EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

A Lightweight Face Recognition Model based on MobileFaceNet for Limited Computation Environment

Jianyu Xiao[1], Guoli Jiang[1], Huanhua Liu[2,*]

[1]School of Computer Science and Engineering, Central South University, Changsha 410075, China
[2]School of Information Technology and Management, Hunan University of Finance and Economics, Changsha 410205, China

Abstract

The face recognition method based on deep convolutional neural network is difficult to deploy in the embedding devices. In this work, we optimize the MobileFaceNet face recognition network MobileFaceNet so as to deploy it in embedding environment. Firstly, we reduce the model parameters by reducing the number of layers in MobileFaceNet. Then, the h-ReLU6 activation function is used to replace PReLU in the original model. Finally, the effective channel attention module efficient channel attention is introduced to obtain the importance of each feature channel by learning. After the optimization, the MobileFaceNet parameters are compressed to 3.4 MB, which is smaller than the original model (4.9 MB), and the mAPs reach 98.52%, 97.54% and 91.33% on the test sets of LFW, VGGFace2 and the self-built database, respectively, and the recognition time is about 85 ms/photo. It shows that the proposed method achieves a good balance between the model complexity and model performance.

Keywords: Face recognition, MobileFaceNet, weak computing environment, channel attention mechanism.

Received on 02 February 2022, accepted on 26 February 2022, published on 28 February 2022.

doi: 10.4108/eai.28-2-2022.173547

*Corresponding author. Email: liuhuanhua@hufe.edu.cn

1. Introduction

Face recognition, as a biometric identification technology, has been used in a wide range of applications, such as security systems and identity verification systems. The traditional face recognition methods are based on Principal Component Analysis (PCA)[2]. Turk M *et al.*[1] proposed Eigenface which extracts main features and reduces the data dimension for face recognition. Fisher face proposed by BELHUMEUR P N *et al.*[3] calculates the minimum dispersion by Linear Discriminant Analysis (LDA) to distinguish faces. However, these algorithms are not robust to the change of illumination and face pose. Between 2009 and 2012, face recognition algorithm based on sparse representation[4] became a hot research topic because of its better robustness to the occlusion problem. Researchers also proposed algorithms such as face super-resolution, face illumination normalization, and face pose correction to solve the effects of illumination and pose change.

With the rapid development of deep learning technology, it has also made an important breakthrough in the field of face recognition[5]. This approach trains the model in a large dataset in advance, which enables it to extract more generalized face features[6]. The Alexnet network proposed by Krizhevsky *et al.* started a wave of research in the field of deep convolutional neural networks for face recognition[7]. Facebook proposed the DeepFace algorithm which used a face detection method based on detection point to extract 4096-dimension features through a 9-layer CNN network and achieved an accuracy of 97.35% on Labeled Faces in the Wild (LFW) database. Florian Schroff[8] *et al.* proposed the FaceNet algorithm to apply triplet loss to the CNN network structure and achieved 99.63% accuracy on LFW. However, these networks have huge number of parameters that means with low computational efficiency, which cannot be deployed on mobile and embedded devices[9], so a large number of scholars started to research on lightweight neural networks. SqueezeNet was the first model proposed by Berkeley *et al.*[10], and Google proposed the Xception[11] model, which implements a lightweight neural network by compressing the neural network parameters. Google proposed a lightweight network based on deep separable convolution MobileNet[12] model, which introduces two hyper

parameters, not only reduces the network weight parameters, but also improves the computational speed. The MobileNetV2[13] improved by adopting the strategy of expansion and compression, which not only improves the accuracy on but also reduces the number of parameters of the model. MobileNetV2 reduces the storage size and improves the computation speed, which is suitable for mobile devices.

MobileFaceNet[14] model is a lightweight face recognition network based on MobileNetV2 with only 4M and high accuracy, which is tailored for mobile and embedded devices and is ideal for implementing face recognition in weak computing environments. The model replaced the average pooling layer with a separable convolution, used the insight face loss function for training, and introduced batch normalization[15]. Zhang [16] *et al.* introduced the style attention mechanism in the MobileFaceNet network[17] to enhanced the feature representation and use AdaCos[18] face loss function to train the model to improve its accuracy and robustness. Hang[19] *et al.* introduced the SE module in MobileFaceNet[20] and successively used softmax loss and insightface loss[21]. The two loss functions were used to train the model to improve the recognition accuracy in mobile. Bihao [22] *et al.* optimized the network structure of MobileFaceNet and proposed a new loss function, Focal-angle Loss, to improve its recognition rate.

Although MobileFaceNet is already a lightweight neural network for embedded devices, however, there are still many shortcomings when the model is ported to embedded devices due to its hardware condition limitations. The higher resolution input image results a deeper network structure to gradually extract face features, and the computational effort increases. Therefore, in this paper, we lightened the network structure of MobileFaceNet, and the accuracy of the model is ensured by replacing the activation function and introducing the effective channel attention module ECA[23].

