J Supercomput (2018) 74:6729–6741
https://doi.org/10.1007/s11227-017-2198-0

Compact deep learned feature-based face recognition for Visual Internet of Things

Seon Ho Oh[1] · **Geon-Woo Kim**[1] · **Kyung-Soo Lim**[1]

Published online: 28 November 2017

Abstract The Visual Internet of Things has received much attention in recent years due to its ability to get the object location via image information of the scene, attach the visual label to the object, and then return information of scene objects to the network. In particular, face recognition is one of the most suitable means to Visual IoT because face feature is inherent label for human being. However, current state-of-the-art face recognition methods based on huge deep neural networks are difficult to apply in the embedded platform for Visual IoT due to the lack of computational resources. To solve this problem, we present compact deep neural network-based face recognition method for Visual Internet of Things. The proposed method has a low model complexity to operate in an embedded environment while using deep neural networks, which is strong against posture and illumination changes. We show competitive accuracy and performance results for the LFW verification benchmark and the collected mobile face recognition dataset. Additionally, we demonstrate that the implementation of the proposed system can be run in real time on the Android-based mobile embedded platform.

Keywords Face recognition · Deep learned feature · Visual Internet of Things · Mobile platform

✉ Kyung-Soo Lim
luke.kyungsoo@gmail.com; lukelim@etri.re.kr

Seon Ho Oh
seonho@etri.re.kr

Geon-Woo Kim
kimgw@etri.re.kr

[1] Electronics and Telecommunications Research Institute, Daejeon, Korea

1 Introduction

The "Internet of Things (IoT)" technology, which connects sensors, actuators, and processor-equipped objects together and communicates with each other to achieve meaningful goals, has become a major trend in information technology. However, the IoT system using RFID or non-contact wireless technology as their sensor cannot be applicable in some situation. Besides, the cost of RFID label should be taken into consideration when there are huge amounts of objects. To overcome these limitations, Visual IoT, a technique for extracting and using scene object information directly from images, has been proposed. With the help of cameras, the Visual IoT can get the object location via image information of the scene, attach the visual label to the object, and then return information of scene objects to the network [14].

As personal security becomes a critical issue, biometric recognition systems for self-authentication based on face or voice have received great attention in recent decades. In particular, face recognition is one of the most popular techniques because it is less repulsive and user-friendly due to its non-contact, non-aggressive, and non-intrusive nature [12]. In addition, since facial features are unique to humans, facial recognition is one of the most appropriate tools for Visual IoT. Despite significant recent advances in the field of face recognition, however, implementing face verification and recognition to the Visual IoT efficiently at scale is very challenging due to limited computing power. Moreover, existing face recognition methods applicable to Visual IoT only worked well in limited environments such as constrained illumination condition and approximately frontal posture.

Deep convolutional neural network-based methods have recently achieved significant improvements in face recognition. For example, Facebook's DeepFace [22] and Google's FaceNet [20] have achieved human-level face recognition performance. However, face recognition techniques based on a huge deep neural network are not applicable in IoT environment due to such devices' limited processing and storage capabilities.

The goal of this work is the development of a face recognition system for Visual IoT using deep neural network. Our main consideration is the development of a practical system that can give high accuracy and real-time performance using deep neural network in embedded environment for Visual IoT.

The paper is structured as follows: Sect. 2 briefly overviews previous studies on face recognition in general and embedded environments. Section 3 focuses on our design consideration and implementation details. Section 4 describes the collection, analysis of dataset, and preprocessing steps for training and evaluation. Our accuracy and performance are evaluated and compared to other face recognition techniques in Sect. 5, and finally, Sect. 6 summarizes this work.

2 Related work

Face recognition has been an active research topic in recent decades. Given an input image with one or more faces, a typical face recognition system consists of four stages: face detection, preprocessing, representation, and matching. The face detection

stage isolates the faces and gives a list of bounding boxes. The preprocessing stage is required because the face is not a rigid object and images of the face can be taken from many different perspectives. The preprocessing stage normalizes the faces so that the eyes, nose, and mouth appear at similar locations. The representation stage translates the preprocessed face image into a low-dimensional representation (or embedding). Finally, the matching stage identifies or verifies enrolled users.

