Journal of Computers Vol. 31 No. 1, 2020, pp. 176-183
doi:10.3966/199115992020023101014

A Face Recognition System on Embedded Device

Lianfen Huang, Jia Guo, Zhibin Gao*

Department of Communication Engineering, Xiamen University, Xiamen, China
lfhuang@xmu.edu.cn, igjauimoa@163.com, gaozhibin@xmu.edu.cn

Received 13 January 2019; Revised 26 March 2019; Accepted 26 March 2019

Abstract. Face recognition algorithm has been widely used in many scenarios due to the great improvement of accuracy by using Convolutional Neuro Networks. People have achieved nearly 99.7% accuracy on face authentication task in certain dataset. However, face recognition product still faces technological and cost problem due to the conflict of probable large amount of identities and the insufficient computing power on embedded devices. To have better performance on a lager dataset, we usually train a bigger network, resulting hard implementation on smart devices. Another facing problem is called one-shot learning, we usually don't have more than one image of each identity to form face recognition database, resulting unreliable result. The main contribution in this paper is: (A) Implement a face recognition system on embedded device with specific hardware accelerator. (B) Bring up an easy method recognizing and augmenting dataset at the same time. (C) Furthermore, we divide the large database into small pieces according to districts and bring up a cloud to server local recognition system.

Keywords: embedded system, face recognition, hardware accelerator, hierarchical service system

1 Introduction

Since CNN (Convolutional Neuro Network) showed promising future on ImageNet (Alex-Net) in 2012 [1], CNN architectural has spread through CV (Computer Vision) research area, including face detection and face recognition. With more plenteous datasets and deeper network, CNNs based on statistical model soon reach valuable result, even surpass human performance [2]. The commercialization process of these technology is just as dramatically rapid as other AI (Artificial Intelligence) method. Baidu once took lead of using face recognition gate instead of RFID ones. Face verification gates are also widely used in most first-tier cities' railway stations in China to verify the passengers' identity with their ID cards. Recently, Alibaba Cloud, Tencent, JDcom detruded their New Retail concept based on face recognition payment successively.

Though face recognition has made great progress, there are still challenges. Once the amount of identities increased, the accuracy will drop exponentially. To maintain high performance, deeper networks are involved, resulting more computation complexity. Deep learning algorithms are getting higher performance, yet they are hardly used on embedded system.

In this paper, we carefully choose a hardware accelerator called Intel Movidius Neural Compute Stick [3], using concurrently pipeline computation speed up the complex computation of CNNs. The workflow of our system is shown as Fig. 1, and will be explained in Section 3. Section 2 introduces some related CNN algorithm and other hardware accelerators. Our work also includes some usable tricks like splitting large dataset into small ones according to region information, which directly reduces the difficult in each model. Also, we implement a recognition and augmentation method, which allow the system augments database during working, to solve one-shot learning problem. There is a cloud service which combines different models of different regions. Some emulation result can be found in Section 4.

* Corresponding Author

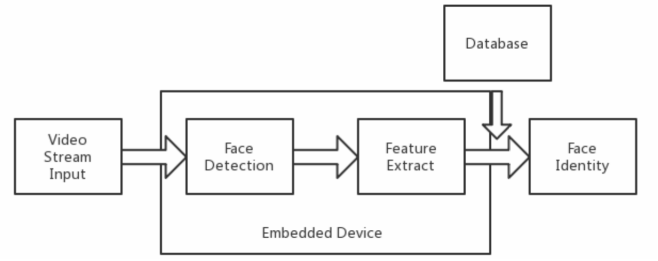

Fig. 1. Structure of the embedded face recognition system

2 Related Work

2.1 Face Detection Algorithm

Face detection technique is to detect and locate faces in images, is usually the pre-process of face recognition. Early face detection was brought up during 1990s, the popular methods then were PCA [4] and LDA [5]. In 2001, Viola and Jones came up the idea of Harr-like feature and Adaboost classification method [6], which surpassed early methods with 2 order of magnitude accuracy. Hand-made feature like Harr feature dominated the area until 2015, Cascade CNN [7] was brought up with the high performance of CNN. Later methods were all based on CNNs. In 2016 MTCNN [8] advance the performance one more step, reaching a state-of-art. Recently, Shan's group published their PCN work [9], which maintained high accuracy on askew faces and upside-down faces.

