

main

April 29, 2018

1 Face detection and object detection

1.1 Part 1 - Face Detection

In this project, we aim to construct a face detection model. We used a method haar to extract features. After that, by applying extracted features to cascade method, we were able to detect people's faces and also count the number of faces through pictures as well as webcam. Pre-trained cascade was used from OpenCV.

We suggest clone or download the whole repository to reproduce the result. You can reproduce them all from this notebook. Besides, you are encouraged to add your own test images simply by putting all the images you want to test into the data/test_data/cascade or data/test_data/tensorflow directory.

You can hit the button below to show the full codes and hit again to collapse all cells.

Note that you may need to set the working directory to the "doc" folder by hand before you start reproduce the results, if not by default. Otherwise, you may run into some directory issues. It is always safe to set the directory back to "doc" before you run any of the functions below. You can do that through running "os.chdir('Put your relative path or absolute path to the doc folder here')".

1.1.1 Part 1.1 - Face Detection without Rotation on Image

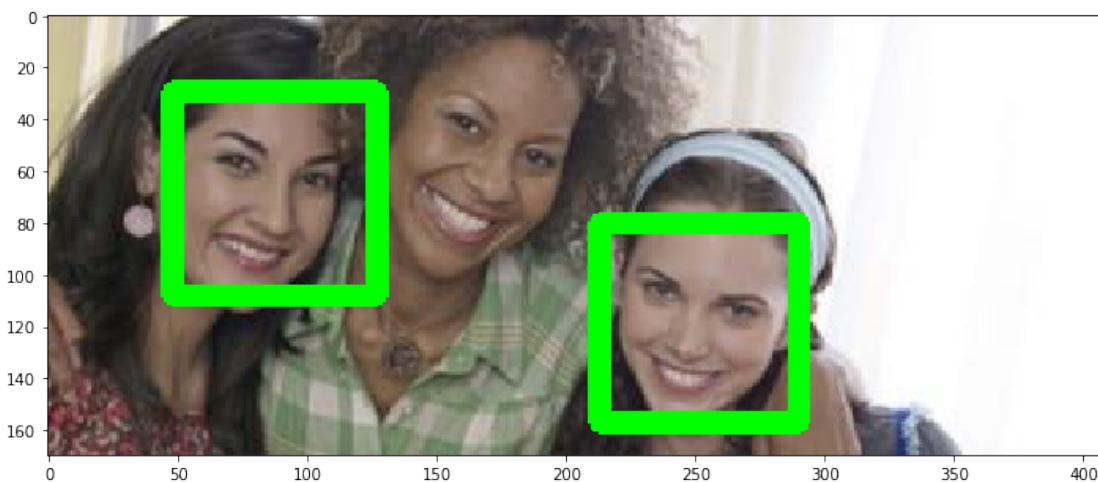
We started off the project with face detection on an image.

```
In [167]: # Collapse cells
    HTML('''<script>
code_show=true;
function code_toggle() {
    if (code_show){
        $('div.input').hide();
    } else {
        $('div.input').show();
    }
    code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
<form action="javascript:code_toggle()"><input type="submit" value="Click here to togg
```

```
Out[167]: <IPython.core.display.HTML object>
```

```
In [1]: import os
import sys
import cv2
import numpy as np
import count_face as cf
import counting_faces_image as cfi
import counting_faces_webcam as cfw
import tensorflowFn as tf_image
import tensorflowFnVideo as tf_video
import ipywidgets as widgets
from pathlib import Path
from matplotlib import pyplot as plt
from IPython.display import HTML
from IPython.display import display
```

```
In [2]: # Apply face detection
cf.counting_face()
```



