Computer Graphics report

TRAN DUY BAO – ITITIU15076

Face Emotion Detection

I. Introduction

1. Decription

This project is a implementation of face detection and tracking via camera (Webcam, CCTV or other devices) in order to recognize the face on the application UI and keep tracking the motion of the face when your head is moving. Face-detection algorithms focus on the detection of frontal human faces. And on this application , OpenCV is used as an open source library to detect the human face based on Haar classifiers. Then it apply machine learning algorithm (Softmax Regression) to give a prediction of your face emotion (happy, neutral, sad, angry, neutral,..) on the camera with the probability of 90% .

2. Functionalities

+ Detect the face on the camera.

+ Tracking the motion of the face when the head is moving.

+ Give a prediction of your emotion on the screen

3. Techiniques

A. PROGRAMMING LANGUAGE

+ Python 3.6

B. INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

+ Anaconda IDE

C. LIBRARIES

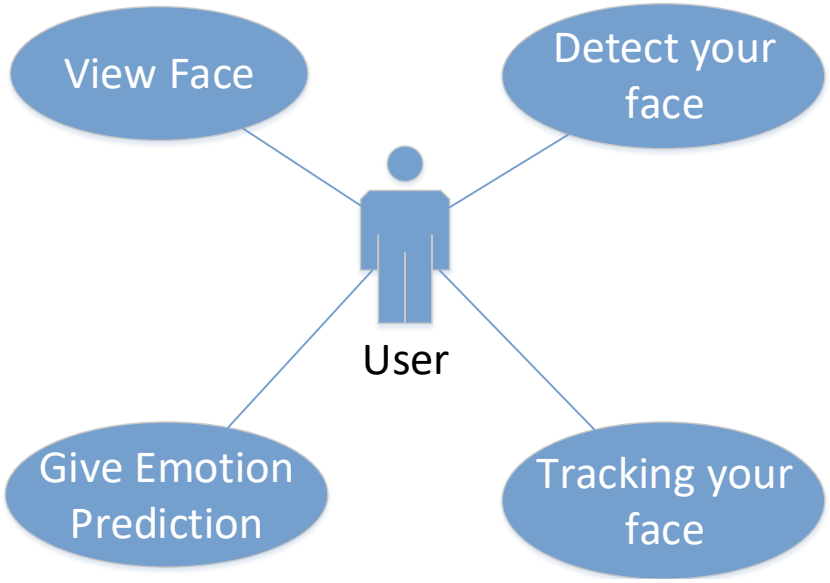
+ OpenCV 3.0

+ Tensorflow

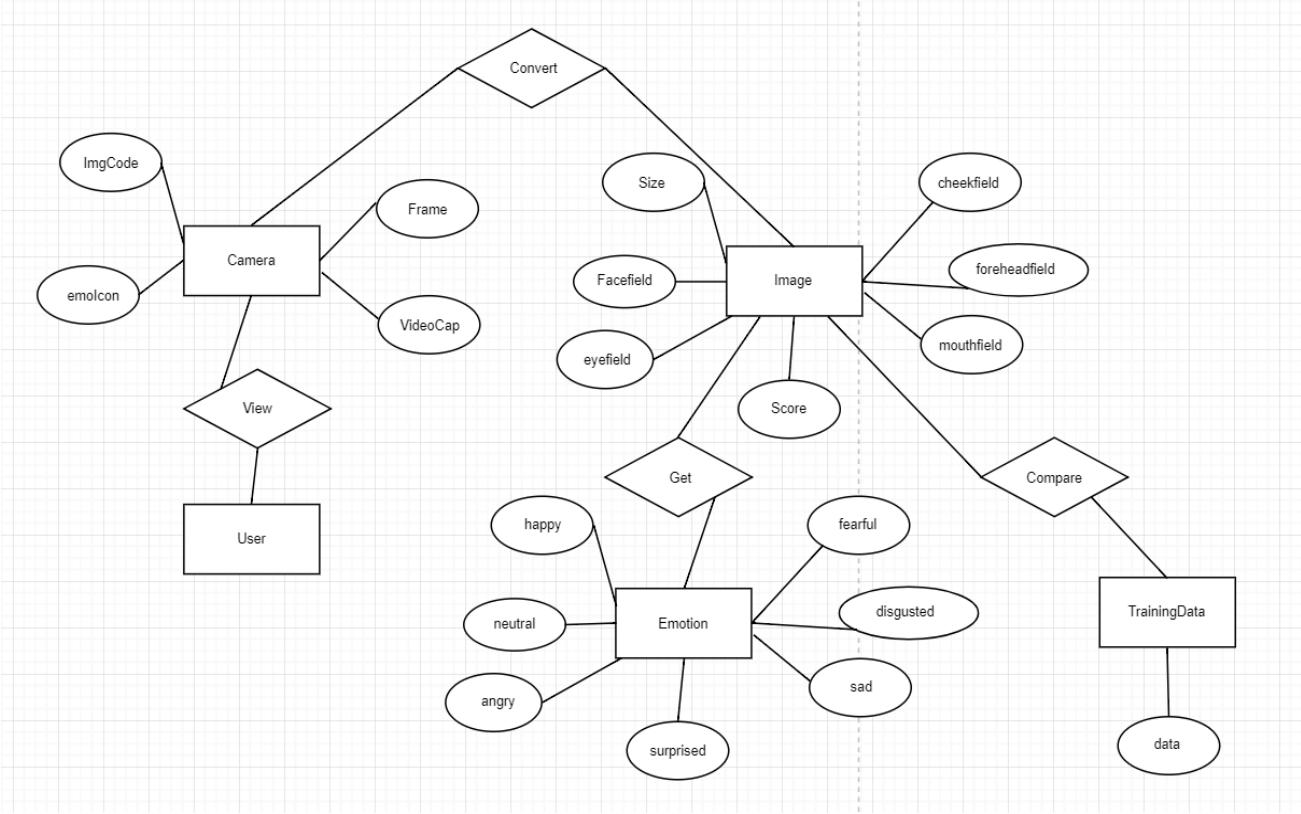
+ TFlearn

II. Design

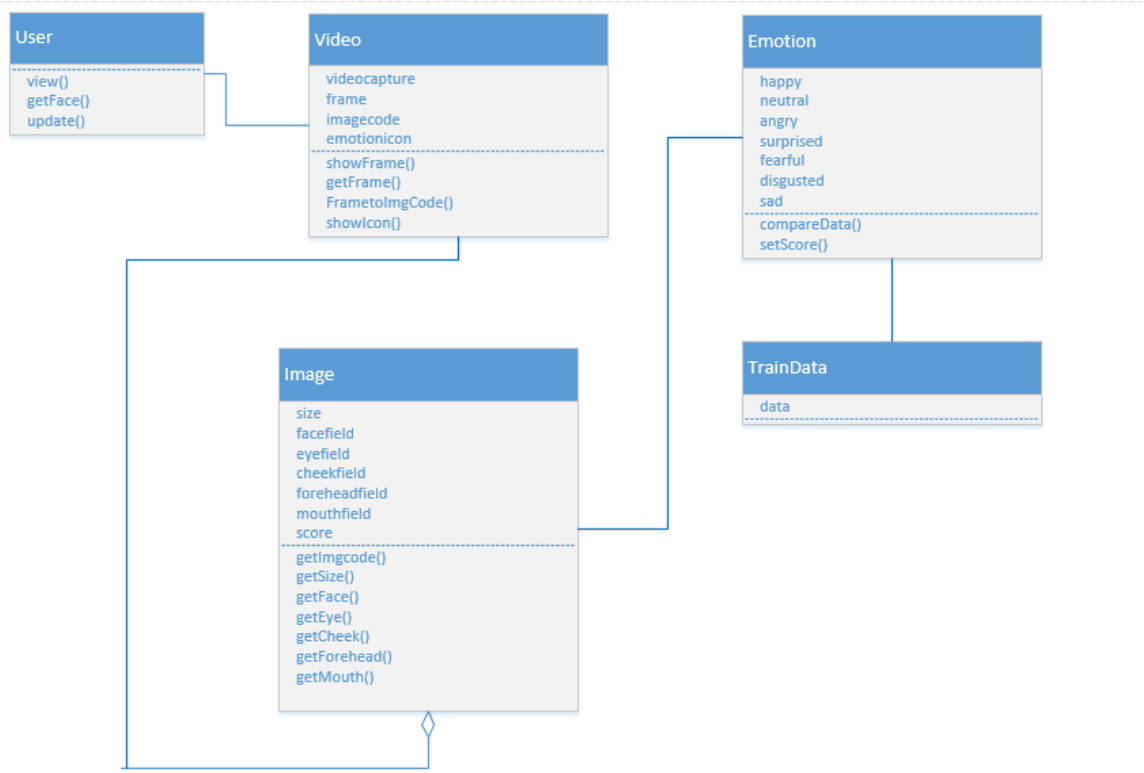
1. Use case diagram



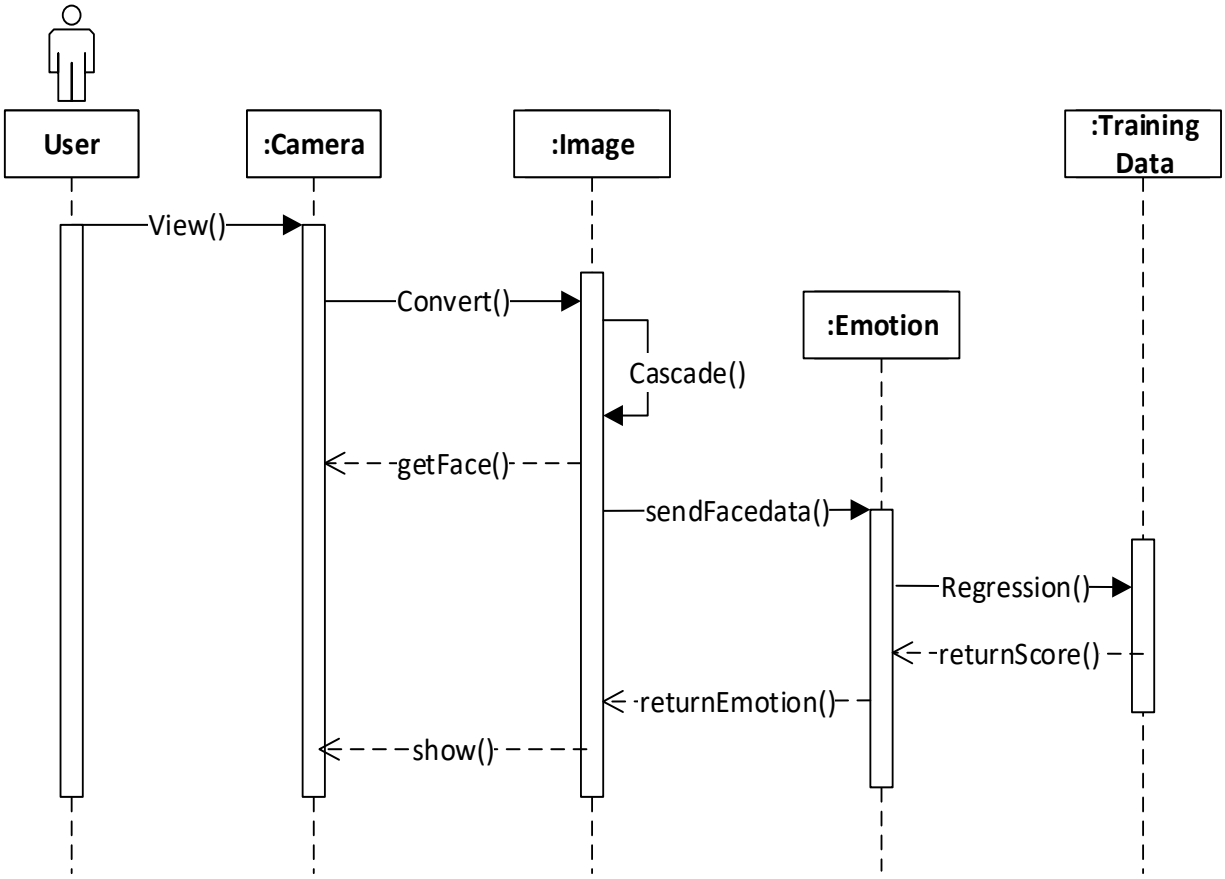
2. Entity Relationship Diagram



3. Class diagram



4. Main Sequence diagram



III. Implementation & Result

1. Implementation

#Begin and initialize the cascade

```
import cv2
import sys
from em_model import EMR
import numpy as np

EMOTIONS = ['angry', 'disgusted', 'fearful', 'happy', 'sad', 'surprised', 'neutral']

# initialize the cascade
cascade_classifier = cv2.CascadeClassifier('haarcascade_files/haarcascade_frontalface_default.xml')

def format_image(image):
    """
    Function to format frame
    """
    if len(image.shape) > 2 and image.shape[2] == 3:
        # determine whether the image is color
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        # Image read from buffer
        image = cv2.imdecode(image, cv2.CV_LOAD_IMAGE_GRAYSCALE)
```

