J. Vis. Commun. Image R. 36 (2016) 142–148

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Rate-energy-accuracy optimization of convolutional architectures for face recognition ☆

L. Bondi [a], L. Baroffio [a,*], M. Cesana [a], M. Tagliasacchi [a], G. Chiachia [b], A. Rocha [b]

[a] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo Da Vinci, 32, 20133 Milan, Italy
[b] Reasoning for Complex Data (RECOD) Lab., Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, Brazil

ARTICLE INFO

Article history:
Received 30 January 2015
Accepted 22 December 2015
Available online 3 February 2016

Keywords:
Convolutional architectures
Convolutional Neural Networks (CNNs)
Optimization
Coding
Face recognition
Analyze-then-Compress (ATC)
Deep learning
Deep neural networks

ABSTRACT

Face recognition systems based on Convolutional Neural Networks (CNNs) or convolutional architectures currently represent the state of the art, achieving an accuracy comparable to that of humans. Nonetheless, there are two issues that might hinder their adoption on distributed battery-operated devices (e.g., visual sensor nodes, smartphones, and wearable devices). First, convolutional architectures are usually computationally demanding, especially when the depth of the network is increased to maximize accuracy. Second, transmitting the output features produced by a CNN might require a bitrate higher than the one needed for coding the input image. Therefore, in this paper we address the problem of optimizing the energy-rate-accuracy characteristics of a convolutional architecture for face recognition. We carefully profile a CNN implementation on a Raspberry Pi device and optimize the structure of the neural network, achieving a 17-fold speedup without significantly affecting recognition accuracy. Moreover, we propose a coding architecture custom-tailored to features extracted by such model.

1. Introduction

Humans are able to identify and recognize a face, automatically assigning it to a given person, with just a few glances. Our brain processes visual stimuli and stores a concise representation of a face in our memory. It is able to correctly distinguish between up to tens of thousands of different faces, although it might not be able to recall the name of the person they correspond to [22]. Furthermore, our brain is capable of recognizing instances of the same face acquired under different conditions (e.g. point of view, illumination, aging).

The problem of automatically detecting and recognizing faces has received significant attention in the past fifty years, with the goal of matching the capabilities of human vision. The first attempts date back to the end of the 1960s, with the models proposed by Bledsoe [3] and Kanade [13]. Such early efforts strived for modeling a face in terms of fiducial points and their relationships, and are quite fragile to changes in imaging conditions. More recently, such approaches have been outperformed by more effective models that achieve better recognition accuracy while requiring low computational resources. Sirovich and Kirby propose Eigenfaces [23], a compact yet effective representation of human faces. Turk and Pentland extended upon such a model to build a computationally efficient face detection and recognition system [25]. Face recognition and verification systems based on aligned images acquired in controlled environments have made great strides in the last twenty years, being able to reduce the error rate by three orders of magnitude [18]. Most current approaches achieve state-of-the-art performance by exploring rich representations of the underlying visual content that consists of up to tens of thousands handcrafted features [1,4].

In the last few years, following a trend also present in several image classification tasks, Convolutional Neural Networks (CNNs) and Deep Learning techniques have been applied to large-scale face verification and recognition systems as well [9,19]. Recently, Taigman et al. proposed DeepFace [24], a deep CNN that has been proven to perform on a par with humans in terms of face recognition accuracy. Such a system exploits a large amount of heterogeneous training data to learn discriminative low-dimensional representations of faces. Likewise, Chiachia et al. [5] have used CNNs to address the problem of familiar face recognition by explicitly learning enhanced person-specific face representations from large amounts of experience with the appearance of individuals.

☆ This paper has been recommended for acceptance by Yehoshua Zeevi.

* Corresponding author.
E-mail addresses: luca.bondi@polimi.it (L. Bondi), luca.baroffio@polimi.it (L. Baroffio), matteo.cesana@polimi.it (M. Cesana), marco.taglisacchi@polimi.it (M. Tagliasacchi), chiachia@ic.unicamp.br (G. Chiachia), rocha@ic.unicamp.br (A. Rocha).

http://dx.doi.org/10.1016/j.jvcir.2015.12.015
1047-3203/

CNNs usually require a large amount of training data and computational resources to be effectively trained. Nonetheless, convolutional architectures based on random weights have been shown to provide good results in terms of accuracy for the task of face recognition and matching.

