Real Time Emotion Recognition via Webcam using Computer Vision and Machine Learning

Final Project Report for "Introduction to Affective Computing"
(CMPSCI 527)
Summer, 2017

Rahul Handa

Abstract

This paper introduces a facial emotion recognition system which uses machine learning to learn the features associated with emotions (viz. "anger", "disgust", "fear", "happy", "sadness", "surprise" and "neutral"). We use Linear discriminant analysis to map the sample vectors of the same class in a single spot and those of different classes are mapped as far apart as possible. This technique is known as computing the FisherFaces. Recent research is trending towards implementing a lot of computer vision techniques like LBP, eigenfaces, etc. and lately using neural networks to implement recognition systems. We need an excellent dataset with faces in various angles and lightning and a lot of computing power to implement those techniques. However, we will be using an open source dataset obtained from Kaggle[1]. Our method takes significantly less time when training with this data as well as making predictions while processing the real time video data.

This software is useful for many applications and can be used on pre recorded videos as well. We can use the classification obtained from our project to make suggestions to the user to improve overall user experience of any activity (example music suggestion while driving, changing difficulty of questions while doing an online test).

Introduction

This paper builds an application to detect emotion in real time by the video feed from the webcam. This project consists of first implementing a face detection algorithm to detect human faces from video feed, and then using supervised machine learning algorithms to classify the detected face on 7 different emotions. We have used the Kaggle dataset to train the model, and tested it using real time webcam video.

There are a lot of existing facial recognition software out there, but they do not add emotion recognition to the features. Mainly, because it is a cumbersome process and consumes a lot of computing power. We are trying to solve this issue and try to tackle whatever problems we face while implementing this system.

This system can be used in many domains. One very important application for this project is to detect student's emotion in an intelligent tutoring system. The aim is to identify, in real time, the affective state of the student, and enabling the tutor to adapt according to the current emotion of the student. For such applications, we need a system with high accuracy in real time.

Related Work

Current state of the art work in this field is highly accurate. A prime example is the Affectiva[7] SDK and API provided by MIT Media Lab start-up Affectiva. They take into account a lot of key landmarks on the face like corners of eyebrows, tip of the nose, corners of the mouth among other thing and analyse them pixel by pixel. The accuracy achieved is in the high 90^{th} percentile.

A landmark paper in this field was the Viola-Jones object detection algorithm[6] proposed in 2001. Our HaarCascade classifier [2] is an advanced implementation of this algorithm. The major motivation behind this paper was face detection though it is now used for all kinds of object detection.

Methodology

We always process videos frame by frame and OpenCV uses machine learning algorithms to search for faces within a picture. For something as complicated as a face, there isn’t one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns/features that must be matched. The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called classifiers.

For something like a face, you might have 6,000 or more classifiers, all of which must match for a face to be detected (within error limits, of course). But therein lies the problem: For face detection, the algorithm starts at the top left of a picture and moves down across small blocks of data, looking at each block, constantly asking, “*Is this a face? … Is this a face? … Is this a face*?” Since there are 6,000 or more tests per block, you might have millions of calculations to do, which will grind your computer to a halt.

To get around this, We are using Haar-Cascade[2] which are like a series of waterfalls(cascade definition). It breaks the classification of faces into multiple stages. For each block in the image it does a quick test. If the test passes, it does slightly more detailed test and so on.

For recognizing the emotion once we find the face, we use Fisher Face recognizer in OpenCV. The method was invented by Sir R.A. Fisher[3]. The idea is that the same classes should cluster tightly together, while different classes should be as far away as possible from each other in the lower-dimensional representation.

Within class differences can be estimated using the within-class scatter matrix, given by-

$$\mathbf{S}_w = \sum_{j=1}^{C} \sum_{i=1}^{n_j} (\mathbf{x}_{ij} - \mu_j)(\mathbf{x}_{ij} - \mu_j)^T,$$

where x_{ij} is the i^th sample of class j, u_{ij} is the mean of class j and n_j is the number of samples in j.

The between class differences are computed using the between-class scatter matrix.

$$\mathbf{S}_b = \sum_{j=1}^{C} (\mu_j - \mu)(\mu_j - \mu)^T,$$

where u represents the mean of all classes.

We now find the basis vector V where S_w is minimized and S_b is maximized, which are given by:

$$\frac{|\mathbf{V}^T \mathbf{S}_b \mathbf{V}|}{|\mathbf{V}^T \mathbf{S}_w \mathbf{V}|}$$

The solution is given by generalized eigenvalue decomposition

$$\mathbf{S}_b \mathbf{V} = \mathbf{S}_w \mathbf{V} \mathbf{\Lambda},$$

where V is the matrix of eigenvectors and $\mathbf{\Lambda}$ is the matrix of corresponding eigenvalues. Refer [4] for further information on this.

