Emotion recognition

Edoardo De Matteis

February 1, 2021

Contents

1 Task and motivations

The goal of this project is developing a facial emotion recognition system, facial emotion recognition [2] is the process of analyzing someone's human state from facial expressions. This is done automatically by the brain but software has been and is developed such that machine could recognize human emotions too. The technology is improving all the time and some are confident it will be able to read emotions as our brains do . Emotion AI has a wide range of uses from scanning for signs of terrorism [11] to emotion judgement for commercial purposes as *Disney* did or at least wanted to do for the release of *Toy Story 5* [6]. All that glitters is not gold and Emotion AI could be a threat to freedom and equality [3], in China *Hanwang Technology* deployed a system that tracks students' behaviour during lessons analyzing if they're keeping attention or their action (e.g. "Answering question") [4]. This software seems to be unefficient and based on pseudoscientific assumptions, these allegations didn't stop *Amazon*, *Microsoft* and *Google* from offering emotion recognition to their customers even though the first two companies claim that their product can't determine a person's internal emotional state from only facial expressions [8].

The ethical dilemma of these technologies can be reduced to the old saying *quis custodiet ipsos custodes?*.

2 Strategy

To recognize emotion first is needed to find a face in an image, at first I considered using LBP histograms but since I was interested only in facial expression's features seemed reasonable to use landmark points. Assuming that similiar expressions have a similiar position on different faces through an SVM classifier it could be possible to predict someone's "emotion" [1]. For our purposes not all points where considered but only ones representing eyes and the mouth, this means only the points from 37 to 68 in figure 2.1. In general to get good results is reccomended to use neural networks [1], anyway I was interested SVM classifiers and how one would perform.

The adopted dataset is the *First Affect-in-the-Wild* (affwild) [7], it consists of 298 videos of which 252 for training and 46 for testing. Only videos in the train set have responses associated to them so I ignored the test set, for each video in which is identified a bounding box are associated the responses. The emotion is definex as a point on a two dimensional cartesian plane where the x axis is the *valence*, it expresses if an emotion is positive or negative, and the y axis is the *arousal*, it discriminates animated emotions from languid ones as can be seen in figure 2.2 where are defined only 4 different emotions but is possible to define more. Considering I wanted to avoid classifying between too many classes - with the risk of performances reduction - and considering also that SVMs in their purest form are binary classifiers, it came natural to me to just consider only response between valence or arousal and I chose valence.

[1] As said earlier someone's emotions are more deep than just their facial expression.

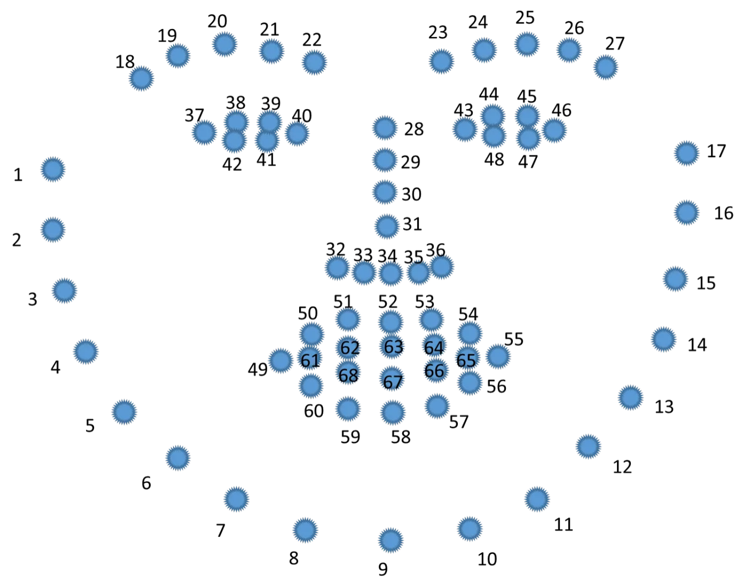


Figure 2.1: Face landmark points from [10].