+ The haarcascade_frontface_default.xml is the data to recognize a face by OpenCV

+ the cvtColor : The function converts an input image from one color space to another

#Initialize the first face as having maximum area, then find the one with max_area

```
    max_area_face = faces[0]
    for face in faces:
        if face[2] * face[3] > max_area_face[2] * max_area_face[3]:
            max_area_face = face
    face = max_area_face
```

#Extract ROI of face

```
    image = image[face[1]:(face[1] + face[2]), face[0]:(face[0] + face[3])]

    try:
        # resize the image so that it can be passed to the neural network
        image = cv2.resize(image, (48,48), interpolation = cv2.INTER_CUBIC) / 255.
    except Exception:
        print("----->Problem during resize")
        return None

    return image
```

#Initialize object of EMR class

```
network = EMR()
network.build_network()

cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_SIMPLEX
feelings_faces = []
```

#EMR class

```
class EMR:

    def __init__(self):
        self.target_classes = ['angry', 'disgusted', 'fearful', 'happy', 'sad', 'surprised', 'neutral']

    def build_network(self):
        """
        Build the convnet.
        Input is 48x48
        3072 nodes in fully connected layer
        """
        print("\n---> Starting Neural Network \n")
        self.network = input_data(shape = [None, 48, 48, 1])
        print("Input data",self.network.shape[1:])
        self.network = conv_2d(self.network, 64, 5, activation = 'relu')
        print("Conv1",self.network.shape[1:])
        self.network = max_pool_2d(self.network, 3, strides = 2)
        print("Maxpool",self.network.shape[1:])
        self.network = conv_2d(self.network, 64, 5, activation = 'relu')
        print("Conv2",self.network.shape[1:])
        self.network = max_pool_2d(self.network, 3, strides = 2)
        print("Maxpool2",self.network.shape[1:])
        self.network = conv_2d(self.network, 128, 4, activation = 'relu')
        print("Conv3",self.network.shape[1:])
        self.network = dropout(self.network, 0.3)
        print("Dropout",self.network.shape[1:])
        self.network = fully_connected(self.network, 3072, activation = 'relu')
        print("Fully connected",self.network.shape[1:])
        self.network = fully_connected(self.network, len(self.target_classes), activation = 'softmax')
        print("Output",self.network.shape[1:])
        print('\n')
        self.network = regression(self.network,optimizer = 'momentum',metric = 'accuracy',loss = 'categorical_crossentropy')
        self.model = tflearn.DNN(self.network,checkpoint_path = 'model_1_nimish',max_checkpoints = 1,tensorboard_verbose = 2)
        self.load_model()
```

