Creating Your Own Real Time Emojis Using Deep Learning

Amit Upadhyay
State University at Buffalo
Buffalo
amitupad@buffalo.edu

Abstract

Emojis have become the new language of today's world. Gone are the days when it was just used for fun, today, its more often used to express ones feeling while texting and thus helps in conveying feeling in more precise way. There was a project on Kaggle in 2016-2017 in which, people were asked to make a model for human emotion detection. The data-set used was Facial Expression Recognition Data-set. There were 7 human emotions categories in data-set namely: angry, happy, sad, scared, surprised, disgusted and natural and people were asked to train their models to detect these emotions. The training set consisted of about 28709 images of 7 categories and test data set had 7000 images. In this project, I am going to take this competition a bit further. Instead of depicting the emotions of a stationary picture, I will be capturing images in real time and in parallel predict their emotions. I will try to use one of the models which we used in project-1, a convolution neural network. As I got good accuracies in VGG-16,the architecture used will also be VGG-16. Of course, as per our learning in project 1, I will try to make lot of changes in this model's architecture) and hence makes sure that it fits to our data-set. Ones I get a good accuracy, (here we assume an accuracy above 60 percent can be considered a good accuracy and will work), I will proceed to the next stage. I will then apply few concepts of computer vision to capture images from real time, (webcam in this case) and in parallel detect the emotions. I will use two face detection techniques, ie open CV's Cascade classifier and Deep learning pre trained model ie MTCNN. We will compare the performance of both of these face detection techniques and based on the outcome, I will select one. Apart from that, I will also discuss the working technique behind these methods in brief and understand what are their advantages and disadvantages. Finally, ones I am able to detect faces , I will send every frame to our previously trained model and predict its emotion. This will be then linked to seven different emojis (which too represent seven emotions) and the emoji corresponding to the detected emotion will be displayed. Please note that its not a major project. The aim of this project is to put in use, the concepts of deep learning and computer vision that I have learned this semester.

1. Introduction

Emotion detection is the process of recognizing or identifying different human emotions to include happiness, sadness, surprise, disgust, fear, anger, neutral, and more.With a change in human's emotion, their body language changes too and this is visible in their facial expressions,speech gesture,movements etc. Deep learning can detect emotions by learning what these body language traits means and apply this gained information to the new set of data and information provided. This is how deep learning helps in emotion detection.Deep learning based facial recognition is a commonly used method for emotion detection. It utilizes the fact that our facial features undergo significant changes with emotions. For example, when we are happy, our lips stretch upwards from both ends. Similarly, when we are excited, our eyebrows get raised.[1] In fact ,when we are stressed, we seem to bite our nails and this is an information feature extractor to detect our current facial emotion. Even simple multi layer perceptrons can be used to successfully detect facial emotions, however,VGG architecture , which contains convolutions layer gives better performance compared to many machine learning architectures. I will compare the the working mechanism of both VGG and a multi layer perceptron to understand why VGG beats MLP.

1.1. Language

All manuscripts will be in English.Graphs and tabular information will be provided when needed.

1.2. Motivation

The paper will start by first giving a detailed information about the data-set used.The source of data set, the train-

[1]Reference -https://bluewhaleapps.com/blog/implementing-machine-learning-for-emotion-detection

test split , the number of classes and even the number of each classes in training data set will be disclosed. Ones a complete information about data-set is given, the focus will be shifted to the model development. 2 different architecture ie MLP(Multi-level perceptron) and CNN (convolution Neural Network) will be used and their performances will be compared and based on that, a model ,to detect the facial emotion will be chosen. After deciding the model, a face-detector method will be needed. Again, two methods, ie OpenCV's Cascade classifier and Multi-Task Cascaded Convolution Neural Network will be used and there advantages and downsides will be checked. A small description about both the methods will be provided. Based on the performance of these methods, one will be chosen to detect faces from a video device (webcam in this case). Ones real time faces are detected, each frame will be sent the previously trained model which will in turn predict its facial emotion. The name of that emotion will be depicted on top of image. At last seven different emojis (each depicting one emotion) will be displayed based on the emotion predicted by model.

