Multimedia Systems (2023) 29:129–138
https://doi.org/10.1007/s00530-022-00973-z

REGULAR PAPER

A lightweight CNN-based algorithm and implementation on embedded system for real-time face recognition

Zhongyue Chen[1] · Jiangqi Chen[1] · Guangliu Ding[1] · He Huang[1]

Received: 31 May 2021 / Accepted: 24 June 2022 / Published online: 10 August 2022

Abstract
Deep learning has become the main solution for face recognition applications due to its high accuracy and robustness. In recent years, a batch of research on lightweight convolutional neural networks (CNNs) has emerged, bringing new ideas to the economic application of face recognition systems. In this paper, a lightweight face recognition algorithm is proposed to reduce the number of parameters and calculations of the face feature extraction network. The most important part of our approach lies in designing a novel inverted residual shuffle unit (IR-Shuffle). After being trained by ArcFace loss on a GPU workstation, our model built on improved IR-Shuffle blocks of size 1.45 MB achieves an accuracy of 98.65%. In terms of running time, our model is 5 ms faster than the current fastest MobileFaceNet, with only about 0.5% drop in accuracy. The proposed algorithm is implemented and optimized on the Jetson Nano embedded platform, and accurate and real-time deployment of the face recognition system is realized. The system takes 37 ms to perform the complete face detection and recognition and is robust to complex backgrounds and ambient light changes. Experimental results show that our system is of practical application value.

Keywords Lightweight convolutional neural networks · Face recognition · Embedded systems · Shuffle block

1 Introduction

With social informatization development, face recognition has been widely applied in various fields such as access control, public security, consumer entertainment, etc. Face recognition algorithms can be divided into two categories: traditional methods and deep learning based approaches. Convolutional neural networks (CNNs) [1] have become one of the most successful deep learning models for image processing due to their good feature extraction ability since AlexNet [2] winning the ImageNet Challenge (ILSVRC 2012 [3]). Compared with the traditional methods, such as template matching and subspace analysis, deep learning based face recognition algorithms are more robust to environmental variation [4].

By building deeper and larger CNN models, it is possible to aggregate adequate spatial information within a face image and achieve accurate recognition. However, high-accuracy face recognition based on big CNN models would inevitably lead to high computational resources and storage requirements. For applications on embedded platforms with limited resources, lightweight design is necessary.

Several computation-efficient basic architectures such as MobileNet [5, 6] and ShuffleNet [7, 8] have been proposed for general visual tasks. However, these common lightweight CNNs become less accurate in face recognition compared with other state-of-the-art results with big models. Face recognition is more effective when the MobileNet model is larger. However, when the model needs to be compressed further, the drawback of no communication among channels starts to be highlighted and the feature extraction ability of the network is reduced. The ShuffleNet with enhanced channel communication can improve this, but for small models with limited convolutional kernels, the traditional residual structure of feature extraction cannot achieve satisfactory recognition results.

Communicated by F. Wu.

Zhongyue Chen and Jiangqi Chen contributed equally to this work.

✉ Zhongyue Chen
chenzy@suda.edu.cn

[1] School of Electronic and Information Engineering, Soochow University, Suzhou 215006, Jiangsu, China

In this paper, we carefully design a lightweight CNN structure for feature extraction from face images. To overcome the memory and computational limitations of embedded systems, we first compress the model size and reduce the computational complexity. Face images are usually RGB images, and the color of skin, texture, shape of key parts, and the interrelationship among key parts are important features that affect the recognition performance. Therefore, when designing the face feature extraction network, both spatially correlated and channel correlated features are important. Based on the above considerations, we use channel shuffle to improve the extraction ability of channel correlated features and present a novel inverted residual shuffle (IR-Shuffle) unit with enhanced feature comprehension to compensate for insufficient model depth. Inspired by ShuffleNet, we introduce deep separable convolution with channel shuffle to achieve inter-channel feature communication. However, for the embedded system deployment, the model size is further compressed. As a result, the number of kernels in each layer is limited, whereas there is rich feature information in faces, and using Shuffle units with conventional residual structures will extract insufficient special information and degrade the recognition performance. Therefore, we replace the conventional shuffle units with improved IR-shuffle units, in which an inverted residual structure is incorporated. The number of channels is first expanded and then compressed, which further improves the expression of face features. Constructing networks using IR-Shuffle units can reduce the number of parameters required for comparable recognition accuracy.

We evaluate the proposed network on the challenging labeled faces in the wild (LFW) benchmark [9]. After being trained on the large-scale face dataset CASIA WebFace [10] containing massive noise data, our 1.45 MB size model achieves 98.65% accuracy on LFW in 13.4 ms. We also build a face recognition system to examine our model on an embedded platform. Optimized by both software and hardware acceleration, the system takes 14∼16 ms to extract face features for identification and is robust to changes in complex backgrounds and ambient light.

