Computer Vision

Final Project: Transfer Learning for Face Detection and Recognition

Ahmed Aboukhadra

May 2020

1 Introduction

In this project, we aim to build a customized face recongition system by utilizing the abilities of pretrained models in object detection and face Embeddings. Face recognition is useful in security systems and commercial applications. To build such a system, three steps must be performed: Face Detection, extracting face Embeddings and training a model to recognize faces. We also investigate the usability of the system on surveillance images to find how powerful can those systems be in security applications. In addition, a custom dataset of personal images was created to evaluate the system.

2 Related work

Face detection has it's own dedicated architectures like RetinaFace[3] that was trained on large-scale face datasets such as WIDERFACE [9]. However, we think of face detection as a subtask of object detection. Detectron2 [8] is a state-of-the-art object detection algorithm from Facebook Research that is based on Mask R-CNN [4], is built in PyTorch and was pretrained on the COCO dataset [5]. Detectron2 can be finetuned to detect new types of objects, in this case, faces.

Facenet [7] is a state-of-the-art Face Recognition system from Google and is based on ConvNets that can extract feature vectors from faces. The Network was trained on triplets that have an anchor, a positive face and a negative face and the triplet loss was used to minimize the distance between the positive and the anchor while maximize the distance between the negative and the anchor. VGGFace [6] is another face recognition network that use the same technique in training.

3 Methodology

As previously introduced, face recognition consists of 3 steps. Detectron2 [8] was finetuned to detect faces, Facenet [7] was used to extract feature vectors for faces, and finally, the embeddings were used to train an SVM for classifying faces. To test the overall pipeline, we create a small custom dataset of 4 subjects with 10 images for training and 5 for testing for each subject.

3.1 Finetuning Detectron2 to detect faces

Detectron2 was finetuned to detect faces in images by following the tutorial found on Google Colab[1] to finetune detectron2. To do so, a Face Dataset from Kaggle by DataTurks[2] was used.

The dataset contains 409 images with 1132 faces with their bounding boxes. The dataset is in JSON format and was preprocessed to a format suitable for Detectron2 to use. First, every box in every image was saved as a row in a csv file. Afterwards, the images were represented with a dictionary that has the file name and a list of annotations where each annotation has the coordinates of the bounding box and a polygon that represents that box for segmentation purposes. A subset of 5% from the data was reserved for testing.

To train the model, both the configuration file and the pretrained model weights were loaded. The model was trained for 1500 epochs on 0.0001 learning rate. The operation took 45 minutes on a GPU and the evaluation results are shown in Section 4.1. The finetuned model can be downloaded from a google link to avoid training every time. For any given input, the bounding boxes are detected and saved to be consumed by the face recognizer. It's important to note that the confidence threshold is set at 0.85 to extract the face bounds.

3.2 Transfer Learning from Facenet

After detecting faces, the image is loaded alongside the detected bounding boxes that were saved. The image is cropped with the bounding boxes coordinates and all faces are resized to 160x160. Afterwards, the faces were normalized by removing the mean and the standard deviation. Finally, Facenet [7] extracts a feature vector of size 128 given the cropped normalized face image. The Embeddings are also normalized using L2 normalization. We followed the tutorial [3] about Facenet for implementation .

3.3 Face recognition using Embeddings and SVMs

The faces were detected and embedded as discussed in Sections 3.1 and 3.2. Afterwards, an SVM [2] is fit to the Embeddings given the identities of the

[1] `https://colab.research.google.com/drive/1Jk4-qX9zdYGsBrTnh2vF52CV9ucuqpjk`

[2] `https://www.kaggle.com/dataturks/face-detection-in-images`

[3] `https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/`

subjects in the images. A dataset of low-quality surveillance images [1] that contain cropped faces was used to evaluate the face recognition subsystem and the results are reported in Section 4.2.

