Image Analysis and Computer Vision Project

Sign Language and Emotion Translator

Yassine Ghannane, Pegah Khayatan
Ecole Polytechnique

1 INTRODUCTION

Our image analysis and computer vision project revolves around sign language interpretation and emotion detection done in real time.
What was achieved is the implementation of an application able to translate ASL(American sign language) characters into what corresponds to numbers and letters from the roman alphabet, coupled with a tool capable of detecting faces in an image and interpreting the emotion displayed, within a range of human emotions.

What motivated our work on these specific topics, was the challenge behind these tasks which cover interesting and new aspects of computer vision (histogram algorithm, CLAHE, face localization), while working on deep learning components and anchoring notions seen in class.
Also, alongside the learning and technical motivation, sign language interpretation is an important piece of technology promoting accessibility for disabled individuals. Such tools have long been a way to offer deaf and hard of hearing people independence in their daily life. However, sign languages being natural and complete languages, they have their own rules of meta-communication. In "classic" languages, an important part of this meta-communication relies on intonation of the voice. Hence, a way around used by sign languages, and ASL in particular, is facial expressions. For example, English speakers may ask a question by raising the pitch of their voices and by adjusting word order; ASL users ask a question by raising their eyebrows, widening their eyes. Thus, emotion detection seemed a good first step to add such considerations to a simple sign language interpreter.

As said before, due to their respective importance, the topic of sign language detection and emotion interpretation have been fairly explored. What we tried via our project is to reach a coherent implementation of a program combining both of these aspects, while at the same time re-implementing some computer vision algorithms (in particular work on CLAHE) .

2 BACKGROUND

Pattern recognition and Gesture recognition are developing fields of research. Basically there are two approaches for sign recognition : vision based and sensor based gesture recognition. Lots of study has been done on sensor based approaches like gloves, wires, helmets etc , but due to disadvantage of wearing it continuously, therefore further work is concentrated on Image based approaches.
Our approach is an image based one, relying on a histogram method to extract and detect hand contours from an image, based on HSV distribution.
As for the emotion detection, we chose face landmark detection as the driving method. Face landmark detection is a computer vision task where we want to detect and track keypoints from a human face, being robust to rigid and non-rigid facial deformations which may be due to head movements and facial expressions.

3 SIGN LANGUAGE

The sign language detection of the implementation can predict 44 characters in the ASL, the predominant sign language of Deaf communities in most anglophone countries, in real time, from a video stream. It is done via a deep learning approach, with convolutional neural networks, classifying hand gesture shapes, extracted from images pre-processed using a histogram algorithm (to which we will give a general description below and more details afterwards). The main implementation parts can be divided as follows :

(1) *Building a histogram* As a first step of the detection algorithm, we need to setup an image histogram adapted to the particular lighting and users' skin tone conditioning the current use of the application. Thus, we build an image histogram describing color distribution of the user's hand by sampling a part of the image only corresponding to his hand, and which we will use afterwards for thresholding and detecting the gesture (implementation-wise, we use a method implemented in Open CV and store the histogram in a file for future use).
(2) *Data set* The initial data set contains 1900 images for each sign recorded manually, to which we add another 1900 images by adding the flipped images around the vertical axis (to handle both right and left handed users). We then take the dataset and divide it into three subsets. A brief description of the role of each of these datasets is as follows: A train dataset used for learning (by the model), that is, to fit the parameters to the machine learning model. A vaidation set used to provide an unbiased evaluation of a model fitted on the training dataset while tuning model hyperparameters (it can also play a role in other forms of model preparation, such as feature selection, threshold cut-off selection.). And finally, a test set to provide an unbiased evaluation of a final model fitted on the training dataset. The ratio employed was 5/6 : 1/12 : 1/12
(3) *CNN Model* The CNN model used in the sign detection is formed by a stack of 2D convolution layers and max-pooling layers, followed by 2 fully-connected layers, the latter giving as output the final classification. This structure follows a

fairly common pattern in simple convolutional networks; building a convolution basis with pairs of convolution and pooling layers and doing the final classification with dense layers.The last layer relies on a Softmax activation function adapted to the multi-classification context.
The model used was a pre-trained one. Building a large enough and clean dataset, and training the model from scratch revealed to be too much of a time consuming task, and we preferred using an already functional model upon which we would build the rest of our work.