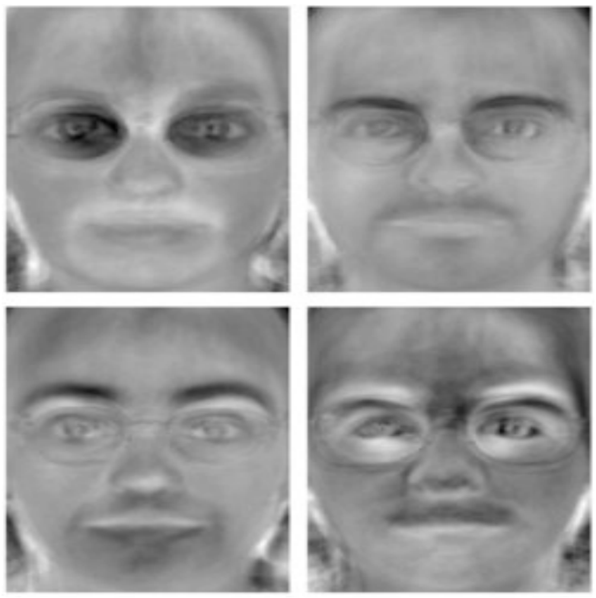


Figure 1: Example of Fisher-faces

Approach

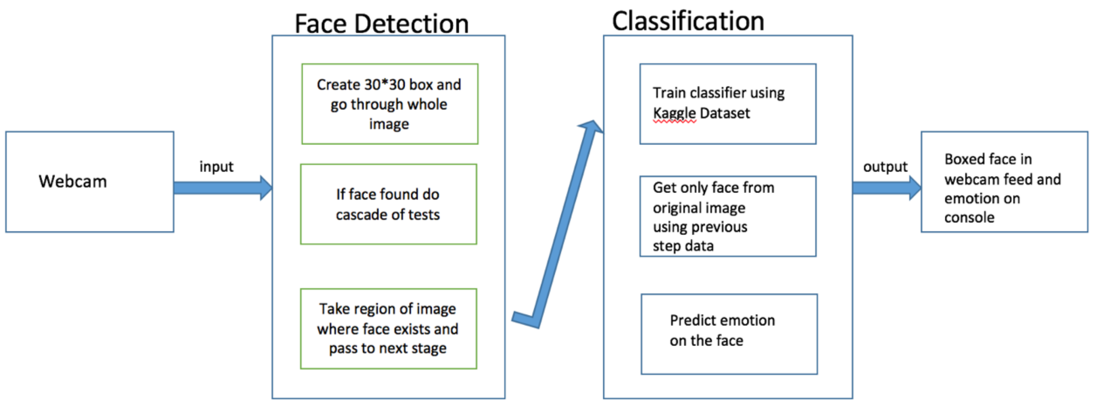

Face Detection
Classification
Webcam
input
Create 30*30 box and go through whole image
If face found do cascade of tests
Take region of image where face exists and pass to next stage
Train classifier using Kaggle Dataset
Get only face from original image using previous step data
Predict emotion on the face
output
Boxed face in webcam feed and emotion on console

Figure 2 Application Design

The complete design of the application is shown in the figure above. The final goal is to predict the emotion displayed by the user correctly in real time using the classifier for each frame of the video.

We can even save this data to a text/csv file for further processing.

Code snippets for the application are given below:

Reading the data:

```
df = pd.read_csv('./fer2013.csv')
df_training = df[df['Usage']=='Training']
df_training['emotion']=df_training['emotion'].astype(int)
training_data = df_training['pixels'].values
training_labels = df_training['emotion'].values
```

