Framework Optimization for Face Recognition

Chao Chen[1], Xin Wang[1](✉), and Yong-Xin He[2]

[1] Jiangsu Vocational Institute of Architectural Technology, Xuzhou, China
920658920@qq.com, 54520081@qq.com

[2] China University of Mining and Technology, Xuzhou, China
719788101@qq.com

Abstract. In recent times, with the increasing interest in face recognition for smart homes. However, most of these studies are focused on the individual modules of such a system, and there is an evident lack of research on a face recognition system framework that can integrate and manage the entire face recognition system. Therefore, in this study, we propose a framework that enables the user to effectively develop an face recognition system in different data volume applications. This paper designs an edge computing architecture and a cloud architecture. The edge computing architecture is designed with a Centralized-Edge and Peer-to-Peer Edge. At the same time, the face screening, face disguise, training timing and dynamic adjustment of training samples in face recognition are analyzed to give a feasible solution. In particular, the face screening rules are designed to reduce unnecessary training and repeated training. This paper has important application value for the intelligentization of the Internet of Things.

Keywords: Raspberry Pi · DNN · Face recognition · Framework · OpenCV

1 Introduction

Face recognition is one of the research hotspots in computer vision, image processing and neural networks in recent years. It is widely used in the fields of public safety, verification systems and human-computer interaction [5]. The development of face recognition has mainly gone through three stages: based on structural features, based on statistical features, based on big data and complex models [19]. From the earliest methods of geometric features and template matching to a scheme based on artificial features and classifiers, face recognition technology has begun to enter the automatic machine recognition stage. In recent years, with the continuous development of deep learning technology, face recognition technology has also begun to transform from traditional machine learning methods to deep neural networks. However, the calculation of neural networks often involves a large number of matrix operations, which puts high requirements on the computing power of hardware devices [16, 18]. The emergence of powerful GPU hardware devices has greatly reduced the model calculation time, which has promoted the widespread application of face recognition technology. At present, researchers have trained complex deep neural network models with tens or even hundreds of millions of undetermined parameters, and constantly refresh the highest record of face recognition

X. Chen et al. (Eds.): ML4CS 2020, LNCS 12488, pp. 74–85, 2020.
https://doi.org/10.1007/978-3-030-62463-7_8

accuracy. The DeepFace [14] released by Facebook, the DeepID [6–8] series researched by the Chinese University of Hong Kong, and the FaceNet [12] released by Google have reached or surpassed human recognition capabilities.

The breakthrough progress of artificial intelligence has prompted facial recognition technology to integrate into people's lives, which not only promotes the construction of smart cities, but also improves the efficiency of social operations. However, the deployment of most face recognition systems relies on cloud computing resources. With the rapid growth of the number of connected devices, in order to meet the fast and accurate identification effect, high requirements are placed on the network bandwidth and computing processing capabilities of the centralized physical data center that reflects the "cloud" [11]. In addition, the openness of the network environment and the sensitivity of identity data lead to the risk of privacy leakage in practical applications of face recognition technology [17]. The cloud-based method transmits sensitive data of a large number of users to the cloud. If attacked, it is likely to cause user privacy to leak. Therefore, in terms of protecting users' private data, there is a certain degree of insecurity in cloud-based deployment [3]. As a supplement to cloud computing, in recent years, edge computing has attracted great interest from researchers. Unlike cloud computing, edge computing provides end-to-end services. Data can be processed directly on the edge device without being transferred to the cloud, so it shows excellent performance in reducing communication delay and reducing bandwidth load. Considering the superiority of edge computing, some face recognition systems based on edge computing have been proposed. He et al. [2] proposed a lightweight and fast face detector (LFFD) for edge devices. Prentice et al. [9] developed a set of Raspberry Pi-based end-to-end smart office applications. The developed solution can monitor various environmental conditions and can use facial recognition to identify users.