1.3. Human Emotions

Although human beings are capable of depicting many emotions,some even can be classified as a combination of few basic emotions, here all 7 basic emotions are used as classes. **These classes are : Anger,Surprise,Happy,Sad,Neutral,Disgusted,Fearful Dec 2020.**

0:"Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"

2. Data-Set

The data-set used for this project is FER-2013(Facial Expression). Its available on Kaggle and was last updated six months ago. This data-set is widely used for training different models to detect facial emotions. It contains 35,887 facial images, each displaying one of seven above depicted emotions. Of these 35,887 images,28709 images are for training with the following composition: Anger-3995,disgust-436,fear-4097,happy-7215,neutral-4965,sad-4830 and surprised-3171 [2] The test set on other hands contain 7178 images with Anger-958,disgust-111,fear-1024,happy-1774,neutral-1223,sad-1247 and surprised-831 .[3]Each image in data set is of dimention (48,48,1). The format is grayscale.

3. Method

As mentioned earlier, the process is divided into five stages: 1. choosing the right model to train on data-set 2. Choosing the right method to capture face from a video device 3. Reshaping the detected face to (48*48*1) by resizing them and graying them to bring them down to a single channel from original three channels. 4.Prediction of emotion of face obtained from previous step. 5.assigning it an emoji based on the emotion obtained.

Figure 1. Few examples of images from train dataset

3.1. Model Selection

The first task in list is to select the right model for our images. The one that can accurately and quickly detect the facial expression of humans after training. In order to get the right model, we need to understand how emotion detection is carried out. What are the steps required to do so .

There are three steps to detect facial emotions and they are : 1.Image Pre-processing 2.Feature Extraction 3.Feature Classification[4]

Its important to understand that no matter what technique or model we use, it needs to carry out the above mentioned tasks in order to correctly identify any human emotion.

In order to select the model, we used two architectures ie, Multi layer perception and a modification of VGG-16 (we removed few layers in order to make model simple as we have small data set and if we use a big architecture, for a small data-set, it will start over-fitting the model. In this section, we will briefly discuss about the architecture and performance of both models

3.2. MLP- model

We used a simple sequential MLP model which has three dense hidden layers and 1 output layers where 7 units, each corresponding to one class of emotion. The input size was (1024,1), which in facts represents the feature form of each image, thus we are manually feeding the MLP with features.

Figure 2 shows the model summary() of the architecture used.

The thing worth noting is the huge number of trainable weight generated even when the model is so simple. This

[2]Reference https://www.kaggle.com/msambare/fer2013?select=train

[3]Reference https://www.kaggle.com/msambare/fer2013?select=test

[4]Reference https://towardsdatascience.com/emotion-detection-a-machine-learning-project-f7431f652b1f

Model: "sequential_7"

Layer (type)	Output Shape	Param #
dense_25 (Dense)	(None, 512)	524800
dense_26 (Dense)	(None, 512)	262656
dense_27 (Dense)	(None, 512)	262656
dense_28 (Dense)	(None, 7)	3591

Total params: 1,053,703
Trainable params: 1,053,703
Non-trainable params: 0

Figure 2. MLP model Architecture

will in turn raise the time taken to train these kinds of models.Also here, we need to give features to model, ie reshaping the (48,48) image to (2304,1) We checked the performance of this model and figure-3 shows a graphical representation of it. even thought I kept the patience of early stopping a bit high ie 8 and min-delta,about 0.005, so may be the performance may go +-5.Finally, for my model, with the given data-set the validation accuracy was around 46-47 percent. Please check figure-3 and figure-4 for further performance related information.