3.1 A primer on sign language

American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. The exact beginnings of ASL are not clear, but may originate from the intermixing of local sign languages and French Sign Language (LSF, or Langue des Signes Française). Today's ASL includes some elements of LSF plus the original local sign languages; over time, these have melded and changed into a rich, complex, and mature language. Modern ASL and modern LSF are distinct languages. While they still contain some similar signs, they can no longer be understood by each other's users.
ASL is a language completely separate and distinct from English. It contains all the fundamental features of language, with its own rules for pronunciation, word formation, and word order. While every language has ways of signaling different functions, such as asking a question rather than making a statement, languages differ in how this is done. Fingerspelling is part of ASL and is used to spell out English words. In the fingerspelled alphabet, each letter corresponds to a distinct handshape. Fingerspelling is often used for proper names or to indicate the English word for something.This is the part implemented in our project.

3.2 Histogram algorithm for hand detection

A histogram is used to count or visualize the frequency of data (i.e. the number of occurrences) over units of discrete intervals, called bins. Histograms have many applications within data and image processing by using it as a threholding tool. This threshold value can then be used for edge detection, image segmentation, and co-occurrence matrices.
Such a histogram represents the distribution of pixel intensities (whether color or grayscale) in an image. It can be visualized as a graph (or plot) that gives a high-level intuition of the intensity (pixel value) distribution. In our case we consider the hue and saturation to build the histogram, on the part of the image corresponding to the hand. The reverse function which corresponds to obtaining the part corresponding to hand from a given histogram is done by what is known as Back Projection : In each pixel of our image $p_{i,j}$, we collect the data and find the correspondent bin location for that pixel $h_{i,j}, s_{i,j}$, we then lookup the model histogram in the correspondent bin, read the bin value and store it in a new image (Back Projection). Also, it is important to normalize the model histogram first, so the output corresponds to visible values. In terms of statistics, the values stored in the back projection image represent the probability that a pixel in input image belongs to a skin area, based on the model histogram that we use. For instance in our output image, the brighter areas are more probable to be skin area, whereas the darker areas have less probability.

3.3 Convolutional neural network

The model solicited in this part of the project works with several features proper to convolutional neural networks, which we will describe more thoroughly in this section.

(1) *loss function* For this multi-classification task, the loss function chosen was categorical cross-entropy, which helps quantifying the difference between two different probability distributions. The categorical crossentropy calculates the loss of an example by computing : $-\sum_{i=1}^{n} y_i \cdot log(p_i)$ where n is the number of classes, y_i is the i-th value of the target vector value and p_i is the i-th prediction value with $\sum_i p_i = 1$. We can clearly see that the loss is 0 if and only if the prediction vector coincides with the target vector.
(2) *over-fitting* A fully connected layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to over-fitting of training data. Thus we add a dropout layer which ignores units (i.e. neurons) during the training phase of certain sets of neurons which is chosen at random; these units are not considered during a particular forward or backward pass.

4 EMOTION DETECTION

The process of emotion detection in our implementation can be decomposed into four parts: face location determination, facial landmark detection, feature extraction and emotion classification.

(1) *Image classifier:*
  (a) Dataset class: to handle the landmarks which are stored as a csv dataset. In this dataset we have 136 landmarks and a label that corresponds to the matched emotion. We will explain later what is the structure and idea behind a landmark and how it functions as a signature for different facial expressions. By handling the dataset we mean accessing the part used for training and labels separately.
  (b) The other class here is ImageClassifier. This class is initialized by extracting the features (landmarks) that are already provided by using the Dataset class, and then training two types of classifiers: Random forest and SVM. The methods in this class use another python code: landmarker.py. This latter provides us with methods to extract the rectangle containing the face and its landmarks. This information is then used to classify the current image. Other than making predictions using the landmarks, this class contains two functions that will be used to draw the rectangle and the landmarks on the video that we capture.
  (c) The dataset used to train the classifier is from this github.
(2) *Dataset transformer:* This code transforms the outputs of dlib library functions and makes them compatible with OpenCv data structures. In other words, functions in this code translate the data structures from dlib to the conventional structures. This is needed as we use OpenCv to interact with the camera. The more complicated function here is "point to

vectors" which creates landmarks from the landmark points. We later explain how this encoding is performed.

