Facial Expression Recognition

SML Project Report

Rameshwar Mishra
2017180
IIIT DELHI

Problem Statement/Introduction

Facial expression plays a crucial role in human interaction and behavioural interpretation of emotion. Facial expression recognition has been a challenging area in facial recognition due to high intra-class variation.
Automatic facial recognition is an interesting problem in computer vision since it can be used in human-computer interaction, behavioural sciences, etc. In this project, we have proposed architecture to classify the given image into seven basic human expressions - ANGER, DISGUST, FEAR, HAPPY, NEUTRAL, SAD, SURPRISE. In addition to it, we have implemented a system that will take the frames from a video and predict the emotions of the person in it. We've also implemented a system which will predict emotion in real time through web-cam.

Keywords—Facial Expression human-computer interaction, computer vision

1. Literature Review

There are many other methods available for recognition of human facial expression, which includes traditional two-level machine learning approaches which rely on HOG, SHIFT and LBP feature extraction and then applying classifier on the data. These methods work well with images captured in controlled and well defined. Environment and fail to work on a little sparse dataset. A study on facial expression using neural network and transfer learning is available in (5) It tries to solve the problem using Spectrograms and Inception Net v3 Model. The inception model is used for automatic image classification and image labelling, and they

achieve an accuracy of 36% on the validation set, which leaves a lot to improve. (3) Here the author uses VGG-16, which is a pre-trained neural network model and contains 16 layers out of which 13 layers are Conv. layers to extract bottleneck features from Images. VGG-16 works well because it has been trained on the Image-net dataset, which contains millions of images. We have also tried this model (Not our final model) in our analysis and reported the accuracy of the dataset used. A detailed study of using deep convolutional neural network(1) for facial expression recognition is done in(Automatic Facial Expression Recognition Using DCNN). This paper uses Caffe ImageNet pre-trained model, which has five convolutional layers. After extracting features from the pre-trained model, it applies the SVM classifier. We will also try to build our convolutional network similar to the model used in Caffe ImageNet but will change the parameter and add/drop some layers to get the best possible results.

2. Dataset Details

We have used the fer2013 dataset from Kaggle. It consists of Images with seven categories (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral). The training set contains 28,709 images in total, and the validation set contains 7178 images in total. This dataset was prepared by Pierre-Luc Carrier and Aaron Courville. The size of each image is 48*48*1. The distribution of data is as follows