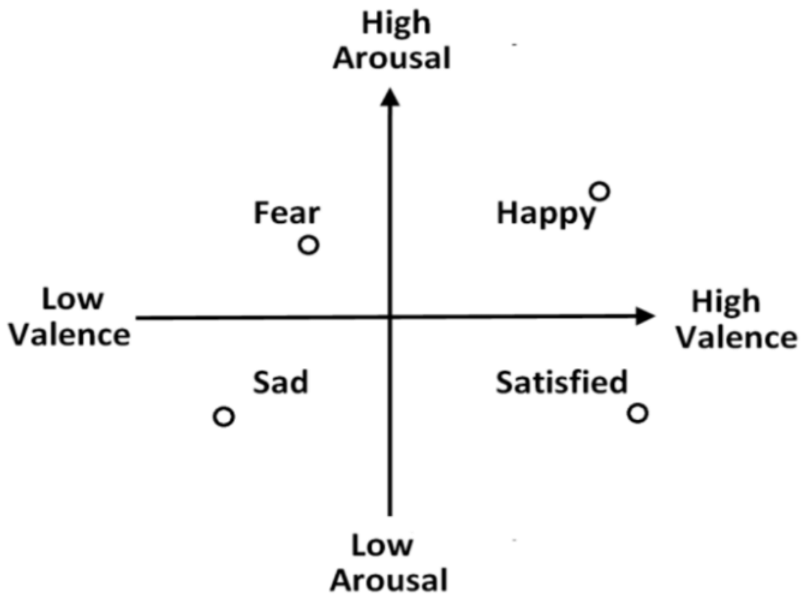


Figure 2.2: Emotion classification model from [12].

3 Design

I decided to write this project in Python with the external library OpenCV, in figure 3.2 can be seen the diagram of the packages with relative classes. Since I do not use the bounding boxes that are downloaded with the dataset the first thing I need to do is getting a correspondence landmark (using dlib) to valence for each frame of each video, the execution took almost 21 hours so I made possible to save the results as text files. Each valence ranges from -1 to 1, I explicitly ignored valences v such that $-0.5 \leq v \leq 0.5$ to minimize noise and rule out uncertain valences, in the end were extracted 136909 landmark-valence tuple. Is possible for a single frame to contain more than a face as seen in figure 3.1 [2]. Since the affwild datasets collects videos with foreground faces is reasonable to assume that the face we're interested in is the widest one, hence I pick the one with the largest euclidean distance between the first and last point of the feature points (respectively the lateral commissure of the viewer's left eye and the lower lip).

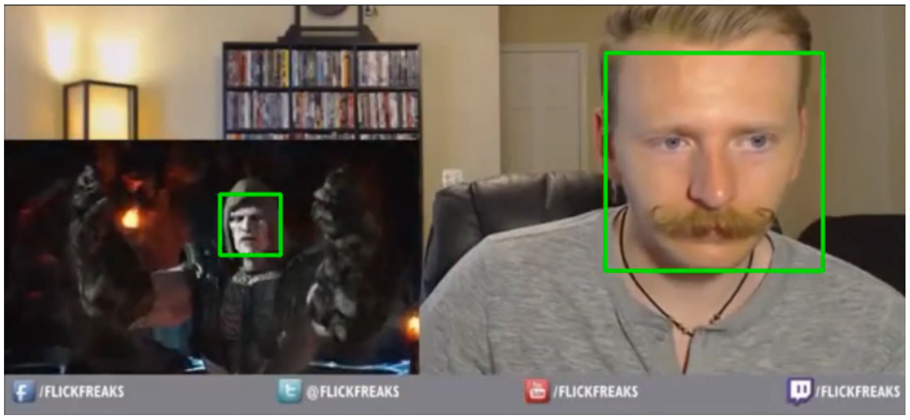


Figure 3.1: Two faces detected in the video 309.

