# Real-time DNN-based Face Identification for the Blind

Jhilik Bhattacharya[1], Francesco Guzzi[2], Stefano Marsi[2], Sergio Carrato[2], and Giovanni Ramponi[2]*

[1] Thapar University, Patiala, India
[2] University of Trieste, Trieste, Italy

**Abstract.** We present some results from an ongoing project about face detection and recognition in an apparatus usable by a visually impaired person. Specifically, we explore the usable equipment and we experiment on the realization of three prototypes that give the opportunity of dealing with different topics, ranging from the architecture of the network to database creation, from the reliability of the identification results to real-time operation issues.

**Keywords:** Face identification, convolutional neural networks, visual impairments, persons with disabilities.

## 1 Introduction

We present in this paper some results from an ongoing project that studies face detection/recognition methods to build an apparatus usable by a visually impaired person.

Notwithstanding the huge progress of computer vision tools, especially since the advent of Deep Neural Networks (DNNs), techniques for face recognition that may help blind users in their relational life have not yet found practical usage. The layman perceives that this problem has been already solved, since many tools have been recently made available in the consumer world (especially in social networks) to determine the identity of people. However, the reliability that is required for an apparatus to be used by persons with disabilities (PwD) is the same as the one required by a professional user (say, a bank), while its cost must be much lower: the reliability/cost ratio is far from satisfactory presently.

A peculiarity of the system we are realizing, which must be usable by a blind person, is the availability of a limited number of faces for the training of the network [1]. For this reason, we decided to fine-tune an existing DNN rather than train one from scratch. The same acquisition hardware will be used in the two different operating phases: during the implementation of the user's friends

* This work is supported by the University of Trieste - Finanziamento di Ateneo per progetti di ricerca scientica - FRA 2016, and by a private donation in memory of Angelo Soranzo (1939-2012). The authors also thank Eugenio Culurciello and Alfredo Canziani for kindly providing the basenet, and Marko Vitez for the *thnets* library.

database (recording a video of the friend) and during the so called inference (i.e. the operative) phase of the device.

Section II describes different approaches that can be followed for fine-tuning; we provide detailed information about the involved procedures and the possibility of using single or multiple networks. Section III discusses three different realizations of the system, aiming at achieving real-time operation. Some quantitative data about the effectiveness and reliability of the identification is provided in Section IV.

## 2 Face Verification Approaches

The scenario we consider in this phase of the project is the one of a blind person who has planned to meet a friend at a predefined spot and wishes to be able to recognize him as early as possible during the approach. It can be cast as a 1-to-1 face *verification* task: the equipment has to acquire the scene, detect faces, compare each detected face to the faces representing the class of the specific friend the user has to meet, and inform the user if a positive answer results. Information about the reliability of the verification may be provided too, e.g. as a percent number. Before the meeting the user must activate a classifier dedicated to that specific friend. However, the scenario can also be cast in a different way, as a 1-to-$N$ *open recognition* task: each detected face is compared to $N$ classes representing $N$ possible friends of the user's, and a single (but more complex) classifier is adopted; the recognition is deemed *open* because the system must be able to also label the face as not belonging at all to the set of friends [2]. In this 1-to-$N$ context, too, some information may be provided to the user about the reliability of the identification; moreover, a suitable threshold will be needed to label out-of-the-set faces. We will show in the following some preliminary results for both the 1-to-1 and the 1-to-$N$ approaches.

We take three different approaches towards handling the identification scenario, utilizing a feature extractor and distance classifier, a single-class verification network, and a multi-class verification network with class criteria threshold. In all the cases we start from a basic deep convolutional network, a variation of FaceNet [3], henceforth referred in this paper as the *basenet*. The face detection results are hence directly fed to the various finetuned version of the basenet developed for this work. For each of the cases further elaborated, the same groundtruth was used.