2. MobileFaceNet Network Structure

As shown in Table 1, the MobileFaceNet network takes a 112 × 112 resolution image as input and extracts the face features from it, while the output features are 512-dimension. The model contains a total of 20 layers of network, and the input feature image is received at the beginning stage using a fast down sample strategy. The next five bottleneck layers are repeated 1, 5, 1, 6, and 1 times to extract the shallow to deep features of the face, and the last few convolutional layers are down sampled and a 1×1 linear convolutional layer is added after the linear global depth wise convolutional layer as the feature output.

In MobileFaceNet convolutional neural network, the average pooling layer is replaced with global depth convolution (GDConv), and the output of global depth convolution is given by the following equation (1), where K is the depth-separable convolution kernel, F is the dimension of the input feature map, i, j are the spatial width and height dimension, and r is the current channel.

$$G_r = \sum_{i,j} K_{i,j,r} \times F_{i,j,r} \quad (1)$$

When both the input feature map size and the convolution kernel size are $W \times H \times R$, and the output feature map G has a size of $1 \times 1 \times R$, the ratio of the computational overhead of the global depth convolution layer to the computational overhead of the global average pooling layer is given by the following equation (2), where W, H, R indicate width, height and number of channels respectively, and Q is the number of filters.

$$\frac{W \times H \times R}{W \times H \times R \times Q} = \frac{1}{Q} \quad (2)$$

The specific network structure of MobileFaceNet is shown in Table 1, where t denotes the "expansion" multiplier, *i.e.*, the channel expansion multiplier of the reverse residual network, c denotes the number of output channels, n denotes the number of repetitions, and s denotes the step stride.

Table 1. MobileFaceNet network structure

Input	Operator	t	c	n	s	ECA
$112^2 \times 3$	conv3×3	-	64	1	2	-
$56^2 \times 64$	depthwise conv3×3	-	64	1	1	-
$56^2 \times 64$	bottleneck	2	64	5	2	√
$28^2 \times 64$	bottleneck	4	128	1	2	-
$14^2 \times 128$	bottleneck	2	128	6	1	-
$14^2 \times 128$	bottleneck	4	128	1	2	-
$7^2 \times 128$	bottleneck	2	128	2	1	√
$7^2 \times 128$	conv1×1	-	512	1	1	-
$7^2 \times 512$	Linear GDConv7×7	-	512	1	1	-
$1^2 \times 512$	linear conv1×1	-	128	1	1	-

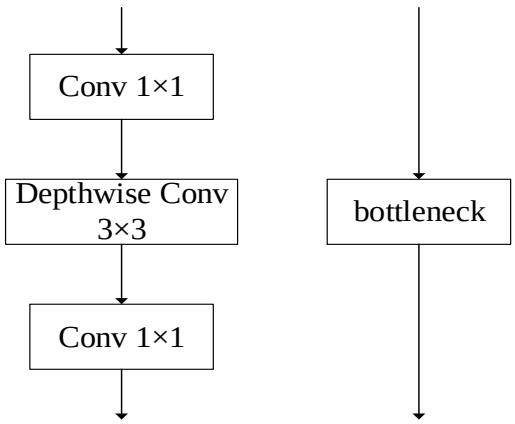
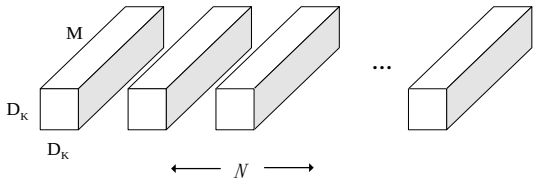


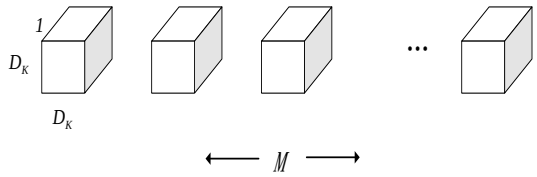
Figure 1. Bottleneck layer structure in MobileFaceNet

MobileFaceNet continues the bottlenecks in MobileNetV2[24] and uses it as the main module to build the network, as shown in Figure 1 for the structure of the bottleneck layer in MobileFaceNet. The expansion factor in bottlenecks is also reduced, and PReLU is chosen as the activation function and insight face loss as the loss function. In addition, MobileFaceNet network also introduces 7×7 separable convolution before the fully connected layer to replace the original average pooling layer, so that the features extracted by the network are more generalized.

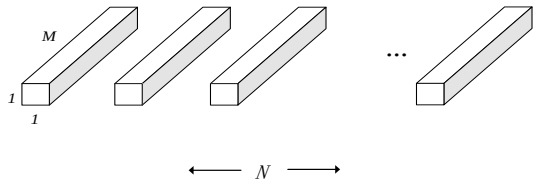
Deep Separable Convolution[25] is an important part of the MobileFaceNet network because mapping the channels and spaces of the convolutional layers separately gives better results. As shown in Figure 2, the depth-separable convolution is based on this decomposition of the traditional convolution into a depth convolution plus a 1×1 convolution, and then followed a 1×1 point-by-point convolution.