An overview of the rest of the paper is as follows: Sect. 2 reviews the related field literature. Section 3 describes our novel IR-Shuffle unit and demonstrates the feature extraction network structure. In Sect. 4, we present some experimental results of our design. Finally, we implement the proposed face recognition method on the embedded platform in Sect. 5.

2 Related work

2.1 Model design

Compared with the traditional methods, such as template matching and subspace analysis, deep learning based face recognition algorithms are more robust to environmental variation [4]. With the intensive study of neural networks, a series of deep learning models have been proposed for face recognition. For example, the DeepFace network proposed by Facebook [11] has achieved a 95.9% recognition rate on LFW. The VGG network proposed in [12] greatly improved face recognition accuracy by increasing the number of layers in the network and using small-scale convolutional kernels. As reported, the recognition rate on the LFW dataset was 98.9%. GoogleNets [13–16] utilizes an inception structure to allow the fusion of features at different scales. The DeepID network [17] integrates the local and global features and employs Bayes' theorem to complete face recognition. In many real-world scenarios, face images are of low quality due to the limitations of imaging systems and conditions. Deep learning methods are also used for face hallucination to improve the subsequent processing by recovering high-resolution face images [18–20].

The metric and loss function design for training will also gravely affect face recognition accuracy. Face recognition involves thresholding the distance between extracted face features [21]. To this end, distance metrics, such as Euclidean distance, cosine distance, and joint Bayesian distance, are considered in loss function design. Note that a good loss function, such as contrastive loss of DeepID [17, 22, 23], triplet loss of FaceNet [21], and center loss [24], benefits generalization, and makes the system more robust. To increase the inter-class distance and decrease the intra-class distance, Sphereface [25] utilizes an A-softmax loss function, which makes the learned features discriminatively span on a hypersphere manifold as prior. Variations of the A-softmax function, including AM-softmax [26] and CosFace [27], can further enhance face recognition.

The increasing demands to run high-quality deep neural networks on embedded devices inspire lightweight model designs. MobileNetV1 [5] presents a streamlined architecture that uses depthwise separable convolutions and achieves good performance on ImageNet compared to other networks at about the same size. MobileNetV2 [6] introduces an inverted residual structure with a linear bottleneck and improve the mobile model performance on various tasks. ShuffleNet [7, 8] uses pointwise group convolution to reduce computation cost and utilizes channel shuffle to maintain accuracy. Channel shuffle turns out to be important for achieving good performance in terms of speed and accuracy tradeoff. This paper presents a novel shuffle unit with an inverted residual structure inside. We use such new units rather than conventional shuffle units in the feature extraction networks to explore the rich details in face images sufficiently.

2.2 System implementation

The current mainstream solution is to deploy the well-trained CNN model on the cloud platform or utilize a high-performance computing host (HPC) at the back end of image acquisition [28]. In this situation, local images are transmitted to the cloud, and then the recognition results are returned to the edge device. Although the cloud provides efficient computing power for complex tasks, recognition would be so affected by network conditions that long transmission delays may degrade the user experience. The cloud HPC is also expensive and difficult to configure. Moreover, the local edge devices remain idle most of the time, wasting significant amounts of computational resources. Increased edge devices produce huge amounts of data that uploading all the data is neither pragmatic nor feasible. Edge computing [29–31] on embedded devices is necessary for real-time processing, requiring the localization of deep learning algorithms. In recent years, several dedicated chip design solutions for face recognition based on neural networks have become available [32–34] at the expense of additional hardware costs. System implementations on general local devices are still necessary.

Built on the novel shuffle unit, we propose a lightweight face feature learning network with a smaller model and lower computation, suitable for edge computing. We also build a complete face recognition system with the proposed network on a general embedded device fit for edge computing. The system performs well in terms of accuracy and latency tradeoff and takes up less storage space at the same time.

3 Lightweight feature extraction network

3.1 The inverted residual shuffle unit

This section proposes a novel shuffle unit called IR-Shuffle to alleviate performance degradation in small models and accelerate the convergence. Taking IR-Shuffle as the building blocks, we design a compact feature extraction model for face recognition.

