Artificial Intelligence Project

Thomas Vandermeersch 17030 and Martin Sing 16116

31 December 2021

Contents

1 Introduction

This artificial intelligence project is to be realised by 2 or 3 students in their Master 2 year at the ECAM, for the IT specialisation. In this project, we have to use the live feed of the camera and superpose an emoji, depending on the mood detected on the subjects, on their faces. Here are some of the asked features :

- Detect as many faces as possible in real time.
- Detect mood to pick suitable emoji.
- Detect masked persons and display masked emoji.
- Detect head orientation to tilt emoji.

We have decided to do this project using Python programming language and based our detection on an OpenCV library. This report will go over how we detected the faces and their corresponding moods, how has the AI been trained, how have the emoji been placed. We will also go over some points that we did not managed to put in place, but we will explain how we planned to do it and why it did not work.
The code for our project is available on GitHub, at this address :

`https://github.com/MartinGongSing/AI_face_emoji`

Here are the libraries used in the project :

- OpenCV
  - OpenCV is an open-source library for computer vision. It provides the facility for the machine to recognize the faces or objects.
- NumPy
  - NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- pandas
  - pandas is a software library written for the Python programming language for data manipulation and analysis.
- Keras
  - Keras is an open-source software library that provides a Python interface for artificial neural networks.

2 Multiple face detection (3 points) - in real time (1 point)

We decided to base our project on the OpenCV library[1]. The following lines of code show us how the face detection works. This allows us to store the coordinates of detected faces in a list (*faces_detect*). As well as their height and width.

```
face_haar_cascade = cv2.CascadeClassifier( cv2.data.haarcascades
                                          + 'haarcascade_frontalface_default.xml')
while True:
    ...
    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.32, 5)
```

The image is then cropped to have just a image of the face. We then pass the cropped image through the model to predict the mood (step detailed in chapter 3).

The picture on figure 1 has been captured through the webcam of a laptop :

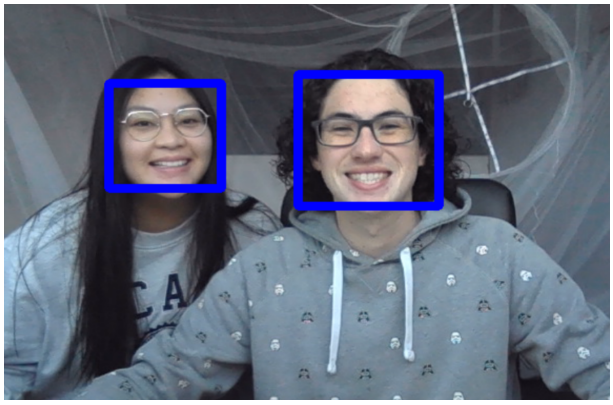

Figure 1: Multiple Face Detection

How does cv2.CascadeClassifier work ?

In 2001, Paul Viola and Michael Jones proposed in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features", an effective method of object-detection. This method is called "Haar feature-based cascade classifiers". *"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."*[2]

To start the process, to train the algorithm, we need positive and negative images (with and without faces) and extract the features out of them. Haar features on Figure 2 are used for this. For each feature, we subtract the sum of pixel of the white rectangle from the ones under the black rectangle. We can compare those features to *convolutional kernel*.

To calculate the features, all possible location for all sizes of EACH kernel are used, which is a lot (24x24 window will give over 160 000 features). What they did to go around this problem, is introducing the *integral image*. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels.

Most of the calculated features are irrelevant, which is why we will have to select the best ones. On Figure 3 we can see on the top row, 2 good features. The first one focuses on the eyes

[1] https://pypi.org/project/opencv-python/
[2] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

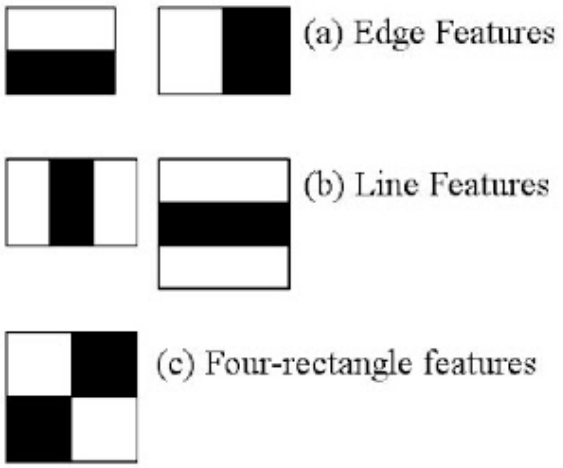

Figure 2: Haar Features

being darker than the nose and cheeks, and the second relies on the eyes being darker than the nose. On the otherside, this "window" can apply on cheeks or anywhere else... which would be irrelevant here. What we want to do is select the most appropriate, the best, features possible. This is achieved by **Adaboost** [3]