4 Results

4.1 Face detection on DataTurks face dataset

To evaluate the Face Detector, Intersection Over Union (IoU) was used as a threshold to distinguish between true positive and false positive. Afterwards, The Average Precision (AP) is calculated for different thresholds. Table 1 shows the AP with different thresholds and mAP -averaging AP for multiple thresholds- when the model was evaluated on the testing set. AP isn't expected to be as high as the accuracy since exact matching between the actual bounding box and the predicted one can be very difficult. Figure 1 shows the performance of the finetuned model on an image from the testing set. It can be seen that the model can detect multiple bounding boxes for the same face with different certainty. However, given the low amount of training data, the results are very satisfying and can be used practically for face recognition.

AP	AP50	AP75
26.74	68.47	20.91

Table 1: The results of finetuning Detectron2 for face detection

When the custom dataset was used to evaluate the face detector, some faces were detected with low confidence like the ones shown in Figures 2 and 3. Given the fact that the acceptance threshold was 0.85, those faces were not retrieved by the model causing the detection to fail. An easy solve is to change the acceptance threshold to 0.8.

4.2 Face recognition on surveillance data

The original plan was to evaluate the system on Surveillance videos. However, the only suitable dataset found is FaceSurv[4]. The dataset required a license that was hard to get and, therefore, QMUL-SurvFace [1] that contains cropped faces was used instead. The dataset contains low-quality images of dimensions less than 30 pixels as shown in Figure 4. 20 identities were chosen where each of them contains no less than 200 samples. The total number of samples were 3947 for training and 987 for testing. All samples were processed as described in Section 3.2 and a grid search with cross validation on the Hyperparameters of the SVM was used to select the optimal parameters. The search lead us to choose an RBF kernel and C=100, where the testing accuracy reached 71%. Table **??** shows the metrics for a subset of the identities.

[4]`http://iab-rubric.org/resources/FaceSurv.html`

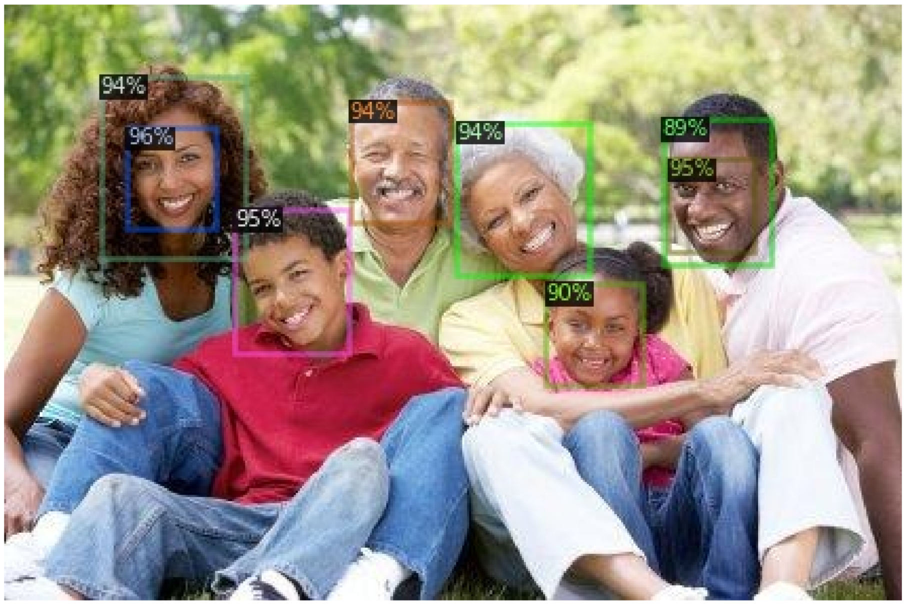

Figure 1: Face detection with Detectron2

Figure 2: A face that was detected with 0.82 confidence

4.3 Face detection and recognition on custom dataset

To test the system, a custom dataset of personal faces is prepared [5]. All faces in the images were detected, embedded and recognized using an SVM. The testing accuracy was 95% with only 1 mislabeled image. In Figure 5, one can see three faces that were detected and recognized correctly. In addition, attached to this report a personal video that contains a single face that was recognized. It can be seen in the video that when the face is partially shown, the recognition quality is reduced. this may be due to the low number of training samples. The average