Total angry images 3995, label-1

Total disgust images 436, label-2

Total fear images 4097, label-3

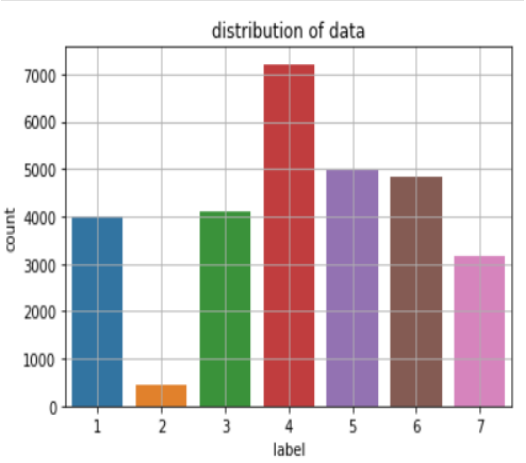
Total happy images 7215, label-4

Total neutral images 4965, label-5

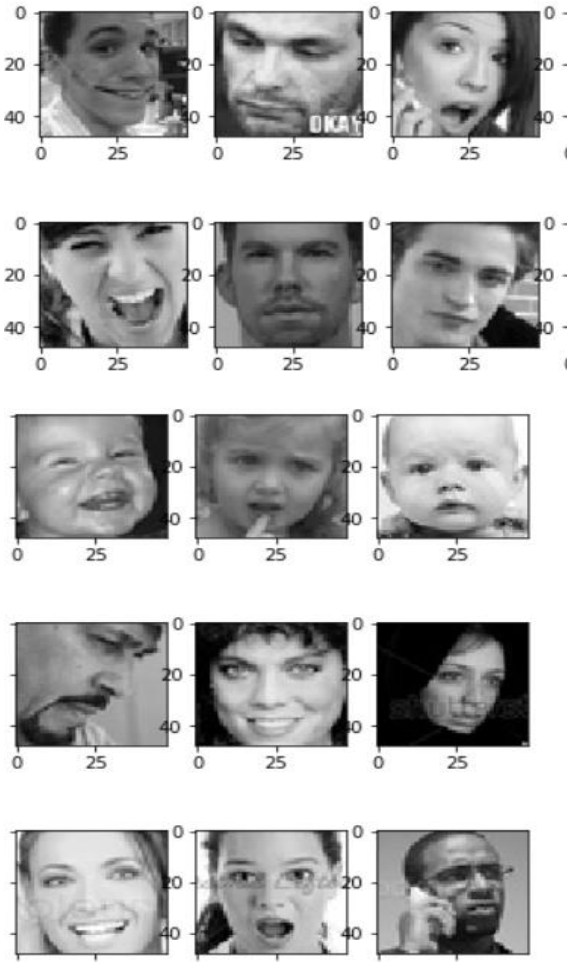
Total sad images 4830, label-6

Total surprise images 3171, label-7

Total images 28709



Some example images are shown below.



Since the disgust image class contains only 436 images out of 28709, we had three options, first concatenate the disgust label data with angry label data since both of them were quite similar, second use SMOTE to balance the data class wise, and third was to use as it is.

We used all three approach and got similar results in all three cases. since deep neural networks can be quite precise so we decided to left the data the way it is. One other reason for not using other pre-processing techniques on the data was to get the results with as sparse data set as possible

Model/architecture Design

Model 1

In this model we created batches of size 50 and extracted their features using pre-trained VGG-16 network. Fortunately, VGG-16 net. is available in Keras and Keras also contains a data pre-processor for VGG-16.

Step 1 - create batches from the images

Step 2 – extract bottleneck feature using VGG net

Step 3 – apply random forest classifier

Step 4 – report accuracy of validation set

Model architecture

Images ➔ create batches ➔ extract VGG-16 features ➔Train Classifier ➔ Predict Validation Labels

Results and analysis

Validation accuracy ➔ (28% to 31%)

The model performance was poor and keeping in mind that our model should give us reasonable results while making predictions in real time we discarded this model.

The model classified happy images quite accurately and a possible reason for that is 25% of total training images are from happy class and our model is well trained for such images.

This problem is quite complex for basic non-linear classifiers like random forest and other linear classifiers.

Model 2

In this model we used fully connected feed forward neural network (FFNN) in place of random forest classifier. Other steps are similar to previous model.

The Keras sequential model is suitable for this problem. We used 5 fully connected layer with 512,256,256,128,64 nodes respectively. Adam optimizer with learning rate 0.01 was used.

Step 1 follow step 1, 2 of model 1

Step 2 apply the above described model and fit the data.

Step 3 report accuracy of validation set

Model architecture

Images ➔ create batches ➔ extract VGG-16 features ➔Train FFNN ➔ Predict Validation Labels

Results and analysis

Train data accuracy and loss after 20 epochs

Epoch	Train (cross-entropy loss)	train accuracy
1	1.570199	0.368219
2	1.474361	0.416720
3	1.427271	0.439859
4	1.358513	0.476226
5	1.278631	0.510653
6	1.189090	0.552945
7	1.117018	0.582187
8	1.031164	0.617848
9	0.947892	0.649735
10	0.874658	0.677954
11	0.809196	0.705820
12	0.749755	0.728854
13	0.691173	0.748254
14	0.638825	0.768254
15	0.599822	0.782998
16	0.556842	0.799506
17	0.499590	0.820035
18	0.465165	0.833298
19	0.429047	0.846631
20	0.398912	0.856790

Validation data accuracy =47 %

The accuracy is improved by 18 % from the previous model since FFNN are more powerful than Random forest and some linear classifiers.

