Video face recognition based on RetinaFace and FaceNet

Qimeng Tao
Department of Electrical Engineering,
Columbia University
New York, USA
qt2139@columbia.edu

***Abstract*—In this project, I combined a face detection algorithm (RetinaFace) with a face recognition algorithm (FaceNet). And use the new model for face recognition in video. The project Github are available at https://github.com/qt2139/COMS4995DLCV.**

***Index Terms*—RetinaFace, FaceNet, MobileNet**

I. Introduction

A. First Creative Idea - Combine two models

As the professor stated in lecture 2, face recognition is important to us but difficult for computers. Especially in the video field. For example, in lecture 2, we can see that the input to the model is an image, and this image contains only one face (the area of the face exceeds 90% of the image area). Let's look at the widely used face dataset CASIA-WebFace, As shown in the figure 1, most of the images contain only human faces.
But when we take a photo or record a video, usually the face only occupies a small area, and most of the area is the background. At this time, when we directly send each frame in the video into the existing face recognition model, I believe the model will not have very good performance.

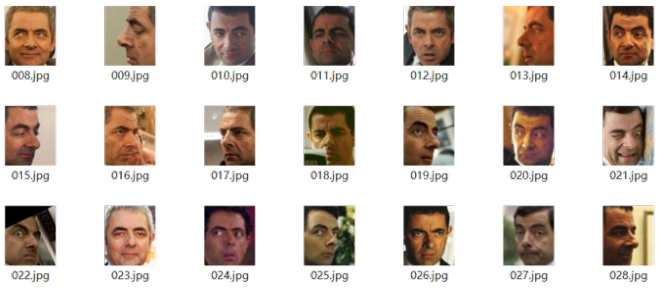

Fig. 1. Face Recognition Process.

In order to solve this situation, I first use the face recognition algorithm to identify the face of each frame in the video and crop it, and then send the cropped image to the face recognition algorithm, and finally get the recognition result. The algorithm flow is shown in Fig 2.

Fig. 2. Face Recognition Process.

B. Second Creative Idea - Face Alignment

When we crop the images, we will find that sometimes the head of the person is inclined. In order to reduce the bias, I use the coordinates of the eyes to calculate the angle between the eyes and the horizontal line. And rotate the image for a degrees. Then send the rotated image to the face recognition network. As shown in the figure 3. We will discuss it more closely in the next section.

Fig. 3. Face Alignment

C. Third Creative Idea - Merge Intervals

Another creative idea is, when we do the inference, there is also a return result that specifies the time interval in which the person appears in the video. For example, a person appears at frames [[0, 60], [80, 120]].

In fact, we can return [0, 1, 2, 3,...58, 59, 60, 80, 81,...,119, 120]. Obviously, this is not intuitive. And when the model is misdetected, there will be some breakpoints, such as [0, 2, 4, 8, 10]. But in real situations, at the 1st, 3rd, 5th, 7th, and 9th frames, the characters appear in in the video. At this point, I set up a little trick. When the model does not detect a person, I will record the current index first, and then continue to detect. If the person is detected within 10 consecutive frames, I ignore the false detection at that time. In contrast, we did not detect the person in the i-th frame, and did not detect the person in the last 5 consecutive frames, then a reasonable interval is [,i], just like we merge intervals and return [[0, 60], [80, 120]].

II. Related Work

Face detection is more like a subproblem of object detection. Both of them have good performance in detection algorithms.

One stage methods have slightly lower map than two stages methods, but one stage methods are faster in inference than two stages methods. The MAP of one stage method is slightly lower than that of two stages method, but the inference speed of one stage method is faster.

In the two stage approach, just like Mask r-cnn [6], the first step is to generate regional proposal from the image and finally output a number of ROI, then use ROI pooling and a number of fully connected layers to generate the final object borders. The one-stage method like YOLO [7] does not need regional proposal, and directly generates the class probability and position coordinate values of the object. The final detection result can be obtained directly after a single detection, so it has a faster detection speed.

