*Face Analyzer  
Predicting Age using OpenCV*

Table of Contents

[Overview 2](#_Toc79673784)

[Scope 2](#_Toc79673785)

[Pre-requisites 2](#_Toc79673786)

[Definitions 3](#_Toc79673787)

[Setup 3](#_Toc79673788)

[Final Words 9](#_Toc79673789)

Overview

Recently, wide attention has grown in the field of computer vision, especially in face recognition, detection, and facial landmarks localization. Many significant features can be directly derived from the human face, such as age, gender, and emotions.   
  
Age estimation (AE) can be defined as the automatic process of classifying the facial image into the exact age or to a specific age range. Basically, age estimation from the face is still a challenging problem and guessing an exact age from a single image is very difficult due to factors like makeup, lighting, obstructions and facial expressions.

Scope

Inspired by many ubiquitous applications spread across multiple channels like “[AgeBot](https://play.google.com/store/apps/details?id=com.testa.agebot&hl=en_IN" \t "_blank)” on Android, “Age Calculator” on IPhone, we are going to build a simple Age estimator using OpenCV, Deep Learning, and Python.   
  
The main purpose of this tutorial is to develop a lightweight command line based utility, through Python based modules and it is intended to describe the steps to automatically detect faces in a static image and to predict the age of the spotted persons using a deep learning-based age 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.  
    
  Using OpenCV come with many benefits among which:
  + OpenCV is an open-source library and it is free of cost.
  + OpenCV is fast since it is written in C/C++.
  + OpenCV supports most of the Operating Systems like Windows, Linux and MacOS.
* **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.

# Definitions

# **Computer vision:** It is one of the most interesting and challenging tasks in Artificial Intelligence and acts like a bridge between Computer Software and visualizations around us. It allows the computer software to understand and learn about the visualizations in the surroundings.

**Deep Learning**: is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

**Convolutional Neural Network (CNN)**: is a Deep Learning algorithm which can take an input image, assign importance (learnable weights and biases) to various aspects/objects in the image that in order to differentiate one from the other.

**Caffe**: is a deep learning framework made with expression, speed and modularity in mind.

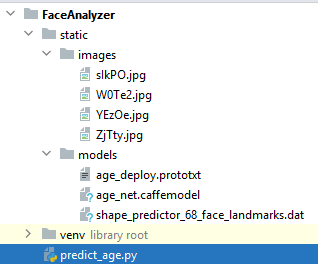
# Setup

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 |

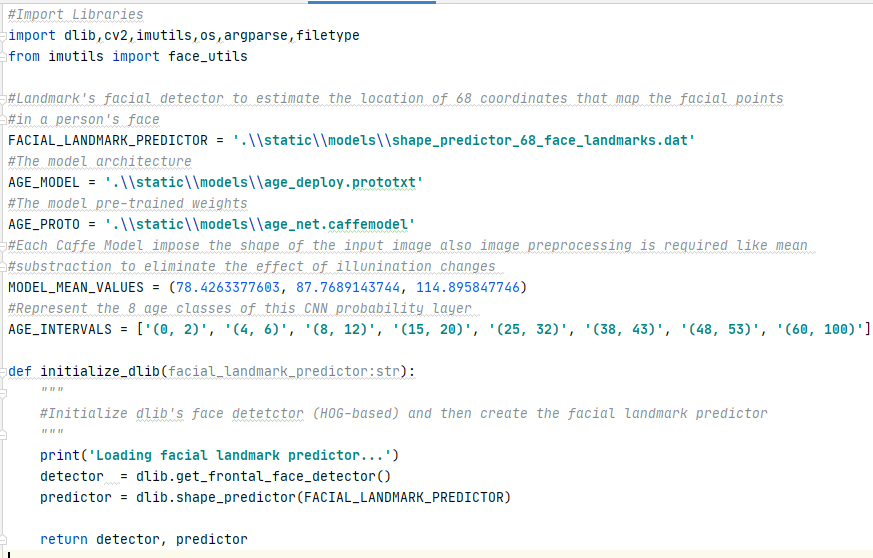
1. Now, let’s install the required libraries to the project.  
   pip install –r requirements.txt
2. Create a folder for our project called “FaceAnalyzer”.