(3) *Data land marker:* This is the core part of the emotion detection side of our code and does the two first tasks defined earlier: face location determination, facial landmark detection. To perform these tasks it uses the dlib library. The landmark detection of this library is based on the paper [4].

4.1 Few concepts of Image Processing

4.1.1 Histogram equalization. In the preprocessing of the image, we perform a histogram equalization operation on it. The goal is to make the faces independent of their environmental lighting. Before explaining the method we used, we take a closer look at the math behind it: The method is called Histogram equalization and adjusts the contrast using the image's histogram. The histogram of an image plots the luminance distribution of pixels in an image. To be more specific, the horizontal axis is for the luminance and the vertical axis stands for the number of pixels. A good image will have pixels from all ranges of luminance. Hence, one way to improve the image is to stretch the histogram so that it would cover a wider range of values for luminance. This stretching is equivalent to a mapping that can be performed on the histogram and the image. The result of this stretching is even more obvious when the histogram is narrow, meaning that the majority of the pixels have the same brightness; stretching would make small differences more significant and so the contrast increases. This operation is called Histogram equalization and has found many applications in x-ray images and also for capturing better details in photographs that are either over or underexposed. We still have to specify what kind of mapping is performed over the histogram. This is the histogram equalization function that specifies the operation. This usually should be an injective function (as its application is to widen the range of histogram) and so the operation is invertible knowing this function. Examples of such methods include adaptive histogram equalization, contrast limiting adaptive histogram equalization or CLAHE, multipeak histogram equalization (MPHE), and multipurpose beta optimized bihistogram equalization (MBOBHE).
To implement:
We perform the following operations on a grayscale image. This only contains the information about the brightness of pixels. It is possible to implement this part with the formula

$$imgGray = 0.2989 * R + 0.5870 * G + 0.1140 * B$$

. In the gray image we have L ranges of brightness. Let us call the original image r and the transformed one s and the distribution of pixels on each image $p_r(i)$ and $p_s(i)$.

$$p_r(i) = p(r = i) = \frac{r_i}{r}$$

$$p_s(i) = p(s = i) = \frac{s_i}{s}$$

Consider the transformation $s = T(r)$. s should satisfy the condition: $p_s(i) = \frac{1}{L-1}$. Here L is the number of levels/ranges that we have in the grayscale image. Usually $L = 256$.
We can prove that the following transformation makes the image s satisfy the imposed condition:

$$s_k = T_k(r) = round(cdf_r(k)) = round((L-1)\sum_{i=0}^{k} p_r(i))$$

$$k \in \{0, 1, ..., L-1\}$$

Meaning that the pixels with intensity k get mapped to the above expression. The round operation is performed because the $cdf * (L-1)$ might be a non-integer. We can explain this result by considering the intensity of two images, original and the transformed one, as continuous functions, instead of functions that take values in $\{0, 1, ..., L-1\}$.

$$s = T(r) = (L-1)\int_r p_r(w) d_w \tag{1}$$

$$p_s(s) = p_r(r) * \left\| \frac{d_r}{d_s} \right\| \tag{2}$$

But we know that $\left\| \frac{d_r}{d_s} \right\| = (L-1) \cdot p_r(r)$ and so $p_s(s) = \frac{1}{L-1}$. This is the condition that we wanted the transformed image to satisfy.
The disadvantage of this method is that it does not discriminate between contrasting the noise and the real content of the image and might inject some irregularities to the photo.
To partially overcome this flaw, there is another similar technique called CLAHE (Contrast Limited Adaptive Histogram Equalization).

4.1.2 CLAHE. In the last section we considered the global contrast of the image and tried to widen its intensity histogram. This however might cause loss of information in some cases; for instance, when there is a very bright object, the information on this object may be partially lost. To solve this problem we can confine the histogram to smaller regions and then perform the operation. This latter is adaptive histogram equalization. However, this last strategy may over-amplify the the contrast in near-constant regions of the image, since the histogram in such regions is highly concentrated. But, it is possible to solve it by applying a contrast limiting; if any histogram bin is above the specified contrast limit (by default 40 in OpenCV), those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. In other words, CLAHE limits the amplification by clipping the histogram at a predefined value before computing the CDF. This limits the slope of the CDF and therefore of the transformation function. By distributing uniformly the clipped pixels, we mean something like the following graph: After equalization, bilinear

