Face Recognition Analysis

Gerard Burgués

22 January 2021

TOPICS

Introduction

Face recognition system is a technology capable of matching a human face from an input image against a database of multiple faces from multiple people. It is usually used for ID verification. In this project will be used for sorting. There are many places we can see the usage of face recognition, for example, mobiles, laptops, security cameras, etc. In this project, we will go a bit away from security. I'll use face recognition to perform the backend of a simple application for sorting images.

The goal of this project would be:

1. With a given input image perform face recognition against the database.
2. When a face is detected and recognised we will create a folder for that specific person. If the folder already exists the image will be saved in that same folder.
3. Experiment with the results and analyse in what cases there is better performance of the algorithm.
4. Predict how many images will be taken in a near future using Markov Chain.

Technology

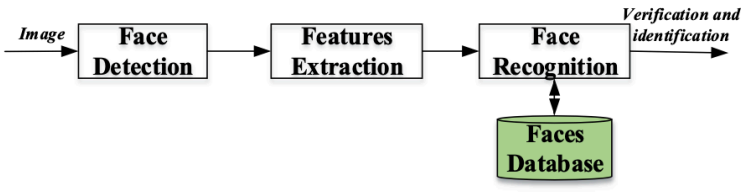
During this project, I have been using technology as Python including, NumPy, OpenCV, as Libraries.

Apart from the coding technology terms, the project would need a database where we would store and Id, email, image and date. Furthermore, pictures from the same camera would be needed to create a consistent database.

Face Recognition

Algorithm

Three basic steps are used to develop a proper face recognition system: Face detection, Face extraction and Face recognition



[1]

Face detection: There are many ways to implement a face detection program but in this project, it has been used a Haarcascade classifier focusing on the frontal-face and profile-face. Usually, this classifier should be trained using input images with and without faces and applying the Haar algorithm the training model would start learning to detect faces in the images. In our case, the classifier has been already trained and ready to use.[3]

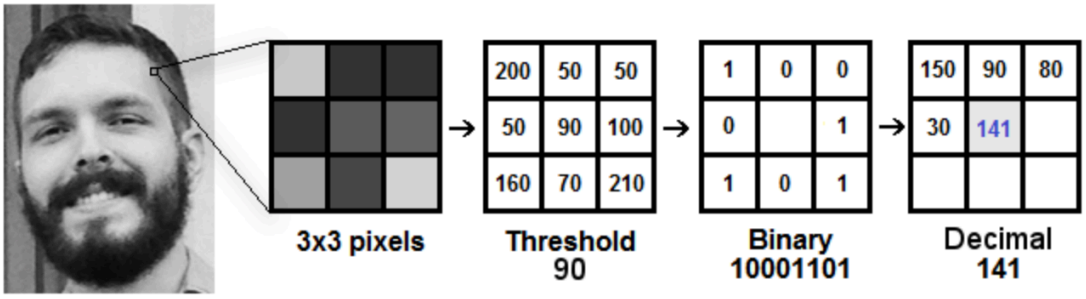
Face extraction: Is the process where will use LBPH Algorithm to finally perform the recognition.

LBPH (Local Binary Pattern Histograms)[2]

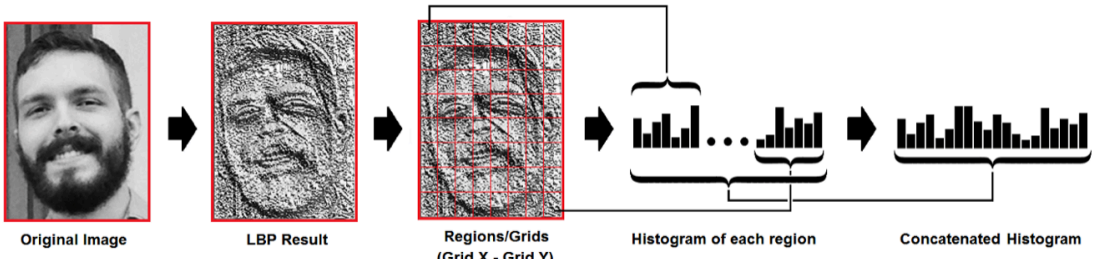
LBP is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number.

Steps:

- First, we need to train our model. To do so, a dataset of facial pictures of people will be needed where each picture or person will have its ID.
- LBP Operation:
    - Turn an image into grayscale
    - Get a 3x3 matrix containing the intensity of each pixel.
    - We select the central value of the matrix and with this value, we will define the new values from the 8 neighbours. If the value is bigger than the threshold, as a result, the new value is set at 1, if it is lower than the threshold the new value is set at 0.
    - Now the matrix will have only binary values. We need to concatenate them starting from the (0,0).
    - Convert the binary value into decimal value and set it in the central value of the matrix.
    - At the end of this process, we will have a new image which represents better characteristics than the original image.



- Extraction of Histograms:
    - We will break the image for regions (look at the image). For each region, we will calculate the histogram. Once we have all histograms we will concatenate them all together creating a massive histogram. Usually, the image is broken by 8x8 or 4x4 supposing a final histogram of 8x8x256= 16.384 positions and 4x4x256=4.096 positions.