1.1.2 Part 1.2 - Face Detection with Rotation

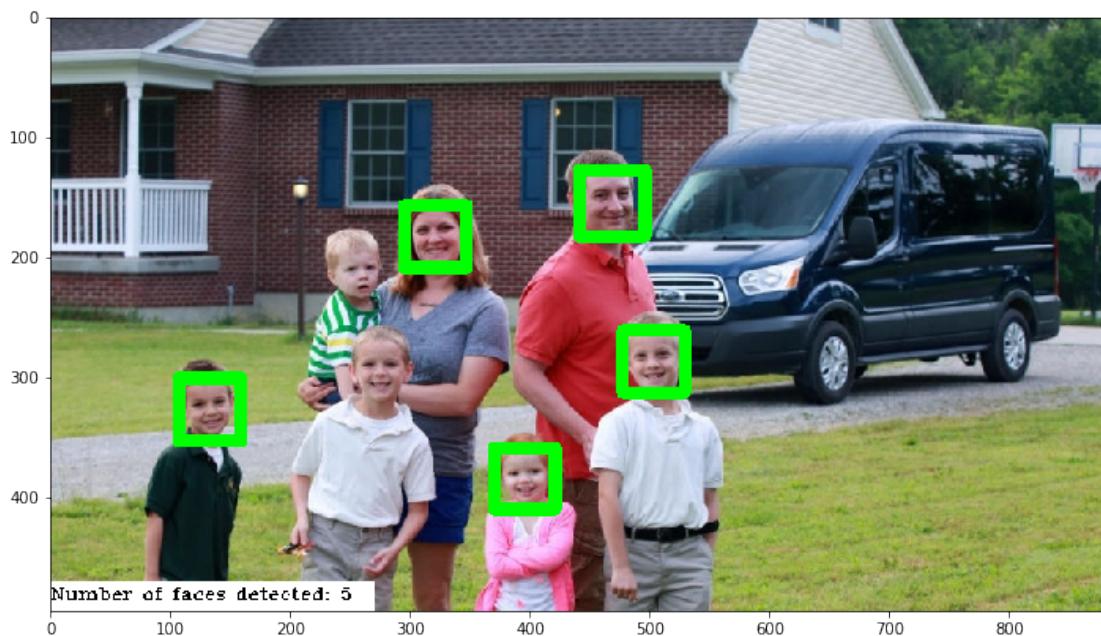
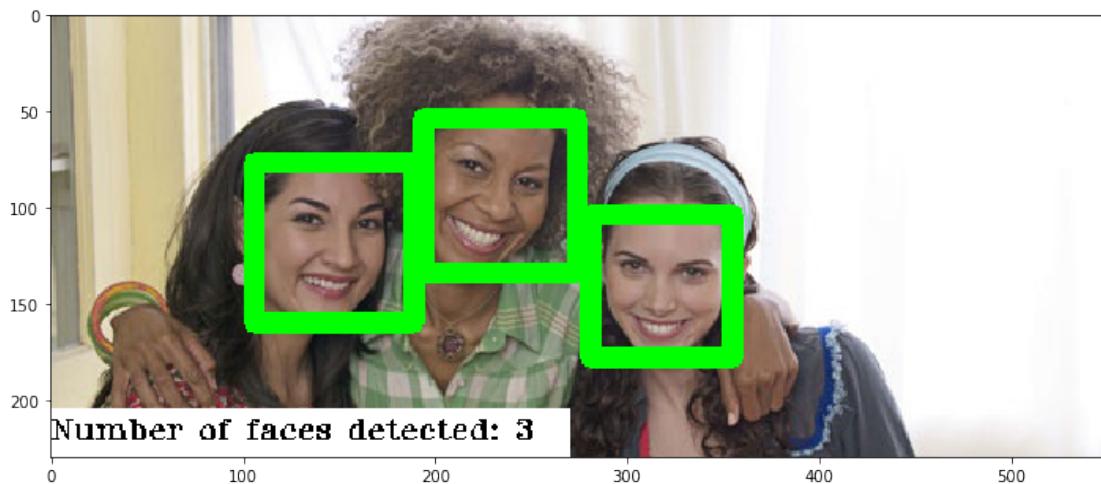
After implementation of Part 1, we realized the OpenCV front face cascade could not detect rotated