|  |  |
| --- | --- |
| **Practicum Case** |  |
| COMP7116 | COMP7116001 | COMP7116016 | MATH6168 | MATH6168016  Computer Vision |
| **Computer Science** | **O211-COMP7116-RV01-10** |
| ***Valid on*** *Odd Semester 2020/2021* | **Revision 00** |

**Learning Outcomes**

* Various computational principles and standard image processing operators in computer vision
* The local features with their detectors and descriptors in computer vision
* Various features to find correspondence between images and perform recognition in computer vision
* Various image recognition system in computer vision

**Topics**

* Session 10 – Object Detection

## Subtopics

* Face Detection
* Face Recognition

**Object Detection**

Object detection is a computer technology which was related to computer vision and image processing that deals with detecting objects of a certain class (such as cats, dogs, humans, etc) in digital images and videos. Simply, object detection acts as a force which can draws a bounding box around each object of interest in the image and assigns them a class label. In short, object detection wraps up 2 part of computer vision application, which is image classification and also object localization. So for this course, we are going to detect those cool looking fast food logos, and try to detect and also recognize what’s the tested logo is. For this course, our folder structure goes like this :

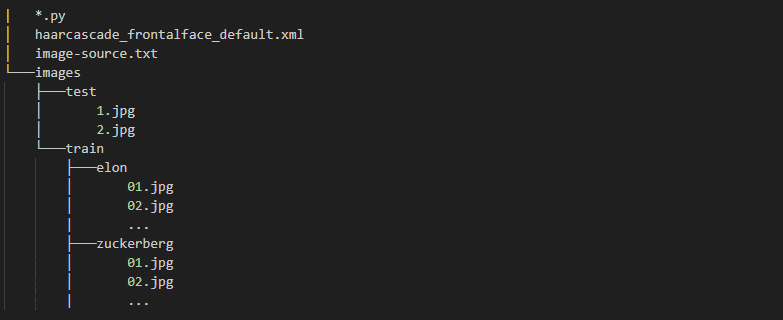


Figure . Folder structure

1. **Preparing the library**

For object detection, we will import 4 library which consists of

* **cv2** - library for image manipulation
* **os** - library for listing directory and its content
* **math** - library for basic built-in math library
* **numpy** - libary for modify python array



Figure . Importing library

1. **Building the classifier**

The first thing we need to do is building the object detection classifier. To do this, you can made it yourself or import from pre-made xml files, which was created by Paul Viola and Michael Jones. It has many object-oriented feature, like for this instance here we are going to use its face classifier. Full classifier from **Haar-Cascade** can be found in the link below :

<https://github.com/opencv/opencv/tree/master/data/haarcascades>



Figure . Building classifier from pre-made xml file

1. **Detecting faces and assigning class labels**

Now, we can use the earlier classifier object to detect faces from our training images. The first thing to do is of course iterate across our image paths and read our images one by one. Through each image, we use our classifier which we configure its **scaling factor** and **nearest neighbor**. After that, we can assign the detected face with its class name. Don’t forget to add our detected face to the list of detected images library.