- Saving a trained model:
    - After performing histogram for each picture in the database we will add each picture in a dictionary and save them in a .yml file called the model.
- Performing Recognition:
    - Given an input image we performed the steps again (Face detection - LBP Operation - Extracting histogram).
    - We compare those histograms using euclidean distance. As lower is the distance or confidence better results. Meaning that the person the input image will be recognised will be the image with less euclidean distance.

Coding Process

Having a proper Dataset is so important to have success in the project. The first step was taking pictures of my family and save them in my directory. To increase the number of pictures I coded a program intending to flip the image.

Secondly, I started to develop the face detection step. Using the Frontal and Profile Haarcascade module and saving each image in the database (just the face) MySqlWorkBench.

Once training images were classified I started implementing the LBP Algorithm of a single image, and at the time the implementation worked I started retrieving pictures from the Database and implementing the training model.

Additionally, I performed the Face Recognition step. Using Euclidean distance between one input image and the already trained model. After analysing the results once it worked I decided to implement the “END GAME”. The End Game consists of creating a folder for each new person that it recognise or save the image in an already existing folder.

Finally, I performed a test for all the input images and analysed the results.

Results

LBP Operation with an Image size of 800x950 and 8x8 for the grid:

> Out of 34 picture 11 of them succeeded of the recognition, 15 were wrongly recognised and 9 were not detected. This makes a percentage of success in 42.3%.

LBP Operation with an Image size of 800x950 and 4x4 for the grid:

> Out of 34 picture 7 of them succeeded of the recognition, 26 were wrongly recognised and 8 were not detected. This makes a percentage of success in 27%.

LBP Operation with an Image size of 512x768 and 8x8 and 4x4 for the grid:

Out of 34 picture 7 of them succeeded of the recognition, 18 were wrongly recognised and 9 were not detected. This makes a percentage of success in 28%.

Analysis

With this given results we see that something is wrong with the code but still, we can conclude that:

1. As bigger is the picture easier is for the Face Detection algorithm to detect a face in an image.
2. When the picture is bigger we can notice an improvement in the program.
3. Furthermore, we can observe that when there are more grids and the image is enough big we have the best performance.

But various factors affect the result of this project:

- Dataset of pictures
- Database
- Face Detection Algorithm

Dataset of pictures: Creating a new dataset requires time and quality. The training dataset has to be done with a single camera, the same one as the dataset test. The resolution or quality of the camera may affect the result.

Database: When retrieving the picture from the database the program uses a library called Pillow. This library modifies the values of the image and therefore the values of the histogram. In conclusion, retrieving images from the database affects the solution.

Face Detection Algorithm: The algorithm may detect sometimes more than just the face. When this happens we are not calculating the pixels and the

edges of just the face but also the background. When we have to compare the histograms the program does not work.

Markov Chain

Markov chain describes a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.
In our case we will have 4 states that will classify the pictures: Important, normal, not_important, redundant.
Given a dataset of pictures and dates we will try to show a graphic on how many picture of each classifier will appear in the following months.

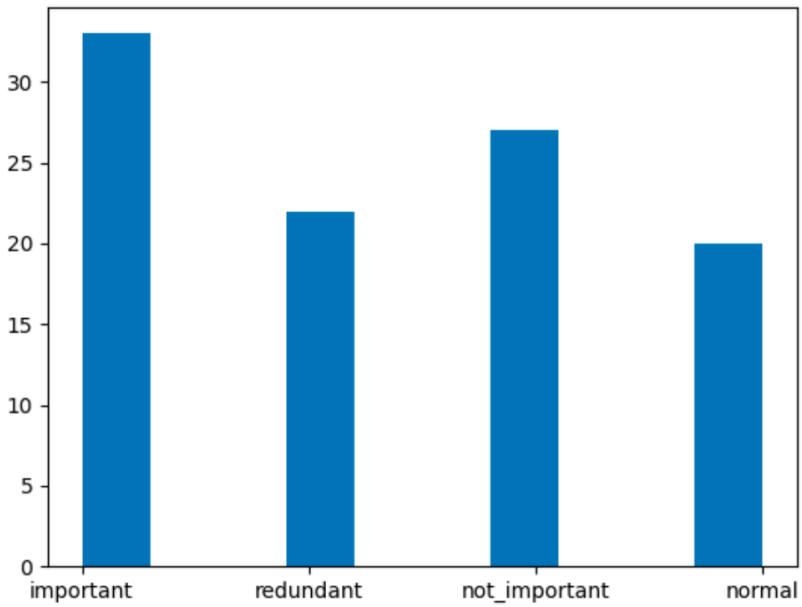
```
"label_id","email","day_pic","prediction"
1,"jordi_brother@gmail.com","2020-01-30","important"
1,"jordi_brother@gmail.com","2020-01-30","important"
1,"jordi_brother@gmail.com","2020-01-23","important"
1,"jordi_brother@gmail.com","2020-04-20","redundant"
1,"jordi_brother@gmail.com","2020-03-12","not_important"
1,"jordi_brother@gmail.com","2020-03-02","not_important"
1,"jordi_brother@gmail.com","2020-01-02","important"
1,"jordi_brother@gmail.com","2020-04-23","redundant"
```