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 in 2015.

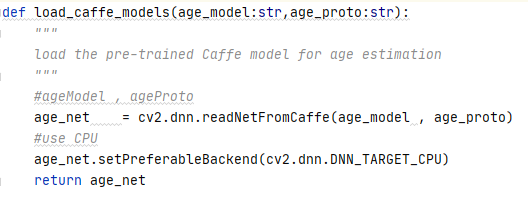
* age\_net.caffemodel: contains the pre-trained weights.
* age\_deploy.prototxt: is the model architecture (a plain text file with a JSON like structure containing all the neural network layer’s definitions).

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>.

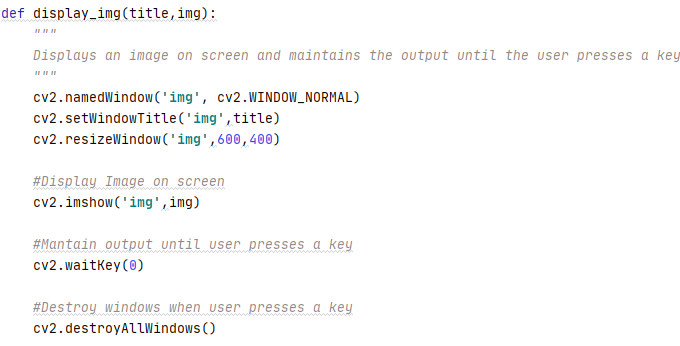
Let’s move into coding:  
 *#predict\_age.py*



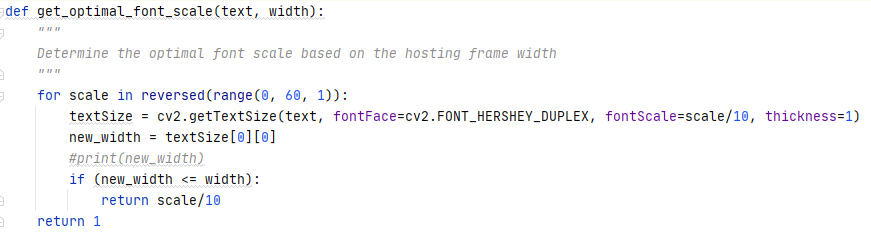
* This function initializes the dlib using the pre-trained model and returns:
  + detector: used for detecting the face in an image.
  + predictor: shape or landmark predictor used to predict the coordinates of a given shape. The facial landmark predictor is used to localize individual facial structures.



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

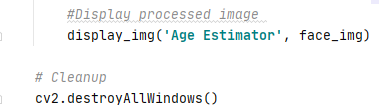
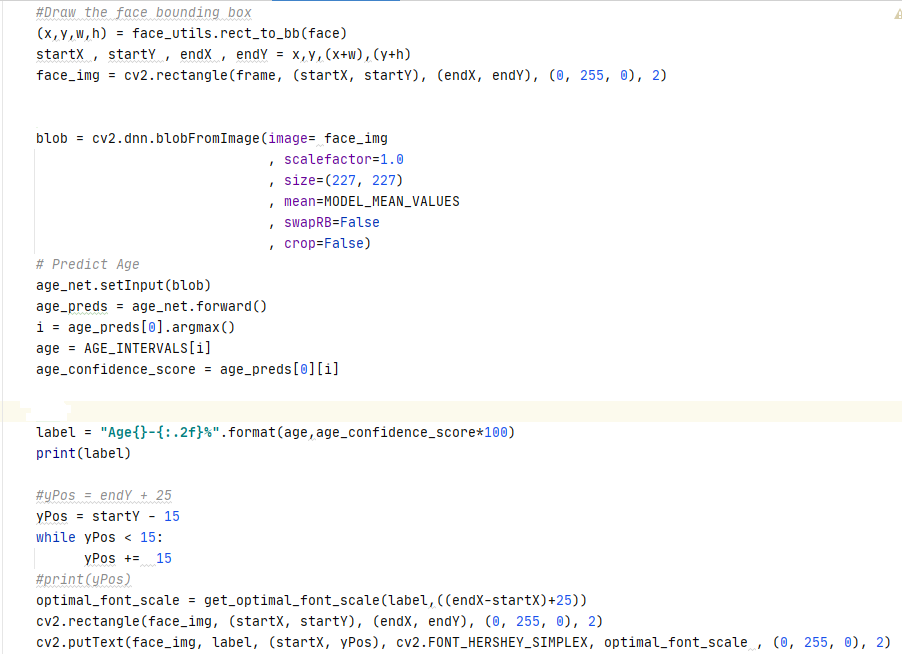


* This function displays an image on screen and maintains the output the user presses a keyboard key.

