FEBEI - Face Expression Based Emoticon Identification
CS - B657 Computer Vision

Nethra Chandrasekaran Sashikar - necsashi
Prashanth Kumar Murali - prmurali
Robert J Henderson - rojahend

Abstract
The Face Expression Based Emoticon Identification (FEBEI) system is an open source extension to the Tracker.js framework which converts a human facial expression to the best matching emoticon. The contribution of this project was to build this robust classifier which can identify facial expression in real time without any reliance on an external server or computation node. An entirely client-side JavaScript implementation has clear privacy benefits as well as the avoidance of any lag inherent in uploading and downloading images. We accomplished this by utilizing several computationally efficient methods. Tracking.js provided a Viola Jones based face detector which we used to pass facial images to our own implementation of an eigenemotion detection system which was trained to distinguish between happy and angry faces. We have implemented a similar eigenface classifier in python and have trained a Convoluted Neural Network (CNN) to classify emotions to provide a comparative view of its advantages. We aim to make FEBEI easily extendable so that the developer community will be able to add classifiers for more emoticon.

Introduction
Emoticons have become an essential part on today’s digital communication. Emoticons (also known as Emojis) are used to express the emotions of a person through text in a way that is not possible with just words. The importance of emojis has become so huge that they have even been annotated in a few WordNets as well. Computer Vision has grown to a large scale and every commercial product is using the benefits of high speed image processing and machine learning to interpret meaningful details from the visual world. One of the flourishing directions in computer vision is the identification of human faces and face expression detection. Many social and customer based applications like snapchat and Facebook have started using face expression techniques and are investing heavily in its research. One major component to consider while using customer images and data is privacy. For every action in this area one needs to upload data to a server which can be a bit of concern as image data is sensitive.

FEBEI is a real time face expression detection system which runs on a browser with the help of JavaScript. This system thereby eliminates the need for a server and the data is not uploaded to any server. Hence, effectively it reduces the time taken to classify as the network time is avoided and the privacy of data is ensured. One of the major challenges faced while building such a system is that the system should work in real time without any lag on a browser and not overburden the processor. For this reason, we cannot build a neural network with a million parameters but have instead chosen a more lightweight classifier which will perform a more efficient classification.

Background & Related Work
The key tasks at hand are Face Detection and Face Expression Recognition. For the former task we have various techniques like fisherfaces, eigenfaces, viola jones object detection framework, hausdorff distance etc. A survey paper by Vinay Bettadapura[8] provides an excellent overview of the state of computer vision research as it relates to emotion detection. Many of the primary sources mentioned in the following sections were brought to our attention through this paper.

Facial Action Coding System (FACS)
Ekman and Friesen[1] observed the facial muscles which are important in expressing emotion and compiled their findings to a system of 46 action units (AUs). These action units, some of which are raising of the inner eyebrow and raising of the outer eyebrow, were immensely important in quantifying human expressions. Before this system was published, facial expression research relied heavily on human labeling of example expressions and many researchers were concerned about bias related to cultural context or the labeler’s emotional state at the time. The advent of Ekman and Feisen’s facial action coding system in 1977 put these concerns to rest and quickly became the golden standard.

Facial action units are closely tied to the musculature of the face and can therefore be combined in ways that are either independent of each other or which interact
Facial Action Parameters (FAPs)
Ekman’s Facial action coding system was found to be extremely comprehensive as researchers were able to identify over 7000 combinations of the 46 atomic AUs [4] but it’s important to note that real-life expressions are dynamic and there’s more to them than still images of contracted muscles. Even after the FACS was adopted within the computer vision community, computer animation researchers were struggling to agree upon a system for representing a face in motion. The Motion Pictures Expert Group (MPEG) introduced a well thought out set of facial animation parameters (FAPs) which became standard in 1999.

This system describes a neutral face and a set of 84 key facial feature points. The FAPs are a set of 68 parameters describing the magnitude of movement of these facial feature points resulting from an action unit that is very closely related to those defined by FACS and researchers [6,7] have had some success in mapping the AUs of FACS to FAPs.