At present, although face recognition technology has achieved some important results, there is still a lot of work to be done to design and implement a practical face authentication system. For the research based on Raspberry Pi face recognition at home and abroad, most of them focus on the Raspberry Pi [1, 10] software and hardware optimization and face recognition algorithm optimization, applied to specific smart home scenarios, and the multi-Raspberry pi collaboration constitutes a whole architecture research. It remains to be seen, and it has important research value for architecture optimization. This paper studies the face recognition in smart family as the application scenario. For the sake of family privacy protection, this paper studies the face training on the local Raspberry Pi, and uses the Raspberry Pi as the terminal and recognition for collecting face images. Through its continuous training, the end can make it possible to identify family members in almost any dress, hairstyle and face occlusion. By comparing several face recognition system architectures designed in this paper, the face recognition performance is continuously optimized.

This paper mainly studies the following two aspects:

1. In the Raspberry Pi collection terminal, how to choose a suitable screening image method to reduce unnecessary repetitive training.
2. To meet the needs of different data volumes, design non-cloud architecture and cloud architecture for face recognition, and study and design many details.

2 Raspberry Pi System Configuration

First, we will embody the scene. In the Raspberry Pi terminal, the face image and the person's identity are matched one by one through RFID. If the face reaches the training condition, the next step of training is performed; if not, only the door opening operation is performed. In the Raspberry Pi, the basic configuration is shown below (see Fig. 1):

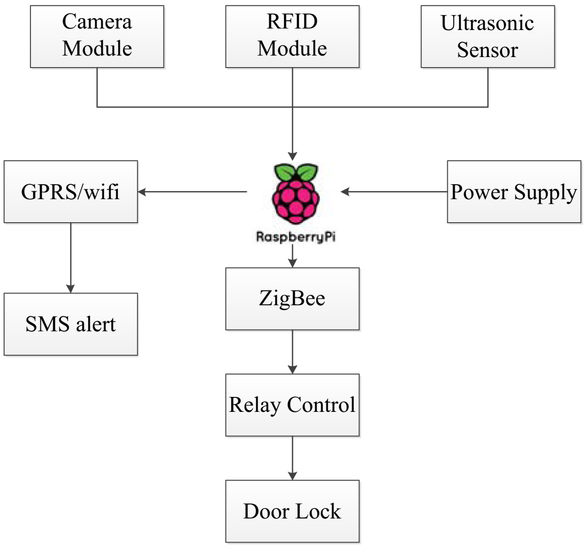

Fig. 1. Raspberry Pi configuration

2.1 Hardware Section

The Camera Module functions to capture a face image.

The role of RFID is to make the face image and the identity of the person correspond.

The Ultrasonic Sensor detects the distance of the person. When the distance is less than the preset distance, the command is sent to allow the camera to continuously capture 6 face images.

After we collect the image of the character, we send the image to the Raspberry Pi for processing. We will first screen it. The screening rules will be further explained below. If the training conditions are met, the model will be retrained.

In this architecture, the control module and communication module use the ZigBee platform. This technology is energy efficient, self-configuring, low cost, and provides high precision transmission. After we confirm the identity and determine that we can perform the operation such as opening the door, the Raspberry Pi sends a command to the relay through ZigBee, allowing the relay to control the corresponding hardware to perform the opening operation.

A whitelist is a collection of information about all legally identifiable people, and a blacklist is opposed to it. If the face image detection result is not in the white list but the RFID information is in the white list, or the existence of face disguise, the Raspberry Pi will not perform the action such as opening the door. At the same time, the Raspberry Pi sends a warning message to the administrator via GPRS or wifi, including the warning time and place.