Model 3 (Final Proposed Model)

In earlier models we were relying on the VGG-16 model for feature learning and had no control over the architecture of it, VGG-16 is a very powerful and popular network but it does not guarantee that it will work best for our dataset (which is quite diverse and not very elegant). So, we decided to develop our own Convolutional Neural Network (CNN).

Model 3.1

We used 4 2D convolutional layers and 2 fully connected dense layers for final classification

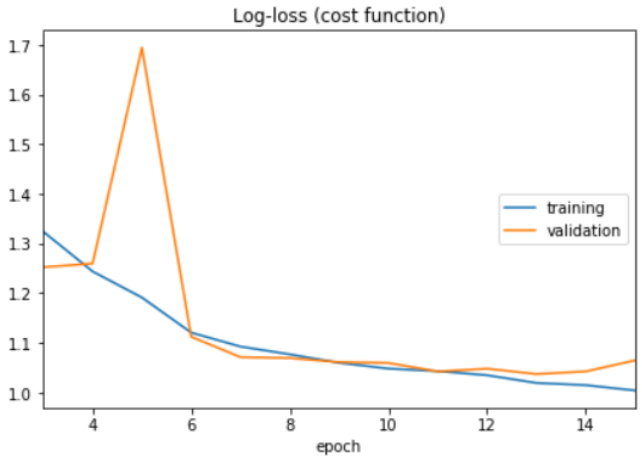
The architecture is as follows

Note – there is one Relu activation layer, Pooling (Max Pooling) layer, and a dropout (0.25) after each Conv layers.

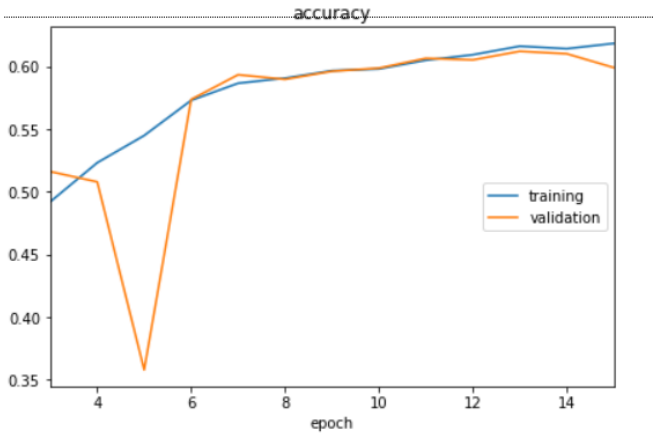
Input Images ➔ Conv layer 1 with 64 filters ➔Conv layer 2 with 128 filters ➔ Conv layer 3 with 512 filters ➔ Conv layer 4 with 512 filters ➔ Flatten ➔ Fully Connected layer with 256 nodes ➔ Fully Connected Layer with 512 nodes ➔ Output layer with 7 class nodes

Results and analysis

Log-loss for training and validation (generated using liveloss plot library) for 15 epochs.



Accuracy plot for training and validation set for 15 epochs.



Min – Max loss and accuracy results

Max accuracy on Train set 61.8%

Max accuracy on validation set 61.2%

```
Log-loss (cost function):
training   (min:    1.004, max:    1.816, cur:    1.004)
validation (min:    1.037, max:    1.694, cur:    1.065)

accuracy:
training   (min:    0.305, max:    0.618, cur:    0.618)
validation (min:    0.343, max:    0.612, cur:    0.599)

Epoch 00015: saving model to model_weights.h5
448/448 [==============================] - 962s 2s/step
Wall time: 4h 33min 15s
```

Model 3.2

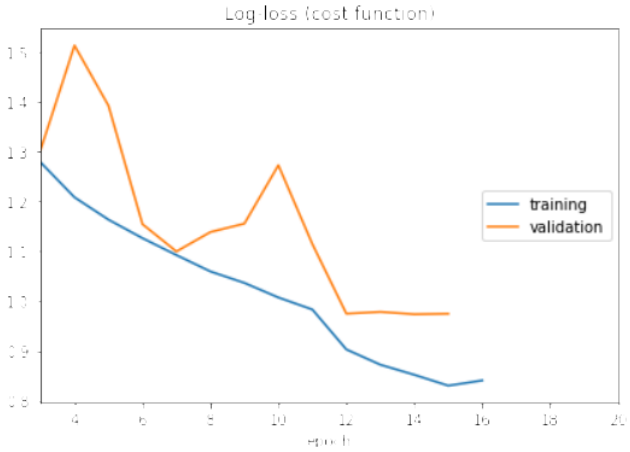
(Similar to 3.1 but different parameters)