Also we have to take into account that each transition(important, redundant, ...) will have a probability value.

```
transitions = [["II","IN","INT","IR"],["NI","NN","NNT","NR"],["NTI","NTN","NTNT","NTR"],["RI","RN","RNT","RR"]]
transitionMatrix = np.array([[0.7,0.1,0.1,0.1],[0.5,0.2,0.1,0.2],[0.1,0.2,0.6,0.1],[0.4,0.4,0.1,0.1]])#example . Com
```

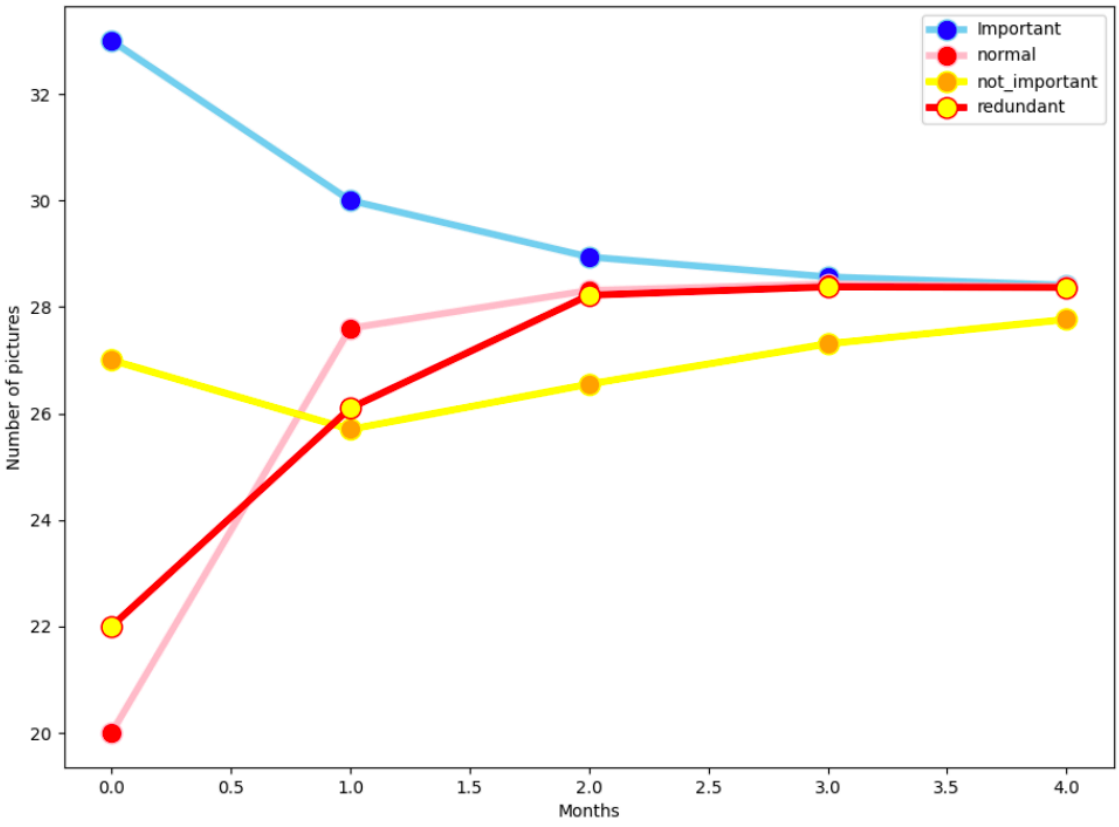
So the result will be based on how many times each state appears and the probability of each

Result



We can observe in this bar chart that out of 102 pictures that we have in out dataset, 33 are important, 22 redundant, 20 normal and 27 not important.

Based on this numbers and the specific probability (which can be modified) we can show our 4 month prediction:



Analysis

We can observe that based on the probability and the times that a state appear in our database we are able to tell how many picture will be taken the following months.
More data we have more precise the solution will be. We have to take into account that the probabilities have been done randomly without any criteria.

Conclusion

To conclude, this project has been a great experience to improve my skills as a programmer, to understand better how “face recognition” works and the importance of a big dataset. Also, I’m pleased that even though was a complicated project and I had a lot of trouble I could make the code work.
Furthermore, Markov chain has taught me some basic of Artificial Intelligence which will help me in a near future.
To end this project I want to thank my family for the pictures and specially the teacher for helping me even when was holidays.

Bibliography

[1] Face Recognition Systems: A Survey by Yassin Kortli , Maher Jridi [1], Ayman Al Falou [1] and Mohamed Atri

[2] Face Recognition: Understanding LBPH Algorithm by Kelvin Salton

[3] Evaluation of Haar Cascade Classifiers for Face Detection

Face Recognition using SIFT, SURF and PCA for Invariant Faces by Sukhvir Kaur and Prince Verma

SIFT Meets CNN: A Decade Survey of Instance Retrieval by Liang Zheng, Yi Yang and Qi Tian, Fellow, IEEE