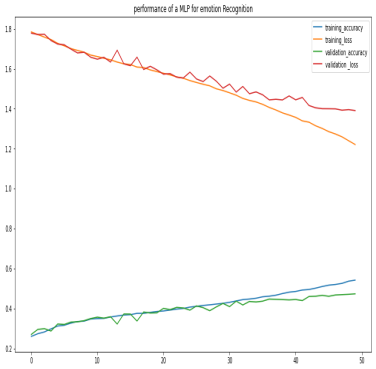


Figure 3. MLP Performance summary

3.3. Convolution Neural Network Model

Initially the plan was to use a pre trained VGG-16 model, however the architecture of VGG-16 is too complicated for this simple data-set. Lets see how my model differs from VGG-16. In order to do so, first lets check the model summary of both models.Figure 5 and Figure 6 show model summary of both architecture.

If we compare both models, I have brought many changes in the original VGG-16 and its hardly recognizable now. First, VGG-16 contains 13 convolutions layers and 3 dense layers. Also the number of channels in first two layers is 64 layers 3 and 4 has 128 channels, layers 5,6,7

Confusion Matrix
```
[[ 210    0  242  177  126  169   34]
 [  29    0   33   21   19    6    3]
 [  71    0  435  146  100  158  114]
 [  53    0  132 1345   84  122   38]
 [  63    0  181  240  479  209   61]
 [  96    0  246  241  206  419   39]
 [  19    0  175   66   27   32  512]]
```
Classification Report

	precision	recall	f1-score	support
Angry	0.39	0.22	0.28	958
Disgust	0.00	0.00	0.00	111
Fear	0.30	0.42	0.35	1024
Happy	0.60	0.76	0.67	1774
Neutral	0.46	0.39	0.42	1233
Sad	0.38	0.34	0.35	1247
Surprise	0.64	0.62	0.63	831
accuracy			0.47	7178
macro avg	0.40	0.39	0.39	7178
weighted avg	0.46	0.47	0.46	7178

Figure 4. MLP Performance summary

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 32)	320
leaky_re_lu (LeakyReLU)	(None, 46, 46, 32)	0
conv2d_1 (Conv2D)	(None, 44, 44, 32)	9248
leaky_re_lu_1 (LeakyReLU)	(None, 44, 44, 32)	0
max_pooling2d (MaxPooling2D)	(None, 22, 22, 32)	0
dropout (Dropout)	(None, 22, 22, 32)	0
conv2d_2 (Conv2D)	(None, 20, 20, 64)	18496
leaky_re_lu_2 (LeakyReLU)	(None, 20, 20, 64)	0
dropout_1 (Dropout)	(None, 20, 20, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 10, 10, 64)	0
conv2d_3 (Conv2D)	(None, 8, 8, 64)	36928
leaky_re_lu_3 (LeakyReLU)	(None, 8, 8, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 4, 4, 64)	0
dropout_2 (Dropout)	(None, 4, 4, 64)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 512)	524800
leaky_re_lu_4 (LeakyReLU)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 7)	3591

Total params: 593,383
Trainable params: 593,383
Non-trainable params: 0

Figure 5. Model summary of used model

No	Convolution	Output Dimension	Pooling	Output Dimension
layer 1&2	convolution layer of 64 channel of 3x3 kernel with padding 1, stride 1	224x224x64	Max pool stride=2, size 2x2	112x112x64
layer3&4	convolution layer of 128 channel of 3x3 kernel	112x112x128	Max pool stride=2, size 2x2	56x56x128
layer5,6,7	convolution layer of 256 channel of 3x3 kernel	56x56x256	Max pool stride=2, size 2x2	28x28x256
layer8,9,10	Convolution layer of 512 channel of 3x3 kernel	28x28x512	Max pool stride=2, size 2x2	14x14x512
layer11,12,13	Convolution layer of 512 channel of 3x3 kernel	14x14x512	Max pool stride=2, size 2x2	7x7x512