(a) Standard convolution filters



(b) Depth-wise convolution filters



(c) 1×1 convolution filters of depth-wise separable convolution

Figure 2. Deeply separable convolution

Under standard convolution, the input $D_f \times D_f \times M$ feature map F, the obtained output is $D_g \times D_g \times N$ feature map G. The parameters of the convolution kernel used are $D_k \times D_k \times M \times N$, where D_k is the convolution kernel size, D_f is the feature map size, M is the number of input channels, and N is the number of output channels, then the calculation formula is shown below.

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m} \quad (3)$$

Then the calculated amount is shown in the following equation.

$$S_1 = D_k * D_k * M * N * D_f * D_f \quad (4)$$

The depth separable convolution is calculated as shown in Figure 3. Firstly, the depth convolution is multiplied by bit according to the channels, and one convolution kernel is responsible for one channel, and the number of channels does not change at this time. Since one convolution kernel can only obtain part of the information of the feature map, it is necessary to use 1×1 convolution kernels to perform the traditional convolution operation to combine all the feature maps to obtain a new feature map with all the information. At this time, the number of channels can be changed.

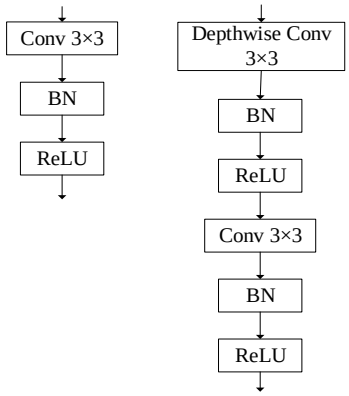


Figure 3. Conventional convolution (left) and depth-separable convolution (right)

The computational effort is given in the following equation (5), which will decrease by $1/N + 1/ D_k^2$ compared to traditional convolution.

$$S_2 = D_k * D_k * M * D_f * D_f + 1 \times 1 * M * N * D_f * D_f \quad (5)$$

Compared with MobileNet, MobileFaceNet is more lightweight, with only 0.99 million parameters. The model with smaller size and higher accuracy is more suitable in computation limited environment. Table 2 shows the accuracy and number of parameters comparison between MobileFaceNet and MobileNet.

Table 2. Comparison of MobileFaceNet and MobileNet

Network	LFW	AgeDB-30	Number of parameters
MobileNetV1	98.63%	88.95%	3.2M
MobileNetV2	98.58%	88.81%	2.1M
MobileFaceNet	99.28%	93.05%	0.99M

In this paper, the neural network in the MobileFaceNet algorithm is appropriately optimized to make it better applicable to the embedded platform with limited computation capacity. The computation capacity of the embedded platform is limited, since this paper adopts the way of transmitting feature values to achieve the overall process of face recognition, the number of feature points in the recognition process and the size of the output feature values can be appropriately reduced. However, this will inevitably lead to a lower recognition accuracy and cause bad results. In this paper, we use replacement of the activation function and the introduction of the Efficient Channel Attention (ECA) module to optimize the neural network to ensure that the model has a high recognition accuracy. In this paper, the network model is optimized in the following aspects.

1) We adjust the input of the model to 96×96 resolution images, then appropriately reduce the number of network layers by reducing the number of repetitions of the first and third bottleneck layers to reduce the number of network parameters and lowering the computational effort.

2) We replace the activation function of the bottleneck layer from PReLU to h-ReLU6 to improve the network performance.

3) We used ECA module in the first and last bottleneck layers to optimize the parameters automatically.

3. MobileFaceNet Optimization

3.1. Network Structure

Considering that this paper targets face recognition in a weak computing environment and does not need to process complex images, the network input is adjusted to a 96×96 resolution image. Since the biggest advantage of small embedded devices is their small size and portability, the face is close to the camera when acquiring face images. Figure 4(a), (b), and (c) shows the image with the detected face image, processed image with resolution 112 × 112, and processed image with resolution 96 × 96, respectively. It can be found that the image with resolution 96 × 96 is sufficient to contain most of the information of a human face.

(a) (b) (c)

Figure 4. Face image processing. (a) Detected face (b) 112×112, (c) 96×96

Due to the reduction of features in the input image, it is no longer necessary to have an excessively deep network structure. Therefore, we reduce one of the bottleneck layers to make the number of neural network parameters smaller and lighter. In order to reduce the computation without affecting the accuracy too much, we introduce the ECA module in the first and last bottleneck, and the adjusted network structure is shown in Figure 5.