Viola-Jones and Haar-like Features Applied to Emotions
In 2004, Paul Viola and Michael Jones developed an extremely efficient face detector with high performance by using an adaboost learning algorithm to classify features derived from Haar-like features. [9] Wang et al applied this method to facial expressions and was able to sort faces into one of 7 archetypal facial expression with 92.4% accuracy. [10] An investigation by Whitehill and Omlin [11] found that the use of Haar features in combination with the Adaboost boosting algorithm was at least two orders of magnitude faster than the more standard classification using SVMs and Gabor filters without a significant drop in performance. More recently, work done by Happy et al [12] in 2015 found that the viola-jones algorithm was valuable not in direct detection of emotions but as a pre-processing step to efficiently detect the most important face regions (IE lip corners and eyebrow edges) which were then further processed with local binary pattern histograms. Their solution performed similarly to other state-of-the-art methods but required far less computational time.

Emotion Detection from Frontal Facial Image[14]
By this approach, segmentation of faces from the rest of the image is done by a skin color model to avoid noise. Facial parts like eye and lip region are extracted from the image by viola-jones algorithm. Emotion detection is then done by analyzing the position of the mouth and eyes in each of the test images.

Novel approach to face expression analysis in determining emotional valence[16]
By this technique image acquisition is done from video. Image is segmented into points of interest such as nose, eyes, mouth. Texture analysis is done based on cheek and forehead activity. Facial characteristic points are based on changes in the gradient, which is the vector sum of color differences in the neighborhood of a pixel. Texture is the brightness difference, color difference, shape and smoothing factors. Based on the texture and features, specific scores are obtained, we classify the emotion as positive or negative.

Real Time Classification of Evoked Emotion[15]
Real time classification of evoked emotions utilizes face, cardio activity, skin conductance and somatic activity. Features are extracted from each of the images. Facial features are taken by feeding the image to Neven Vision along with their cardio activity. Relevant features are taken from the output by using Waikato Environment for Knowledge Analysis (WEKA). Finally, classification is done using two different techniques, Neural network and Linear regression. A comparative study is done with the results from two techniques.

The algorithm that we pick needed to give robust results in real time and Viola-Jones object detection framework best fit this requirement while detecting faces. Our initial efforts were in the view that we can use the same Viola-Jones technique to classify facial expressions, but on further research we found that Viola-Jones did not do a good job on classifying emotions as it did with just detecting the faces. We then proceeded with other methods which can do this real time and came to a conclusion that it is possible with Eigenfaces. The following paper provided us with considerable insight and confidence with this technique.

Eigenface Based Recognition of Emotion Variant Faces[17]
In this process we create a face library and preprocessing is done to get the frontal faces. Eigenfaces are computed for all the training images and classification of test image is done by the following steps:

1. Generate vector of all the images and matrix of vector is created for each image type.
2. Mean of the training faces is created
3. Subtract the test image with each of the mean image.
4. Co-variance matrix is calculated for each of the difference vectors created
5. Eigenvalue and Eigenvectors are computed from each of
the covariance matrix
6. We classify the test image to train image type for which
eigen value is maximum

Model and Experiments

Datasets
The initial hurdle that we had to overcome in our project
was to find a large and well labeled image data set for
facial expressions. Though the internet is flooded with a
large number of face data sets, we could not find well
labeled expression data sets. Data set for this project were
built from the Cohn-Kanade AU-Coded Facial Expression
Database[18] by University of Pittsburgh. We have col-
lected images for two facial expressions happy and sad.
We have built our own data set for tongue out as none of
the image data set has tongue out expression as it is very
specific to social media and chat rooms. We were not able
to collect many facial expression images from any other
resources since each of the image data set available were
all proprietary. All the images used in this data set are grey
scaled image and re-sized to a dimension of $256 \times 256$.