Effective technologies for automatic face recognition enable a large number of services, such as automated surveillance systems, person authentication, image tagging, and advanced human–computer interaction. Such tasks are often performed in a distributed fashion on battery-operated low-power devices such as mobile phones, smart cameras or nodes of a Visual Sensor Network (VSN). The problem of performing distributed visual analysis tasks on low-power devices has been extensively addressed. The traditional approach to such kind of tasks, hereinafter denoted as *Compress-Then-Analyze* (CTA) [21], consists in the acquisition of visual content on a low-power node, its compression by means of image (e.g., JPEG) or video (e.g., H.264/AVC) coding primitives, and its transmission to a central node, where processing takes place.

Recently, the novel *Analyze-Then-Compress* (ATC) paradigm has been proposed [21,20,2,17]. In this paradigm, low-power nodes acquire visual content and extract higher-level information from it by means of efficient feature extraction algorithms. Such content is then compressed resorting to ad hoc coding primitives and transmitted to a central node, that performs visual analysis. Such approach has been proven to achieve good results in terms of rate-energy-accuracy performance for a number of tasks including content-based retrieval [21,20], object tracking [2] and mobile augmented reality [17]. Moving part of the computational burden, that is, detecting faces and computing compact yet representative features, from a centralized node to sensing devices produces several positive effects:

1. it fairly distributes computational complexity over network nodes, without requiring power-eager central units to serve a high number of clients;
2. it requires less bandwidth than transmitting images or video sequences; and
3. it reduces the risk of privacy violations, avoiding the transmission of the pixel-level visual content.

In this paper, we aim at investigating the adoption of convolutional architectures in distributed battery-operated devices, with the objective of enabling the *Analyze-Then-Compress* paradigm for face identification/recognition.

According to [10–12,16], there are two main problems in biometrics: verification and identification. Depending on the application context, a biometric system may operate either in verification mode or in identification/recognition mode.

In the verification mode, we have a biometric system which validates a person's identity by comparing the acquired biometric sample with his own biometric template previously stored in the system database. In this setup, an individual who wants to be recognized claims an identity (e.g., "Bob"), normally using a PIN, a user name, or even a smart card, and the system retrieves Bob's biometric sample from the database and conducts a 1:1 comparison of the retrieved model to the one just acquired from the person claiming to be "Bob" to determine whether the claim is true or not (e.g., "Is this sample really from Bob?").

In turn, in the identification/recognition mode, we have a system which recognizes an individual by searching the templates of all the users in the database for a match. Therefore, in this setup, we have a 1:*N* comparison to establish an individual's identity (or fail to do so if the subject is not enrolled in the system database) without the subject having to claim any identity (e.g., "Whose biometric data is this?").

In this work, therefore, we focus our efforts on the identification/recognition mode in which we have a face sample acquired by a distributed battery-operated device and need to compare it to several other instances (faces) in a server when trying to recognize a given person. In this operational mode, it is a pre-requisite to have some individuals enrolled into the system, otherwise it is not possible to perform any recognition. Posing this problem in the context of machine learning classification, the database or gallery of enrolled individuals represents the training data to which we need to compare an unseen test sample when performing recognition. Here, if "Bob" is enrolled in the system, it means that the system has been fed with some of his face samples. When he steps in for recognition, the system collects a new face sample and compares it to all existing user face samples in the system database when performing recognition.

The adoption of complex convolutional models in the context of face recognition requires to address two issues, i.e., (i) the high demand of computational resources; and (ii) the apparent data expansion introduced by such architectures. In particular, we empirically evaluate the computational requirements of the feature extraction process, investigating the impact of all hyperparameters. We perform the experiments on a low-power ARM-based Raspberry Pi computer taking the convolutional architecture proposed by Pinto et al. [19] as a reference. Furthermore, we optimize the CNN considering two main aspects: first, we propose an energy-efficient yet effective convolutional network, custom-tailored to the computational resources of the available hardware; second, we introduce ad hoc feature coding primitives aimed at significantly reducing the output bitrate of such a model.

We organized the remaining of this paper into four sections. Section 2 gives an overview of Convolutional Neural Networks. Section 3 introduces an optimized convolutional architecture that dramatically speeds up feature extraction. Section 4 describes the proposed architecture and a rate-accuracy comparison between ATC vs. CTA. Finally, Section 5 presents the conclusions and future research directions.

