EE 417 – Final Project

Project Title: Face Detection Based on FaceNet and Viola-Jones Algorithm

Group Members:

Aras Bozkurt - 19194

Deren Ege Turan - 20989

Supervisor: Mehmet Keskinöz

1. Introduction

1.1. Problem and Problem Definition

Face Detection is the initial and primary step of face recognition, face tracking and face analysis. It is used with the intention of detecting faces in the images. Usage of face detection has been growing gradually in time. Security, biometrics, law enforcement, personal safety, entertainment are some of the examples of usage areas of face detection [1].

There are two types of techniques that detects faces from given images, they are called as Feature Based and Image Based techniques. Feature based technique determines the features from the presented image and tries to link those features with already learned facial features, whereas the image based technique obtains the training and testing images, and then the algorithm tries to find the best match [2].

The main problems and challenges of face detection are increasing the detection rate and the accuracy of the face detection due to illuminations (Lighting effects), extremely high number of faces in an image, low resolution, face occlusion (In other words, hiding face by an object or collusion of faces with other objects), odd expressions, skin color, orientation of a face and distance etc. [2].

1.2. Problem Formulation and Solution Method

In this project we used and compared two methods for face detection as following:

- **FaceNet**

Google is the designer of FaceNet which is designed as a deep convolutional network that is trained to detect faces, verify, and recognize the face images and to solve the clustering problem which is efficiently scaled.

FaceNet learns a mapping from face images to a compact Euclidean space where distances are directly correspond face similarity measure [4].

- **MATLAB Built-in Functions based on Viola-Jones**

To use the built-in functions, the Computer Vision toolbox should be downloaded for MATLAB. vision.CascadeObjectDetector function uses Viola-Jones algorithm for detection of faces. Viola-Jones is one of the Feature-Based techniques that decreases computation time and procures high detection accuracy compared to other face detection algorithms.

The limitations of the Viola-Jones algorithm composed of the following topics:

- It takes extremely long training time.
- There are limited head/face poses for the detection algorithm.
- It is generally unable to detect the faces of dark colored people [2].

2. Implementation

2.1. FaceNet and MTCNN

Previously mentioned problems such as face recognition, verification and clustering can be implemented easily using standard techniques with FaceNet embeddings as feature vectors after this space is created. In contrast to traditional methods of using intermediate bottleneck layer, it contains a deep convolutional network trained for optimizing the embedding. Additionally, one of the most crucial optimizations and better applications of FaceNet is, it has an optimized embedding face recognition performance since it only uses 128-bytes per face. The convolutional neural network (CNN) is trained using Stochastic Gradient Descent (SGD) with standard backpropagation and AdaGrad [4].

In order to extract and align the face images, Multi-task Cascaded Convolutional Neural Networks (MTCNN) and Visual Geometry Group's Face (VGG Face) Dataset are used. MTCNN algorithm consists of 3 parts with detecting the bounding boxes of the faces in each part. In the first part, the image pyramid is constructed by scaling the image down multiple times. In the next parts, image parts are extracted to form each bounding box, then resized, and forwarded through the CNN. The VGGFace consists of models constructed for face recognition which can be downloaded from Tensorflow [5].

In this part of the project, experiments took place based on face detection. Face detection is used to detect human faces and one-to-one mapping of an input face and a known (person) face. Face detection could be further used for the, face identification which is used for determining the given person's identification or name, which is a one-to-many mapping for a given input face against a database of known faces in VGG Face.

2.2. Viola-Jones Algorithm

Viola-Jones algorithm consists of three main parts. The first part is related to Integral Image which is an image representation and it leads to quick evaluation of features. It does not directly concern with image intensities and this is one of the reasons why this algorithm is faster than most of the other algorithms.

The second part contains the usage of AdaBoost, which is used to create a basic, fast, and efficient classifier composed of less features from an enormous sized library of possible features. The small number of features, in other words feature selection, is achieved through the AdaBoost by forcing each weak classifier to depend on a single feature.