Numerous studies have been proposed in the literature. Among the huge number of proposed methods, we distinguish the ones prior to deep learning, which we call "non-deep," from those that do, which we call "deep." Non-deep methods can be categorized into a local feature-based and holistic approaches. Local feature-based approaches first extract hand-crafted local image descriptors such as SIFT, LBP, HOG and then aggregate them into an overall face descriptor [7,8], whereas holistic approaches are based on statistics such as principal component analysis (PCA) and independent component analysis (ICA), which represent faces as a combination of eigenvectors or features that characterize or separate two or more classes. Eigenfaces [23] and FisherFaces [5] are the most well-known techniques in PCA-based methods. Jafri and Arabnia [12] provided a nice survey of the face recognition methods developed up to 2009.

Deep methods are based on convolutional neural networks. Facebook's DeepFace [22] and Google's FaceNet [20] achieved the highest accuracy in the LFW dataset [17], which is a standard benchmark in face recognition research. VGG Face Descriptor [18] and Lightened Convolutional Neural Networks (CNNs) [25] achieved comparable performance.

In mobile embedded environments, studies often use techniques with an order of magnitude less accuracy than state-of-the-art approach due to the lack of computational resources. Soyata et al. [21] proposed an Eigenfaces-based face recognition system that uses a mobile device, cloudlet, and cloud. Hsu et al. [10] introduced a cloud-based face recognition service for drones. Ye et al. [26] presented a face authentication system on a distributed computing environment. The advance of efficient GPU architecture for mobile devices such as NVIDIA's Tegra has accelerated computational performance [24]. However, it remains incapable of executing recent top-performing deep neural network-based face recognition techniques.

3 Methodology

3.1 Deep learning framework selection

In this section, the compatibility of current deep learning frameworks will be investigated. It will not compare other features or the performance, as this is beyond the scope of this paper and already good benchmarks exist [4].

Caffe [13] was the first mainstream industry-grade deep learning framework developed by the Berkeley Vision and Learning Center and by community contributors. It remains the most popular toolkit within the computer vision community. It has command line, Python, and MATLAB interfaces. Since it was developed in C++, it can be complied on various platforms; recent unofficial ports support mobile platforms such as Android and iOS.

Torch [6] is a scientific computing framework that supports a MATLAB-like environment built on Lua; it can provide C++ and Lua interfaces. Recent unofficial ports have supported running on mobile platforms such as iOS and Android.

Theano [2] is a symbolic expression compiler that efficiently defines, optimizes, and evaluates mathematical expressions that involve multi-dimensional arrays. It only supports a Python interface, and support for other embedded platforms is unfortunately not yet considered a core feature.

TensorFlow [1] is an open-source framework for numerical computation using data flow graphs. It allows the deployment of computation to one or more CPUs or GPUs in a desktop, server, or mobile device environment with a single API, and it has APIs available in several languages such as Python, C++, Java, and Go. It is the only framework that officially supports mobile embedded environments such as Android and iOS at the time of writing this paper.

The Microsoft Cognitive Toolkit (CNTK) [27] is a unified deep learning toolkit that was recently released by Microsoft Research. It provides Python, C++, C#, and BrainScript interfaces. However, it does not yet support a mobile embedded platform.

As the development shows, the addition of support to run deep neural networks in mobile embedded environments is rising. However, many do not officially support embedded devices at the time of writing this paper. Thus, we choose TensorFlow as our main deep learning framework. The same model can be run on a dedicated server or embedded device without any code changes, and it can be easily ported to other IoT environments because it is implemented in C++. Additionally, TensorFlow has a fast-growing community of users and contributors, making it the most promising deep learning framework.