Input Images ➔ Conv layer 1 with 128 filters ➔Conv layer 2 with 128 filters ➔ Conv layer 3 with 256 filters ➔ Conv layer 4 with 512 filters ➔ Flatten ➔ Fully Connected layer with 256 nodes ➔ Fully Connected Layer with 512 nodes ➔ Output layer with 7 class nodes

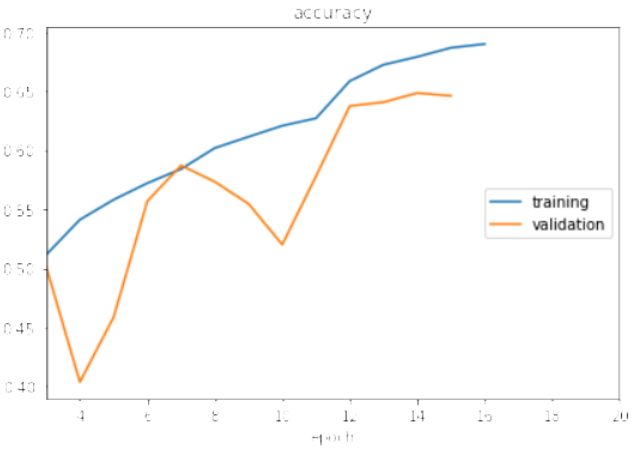
In this model I added more filters in the beginning and reduced number of filters in 3rd Conv layer to reduce redundancy of parameters. This model has **almost half parameters** than model 3.1.

Results and analysis (read clarification point 8)

Log-loss for training and validation after 15 epochs



Accuracy plot for training and validation set for 15 epochs



Min – Max loss and accuracy results

Max accuracy on Train set 69%

Max accuracy on validation set 64.9%

```
Log-loss (cost function):
training   (min:    0.831, max:    1.751, cur:    0.841)
validation (min:    0.974, max:    1.559, cur:    0.975)

accuracy:
training   (min:    0.324, max:    0.690, cur:    0.690)
validation (min:    0.387, max:    0.649, cur:    0.646)
```

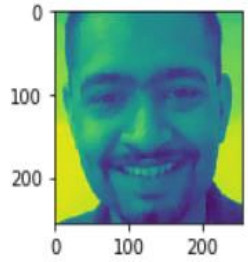
Note:

both the model weights (model_weights.h5 (3.1) and model_weights_new.h5 (3.2) will be submitted in final submission)

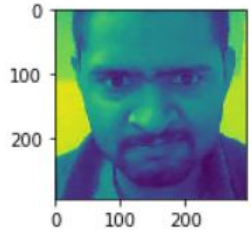
Some example predictions

(These predictions were done on my images to test the model in real world scenario)

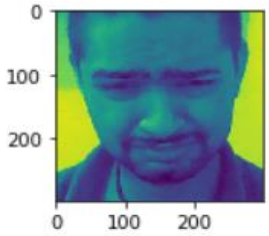
this is a Happy Image



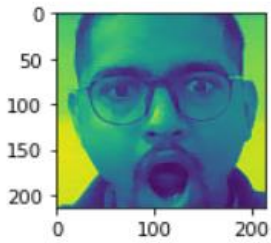
this is a Angry Image



this is a Sad Image



this is a Surprise Image



Real Time prediction in a video

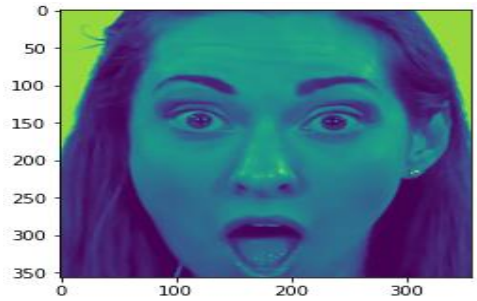
We've implemented a system which fetch frames from an input video and use are model to predict facial expressions in that frame.