face. Hence, we made some adjustments and declared additional functions to allow our model to analyze images from different rotation angle.

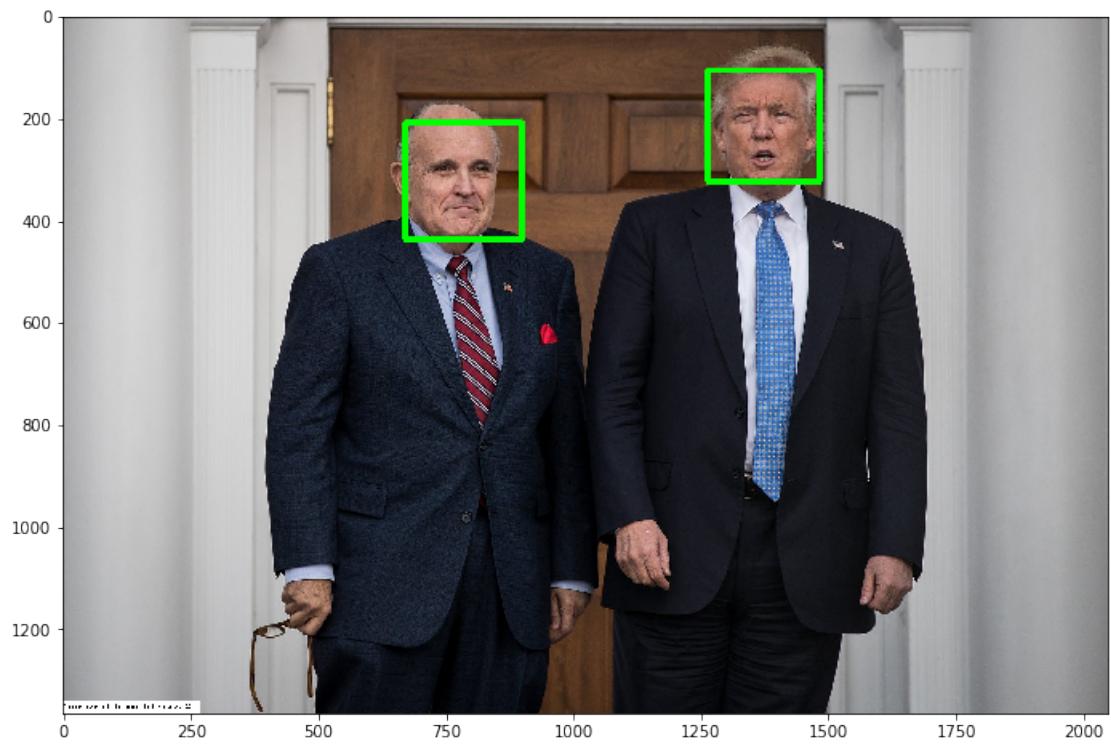
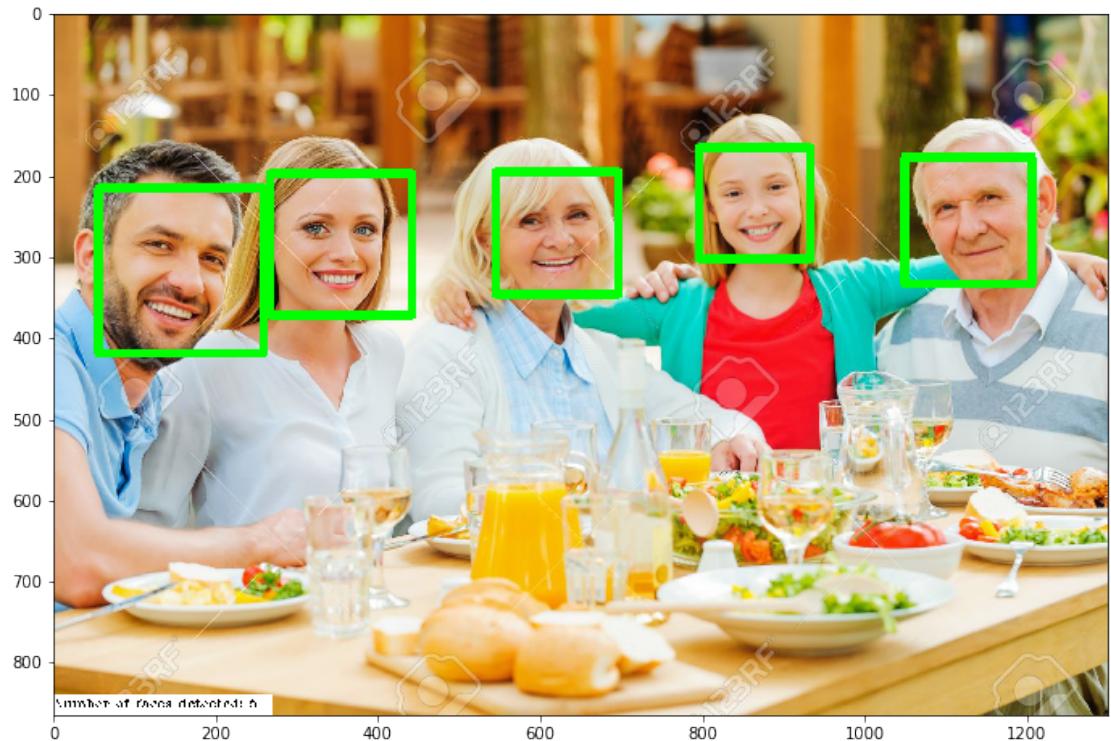
```
In [3]: '''
```

