*Face Analyzer  
Predicting Gender using OpenCV*

Table of Contents

[Overview 2](#_Toc79737213)

[Scope 2](#_Toc79737214)

[Pre-requisites 2](#_Toc79737215)

[Setup 2](#_Toc79737216)

[Final Words 9](#_Toc79737217)

Overview

Automatic prediction of gender from face images has drawn a lot of attention recently, due to its wide application in various facial analysis problems. However, due to the large variations of face images (such as variation in lighting, scale and occlusion) the existing models are still behind the desired accuracy level which is necessary for exploiting these models in real-world applications.

Scope

The target of this tutorial is to develop a lightweight command line based utility, through Python based modules to automatically detect faces in a static image and to predict the gender of the spotted persons using a deep learning-based gender detection model.  
  
If this tutorial intrigues you, then grab its code from the following GitHub repository: “<https://github.com/bassemmarji/FaceAnalyzer/> ”.

# Pre-requisites

The following components come into play:

* **OpenCV**: is an open-source library for computer vision, machine learning and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java and it is used for all sorts of image and video analysis like facial detection and recognition, photo editing, optical character recognition and a whole heap more.
* **Dlib**: is principally a C++ library however you can use a number of its functions from Python applications.
* **filetype**: is a small and dependency free Python package to infer file and MIME types.
* **imutils**: encompasses a series of functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges.

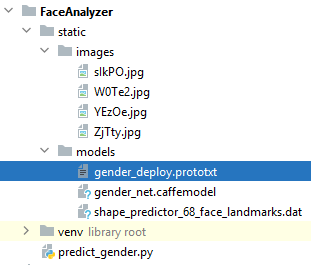
# Setup

To setup the environment, you need python3 installed on your system. It is highly recommended to setup a virtual environment which will host the needed libraries.

1. Create a virtual environment and activate it.
2. Create a file named requirements.txt and add the following lines to it.

|  |
| --- |
| requirements.txt |
| opencv-python==4.4.0.46 dlib==19.17.0 imutils== 0.5.3  filetype ==1.0.7  imutils== 0.5.3 |

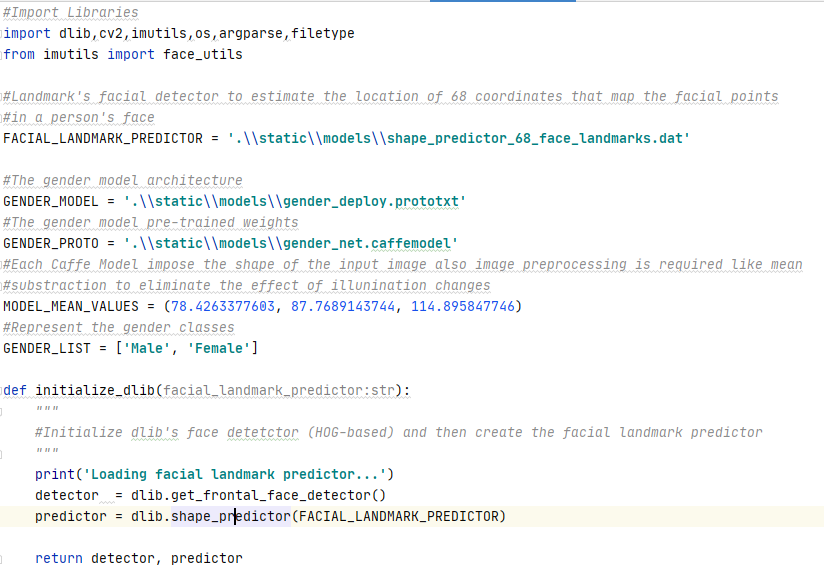
1. Now, let’s install the required libraries to the project.  
   pip install –r requirements.txt
2. Create a folder for our project called “FaceAnalyzer”.

At the end, our folder structure will look like the following:  
  
  
**NB:**   
  
For the purpose of this article we will use a pre-trained Caffe model developed by Gil Levi and Tal Hassner.

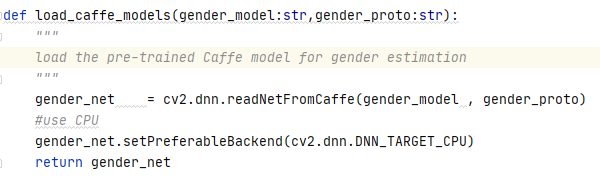
* gender\_net.caffemodel: contains the pre-trained weights.
* gender\_deploy.prototxt: is the model architecture (a plain text file with a JSON like structure containing all the neural network layer’s definitions).

OpenCV Deep Neural Network ‘DNN’ module supports many deep learning frameworks among which Caffe, TensorFlow, PyTorch…  
You can download this model using the following link: <https://talhassner.github.io/home/publication/2015_CVPR>.  
  