Valences are saved and read as an array of floats [3] while landmarks are a two dimensional array [4]. For each face dlib stores landmark points as an array of $[x, y]$ coordinates, to make things more readable - and possibily to make the code faster in corner cases due to caching - I decided to flatten it as a one dimensional vector. I don't think this different encoding should change SVM's predictions.

Regarding SVM classification I first used the implementation in OpenCV with bad performances while the Sklearn one is way better. For this reason the demo uses the latter. During tests the dataset has been splitted as 80-20 per cent for train and test respectively.

[2]I find bounding boxes less invasive, in this case landmark detects two faces as well.
[3]datasets/misc/valences.txt
[4]datasets/misc/landmarks.txt

In the demo instead of directly detecting landmark points I first detect bboxes. This may be odd but I noticed how bounding boxes are less accurate than landmark points, detecting less landmarks makes the demo less precise but is smoother for a presentation.

Both bounding boxes and landmark points were detected through pretained models [9] [5].

The general structure of classes and packages that were defined can be seen in figure 3.2, it is all original code except the function `src.test.Test.plot_roc` used to plot ROC curves. It has been taken from here as commented in the source code.

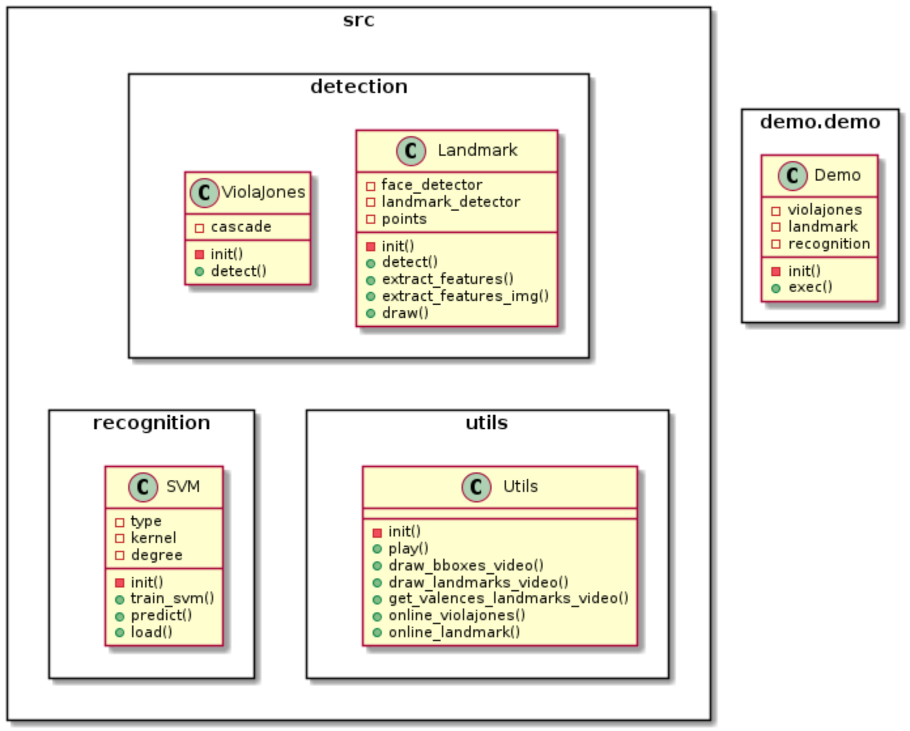


Figure 3.2: Diagram of the packages.

Other code that is not original are clearly external functions. I used the following external librariees:

- **os** To explore the filesystem.
- **re** For regular expressions.
- **fire** To implement command line arguments.
- **numpy** The use of arrays is essential thanks to this faster C implementation.

