddeeplearning.com/chap6.html

12. Panel Comments & Suggestions