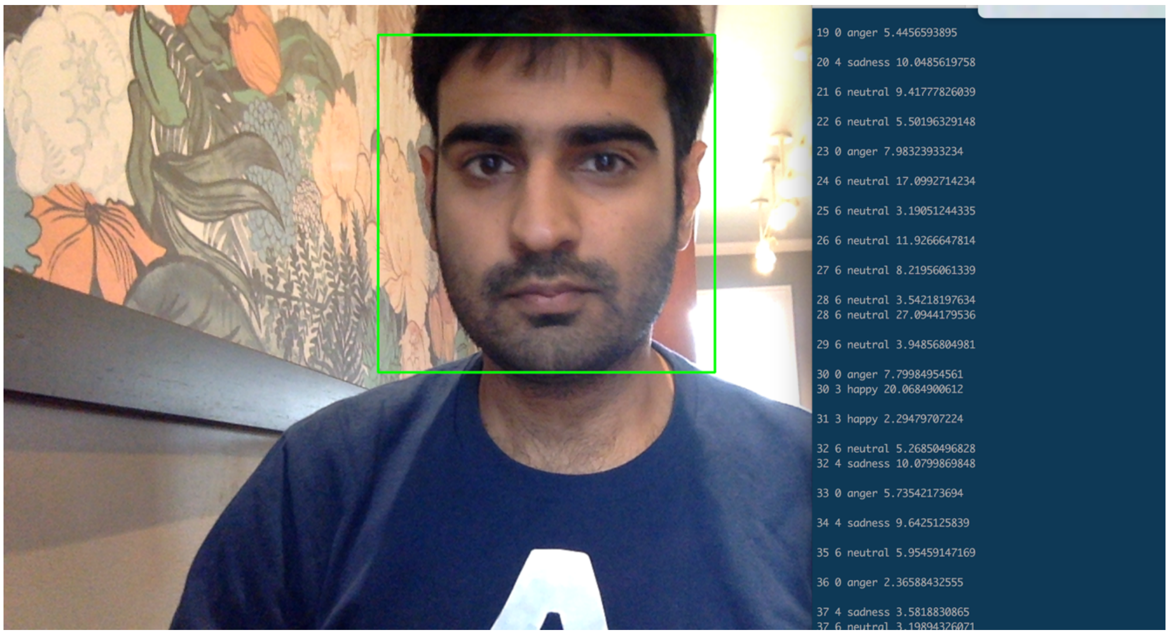
Finding the face:

```
ret, frame = video_capture.read()
 gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


 faces = faceCascade.detectMultiScale( gray,  scaleFactor=1.1,
minNeighbors=5,  minSize=(30, 30),  flags=cv2.CASCADE_SCALE_IMAGE )
# Draw a rectangle around the faces
   if ret:
      for (x, y, w, h) in faces:
         cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

<u>Detecting the emotion:</u>

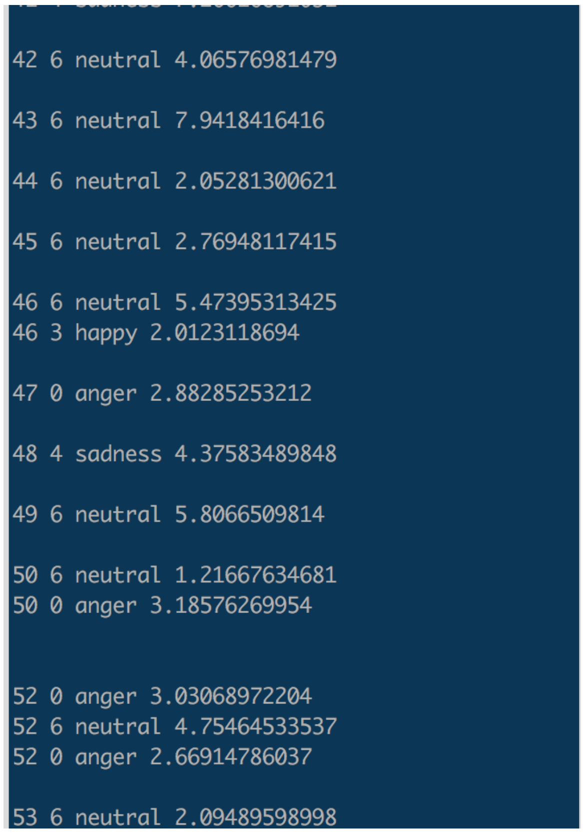
```
roi = gray[y:y+h, x:x+w]
#cv2.imshow('test', roi)
resized_image = cv2.resize(roi, (48, 48))
#histogram normalization
equ = cv2.equalizeHist(resized_image)
cv2.imwrite("./test_images_saved/frame%d.jpg" % counter, equ)

pred, conf = fishface.predict(equ)
print counter, pred ,emotions[pred],conf
```

Following figures show how the app runs



```
19 0 anger 5.4456593895

20 4 sadness 10.0485619758

21 6 neutral 9.41777826039

22 6 neutral 5.50196329148

23 0 anger 7.98323933234

24 6 neutral 17.0992714234

25 6 neutral 3.19051244335

26 6 neutral 11.9266647814

27 6 neutral 8.21956061339

28 6 neutral 3.54218197634
28 6 neutral 27.0944179536

29 6 neutral 3.94856804981

30 0 anger 7.79984954561
30 3 happy 20.0684900612

31 3 happy 2.29479707224

32 6 neutral 5.26850496828
32 4 sadness 10.0799869848

33 0 anger 5.73542173694

34 4 sadness 9.6425125839

35 6 neutral 5.95459147169

36 0 anger 2.36588432555

37 4 sadness 3.5818830865
37 6 neutral 3.19894326071
```

Figure 3 Application at work