The system works as follows

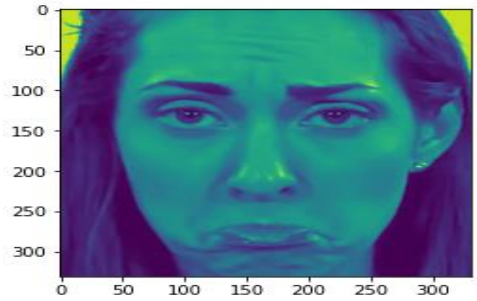
Load model weights and compile model ➔ use cv2 library to read frames from videos ➔ locate the face in the image using open cv haar cascade ➔ apply trained model ➔ Final prediction

Result on facial_exp.mkv file (submitted with final submission)

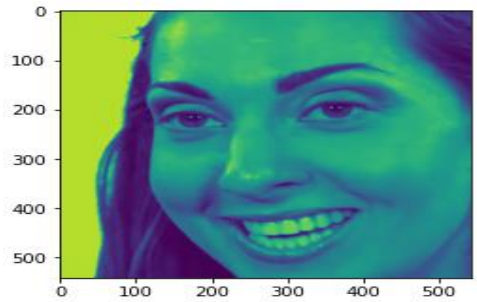
(Note: The below shown frames were fetched from the video)



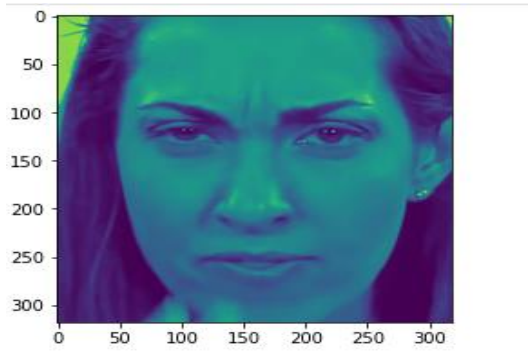
this is a Surprise Image



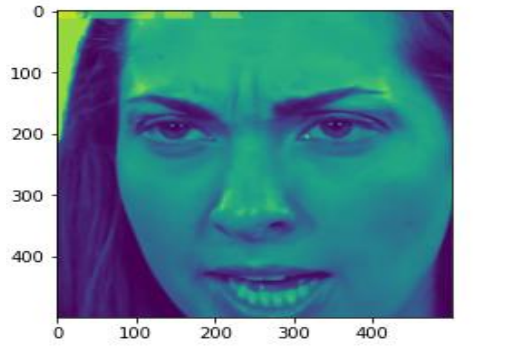
this is a Sad Image



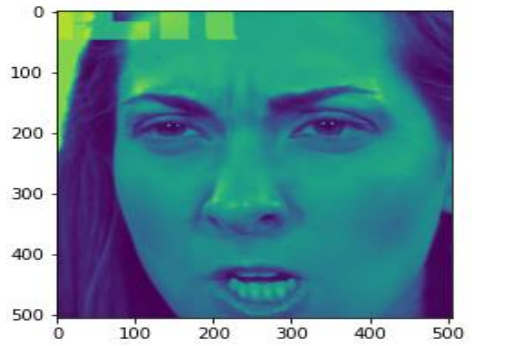
this is a Happy Image



:his is a Angry Image



:his is a Angry Image



:his is a Angry Image

Inference and conclusion

In this project, we implemented a system that can predict the emotion from facial expression in human images. An additional component that predicts facial expressions in the video and in web-cam in real time is also performed.

We explored different Techniques to get best results. At first, we applied the conventional two-way ML technique, which includes feature extraction through transfer learning and then a classifier to get the results. Its performance was poor, which indicates that the problem of detecting facial expression is complex. The accuracy was improved by using an FFNN since it is way more powerful than a random forest classifier. Then later, we proposed a CNN model that did pretty well with our data set and gave good results in real-time prediction as well.