Classification with Viola Jones and Haar Feature

Face expression detection primarily involves detecting
faces in the image frame and applying expression detection
algorithm on faces. Our initial approach involved detecting
faces with viola jones classifier[20] which is a real time
object detection technique.

Viola Jones classifier strictly need frontal face images for
face detection. We detect the face expressions from identi-
ﬁed face frames with facial action units which are computed
by haar technique. FACS detected are then applied with
ADA boosting and classified with cascading classiﬁer. With
this technique we could not achieve signiﬁcant results as it
did not provide ﬁne enough detail to distinguish between
the subtle differences between the same face making happy
or angry faces.

Eigen Emotions

We initially started to search for datasets to use in this
classiﬁcation. To our dismay there wasn’t any reliable
dataset free to download. We tried to obtain the Ohio state
AR dataset and the CMU face images dataset, but both of
these were restricted to only students of those universities.
We then proceeded to collect other datasets like LFW from
University of Massachussets and Cohn-Kanade dataset by
University of Pittsburg. We formed a compound dataset
from these to form 2 types of faces: happy and angry.
We have collected around 300 images for each of these
emotions.

The eigenfaces we formed are with the face recognition
code of scikit learn in python. The face recognition code
is based on the lfw dataset with the lfw.py which handles
finding the faces from the dataset. We have added a new
dataset in scikit learn called face_expressions. We have
added a new handler for this in sklearn datasets called
dataset name to test and train as a text file and the actual image
dataset in localhost and the code downloads it from there. It
processes the set of images and forms the EigenEmotions.
This is then used to classify on the pairs of test images. We
intend to extract the model generated by these EigenEmo-
tions and use them to classify real time in a browser.

For getting this in a browser we have decided to use
a framework called tracking.js. Tracking.js is a computer
vision framework built on Javascript intended to classify
vision based problem on a web browser. It has modules
for integrating a video canvas or an image canvas on the
browser and extracting details from it. The creators of track-
ing.js has added a module for classifying images based on
a viola jones classiﬁcation model. There can be no training
done on tracking.js. We plan to add an additional module for
classifying based on EigenEmotions. The EigenEmotions
model will be extracted from scikit learn and included in
the tracking.js classes. The face will be extracted from the
video stream using the default viola jones classiﬁcation
model in tracking.js and the face region fetched from it will
be tested by the EigenEmotions classiﬁer and a emotion will
be detected. We will have a hash of emoticons associated
with each emoticon and this will be used to identify the
emoticons and sent to the view.

Tracking.js

Tracking.js[19] is an opensource javascript framework with
several computer vision algorithms implemented using
HTML specifications and Javascript. With the arrival of
tracking.js computer vision algorithms are now applied on
real time images which are fed through the computer’s web
cam. It allows us to do detection of colors, faces and facial
features in real time within the browser environment. Track-
ing.js is very responsive as it is built in a very lightcore and
interactive interface. Tracking.js processes images at a high
speed since all the processing happens in the client side and
no lag is introduced from uploading or downloading images.

Tracking.js offers wide range of computer vision algo-
rithms. For example, the color detection function can take
video from live feed and recognize the desired color and
the face detection functionality detects multiple frontal
faces. All these algorithms take input from video cam which
is included with HTML tags and associating a detection
algorithm with it. Tracking.js is available for Chrome,
Firefox, IE, Opera and Safari.
Proposed Model

Our model is an amalgamation of the above experiments and it unfolds in a cascaded manner. The EigenEmotions are trained in Python using the new dataset that was created. The top 150 vectors are chosen and a preliminary classification is produced in python. The PCA component, Mean Image and the fitted training set are exported and placed inside javascript models. A linear SVM is then trained in python using the fitted training set and the test set is classified using that and results are produced. The trained linear SVM model and the PCA model are exported in a pickle format from python.

There is a separate SVM training phase in javascript. The SVM is a modification from svm.js[23]. The fitted training set exported from python is loaded and a linear SVM is trained with a maximum of 100000 iterations. The trained javascript SVM model is exported into another javascript object.