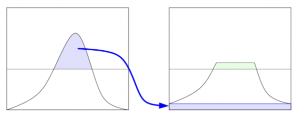

Figure 1: Redistributing clipped pixels

interpolation is applied. What we mean by interpolation is that the transformation applied on the pixel would be the interpolation of the transformation applied over its closest pixels. In other words, we find the desirable transformation for the center of the tiles of the image, and then we find the transformation for other points consequently. This moreover improves the computational cost of

the method.
You can find the implementation of this method in the python file: *Clahe*. There are implementations of this method in the OpenCv library. Please find an example code that compares these two (OpenCv and normal implementation) in the notebook *Clahe_example.ipynb*.

4.1.3 Face localization. Simultaneous and automatic face detection in images is of great use in Computer Vision. We use in our code dlib library face localization which is based on histogram of oriented gradients (HOG) and linear SVM. HOG descriptors provide excellent performance relative to other existing feature sets including wavelets. The technique counts occurrences of gradient orientation in localized portions of an image. Using HOG was first proposed in [2] [3] [1]. In this latter work the detector provides a binary output indicating whether a specific region of an image contains an instance of the desired object. Each image passed into the detector is first converted into a set of gradients which are spatially discretized. A subimage of a given size is extracted from this gradient histogram and converted into a feature vector. This feature vector is then used as input to a binary support vector machine (SVM), and if the output exceeds a threshold, the object is detected at that location.
It is worthy to mention By nature of how the Histogram of Oriented Gradients (HOG) descriptor works, it is not invariant to changes in rotation and viewing angle. But there is another face localization function in this library which is Max-Margin (MMOD) CNN face detector that is both highly accurate and very robust, capable of detecting faces from varying viewing angles, lighting conditions, and occlusion. It needs however a pretrained model as input.

4.1.4 Landmark points and encoding. The implementation of the *shape_predictor* function of dlib takes as input the path to a the facial landmark predictor and as the result returns the landmarks when applied on an image. The constructed predictor takes as input the image along with the detected face inside this image.
The algorithm behind this function is based on the research work [4]: the model relies on a cascade of regressors learned via gradient boosting and improves the estimation of the landmarks that is initialized with another algorithm. The critical point of the cascade is that the regressor makes its predictions based on features, such as pixel intensity values, computed from the image itself and indexed relative to the current landmark estimation. The core of each regression function is tree based regressors. For each regression tree, the underlying function is approximated with a piece-wise constant function where a constant vector is fit to each leaf node; this regressor is trained by fitting this piecewise linear function on the train sets of (image, landmarks).
One of the methods used for initializing the landmarks in the latter landmark detection technique is called "viola jones" [6]: The algorithm has four stages:

(1) *Haar Feature Selection*: selecting properties common to human faces such as eye region that is darker than the upper-cheeks or the nose bridge region that is brighter than the eyes. A simple rectangular Haar-like feature can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image. This modified feature set is called 2-rectangle feature. The values indicate certain characteristics of a particular area of the image, such as edges or changes in texture. For example, a 2-rectangle feature can indicate where the border lies between a dark region and a light region.

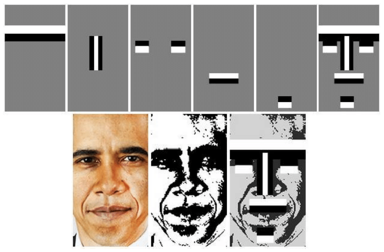
Figure 2: Haar-like features

(2) *Creating an Integral Image*: summed-area table or integral image is a data structure for quickly generating the sum of values in a rectangular subset of a grid. This step would greatly optimize calculations on the Haar-like features. The idea is the following: from the original intensity matrix we create another matrix from which we can evaluate the sum of intensities over any rectangular area by exactly four array references (corners) regardless of the area size. If i is the original matrix, this matrix I is defined as:

$$I(x,y) = \sum_{\prime x \leq x, \prime y \leq y} i(x,y)$$

(3) *Adaboost Training*: Adaboost is an ensemble learning technique that does the following: 1. Uses multiple (weak) classifiers, each based on different features 2. Combines these (weak) classifiers into a single powerful classifier. These weak classifiers satisfy a certain criteria on the weighed error of the train instances. The variant of Adaboost that is used in face detection also selects the best features.
(4) *Cascading Classifiers*: In this step, strong classifiers are arranged in a cascade in order of complexity, where each successive classifier is trained only on those selected samples that pass through the preceding classifiers(this may be a false positive that would be rejected by the subsequent classifiers). The first classifier in the cascade – called the attentional operator – uses only two features to achieve a false negative rate of approximately 0% and a false positive rate of 40% [5].