The maximum validation accuracy for face emotion detection with our dataset on Kaggle is 70% (approx.), and we were able to achieve 65% (approx.) with our model. We also tried to generate more data using data augmentation but got similar results.

We came up with idea of using CNN in place of conventional fully connected Multi-layer perceptron-based (MLP) network for this problem due to following reasons –

1) The number of weights in case of MLP grows very rapidly.

2) MLP is not transition invariant.

3) Spatial information is lost when we flatten the input.

4) CNN conserve spatial information.

Overall CNN works well with image classification related problems and one can explore more on their architecture to increase the accuracy.

Individual Contribution in the project

1. Dataset analysis
2. literature review
3. Face detection
4. Design, Implementation, and testing of proposed CNN model.

References

[1] Veena Mayya, Radhika M. Pai*, Manohara Pai M. M. Automatic Facial Expression Recognition Using DCNN

[2] M. A. Ozdemir, B. Elagoz, A. Alaybeyoglu, R. Sadighzadeh and A. Akan, "Real Time Emotion Recognition from Facial Expressions Using CNN Architecture," *2019 Medical Technologies Congress (TIPTEKNO)*, Izmir, Turkey, 2019, pp. 1-4, doi: 10.1109/TIPTEKNO.2019.8895215.

[3] Gaurav sharma Real Time Facial Expression Recognition article on Medium

[4] Shervin Minaee, Amirali Abdolrashidi Expedia Group University of California, Riverside, Deep-Emotion: Facial Expression Recognition Using Attention Convolutional Network

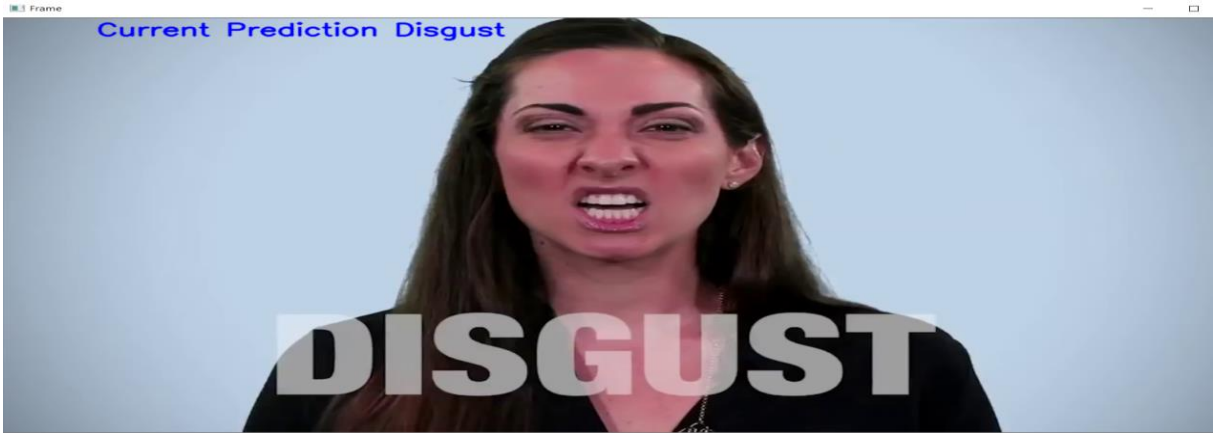
[5] International Journal of Engineering and Advanced Technology (IJEAT), Emotion Recognition from Facial Expression using Deep Learning