The third part contains an optimized method by merging more complex classifier models in ensembles. This optimized method yields a great improvement on the speed of the algorithm by focusing only on the high potential regions on the image where the potential means the location of possible face attributes [3].

vision.CascadeObjectDetector function: this function depends on the Viola-Jones algorithm for face detection. It detects faces, eyes, noses, mouths, or upper body.

vision.ShapeInserter function: plots multiple lines, rectangles, circles or polygons on 2-D images.

3. Results

Results are obtained by using FaceNet and Viola-Jones. The images where faces are containing the green squares are obtained by FaceNet (all figures that are named as: Figure x.1)

and faces that are containing the blue squares are obtained by Viola-Jones (all figures that are named as: Figure x.2). Results are discussed in the Discussion section below.

Figure-1

Figure 1.1

Figure 1.2

FaceNet algorithm detected all 4 faces while Viola-Jones detected 3 out-of 4 faces.

Figure 2

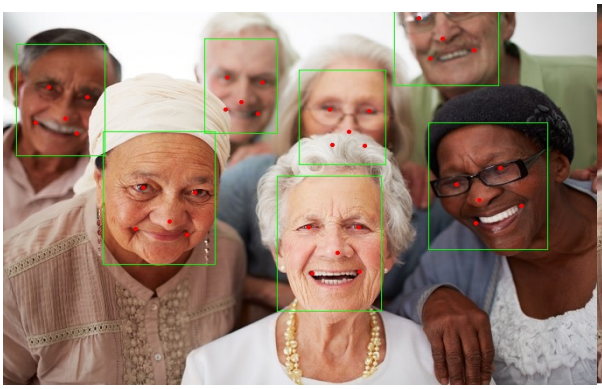

Figure 2.1

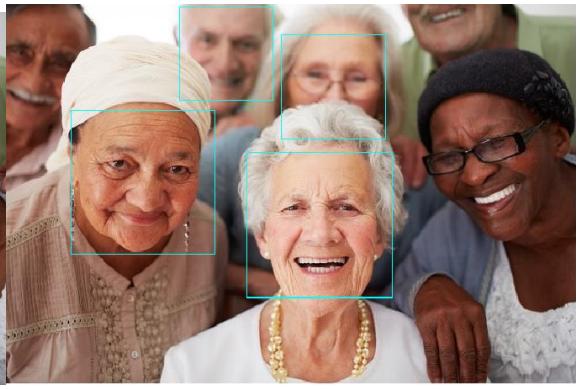

Figure 2.2

FaceNet algorithm detected all 7 faces while Viola-Jones detected 4 out-of 7 faces.

Figure 3

Figure 3.1

Figure 3.2

FaceNet algorithm detected 17 faces (1 outlier) out-of 18 faces while Viola-Jones detected 15 (2 outliers) out-of 18 faces.

4. Discussion

Previously, in the Introduction part, we mentioned that the main problems were increasing the accuracy and detection rate of the face detection. In the Results part, different kinds of images were used for clarifying the reason why those problems occur. For instance, in Figure-1, illumination is the main problem for not being able to detect the rightmost face for Viola-Jones algorithm (Figure 1.2) while FaceNet algorithm ran perfectly for that case. For Figure-2, low resolution of the people behind, occlusion of faces, and skin color were the main problems for not being able to detect faces. FaceNet could detect all of the faces without an error (Figure 2.1), on the other hand Viola-Jones algorithm could detect ⅔ of faces for low resolution, ⅔ of hiding faces and could not detect any dark colored faces (Figure 2.2). Finally, for Figure-3, main problems were, existing too many faces in the image, occlusion of faces, odd expressions, and orientations of faces. Generally, FaceNet gives better results than Viola-Jones due to its underlying highly trained and specialized convolutional neural network. FaceNet detected 16 out-of 18 faces and found 1 false positive result in Figure 3.1, Viola-Jones detected 13 faces and found 2 false positive results in Figure 3.2. False positive results are shown below respectively, in Figure 4.1, Figure 4.2, and Figure 4.3.