2. Background on Convolutional Neural Networks (CNNs) and convolutional architectures

With the "big data" revolution, many applications including computer vision, speech/audio recognition, social networks, among others, are dealing with larger and larger amounts of data. At the same time, the advent of powerful, parallel and scalable computing architectures is enabling more complex and effective statistical and computational models. In this context, large neural networks are getting constantly increasing attention and achieve outstanding results in terms of task accuracy for a number of heterogeneous applications [14,24,7].

In particular, in the context of computer vision, Convolutional Neural Networks (CNNs) have been proposed as effective variants of MultiLayer Perceptrons, inspired by human visual cortex [15] with very heterogeneous applications [14,24]. They combine three main features to achieve invariance with respect to imaging conditions: local receptive fields, shared weights, and spatial pooling. Fig. 1 depicts a block diagram of the 3-layer architecture proposed by Pinto et al. [19] as an example. Each layer of such model comprises sublayers of neurons that perform filtering, thresholding, spatial pooling, and normalization. These linear and non-linear operations generate increasingly more complex feature maps, ultimately outputting a feature-based representation that is fed to a classifier for prediction. Traditional CNNs require a computationally expensive training stage to be performed on a large amount of training data in order to learn the filter weights. Instead, the peculiarity of the convolutional architecture proposed by Pinto

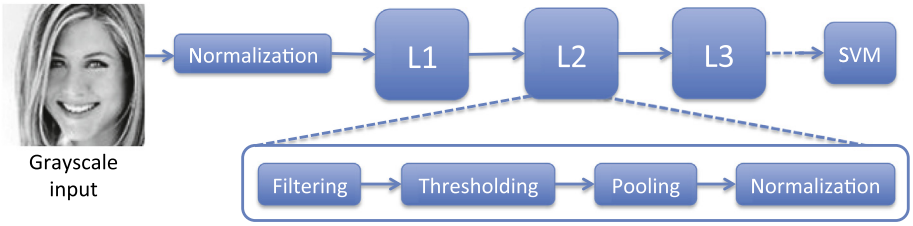

Fig. 1. A block diagram of a 3-layer convolutional architecture.

et al. [19] is that random filter kernels, drawn from a uniform distribution, are employed. Nonetheless, such architecture performs surprisingly well with respect to other state-of-the-art algorithms [5,8].

We consider an architecture composed of layers $Layer^l$, $l = 1,\ldots,L$. Such a network is fed with an input image $\mathbf{I}$ with size 200×200 pixels. We define $\mathbf{I}^l$ as the input of $Layer_l$. Such a representation is, except for the input image, a three-dimensional stack of feature maps consisting of k^{l-1} filtered images having size $M^l \times N^l$, where k^{l-1} is the number of filters employed in the previous layer. Fig. 2 shows an example of feature maps generated at different layers. Precisely, the four operations that define a layer are as follows.

2.1. Filtering

Considering $Layer^l$, a bank $\mathbf{\Phi}^l$ of k^l filters $\Phi_i^l \in \mathbb{R}^{f_s^l \times f_s^l \times k^{l-1}}$, $i = 1,\ldots,k^l$ is defined. Note that each filter is three-dimensional, where the first two dimensions define the shape of the filter and the third one is equal to the number of maps originating from the previous layer. Each filter bank generates a stack of feature maps $\mathbf{F}^l = Filter(\mathbf{I}^l, \mathbf{\Phi}^l)$. The stack is composed of k^l maps with size $M^l \times N^l$, each one obtained by performing a correlation operation with a filter in the bank, i.e. $F_i^l = \mathbf{I}^l \otimes \Phi_i^l$, $i = 1,\ldots,k^l$. The correlation is performed by sliding the mask on the first and second dimensions of the input $\mathbf{I}^l$.