```
42 6 neutral 4.06576981479

43 6 neutral 7.9418416416

44 6 neutral 2.05281300621

45 6 neutral 2.76948117415

46 6 neutral 5.47395313425
46 3 happy 2.0123118694

47 0 anger 2.88285253212

48 4 sadness 4.37583489848

49 6 neutral 5.8066509814

50 6 neutral 1.21667634681
50 0 anger 3.18576269954


52 0 anger 3.03068972204
52 6 neutral 4.75464533537
52 0 anger 2.66914786037

53 6 neutral 2.09489598998
```

Figure 4 Predictions done by the classifier

In Figure 3, the webcam feed along with the rectangle around the face can be seen on the left and console on the right displays the emotions detected along with the frame number and the confidence score. As seen in the example the emotion is predicted as neutral for 12/18 frames, i.e., an accuracy of 66%.

Figure 4 shows the predictions done by the classifier along with their confidence value.

Data Pre-processing

This dataset was prepared by Pierre-Luc Carrier and Aaron Courville.

Our classifier works when the training and test images are of same size. So we need to resize the face cut from the video frame to 48*48 pixels.

We lose some information due to the resizing and hence we use **Histogram Equalization** to improve the contrast of our images.

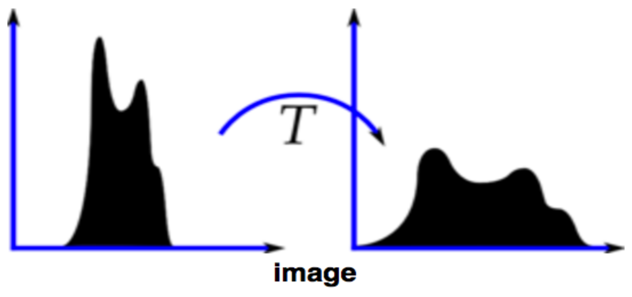


Figure 5 Histogram Equalization

Consider an image whose pixel values are confined to some specific range of values only. For example, brighter image will have all pixels confined to high values. But a good image will have pixels from all regions of the image. So you need to stretch this histogram to either ends and that is what Histogram Equalization does. This normally improves the contrast of the image [5].

Results

The dataset used for training our classifier is not a very great one. It is the only open source dataset which could be found which had the information we were looking for. All the better datasets need special permissions for their usage and this process takes anywhere for 7-10 days.

The dataset has around 29000 images which are classified in this manner:

Emotion	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
No of images	3995	436	4097	7215	4830	3171	4965

We can see that the data is slightly biased (happy/Disgust emotions) and all the images are really small so we resize the test frames accordingly.

When all the dataset is included the accuracy is approx. 50%. We tested the accuracy by manually labelling the emotion for each frame and cross verifying with the predicted value.

Some mistakes are understandable, for instance:

- “Surprise”, classified as “Happy”
- “Disgust”, classified as “Sadness”
- “Sadness”, classified as “Disgust”

This is still much better than a random chance of 1/7 (14%) which is obtained without running any classifier and just guessing the emotion at random

We then pre-process the images before classifying and also train the classifier without the disgust/fear emotions.

We are able to improve the accuracy by a further 10% and now can classify the emotions with 60% accuracy.

Conclusion and Further Work

This paper shows and explains the design and implementation of an emotion recognition system using facial expressions.

We see that it is useful and interesting to see how the computer understands the facial features and what we can do to improve this understanding with the limited resources that we have.

We were able to produce and accuracy of 60% with a not so good dataset and I believe even with the same setup if we use a better dataset we can see improved results.

Another thing that can be done is to use a neural network to classify the emotions and learn the features in various lightning conditions. The neural nets can also make use of infrared information which some of the bigger datasets have. We are currently using eye detection to verify the position of the face in the image. Hence we can use this to add features related to the eyes to enhance the prediction of our emotions.

References:

1) https://www.kaggle.com/c/challenges-in-representation-learning-facial-

expression-recognition-challenge

2) https://en.wikipedia.org/wiki/Haar-like_feature

3) *The use of multiple measurements in taxonomic problems(1936)*

4) http://www.scholarpedia.org/article/Fisherfaces

5)docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.hml

6) Viola, Jones 2001 Rapid Object Detection using a Boosted Cascade of Simple Features retrieved from http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf

7) McDuff et al. Affectiva-MIT Facial Expression Dataset (AM-FED): Naturalistic and Spontaneous Facial Expressions Collected In-the-Wild.