Channel shuffle affords to reduce the accuracy degradation caused by the limited number of channels in tiny models. We first review the standard shuffle unit illustrated in Fig. 1a [8]. At the beginning of each unit, feature channels are divided into C and C' channels. The main branch with C channels consists of three convolutions with consistent channel numbers. Both pointwise convolution (1×1 Conv or PWConv) and depthwise convolution (DWConv) operate only on identical channel groups and significantly reduce computation cost. The other C' channel part (side branch) remains an identity and is concatenated directly with the

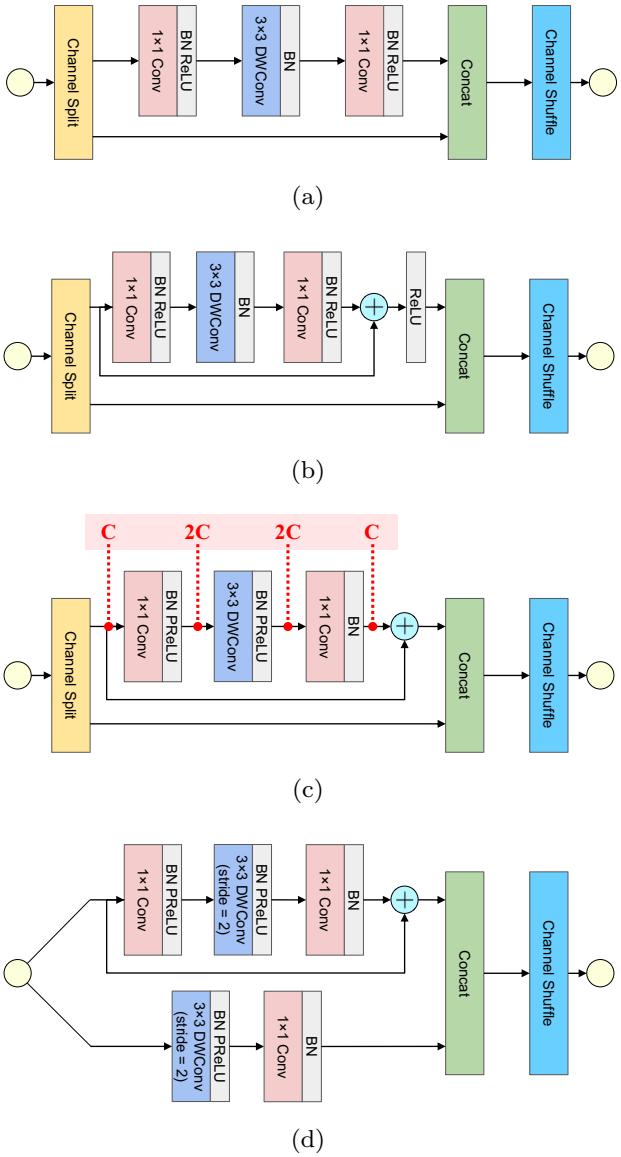


Fig. 1 Shuffle units. **a** the standard shuffle unit; **b** the shuffle unit with residual (Res-Shuffle); **c** our shuffle unit incorporating inverted residual (IR-Shuffle); **d** IR-Shuffle for spatial downsampling (2×). DWConv: depthwise convolution. BN: batch normalization

convolution output to reduce the degree of fragmentation. After concatenation, channel shuffle operation is used. The feature map channels are rearranged for the subsequent convolutions to enable information flow between different channels and strengthen the representation.

For very deep networks with over 100 layers, adding a residual path, as shown in Fig. 1b, turns out to be important for training convergence acceleration [8]. However, if we apply the usual residual structure in the shuffle unit conducting PWConv and DWConv, problems arise. If PWConv is performed to reduce the dimensionality before the standard convolution as the traditional residual

module did, we will obtain few features due to the channel number limitation. In processing face images with rich details, these features are far from enough.

We incorporate the inverted residual with channel shuffle to form a novel IR-Shuffle unit for the above reasons. Figure 1c shows that the proposed IR-Shuffle unit first uses PWConv to expand channels (for example, increase from C to $2C$), perform DWConv to extract features (C remains unchanged), and finally use PWConv again to compress the dimensionality (reduce from $2C$ to C). Such spindle structure enhances the feature extraction capability for economic tiny models.

As the spindle-shaped convolutions generate compressed features, ReLU transformation will further damage these features. Therefore, we delete the ReLU layer before the channel shuffle operation and adopt a structure like the linear bottleneck in our IR-shuffle. To prevent model streamline from increasing the over-fitting risk, we use the PReLU activation function [35] after convolution rather than ReLU, at the cost of a tiny increase in parameters. Like the original shuffle unit, the IR-shuffle is slightly modified for spatial downsampling. As illustrated in Fig. 1d, the channel split operator is removed to double the output channels. Additional convolutions are performed on the side branch before concatenation.

