EmotionRecognition

Edoardo De Matteis

February 6, 2021

Goal

The goal is to develop a **facial emotion recognition** system able to determine if a face shows positive or negative emotions, note that emotions are something deeper than just facial expressions.

Use cases of facial recognition

- Mood analysis.
- Prevention of terrorist attacks.
- Refinement of marketing strategies.
- Behavioral analysis.

Strategy

The strategy follows this pipeline:

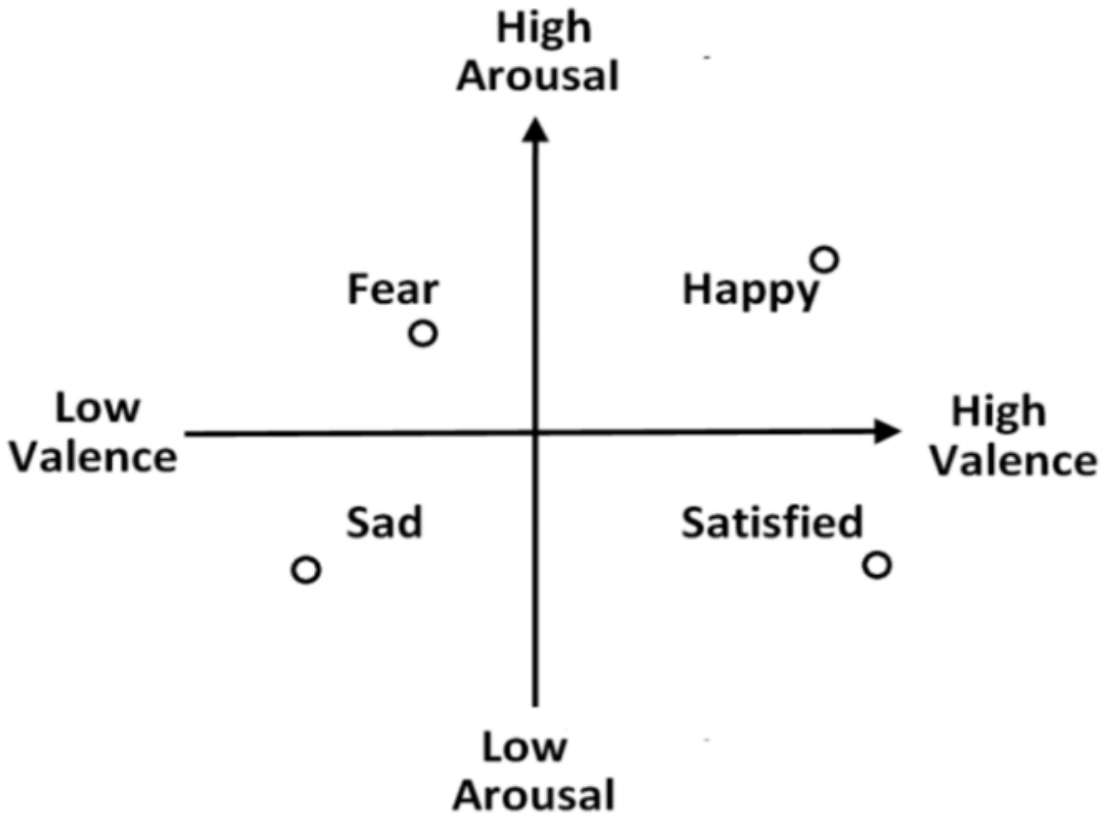
1. Face detection with Haar cascade classifiers.
2. Feature extraction with landmark features.
3. Classification with support vector machines.

Note that to get good results is reccomended to use neural networks.

Dataset

The adopted dataset is *First Affect-in-the-Wild* (affwild), it is composed of videos and to each frame is associated an emotion. These emotions are represented as points on a 2D cartesian plane.