In face recognition, most methods start by extracting features from the image through a convolutional network. But the methods differ slightly when comparing, for example, Zhenyao et al. [8] using PCA and SVM classifiers. Deepface [9] uses L1-distance to calculate the distance of two vectors.

III. Introduction to Datasets

In this project, I will use two datasets, WIDER FACE and CASIA-WebFace, use WIDER FACE to train RetinaFace, and use CASIA-WebFace to train FaceNet.

A. WIDER FACE

The WIDER FACE dataset is a face detection dataset produced by the Chinese University of Hong Kong, containing 32,203 images and 393,703 labeled faces. Among them, there are 158,989 labeled faces in the training set and 39,496 face images in the validation set. Each subset contains 3 levels of detection difficulty: Easy, Medium, Hard. Since the dataset brings together various influencing factors such as face size changes, face pose changes caused by camera angles, different degrees of face occlusion, expression changes, differences in light intensity and makeup, this dataset is widely used in global face detection. The field is extremely challenging. We can download the dataset directly from the official website.

B. CASIA-WebFace

The CASIA-WebFace dataset is used for face recognition tasks. The dataset contains 494,414 face images of 10,575 real identities collected from the web. I uploaded the dataset to Google Drive.

IV. Face Detection - RetinaFace

First, I will use a face detection algorithm to detect the faces in the picture and crop the faces according to the coordinates. Regarding the face detection algorithm, in this project, I will use RetinaFace [1]. I used RetinaFace because RetinaFace is a one-stage detection algorithm, and its inference speed will be very fast. Second, I found that Easy, Medium, and Hard in the dataset correspond to the proportion of face area and image area, respectively. When the face area is larger, its difficulty is easier. Therefore, we should focus more on the Easy part. Because for the Hard part, the face area is relatively small. Even humans are difficult to identify. In the part of the Easy, Medium dataset, the model has good performance, it can achieve over 96% accuracy in the WIDERFACE validation subset.

A. Introduction to Backbone

In the paper, the author gave two backbones, the first is ResNet-152 [2] and the second is MobileNet-0.25 [3]. The model can achieve higher accuracy when we use ResNet-152 as the backbone. But as we all know, inference speed is also a very important metrics. Using MobilenetV1-0.25 as the backbone will indeed reduce some accuracy, but the inference speed will be greatly improved. And the performance of the network is also good enough when using MobileNet-0.25. So I choose MobileNet-0.25 as the backbone in the project.

B. MobilenetV1-0.25

MobileNet is a network proposed by the Google team in 2016, which uses Depthwise Convolution to build lightweight deep neural networks. In the network, each channel in the feature map is only convolved with each channel of the Depthwise convolution. This can greatly reduce the parameters, but only using Depthwise convolution can not change the number of channels. To this end, we are using a Pointwise Convolution, and the size of the convolution kernel of Pointwise Convolution is 1*1*n. After Pointwise Convolution, we can change the number of channels to n. Table I is the network structure of MobileNet [3].

TABLE I
MobilenetV1 Architecture

Type / Stride	FilterShape	InputSize
Conv / s2	3×3×3×32	224×224×3
Conv dw / s1	3×3×32 dw	112×112×32
Conv / s1	1×1×32×64	112×112×32
Conv dw / s2	3×3×64 dw	112×112×64
Conv / s1	1×1×64×128	56×56×64
Conv dw/s1	3×3×128 dw	56×56×128
Conv / s1	1×1×128×128	56×56×128
Conv dw / s2	3×3×128 dw	56×56×128
Conv / s1	1×1×128×256	28×28×128
Conv dw / s1	3×3×256 dw	28×28×256
Conv / s1	1×1×256×256	28×28×256
Conv dw / s2	3×3×256 dw	28×28×256
Conv / s1	1×1×256×512	14×14×256
5 * (Conv dw / s1, Conv / s1)	3×3×512 dw 1×1×512×512	14×14×512 14×14×512
Conv dw / s2	3×3×512 dw	14×14×512
Conv / s1	1×1×512×1024	7×7×512
Conv dw / s2	3×3×1024 dw	7×7×1024
Conv / s1	1×1×1024×1024	7×7×1024
AvgPool / s1	Pool 7×7	7×7×1024
FC / s1	1024×1000	1×1×1024
Softmax / s1	Classifier	1×1×1000