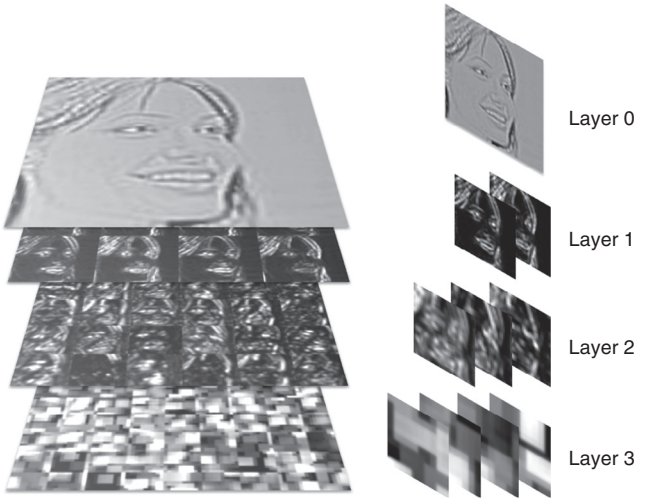

Fig. 2. Visualization of the feature maps generated by a 3-layer architecture. Top to bottom: input image (after input normalization), subset of feature maps generated as output of the first, second and third layers, respectively. Former layers detect edges and details, whereas deeper layers identify complex structures and facial features.

2.2. Thresholding

A stack of thresholded maps $\mathbf{A}^l = \{A_i^l\}$, $i = 1,\ldots,k^l$ is obtained by applying a non-linear thresholding and saturation activation function to the elements of each map generated by the filtering process, i.e.,

$$A_i^l(x,y) = \begin{cases} \gamma_{max} & \text{if } F_i^l(x,y) > \gamma_{max} \\ \gamma_{min} & \text{if } F_i^l(x,y) < \gamma_{min} \\ F_i^l(x,y) & \text{otherwise} \end{cases} \tag{1}$$

$x = 1,\ldots,M^l$, $y = 1,\ldots,N^l$, $i = 1,\ldots,k^l$.

2.3. Pooling

A stack of representations $\mathbf{P}^l = \{P_i^l\}$, $i = 1,\ldots,k^l$ is obtained by applying the following pooling function:

$$P_i^l = Downsample_\alpha\left(\sqrt[p^l]{(A_i^l)^{p^l} \odot \mathbf{1}_{a^l \times a^l}}\right) \tag{2}$$

where $\odot$ indicates a 2D convolution and $\mathbf{1}_{a^l \times a^l}$ is a $a^l \times a^l$ matrix of ones. In other terms, a^l indicates the pooling neighborhood size, p^l denotes the exponent of the pooling function, and α refers to the stride of the subsampling operation.

2.4. Normalization

The last step consists in a normalization of each response. The amount of normalization applied to each sample depends on the amount of activity of its neighbors (across space and feature maps), as thoroughly reported in [6]. In particular, a response is divided by the magnitude of the vector obtained by considering responses in a $b^l \times b^l \times k^l$ neighborhood, if the magnitude of such a vector is greater than a given threshold τ^l. Finally, ρ^l defines the stretch (or gain) applied to the vector of neighboring samples. The output of the normalization step provides the input of the following layer, that is, $\mathbf{I}^{l+1} = Normalize(\mathbf{P}^l)$.

3. Energy profiling and optimization

Considering visual analysis tasks distributed over a network, efficient feature extraction algorithms and coding methods are required for the *Analyze-Then-Compress* paradigm presented in Section 1 to be enabled. As a first step in this direction, we accurately profiled the amount of computational time needed to run a convolution-based feature extraction process on a low-power ARM-based device. We selected the Raspberry Pi (model B) single-board computer as a reference hardware platform. It is based on a Broadcom BCM2835 SoC, including an ARM1176JZF-S (ARMv6) processor at 700 MHz, along with a VideoCore IV GPU module and 512 MB of RAM. We equipped the micro-computer with the Debian-based Raspbian OS, with kernel version 3.12.28+,