3.2 Network architecture

Several IR-shuffle blocks are stacked to construct the feature extraction network for face recognition. The overall structure is summarized in Table 1. The network first boosts the feature map dimensionality through a standard convolution layer, then uses 8 IR-shuffle units for feature extraction, and finally outputs a face feature vector of 128 dimensions through PWConv and DWConv. For simplicity, we always split the channels in half. In the stage of extracting features using IR-Shuffle units, we perform spatial downsampling (IR-Shuffle, stride = 2) three times followed by dimension invariant IR-Shuffle units (stride = 1) for high level feature extraction. To obtain highly discriminative features for face recognition, we use ArcFace loss [36] to train our model. Distributing the embedding features on the hypersphere, the loss function is defined by

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_j}} \tag{1}$$

where θ_j is the angle between the weight W_j and the feature x_i. $W_j \in \mathbb{R}^d$ denotes the j-th column of the weight $W \in \mathbb{R}^{d\times n}$ and $x_i \in \mathbb{R}^d$ denotes the deep feature of the i-th sample belonging to the y_i-th class. N and n are the batch size and the class number, respectively. m is the additive angular margin penalty between x_i and W_{y_i} to enhance the intra-class compactness and inter-class discrepancy. s is a feature scale

Table 1 Overall structure of the face feature extraction network

Layer	Output Size	Kernel Size	Stride	Main Branch Channels[1]	Side Branch Channels[2]	Output Channels
Face Image	112×96	–	–	–	–	3
Conv	56×48	3×3	2	–	–	32
IR-Shuffle1	28×24	–	2	32-96-96-16	32-64-32	48
IR-Shuffle2	28×24	–	1	24-48-48-24	24	48
IR-Shuffle3	14×12	–	2	48-192-192-48	48-96-48	96
IR-Shuffle4	14×12	–	1	48-96-96-48	48	96
IR-Shuffle5	14×12	–	1	48-96-96-48	48	96
IR-Shuffle6	14×12	–	1	48-96-96-48	48	96
IR-Shuffle7	7×6	–	2	96-192-192-96	96-192-96	192
IR-Shuffle8	7×6	–	1	96-192-192-96	96	192
PWConv1	7×6	–	1	–	–	512
DWConv	1	7×6	1	–	–	512
PWConv2	1	–	1	–	–	128

[1] Main branch channels illustrate the channel change on the main branch. "24-48-48-24" means that we allocate 24 channels for the main branch, and the output channel number of the three sequential convolution blocks are 48, 48, and 24, respectively

[2] Side branch channels demonstrate the channel change on the side branch. For direct connections, "24" is the number of channels assigned. In the IR-Shuffle units for spatial downsampling (stride = 2), "32-64-32" implies side branch channel change (see Fig. 1(d))

factor. Training face recognition models by ArcFace loss greatly alleviates the long-tail data class imbalance and the "lazy learning" of simple-difficult samples.

4 Performance evaluation

The face feature extraction network proposed in this paper uses the CASIA WebFace [10] as the training set, a large-scale face dataset containing 10,575 identity objects and 494,414 images published by the Chinese Academy of Sciences. Data are preprocessed before training. The face area and landmarks are obtained using an elegant detection network, RetinaFace [37, 38]. Then the face is cropped and aligned to 112×96 as inputs. Data enhancement is applied to the processed data. The stochastic gradient descent algorithm is employed to update the parameters, where the weight decay and the momentum are respectively taken as 5 and 0.9. The learning rate is 0.1, which is reduced by a factor of 0.1 after training 36 epochs and 52 epochs. After training, we conduct the face recognition experiments and compute the recognition accuracy to examine the effect of the trained feature extraction network.

Face recognition is to determine who this person is by comparing the degree of similarity between the feature vector extracted from the input image and those registered in the database. Every comparison, in this case, is a face verification. Face verification is a binary classification problem that only requires to determine whether two face feature vectors are similar or not. Therefore, after the feature extraction network, the distance between feature vectors (e.g., Euclidean distance, cosine similarity, etc.) is calculated in the feature space. In our experiments, the cosine similarity is adopted. In the feature space, two vectors are similar if they coincide infinitely, which means that the angle between them tends to zero. The cosine similarity between the two feature vectors can be calculated by

$$\cos\theta = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2}$$

where A_i and B_i denote the element of the two vectors, respectively. If the angle is larger, $\cos\theta$ is smaller, which means that the faces are different. It is possible to determine whether two images belong to the same person by comparing the cosine similarity to a pre-defined threshold.

The test set used in the experiment is the LFW dataset, which has 13,233 face images with 6000 face pairs and contains 5,749 individuals. The database is designed specifically for the face verification task. Pairs of two face images are labeled as either "matched" or "mismatched" depending upon whether the images are the same person or not. The images in this dataset are taken from various natural scenes, such as different postures, lighting changes, and background changes. Our method is implemented in PyTorch [39]. All the experiments in this section are conducted on a workstation with 4GHz CPU, 32 G RAM and RTX 2080Ti GPU.