2.2 Software Part

Face Recognition Algorithm. We installed the OpenCV platform in the Raspberry Pi and implemented the face recognition using the DNN algorithm. There are many platforms for implementing DNN, such as Caffe, TensorFlow, OpenCV, Caffe, etc. We refer to the article by Delia Velasco-Montero [15], who is based on Accuracy, Throughput, and Power Consumption.), FoM (Figure of Merit) and other aspects of the assessment. Define a variable FoM with the following formula:

$$\text{FoM} = \text{Accuracy} \cdot \frac{\text{Throughput}}{\text{Power}} \tag{1}$$

The experimental results show that OpenCV and SqueezeNet are the best performing components; for high throughput, OpenCV and TensorFlow are the best choices; for low energy budgets, OpenCV and Caffe2 are the most suitable tools. In summary, we see that OpenCV has a good performance in performance, throughput and power consumption, so this article decided to using OpenCV platform in the Raspberry Pi.

Face Screening Algorithm. When the system size and the amount of data are small, the old data and the new data can be trained together each time new face images needs training. However, as the scale of the system expands, more and more face images need to be newly identified. If the old and new data are still trained together, the training cost will be greatly improved and the training efficiency will be lower. Improve, remove some unnecessary training face images, and filter the face images. Face screening is considered in two aspects. One is to screen out face images such as unclear and blacklists to avoid wasting training time and resources; The second is to screen out the necessary training faces and not to train those faces that have been accurately identified.

Screening the face image that needs to be trained according to the preset face recognition accuracy threshold. For example, when the recognition rate is less than 50%, the image is discarded; when the recognition rate is greater than 90%, the image is not trained, but is stored; when identifying the image was trained at a rate between 50% and 90%. The specific flow chart is as follows (see Fig. 2):

In this paper, the threshold is used as the screening rule. In practical applications, when collecting human faces, we can require the examiner not to bring decorations such as hats and glasses. The increase of these rules can greatly reduce unnecessary face collection and training.

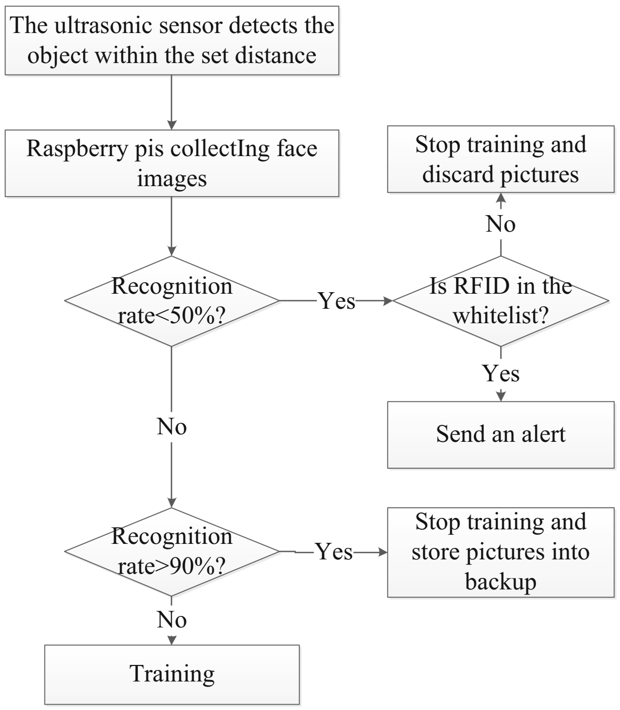
The ultrasonic sensor detects the object within the set distance
Raspberry pis collectIng face images
Recognition rate<50%?
Yes
Is RFID in the whitelist?
No
Stop training and discard pictures
Yes
Send an alert
No
Recognition rate>90%?
Yes
Stop training and store pictures into backup
No
Training

Fig. 2. Face image screening flow chart.

To Prevent Face Deception. There are many ways to prevent face deception. In the method proposed by Piyush Devikar et al., a temperature sensor based detection method is proposed because the surface temperature of the mask is close to the ambient temperature and is not as high as the real surface. At ambient temperature. The system takes images from a camera connected to the Raspberry Pi and then detects the faces in the image by OpenCV. The face temperature captured by the camera is obtained by an infrared temperature sensor. If a face is detected in the image and its temperature is greater than the threshold (skin) temperature, the face is true, otherwise it is false. This method can already block the fake face, face photo and the like formed by the high-end silicone mask.