Figure 6. Actual VGG model

has 256 and the rest got 512. Then there are 3 dense layers. However in our model, as we got just 7 classes and about 28 thousand training images, we don't need this complicated structure.After various computations and testing different sized layers and varying the number of channels, I

found that the model represented in figure-5 seems to be best suited for my project.I first dropped convolution layers from 5 to 13. And ran the model. The performance was not really good. I decided to further simplify the model and reduce the number of channels to 32 for layer 1 and 2 (from 64) and 64 for layers (3 and 4) from 128. Finally, I even removed a dense layer and significantly brought down the number of units to 512. The output layer had 7 units.[5]Each image in data set is of dimention (48,48,3). The format is RBG. The activation function used in this model is leaky relu. The reason for using leaky relu over widely used relu is that relu sometimes leads to dead neuron problem, however leaky relu overcomes this issue. For relu, the function is f(x) = max(0,x), so relu takes 0 value when the input values are negative. While training a deep neural network, during back-propagation stage, zero values' gradient descents become zero again and they do not converge to good local minimum. It is a dead end situation. However, leaky relu substitutes zero values with some small value for example 0.0001. So, for leaky relu, the function f(x) = max(0.0001x, x). Now gradient descent of 0.0001x will be having a non-zero value and it will continue learning without reaching dead end. Hence,I prefer leaky relu over normal relu activation function. The performance evaluation of CNN model is given in figure 7 and 8.

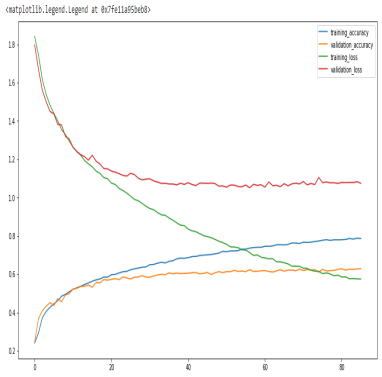


Figure 7. Performance graph of CNN model

Thus after seeing the performance, we can conclude that CNN will be our final model. Not only its giving a good accuracy, but it requires very less trainable parameters(ie 593,383) and we don't have to input extracted features to it. We just input the image and it automatically extracts necessary features. Finally, it can be seen that the accuracy of our model is around 63 percent and we will be using this model to predict emotions of incoming human faces.

[5]Reference the development of this model was inspired from the one in https://data-flair.training/blogs/create-emoji-with-deep-learning/

```
Confusion Matrix
[[ 462    9   81   66  119  190   31]
 [  19   65    6    1    2   16    2]
 [  77    5  418   49  124  241  110]
 [  47    1   46 1479   87   77   37]
 [  71    4   60  124  708  240   26]
 [  89    3  125   94  199  714   23]
 [  13    0   52   34   28   33  671]]
Classification Report
              precision    recall  f1-score   support

       Angry       0.59      0.48      0.53       958
     Disgust       0.75      0.59      0.66       111
        Fear       0.53      0.41      0.46      1024
       Happy       0.80      0.83      0.82      1774
     Neutral       0.56      0.57      0.57      1233
         Sad       0.47      0.57      0.52      1247
    Surprise       0.75      0.81      0.78       831

    accuracy                           0.63      7178
   macro avg       0.64      0.61      0.62      7178
weighted avg       0.63      0.63      0.63      7178
```

Figure 8. CNN Performance Summary

3.4. Choosing the right method for face detection

Initially, the plan was to use haar based Cascade classifier.Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.[6]. However after discussing with the teaching assistant,Mohammad Abuzzar Shaikh, I decided to even try MTCNN,(ie Multi Task Cascaded Convolutional Network, which is a framework developed for both face detection and face alignment.We, however are more interested in face detection. MTCNN work is discussed in the paper "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks" by Zhang, Zhang and Zhifeng. Here, I will try to briefly explain about MTCNN.Please refer to image 9 and 10 for MTCNN face detection example and architecture.