Table 2 Face feature extraction network parameters

Parameter Numbers	Computation Amount	Model Size
431.60 KB	37.712 MB	1.45 MB

Table 3 Comparison of multiple algorithms

Algorithm	Accuracy	Model size	Latency
FaceNet [21]	99.63%	30 MB	–
MobileFaceNet [40]	99.20%	3.95 MB	18.3 ms
ShiftFaceNet [41]	96.00%	3.1 MB	–
ShuffleFaceNet [42]	99.07%	1.9 MB	–
Our model (original shuffle unit)	97.65%	1.8 MB	13.7 ms
Our model (improved IR-Shuffle)	98.65%	1.45 MB	13.4 ms

The number of parameters, calculations, and model size of the proposed face recognition network are shown in Table 2. Table 3 compares the recognition accuracy achieved by several state-of-the-art face recognition networks on the LFW. Our model built on improved IR-Shuffle blocks of size 1.45 MB achieves an accuracy of 98.65%. We find that FaceNet constructs a large model and achieves the best accuracy of 99.63%. MobileFaceNet of 3.95 MB achieves 99.20%, compared to which our network reduces the parameter size by more than 1.5×, with only about 0.5% drop of accuracy. Compared to ShuffleFaceNet, we also obtain a smaller model with a drop of accuracy within 0.5%. Our method is also superior to the ShiftFaceNet in terms of both accuracy and model size. Using latency as the direct metric to measure the computation complexity, our model is 5 ms faster than the fastest MobileFaceNet. Although the proposed algorithm is slightly less accurate than some other face recognition algorithms, both the number of parameters and the model size of our model are much lower. We also compare the accuracy of the face recognition network based on the original shuffle unit and the proposed IR-shuffle unit on the LFW dataset. Results show that using the proposed IR-Shuffle blocks to build the network is 1% more accurate than utilizing the original ones.

5 Implementation of embedded face recognition system

5.1 System framework

With the proposed lightweight CNN algorithm, a face recognition system is designed and implemented on an embedded platform. An overview of the system framework is presented in Fig. 2. Firstly, the target image is captured by a USB camera. Then the image is transmitted to the embedded development board for detection and recognition. Finally, the board is connected to the display device via HDMI to visualize the captured image and the recognition result. The core processing system includes face detection, preprocessing, feature extraction, and face identification. The main function of the detection module is to find the landmarks of the face. Our detection is performed based on RetinaFace [37, 38]. In preprocessing, we crop and align the face to be the same as training exemplars according to the detected landmarks. Then we obtain the feature vectors with the proposed lightweight model and confirm the identity through repeated face verifications. The feature recognition module belongs to the application layer and needs to be designed according to the application requirements.

5.2 Embedded implementation and optimization

To achieve better implementation of the algorithm on an embedded platform, C++ is used in this paper. The system is deployed on an embedded platform while maintaining the accuracy of face detection and recognition. The system is multi-threaded, where face detection and recognition are divided into two separate threads. By setting a semaphore for thread communication, face recognition is performed only after face detection is completed. After the face recognition operation is completed, the result is displayed on the screen.

Firstly, the USB camera captured the target image, and then the face detection module is used to detect whether a face exists in the image. If there is a face in the image, the feature vector is extracted using the face recognition module. The extracted result is then matched with the face feature vector of the registrant in the database for recognition. If the recognition is successful, the face is registered, then the database returns the identity. Conversely, if the recognition is unsuccessful, the face is not registered, the database does not return any information.

The face detection module is the basis of face recognition and is realized using a separate thread. The collected images are sent to the face detection network, and after computation, the face borders are located. Subsequently, the border regression calculation is performed, and redundant borders are removed using a non-maximal suppression algorithm. Then the confidence level of the face existence, the coordinates of the face border and the coordinates of the face feature points are obtained. These results are fed to the face identification module for further processing.

After the face detection thread is completed, the face recognition thread will read the semaphore to determine whether the face recognition is needed or not. If a face recognition operation is required, the face identification module processes the image based on the confidence level, border coordinates, and landmarks provided by the face detection module, aligning and cutting out the positive face from the image. The features are then extracted through the proposed feature extraction network and matched with the candidates in the database.

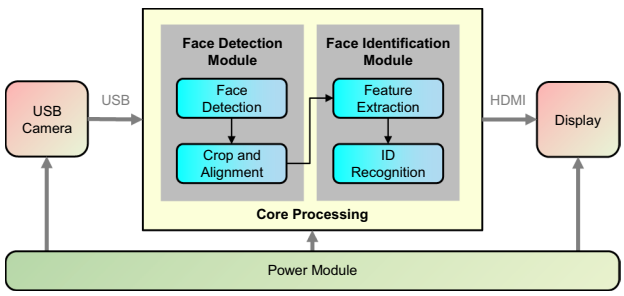
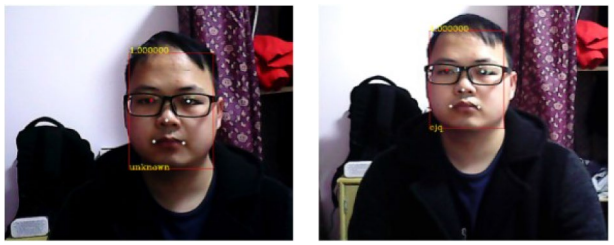