In real time the javascript uses tracking.js, which is invoked on a live video element. The face classifier on the tracking.js is used to find the face region. The face classification system uses Viola Jones classification system to finds the face region and produces a boxed face region detection. This region is then reshaped to the appropriate size that was classified in python. This slice is 50 X 37 pixels. The PCA components and mean image exported from python are then loaded. The image data is extracted from the resized video frame and the mean image is subtracted from it. The resultant image is normalized and fitted with the PCA components. This is then passed to the pre-trained javascript SVM model and a result is obtained. This is then mapped with a happy or an angry emoticon according to the label emitted.

In order to compare this result with the ones in python, a flask application server[24] is created in python. The pickled SVM model and PCA model are unpickled, loaded and readed to classify incoming images. The javascript performs an ajax POST request to local flask server with the pixel data of the resized, grayscaled image that is obtained from the live feed. The python server classifies this image based on the stored models and returns a class label. The javascript uses that class label to display either a happy or an angry emoticon.

In the commercial model the Python evaluation will be hidden and only the JS evaluations are given out. The Python server evaluations are purely for a educative purpose. The javascript classification yielded much better results and is the main motive of this project.

Results & Evaluation

We have a robust static web application with a javascript which can classify emotions. Currently FEBEI can classify 2 emotions, angry and happy. It also maps these emotions to an emoticon and displays it on screen. We can extract the labels from the javascript and use it for other purposes too.

As expected, the javascript implementation of our emotion classification performed more quickly than the python implementation hosted on a flask-based web app. FEBEI was able to classify images at a rate of 11 frames per second which is more than fast enough for our purposes. When the python plugin was used as a classifier, this rate dropped to an average of 7 frames per second.

FEBEI consistently performed better than random guessing when evaluated based on our test set. Our JS expression classifier got an average of 70% of the emotion classifications correct with slightly better performance in recognizing angry faces than happy faces. It’s important to note that the quantitative performance evaluation was conducted using static images. Getting a quantitative measure of accuracy in real time was difficult but, qualitatively, FEBEI worked surprisingly well! Figures 1 and 2 showcase accurate classifications by both FEBIE and our python implementation. Qualitative evidence suggests that the JavaScript implementation outperforms the python version. Even in slightly ambiguous situations where the python model failed, FEBEI was successful (see figure 3 for an example)
The general Python code was evaluated with a one thirds part of the training set the results produced were as below. The Python SVM class produced a 1.0 precision with correct classification of all images. Figure 4 shows the EigenEmotions generated from the training images. Figure 5 shows the classification of the test set. Figure 6 shows the confusion matrix that was produced in python.

In order to evaluate this method with another algorithm we went on to classify the data using the OverFeat CNN[22]. Overfeat yielded a 85% accuracy when compared to the 100% in the EigenEmotions. Figure 7 shows the classification matrix from Overfeat.

Conclusion and Improvements
We have effectively created a system which can detect emoticons from facial expressions using only JavaScript in a browser or python hosted on a web server. We have extended tracking.js framework with a custom library which can detect emotions and map them to the appropriate emoticon. The library provided has easy access to change the classifier parameters and be retrained to even perform a different task. The effectiveness of this system was evaluated against a few other techniques and it was demonstrated that even though there is possibility of better classification in them, this was the best performance that could be achieved given the time constraint.

There are several ways to improve this system. For a start we can include many more emotions and emoticons. The SVM used can be tuned with more appropriate parameters to provide better results. The training data set can be more diverse and training can be done on a color image data set to identify more significant vectors. The JavaScript can be made more elaborate with additions to plugin custom methods on detection of expression.

We feel that robust JavaScript frameworks are the future of commercial software and this system will pace the way for a revolution in the utilization of JavaScript for computer vision. We will continue to work on this system and make improvements to it. We are eager to invite more enthusiastic developers to work on this system and make it even more...
effective. For this purpose we are making the code and data set open source. Our deliverable can be found at the following URL.

https://febei.neocities.org/face_exp.html

References

23. https://github.com/karpathy/svmjs