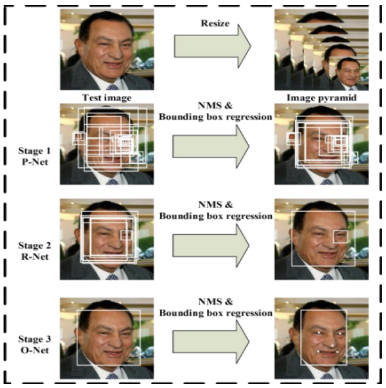


Figure 9. Example of how MTCNN worksl

[6]Reference https://docs.opencv.org/3.4/db/d28/tutorial$_c ascade_c lassifier.html/$

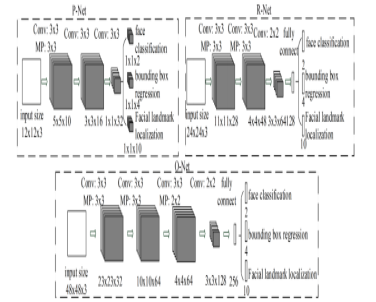
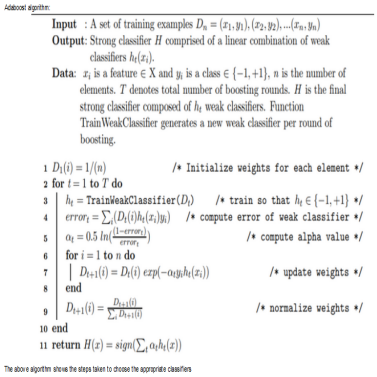


Fig. 2. The architectures of P-Net, R-Net, and O-Net, where "MP" means max pooling and "Conv" means convolution. The step size in convolution and pooling is 1 and 2, respectively.

Figure 10. Architecture of MTCNN



Adboost algorithm:

Input : A set of training examples $D_n = (x_1, y_1), (x_2, y_2), ...(x_n, y_n)$

Output: Strong classifier H comprised of a linear combination of weak classifiers $h_t(x_i)$.

Data: x_i is a feature $\in X$ and y_i is a class $\in \{-1, +1\}$, n is the number of elements. T denotes total number of boosting rounds. H is the final strong classifier composed of h_t weak classifiers. Function TrainWeakClassifier generates a new weak classifier per round of boosting.

$D_1(i) = 1/(n)$ /* Initialize weights for each element */
for $t = 1$ **to** T **do**
  h_t = TrainWeakClassifier(D_t) /* train so that $h_t \in \{-1, +1\}$ */
  $error_t = \sum_i (D_t(i) h_t(x_i) y_i)$ /* compute error of weak classifier */
  $\alpha_t = 0.5\, ln(\frac{1-error_t}{error_t})$ /* compute alpha value */
  for $i = 1$ **to** n **do**
    $D_{t+1}(i) = D_t(i)\ exp(-\alpha_t y_i h_t(x_i))$ /* update weights */
  end
  $D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_i D_{t+1}(i)}$ /* normalize weights */
end
return $H(x) = sign(\sum_t \alpha_t h_t(x))$

The above algorithm shows the steps taken to choose the appropriate classifiers

Figure 11. Ada-boost algorithm in Cascade classifier

The architecture of MTCNN [7]can be divided into three categories(as explained in figure 9). They are P-Net ie proposed net, R-Net ie refinement net and O-Net ie output net. When the image first arrives, the task of P-Net is to create an image-pyramid(ie images of varying sizes) and create windows at the places where there might be possibility of detecting face. Also, it merges over-lapping windows. Refinement net is another,slightly more detailed convolution network and tries to remove the windows that might not contain faces and also merges overlapping windows. Its task to add more refinement. Finally, the output layer takes care of final refinement and removes all the windows that do not have any faces and in addition provides 5 key-points at eye-ball, end of nose and end of mouth. When an image is feed ed to this model, it outputs the cords of bonding box around faces, the key points for both eyes, nose and end of mouth. Also, it gives confidence, which tells how much confident it is about any face detected.