Table I shows the network architecture of mobilenetV1-1. In the project I used mobilenetV1-0.25, which differs from mobilenetV1-1 in that the number of all channels is compressed to 1/4 of the original number. The authors evaluated the model on ImageNet and the experimental results are shown in Table II below. As can be seen from Table II,

the performance of the model is not good compared to more complex networks, but there is a great improvement in the inference speed. The use of mobilenetV1-0.25 is sufficient, especially considering that we are more concerned with larger face areas in our project.

TABLE II
ACCURACY ON IMAGENET

Model	Accuracy [3]
1.0 MobileNet-224	70.6%
0.75 MobileNet-224	68.4%
0.5 MobileNet-224	63.7%
0.25 MobileNet-224	50.6%

C. Feature Pyramid Network

For example, target detection algorithms such as YOLO, RetinaFace also builds a feature pyramid. As shown in the figure 4, after C3, C4, and C5, the number of channels is adjusted by 1*1 convolution, and then fusion the feature by Upsample and Add. Finally, we obtained three effective feature layers P3, P4, and P5.

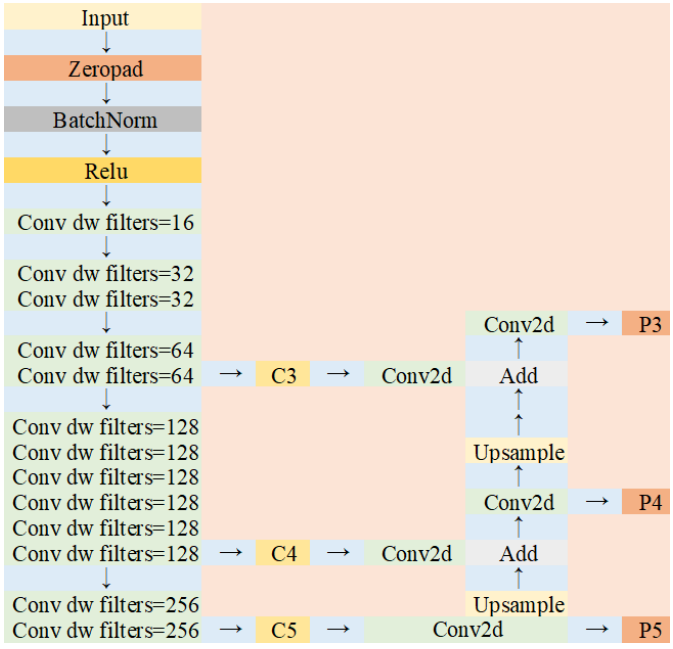

Fig. 4. Feature Pyramid Network.

D. SSH enhances feature extraction

In the previous step, we obtained three effective feature layers P3, P4, and P5. In order to further enhance feature extraction, Retinaface uses the SSH module to enhance the receptive field. SSH is structured as follows 5:

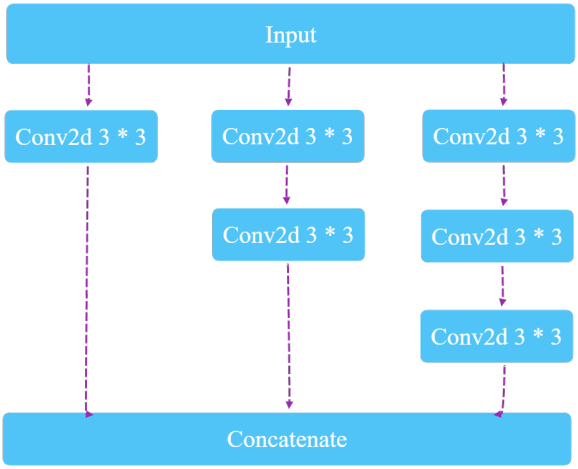

Fig. 5. SSH.