Fig. 2 An overview of the embedded face recognition system

5.3 Compression of CNN model

In general, due to the model complexity, a great number of parameters, and large computation involved in CNNs, it is difficult to run on embedded platforms with limited computational and storage resources. Therefore, it is necessary to compress the CNN model to facilitate the implementation on an embedded device. The model compression algorithm can effectively reduce parameter amount, storage size, communication bandwidth, and computation and benefit the deep network applications. To better enable the proposed lightweight CNN-based face recognition system to run on the Jetson Nano embedded platform, a weight quantization approach is used to compress the model further and reduce the storage cost. The weights of CNN models are typically stored as 32-bit floating-point numbers for training and testing. However, such high accuracy is not necessary for practical implementations. It is shown in [43] that low precision fixed-point weights can achieve competitive results. Thus, weight quantization helps avoid costly floating-point operations and reduce the hardware costs of storing weights and activating data, which is quite important for resource-constrained embedded platforms.



(a) Unregistered (b) Registered

Fig. 3 Recognition results of the face recognition system

5.4 Software acceleration

In this paper, Neon instructions, an extension of the single instruction multiple data (SIMD) with 128 bits, are used to accelerate software. This instruction can process multiple data in a single instruction cycle, which means that it accelerates the progress by processing data in parallel. The Neon instruction supports fixed-point and floating-point operations with different numbers of bits, such as 8-bit fixed-point operations, 32-bit floating-point operations, etc. This parallelism is quite useful for accelerating applications requiring a lot of computation or data processing, such as multimedia processing and neural networks.

5.5 Network inference acceleration

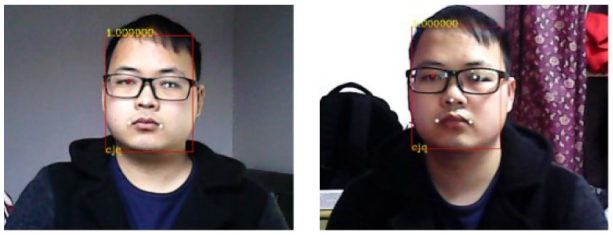
NCNN is an excellent neural network forward computing framework deeply optimized for embedded platforms such as mobile phones. The underlying NCNN computing framework is implemented in C++ and does not rely on any third-party libraries such that it is suitable for a variety of platforms. The NCNN computing framework supports the acceleration of processor multi-core parallel computing, such as the optimization of large and small core scheduling on ARM-based CPUs. Also, it supports the use of the Vulkan interface to call the image computing module on embedded platforms for faster processing. Therefore, the lightweight face recognition algorithm proposed in this paper is implemented under the NCNN software framework.

5.6 Performance testing

Figure 3 shows the results achieved by the face recognition system. The system first outlines the face with a rectangle through face detection, and the number above the box represents the confidence level. Then the identity verification is realized through the face recognition module. The face in Fig. 3a is unregistered, and the recognition result shows unknown in the image. The face in Fig. 3b is registered, and the recognition results show the name of the registered face in the image.

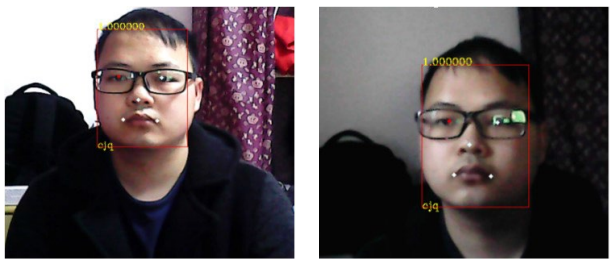
Table 4 Processing time of the embedded face recognition system