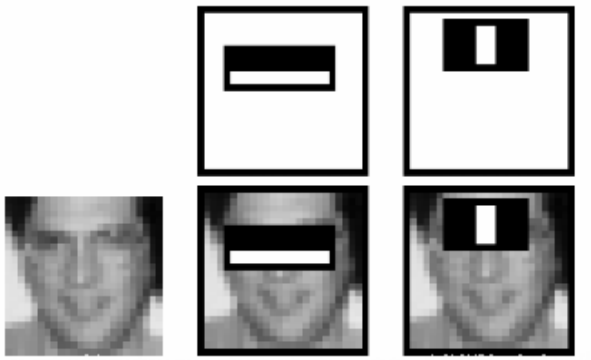

Figure 3: Haar features on image

In order to do this, we apply every feature on all of the training images. The best threshold is found for each feature, which will classify the faces to positive and negative. There will obviously be errors or misclassifications. The features that most accurately classify the face and non-face images are the ones with minimum error rate. In reality, things are not that simple... In the beginning each image is given an equal weight. After each classification, weights of misclassified images are increased and the iteration continues. New error rates and new weights are calculated until the requirements are reached.

[3] *"AdaBoost, short for Adaptive Boosting, is a statistical classification meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Gödel Prize for their work. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner."* **Wikipedia**

This small mathematical by-pass allows us to have an accuracy of over 95% for only 6000 features (instead of the initial 160 000).

That was how the classifier works. But what of the "Cascade" ? Well the expression "en cascade" in French could be called the "domino effect" or "chain reaction". Which is precisely what happens here. We have multiple classifier working one after the other in order to optimise the work done. If a window, an image doe not pass the first classifier, there is no point in passing it through the others. If the images passes through all the features, then it can be considered as a face.

To give a few numbers, the authors' detector had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in the first five stages.

What can we remember from this ?

In simpler words, *cv2.CascadeClassifier* is a pretrained model to detect faces. To do so, an image goes through multiple "filters" with different features analysing it. If the image goes through all of the "filters", we have detected a face.

3 Mood detection (1 point)

3.1 Mood detection

Once the face is discovered (previous chapter), we can start to do some manipulations on it. First thing we will do, is passing it through a pretrained model. The following line of code allows us to load it in our file (model trained in the *Emotion_Detection.ipynb* file and explained later on).

```
model = load_model("best_model.h5")
```

Once the model is loaded, we will compare the discovered face with the information gotten from the dataset, in order to label the face with the proper mood. The prediction of the mood is stored in a list, with a certain value for each mood. We will then select the mood with the highest score to continue the work. The variable *predicted_emotion* now has one of the 7 (8 with "masked") emotion listed.

```
predictions = model.predict(img_pixels)

# find max indexed array -> most suitable
max_index = np.argmax(predictions[0])

#list of available emotions
emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral')

#saves the emotion's name
predicted_emotion = emotions[max_index]
```

Now that we have our mood saved, we can start the task of replacing the face of someone by an emoji. This will be explained in section 4.

The picture on figure 4 has been captured through the webcam of a laptop :

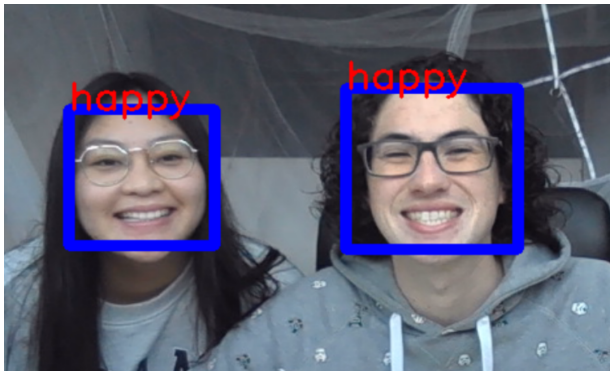

Figure 4: Multiple Mood Detection

3.2 Train the AI ourselves (using public datasets) (2 points)

The AI is trained based on a neural network model. This model was found on the internet [4] but we tried to change some parameters in order to improve it.[5] Here are the major steps:

1. Preparing the notebook
   - First of all, we need to extract the dataset and prepare the images to be readable.
   - Then we will import all the different packages needed for the project.
2. Building our Model to **train the data**
   - This step is here to define the shape of the model (224,224,3)
   - It also sets the activation layer and the number of layers
   - And of course, build the model
3. **Preparing our data** using data generator
   - What we are doing here, is preparing the data. We are defining the batch size as well as the target size
   - But we also prepare the model to receive our camera live-feed (zoom_range, horizontal_flip,...)
4. **Visualizing the data** that is fed to train data generator
   - This part is not the most essential one... It is only used for us to make sure the model is working well.
5. Having **early stopping** and model check point
   - The early stopping allows us to have no overfitting and to win some time.
   - We also set some other settings for the generator.
6. Save the model in a file

[4] `https://www.youtube.com/watch?v=G1Uhs6NVi-M`

[5] If the reader wants more information, the code used is explained and commented in the related Jupyter Notebook.

