Emotion Recognition through facial landmark Identification in Frontal Face images

Project By: Hamna Akram (01-132152-008)

Emotion Recognition through facial landmark Identification in Frontal Face images

Project By: Hamna Akram (01-132152-008)

Submitted To: Sir Rafiullah

Project By: Hamna Akram (01-132152-008)

Submitted To: Dr. Amina Jameel

Table of Contents

[1. Introduction 1](#_Toc514604022)

[2. Overview of the Dataset 2](#_Toc514604023)

[2. Face Detection 2](#_Toc514604024)

[3.1. Haar Cascade 2](#_Toc514604025)

[4. Facial Landmark Identification 3](#_Toc514604026)

5[. Emotion Classification 4](#_Toc514604027)

[6. Results 5](#_Toc514604028)

[7. Conclusion 6](#_Toc514604029)

[8. Citations 7](#_Toc514604030)

# Introduction

Facial expressions convey nonverbal cues which play an important role in interpersonal relations, and are widely used in behavior interpretation of emotions, cognitive science, and social interactions. Extracting and validating emotional cues through analysis of users' facial expressions is of high importance for improving the level of interaction in man-machine communication systems. For establishing emotional interactions between humans and computers, a system to recognize human emotion is of a high priority. An automated system that can determine the emotions of a person via his/her expressions provides the machine with the opportunity to customize its response.

Extraction of appropriate facial features and consequent recognition of the user's emotional state that can be robust to facial expression variations among different users is the main aim of this project.

The said system can be used to detect 7 basic emotions namely: anger, contempt, disgust, sadness, happiness, fear and surprise. Additionally, it also detects neutral expressions.

The system is composed of three main steps: face detection/tracking, facial landmark identification, and emotion classification. Choosing suitable feature extraction and selection algorithms plays the central roles in providing discriminative and thoughtful information.

Each submodule has dedicated sections in this report to better describe its’ working. Figure 1 gives the block diagram of the complete system.

Implementation of this system is done in Python using the CK+ dataset.

Figure 1. Block Diagram of the Complete System

# Overview of the Dataset

The input data of this system is the extended Cohn-Kanade(CK) dataset [1]. It is the successor of the Cohn-Kanade(CK) dataset which was released for the purpose of promoting research into automatically detecting individual facial expressions.

Out of 593 image sequences, 315 image sequences from CK+ database are used in our study, because only those sequences have a given emotional class. Figure gives some images as an example of input to this system.



Figure 2. Some Images from CK+ dataset

# Face Detection

This is the first module of our system. It is responsible for detecting faces in images, cropping them out, converting into grayscale and saving them. This is achieved through OpenCV and its built-in methods. To detect a face we use the Haar Cascade classifiers explained below. Once we have detected a face we use the CvtColor() method to convert image into grayscale.

## Haar Cascade

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones [2] and implemented in CV. 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.

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.

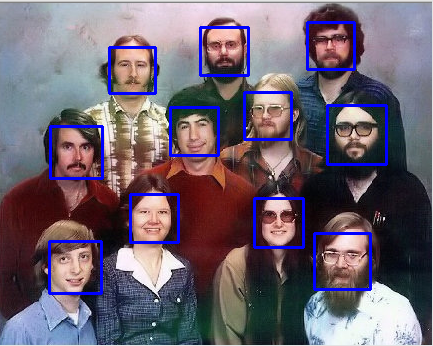
For a face, there are 6,000 or more classifiers, all of which must match for a face to be detected (within error limits, of course). But therein lies the problem: For face detection, the algorithm starts at the top left of a picture and moves down across small blocks of data, looking at each block, constantly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

Figure 3. Output of Haar Cascade Classifier

To get around this, OpenCV uses cascades. What’s a cascade? The best answer can be found from the dictionary: A waterfall or series of waterfalls.

Like a series of waterfalls, the OpenCV cascade breaks the problem of detecting faces into multiple stages. For each block, it does a very rough and quick test. If that passes, it does a slightly more detailed test, and so on. The algorithm may have 30-50 of these stages or cascades, and it will only detect a face if all stages pass. The advantage is that the majority of the pictures will return negative during the first few stages, which means the algorithm won’t waste time testing all 6,000 features on it. Instead of taking hours, face detection can now be done in real time.

These pre-trained Haar Cascade algorithms are included in OpenCV as methods and are used to detect frontal face in our system.