5 EXPERIMENTAL SETUP AND EVALUATION

We tried different experimental setups to test our implementation and we have the following results:

(1) The environment lighting is the an important factor for detecting the hand histogram. It works best when the background and the (hand and face) are distinct in color and brightness.
(2) In order to have more accurate results in detecting hand gestures we use a pre-trained model that *****
(3) We also used a database of sign language gestures and their corresponding letter, number or phrase:***********

(4) We tested the emotion detection implementation with 5 different classification methods from sklearn: SVM, RandomForest, KNN, Logistic Regression and MLP. The best results are for the classification with Logistic Regression:

Emotion	Precision	Recall	f1-score
Anger	1	0.67	0.8
Contempt	0.5	0.29	0.36
Disgust	0.9	0.75	0.82
Fear	89	0.80	84
Happy	1	1	1
Neutral	0.83	0.99	0.91
Sadness	1	0.09	0.17
Surprise	1	0.97	0.98

6 GITHUBS

berksudan/Real-time-Emotion-Detection
EvilPort2/Sign-Language

REFERENCES

[1] Navneet Dalal. 2006. *Finding people in images and videos.* Ph.D. Dissertation. Institut National Polytechnique de Grenoble-INPG.

[2] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Vol. 1. Ieee, 886–893.

[3] Navneet Dalal, Bill Triggs, and Cordelia Schmid. 2006. Human detection using oriented histograms of flow and appearance. In *European conference on computer vision*. Springer, 428–441.

[4] Vahid Kazemi and Josephine Sullivan. 2014. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1867–1874.

[5] Richard Szeliski. 2010. *Computer vision: algorithms and applications.* Springer Science & Business Media.

[6] Paul Viola and Michael J Jones. 2004. Robust real-time face detection. *International journal of computer vision* 57, 2 (2004), 137–154.