2.2 Face Recognition Algorithm

Given a database, face recognition algorithm is to identify an image of one face, and decide whether the face is in the database and which identity in the database matches the image. Early face recognition began with eigen face method [4-5]. After that, handmade features like SIFT, HOG and LBP raised [10]. These carefully designed feature were fast and easy to use. Till 2014, DeepFace [11] used CNN network reached 97.35% accuracy on LFW (Labeled Faces in the Wild) [12] dataset, a new age started. Later on 2015, Google and their Facenet [2] directly mapped images of face into Euclidean space, using distance between embedded vectors represent different of the faces. The result was 99.63%. Early in this year, Facenet model trained on VGGFace2 reached 99.7% on LFW.

2.3 Light CNN Structure

It has been a hot topic to reduce deep learning's computation since the method went red. Some of the researchers brought up light structure of CNN network like Mobilenet [13], Squeezenet [14], Shufflenet [15] and so on. They use 1*1 convolutional kernel to shrink the feature map size, so that reducing the computation complexity while maintain a higher accuracy. To achieve great performance on an embedded device, which lacks computation power, on the one hand, we use hardware accelerator. On the other hand, we choose light network architecture like Mobilenet to cut the need of computer power.

2.4 Hardware Accelerator

In order to apply face recognition technique on embedded machine, numbers of work have been tried. Most of the studies were based on the classical Harr feature [16-18], which was included in OpenCV toolkit. Some of them tried hardware accelerator like FPGAs [18] or DSPs in order to achieve real time performance. When it comes to CNNs, more compute resources are needed. Xilinx proposed a PYNQ based accelerator of CNNs [19], which had great result on LeNet and Cifar10 Network. Some institute used Movidius' hardware and SDK to accelerate CNN reference [3, 20], showing us a feasible way to implement our work.

3 System Architecture

Like Fig. 1 shows in Section 1, our system combines of different modules, such as pre-face detection, face recognition and hardware modules. In this chapter, we will talk about some detail of each module.

3.1 Face Detection Algorithm

We choose MTCNN as the face detection algorithm in our system. MTCNN is consist of three different CNN networks, which all contribute to a same purpose: To find whether there is a face and where the bounding boxes and key points of faces are. The architecture of MTCNN is shown in Fig. 2 below. By using three small CNN networks cascade together, the network reached high performance. Furthermore, cascade structure abandons most useless area at the first stage, thus reduces plenty of computations.

MTCNN is one of the real time face detection algorithm, however, when we first implemented it on our Raspberry device, it took 1.3s to execute one frame. Because of MTCNN's unique structure, it's hard to accelerate its computation over Movidius NCS (Neural Compute Stick). We eventually choose Tencent's CNN reference framework, which provide about 10 times acceleration performance on MTCNN.

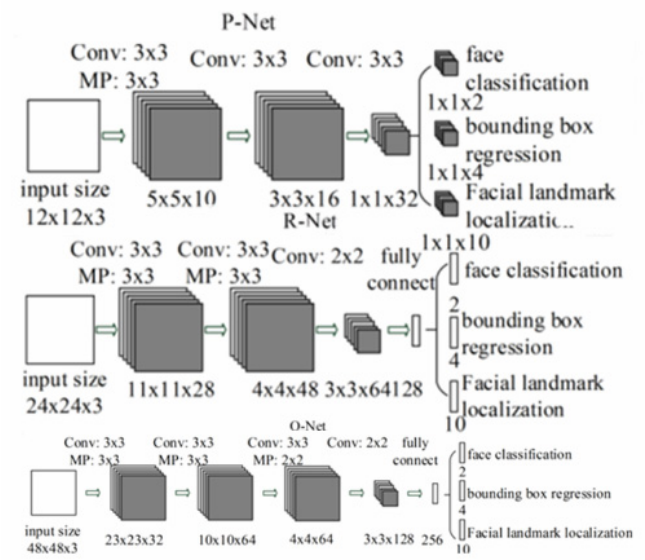

Fig. 2. MTCNN structure

3.2 Face Recognition Algorithm

Facenet is one of the most successful network of face recognition. The main idea of facenet is shown in Fig. 3. Facenet firstly combined 1:1 face verification and 1:N face recognition tasks into one network. The principle here is quite simply, a deep CNN network is used to extract abstract feature of a face image. We know CNNs can extract images' feature into vectors, which could be used for classification or other tasks. This embedded feature could be a representation of the face identity. When training a facenet model, we mainly want the representations of the same identity to be closer, and those of different identities to be further, as it's shown in Fig. 4.