3.2 Face recognition pipeline

Our face recognition pipeline consists of four stages: face detection, face alignment (or normalization), feature extraction, and recognition. The face detection stage gives a list of bounding boxes around the detected face. The face alignment stage normalizes faces with respect to geometric properties so that the eyes, nose, and mouth appear at similar locations. The feature extraction stage extracts the facial features that represent a certain aspect of a detected face. Finally, the recognition stage verifies enrolled users. Figure 1 shows our face recognition pipeline.

3.3 Face detection and preprocessing

The face detection stage gives a list of bounding boxes around the face. We implement a histogram of the oriented gradient (HOG)-based face detector with a structured support vector machine (SVM), and the different pose issue is handled by the alignment stage, which normalizes the faces so that the eyes, nose, and mouth appear at similar locations. Many modern techniques such as DeepFace [9,22] frontalize the face to a 3D model so that the image shows the face looking directly at the camera. Their computational complexity makes these 3D approaches unsuitable for mobile environments. Moreover, face posture variation in the mobile face recognition scenario is

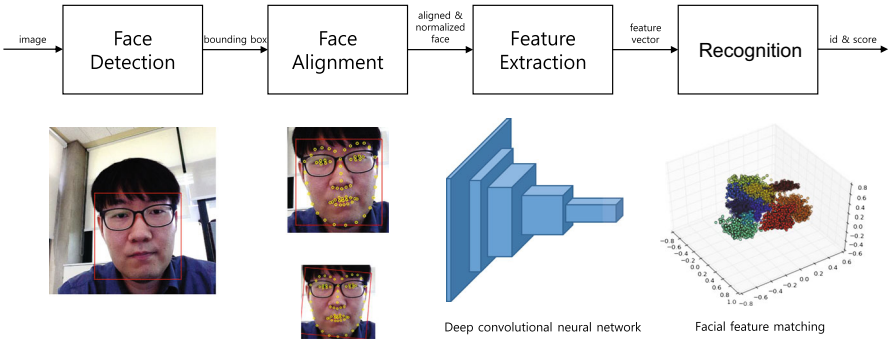

Fig. 1 Face recognition pipeline

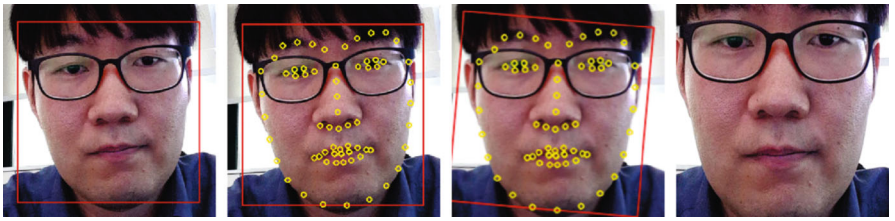

Fig. 2 Face detection and preprocessing

limited to the device's viewing angle. Therefore, we align faces using a simple 2D affine transformation based on facial landmarks [15]. This is less accurate than the 3D methods, but provides a reasonable performance in limited mobile scenarios. Then, we crop the aligned face and resize it into 96 × 96 pixels. Figure 2 shows details of the preprocessing stage. The red rectangle illustrates the detected face, and yellow dots denote the facial landmark points for alignment.

3.4 Network architecture and training

Using a deep neural network on a mobile embedded device requires considering the number of parameters for the network, because it is tightly coupled with the total number operations and memory usage. For example, DeepFace [22] has 120M and VGG Face [18] requires 138M parameters, taking up more than 500MB storage, and needs 15.47 billion floating-point operations (FLOPs); however, this is unacceptable for a mobile environment. Thus, we need a more compact but powerful network model. We choose nn4.small2 [3] for mobile environments, which is a variation of the **nn4** model from FaceNet [20] that has fewer parameters for mobile execution. The network model for compact deep learned feature is shown in Table 1. Each row indicates a layer in the deep neural network. The total number of parameters for our model including batch normalization is 3.74 million and needs 208.16 million FLOPs. It requires only 14.28 MB of memory with single-precision floating-point format.