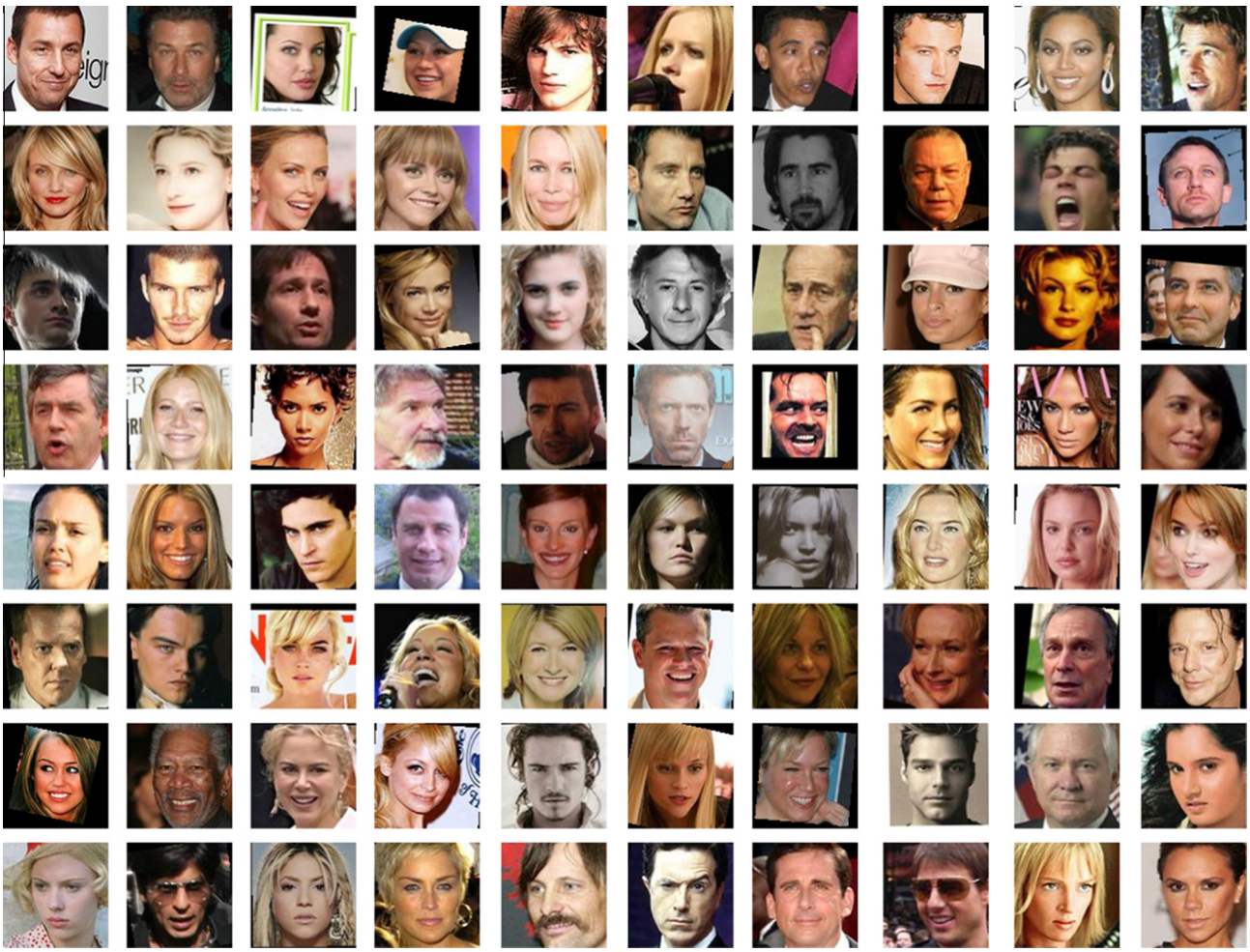

Fig. 3. A small subset of pictures from the aligned version of the *PubFig83* dataset, containing 13,838 pictures corresponding to 83 different public figures regarded as one of the most difficult face recognition datasets in the wild currently.

along with ATLAS 3.8.4 optimized libraries for linear algebra operations.

Moreover, we considered the 3-layer convolutional architecture proposed by Pinto et al. [6], hereinafter denoted as *fg11-ht-l3-1*,[1] as a starting point.

Regarding the dataset for experiments and validation, we resorted to an aligned version[2] of the *PubFig83* dataset [19], containing 13,838 cropped and aligned grayscale face images, corresponding to 83 different public figures, as shown in Fig. 3. Each image has a native resolution of 100×100 pixels.

3.1. Experimental setup

To avoid any bias regarding the learning and optimization of parameters related to the convolutional networks and the actual training of classifiers for recognition, we have split the *PubFig83* dataset into two disjoint subsets: a *development* set and an *evaluation* set. The *development* set serves as our training set for learning any properties related to the networks and their optimization. This set comprises 41 subjects. After finding the necessary network parameters, this set is completely disregarded and the calculated network and its parameters are then used for feature extraction in the *evaluation* set. Complementarily, the *evaluation* set comprises the remaining 42 subjects of the *PubFig83* dataset. In this set, we train the appropriate classifiers for recognition and test them on an individual basis.