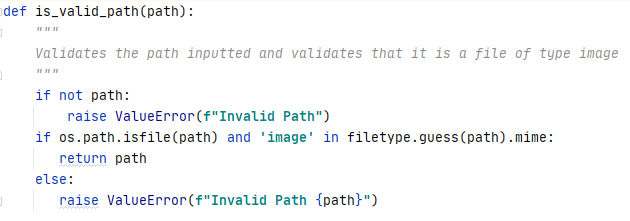


* This function determines the optimal font scale based on the width of a hosting frame.

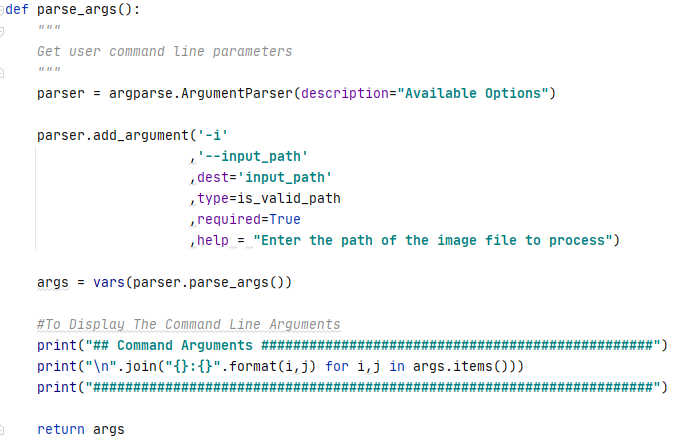




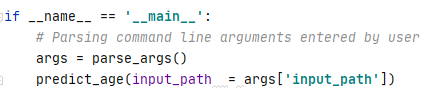
* This function constitutes the core of our program and executes the following:
  + Initialize the dlib face detector.
  + Load the age estimation model.
  + Reads the input image.
  + Copies and resizes the original image in order to label the copied image.
  + Converts the image to gray scale.
  + 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 age classification.
  + Feeds the blob to the age 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 estimation accuracy.
  + 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 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”.

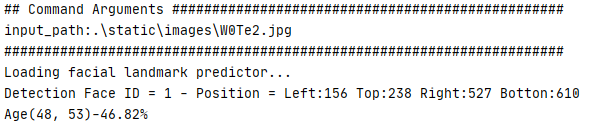


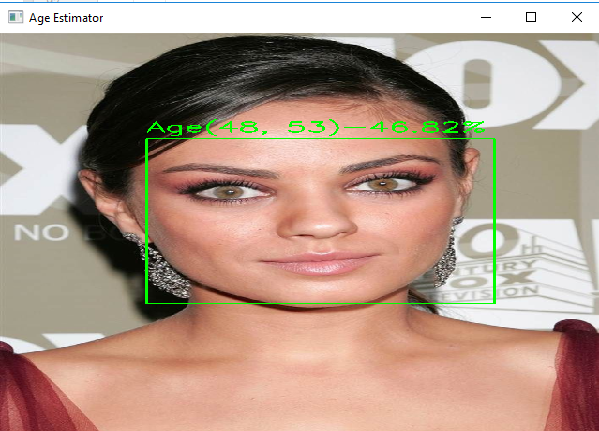
* The above represents the main function of our program.

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\_age -i** ".\static\images\W0Te2.jpg"

You will get the following output in the terminal:  
  




Final Words  
  
The Levi and Hassner deep learning age detection model is heavily biased toward the age group [25-32] therefore you may pinpoint this discrepancy while testing this utility.  
  
Hope you enjoyed this article.

|  |  |
| --- | --- |
|  | ***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.* |