4 Emoji placement (1 point)

When the mood is detected, it will be saved in the variable "*predicted_emotion*". We shall use this variable to choose the corresponding emoji to display. A simple "*if-elif-else*" is used to do so. Once the correct emoji is chosen, it goes through a series of operations, scaling it for example.

Once this is done, the cropped emoji is superposed on the face, using the cv2 function "*addWeighted()*".

```
dim = (w,h)
resized = cv2.resize(emoji, dim)

#test_img : the one captured by camera

#if : if no faces are detected or if dim did not work -> do nothing
#else : if faces are detected -> addWeighted : superpose emoji + camera

if test_img[x:x+w, y:y+h,:].shape != resized[0:w,0:h,:].shape:
    added_img = test_img
    # print("Face not detected")
else :
    # print("Face detected")
    added_img = cv2.addWeighted(test_img[x:x+w, y:y+h,:], 0, resized[0:w,0:h,:], 1, 0)

    #right axis + add of images
    test_img[y:y+h, x:x+w,:] = added_img     # Comment this to remove the emoji
```

The picture on figure 5 has been captured through the webcam of a laptop :

Figure 5: Multiple Emoji Display

5 Upgrades

Here following are the points we did not manage to achieve by the end of the project. For some of those, we have a non functional solution and an explanation on why it did not work.

5.1 Orientation detection (2 points)

Here, we are trying to use the position of the eyes to detect an orientation for the face. Having the corner detected, we divide the height by two to get to the center of it, and compare it with the other eye. The eyes being stored in a list, we then can see if one is higher than the other and by so, we know if the face is tilted.

```
eyes = eye_cascade.detectMultiScale(roi_gray)


for (ex,ey,ew,eh) in eyes:
    cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
    eye_c = ey + eh/2

# trying to change emoji based on eye hight
    if (previous_ey - 0.5) <= eye_c <= (previous_ey + 0.5) :
        emoji = straight

    else :
        emoji = tilted

previous_ey = eye_c
```

Why has it not been done ? We encountered a problem with the "*eye_cascade*", it detected to many eyes for a normal subject (with only 2 eyes), making our calculations too complicated. We have then decided to aboard this part of the task to focus on the other ones.

5.2 Masked people detection and suitable emoji (2 points)

For this, we used another model. If a mask is detected, the mood will be set to "masked" and we will directly see a masked emoji on the face. If not, we shall proceed as planned in the previous steps.

```
if emotion2 == 'masked' :
    emoji = mask
else :
...
```

We wanted to implement the "masked" feature as follow :

- Identify if the subject has a mask
- If masked, the emotion detection is by-passed and a mask emoji is added on the face
- If not, the emotion detection works as planned

Why has it not been done ? We did not find a dataset that has emotion and mask mixed. Nor did we find a pretrained-model corresponding to our criteria. We also thought about adding some masked people to the original dataset, but it might have compromised the emotion detection.

5.3 The App runs on a phone ! (1 point)

We first thought about implementing a real-time analysis of our camera feed, but the phone browser did not allow us to use it. We then decided to modify our code, to make it work on a webserver. We can now only upload one picture at the time, where the emotion is detected and the face replaced by an emoji, but then it is not "live". Here is the major change done in the code for the emotion detection:

```
cap = cv2.VideoCapture(0)

    ||
    ||
    \/

cap = cv2.imread("[name_of_file]")
```

We implemented a web server fot his matter, but this is out of the scope of this work.

6 Conclusion

What can we say about this project ? First thing to do, was to discover faces, then we have had to find out a way to detect their mood. Once this was done, we have had to retrieve this mood, to display it and to select a corresponding emoji. When the emoji is selected, OpenCV allows us to superpose the two images. With this project, we discovered how to mix several small artificial intelligence (face and mood detection) and how different libraries, such has OpenCV, work.

7 Sources

- https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
- https://www.youtube.com/watch?v=voRFbl-GKGY
- https://www.youtube.com/watch?v=G1Uhs6NVi-M