7 RESULTS

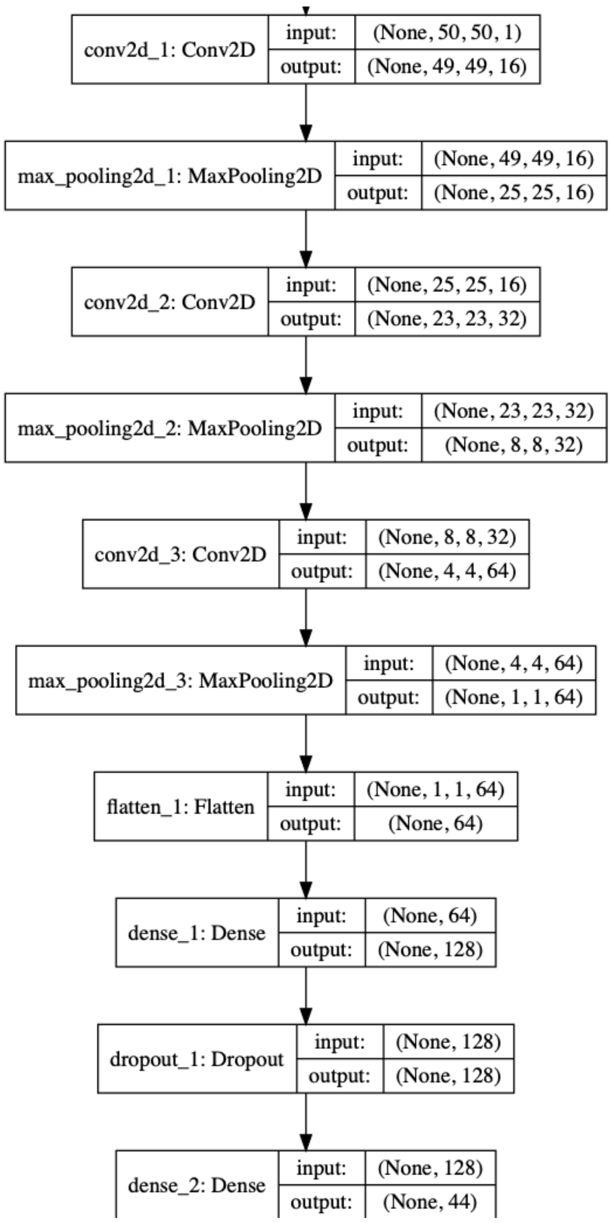

Figure 3: The convolutional neural network used in ASL translator

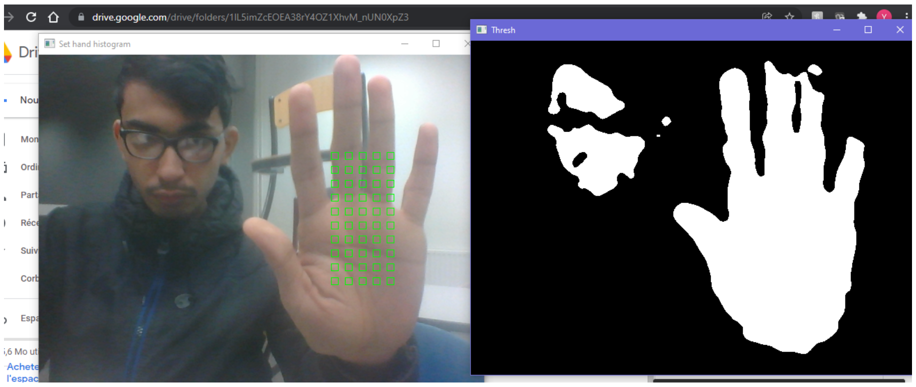

Figure 4: Building histogram

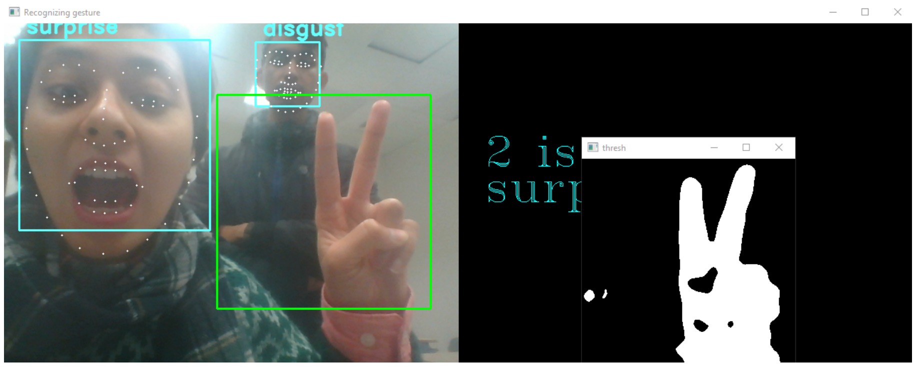

Figure 5: An example of the combined translators: ASL and face emotion

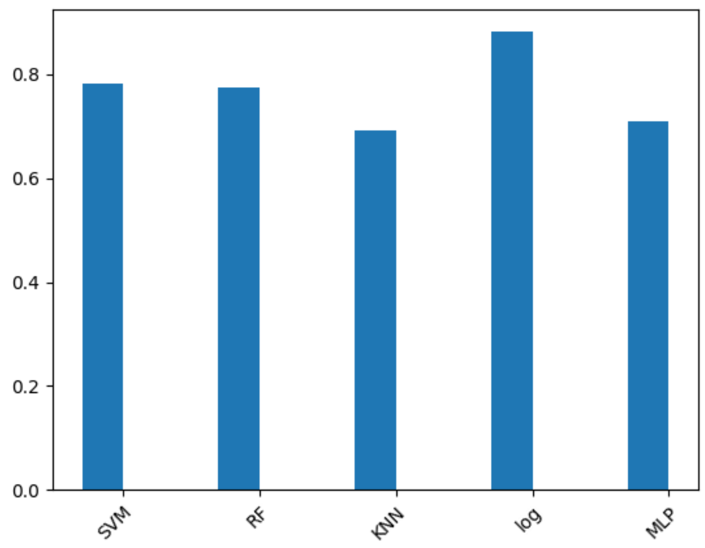

Figure 6: Comparing different methods for emotion classification