The idea of SSH is to simulate convolution. It uses three parallel structures and uses two 3x3 convolutions to simulate a 5*5 convolution. Use three 3*3 convolutions to simulate a 7*7 convolution.

E. Model's predictions

As we can see from the paper, the output of the model is a 16*num anchors matrix. Among them, num anchors represents the number of anchors in each grid. In the paper, the author uses 2 squares as anchors.

First of all, as can be seen from the figure 6, the first two dimensions of the output are the classification results, which represent whether the image contains a face.

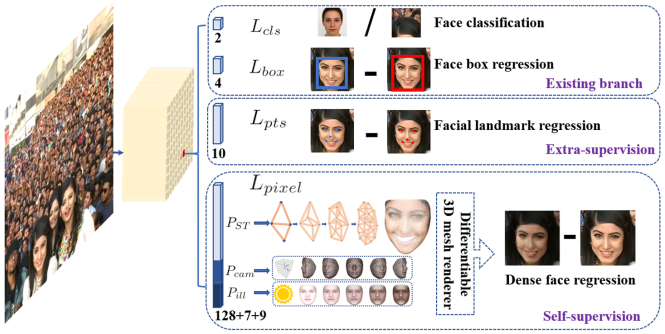

Fig. 6. Output of RetinaFace (image source: RetinaFace [2]).

The next 4 dimensions represent the center point coordinates(first two dimensions) and width and height(last two dimensions) of the face prediction box.

The last 10 (5*2) dimensions represent the key point parameters of the face, representing the adjustment parameters of the eyes, nose and mouth respectively.

When the model completes the prediction, there will be many boxes on the original image. This is because a face will

occupy multiple grids, each of which will output a result. In this case, we need to use non-maximum suppression. Choose the maximum value of the IOU for the prediction box. Finally, we use OpenCV to draw face prediction boxes on the original image.

F. Experiment Results

The authors have tested the model on the WIDER FACE validation subset dataset. The experiments are shown in the table III below.

TABLE III
EXPERIMENT RESULTS ON WIDER FACE VALIDATION SUBSET

Method	Easy	Medium	Hard	mAP
FPN+Context	95.532	95.134	90.714	50.842
+DCN	96.349	95.833	91.286	51.522
$+L_{pts}$	96.467	96.075	91.694	52.297
$+L_{pixel}$	96.413	95.864	91.276	51.492
$+L_{pts}$ $+L_{pixel}$	96.942	96.175	91.857	52.318

The figure 7 is a sample output from Retinaface. The upper left corner of the blue face box is the probability of whether the box contains a face. As can be seen from the figure, the probability of being a face is 99.98%. The model performs well.

Fig. 7. Example Result of Retinaface.

V. FACE RECOGNITION - FACENET

A. Introduce of few shot learning

In HW1 I learned few-shot learning. Few-shot learning is designed to deal with insufficient samples. For example, it is impossible for us to upload hundreds of photos of the same person to the model. The idea of few-shot learning is to pre-train on a large-scale dataset to obtain a feature representation (a feature vector). Then use 1-5 training samples to train the model. When predicting, we feed the predicted samples into the same network and also get a column vector. At this point we check the distance between the two vectors. If the distance is less than the threshold, we consider the recognition successful.

B. Introduce of FaceNet

FaceNet [5] is a face recognition algorithm proposed by Google in 2015. FaceNet utilizes the invariance of the same face at different angles, different lighting, and different poses to complete the model design. After it was released, it was SOTA at that time. It can achieve 99.63% accuracy on LFW dataset. [5]
As described in the previous subsection, the model encodes an image as a 128-dimensional column vector. In order to facilitate the comparison later, we will also use L2 normalization, so that the feature vectors of different faces can belong to the same order of magnitude.

$$||x||_2 = \sqrt{\sum_{i=1}^{N} x_i^2} \tag{1}$$

C. Face capture and alignment

First of all, as mentioned in the previous chapter, we first use RetinaFace to get the position coordinates of the face prediction frame and the human eye. We observed that the faces are not always vertical in the photos, and sometimes the faces are inclined. At this point we need to align it. We obtain the angle between the line connecting the two eyes and the horizontal plane, and rotate the angle of the image to display the face vertically. Our approach is to get:

- The angle of inclination of the line connecting the two eyes relative to the horizontal.
- The center of the image.