Figure 4.1

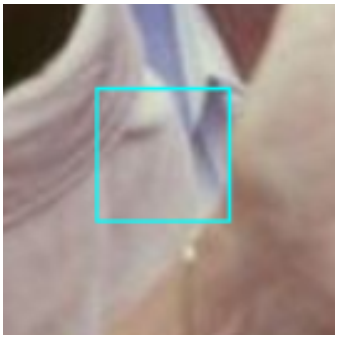

Figure 4.2

Figure 4.3

This outliers also show that, even with FaceNet, there is a minor chance of detecting incorrect faces in a given image when there exist noise and occlusions in the images. In general, using FaceNet algorithm produced better results than Viola-Jones algorithm.

5. References

[1] Singth, Shilpi, and S.V.A.V Prasad. "Techniques and Challenges of Face Recognition: A Critical Review." *Procedia Computer Science* 143 (2018), 536-543. https://www.sciencedirect.com/science/article/pii/S1877050918321252.

[2] Kumar, Ashu, Amandeep Kaur, and Munish Kumar. "Face detection techniques: a review." *Artificial Intelligence Review* 52, no. 2 (July 2018), 927-948. doi:10.1007/s10462-018-9650-2.

[3]Viola, Paul, and Michael J. Jones. "Robust Real-Time Face Detection." International Journal of Computer Vision 57, no. 2 (May 2004), 137–154. https://doi.org/10.1023/B:VISI.0000013087.49260.fb.

[4] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. doi:10.1109/cvpr.2015.7298682.

[5] Mühler, Vincent. "Realtime JavaScript Face Tracking and Face Recognition Using Face-api.js? MTCNN Face Detector." Medium. Last modified July 16, 2018. https://itnext.io/realtime-javascript-face-tracking-and-face-recognition-using-face-api-js-mtcnn-face-detector-d924dd8b5740?.

6. Appendix

Appendix A

MATLAB - Viola-Jones

```
detect_face = vision.CascadeObjectDetector;
insert_shape = vision.ShapeInserter('BorderColor','Custom','CustomBorderColor',[0 255 255]);

img = imread('people.jpg');
imshow(img);
shg;

bounding_box = step(detect_face, img);
img_faces = step(insert_shape, img, int32(bounding_box));
imshow(img_faces), title('Faces Detected');
```

Appendix B

Python - FaceNet

```
from __future__ import absolute_import

from __future__ import division

from __future__ import print_function

import sys

import os

import argparse

import tensorflow as tf

import numpy as np
```

```
import detect_face
import cv2

def main(args):
    sess = tf.Session()
    pnet, rnet, onet = detect_face.create_mtcnn(sess, None)

    minimum_size = 40
    THold = [ 0.6, 0.7, 0.9 ]
    Scale_factor = 0.709
    f_name =args.input
    out_fname =args.output
    dr = cv2.imread(f_name)
    image=cv2.cvtColor(dr,cv2.COLOR_BGR2RGB)

    bounding_boxes, points = detect_face.detect_face(image, minimum_size, pnet, rnet, onet,
THold, Scale_factor)
    nrof_faces = bounding_boxes.shape[0]
    for b in bounding_boxes:
        cv2.rectangle(dr, (int(b[0]), int(b[1])), (int(b[2]), int(b[3])), (0, 255, 0))
        print(b)

    for p in points.T:
```

```
        for i in range(5):
            cv2.circle(dr, (p[i], p[i + 5]), 1, (0, 0, 255), 2)


    cv2.imwrite(out_fname,dr)
    print('Total %d face(s) detected, saved in %s' % (nrof_faces,out_fname))
def parse_arguments(argv):
    parser = argparse.ArgumentParser()
    parser.add_argument('--input', type=str, help='image to be detected for
faces.',default='/content/mrr.jpg')
    parser.add_argument('--output', type=str, help='new image with boxed
faces',default='/content/sample_data/new.jpg')
    return parser.parse_args(argv)
if __name__ == '__main__':
    main(parse_arguments(sys.argv[1:]))
```

Note: Addition to the python code above, Detect_face.py , det1.npy, det2.npy, det3.npy ,which are in the .zip file, should be run together.