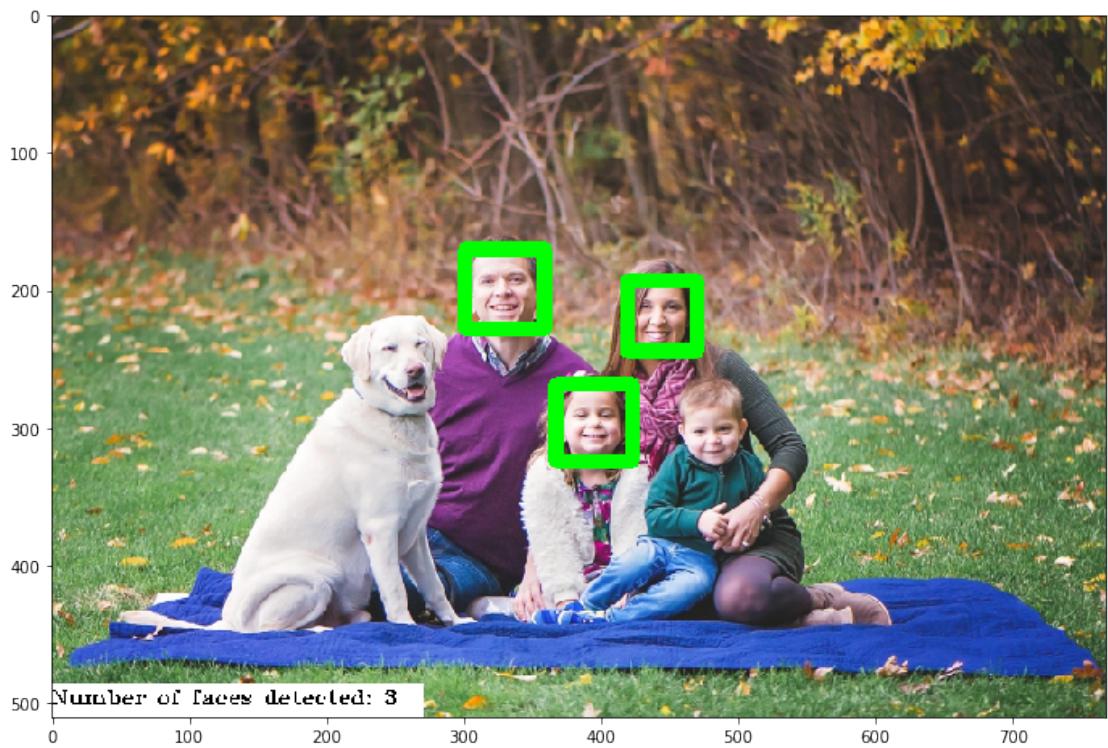
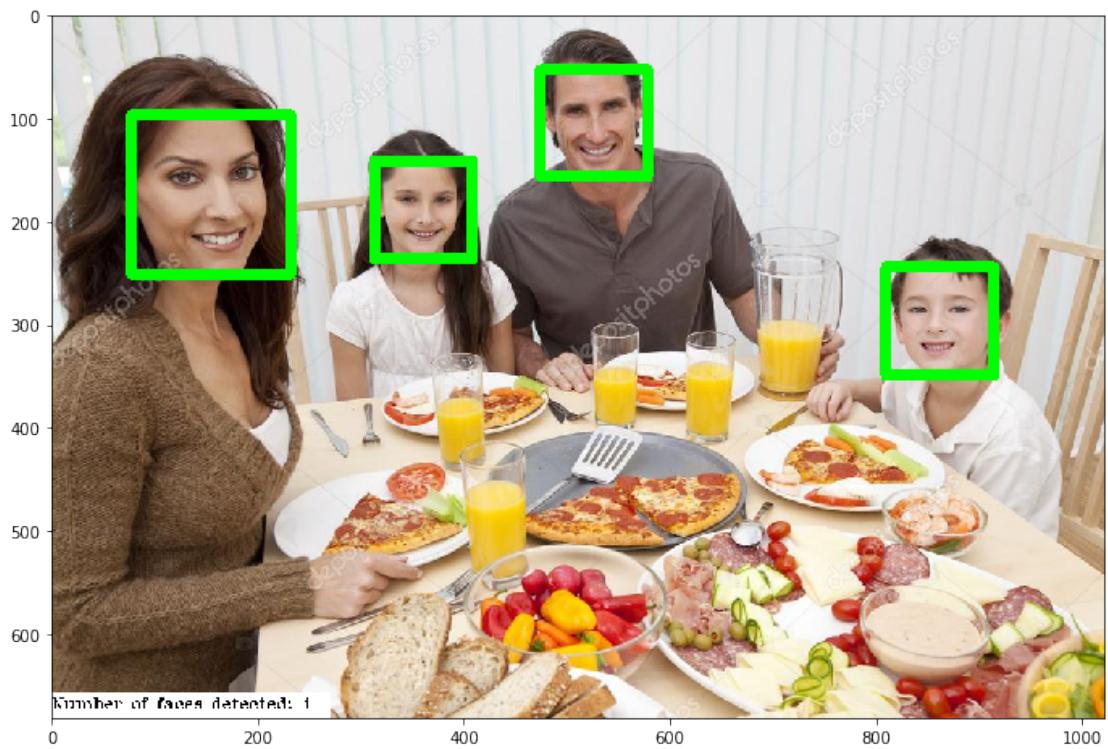
You can put your own images into the directory: '../data/test_image/cascade'