We rotate the center of the image to get the aligned face, as shown in the figure 8. Then we send this new rotated image to the Face recognition model.

Fig. 8. Face Alignment

D. FaceNet Architecture

The FaceNet architecture is shown in Figure 9. The author uses InceptionNet as the backbone in the paper. Considering the inference speed, I still use MobileNetV1-0.25 as the backbone of the network.

Fig. 9. FaceNet Architecture (image source: Facenet [5]).

E. Face comparison

As with few-shot learning, instead of judging the class of an object, we check whether two objects are the same object. Analogy to this idea, in FaceNet, when we know a face picture, we need to check whether the second picture and the first picture are the same person, instead of caring who this person is.

- Send all known faces into the model for encoding, and save these vectors the database.
- Send all unknown faces (people to be recognized) into the model for encoding.
- The vectors of unknown faces (people to be recognized) are compared with the vectors in the database.
- Get the distance of each pair of vectors by circular comparison and find the minimum distance.
- If the minimum value is less than the threshold, That is, we find the correct face.

The flow chart is shown in Figure 10.

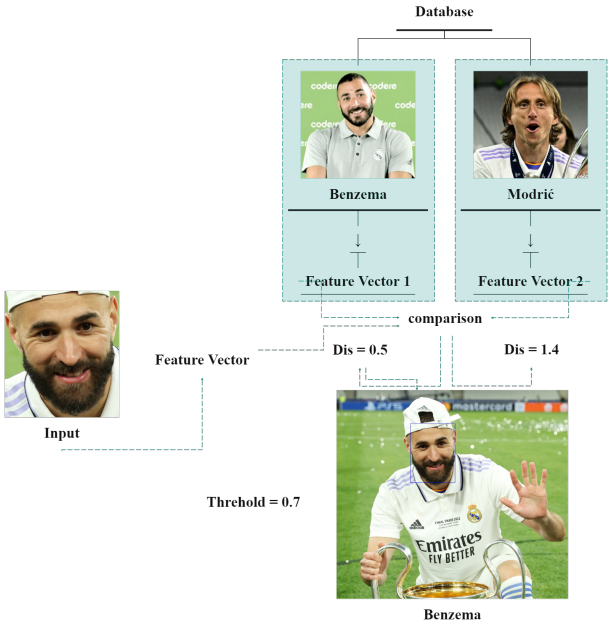

Fig. 10. FaceNet Flow Chart.

F. Experiments

The authors did model evaluation on the LFW (Labeled Faces in the Wild) and YouTube Faces DB datasets. [5], and I evaluate on the CASIA WebFace. The Table IV shows the experimental results. From the results, FaceNet has good performance. The model has achieved 99% accuracy on the large-scale face dataset Labeled Faces in the Wild. The model also has 95% accuracy on YouTube Faces DB.

TABLE IV
ACCURACY OF FACENET

Dataset	Accuracy
Labeled Faces in the Wild	99.63%
YouTube Faces DB	95.12%
CASIA WebFace	98.23%

My predictions are as follows. The input to the model is an image. First we get the face prediction box through RetinaFace, then send the prediction box to FaceNet. In FaceNet, we first use the idea of few shot learning to do pre-training in the large-scale face data set CASIA WebFace to learn the expression of features, and then generate a 128-dimensional column vector for the predicted image and then combine with Column vectors in the database are compared. I set the threshold to 0.7. When the distance between the two smallest vectors is less than 0.7, the model is considered to have recognized the correct face. The example image experiment result is shown in Figure 11.

Fig. 11. Example Result of FaceNet.