In this vein, to avoid any bias regarding the split of images (views of each individual) for training and testing, we divide the *evaluation* set into a *gallery* and a *probe* set in a proportion of 90%/10%, respectively, and repeat this process 10 times (10-fold cross-validation protocol), each time considering a different 10% of data points (individual views) for the probe set. This process gives us a better picture of the variation considering different splits of *gallery* and *probe* sets.

To recognize each individual enrolled in the gallery given a probe, the classification is performed in an *one-versus-all* fashion, in order to support the multinomial classification problem posed by face recognition. In particular, for each class (individual), a Support Vector Machine (SVM) is trained to separate training instances belonging to such a class (gallery images of this particular individual) and instances belonging to all other classes (gallery images of the other individuals). Then, given a test instance (probe), the likelihood of each class is evaluated resorting to the corresponding SVM, and the output class corresponds to the one with the larger classification margin. That is, SVMs are used to classify input instances based on the feature maps generated by the convolution model learned beforehand, in the separated *development* set.

[1] *fg11*: network class, *ht*: high-throughput, *l3*: 3 layers, *1*: top-1 best performing instance.

[2] The aligned dataset can be downloaded at http://www.ic.unicamp.br/chiachia/resources/pubfig83-aligned.

To speed up the classification process, the regularization parameter has been fixed to a sufficiently high value ($C = 10^5$).

We measured the accuracy of the face recognition task in terms of top-1 result precision. All recognition results are reported as the average for the different probe sets. There is no overlapping between train and test images whatsoever.

3.2. Experiments

The original *fg11-ht-l3-1* model has 90.00% of accuracy on the development set, with an average feature extraction time of 17.88 s per image on the reference hardware platform. Tables 1 and 2 show the hyperparameter settings for such a network and the computational time required by each component, respectively. The most demanding operations are the filter-bank correlations at the second and third layer of the network. Such stages consist in 128 and 256 filtering operations at 88×88 and 34×34 resolution for the second and the third layer, respectively. Due to the nature of the original *fg11-ht-l3-1* model, images needs to be upsampled to 200×200 pixels before being fed to the CNN. As a first step towards an energy-accuracy optimized model, we disabled the initial image upsampling process and we removed the last layer of the network, that accounts to more than 20% of the total computational requirements. We obtained a two-layer network, hereinafter denoted as *fg11-ht-l2-s1* (2 layers, 1st search, subsampled input image), that achieves 84.10% recognition accuracy on the development set at 3.39 s per image. Although the computational cost was reduced by more than five times, the accuracy was also reduced by approximately seven percentage points. Once again, the most computationally intensive operation is the second layer filter-bank correlation, requiring as much as 70% of the total CPU time.

Starting from such a simpler and more energy-efficient architecture, a parametric model search was performed to evaluate the impact of network hyperparameters on both execution time and recognition accuracy. First, we evaluated the impact of the number of filters employed at each layer. We considered networks with {16, 32, 64} and {16, 32, 64, 128} filters at the first and the second layer, respectively. Fig. 4 shows the energy-accuracy performance of such CNN instances, in which energy is expressed in terms of CPU time for feature extraction. The two instances corresponding to red dots represent good tradeoffs between complexity and accuracy performance. In particular, the red dot on the left corresponds to a model having 16 and 128 filters, whereas the one on the right corresponds to 32 and 128 filters at the first and second layer, respectively.

Starting from such promising architectures, we performed a stepwise model search by varying other hyperparameters of the model. In particular, we analyzed 413 models, obtained by varying the following hyperparameters:

- Number of filters in the first layer: {16, 32}.
- Normalization neighbors size in all layers: {3, 5, 7, 9}.
- Normalization stretch (ρ) in all layers: {0.1, 1, 10}.
- Pooling neighbors size in layers 1 and 2: {3, 5, 7, 9}.
- Pooling exponent (p) in layers 1 and 2: {1, 2, 10}.

Fig. 5 depicts the energy-accuracy performance of such models. The red dot refers to an energy efficient yet well accurate instance, hereinafter denoted as *fg11-ht-l2-opt*. The settings for such instance are reported in Table 3. It performs almost on a par with the much more complex 3-layer instance, achieving 89.51% task accuracy on the development set, while requiring as few as 1.06 s per image on the target hardware platform, thus achieving

Table 1
Hyperparameters for the *fg11-ht-l3-1* instance.