Haar Based Cascade classifier[8] on the other hand is a spacial case of ensemble learning,called boosting where many weak classifiers are combined to make a strong classifier.It involves 4 steps, ie 1.Haar Feature Selection 2)integral image generation 3)Adboost training and finally getting 4)cascade classifier. Here the classifiers are actually filters that are used to catch special features of face. We used integral image generation because it helps in faster computation and proves quite handy in live face detection. Finally, figure 11 gives the algorithm used in Cascade classifier.Please note that we used just the pre trained library for MTCNN and pre trained classifier (xml file) for Cascade classifier.

Although cascade classifier can be only used on frontal image and has high false predictions, for this project, as we need only facial images, it can be used.Even MTCNN has its own dis-advantages like higher the resolution of image, the more chances this model has to mis-identify objects as face.However this issue can be sorted by threshold-ing based on confidence obtained.And important thing to note is , MTCNN requires to install mtcnn library.

However, when we applied both models, MTCNN seemed a better option, because, MTCNN was able to detect and hence predict emotions even in semi-fontal face, here cascade was not able to even depict face when hairs were on forehead. Please check the figure 12 and 13 to evaluate their performances.

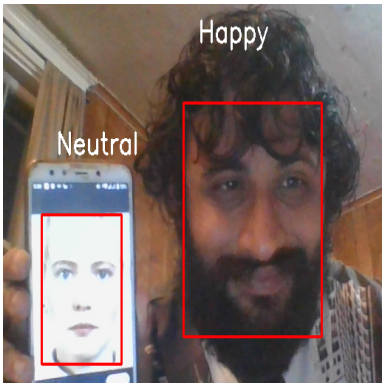


Figure 12. Performance of MTCNN on live imagesl

Please note that when using cascade classifier, I need to lift hairs from my forehead, only then, model was able to detect the face. Also its taking lot of time to detect emotions. Therefore, for the project, we decided to use MTCNN as our face detector mechanism.

3.5. Resizing the input frame

Now, we have the model trained and the mechanism to detect faces(from a video source) ready. However there is one issue, ie the input frame (image) is coming from a live video and its shape is different that what our model needs. Note that while training the model, we specified that it

[7]Reference https://www.youtube.com/watch?v=fUQ1HNCrS7U

[8]Reference https://www.youtube.com/watch?v=ANeuqpI$_V cY t = 15s/$

Figure 13. Performance of cascade model's performance

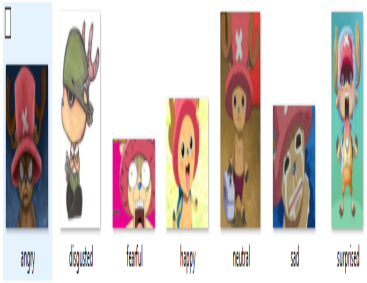


Figure 14. emojis used with their emotions

needs 48*48 gray-scale images. Thus,the incoming frames must be first grayed and then resized to (48*48*1).Then the image will be given to CNN model for emotion prediction. This part is done and shown in code.

3.6. Predicting Emotions

Finally, we have got everything in place. This section will explain how everything will work. We will be using opencv's vidocapture(0) function which automatically turns on the webcam and the screen displays the frontal image. As we know, video is collection of frames. Now, using the face-detector mechanism, we will be looping over each incoming frame and detect the face. The bounding box cords returned will in-turn be used to snip only face area from frame. This snipped section will first be grayed and resized and reshaped to (48*48*1) and then sent to our emotion detection model.The model will detect the emotion and return a vector of length 7 in which , the argmax will be the value which corresponds to the emotion detected. The next section describes the final part of project.