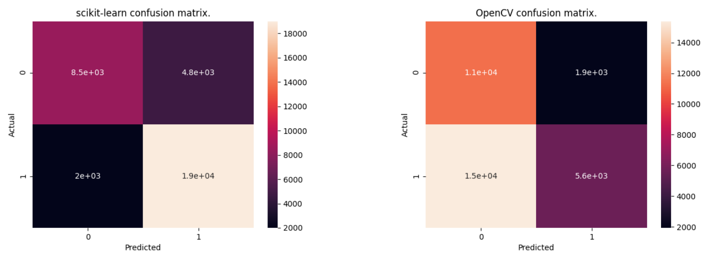
- **cv2** OpenCV as stated above, it has been used for detection with Haar features and an SVM implementation.
- **dlib** For landmark points since for OpenCV there is still no Python implementation.
- **sklearn** In this library is implemented a more performing SVM classifier than OpenCV's one.
- **joblib** used to save sklearn's SVM model.

4 Performance evaluation

The SVM classifier implemented in the scikit-learn library uses a RBF kernel (equation 1) while in OpenCV is possible to choose which kernel and eventually the degree of the model.

$$K(x, x') = \exp\left(\frac{||x - x'||^2}{2\sigma^2}\right) \tag{1}$$

The scikit-learn classifier offers the best performances, in figure 4.1 is possible to see the confusion matrices for both this classifier and the RBF OpenCV one.



(a) Confusion matrix for *scikit-learn.* (b) Confusion matrix for *scikit-learn.*

Figure 4.1: Confusion matrices.

The scores obtained are collected on table 1.

Table 1: Scores for the classifiers.

	scikit	OpenCV
Accuracy	0.805	0.491
Precision	0.814	0.421
Recall	0.639	0.860
F1	0.358	0.283

In images 4.2 are plotted the ROC curves for the scikit-learn SVM classifier and OpenCV ones with linear, 3-degree polynomial and RBF kernels.

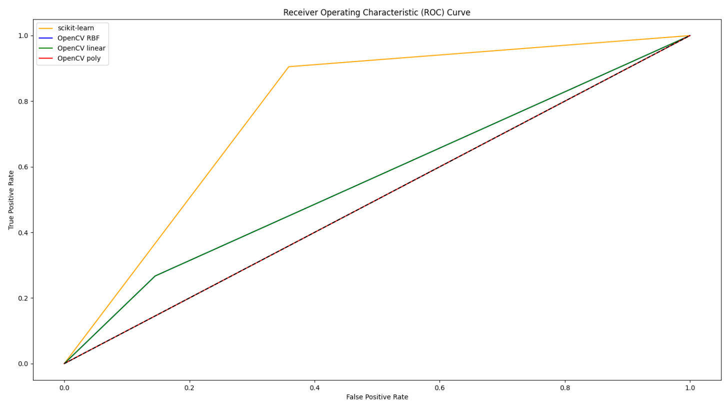


Figure 4.2: ROC.

The polynomial one has terrible performances, is like throwing chances [5], while the linear and rbf ones overlap each other with quite better performances. Still the best performing one is the scikit-learn with a rbf kernel, on table 2 are displayed the AUCs.

Table 2: Scores for the classifiers.

	scikit	OpenCV
AUC	0.774	0.560

On figure 4.3 are plotted the DET curves for only the scikit classifier and the OpenCV one, both with rbf kernel.

On figure 4.4 are plotted the false positive (i.e. false genuine) error rate and the false negative (i.e. false rejection) one, the intersection point is the equal error rate (EER), the ideal threshold seems to be 1.25. Keep in mind that these two error rates are plotted against only three thresholds so this value is not necessary true. The EET is identified only for the scikit-learn SVM since it is the best model so far, for this same reason the online demo uses it.