#Main

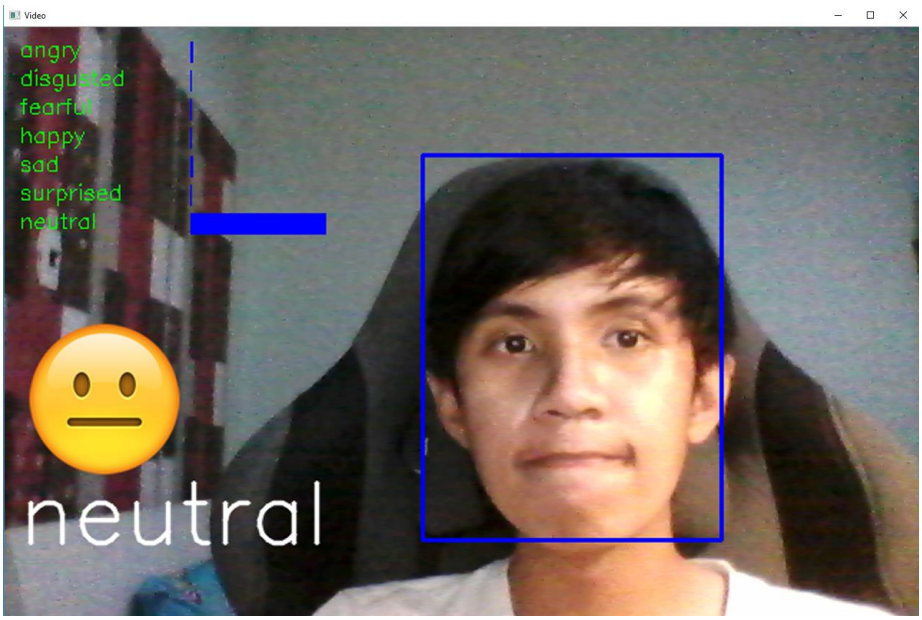
```
# append the list with the emoji images
for index, emotion in enumerate(EMOTIONS):
    feelings_faces.append(cv2.imread('./emojis/' + emotion + '.png', -1))

while True:
    # Again find haar cascade to draw bounding box around face
    ret, frame = cap.read()
    facecasc = cv2.CascadeClassifier('haarcascade_files/haarcascade_frontalface_default.xml')
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = facecasc.detectMultiScale(gray, 1.3, 5)

    # compute softmax probabilities
    result = network.predict(format_image(frame))
    if result is not None:
        if result[0][6] < 0.6:
            result[0][6] = result[0][6] - 0.12
            result[0][:3] += 0.01
            result[0][4:5] += 0.04
        # write the different emotions and have a bar to indicate probabilities for each class
        for index, emotion in enumerate(EMOTIONS):
            cv2.putText(frame, emotion, (10, index * 20 + 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1);
            cv2.rectangle(frame, (130, index * 20 + 10), (130 + int(result[0][index] * 100), (index + 1) * 20 + 4), (255, 0, 0), -1)

        # find the emotion with maximum probability and display it
        maxindex = np.argmax(result[0])
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(frame,EMOTIONS[maxindex],(10,360), font, 2,(255,255,255),2,cv2.LINE_AA)
        face_image = feelings_faces[maxindex]
        print(face_image[:,:,3])

        for c in range(0, 3):
            # the shape of face_image is (x,y,4)
            # the fourth channel is 0 or 1
            # in most cases it is 0, so, we assign the roi to the emoji
            # you could also do:
            # frame[200:320,10:130,c] = frame[200:320, 10:130, c] * (1.0 - face_image[:, :, 3] / 255.0)
            frame[200:320, 10:130, c] = face_image[:,:,c] * (face_image[:, :, 3] / 255.0) +  frame[200:320, 10:130, c] * (1.0 - face_image[:, :, 3] / 255.0)