Additionally we will use the “shape\_predictor\_68\_face\_landmarks.dat”; A dlib pre-trained model to estimate the location of 68 coordinates (x,y) that map the facial points on a person’s face. The model details are available in the below link   
<https://github.com/davisking/dlib-models>.

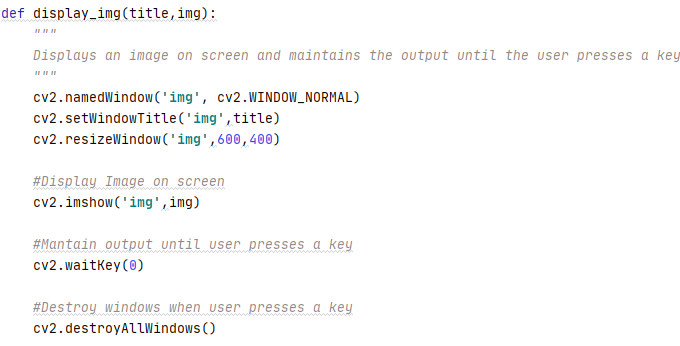
Let’s move into coding:  
 *#predict\_gender.py*



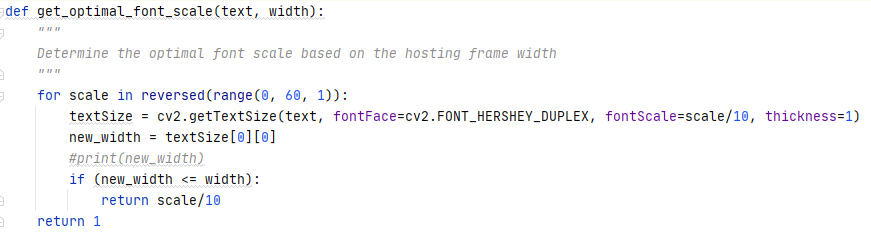
* This function initializes the dlib using the pre-trained model and returns:
  + detector: used for detecting the face in an image.
  + predictor: shape or landmark predictor used to predict the coordinates of a given shape. The facial landmark predictor is used to localize individual facial structures.



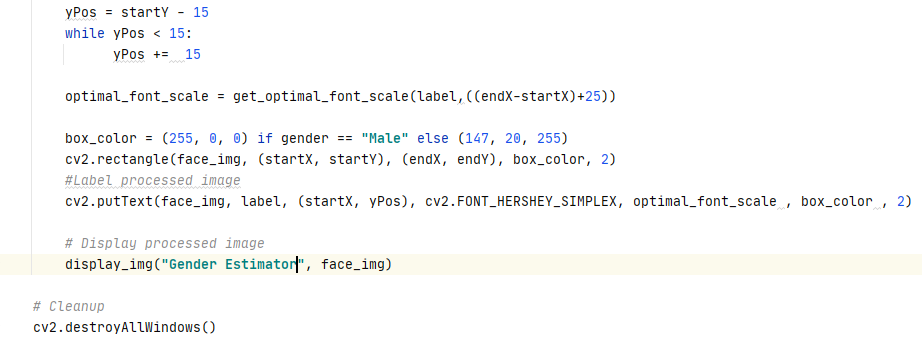
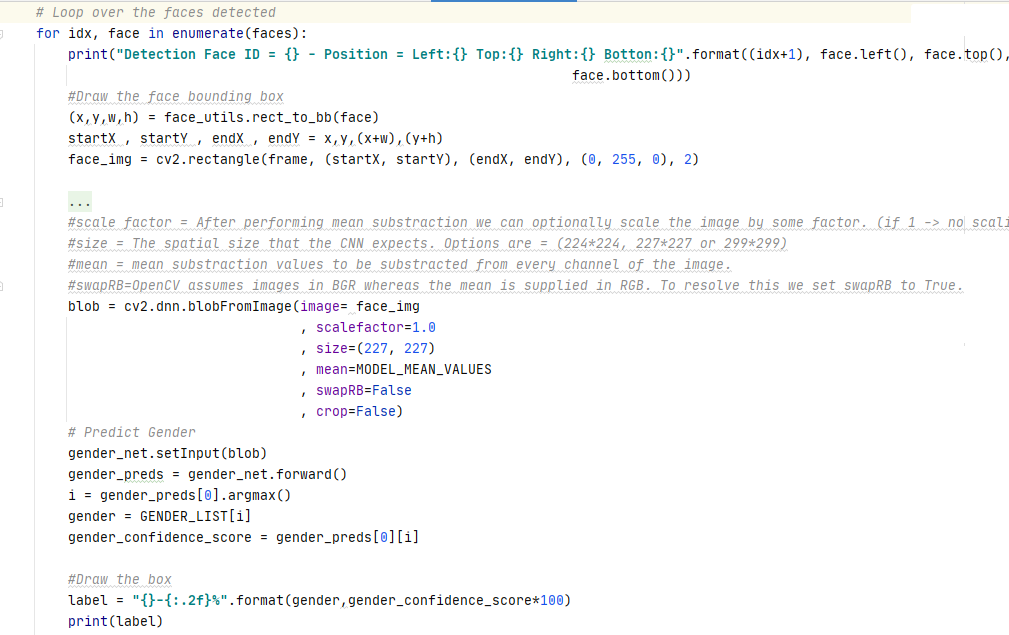
* This function loads the pre-trained Caffe model for gender estimation.