The convolution layer is the core building block of deep convolutional neural networks. A set of filters in each convolution layer produces activation maps that are the

responses to some types of visual feature such as an edge or color in the lower layers, or some complex pattern in the network's higher layers. The stack of these activation maps along the depth dimension is passed to the next layer.

The max pooling layer between successive convolution layers reduces the spatial size of the feature map along both the width and height. It discards 75% of responses, which reduces the number of parameters and computation in the network. Similarly, the average pooling layer downsamples every depth slice in the feature map by a factor of 3 via average operation. The "pool proj." column of the inception layers in Table 1 describes the pooling type, the kernel size, and stride.

The local response normalization (LRN) layer diminishes responses that are uniformly large for their neighborhood and make large activations more pronounced within a neighborhood. In other words, the LRN layer regularizes the responses obtained by different kernels. The LRN layer is used both before and after inception (2).

The inception layers [20] performed cross-channel correlations while ignoring spatial dimensions through a 1×1 convolution; this dramatically reduced the dimensionality in the filter dimension. Then, cross-spatial and cross-channel correlations were conducted via 3×3 and 5×5 kernels. Concatenating the responses from convolutional filters with different sizes covers different clusters of information. In addition, two types of pooling operation, max and L_2, were used to reduce the dimensionality prior to convolutions, which allowed both deeper and larger convolutional layers and more efficient computation. Our network uses four types of inception layer with small variation. The last seven columns describe the parameters of the inception layers from [20] and the number of parameters for each layer. The columns starting with "#N $\times$ N" denote the depth of the output feature map, and "#3 $\times$ 3 reduce" and "#5 $\times$ 5 reduce" represent the number of 1×1 filters that were used in the reduction layer before 3×3 and 5×5 convolutions, and "pool proj." describes pooling type and the size of the dimensions into which it is reduced.

The embedding layer is a composition of the fully connected layer and the L_2 normalization layer. A fully connected layer linearly combines $1 \times 1 \times 736$ feature maps into 128-dimensional vector. Then, the following L_2 normalization layer constrains the vector to the unit hypersphere.

A network is trained with 500 K images from two of the largest public face recognition datasets, CASIA-WebFace and FaceScrub. Triplet loss [20] is used to provide embedding on the unit hypersphere and to effectively represent the similarity between faces.

4 Dataset

4.1 Dataset acquisition and analysis

We have collected a face dataset with which to evaluate the performance in the mobile face recognition scenario and embedded environment. In addition, the distribution of face postures is analyzed to understand the mobile face recognition scenario and develop a non-self-aware face recognition system on mobile.

Table 1 Details of the network model for the compact deep learned feature

Type	Output size	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	Pool proj.	Params
Conv1 ($7 \times 7 \times 3$, 2)	$48 \times 48 \times 64$							9 K
Max pool + norm	$24 \times 24 \times 64$						m 3×3, 2	
Inception (2)	$24 \times 24 \times 192$		64	192				115 K
Norm + pool	$12 \times 12 \times 192$						m 3×3, 2	
Inception (3a)	$12 \times 12 \times 256$	64	96	128	16	32	m, 32p	164 K
Inception (3b)	$12 \times 12 \times 320$	64	96	128	32	64	L_2, 64p	228 K
Inception (3c)	$6 \times 6 \times 640$		128	256, 2	32	64, 2	m 3×3, 2	398 K
Inception (4a)	$6 \times 6 \times 640$	256	96	192	32	64	L_2, 128p	546 K
Inception (4e)	$3 \times 3 \times 1024$		160	256, 2	64	128, 2	m 3×3, 2	717 K
Inception (5a)	$3 \times 3 \times 736$	256	96	384			L_2, 96p	791 K
Inception (5b)	$3 \times 3 \times 736$	256	96	384			m, 96p	662 K
Avg. pool	$1 \times 1 \times 736$							
Embedding	128							94 K
Total								3.7 M