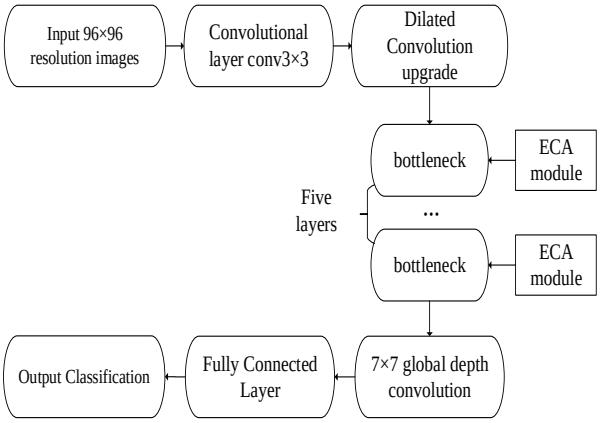
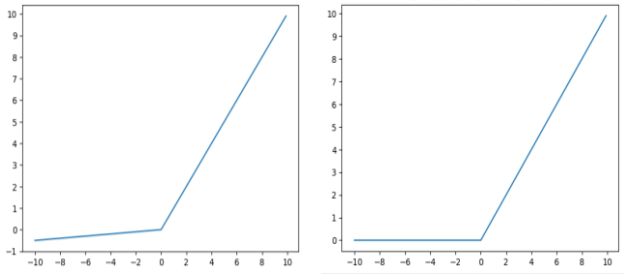


Figure 5. Adjusted network structure diagram

3.2. Activation function

MobileFaceNet uses bottleneck architecture as feature extraction network, *i.e.*, bottleneck layer design, which adds one step of deep convolution between two point-by-point convolutions. Since the Sigmoid[26] function involves exponential and division operations, which is computationally intensive and the embedded devices cannot afford this level of computational consumption. The activation function in the bottleneck layer of the original model is PReLU, which is optimized compared to the ReLU6 used in MobileNetV2, but the effect is very limited. The PReLU is used as the activation function after the first point-by-point convolution and deep convolution, and the linear activation function after the last point-by-point convolution. Figure 6(a) shows the PReLU function, and Figure 6(b) shows the function image of ReLU. ReLU6 function limits the upper bound to 6, which alleviates the problem of Relu[27] gradient disappearance caused by large gradients flowing through the neurons during the computation. In fact, this does not completely prevent this phenomenon, we replace it with the h-ReLU6 function, which is based on the Swish function proposed by Google[28][29]. Google has proven through extensive experiments that the Swish function

outperforms all current activation functions, and its function image is shown in Figure 7.



(a) PReLU (b) ReLU

Figure 6. PReLU(a) and ReLU(b) action function

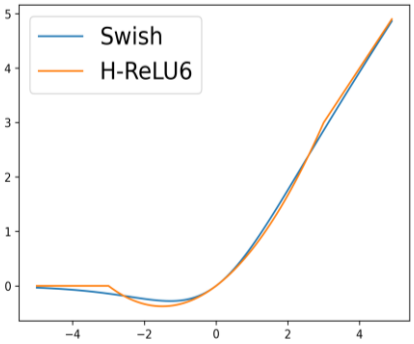


Figure 7. Swish and h-ReLU6 action function

As shown in the equation (6), the h-ReLU6 function is improved by imitating the Swish function though. As shown in Figure 7, the function image is also morphologically close to the Swish function, but it is still essentially a ReLU6 function rather than a Sigmoid function, which enables excellent activation performance to be achieved by a smaller computational effort.

$$f(x) = x \cdot \frac{\text{ReLU6}\cdot(x+3)}{6} \quad (6)$$

Figure 8 shows the structure of the bottleneck layer after replacing the activation function.

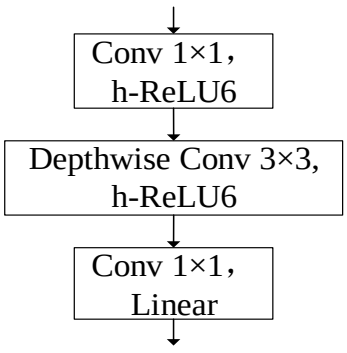


Figure 8. Replace the activated bottleneck

3.3. ECA Module

The ECA module assigns weights to individual channels by introducing a small number of parameters and to help neural network learn important features. There have been many studies from the spatial dimension[30] perspective of optimizing neural networks. We make an attempt from the perspective of weight optimization. In the literature[19], the Squeeze-and-Excitation (SE) module[31][32] is introduced in MobileFaceNet. The structure is shown in Figure 9, and it can be seen from the experimental results that the channel attention mechanism using the feature repositioning[33] strategy improve the recognition rate of the MobileFaceNet model in some cases, but it has some impact on the memory and computational power occupied by the model.