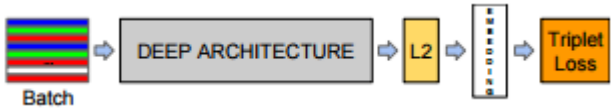

Fig. 3. Basic idea of facenet

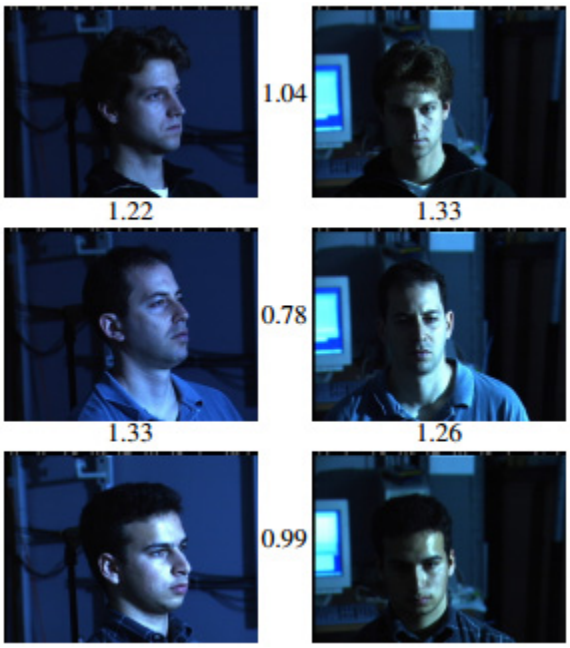

Fig. 4. Distance of images on facenet

We use triple loss to train the CNN network in facenet. The batch of facenet input is combined of different triple pairs. Each pair contains three images of two identities, those images of same person is called A (for Anchor) and P (for Positive), the other one is called N (for Negative). Then the loss function can be represented as:

$$\sum_{i}^{N}[\left\|f(x_i^a)-f(x_i^p)\right\|_2^2-\left\|f(x_i^a)-f(x_i^n)\right\|_2^2+\alpha]_+ \quad \textbf{(1)}$$

The left part of the formula is the distance between the same identity, and the right part is the distance of different identities. When minimize the loss function, the distance between A, P goes smaller and the distance of A, N goes higher. Fig. 4 also shows how this algorithm works, if we choose the threshold distance value to be 1.1, we can easily distinguish these three identities.

3.3 Hardware Framework

We select Raspberry 3B+ as the embedded system's control unit. Raspberry supports Linux OS like Rasbian, and have very convenient hardware connector like USB2.0 input and HDMI output. The framework of the hardware structure is shown as follow. Notice that we use Intel Movidius NCS as hardware accelerator to handle the computation of CNN of face recognition. Each NCS consists of serval shave that like GPU cores to GPU. The shaves allow the compute stick reference the CNN parallelly.

The software flow is shown in Fig. 5. Firstly, the ARM processor reads a frame from USB camera. The MTCNN method then goes through the frame to look whether there is a face. Here we use Cython to import the ncnn-based MTCNN library, which is 10 times faster by using MISD technique on ARM processors. If the detector finds a face, the face will be cropped and resized. The identifier is running on NCS, a SDK port is provided to communicate the NCS with the Raspberry. After process, the identifier returns the embedded vector of the face, and leave the processor to compare it with the database and finally determine the identity.

