Introduction -

In this project we aim to offer more efficient method for Face recognition and Emotion detection in a way to maximize the accuracy without much latency and error . The Goal is to extract face features and build an emotion detection algorithm. We will be using OpenCV , an open source framework for python.

A key problem in computer vision, pattern recognition and machine learning is to define an appropriate data representation for the task at hand. For this we have implemented Fisherface algorithm which uses PCA (for extracting features) and LDA for classification.

Major advantages of using our approach are:

- It's Fast and reliable.
- Not much computation required(can be run on low power machines)
- Easily portable(just take trained model in xml format and reuse)

In our project we are classifying 6 categories of emotions namely:

Anger Disgust Happy Neutral Sad Surprise

Software Requirements:	Hardware Requirements:
1. Python 2.x 2. OpenCV 3.x 3. Windows/Ubuntu	1. Webcam/Camera

Files and Folders:

1. **Image_normal.py -**
   Script to find all JPG files in path : "data\source_images", find faces in a image then send crop and normalized face to "data\sorted_set" path and

move currently processed file to "data/source_images/done" , if file already processed then print the name and skips.

2. **Prepare_model.py -**

This file contains logic responsible for teaching Fisher model on a processed dataset. It also prints its accuracy.
Normalized images in path "data\sorted_set" need to be classified manually to all the respective emotions folder like-sadness,happy etc.
All the folder name will work as Labels for that emotion and script will do training based on those labels.
Trained model will be placed in "models/" folder.

3. **Emotion-self.py**

Script will show WEBCAM image with emoticons emoji interpolated over face dynamically with position and size. Also put text of emotion over frame ,Major 3 inputs for script to run are :

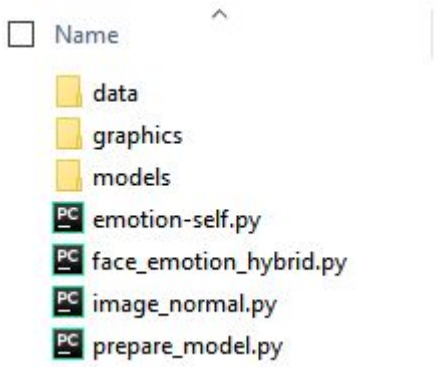
- "models/haarcascade_frontalface_default.xml"- For Face detection.
- "models/emotion_detection_model.xml"- For Expression detection.
- "graphics/anger.JPG","graphics/sadness.JPG" etc- For image interpolation of sample image over face.

4. **Face_emotion_hybrid.py**

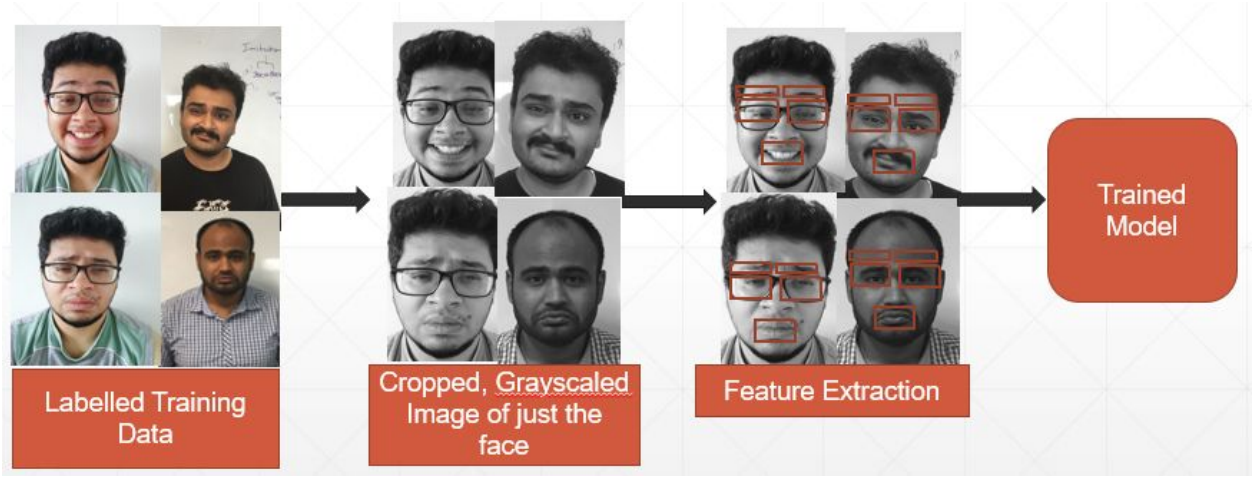
Current script working will be same as previous script with changes that multiple models have been trained and provided input for this and final output will be taken as mode of all models output.
For this script we have trained 5 Fisher faces model viz. Fish_faces0.xml - fish_faces5.xml, and we are classifying expression output from each model, and final result is mode of all results.
Eg. If Output of 5 models are :
Mode [happy,sad,happy,happy,disgust] = happy
So final output will be shown as happy.
Using this scheme our model has become more robust to small changes which can come using single model.