Emotion representation



Design choices

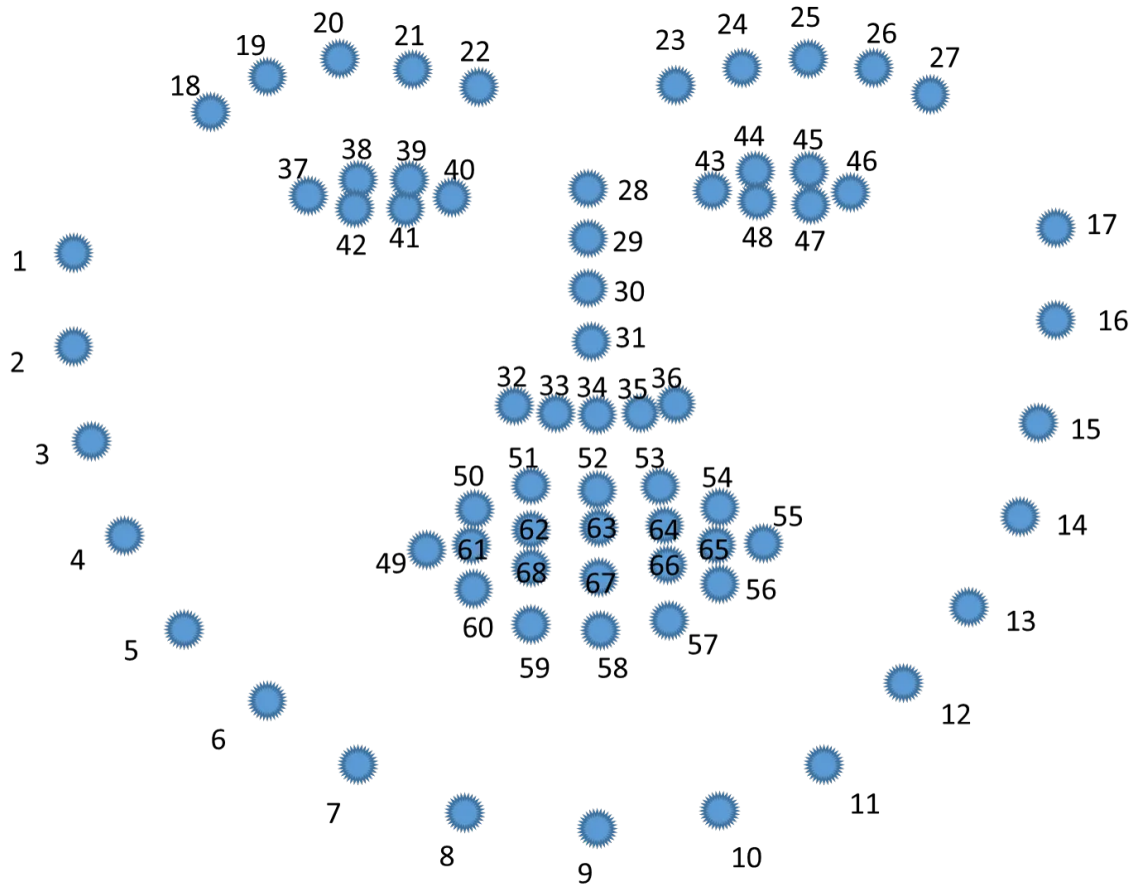
The system is written in the *Python* language and uses *OpenCV*.
The main design choices regard:

- How to encode the sample set.
- How to deal with responses.
- Choose the SVM.
- Online demo.