# Facial Landmark Identification

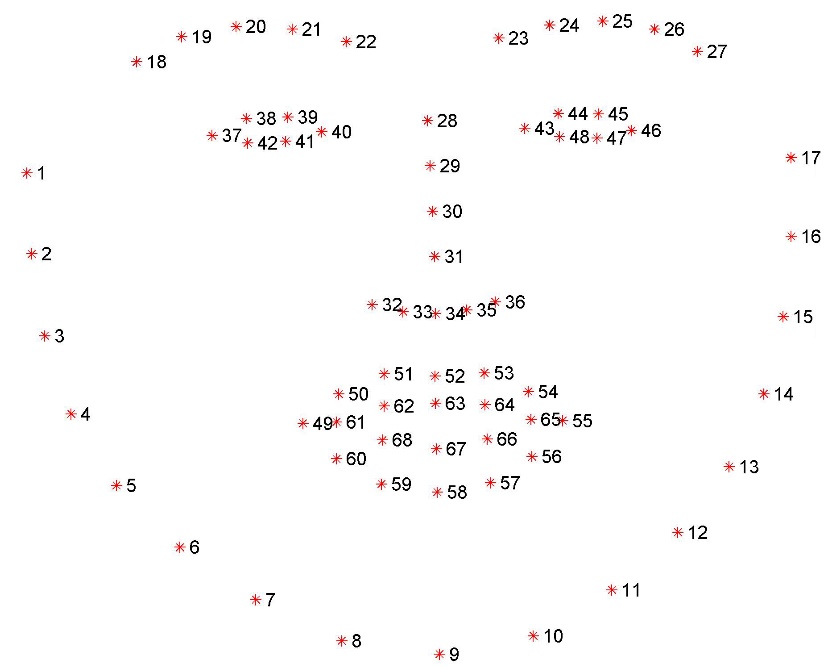
This is the vital block of our system. Facial landmarks are used to label and identify key facial attributes in an image. They help us in classifying images.

For this purpose, I have used the dlib library. The facial landmark detector included in the dlib library is an implementation of the work of Kazemi and Sullivan [3].

This method starts by using:

* A training set of labeled facial landmarks on an image. These images are manually labeled, specifying specific (x, y)-coordinates of regions surrounding each facial structure.
* Priors, of more specifically, the probability on distance between pairs of input pixels.

Given this training data, an ensemble of regression trees are trained to estimate the facial landmark positions directly from the pixel intensities themselves (i.e., no “feature extraction” is taking place).

The end result is a facial landmark detector that can be used to detect facial landmarks in real- time with high quality predictions. The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face.

The indexes of the 68 coordinates can be visualized on the figure 4. These annotations are part of the 68-point iBUG 300-W dataset which the dlib facial landmark predictor was trained on.

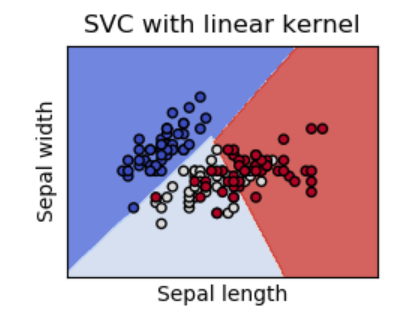
It’s important to note that other flavors of facial landmark detectors exist, including the 194-point model that can be trained on the HELEN dataset. However, dlib provides the fastest face pose estimations.

Figure 4. Visualizing the 68 facial landmark coordinates from the iBUG 300-W dataset

Regardless of which dataset is used, the same dlib framework can be leveraged to train a shape predictor on the input training data — through this property I trained the dlib on our input data, the ck+ dataset.

# Emotion Classification

Once we have marked our facial features in the images we are ready to train our system to predict the afore mentioned emotions. For classification we use the Support Vector Machines.

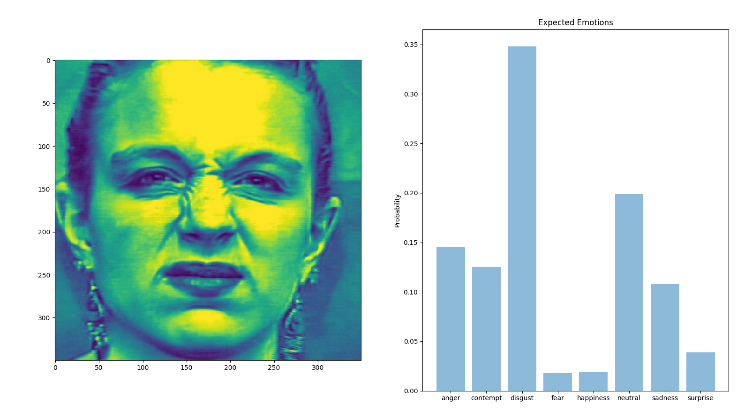
Support vector machines (SVM) exhibit good classification accuracy even when only a modest amount of training data is available, making them particularly suitable to a dynamic, interactive approach to expression recognition.

SVM is a well-known classifier for its generalization capability. SVM classifiers maximize the hyper plane margin between classes. The publicly available implementation of Sklearn.svm is used for the classification of the facial expressions.

The image features belonging to properly labeled emotion databases are extracted in order to generate the file that is used in training the SVM. The method used for classification is a multiclass SVM with a linear kernel, because we aim at distinguishing among seven classes.

Figure 5. SVM with a linear kernel

# Results

To validate our system, we used 20% randomly chosen part of the dataset. Each image goes through before mentioned blocks and creates a matrix where each row corresponds to one prediction, and each column represents a category. To better analyze the results each image is plotted along with the probability chart.