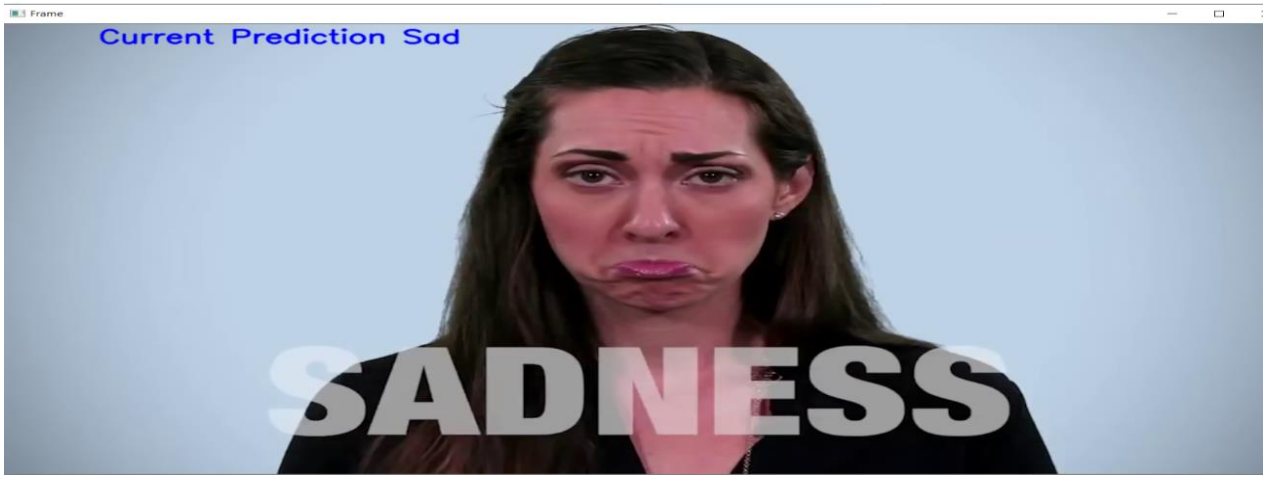
Additional Images to show working of our real time prediction system

1. prediction in a video

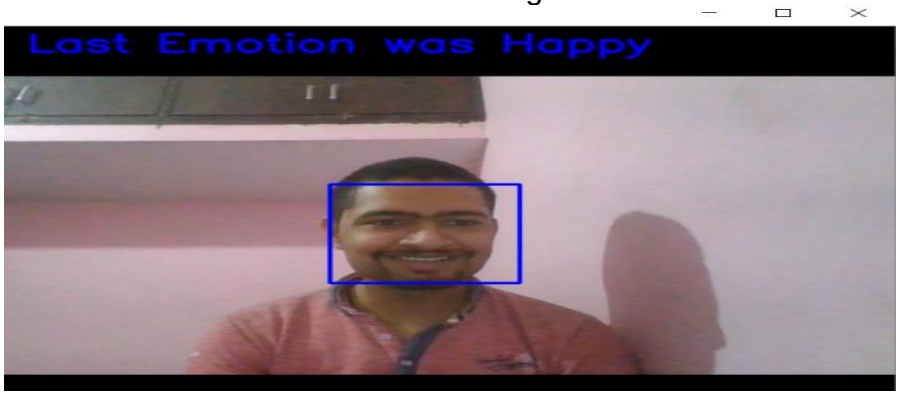
(Note: The blue text is prediction of our model)


Frame
Current Prediction Surprise
SURPRISE


Frame
Current Prediction Disgust
DISGUST


Frame
Current Prediction Sad
SADNESS

2. Prediction in real time through web-cam



Clarification

1. All the references to code used are commented and added with the code.

2. Dataset used from Kaggle is very diverse and sparse.

3. While predicting emotions in real time and in video there is a possibility of lag since computation of label will also take time but frames in a video moves continuously.

4. I've referred to codes provided by A V sir for FFNN.

5. Final submitted code corresponds to the final proposed method (Model 3) and codes for intermediate experiments with learning models are not submitted for more clarity and simple understanding.

6. Final submission contains two weights files
    a) model_weights.h5 corresponds to model 3.1
    b) model_weights_new.h5 corresponds to model 3.2

7. To run the code one need to import all the libraries and compile the model with submitted weights.

8. Plots for model 3.2 are not clear since saved images from jupyter notebook were already faded and it takes around 3 to 4 hrs. to run the code again and generate the plots again.

9. Video used for prediction is also submitted (facial_exp.mkv)