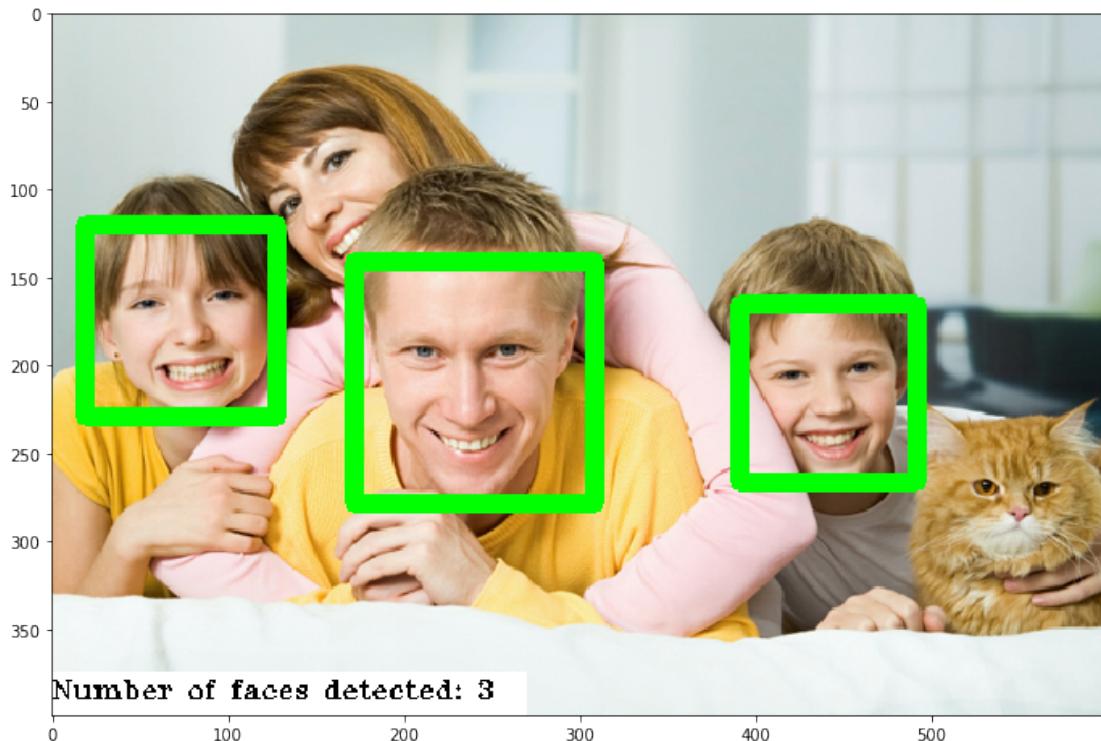
```
'''
```

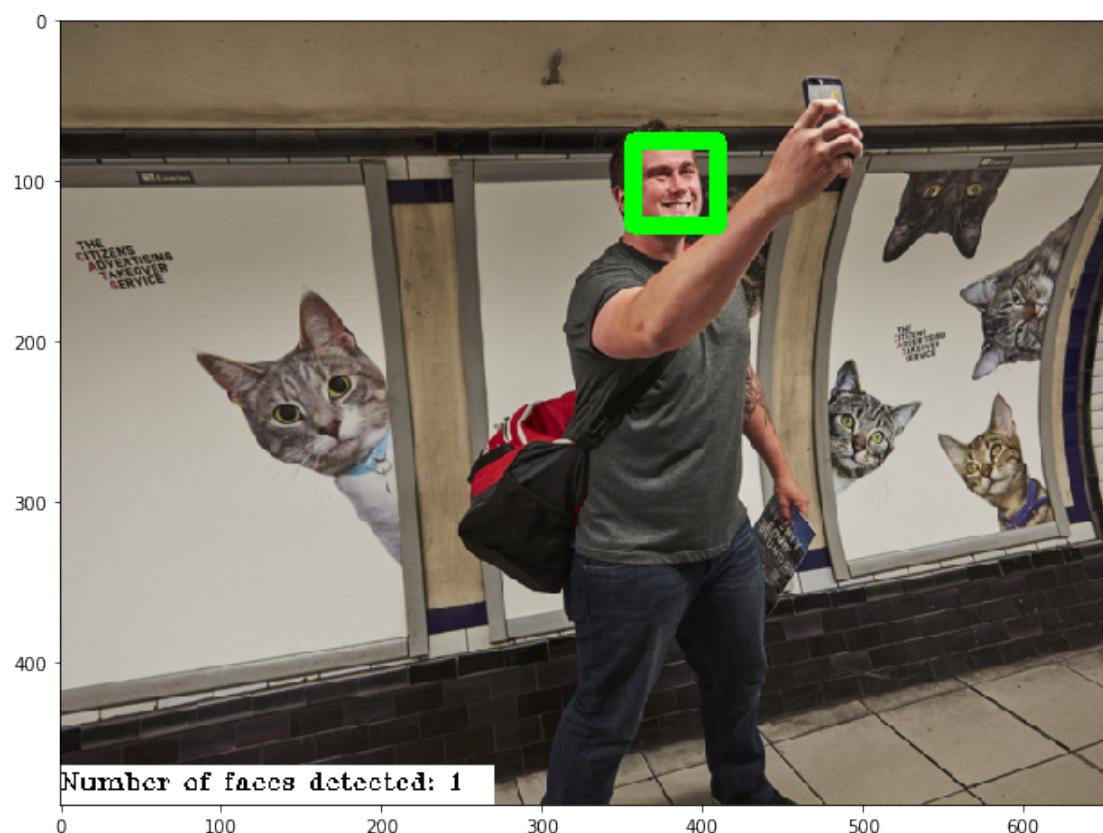
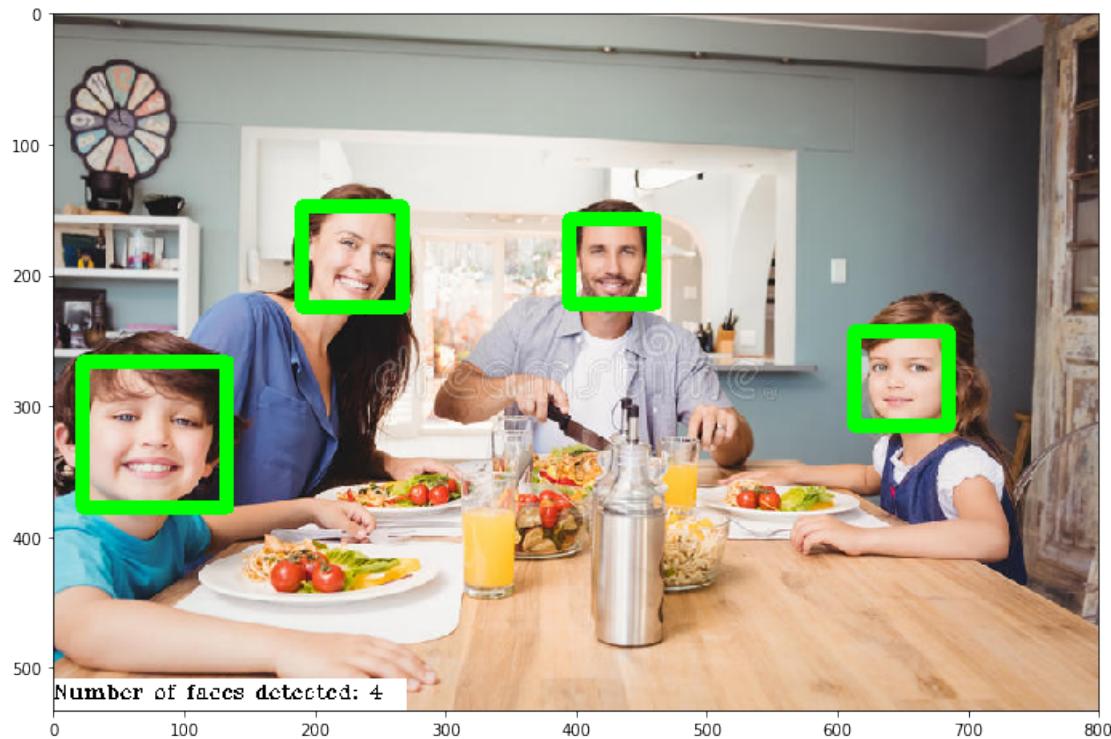
```
cfi.face_dectect_image('..../data/test_image/cascade/')
```











1.1.3 Part 1.3 - Real Time Face Detection with WebCam

After we improved our model, we wanted to further develop our model. Therefore, in this part, we implemented real time face detection using WebCam. You can click on the button below to release the cam. Press "Q" to quit the cam.

In [6]: '''

Press Q to close the camera

*If the cam doesn't close, that is an error introduced by 'opencv' package on MacOS system.
You can simply leave the window and go to the next section.*

'''

`cfdw.face_dectect_webcam()`

1.2 Part 2 - Object Detection using Tensorflow API

After we constructed our model in Part 1, we realized there exist some limitations in cascade model. Cascade model tends to have lower accuracy in side faces or partially showed faces. Also, cascade cannot detect highly rotated faces. To overcome such limitations, a popular and powerful approach is the application of Convolutional Neural Network through TensorFlow. In this section, we implement object detection using a frozen inference graphical model from Tensorflow Object Detection API -- 'ssd_mobilenet_v1_coco'. This model requires intallation of tensorflow. Further instruction of the installation can be referred to https://github.com/tensorflow/models/tree/master/research/object_detection. This model can detect and categorize object, including person, bottle, cellphone, etc. However, cascada model would result better if only faces are showed on an image while this API model would result better if more parts of human body are showed.