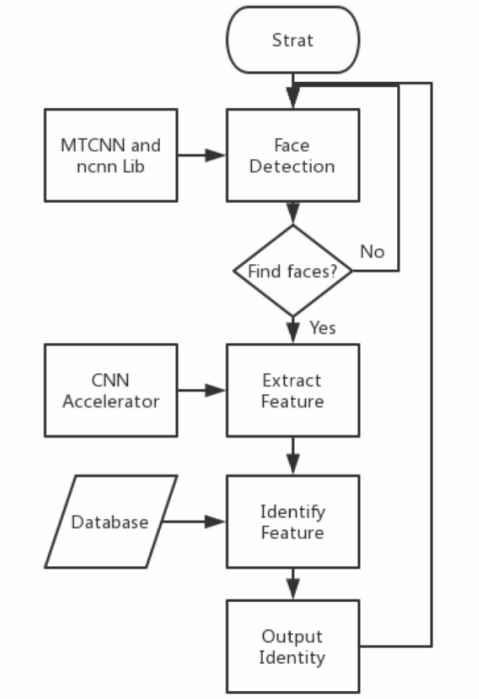

Fig. 5. Diagram of software flow

3.4 Augmentation while Recognition

Another problem of face recognition system is that we often don't have enough data to form a database. For example, if we were building an access control system for a small company, very likely we could only get one photograph for an employee. The insufficient of data would make the system less robust. To solve the problem and deployment our system agilely, we introduce a method called Augmentation while Recognizing. Like it's shown in Fig. 6, the system was based on a origin database which only contains few images. This system couldn't be too precise, so we lowed the threshold a little bit, so the system could be stricter, with high precision but low recall. After each identity was confirmed, we simply took this image mixed with the older database. After a while, the database became strong, and we could relax the condition but still get a good performance. Meanwhile, the stronger database could be used to train a new CNN network, which would fix the distribution of our client better.

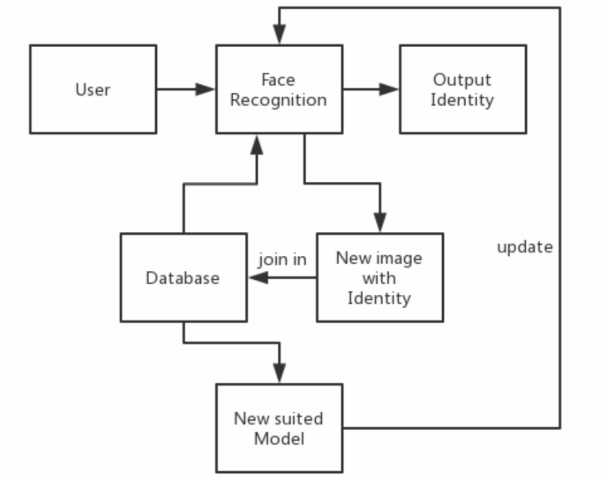

Fig. 6. Diagram of augmentation while recognition process

3.5 Database Split and Cloud Server

In some face recognition application scenarios, it's convenient to split the data base into small ones according to region attribute. For example, if we were building a payment system in a city, we could split the whole city's customers' database into pieces of different neighbors. Because of the lack of computation power, our system runs light CNN architecture which could not handle big database. Instead, a cloud platform is involved. When the customer showed in the local database, which is common for most people don't travel too far, the embedded system could recognize the person and finish the payment efficiently. Whenever the customer was not found in the local database, the image will upload and carefully examined with a bigger network and bigger dataset.

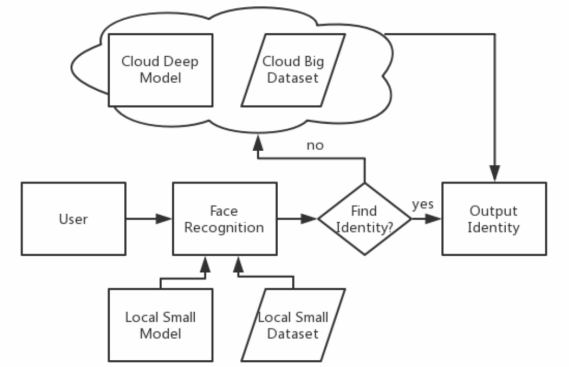

Fig. 7. Diagram of cloud server

4 Tests and Results

This chapter shows some tests and result of our embedded system, including a reference speed test, a simulation of the improvement of Augmentation while Recognition method. We use Taigman Y. et al's MTCNN as face detection model. The Facenet work base on Schroff F's work is trained on VGGFace2. There are two version of this network. The one with Mobilenet_v1 is a smart one and can be inferenced on embedded device. The other one with Inception_Resnet_v1 is a heavy model which contains over 140 layers. The heavy model outperformances the smart one, with no doubts, shown in Table 1. Notice that with the increase of amount N, the 1: N recognition accuracy drops dramatically.

Table 1. Accuracy test on LFW

Algorithms	1:1 test (verification)	1:N test (recognition)
Facenet with Mobilenet_v1	0.991	0.764
Facenet with Inception_Resnet_v1	**0.997**	**0.801**

Fig. 8 shows the hardware of the system and a simply demonstration. Table 2 explains the reference speed test of the system. We use the ncnn framework shorten the MTCNN inference time form 1.10s to 0.13s. The Mobilenet of facenet is still too heavy for ARM CPU to process, so there is not a reliable time. At last, we test the overall spend time is around 0.5s.