Layer 0			Layer 1		
fbcorr	shape	–	fbcorr	shape	3×3
	filt n	–		filt n	64
thresh	th min	–	thresh	th min	0
	th max	–		th max	∞
lpool	shape	–	lpool	shape	7×7
	order	–		order	1
	stride	–		stride	2
lnorm	shape	9×9	lnorm	shape	5×5
	stretch	10		stretch	0.1
	thresh	1		thresh	1
Layer 2			**Layer 3**		
fbcorr	shape	5×5	fbcorr	shape	5×5
	filt n	128		filt n	256
thresh	th min	0	thresh	th min	0
	th max	∞		th max	∞
lpool	shape	5×5	lpool	shape	7×7
	order	1		order	10
	stride	2		stride	2
lnorm	shape	7×7	lnorm	shape	3×3
	stretch	1		stretch	10
	thresh	1		thresh	1

Table 2
Energy profiling for the *fg11-ht-l3-1* and *fg11-ht-l2-opt* models.

	fg11-ht-l3-1		*fg11-ht-l2-opt*	
	Time [s]	%	Time [s]	%
Layer 0	00.06	00.34	00.02	01.86
lnorm	**00.06**	**00.34**	**00.02**	**01.85**
Layer 1	03.76	21.00	00.20	18.61
fbcorr	00.69	03.84	00.05	04.29
lpool	**02.90**	**16.22**	**00.13**	**12.66**
lnorm	00.16	00.90	00.02	01.58
Layer 2	09.42	52.69	00.84	79.49
fbcorr	**08.44**	**47.23**	**00.56**	**52.96**
lpool	00.93	05.22	00.27	25.33
lnorm	00.04	00.22	00.01	01.09
Layer 3	04.64	25.96		
fbcorr	**04.20**	**23.51**		
lpool	00.42	02.36		
lnorm	00.01	00.08		
Total	17.88	100.0	01.06	100.0

Bold values indicate the operations that require most computational time within each layer.

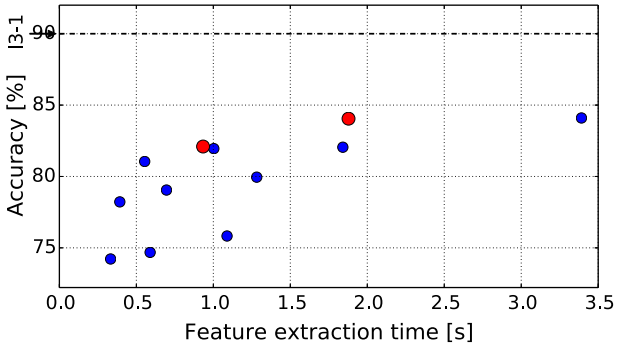

Fig. 4. Energy-accuracy trade-off on the *development* set for the 2-layer convolutional architectures obtained by varying the number of filters in the first and second layer. Each point represents a different combination of parameters. The left red dot represents the model with 16 filters in the first layer and 128 filters in the second layer. The right red dot represents the model with 32 filters in the first layer and 128 filters in the second layer. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

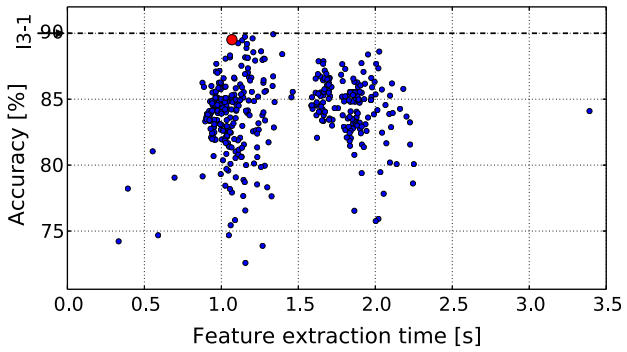

Fig. 5. Energy-accuracy trade-off on the *development* set for the 2-layer models obtained by varying the number of filters in the first layer, and the normalization and pooling hyperparameters.

Table 3
Hyperparameters for the *fg11-ht-l2-opt* instance.

Layer 0					
lnorm		shape		9×9	
		stretch		10	
		thresh		1	
Layer 1			**Layer 2**		
fbcorr	shape	3×3	fbcorr	shape	5×5
	filt n	16		filt n	128
thresh	th min	0	thresh	th min	0
	th max	∞		th max	∞
lpool	shape	5×5	lpool	shape	5×5
	order	1		order	10
	stride	2		stride	2
lnorm	shape	5×5	lnorm	shape	3×3
	stretch	10		stretch	1
	thresh	1		thresh	1

a 17-fold speedup with respect to the reference *fg11-ht-l3-1* instance.