1.2.1 Part 2.1 - Object Detection API with Tensorflow on Image

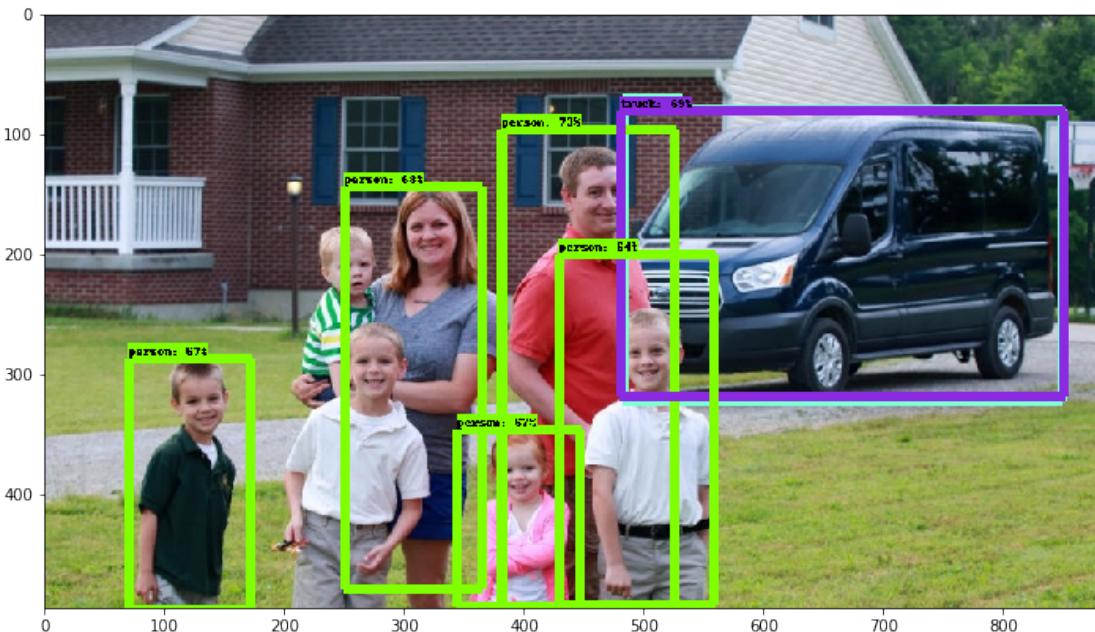
Similar to Part 1.1, we started off with object detection using image. We can see now the model can not only detect human faces but the entire human figure. Moreover, this more advanced model allows us to detect objects other than human, for example, dogs, cats, bottles, cell phones, clocks and so on.

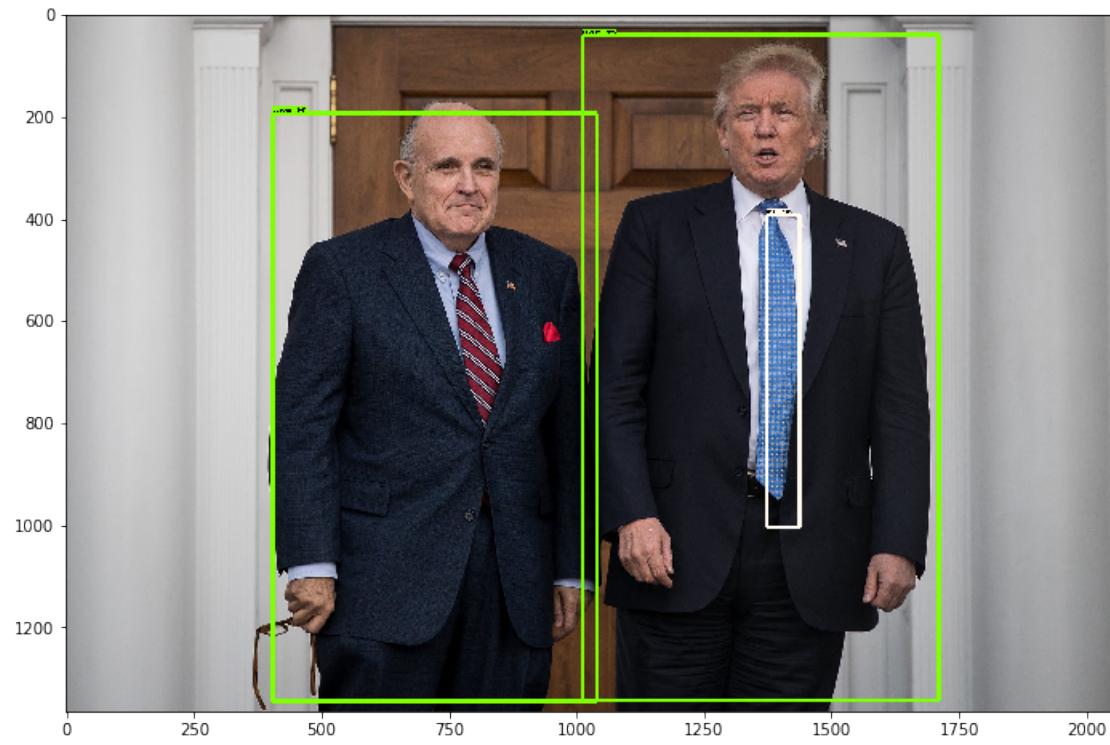
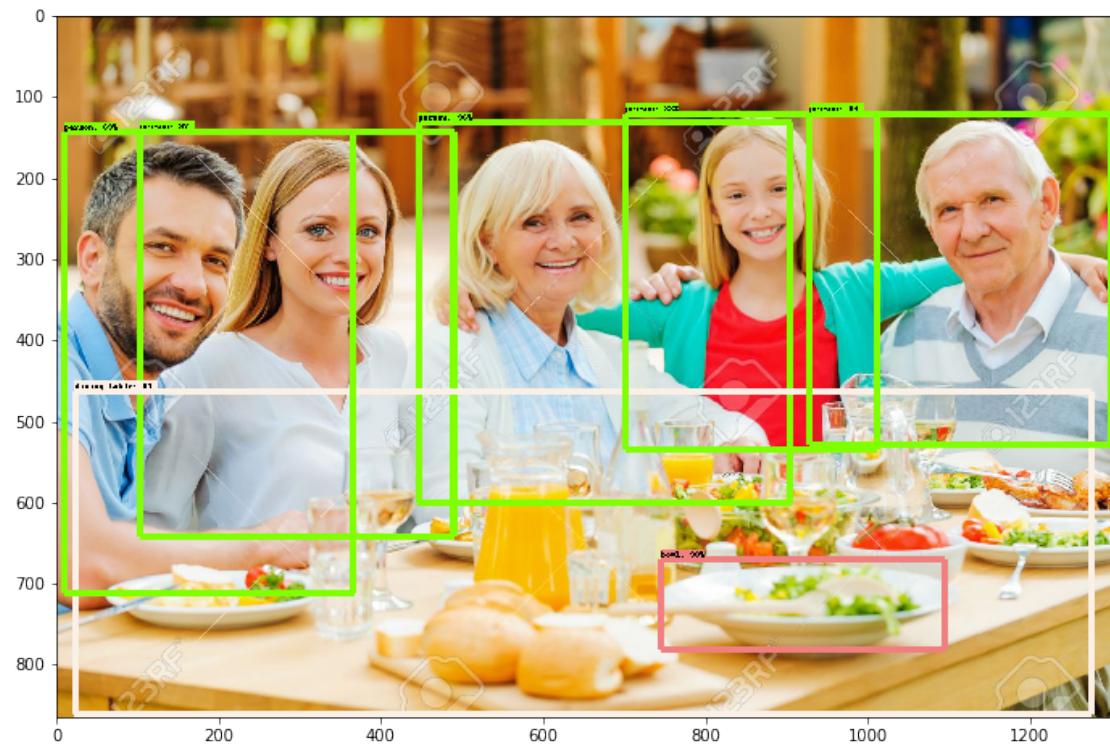
In [5]: '''