* This function displays an image on screen and maintains the output the user presses a keyboard key.



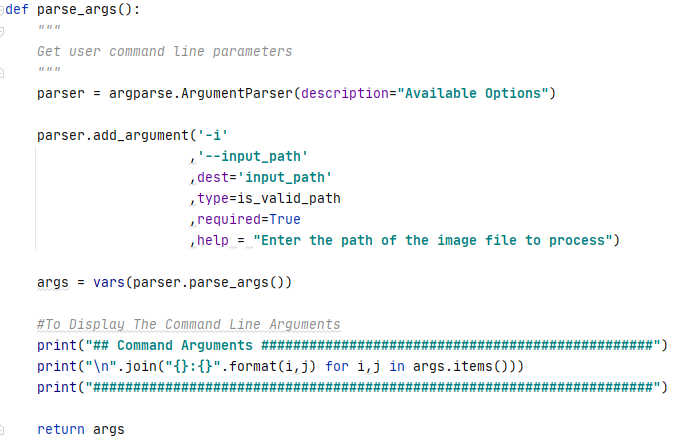
* This function determines the optimal font scale based on the width of a hosting frame.



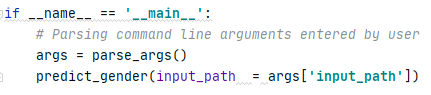
* This function constitutes the core of our program and performs the following:
  + Initialize the dlib face detector.
  + Load the gender estimation model.
  + Reads the input image.
  + Copies and resizes the original image in order to label the copied image.
  + Converts the image into gray scale.
  + Detects faces in the grayed image.
  + Iterates throughout the faces detected and displays the positions of the faces detected.
  + Draws a bounding box around the faces detected.
  + Creates a blob using the image data (Region of Interest) that we will use for gender classification.
  + Feeds the blob to the gender classifier and picks the “Class ID” with the highest confidence score.
  + Draws a frame around the face signaling the estimated age and a confidence score indicating the accuracy of the estimation. Depending on the gender whether Male or Female, the color of the bounding box will change.
  + Displays the labeled image.



* This function validates a path inputted as a parameter and ensures that it is a file path also it ascertains that the type of the file chosen is an image.



* This function defines and sets the appropriate constraints for the command line arguments to be specified by the user when running this utility.
  + input\_path: A required parameter containing the path of the image file to process associated with the predefined function “is\_valid\_path”.

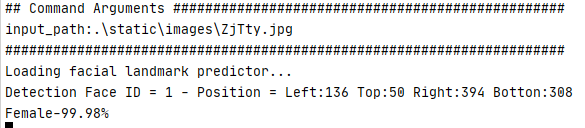


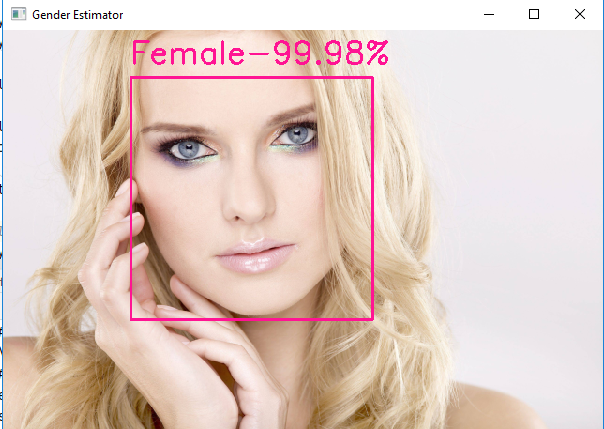
* The above represents the main function of our program.

Let’s test our program:  
  
Kindly proceed as per the following steps:

1. Open up a terminal window and type the following in it:

**predict\_gender –i ".\static\images\ZjTty.jpg"**

The following summary will be displayed in the terminal:  
  




Final Words  
  
The Levi and Hassner deep learning gender detection model seems accurate to a certain extent.  
Hope you enjoyed this article and you found it useful.

|  |  |
| --- | --- |
|  | ***Bassem Marji*** *is a project implementation manager at BLOM Bank with a proven track record of success.  He managed the implementation of over 50 projects and propelled the digital transformation of mission critical applications. He spends his free time discovering the latest technology trends in the IT field.* |