In order to further prevent the occurrence of fraud, this article is supplemented by blink detection. We use the blink detection method of Tereza Soukupova and Jan Cech [13], using haar features to locate faces, and shape predictor 68 face landmarks.dat to mark face structures with 68 points to monitor human eyes, As shown in Fig. 3. The change in the distance between the upper and lower eyelids determines whether or not the eye is blinking. The principle is as shown below (see Fig. 4):

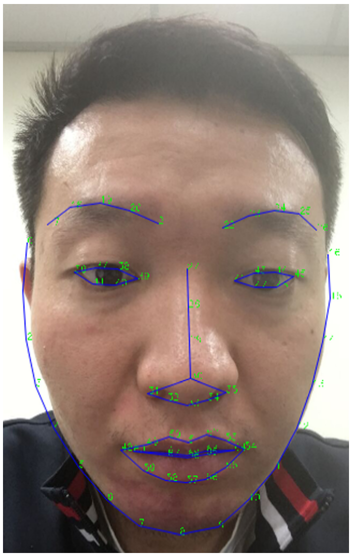

Fig. 3. Mark face structures with 68 points to monitor human eyes.

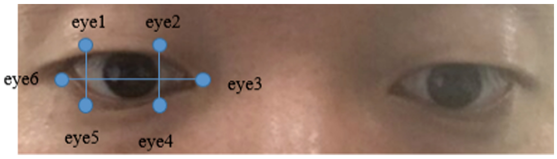

Fig. 4. Human eye mark.

As you can see in the image above, one eye will mark six points and define a variable EAR using the following formula:

$$\mathrm{EAR} = \frac{\|\mathrm{P}_2 - \mathrm{P}_6\| + \|\mathrm{P}_3 - \mathrm{P}_5\|}{2\|\mathrm{P}_1 - \mathrm{P}_4\|} \tag{2}$$

The person blinks once for about 0.2–0.4 s, and when it detects that the EAR is less than 0.3, it is considered to be blinking. For face recognition, we need to label it, including the name, identity and age, as shown below (see Fig. 5):

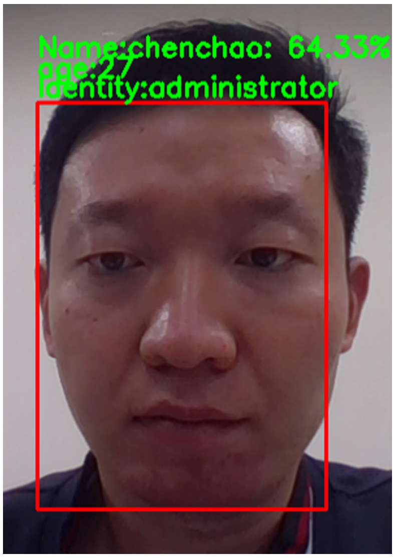

Fig. 5. Face recognition effect.

As you can see, the recognition result Name is chenchao, the accuracy rate is 64.33%, the identity is administrator, and the age is 27.

3 Edge Computing Architecture Design

First, we design the edge computing architecture [4]. The face recognition and model training are all performed on the edge. The edge can be the Raspberry Pi or the NVIDIA TX2. Keeping the data locally saves user privacy to a large extent.

Use SQLite Studio to store information in any particular sequence. This article is used to store training data, trained models, whitelists and other information, and can be displayed during the detection process. For any new user registration, you need to add his/her information to this database.

In the edge computing architecture, there are two design approaches, one is a Centralized-Edge and the other is a Peer-to-Peer Edge. Centralized-Edge we use one or several Raspberry Pi as a training server. All the face images that need to be trained are sent to this training server. After training, the model is synchronized to each Raspberry Pi. In the Peer-to-Peer Edge, each Raspberry Pi is an independent training terminal. If any Raspberry Pi is trained and the model is updated, the latest models will be synchronized to other Raspberry Pis.