Module	Processing Time
Face Detection	19~21 ms
Face Identification	14~ 16ms



(a) Simple background (b) Complex background

Fig. 4 Recognition results in different backgrounds

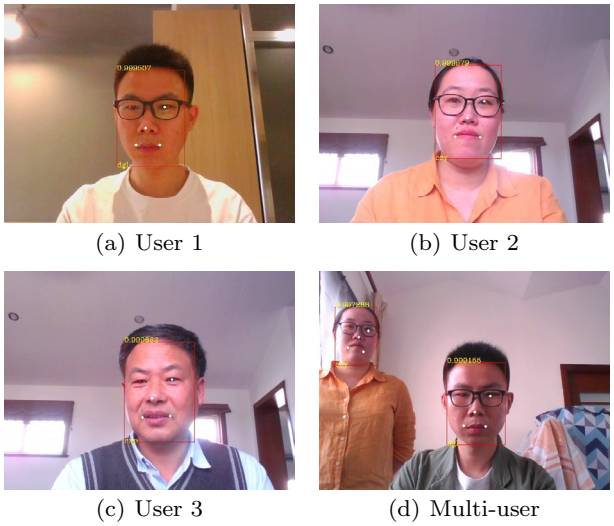


(a) Bright lighting (b) Dark lighting

Fig. 5 Recognition results at different brightness

To speed up the algorithm as much as possible, we employ the Vulkan interface function managing the graphics processing unit that comes with the CPU to speed up the algorithm. It can be seen from Table 4 that it takes about 19~21 ms to perform face detection and about 14~16 ms for face identification. Therefore, the embedded system can reach the rate of more than 25 frames per second, which illustrates that the lightweight CNN based face recognition system can be used for real-time recognition and is of great practical value.

To verify the robustness of the designed face recognition system, experiments are conducted to investigate the effects of background and illumination on face recognition. Figure 4 shows the recognition results with different backgrounds. For the simple and complex backgrounds (i.e., the cases in Fig. 4a, b), our system can accurately identify the faces. Figure 5 shows the recognition results under different illumination conditions. Figure 5a shows a bright lighting environment, and Fig. 5b shows a dark lighting environment, both of which can be accurately identified.



(a) User 1 (b) User 2

(c) User 3 (d) Multi-user

Fig. 6 Recognition results of different humans

From the comparative experiments, it is clear that environmental complexity has little influence on the face recognition system. The change of illumination within a certain degree has little effect on the face recognition system. Therefore, the face recognition system realized in this paper is robust, which can adapt to changes in background and illumination to make accurate recognition in various scenarios.

To evaluate the performance of the designed face detection recognition system in diverse scenarios, we conducted tests on different humans. We had a total of 31 volunteers recruited, including 20 males and 11 females. After we registered their identities in the system, the system was able to recognize each of them. Figure 6 gives several of the typical recognition cases. As shown in Fig. 6a–c when different registered users appeared, the system was able to successfully detect faces and recognize them with correct identities. In Fig. 6d, two different users appear at the same time, and both of them are detected and identified correctly.

6 Conclusion

The CNN based algorithms have shown remarkable performance in face recognition. However, on account of model size and computation complexity, these methods need further improvement before being applied to low-cost embedded devices. In this paper, a lightweight face recognition algorithm along with a novel IR-Shuffle unit has been proposed to reduce the number of parameters and calculations of the face feature extraction network. Experimental results have shown that the lightweight face recognition system designed in this paper greatly reduces the model size and computation complexity while maintaining comparable accuracy. The proposed algorithm has been implemented and optimized on the Jetson Nano embedded platform. Model compression and software acceleration have been invoked for better system efficiency. System tests have shown that it takes 37 ms to complete face recognition, and the system implemented is robust to changes in complex backgrounds and ambient light. The system designed in this paper has met the real-time and stable requirements of practical face recognition applications.

The application scenario considered in this paper focuses on indoor environments, however, applications of face recognition are not limited to this. For outdoor scenarios, the variable environmental conditions make it more difficult to extract face features and require specific network design to achieve satisfying performance. In practical applications, it is difficult to collect and label enough samples for innumerable real-world scenes. One promising solution is to first learn a generic model and then transfer it to application-specific scenarios. Domain adaptation [44–46] can be applied to reduce algorithmic bias in different scenarios. However, the design of embedded systems based on these models is still an open problem and we will further work on this aspect in future research.

Declarations

Conflict of interest This study was supported by Natural Science Research of Jiangsu Higher Education Institutions of China (21KJB510021). The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Bouvrie, J.: Notes on convolutional neural networks (2006)
2. Krizhevsky, A., Sutskever, I., Hinton GE Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp 1097–1105 (2012)
3. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
4. Huang, G.B., Lee, H., Learned-Miller, E.: Learning hierarchical representations for face verification with convolutional deep belief networks. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 2518–2525 (2012)
5. Howard, AG., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. (2017) arXiv preprint arXiv:1704.04861
6. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, LC.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4510–4520 (2018)
7. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In:

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 6848–6856 (2018)
8. Ma, N., Zhang, X., Zheng, HT., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision, pp 116–131 (2018)
9. Huang, GB., Mattar, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database forstudying face recognition in unconstrained environments (2008)
10. Yi, D., Lei, Z., Liao, S., Li, SZ.: Learning face representation from scratch. (2014) arXiv preprint arXiv:1411.7923
11. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1701–1708 (2014)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. (2014) arXiv preprint arXiv:1409.1556
13. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1–9 (2015)
14. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. (2015) arXiv preprint arXiv:1502.03167
15. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2818–2826 (2016a)
16. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-resnet and the impact of residual connections on learning. (2016b) arXiv preprint arXiv:1602.07261
17. Sun, Y., Wang, X., Tang, X.: Deep learning face representation from predicting 10,000 classes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1891–1898 (2014)
18. Zhang, Y., Tsang, IW., Luo, Y., Hu, CH., Lu, X., Yu, X.: Copy and paste gan: Face hallucination from shaded thumbnails. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 7355–7364 (2020)
19. Zhang, Y., Tsang, I., Luo, Y., Hu, C., Lu, X., Yu, X.: Recursive copy and paste gan: Face hallucination from shaded thumbnails. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
20. Gan, Y., Luo, Y., Yu, X., Zhang, B., Yang, Y.: Vidface: A full-transformer solver for video facehallucination with unaligned tiny snapshots. (2021) arXiv preprint arXiv:2105.14954
21. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 815–823 (2015)
22. Sun, Y., Chen, Y., Wang, X., Tang, X.: Deep learning face representation by joint identification-verification. In: Advances in Neural Information Processing Systems, pp 1988–1996 (2014)
23. Sun, Y., Wang, X., Tang, X.: Deeply learned face representations are sparse, selective, and robust. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2892–2900 (2015)
24. Wen, Y., Zhang, K., Li, Z., Qiao, Y.: A discriminative feature learning approach for deep face recognition. In: European Conference on Computer Vision, pp 499–515 (2016)
25. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: Sphereface: Deep hypersphere embedding for face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 212–220 (2017)
26. Wang, F., Cheng, J., Liu, W., Liu, H.: Additive margin softmax for face verification. IEEE Signal Process. Lett. **25**(7), 926–930 (2018)
27. Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 5265–5274 (2018b)
28. Potluri, S., Fasih, A., Vutukuru, LK., Al Machot, F., Kyamakya, K.: CNN based high performance computing for real time image processing on GPU. In: Proceedings of the Joint INDS'11 & ISTET'11, IEEE, pp 1–7 (2011)
29. Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., Wang, W.: A survey on mobile edge networks: Convergence of computing, caching and communications. IEEE Access **5**, 6757–6779 (2017)
30. Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., Young, V.: Mobile edge computing-a key technology towards 5G. ETSI White Paper **11**(11), 1–16 (2015)
31. Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B., Koldehofe, B.: Mobile fog: A programming model for large-scale applications on the internet of things. In: Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, pp 15–20 (2013)
32. Kim, S., Howe, P., Moreau, T., Alaghi, A., Ceze, L., Sathe, V.S.: Energy-efficient neural network acceleration in the presence of bit-level memory errors. IEEE Trans. Circuits Syst. I Regul. Pap. **65**(12), 4285–4298 (2018)
33. Jo, J., Kim, S., Park, I.C.: Energy-efficient convolution architecture based on rescheduled dataflow. IEEE Trans. Circuits Syst. I Regul. Pap. **65**(12), 4196–4207 (2018)
34. Kim, S., Lee, J., Kang, S., Lee, J., Yoo, H.J.: A power-efficient CNN accelerator with similar feature skipping for face recognition in mobile devices. IEEE Trans. Circuits Syst. I Regul. Pap. **67**(4), 1181–1193 (2020)
35. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp 1026–1034 (2015)
36. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4690–4699 (2019)
37. Lin, TY., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp 2980–2988 (2017)
38. Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., Zafeiriou, S.: Retinaface: Single-stage dense face localisation in the wild. (2019) arXiv preprint arXiv:1905.00641
39. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: 31st Conference on Neural Information Processing Systems (2017)
40. Chen, S., Liu, Y., Gao, X., Han, Z.: Mobilefacenets: Efficient CNNs for accurate real-time face verification on mobile devices. In: Chinese Conference on Biometric Recognition, pp 428–438 (2018)
41. Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., Gholaminejad, A., Gonzalez, J., Keutzer, K.: Shift: A zero flop, zero parameter alternative to spatial convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 9127–9135 (2018)
42. Martindez-Diaz, Y., Luevano, LS., Mendez-Vazquez, H., Nicolas-Diaz, M., Chang, L., Gonzalez-Mendoza, M.: Shufflefacenet: A lightweight face architecture for efficient and highly-accurate face recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pp 0–0 (2019)

43. Lai, L., Suda, N., Chandra, V.: Deep convolutional neural network inference with floating-point weights and fixed-point activations. (2017) arXiv preprint arXiv:1703.03073
44. Wang, M., Deng, W.: Deep visual domain adaptation: A survey. Neurocomputing **312**, 135–153 (2018)
45. Luo, Y., Liu, P., Guan, T., Yu, J., Yang, Y.: Significance-aware information bottleneck for domain adaptive semantic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 6778–6787 (2019)
46. Luo, Y., Liu, P., Zheng, L., Guan, T., Yu, J., Yang, Y.: Category-level adversarial adaptation for semantic segmentation using purified features. IEEE Transactions on Pattern Analysis and Machine Intelligence(2021)