(a) Hardware System (b) Face Recognition Demonstration

Fig. 8. Reference speed test

Table 2. Reference speed test on raspberry

Algorithms	Raspberry3 B+ (s)	Raspberry3 B+ & NCS (s)
MTCNN	1.10	0.13
Facenet with mobienet_v1	-	0.34
MTCNN+Facenet	-	0.49

5 Conclusion

To implement face recognition system on embedded devices, the paper does the rest work: (A) Train a Facenet model with smart network architecture Mobilenet, and implement a face recognition system on Raspberry along with MTCNN face detection method. (B) Use Movidius NCS as an accelerator speed up the CNN reference and use ncnn framework fasten the MTCNN process. (C) Come up with an approach called Augmentation while Recognition which would be useful in practice. (D) By splitting up the large dataset and coming up a cloud service, we find a solution to reduce the local database so that light model could work properly.

Acknowledgements

The work was supported in part by National Natural Science Foundation of China (Grant number 61871339, 61971365), the Key Laboratory of Digital Fujian on IOT Communication, Architecture and Security Technology (No. 2010499), and Xiamen Science and Technology Plan Project (No. 3502Z20183008).

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, Curan Associates, 2012, pp. 1097-1105.

[2] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proc. the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[3] M.H. Mircea, D. Gregg, The movidius myriad architecture's potential for scientific computing, IEEE Micro 35(1)(2015) 6-14.

[4] M. Turk, A. Pentland, Eigenfaces for recognition, Journal of cognitive neuroscience 3(1)(1991) 71-86.

[5] P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman, Eigenfaces vs. fisherfaces: recognition using class specific linear projection, IEEE Transactions on Pattern Analysis & Machine Intelligence 7(1997) 711-720.

[6] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proc. the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.

[7] H. Li, Z. Lin, X. Shen, J. Brandt, A convolutional neural network cascade for face detection, in: Proc. the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[8] K. Zhang, Z. Zhang, Z. Li, Y. Qiao, Joint face detection and alignment using multitask cascaded convolutional networks, IEEE Signal Processing Letters 23(10)(2016) 1499-1503.

[9] X. Shi, S. Shan, M. Kan, S. Wu, Real-time rotation-invariant face detection with progressive calibration networks, in: Proc. the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[10] T. Ahonen, A. Hadid, M. Pietikainen, Face description with local binary patterns: application to face recognition, IEEE Transactions on Pattern Analysis & Machine Intelligence 12(2006) 2037-2041.

[11] Y. Taigman, M. Yang, M.A. Ranzato, Deepface: Closing the gap to human-level performance in face verification, in: Proc. the IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[12] G.B. Huang, M. Mattar, T. Berg, Labeled faces in the wild: a database forstudying face recognition in unconstrained environments, in: Proc. Workshop on faces in'Real-Life'Images: detection, alignment, and recognition, 2008.

[13] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, Mobilenets: efficient convolutional neural networks for mobile vision applications, https://arxiv.org/abs/1704.04861.

[14] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size, https://arxiv.org/abs/1602.07360.

[15] M.G. Hluchyj, M.J. Karol, Shuffle Net: An application of generalized perfect shuffles to multihop lightwave networks, Journal of Lightwave Technology 9(10)(1991) 1386-1397.

[16] P. Viola, M.J. Jones, Robust real-time face detection, International Journal of Computer Vision 57(2)(2004) 137-154.

[17] A. Bigdeli, C. Sim, M. Biglari-Abhari, B.C. Lovell, Face detection on embedded systems, in: Proc. International Conference on Embedded Software and Systems, 2007.

[18]S. Ghaffari, S. Sharifian, FPGA-based convolutional neural network accelerator design using high level synthesize, in: Proc. 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS), 2016.

[19] Y. Umuroglu, N.J. Fraser, G. Gambardella, Finn: A framework for fast, scalable binarized neural network inference, in: Proc. the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017.

[20] X. Xu, J. Amaro, S. Caulfield, G. Falcao, Classify 3D voxel based point-cloud using convolutional neural network on a neural compute stick, in: Proc. 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2017.