The first one adopts a Centralized-Edge, as shown in the following figure (see Fig. 6):

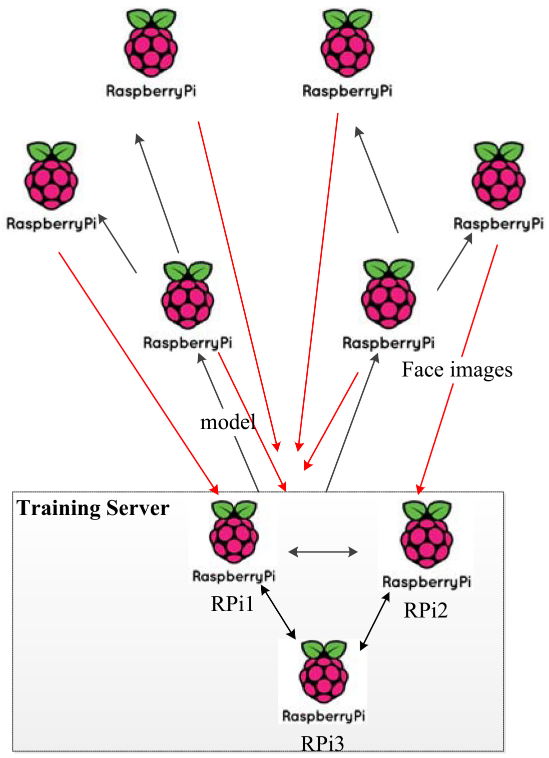

Fig. 6. Centralized architecture.

One or more Raspberry Pis are used as training servers, and each other Raspberry Pi is used as a face collection terminal. The collected faces are filtered and submitted to the Raspberry Pi server for training. After the trained, the model is distributed to each tree. Raspberry pi terminal. When the server has multiple Raspberry Pis, face recognition is performed by means of loop recognition.

Suppose now that the server consists of three Raspberry Pis. After receiving the identification request, it first asks if RPi1 can be identified. If it cannot be sent to RPi2, if RPi2 can't identify it, it will send it to RPi3. If RPi3 can recognize it, it will return recognition result. As a result, if it is not identifiable, training is performed at RPi3. At the same time, if the three Raspberry Pis can recognize different face sets different from each other, and exchange training data that cannot be recognized for training, the advantage of this is that on the one hand, the server performance can be improved, and on the other hand, the server side can be guaranteed. After a problem occurs in any Raspberry Pi, it doesn't affect the normal operation of the system.

The second adopts the Peer-to-Peer Edge. When any Raspberry Pi collects the face, it first filters and sends the filtered photos to the edge for training and updating the model. The structure is as follows (see Fig. 7):

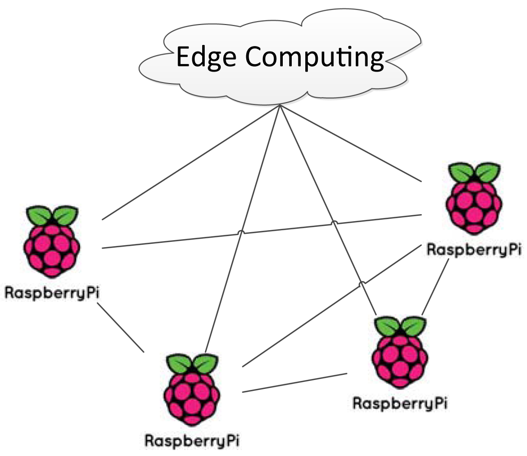

Fig. 7. Edge computing structure.