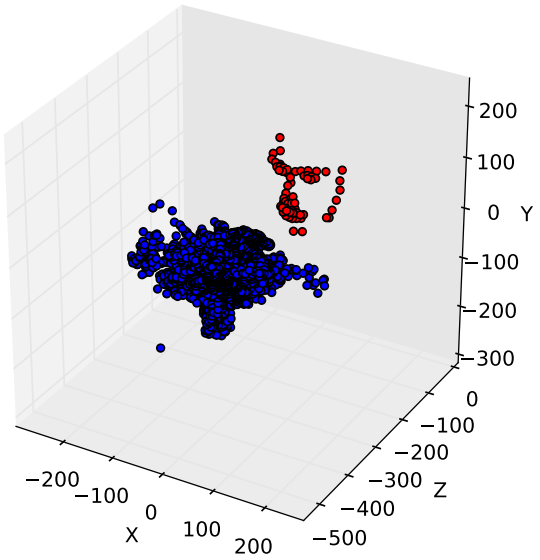

Fig. 3 3D camera location with respect to face

The dataset was captured using a Samsung Galaxy S6, and the video capture resolution is 720×960. The dataset consists of distinct sessions that were usually separated by one or two weeks, and the data were captured over two months. In total, 10,360 images from 10 identities are captured.

Capturing the dataset on a mobile device is inherently uncontrollable because the devices are given to users. Specifically, capturing data from a mobile device allows high variability in face poses and illumination conditions. Ensuring that the captured data are meaningful and useful requires enforcing minimal constraints upon participants and validating the captured data.

pcTwo constraints were placed upon users when recording their data; we asked that they ensure that they were able to read the text on the screen and that most of their face was in the captured image. We provided simple random text and live video feedback to assist with this. Additionally, we asked that the user be seated in an indoor office environment. In addition to these constraints, we validated the captured images.

Figure 3 visualizes estimated poses from images. Red dots represent facial landmarks, and blue dots represent the estimated camera location. Interestingly, the horizontal and vertical face rotation of the collected dataset follows a Gaussian distribution centered at (0, − 5) degrees. The rotation ranges are, respectively, − 30 to 30 ° and − 20 to 10 ° for horizontal and vertical directions. Figure 4 shows the density of vertical and horizontal face rotations in the mobile face recognition dataset.

4.2 Dataset preprocessing

Face detection and alignment are conducted on the collected mobile face recognition dataset. Faces are detected using a HOG-based detector as described in Sect. 3.3, but

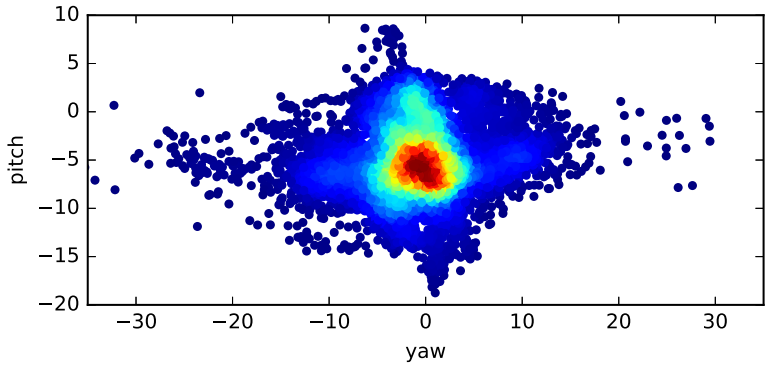

Fig. 4 Rotational distribution of the mobile face recognition dataset

Fig. 5 Example images in the mobile face recognition dataset

some of the collected faces were blurry because of the motion of users and/or camera focus. We use a simple focus measure to filter out blurred faces; if the detected face has a focus measure response of less than a certain threshold τ_f, it is discarded. For the sake of simplicity, the variance of Laplacian (LAPV) [19] is used as focus measure. Face alignment is performed after blurry faces are filtered. Consequently, 9798 faces from 10,360 images are successfully aligned. Figure 5 shows example faces in our mobile face recognition dataset.