The basenet is a feature extraction network with a length-128 feature vector output. The duty of the network is to *embed* in the feature space faces that belong to the same individual. Thus, using the basenet as a pure feature extractor and adopting a distance classifier with a suitable threshold approach, the test feature vector $x$ is matched with the ground truth data to verify the face. The number of classes activated for the ground truth comparison depends on how many faces the user wants to identify. In order to get the minimal false positives we select an optimal threshold (on the network classifier output) determined using an ROC curve. We compare the result with the mean vector $\boldsymbol{\mu}$ of each class and we define the acceptance ratio of the result thresholding the absolute distance above the standard deviation $\boldsymbol{\sigma}$. Indeed, it has been proved that $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are capable of tackling the pose and the lighting variations.

For the single-class approach, we finetune the basenet to build a 1-to-1 face verification system. In this case five separate networks for the five subjects are created by adding a single classifier to the basenet. The user in this case will be required to load the network corresponding to the person he or she wants to meet. The network will return a verification confidence of each face it sees. It is left to the user to accept a low confidence verification or to wait for a high confidence one, and in general to control the flow of information the equipment provides. Finally,in the multi class verification, we finetune the basenet to build a 1-to-5 face verification system. In this case a five-class classifier is added to the basenet. The network will return the confidence value of each of the five subjects, for each face it sees.

## 3 System Details

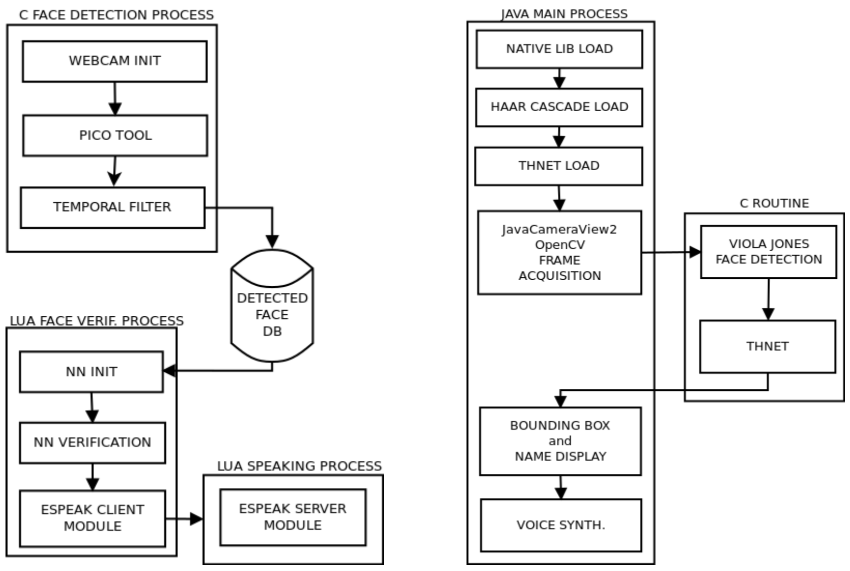
The system we developed is at its early stage but is already capable to fulfill all of the basic requirements needed:

– acquisition of a video stream from a webcam with HD resolution;
– detection of multiple faces from the scene, even if people are not perfectly in front of the camera and the faces have some yaw, pitch and roll;
– verification of one or more candidates in the scene;
– generation of a vocal output as a synthesized voice saying the name of the recognized faces and a confidence value.

Different implementations of the system have been realized, each having its own peculiarities and drawbacks: a Linux x86_64 program for a PC, a Linux (Arm-based) program for a single board computer (SBC) and an Android software for smartphone. The first two implementations are tightly coupled, since they basically run the same software.

The system was first programmed and tested on an Intel i7-6700, 3.4 GHz, quad-core CPU with 16 GB RAM, Linux OS. The fine-tuning of the basenet is carried out on this machine, within a Torch environment; the overall learning time is, on the average, around 0.33 seconds per sample (on the CPU). The basic system (fig. 1a) for the inference phase is composed of three modules: a face detection tool, written in C, that utilizes the PICO [4] tool modified in order to exploit temporal information for the reduction of the false faces; a face verification tool, written in Lua; and an audio message synthesizer, which uses a Linux ESPEAK module.