In the Raspberry Pi training, due to the limitations of its hardware performance, it can't meet the training work of a large sample. Therefore, on the Raspberry Pi training server. We can add NVIDIA TX2 or Intel Movidius Neural Computing Stick to the Raspberry Pi to improve computing performance.

4 Cloud Architecture Design

The architecture of the previous section is suitable for application scenarios where the model is small and the training data is small. As the amount of data continues to increase, the architecture of the previous section is clearly unable to meet the needs of big data processing. This chapter designs the cloud architecture and puts the training of preprocessing and model into the cloud.

The following Table 1 is a raspberry pi information table that records the id of the raspberry pi, the ip address of the Raspberry Pi, the version of the Raspberry Pi model, and the number of photos that need to be trained and the time the request was sent.

Table 1. Raspberry pi information.

ID	Time	Raspberry Pi ID	Number of images	Raspberry Pi address
1	2018-12-10-12:10:10	3203001	5	120.109.0.111
2	2018-12-10-13:20:10	3203002	6	120.109.0.112

Here is the cloud architecture diagram (see Fig. 8):

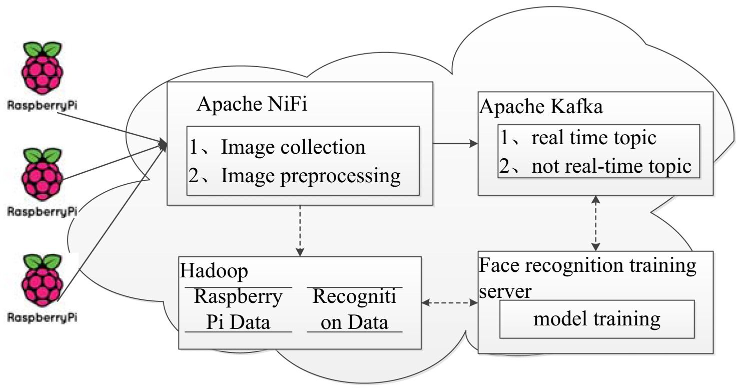

Fig. 8. Face recognition in cloud architecture.

Apache NiFi is a web project, we can use nifi to do a lot of pre-processing work, to reduce the computing burden of the cloud server. Specific to this article, we can add time and other attributes to the Raspberry Pi data, perform some pre-processing operations before the image training, for some data that needs to be stored, you can send it to Hadoop for storage.

Apache Kafka will be used as a messaging service because it provides high throughput, reliable delivery, and horizontal scalability. Kafka classifies messages according to Topic when they are saved. The sender becomes the Producer and the message receiver becomes the Consumer. In addition, the Kafka cluster consists of multiple Kafka instances, and each instance becomes a broker. Specific to this article, we can divide the data processed by nifi into two parts that need to be processed in time and not processed in time according to the training timing, and sent to different consumers for processing.

The cloud training host is responsible for training the data sent by Kafka. We can use cluster-based organization or only set up a high-performance training host. For priority training of data that needs timely training, meet the needs of real-time identification, and complete the training. Then send the model to the corresponding Raspberry Pi and let Hadoop store it.

Hadoop implements a distributed file system, referred to as HDFS. The core design of Hadoop's framework is HDFS and MapReduce. HDFS provides storage for massive amounts of data, while MapReduce provides calculations for massive amounts of data. This article is mainly used to store data such as Raspberry Pi and some trained models.

We can see that all parts of the cloud work together. First, nifi preprocesses the data sent from the Raspberry Pi, including image preprocessing, image categorization, and basic information storage. After that, nifi sends the pre-processed data to Kafka. Kafka further classifies it according to Topic, then passes the data to storm and spark for model training and recognition, and finally passes some model information to Hadoop for storage.

As time increases, the training data will become larger and larger. In order to maintain the efficiency of system training, on the one hand, we can increase the hardware configuration and improve the computing speed. On the other hand, we can reduce the training amount by reasonablely reducing the training data. We can set a threshold. When the number of training photos of a certain precision exceeds this threshold, we randomly delete the extra photos so that the training samples of a certain precision remain the same.