5 Evaluation

In the following, we first test the proposed system using the LFW verification dataset to evaluate its accuracy in comparison with other techniques. Then, we evaluate the accuracy on the mobile face recognition dataset and the performance on mobile embedded devices.

Table 2 LFW verification accuracies

Technique	Accuracy
Human-level (cropped) [16]	0.9753
Eigenfaces (no outside data) [23]	0.6002 ± 0.0079
FaceNet [20]	0.9964 ± 0.009
DeepFace-ensemble [22]	0.9735 ± 0.0025
Ours	0.9331 ± 0.0015

5.1 LFW verification

The LFW dataset consists of 13,233 images from 5750 people; the verification test provides 6000 pairs separated into ten equally sized fold. The LFW verification test [11] predicts whether given pairs of images are of the same person.

Table 2 shows the LFW verification accuracies of other techniques. The accuracy is obtained by tenfold cross validation; the ninefolds are used for training the threshold and the remaining onefold is used for testing, and the testing is done 10 times.

The pair is labeled as the sample person if the Euclidean distance on the pair is less than a certain threshold τ_d; otherwise, it is labeled as different people. The best threshold of the training fold is used as the threshold on the remaining fold. In nine out of ten experiments, the best threshold was 1.01. These results demonstrate that our accuracy is close to the accuracy of state-of-the-art deep learning-based techniques.

5.2 Mobile face recognition dataset

The recognition accuracy on a mobile face recognition dataset using 20 samples per person is 98.88%. The false accept rate (FAR) and false rejection rate (FRR) were 1.03 and 1.25%, respectively.

Figure 6 shows the accuracy variation with respect to the training sample size. As can be seen in Fig. 6, the accuracy rapidly increased in up to 10 samples, and the accuracy slowly increased with more than 20 samples. Thus, we used 20 samples for the enrollment stage of the mobile implementation for efficiency and usability.

Table 3 shows the comparison accuracy by classifiers. As can be seen in Table 3, the nearest neighbor with Euclidean distance achieves comparable performance to the others. For the sake of simplicity and ease of implementation on a mobile platform, we use the nearest neighbor classifier as our baseline classifier.

5.3 Device performance

The evaluation was conducted on a Samsung Galaxy S6 (1.5 GHz octa-core CPU, 3 GB RAM) running Android 6.0.1. The camera resolution was 480×320 pixels, and a cropped 320×320 pixel square was used as the initial input. Table 4 shows the execution time for all tasks. The total execution time on the device including all tasks

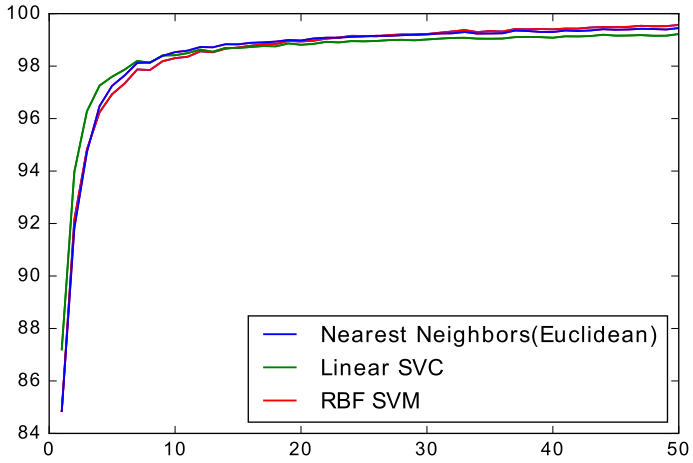

Fig. 6 Accuracy variation with respect to the training sample size

Table 3 Comparison of face recognition by classifier

Classifier	Accuracy	FAR	FRR
Nearest neighbor (Euclidean dist.)	0.9888	0.0103	0.0125
Linear SVC	0.9882	0.0104	0.00127
RBF SVM	0.9899	0.0087	0.0105

Table 4 Comparison of execution time by device

Devices	Tasks			Total
	Detection + Preproc.	Feature ext.	Authentication	
Samsung Galaxy S6 (ms)	76	200	< 1	277
LG G5 (ms)	41	82	< 1	124

was 277 ms (76 ms for the first two tasks, 200 ms for feature extraction, and 1 ms for authentication), i.e., 3.6 fps.