Adopting a single-board PC is a convenient way to port the previously described solution on a wearable lightweight platform without having to perform substantial changes. Of course the typical drawback of these platforms is their lower performances. However, analyzing the vast panorama of such systems very interesting solutions can be discovered. Indeed, several SBCs have recently appeared on the market, the most renowned being Raspberry, Udoo, Odroid, Lattepanda, BeegleBone; they are released at an impressive cadence in progressively more powerful versions. We decided to adopt an Odroid XU4 board to develop our prototype. It can be noted that even if most developers seem to consider Raspberry Pi 3 the best platform [5], while the Odroid XU4 currently is just ranked 5th, this ranking is related not only to system performance, but also to other aspects that we did not consider mandatory for the present project, such as the platform cost, the availability of software in the Web or the presence of communities that support software updates and forums. The most attractive feature of the Odroid XU4 platform is its processor: the Odroid XU4 adopts a Samsung Exynos5422 octa-core working at 2 GHz and presents performances highly superior to those of all other competitors. High performances in our case are fundamental in order to process the data through the DNN in real time, since this architecture is computationally very demanding. Additionally, the presence of an ARM Mali - T628 GPU may be very useful to further improve in the future the processing throughput by exploiting parallel computation, thanks to the OpenCL support. Another important aspect is the large amount of available memory, 2 GB of embedded DDR3 RAM, which allows easy storage of both the image data and the DNNs configuration parameters. Moreover, the memory can be increased up to 64 GB in eMMC format, allowing much faster access with respect to a common SD memory. Finally, two USB 3.0 ports allow us to easily capture high-resolution video streams from two cameras, while only slower USB 2.0 ports are available for example on the Raspberry. The system currently runs at about 1 fps. We decided to use Android as the software platform for our first smartphone implementation, using the rear camera of the device. Some testing has been performed on an external USB camera too, but the smartphone solution becomes very interesting specially if used as is, without external hardware. The main advantage of smartphone-based implementations is of course their ubiquitous presence. This gives us the opportunity to exploit a very powerful and sophisticated platform, without the need of designing any hardware or any low-level software. In the preliminary implementation we are developing (fig. 1b), for the face detection part we use the Viola-Jones (VJ) algorithm [6] already provided in OpenCV; in particular, in order to improve the speed of detection we use C code instead of Java, implemented using the Native Development Kit (NDK) provided by Google. The porting of the face recognition algorithm, in turn, has been possible thanks to the *thnets* library [7] freely available on the *github* repository; it is a stand-alone library for Torch neural networks, and relies on openBLAS and OpenMP. In order to fulfill these dependencies, and since part of the code is written in assembly language for performance reasons (and is thus strictly hardware dependent), presently we limit ourselves to Arm devices only, that anyway represent the largest portion of the smartphone market; further work will be necessary for the x86 Android version. Finally, for what concerns the voice synthesis, we use the Java interface for the embedded talk system provided by Android. The mobile phone used for testing uses an octa-core Huawei



(a) Multiple process face verification system, PC/Odroid prototype

(b) Android prototype

**Fig. 1.** Processing scheme for the tree prototypes

HiSilicon Kirin 655 SoC (Arm-v8a architecture): four cores run at 2.1 GHz and the remaining four at 1.7 GHz. The SoC embeds even an OpenCL-ready Arm Mali-T830 MP2 GPU, that is presently not used. In order to improve compatibility with many devices, all the native codes are compiled for a 32 bit Arm-v7 architecture.

## 4 Results

The accuracy of the three different verification systems discussed in Section 2 was tested using the same set of 1500 images, with 250 positive samples for each subject and 250 negative samples, while the fine tuning was done using roughly 2500 images both for positives and negatives. In particular, the negative sample test set is composed by 25 images each of 10 different individuals. There is no intersection between the training set and the test set negative images for a single class verification not only in terms of the samples (which is obvious) but also in terms of the individuals.