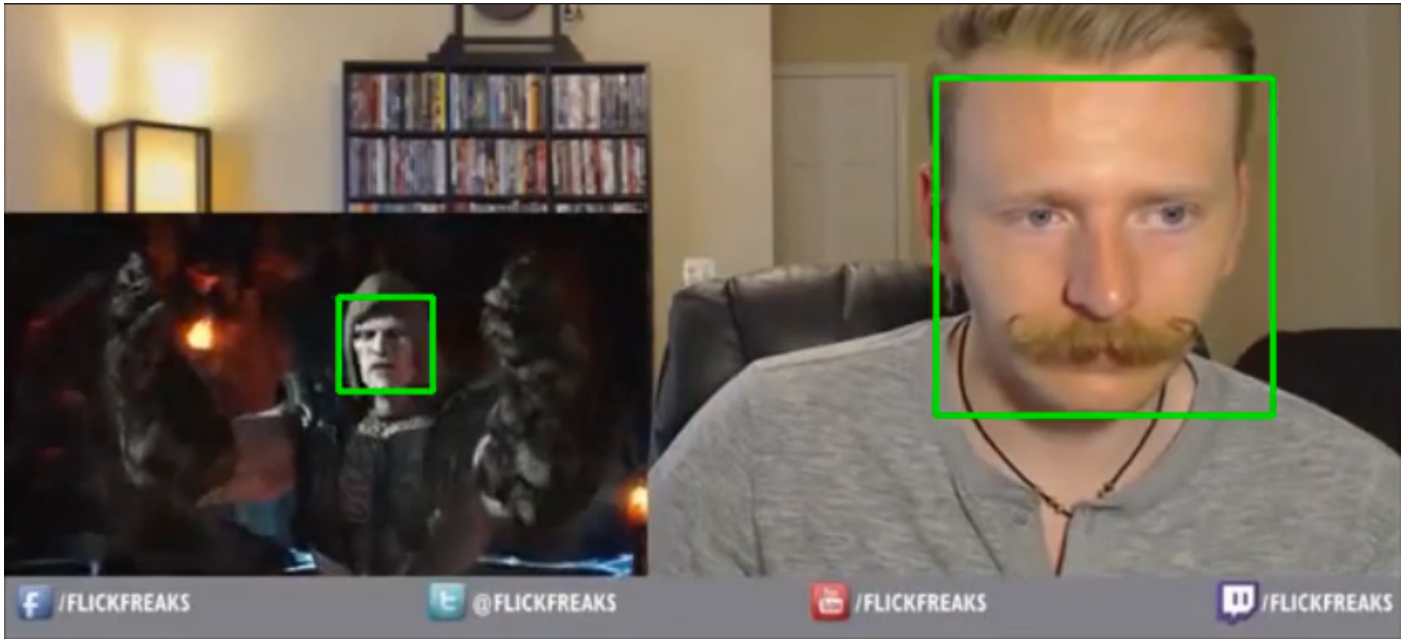
Samples

- To represent expressions I considered only landmark points corresponding to eyes and mouth. Instead of a vector of points we have a 1D vector of coordinate points.
- I assume that the face we're interested in is the foreground one. For this reason I pick the one with the maximum euclidean distance between the first and last point.

Feature extraction



Feature extraction



Responses

Since classifying between too many classes could lower performances I decided to drop arousal and just consider valence, thus not classifying emotions but only if they are positive or not. For each frame were considered only valences such that $-0.5 < v < 0.5$, in the end were extracted 136909 examples.

SVM classifier

The *OpenCV* SVM classifier has terrible performances, for this reason is used the better one *scikit-learn*, it uses a RBF kernel. Still the *OpenCV* SVM with RBF is worse than the *scikit-learn* one.

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

Online demo

I noticed how, given the same frame, landmark points are more easily found than bounding boxes. To make the demo smoother I first detect bounding boxes and then landmark features.