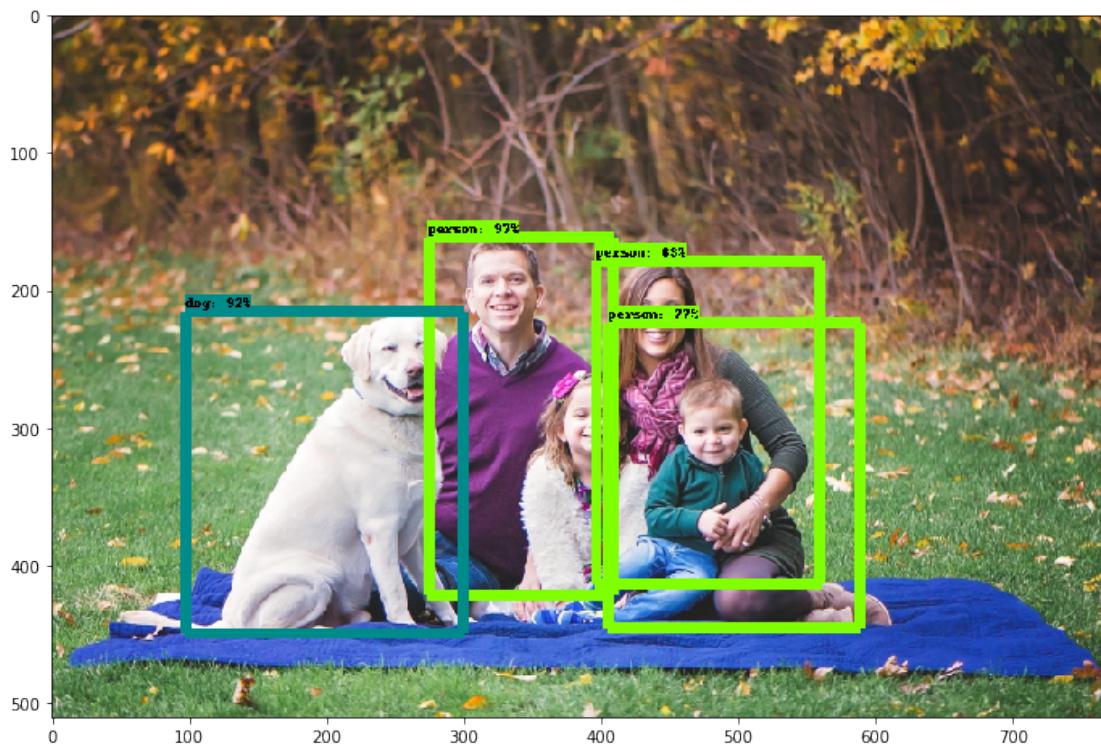
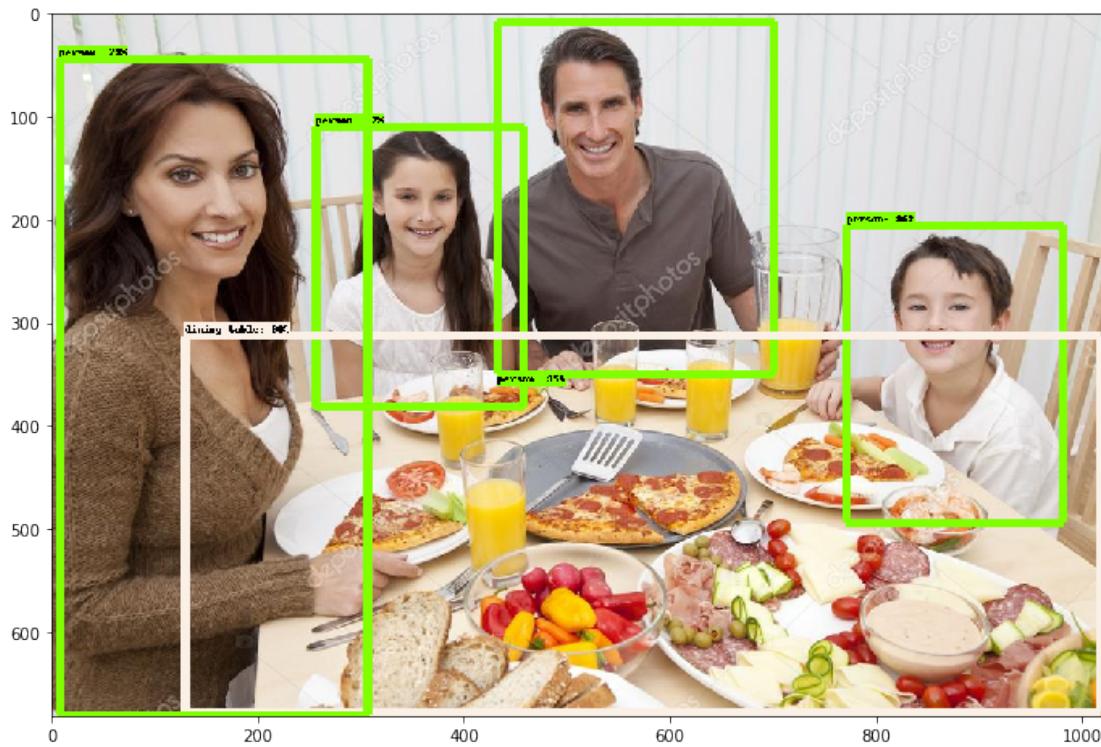
*You can put your own images into the directory: '../data/test_image/tensorflow'
It may take a few seconds to do the detection. So please be patient.*