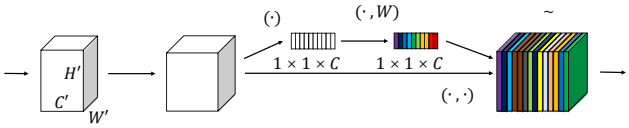


Figure 9. SE module

In this paper, we choose the effective channel attention ECA module, whose structure is shown in Figure 10. It improves on SE to avoid dimensionality reduction and effectively capture cross-channel interactions and calculates the weights of each channel by a very simple structure. The module implements a local cross-channel interaction strategy without dimensionality reduction. An adaptive selection of the one-dimensional convolutional kernel size reduces the complexity of the model while achieves a performance gain from the complex attention module. Figure 11 shows the process of ECA module, firstly, the global average pooling (GAP) is performed on the original features of the input image to obtain all the unidimensional features. Then ECA captures the local cross-channel interactions by fast one-dimensional convolution of size k, where the parameter k can be generated by the adaptive function according to the size of the input channel *C*. The next step is to generate the weight shared of each channel by the Sigmoid function, and to combine the original input features with the channel weights to obtain the features with channel attention. Unlike the SE module, the ECA module aims to learn effective channel attention with low model complexity, which minimizes its additional parameters and computational effort.

The ECA module has only one 1×1 convolutional layer with kernel size k, which indicates the coverage of local cross-channel interactions, because it is unnecessary to calculate the attention between two channels in the SE module. Two fully connected layers will introduce too many parameters and computation, which is not suitable for the weak computational environment in this paper. Considering that the introduction of additional modules will inevitably increase a small amount of parameters and computation, only the ECA module is embedded in the

very first and last bottleneck for extracting shallow and deepest features, respectively. Figure 12 shows the structure of the bottleneck layer after the ECA module is embedded.

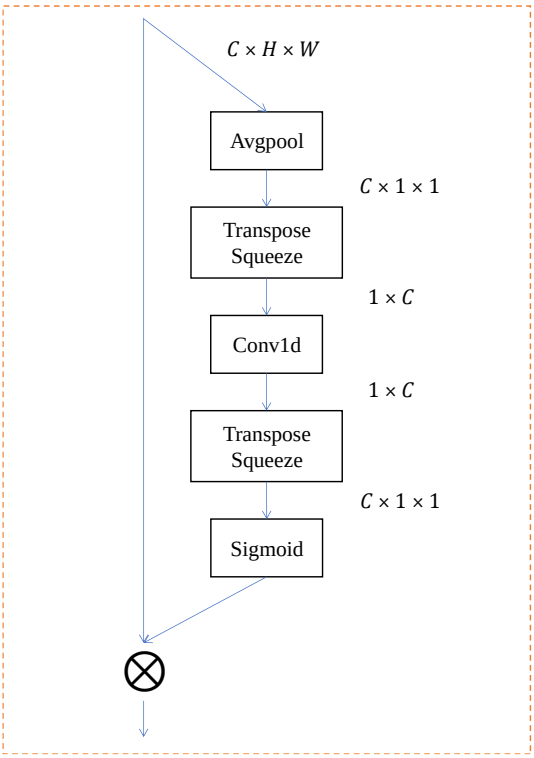


Figure 10. Structure of ECA module

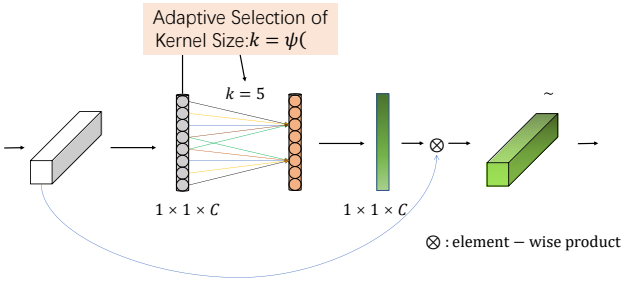


Figure 11. ECA module

4. Experimental Results and Analysis

4.1. Experimental Configuration

The experiments were conducted using Pytorch as the framework for building deep learning network structures, with Windows 10 as the operating system, and Python 3.6 as the programming language for training and testing. The hardware platform is an Intel Core TM i7-10700 CPU with 32 GB of memory and a NVIDIA GeForce RTX 2080 Ti GPU with 11 GB of video memory. Opencv is used to complete the work related to image preprocessing. The total number of training rounds of the network model is 500, and the batch size is 32. The learning rate is initially 0.1 using the SGD optimizer, and gradually decreases as the training progresses, and the minimum learning rate is 0.0001.

In addtion, we deployed the network structure of MobileFaceNet before and after optimization to the embedded platform in turn, and evaluate the merits of the model in terms of memory consumption, recognition rate and recognition accuracy.