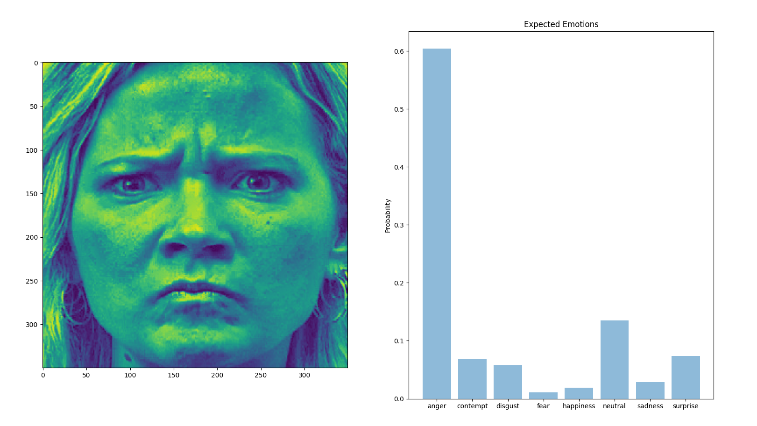
The emotions that offer better results are Surprise, Joy, Sadness and Anger. The results are acceptable when the facial expression reflects Surprise, Disgust and Fear. Emotion contempt provides the falsest positive results, probably because

Figure 6. Correct Detection for Disgust

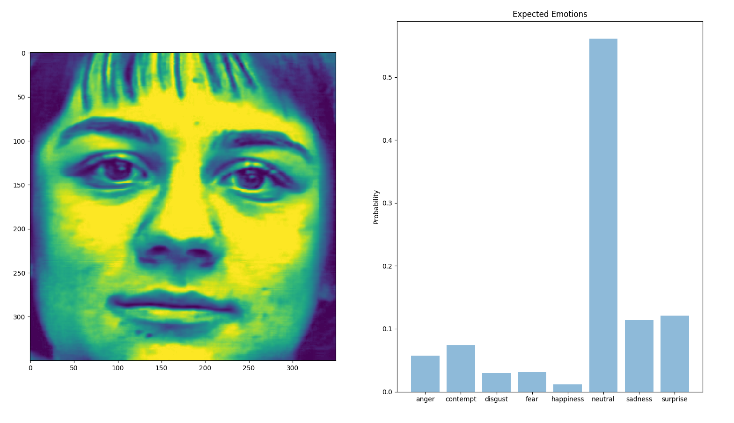
* it contains features that can lead to confusion with other emotions, such as frowning (characteristic of emotion Anger) or lips down (characteristic of Sadness and Fear).
* The dataset has fewer images of contempt, so the training of this particular emotion isn’t reliable.

Figure 7. Correct Detection of Neutral

Figure 8. Correct Detection for Anger

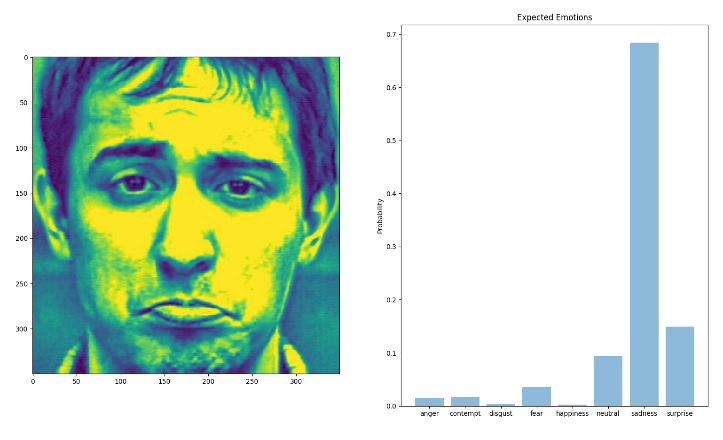


Figure 9. Correct Detection of Sadness

# Conclusion

It is concluded that we have successfully created a system that through facial expressions recognizes emotions. The steps involved in its creation majorly include:

1. **Image Processing:** Detecting faces in images and modifying these images such that they can provide major effect in the next stages. OpenCV’s Haar Cascade functions are used here.
2. **Detection of facial points:** Currently, the detection of emotions is based on the analysis of facial expression from different facial points. The first step is to generate points on a facial expression in the simplest possible way. We use the dlib library for this purpose.
3. **Training and classification**: The third step consists of the selection of images for training, the choice of the most appropriate SVM kernel function, and the generation of a classification model (SVM Linear model) that operates in real-time.
4. **Detection of emotions:** At the last step, an emotion detection system is obtained. It is built from the models generated in the previous steps.

By following these procedures, I have successfully created a system with greater than 80% accuracy. This accuracy was achieved using the SVMs built-in mean score() function.

# Citations

1. Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010). The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression. Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010), San Francisco, USA, 94-101.
2. "Rapid Object Detection using a Boosted Cascade of Simple Features" (2001) by Paul Viola and Michael Jones
3. One Millisecond Face Alignment with an Ensemble of Regression Trees paper by Kazemi and Sullivan (2014).