G. Video Face Recognition

My ultimate goal is to use this algorithm in a video. Now we have 300 frames and the person appear all the time. The algorithm will returns a time interval [0, 300]. But sometime the model failed. If the model failed in 121, 122, 123, 180 frames. At this time, the model will return [[0, 120], [124, 179], [181, 300]], which is obviously not a reasonable answer. My approach is that when the model fails at frame i, I remember i. And continue to detect 5 consecutive frames. As long as the model recognizes the person in 5 consecutive frames, even if it fails at the i-th frame, I think the model recognizes it. At this time, we will forget the index i and continue to recognize backward from the current frame. Just like the pseudocode, if interval[i][1] + 5 ¿ interval[i + 1][0], we do the merged() operation. This way the model will return a more reasonable interval [0, 300]. The algorithm flow is shown in Fig 12.

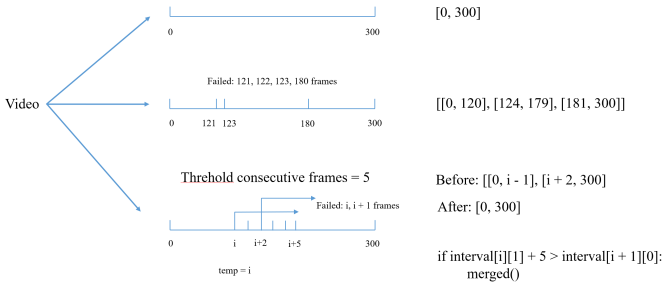

Fig. 12. Video Face Recognition Process.

VI. Conclusion

A. Discussion

The video experiment results are available at Google Drive. In the experiments I did with the video, I found that the model performed perfectly when the person was facing the camera. Thanks to the powerful performance of RetinaFace, even the two steps approach has great performance. Also interval merging brings a performance improvement to the algorithm.

B. Limitations and Future Work

There are two potential improvements of this project to explore in the future. For example, when the person does not look directly at the camera or sideways, or the person turns his head. In these cases, it is difficult for the model to recognize accurately.

I also tried some corner cases, such as adding only one picture to the dictionary to detect a video of a person from birth to middle age. I observed that when adding a picture of David Beckham when he was 37 years old to the dictionary, the model had difficulty detecting pictures of him as a baby or as a teenager. In another test, the video was an excerpt from a different movie of an actor. This is because the actor wore different makeup in different movies. At this point, it was also difficult for the model to recognize correctly. I guess these two unsuccessful examples are because their feature vectors are too far apart in space. In fact, even with the naked eye, it is difficult for me as a human being to recognize correctly.

Finally, the model is not end-to-end, but two-step. It is well known that end-to-end models perform better in deep learning. But this does not affect this project, because the accuracy of the first model RetinaFace is particularly high, exceeding 95%. So even the model has two steps, the model has a good performance.

VII. Acknowledgment

This project is a course project of COMS W 4995 006 Deep Learning for Computer Vision. Here, I would like to thank Professor Peter Belhumeur for his wonderful lecture and all TAs for their dedication to this course. Through the study of this course, I learned the basics such as the architecture of neural networks, and learned about tasks such as image classification and object detection. I believe this course will be of great help in my later career.

References

[1] Deng, Jiankang, et al. "Retinaface: Single-shot multi-level face localisation in the wild." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[2] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[3] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[4] C. C. Loy, D. Lin, W. Ouyang, Y. Xiong, S. Yang, Q. Huang, D. Zhou, W. Xia, Q. Li, P. Luo, et al. Wider face and pedestrian challenge 2018: Methods and results. arXiv:1902.06854, 2019.

[5] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[6] He, K., Gkioxari, G., Dollár, P., Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[7] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[8] Z. Zhu, P. Luo, X. Wang, and X. Tang. Recover canonicalview faces in the wild with deep neural networks. CoRR, abs/1404.3543, 2014. 2

[9] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In IEEE Conf. on CVPR, 2014. 1, 2, 5, 7, 8, 9