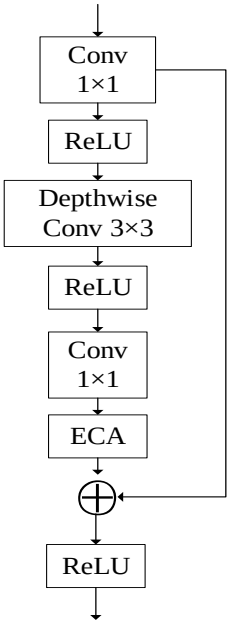


Figure 12. Bottleneck after embedding ECA

4.2. Data Processing

In this paper, we use the CASIA-Webface[33] face dataset as the training dataset. Using MTCNN[35] face detection method is used to re-detect the images in the dataset, and the detected face images are cropped to 96 × 96. As shown in Figure 13, the algorithm is based on the three-level neural networks P-Net, R-Net, and O-Net, progressively generating, correcting, and accurating candidate frames and finally generate the positions of five feature points of the left eye center, right eye center, nose tip, left mouth corner, and right mouth corner of the face.

Since the captured faces have a certain angle of in-plane deflection and some different distances between the faces and the camera, which leads to inconsistent face sizes. Therefore, it is necessary to align the captured faces. In this paper, we directly calculate the transformation matrix based on the positions of the five feature points localized by MTCNN and the standard face template feature point positions using Similarity Transformation.

Assuming that the original coordinates are (x, y) and the coordinates after similarity transformation are (x_1, y_1). As shown in formula (7), s_1 and s_2 is the scaling factor,

(t_1, t_2) is the translation factor, r = 1 means x-flip, otherwise no flip, and θ is the rotation angle. The transformation matrix is solved according to the coordinates of the feature points generated by MTCNN and the standard face template feature points to achieve face alignment.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} s_1\cos\theta & s_1\sin\theta & t_1 \\ -r\cdot s_2\sin\theta & r\cdot s_2\cos\theta & r\cdot t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (7)$$

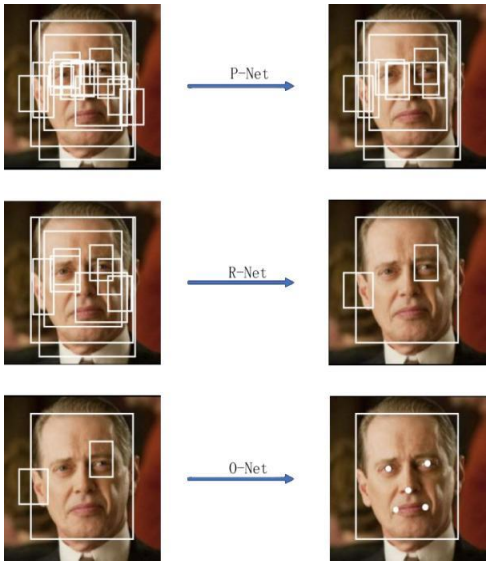


Figure 13. MTCNN detection effect

4.3. Accuracy Evaluation

The purpose of this experiment is to verify the degree of improvement of the recognition effect of each step of the MobileFaceNet network compared with the original network, so the original MobileFaceNet network is chosen as the reference object. The LFW[36] and VGGFace2[37] of the public dataset and the self-built test set were chosen respectively, where the self-built test set contains 30 people with 12 photos each, which is divided into two parts. Firstly, two photos are randomly selected from each person, totaling 60 photos, to form the face library. The remaining 10 photos from each person are used as test samples, totaling 300 photos. The number of people and the number of images in each data set are shown in Table 3 below.

Table 3. Face data set

Dataset	Number of people	Number of images
CASIA-Webface	10575	494414
LFW	5749	13233
VGGFace2	9131	3.31M
Self-built face	30	360

dataset

The accuracy test results are shown in Figure 14. The accuracies of the original network on LFW, VGGFace2 and the self-built dataset are 98.62%, 97.48%, 90.67%, respectively, which become to 98.14%, 96.64%, 90.13%, after adjusting the network structure. The accuracies after replacing the activation function are 98.35%, 96.25%, 90.33%, respectively, which are 98.52%, 97.54%, 91.33% after introducing the ECA module. Theoretically, reducing the number of network layers will lead to a significant decrease in recognition accuracy, but from the experimental results, it seems that the recognition accuracy of the model for the three test sets of LFW, VGGFace2 and self-built datasets decreases by 0.48%, 0.84% and 0.52%, respectively. It is not a significant decrease, which shows that there are few features in face recognition, the resolution of the input image can be reduced appropriately, and the number of network layers and parameters can be reduced accordingly.