For the distance classifier and the multi-class verification, all 1500 images were tested, whereas for single-class verification 250 positive and 250 negative faces were tested. The true positive and false positive rates in each case are shown in Table 1.

It may be seen that the true positive and false alarm rates of the distance classifier and of the single-class verifier are good on this testset, while the false

| | DC | MV | C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|---|---|---|
| TA(%) | 93 | 97 | 94.2 | 92.0 | 89.3 | 89.8 | 94.7 |
| FA(%) | 3 | 30 | 0 | 0 | 0 | 0 | 1.6 |

**Table 1.** The left part of the table shows the percent accuracy of the distance classifier (DC) and of the multi-class verifier (MV). TA is the rate of ground-truth positive samples correctly classified as positive, FA is the rate of negative samples uncorrectly classified as positive ones. The right part of the table shows the percent accuracy of the single-class verifier for classes 1...5 (the threshold of the network's output is 0.9).

| | Face Detection | Face Verification | Total |
|---|---|---|---|
| x86 | 0.2 | 0.02 | 0.22 |
| Odroid | 0.08 | 0.8 | 0.9 |
| Android | 0.07 | 0.3 | 0.5 |

**Table 2.** Processing times per frame (seconds).

alarm rate of the multi-class verifier is by far too large. A basic reason may be that the fine tuning of the classification layer for the latter , made with 10,000 images, is not sufficient for the detection of outliers. Another reason could be that the number of classification layer is too low. In such a case it is advisable to use the other two approaches. However, it is also noted that the distance classifier works better with the fine-tuned network as compared to the base feature extractor network.

It is left to the user to select the reliability threshold for the verification. Table 1 shows results at 0.9 threshold in a scale of 0-1. If the user receives no verification output from the system, he/she may lower the threshold for a weak match; for example, (TA, FA) for C4 in Table 1 becomes (93, 0.8) at 0.85 threshold. It may be noted that the user can set separate thresholds for the different single-class verifiers at the same time, or a single threshold for all classes in case of the distance classifier.

Some real-time experiments of the entire system were run on the different hardware versions and their average performance in terms of speed is reported in Table 2. The actual frame rates depend on the scene content. For example, the detection time for the VJ algorithm used for Android varies according to the number of faces found in a frame. This is because it uses a cascade of classifiers: if a classifier in the first layer finds something, information is passed to the other classifiers, searching for Haar features. For the Android system used in this work, face detection runs at about 25 fps with no face detected, while it drops to 6-10 fps with 6 faces in the scene. Similarly, the time for verification per frame increases with the number of faces to be verified.

## 5 Conclusions

The main indication that our results provide is that, thanks to the effectiveness of novel DNN techniques and to the ever increasing computational power of the available hardware, we are getting closer to the realization of effective and usable devices. We have realised several different preliminary versions of our system: a Linux x86_64 program for a PC, mostly used for software development and debugging, a Linux (Arm-based) program for a single-board computer and an Android software for smartphones. The two latter solutions are real mobile implementation which will be given to the members of our Users Group in order to collect information on their actual advantages and drawbacks.

## References

1. S. Carrato, S. Marsi, E. Medvet, F.A. Pellegrino, G. Ramponi, M. Vittori, "Computer vision for the blind: a dataset for experiments on face detection and recognition", 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), May 2016.
2. A.K. Jain, R.M. Bolle, S. Pankanti, (Eds.) "Biometrics: personal identification in networked society", Springer, 2006, ISBN 978-0-387-28539-9.
3. Florian Schroff, Dmitry Kalenichenko, James Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2015.
4. Markus, N., Frljak, M., Pandzic, I.S., Ahlberg, J., Forchheimer, R., "Object detection with pixel intensity comparisons organized in decision trees". arXiv preprint arXiv:1305.4537 (2013).
5. https://www.slant.co/topics/1629/community/best-single-board-computers
6. P. Viola and M. J. Jones, "Robust real-time face detection", International Journal of Computer Vision, vol. 57, no. 2, pp. 137154, 2004.
7. https://github.com/mvitez/thnets.