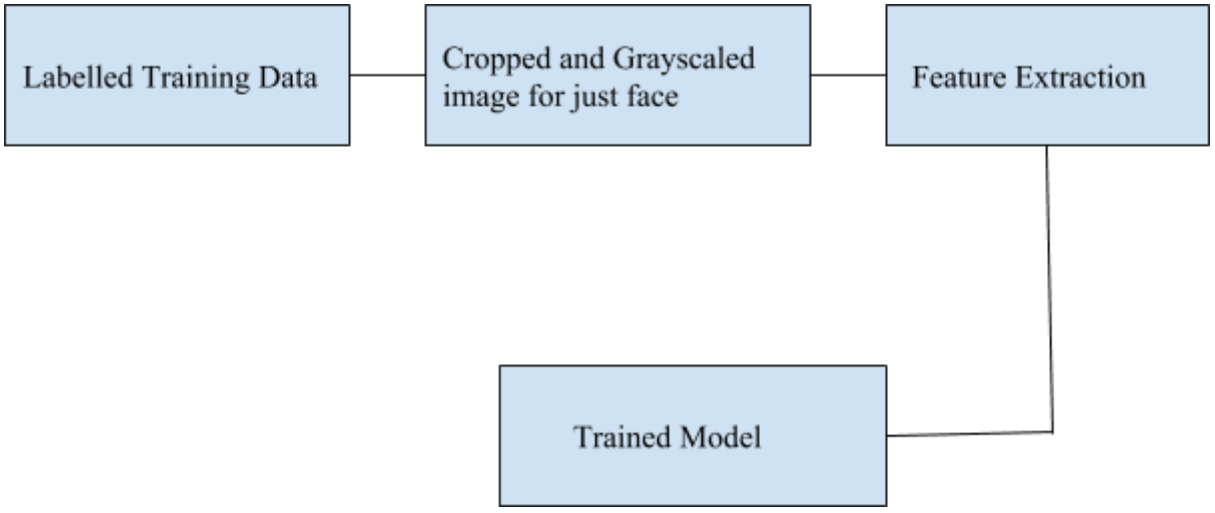
[**NOTE**: Inline comments are also mentioned in each script for better understanding]

Final Folders will be like below:

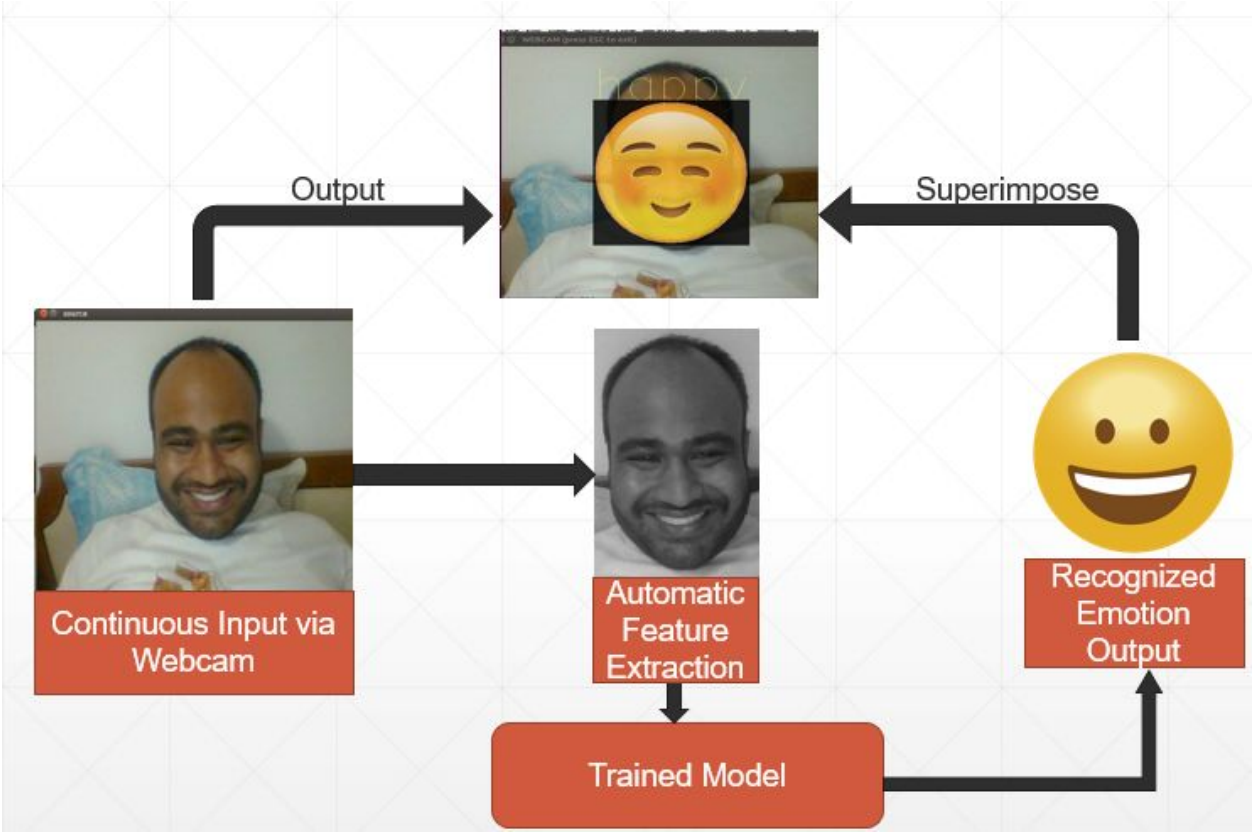


System Workflow -
Training:





Classification:



The database comprises of **220 images** of which 80% form the training set. The Input features and the training data are normalized to a size of 350 X 350 and Converted to Grayscale images.

In order to reduce noise or to produce less pixelated image , Gaussian filter(our case- [5,5]) is used which is done by convolving each point in the input array with a *Gaussian kernel* and then summing them all to produce the output array. This type of filtering also called as linear filters.

$$G_0(x,y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2}+\frac{-(y-\mu_y)^2}{2\sigma_y^2}}$$

Once all the images are normalized for our training , Fisherface Algorithm is applied for face emotion classification

Fisherface Algorithm -

We have a problem of representing data which can be solved by representing input data by finding a subspace which represents most data variance. This can be obtained by applying PCA(Principal Component Analysis) which yields a set of eigenfaces. These eigenfaces are eigenvectors associated with largest eigenvalues of covariance of training data . Since the goal is classification we can use LDA which can find a subspace where the ratio of separation between classes and within classes is higher .

Assumption: Data should be normally distributed

Working-

Consider N samples of Images with class labeled (x1,x2…….,xn) We have to choose the orthonormal vector (W_{out}) which maximizes the ratio of within-class scatter to between-class scatter. Let

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

μ => *Mean image of sample and class* X_i

$$S_W = \sum_{i=1}^{c} \sum_{xk \, \varepsilon \, Xi} (x_k - \mu_i)(x_k - \mu_i)^T$$

$$W_{out} = arg\ max\ \frac{W^T S_B W}{W^T S_W W} \Rightarrow [w1,w2....wn]$$

Algorithm for Fisher Face:

- Construct the Imagematrix `X` with each column representing an image. Each image is a assigned to a class in the corresponding class vector `C`.
- Project `X` into the `(N-c)`-dimensional subspace as `P` with the rotation matrix `WPca`identified by a Principal Component Analysis, where
  - `N` is the number of samples in `X`
  - `c` is unique number of classes (`length(unique(C))`)
- Calculate the between-classes scatter of the projection `P` as `Sb = \sum_{i=1}^{c} N_i*(mean_i - mean)*(mean_i - mean)^T`, where
  - `mean` is the total mean of `P`
  - `mean_i` is the mean of class `i` in `P`
  - `N_i` is the number of samples for class `i`
- Calculate the within-classes scatter of `P` as `Sw = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - mean_i) * (x_k - mean_i)^T`, where
  - `X_i` are the samples of class `i`
  - `x_k` is a sample of `X_i`
  - `mean_i` is the mean of class `i` in `P`
- Apply a standard Linear Discriminant Analysis and maximize the ratio of the determinant of between-class scatter and within-class scatter. The solution is given by the set of generalized eigenvectors `Wfld` of `Sb` and `Sw` corresponding to their eigenvalue. The rank of `Sb` is atmost `(c-1)`, so there are only `(c-1)` non-zero eigenvalues, cut off the rest.
- Finally obtain the Fisherfaces by `W = WPca * Wfld`.

Results -

Total trained	-	213 Images
Miss classification	-	32 Images
Accuracy	-	85% with 182 classified images correctly.

Sample Output:

References-

- http://note.sonots.com/?openfile=EigenFisherFace.pdf&plugin=attach&refer=SciSoftware%2FFaceRecognition
- http://www.bytefish.de/blog/fisherfaces/
- http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html
- Arumugam, Devi & Purushothaman, S. (2011). Emotion Classification Using Facial Expression. International Journal of Advanced Computer Science and Applications - IJACSA. 2. . 10.14569/IJACSA.2011.020714.

- Eigenface based recognition of emotion variant faces

APPENDIX:

ALL FILES need to attach with report.