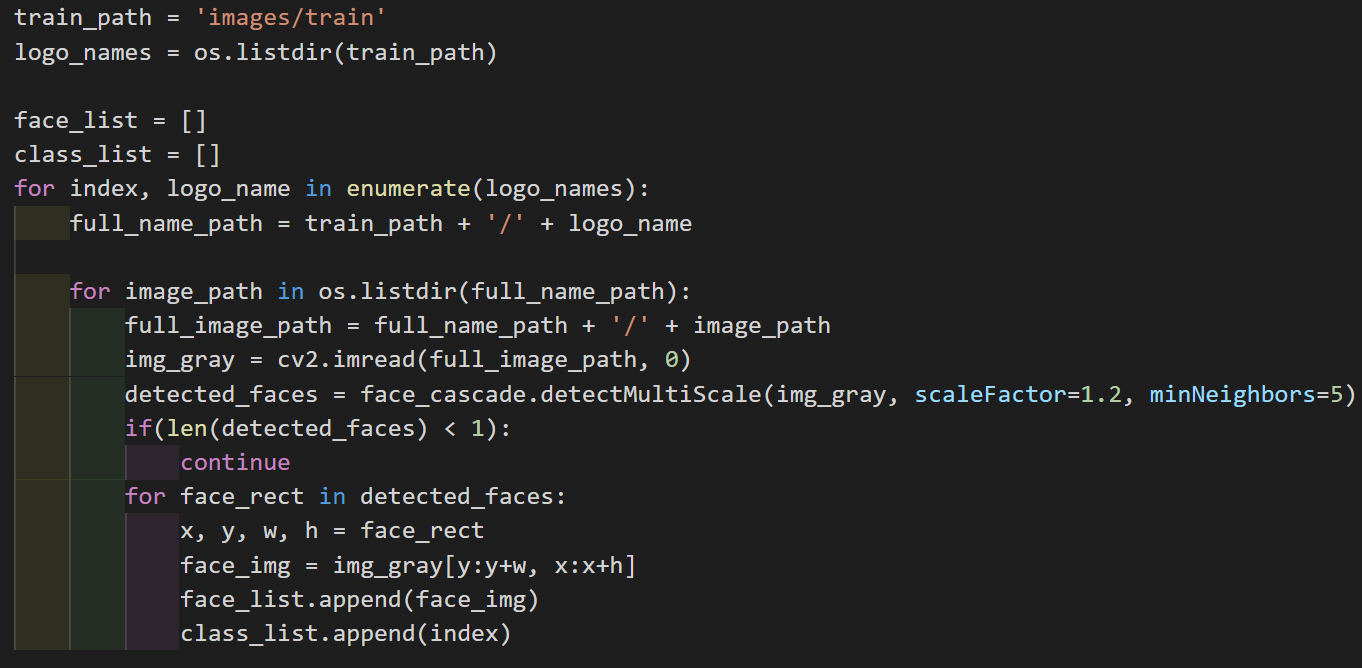


Figure . Detecting face and assign classifier

1. **Building the image recognition**

We already have our list of training images, specially created from the previous step. Now we can build our image recognition based on our training images. Simply image recognition is an algorithm which is responsible for finding characteristics which best describes the image. This characteristic then will be use to predict our tested image. For our image recognition, we could use the built-in face recognition from **cv2**, **LBPH** (**Local Binary Pattern Histogram**)

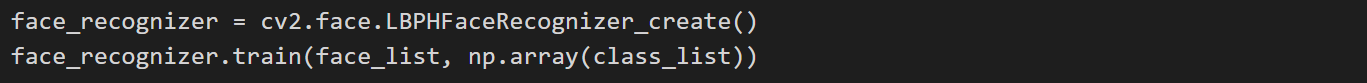


Figure 5. Building LBPH image recognition

1. **Predicting tested images**

We already build the necessary information and classifier, not to mention our image recognition. It’s time to predict our tested image based on our image recognition. To see if our image recognition is confident enough with its prediction, we can calculate its loss percetage. Higher loss here means that it’s unsure of its prediction.