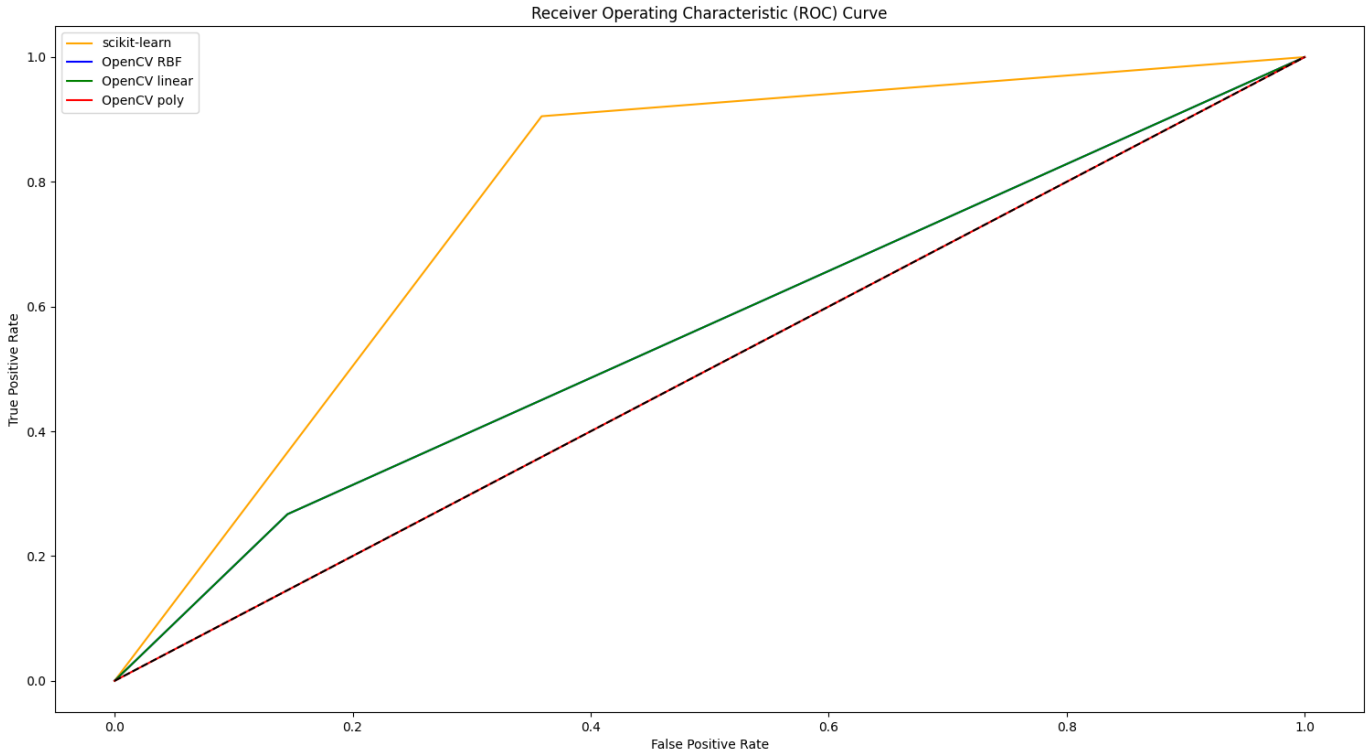
Performance evaluation

Table: Scores for the classifiers with a RBF kernel.

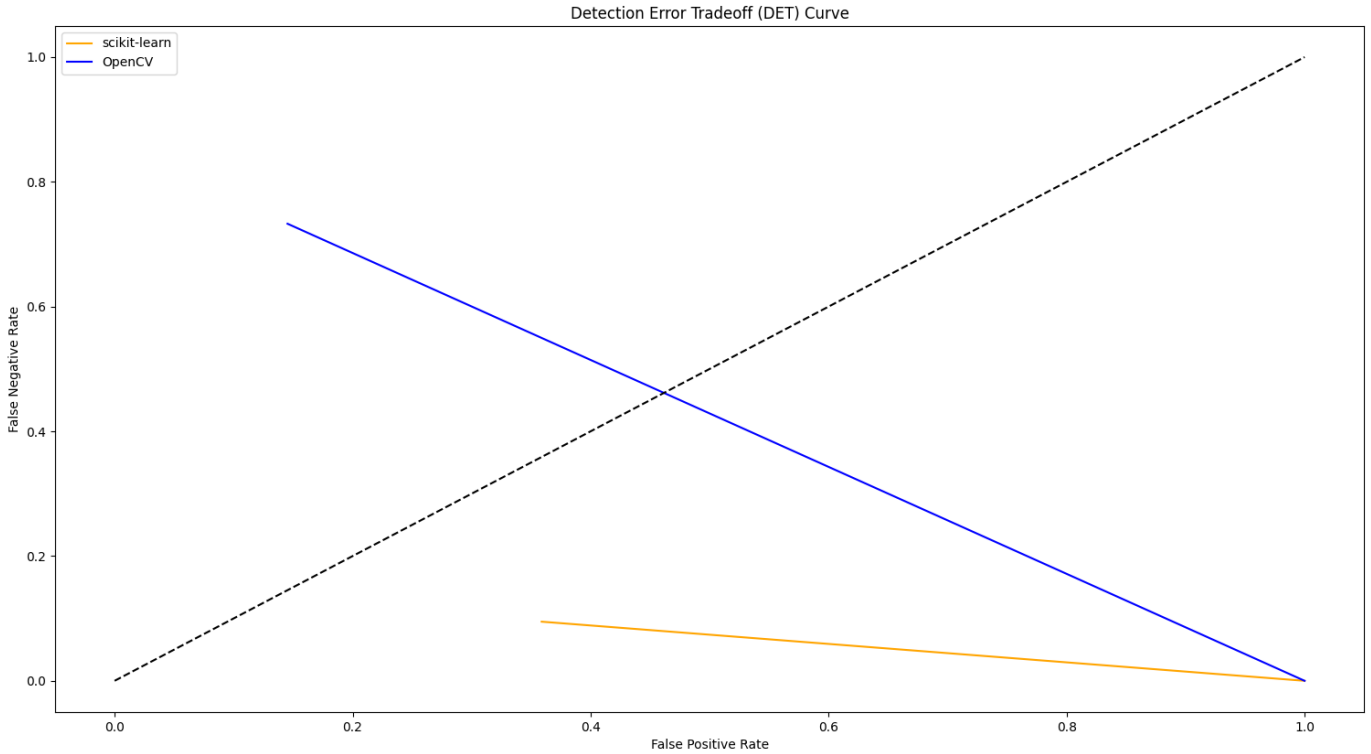
	scikit-learn	OpenCV
Accuracy	0.805	0.491
Precision	0.814	0.421
Recall	0.639	0.860
F1	0.358	0.283

- Receiver Operator Characteristic curve.
- Detection Error Tradeoff curve.
- Equal Error Tradeoff.