5 Conclusions

Starting from the fact that I did not expect too high performances a 76% AUC is not too bad, unfortunately I did not found notebooks or any other data to

[5]has an AUC of 0.5

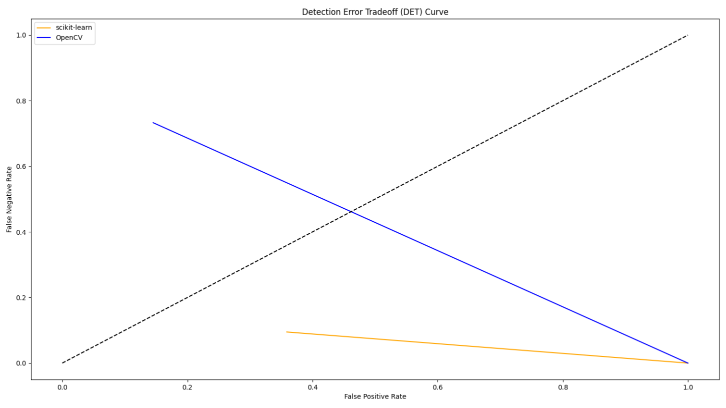


Figure 4.3: DET.

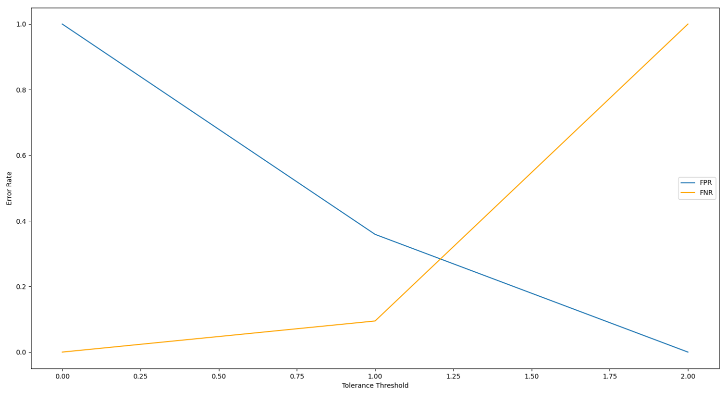


Figure 4.4: EET.

compare my results with. I was surprised of how a 3-degree polynomial model overfits and the linear is better than it.

5.1 Future works

It could be possible to implement a classifier that predicts both valence and arousal, it could be done with a SVM multiclass classifier that uses softmax regression or with two binary SVMs.

References

[1] Bleed AI. Emotion/facial expression recognition with opencv. https://bleedai.com/facial-expression-recognition-emotion-recognition-with-opencv/, 2020.

[2] Algorithmia. Introduction to facial emotion recognition. https://algorithmia.com/blog/introduction-to-emotion-recognition, February 2018.

[3] Article19. Emotion recognition technology: A threat to free speech, equality and privacy. https://www.article19.org/resources/emotion-recognition-technology/, January 2021.

[4] The Disconnect. Camera above the classroom. https://thedisconnect.co/three/camera-above-the-classroom/, 2019.

[5] dlib. shape_predictor_68_face_landmarks. http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2.

[6] Gizmodo. Disney is building facial recognition to figure out when you'll laugh during *Toy Story 5*. https://gizmodo.com/disney-is-building-facial-recognition-to-figure-out-whe-1797267294, July 2017.

[7] ibug: Intelligent Behaviour Understanding Group. First affect-in-the-wild challenge. https://ibug.doc.ic.ac.uk/resources/first-affect-wild-challenge/, 2017.

[8] Rest of World. China is home to a growing market for dubious "emotion recognition" technology. https://restofworld.org/2021/chinas-emotion-recognition-tech/, January 2021.

[9] Lalitha Rajesh. Haarcascades. https://www.kaggle.com/lalitharajesh/haarcascades.

[10] Sefiks. Facial landmarks for face recognition with dlib. https://sefiks.com/2020/11/20/facial-landmarks-for-face-recognition-with-dlib/, November 2020.

[11] Ars Technica. The premature quest for ai-powered facial recognition to simplify screening. https://arstechnica.com/information-technology/2017/06/security-obsessed-wait-but-can-ai-learn-to-spot-the-face-of-a-liar/, February 2017.

[12] Şule Şahaboğlu. A research on the machine learning algorithms' applications on brain data analysis. 06 2019.