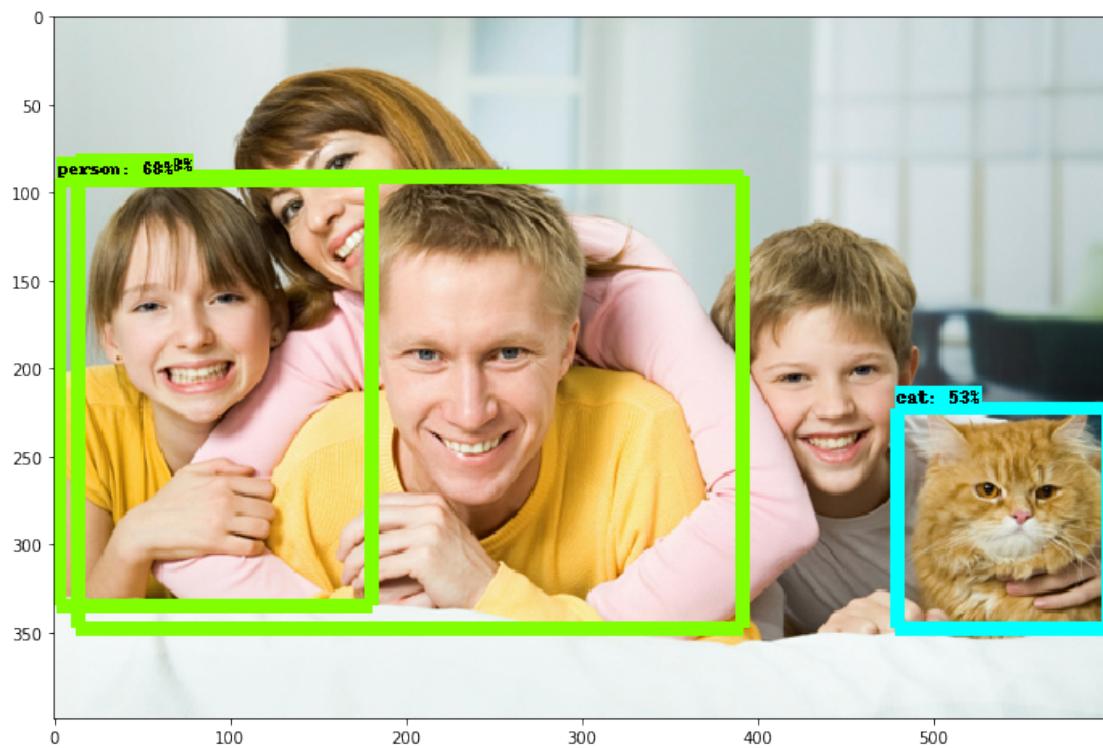
///

```
tf_image.objectDetection('.../.../data/test_image/tensorflow')
```









1.2.2 Part 2.2 - Real Time Object Detection API with Tensorflow using WebCam

Part 2.1 resulted in highly accurate result detecting most of the objects into their corresponding categories. We further improved the model by implementing the model using WebCam as an input. The result was again very accurate. You can click on the button below to release the cam. Press "Q" to quit the cam.

In [7]: 

Press Q to close the camera



`tf_video.objectDetectionCap()`

2 Conclusions and some thoughts

In general, it seems that the Harr-feature cascade method does a pretty good job at detecting faces, especially when it comes to frontal faces. However, when given half faces and rotated faces, there might be some issues during detection. In comparison, CNN might not be as sensitive as Harr-feature cascade at detecting frontal faces but with the pre-trained model, it is a very powerful and amazing tool at detecting all kinds of objects.

Given a certain amount of data, perhaps millions of images in this case, we can expect CNN to produce models with incredibly amazing performance but still, the cost is the amount of data and the computation power we need. One important thing to note is that the framework of CNN within academia is still not quite complete and it is still like black box. Many of the reasons behind why certain things work are still given by "well, in practice this seems to work well". However, compared with CNN, the framework introduced by Viola and Jones (2001) is already considered to be a state-of-art method. It is well researched and understood. In practice, which one we should use really depends on our purposes and the recourses we have (amount of data, computational power, time, etc.). With the development of computer hardwares and softwares, I believe the day that CNN becomes a state-of-art method will not be too far.