6 Conclusion

This paper proposes a face recognition system for Visual IoT using compact deep learned feature. A compact deep neural network is adopted to enable execution in mobile embedded environments while achieving higher accuracy. We show competitive accuracy and performance results on the LFW verification benchmark; furthermore, we show promising results using a mobile face recognition dataset and the proposed system runs in real time in a mobile embedded environment despite using deep neural network. Additionally, the face posture in the mobile face recognition sce-

nario was analyzed. In future, more efficient and compact deep neural network-based face recognition methods for Visual IoT can also be investigated.

Acknowledgements This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea Government (MSIT) (No. 2015-0-00168, Development of Universal Authentication Platform Technology with Context-Aware Multi-Factor Authentication and Digital Signature and No. 2016-0-00109, Development of Video Crowd Sourcing Technology for Citizen Participating-Social Safety Services).

References

1. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. http://tensorflow.org/
2. Al-Rfou R, Alain G, Almahairi A, Angermueller C, Bahdanau D, Ballas N, Bastien F, Bayer J, Belikov A, Belopolsky A, Bengio Y, Bergeron A, Bergstra J, Bisson V, Bleecher Snyder J, Bouchard N, Boulanger-Lewandowski N, Bouthillier X, de Brébisson A, Breuleux O, Carrier PL, Cho K, Chorowski J, Christiano P, Cooijmans T, Côté MA, Côté M, Courville A, Dauphin YN, Delalleau O, Demouth J, Desjardins G, Dieleman S, Dinh L, Ducoffe M, Dumoulin V, Ebrahimi Kahou S, Erhan D, Fan Z, Firat O, Germain M, Glorot X, Goodfellow I, Graham M, Gulcehre C, Hamel P, Harlouchet I, Heng JP, Hidasi B, Honari S, Jain A, Jean S, Jia K, Korobov M, Kulkarni V, Lamb A, Lamblin P, Larsen E, Laurent C, Lee S, Lefrancois S, Lemieux S, Léonard N, Lin Z, Livezey JA, Lorenz C, Lowin J, Ma Q, Manzagol PA, Mastropietro O, McGibbon RT, Memisevic R, van Merriënboer B, Michalski V, Mirza M, Orlandi A, Pal C, Pascanu R, Pezeshki M, Raffel C, Renshaw D, Rocklin M, Romero A, Roth M, Sadowski P, Salvatier J, Savard F, Schlüter J, Schulman J, Schwartz G, Serban IV, Serdyuk D, Shabanian S, Simon E, Spieckermann S, Subramanyam SR, Sygnowski J, Tanguay J, van Tulder G, Turian J, Urban S, Vincent P, Visin F, de Vries H, Warde-Farley D, Webb DJ, Willson M, Xu K, Xue L, Yao L, Zhang S, Zhang Y (2016) Theano: A Python framework for fast computation of mathematical expressions. arXiv:1605.02688
3. Amos B, Ludwiczuk B, Satyanarayanan M (2016) Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science
4. Bahrampour S, Ramakrishnan N, Schott L, Shah M (2015) Comparative study of deep learning software frameworks. arXiv:1511.06435
5. Belhumeur PN, Hespanha JP, Kriegman DJ (1997) Eigenfaces vs. fisherfaces: recognition using class specific linear projection. IEEE Trans Pattern Anal Mach Intell 19(7):711–720. https://doi.org/10.1109/34.598228
6. Collobert R, Kavukcuoglu K, Farabet C (2011) Torch7: A matlab-like environment for machine learning. In: BigLearn, NIPS Workshop, EPFL-CONF-192376
7. Gao Y, Lee HJ (2015) Viewpoint unconstrained face recognition based on affine local descriptors and probabilistic similarity. J Inf Process Syst 11(4):643–654
8. Han S, Lee IY, Ahn JH (2016) Two-dimensional joint bayesian method for face verification. J Inf Process Syst 12(3):381–391
9. Hassner T, Harel S, Paz E, Enbar R (2015) Effective face frontalization in unconstrained images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4295–4304
10. Hsu HJ, Chen KT (2015) Face recognition on drones: Issues and limitations. In: Proceedings of the 1st Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use, ACM, New York, NY, USA, DroNet '15, pp 39–44, https://doi.org/10.1145/2750675.2750679
11. Huang GB, Ramesh M, Berg T, Learned-Miller E (2007) Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Technical report 07-49, University of Massachusetts, Amherst