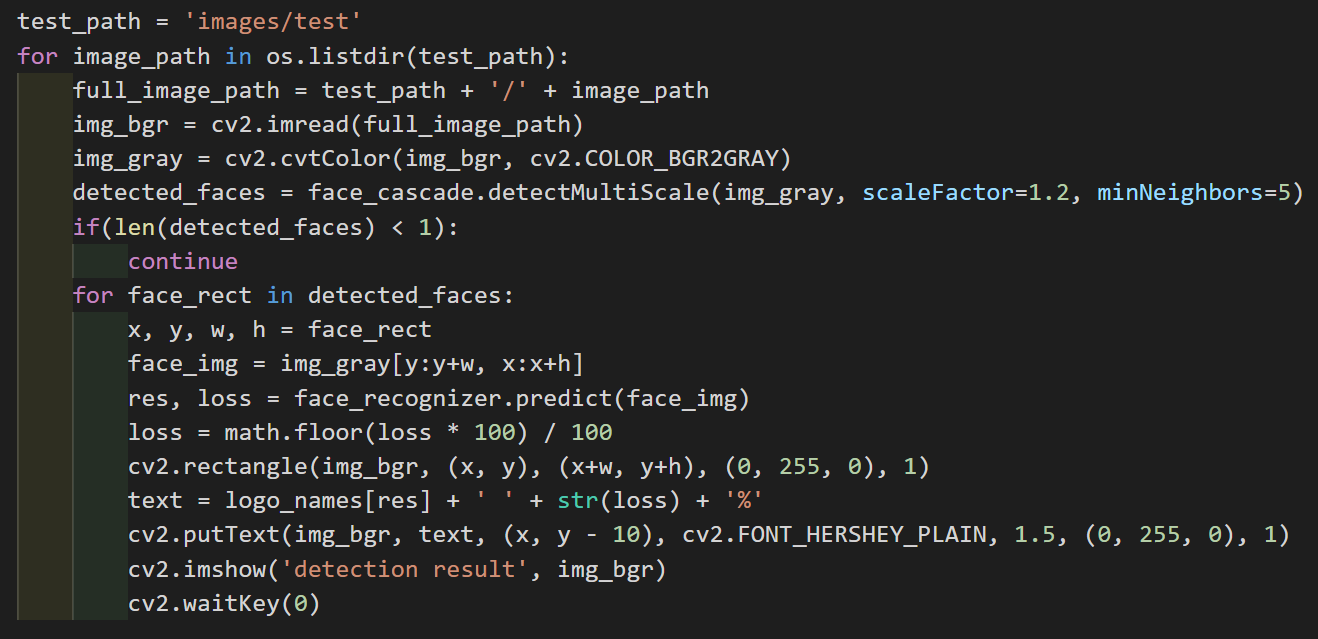
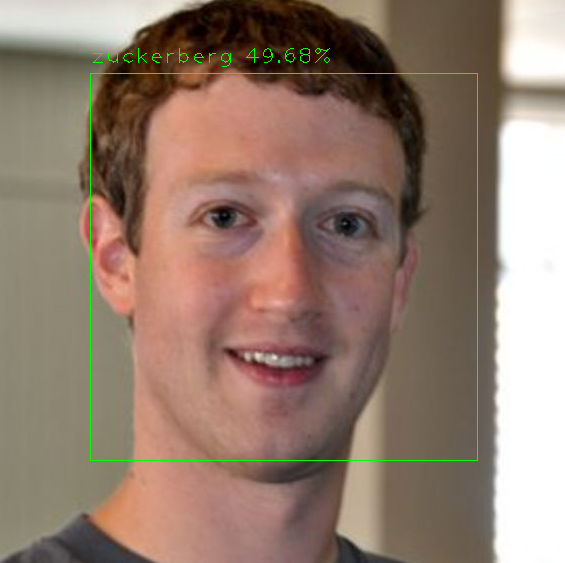


Figure 6. Predicting images with trained image recognition

Result :



We can see that Zuckerberg’s picture has a high loss. This is mostly caused by some disturbances such as:

* The image dimension across training and testing is too different. This can cause in fewer or higher detection, depending on the dimension
* The image is too blurry. While this is a trivial thing that most people find it not so bad, in reality the image that we need for training and testing must be in clear condition as possible, that means few noise and possibly no blur
* The main focus of image blends too well with the background. This may cause the classifier fails to detect the possibly object of interest
* Distance. The distance the image was taken can impact greatly with the size of object of interest

Fixing the situation can be done with some solution like :

* Try to make the dimension across all images as evenly as possible
* Try to train and test image with few noise and blur
* Try to make the object of interest stand out from its background color

**References:**

**Test**

* <https://lh3.googleusercontent.com/proxy/IOR6MvwKpazFHfh01mlmGshy85iJfKJimy9BjaC4xL3ZsCCGhOrqREd7n7pK114KbR7B9y7TuU-bBqSUb9Ou38Tw6OSfBeNjDw>
* <https://vignette.wikia.nocookie.net/theofficalsovietronalds/images/3/3a/Kfc.jpg/revision/latest?cb=20120609221350>
* <https://upload.wikimedia.org/wikipedia/commons/thumb/4/49/Elon_Musk_2015.jpg/800px-Elon_Musk_2015.jpg>
* <https://ichef.bbci.co.uk/news/976/mcs/media/images/57036000/jpg/_57036173_markzuckerbergfounderoffacebook2.jpg>

**Train (Elon)**

* <https://getwallpapers.com/wallpaper/full/1/e/c/516645.jpg>
* <https://getwallpapers.com/wallpaper/full/0/8/4/516023.jpg>
* <https://getwallpapers.com/wallpaper/full/0/e/3/515854.jpg>
* <https://i.pinimg.com/564x/94/da/f8/94daf812e4a480ffc7b025ad9014b557.jpg>
* <https://www.pixel4k.com/wp-content/uploads/2018/09/elon-musk_1536946305.jpg.webp>

**Train (Zuckerberg)**

* <https://wallpapercave.com/wp/wp2126146.jpg>
* <https://wallpapercave.com/wp/wp2126169.jpg>
* <https://wallpapercave.com/wp/wp2126173.jpg>
* <https://wallpapercave.com/wp/wp2126189.jpg>
* <https://wallpapercave.com/wp/wp2126275.jpg>