3.7. Assigning emojis to emotion predicted

Figure 14 shows all the emojis we will be using in this project. They were selected to clearly distinguish between the emotions predicted. Please note that these emojis can be changed based on users choice.

Figure 14 shows all the emojis we used in this project. The argmax obtained in previous section will be used to called the value from a dictionary. The values of these dictionary will be these emojis. Thus, this is how, when an emotion is predicted, the appropriate emoji will be displayed. I will be submitting a video which shows how instantly , with change in facial emotion, the corresponding emojis change.

4. Result

As this is a live project,I can not provide any performance related graph or chart. so snippets from the live demonstrated will be shown here to give a sense how of the project worked. Please refer to the images to find the result of this project.

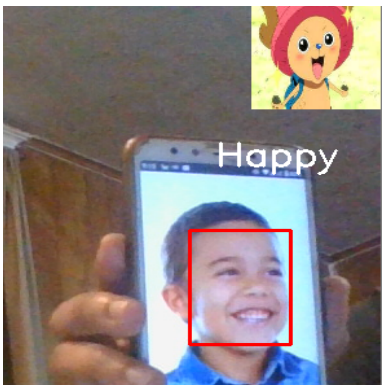


Figure 15. Emotion Happiness getting predicted

5. Conclusion

Thus in the above project, we saw how, using deep learning, its possible to capture human emotions. In fact,in an article on towards data science, it was stated that today's models can predict emotions more accurately than human beings.A MLP does give competition, however is no where near the performance of a deep learning based CNN model. However this marks the journey of many good projects, like how can we detect more emotions.How to detect if someone is both happy and surprised, ie surprisingly happy. Also human face is not the only give outs of emotion. There are various other give away like change in voice, walking

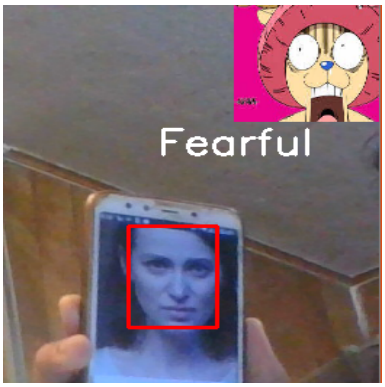


Figure 16. Emotion fear getting predicted

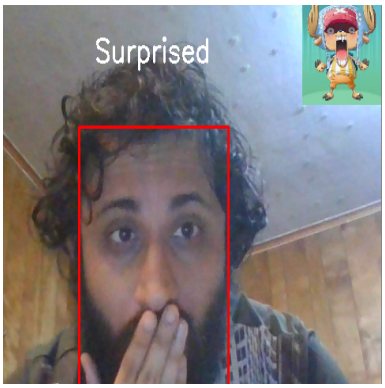


Figure 17. Emotion surprise getting predicted

movements etc. In future, I will try to predict human emotions using voice or walking pattern. For it, may be I will have to use a very different architecture and even the ways to detect the voice etc will be different. However the aim of this project was fulfilled and with various advancements, it can be used in many day today activities.

6. Acknowledgement

I am thankful to my professor Dr Sargur Srihari sir for allowing me to work on this project.Without his approval and motivation, completion of this project would have been just a dream.As there is no research paper on this topic, I had to explore various sources on my own and I would like to thank many people like Ayush Chaturvedi and Michael Allen for running such informative youtube channels and even allowing everyone to freely access and use their github. Without their explanation, this project would have not been possible. Also I would like to thank Data Flair which is an Indian educational institution from where I got this project's idea. Last, but most importantly, I would like to thank my teaching assistant Mohammad Abuzzar Shaikh for meeting up and giving some key ideas that proved to be turning point for this project.It would have been never be possible to complete it without his advice.

7. References

All the references are mentioned at the footnotes.Also the name of people who directly or indirectly helped me in this project is mentioned in Acknowledgement section.