[5]`https://drive.google.com/drive/folders/1O34bNSHf0cwVOQIqrbXQMMl4TI7Es9tu?usp=sharing`

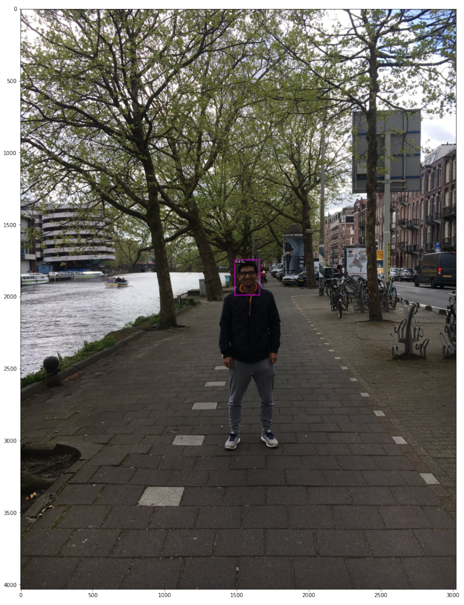

Figure 3: A face that was detected with 0.84 confidence

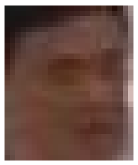

Figure 4: A low-quality image from the QMUL-SurvFace

time to detect and recognize 1 frame was 0.375 seconds while the recognition time was 0.095 seconds.

5 Discussion

Even though the number of samples used to finetune Detectron2 was very small, the model was able to detect faces with high performance, and to improve it, data augmentation can be used. In addition, Facenet showed powerful performance when it was tested against the custom dataset with small number of images while it wasn't shown to be that powerful when tested with SurvFace on high number of images. This could be due to using SVMs instead of ANNs or it could be due to the very low quality of surveillance faces. The latency of detecting and recognizing faces can be up to 0.5 second which isn't feasible in a real-time application.

Face ID	Precision	Recall	F1-score
2	0.90	0.88	0.89
3	0.89	0.94	0.92
7	0.71	0.60	0.65
12	0.46	0.52	0.48

Table 2: Face recognition performance on a subset of the 20 identities. Variance in recognition performance can be found depending on the class.

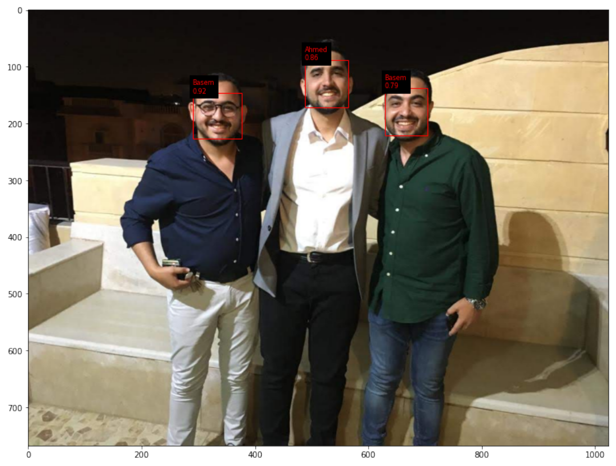

Figure 5: An image that contains 3 faces that were correctly detected and recognized

6 Conclusion

In this project, we investigated face detection and recognition systems and learned how to finetune object detection models and do Transfer Learning for Face recognition. Combining Detectron2 and Facenet to this task is the first to our knowledge. The results show the power of pretrained models, however, investigating more models like RetinaFace[3] and VGGFace[6] is an essential next step to compare the performance and the efficiency. Acquiring Surveillance videos and using them to test this system can also be a possible improvement.

References

[1] Zhiyi Cheng, Xiatian Zhu, and Shaogang Gong. Surveillance face recognition challenge. *arXiv preprint arXiv:1804.09691*, 2018.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[3] Jiankang Deng, Jia Guo, Zhou Yuxiang, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. In *arxiv*, 2019.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[6] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[7] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[8] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[9] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *CVPR*, 2016.