12. Jafri R, Arabnia HR (2009) A survey of face recognition techniques. J Inf Process Syst 5(2):41–68. https://doi.org/10.3745/JIPS.2009.5.2.041
13. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: Convolutional architecture for fast feature embedding. arXiv:1408.5093
14. Jiang Z, Lin Y, Li S (2013) Accelerating face recognition for large data applications in visual internet of things. Inf Technol J 12:1143–1151
15. Kazemi V, Sullivan J (2014) One millisecond face alignment with an ensemble of regression trees. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1867–1874
16. Kumar N, Berg AC, Belhumeur PN, Nayar SK (2009) Attribute and simile classifiers for face verification. In: IEEE 12th International Conference on Computer Vision, 2009, pp 365–372, https://doi.org/10.1109/ICCV.2009.5459250
17. Learned-Miller GBHE (2014) Labeled faces in the wild: updates and new reporting procedures. Technical report UM-CS-2014-003, University of Massachusetts, Amherst
18. Parkhi OM, Vedaldi A, Zisserman A (2015) Deep face recognition. In: Proceedings of the British Machine Vision, vol 1
19. Pech-Pacheco JL, Cristóbal G, Chamorro-Martinez J, Fernández-Valdivia J (2000) Diatom autofocusing in brightfield microscopy: a comparative study. In: Proceedings of 15th International Conference on Pattern Recognition, vol 3, pp 314–317
20. Schroff F, Kalenichenko D, Philbin J (2015) Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 815–823
21. Soyata T, Muraleedharan R, Funai C, Kwon M, Heinzelman W (2012) Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In: IEEE Symposium on Computers and Communications (ISCC), 2012, pp 59–66, https://doi.org/10.1109/ISCC.2012.6249269
22. Taigman Y, Yang M, Ranzato M, Wolf L (2014) Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1701–1708
23. Turk MA, Pentland AP (1991) Face recognition using eigenfaces. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991, pp 586–591, https://doi.org/10.1109/CVPR.1991.139758
24. Wang YC, Cheng KT (2011) Energy-optimized mapping of application to smartphone platform—a case study of mobile face recognition. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp 84–89, https://doi.org/10.1109/CVPRW.2011.5981820
25. Wu X, He R, Sun Z, Tan T (2015) A light cnn for deep face representation with noisy labels. arXiv:1511.02683
26. Ye P, Yu M, Wu M (2015) Implementation: mobile face identity authentication system on android platforms. Int J Secur Appl 9:51–60
27. Yu D, Eversole A, Seltzer M, Yao K, Kuchaiev O, Zhang Y, Seide F, Huang Z, Guenter B, Wang H, Droppo J, Zweig G, Rossbach C, Gao J, Stolcke A, Currey J, Slaney M, Chen G, Agarwal A, Basoglu C, Padmilac M, Kamenev A, Ivanov V, Cypher S, Parthasarathi H, Mitra B, Peng B, Huang X (2014) An introduction to computational networks and the computational network toolkit. Technical report, https://www.microsoft.com/en-us/research/product/cognitive-toolkit/