For the training time of cloud data, we are divided into two situations: one is that for the first time the system needs to be able to identify, the system starts training immediately after the data comes in; the other is the data that does not need timely training, such as the training accuracy is located at [50, 90] Photo, we arrange non-working hours for training.

5 Conclusions

In this study, we proposed a face recognition system framework. We designs an edge computing architecture and a cloud architecture. For each part of the architecture, such as face recognition, face screening, prevention of face disguise, and the design and connection of various components in the cloud architecture, solutions are given. The experimental results indicate that the proposed framework is effective and easy-to-use.

In the future, a face recognition system framework that is more useful can be expected if it can provide a guide that will help developer-s correct their own errors, or if it can correct errors automatically. In addition, for the face screening rule, the determination of the threshold needs further to be improved. We need further experimentation to find the most appropriate threshold. It can let us further improve efficiency and accuracy.

References

1. Gunawan, T.S., Gani, M.H.H., Rahman, F.D.A., Kartiwi, M.: Development of face recognition on raspberry pi for security enhancement of smart home system. Indonesian J. Electr. Eng. Informatics (IJEEI) **5**(4), 317–325 (2017)
2. He, Y., Xu, D., Wu, L., Jian, M., Xiang, S., Pan, C.: LFFD: a light and fast face detector for edge devices. arXiv preprint arXiv:1904.10633 (2019)
3. Li, J., Kuang, X., Lin, S., Ma, X., Tang, Y.: Privacy preservation for machine learning training and classification based on homomorphic encryption schemes. Inf. Sci. **526**, 166–179 (2020)
4. Marjanovic, M., Antonic, A., Zarko, I.P.: Edge computing architecture for mobile crowdsensing. IEEE Access **6**, 10662–10674 (2018)
5. Masi, I., Wu, Y., Hassner, T., Natarajan, P.: Deep face recognition: a survey. In: 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 471–478. IEEE (2018)
6. Ouyang, W., et al.: Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. arXiv preprint arXiv:1409.3505 (2014)
7. Ouyang, W., et al.: DeepID-Net: deformable deep convolutional neural networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2403–2412 (2015)
8. Ouyang, W., et al.: DeepID-Net: object detection with deformable part based convolutional neural networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(7), 1320–1334 (2016)

9. Prentice, C., Karakonstantis, G.: Smart office system with face detection at the edge. In: 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 88–93. IEEE (2018)
10. Sajjad, M., et al.: Raspberry pi assisted face recognition framework for enhanced law enforcement services in smart cities. Future Gener. Comput. Syst. **108**, 995–1007 (2017)
11. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)
12. 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)
13. Soukupova, T., Cech, J.: Eye blink detection using facial landmarks. In: 21st Computer Vision Winter Workshop, Rimske Toplice, Slovenia (2016)
14. 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)
15. Velasco-Montero, D., Fernandez-Berni, J., Carmona-Galan, R., RodrıguezVazquez, A.: Performance analysis of real-time DNN inference on raspberry pi. In: Real-Time Image and Video Processing 2018, vol. 10670, p. 106700F. International Society for Optics and Photonics (2018)
16. Wang, X., Kuang, X., Li, J., Li, J., Chen, X., Liu, Z.: Oblivious transfer for privacy preserving in VANET's feature matching. IEEE Trans. Intell. Transp. Syst. (2020)
17. Wang, X., Li, J., Kuang, X., Tan, Y.A., Li, J.: The security of machine learning in an adversarial setting: a survey. J. Parallel Distrib. Comput. **130**, 12–23 (2019)
18. Wang, X., Li, J., Li, J., Yan, H.: Multilevel similarity model for high-resolution remote sensing image registration. Inf. Sci. **505**, 294–305 (2019)
19. Yang, J., et al.: Neural aggregation network for video face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4362–4371 (2017)