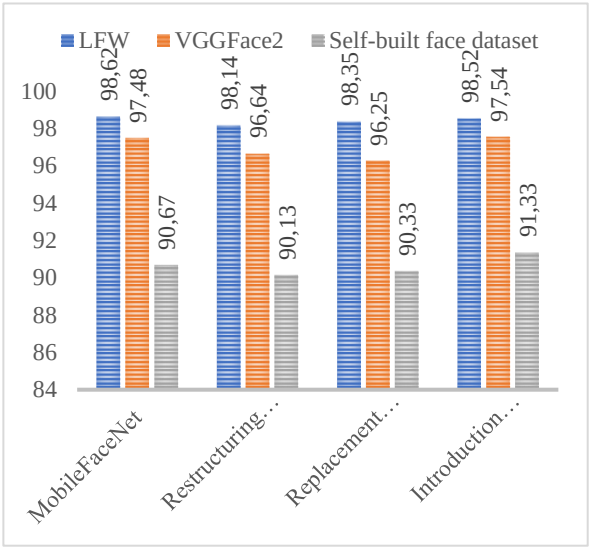


Figure 14. Recognition accuracy in different databases

After replacing the activation function, the model improves the accuracy by 0.21% and 0.20% in LFW, and self-built datasets, respectively, compared to the previous step, although the accuracy is reduced by 0.39% in VGGFace2. It proves that in some cases h-ReLU6 outperforms the original PReLU.

Finally, the accuracy of all three datasets improved after introducing the ECA module, especially in the VGGFace2 and self-built datasets, and were even higher than before the lightweight treatment. Overall, the recognition rate of the model after the lightweighting treatment still performs well, reaching 98.52%, 97.54%, and 91.33% in the LFW, VGGFace2, and self-built datasets, respectively.

4.4. Complexity Test

In this paper, we use the memory occupied by the model and the recognition speed on the embedded device as the criteria for complexity testing. The test results are shown in Table 4, where the recognition speed is the average time taken to recognize each face image in the embedded device. From Table 4, we can see that the model initially occupies 4.9 MB of memory, and the recognition speed is 117 ms per image. For the optimized network structure, the network layers and the amount of computation are significantly reduced, which makes it occupy 1.8 MB of memory and the recognition speed per image is increased by 34 ms. After replacing the activation function, the memory occupancy is increased by 0.1 MB and the recognition speed per image is increased by 5 ms. As shown in the following equation (8) for the function of PReLU, compared with it, the complexity of h-ReLU6 as shown in the equation(6) is relatively higher and the computation is larger, but the overall impact seems to be small. The introduction of the ECA module also leads to an increase of 0.2MB of occupied memory and an increase of 0.4 ms per image recognition speed. It increases the complexity of the model, however, compared with other channel attention modules, the complexity of the ECA module occupies less memory and computation.

$$f(x) = \begin{cases} x & ,x > 0 \\ ax & ,x \leq 0 \end{cases} \tag{8}$$

Although replacing the activation function and introducing ECA both lead to an increase in the model's memory and a decrease in recognition speed, overall it takes up 1.5 MB less memory than processing and increases recognition speed by 32 ms per image.

Table 4. Model Complexity Test Results

Models	Memory(MB)	Time(ms/sheet)
MobileFaceNet	4.9	117
Restructuring the network	3.1	76
Replacement activation function	3.2	81
Introduction of ECA module	3.4	85

5. Conclusion

In this work, we optimized the face recognition model MobileFaceNet for limited computation environments. Firstly, we reduced the resolution of the input image and the number of network layers to reduce the parameters of the model and the computation complexity. Then, the activation function in the bottleneck layer is replaced with h-ReLU6, and the ECA mechanism is introduced to achieve automatic parameter tuning. Finally, the proposed model was test on public datasets LFW, VGGFace2 and the self-built datasets. Moreover, we deployed the optimized model in the embedded devices for testing. Experiment results shows that there is a slight decrease in the recognition rate in the LFW dataset for the optimized model, however it is negligible overall. In general, the lightly processed model is more suitable for use in a weak computing environment than the original model.

Funding. This work was supported by Special Funds for the Construction of an Innovative Province of Hunan, No. 2022GK4009, 2021 Central South University Graduate School-Enterprise Joint Innovation Project (Second Batch), No. 40, and Scientific research project of Hunan Provincial Department of Education, No. 21B0833.

References

[1] Turk M, Pentland A. Eigenfaces for recognition[J]. Journal of cognitive neuroscience, 1991, 3(1): 71-86.
[2] Wold S, Esbensen K, Geladi P. Principal component analysis[J]. Chemometrics and intelligent laboratory systems, 1987, 2(1): 37-52.
[3] Belhumeur P N, Hespanha J P, Kriegman D J. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2002, 19(7):711-720.
[4] Wright J, Yi M, Mairal J, et al. Sparse Representation for Computer Vision and Pattern Recognition[J]. Proceedings of the IEEE, 2010, 98(6):p.1031-1044.
[5] Li S, Deng D. Editorial: Smart Technologies Improve our Daily Lives[J]. EAI Endorsed Transactions on Internet of Things, 2019, 5(18):163845.
[6] Krizhevsky A, Sutskever I, Hinton G. ImageNet Classification with Deep Convolutional Neural Networks[J]. Advances in neural information processing systems, 2012, 25(2).
[7] TAIGMAN, YANIV, YANG, MING, RANZATO, MARC'AURELIO, et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification[C]. //2014 IEEE Conference on Computer Vision and Pattern Recognition: 2014 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014), 23 -28 June 2014, Columbus, Ohio. 2014:1701-1708.
[8] F Schroff, Kalenichenko D, Philbin J. FaceNet: a Unified Embedding for Face Recognition and Clustering[J]. IEEE, 2015.
[9] Chen C, Cheng R. Improving Indoor Localization Based on Artificial Neural Network Technology[J]. EAI Endorsed Transactions on Internet of Things, 2018, 4(16):159633.
[10] IANDOLA, FORREST N., HAN, SONG, MOSKEWICZ, MATTHEW W., et al. SqueezeNet:

AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size[J]. . 2016.
[11] CHOLLET, FRANÇOIS. Xception: Deep Learning with Depthwise Separable Convolutions[J]. 2016.
[12] Howard A G, Zhu M, Chen B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications[J]. 2017.
[13] Sandler M, Howard A, Zhu M , et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks[C]// 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Recognition (CVPR). IEEE, 2018.
[14] CHEN S, LIU Y, GAO X, et al. MobileFaceNets: Efficient CNNs for accurate real-time face verification on mobile devices [C] // 1761 Journal of Beijing University of Aeronautics and Astronautics 2020 Chinese Conference on Biometric Recognition.Berlin: Springer, 2018: 428-438.
[15] IOFFE, SERGEY, SZEGEDY, CHRISTIAN. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. 2015.
[16] Zhang Z. H.,Wang R.. An improved face recognition method based on MobileFaceNet network[J]. Journal of Beijing University of Aeronautics and Astronautics,2020,46(9):1756-1762.DOI:10.13700/j.bh.1001-5965.2020.0049.
[17] Lee H J, Kim H E, Nam H. SRM : A Style-based Recalibration Module for Convolutional Neural Networks[J]. 2019.
[18] Zhang X, Zhao R, Qiao Y, et al. AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.
[19] Li Hang. Design and implementation of a lightweight face recognition system based on MobileFaceNet [D]. Southwest University, 2020. DOI:10.27684/d.cnki.gxndx.2020.001575.
[20] Jie H, Li S, Gang S , et al. Squeeze-and-Excitation Networks[J].IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, PP(99).
[21] Deng J, Guo J, Zafeiriou S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition[J]. 2018.
[22] Wang Bihao. Research and optimization of face screening algorithm [D]. University of Electronic Science and Technology,2019.
[23] Wang Q, Wu B, Zhu P, et al. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.
[24] ZHAO, QI, LIU, JIAHUI, ZHANG, BOXUE, et al. Interpretable Relative Squeezing bottleneck design for compact convolutional neural networks model[J]. Image and vision computing,2019,89(Sep.):276-288. DOI:10.1016/j.imavis.2019.06.006.
[25] Chen Y, Peng F, Kang X, et al. Depthwise Separable Convolutional Neural Network for Image Forensics[C]// 2019 IEEE Visual Communications and Image Processing (VCIP). IEEE, 2019.
[26] Haykin S S , Gwynn R. Neural Networks and Learning Machines [M]. China Machine Press, 2009.
[27] Grimstad B, Andersson H. ReLU Networks as Surrogate Models in Mixed-Integer Linear Programs[J]. Computers & Chemical Engineering, 2019, 131(Dec.5):106580.1-106580.15.
[28] Morrison A F, Epifanovsky E, Herbert J M. Double-buffered, heterogeneous CPU + GPU integral digestion algorithm for single- excitation calculations involving a large number of excited states[J]. Journal of Computational Chemistry, 2018, 39(26).
[29] Petersen P, Voigtlaender F. Optimal approximation of piecewise smooth functions using deep ReLU neural networks[J]. 2017.
[30] Lecun Y, Bengio Y , Hinton G. Deep learning[J]. Nature, 2015, 521(7553):436.
[31] Ghimire D, Jeong S, Joonwhoan Lee.... Facial expression recognition based on local region specific features and support vector machines[J]. Multimedia Tools & Applications, 2017, 76(6):7803-7821.
[32] Jie, Shen, Samuel, et al. Squeeze-and-Excitation Networks.[J]. IEEE transactions on pattern analysis and machine intelligence, 2019.
[33] Seo S, Ki S, Kim M. A Novel Just-Noticeable-Difference-based Saliency-Channel Attention Residual Network for Full-Reference Image Quality Predictions[J]. 2019.
[34] Dong Y, Zhen L, Liao S, et al. Learning Face Representation from Scratch[J]. Computer Science, 2014.
[35] Kaziakhmedov E, Kireev K, Melnikov G, et al. Real-world attack on MTCNN face detection system[C]// 2019.
[36] Huang G B, Mattar M, Berg T, et al. Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments[J]. Month, 2008.
[37] Cao Q, Shen L, Xie W, et al. VGGFace2: A dataset for recognising faces across pose and age[C]// IEEE Computer Society. IEEE Computer Society, 2017.