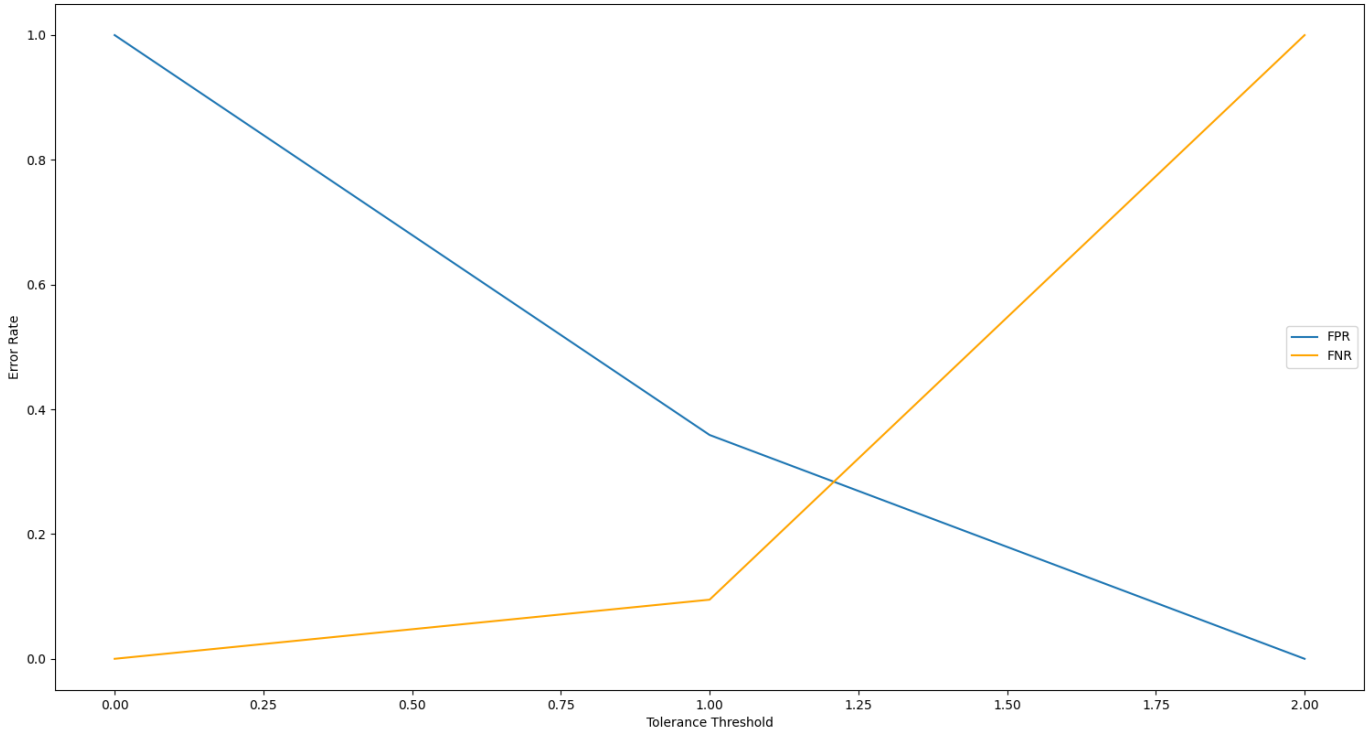
ROC curve


Receiver Operating Characteristic (ROC) Curve
scikit-learn
OpenCV RBF
OpenCV linear
OpenCV poly
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Detectior Error Tradeoff curve



Equal Error Tradeoff



Conclusions

- The scikit-learn SVM classifier has quite good results, with a 76% AUC.
- The polynomial kernel performs worse than the linear, it seems due to overfitting.
- In the online demo PIE variations are crucial for the outcome.

Future works

- Predict both valence and arousal.
- Use a neural network instead of a SVM.

Thank you.