```

```
    if not len(faces) > 0:
        # do nothing if no face is detected
        a = 1
    else:
        # draw box around face with maximum area
        max_area_face = faces[0]
        for face in faces:
            if face[2] * face[3] > max_area_face[2] * max_area_face[3]:
                max_area_face = face
        face = max_area_face
        (x,y,w,h) = max_area_face
        frame = cv2.rectangle(frame,(x,y-50),(x+w,y+h+10),(255,0,0),2)

    cv2.imshow('Video', cv2.resize(frame,None,fx=2,fy=2,interpolation = cv2.INTER_CUBIC))
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

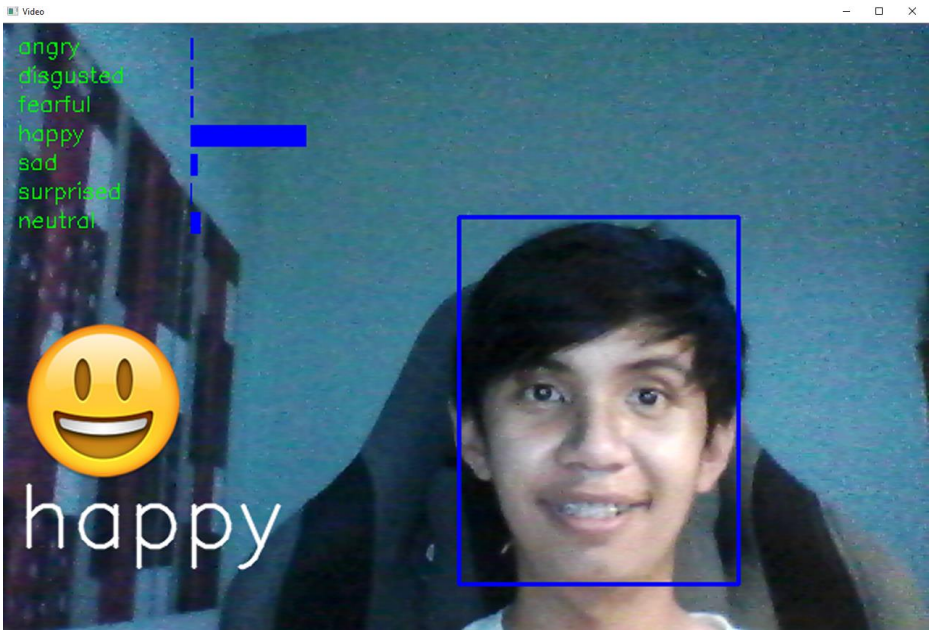
#end of class

2. Result

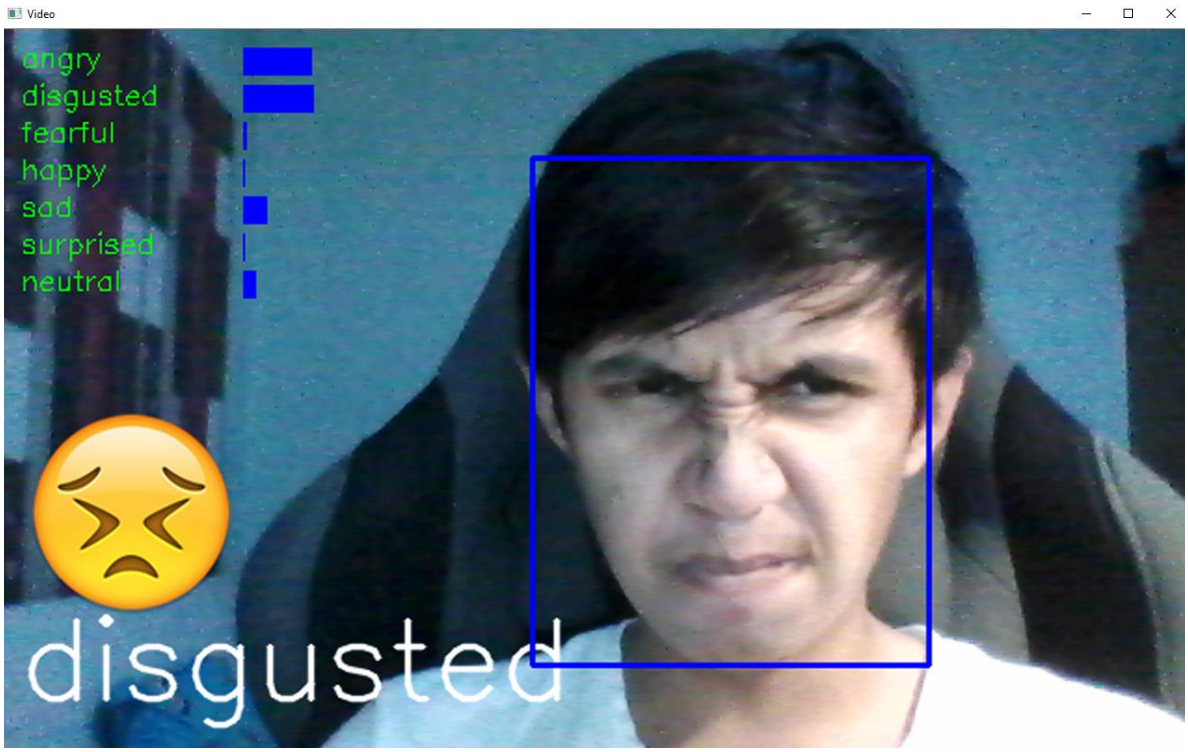
The neutral face :



The happy face :



The disgusted face :



As we can see, the complex emotion is harder for prediction . It is easy to comfuse between disgusted and angry.

IV. Conclusion and Future Work

1. Conclusion

- In the project, OpenCV provides a solution to detecting the face and some parts of the face then tracking the movement. Furthermore, with use of the Haar Feature-Based Cascaded Classifier inside the OpenCV framework, all data is preprocessed. For every image, only the square part containing the face is taken, rescaled, and converted to an array with 48x48 grey-scale values.
- Tensorflow is popular library of Machine Learning which provides many Deep Learing function. The networks are programmed with use of the TFLearn library on top of TensorFlow, running on Python. This environment lowers the complexity of the code, since only the neuron layers have to be created, instead of every neuron. The program also provides real-time feedback on training progress and accuracy, and makes it easy to save and reuse the model after training.

2. Future Work

   Apply in building a complete application which can run on the PC, Website or Smartphone which can capture and recommend some idea based on the real emotion.