In order to validate the results, we tested both the original *fg11-ht-l3-1* model and the reduced *fg11-ht-l2-opt* model on the *evaluation* set. The accuracy for the original 3-layer instance is 89.71% while the accuracy for the reduced 2-layer instance is 89.29%. When considering the whole *PubFig83* dataset (standard protocol), the accuracy of the original *fg11-ht-l3-1* instance reaches 87.13% of accuracy, while the *fg11-ht-l2-opt* instance reaches 86.73% of accuracy.

For the sake of completeness, we compare Pinto's convolutional architecture with a baseline method, exploiting a state-of-the-art Convolutional Neural Network for image classification. In particular, we extracted features from face images with the well known AlexNet CNN. Then we trained a SVM classifier in a one-versus-all approach on top of them, following the same validation protocol used for Pinto's model. The average accuracy on the *PubFig83* dataset is only 52.84%, with an execution time of 8 s on the RaspberryPi. Such results suggest that features learned from a very large, general-purpose training set of images are not suitable to face recognition applications. Such very deep networks can indeed achieve a good accuracy performance if trained on a very large set of faces [24]. Nonetheless, they require up to millions of training samples to be effectively trained, and datasets with such size are not publicly available in the context of face recognition.

The random filter convolutional architectures have also been tested on the challenging *MOBIO* dataset containing 150 different subjects. This dataset has been employed in the 2013 Biometrics Competition on face recognition in Mobile Environment. In this case, we followed a 10-fold cross validation procedure with 10 training samples and 90 test samples per subject, reaching an average accuracy of 94.32% with the *fg11-ht-l3-1* model and 92.29% with the *fg11-ht-l2-opt* one.

4. Rate-accuracy analysis

As previously mentioned, the *Analyze-Then-Compress* (ATC) paradigm has gained popularity as an alternative to the traditional *Compress-Then-Analyze* (CTA) paradigm. In order to be competitive with CTA, ATC calls for efficient coding primitives custom-tailored to task-specific visual features extracted by the remote devices. Thus, we address the problem of coding the output of the *fg11-ht-l2-opt* model presented in Section 3 (best one in terms of energy-accuracy performance), which consists of a feature map composed of 25,088 32-bit floating-point elements. Note that this would require approximately 803 kbit per image, if no compression is applied. On the other hand, with the CTA paradigm, each input image can be encoded resorting to JPEG compression using as few as 62 kbit, even at very high quality (quality factor equal to 100).

We designed a simple encoding pipeline based on transform coding comprising three main steps: (i) Karhunen-Loève Transform (KLT), also known as Principal Component Analysis – PCA in the machine learning/computer vision literature, to decorrelate the input feature vector and reduce its dimensionality to a target number of elements $K = \{1600, 3200, 6400\}$; (ii) uniform quantization with quantization step equal to $\Delta = \{0.05, 0.03, 0.02, 0.01, 0.005\}$; and (iii) entropy coding, resorting to an arithmetic coder. Note that both KLT transform and the statistical models used by the arithmetic coder were learned on a training set, and applied to the test set when performing cross validation.

Fig. 6 shows the rate-accuracy curves obtained on the whole *PubFig83* dataset with this coding architecture (solid lines), applied to the feature map generated by the *fg11-ht-l2-opt* convolutional instance. Each curve corresponds to a different value of K and it is traced by varying Δ. As to CTA, each curve (dashed lines) was obtained by varying the JPEG Quality Factor within the set QF = {10, 30, 50, 70, 90}.

It is possible to observe that the ATC paradigm based on the optimized *fg11-ht-l2-opt* features outperforms CTA at a bitrate less than (approx.) 17,000 bits per image. Even at higher bitrates, the difference in accuracy is very small. With ATC, it is possible to operate at rates for which CTA performs very poorly. For example, ATC attains an accuracy equal to 85.41% (merely 1.72 percentage points less than the *fg11-ht-l3-1* model based on uncompressed images), while requiring as few as 7379 bits per image, when considering $K = 3200$ and $\Delta = 0.01$. This corresponds to a bitrate reduction of 99.1% and 50.1% with respect to uncompressed features and the CTA paradigm, respectively. Furthermore, CTA

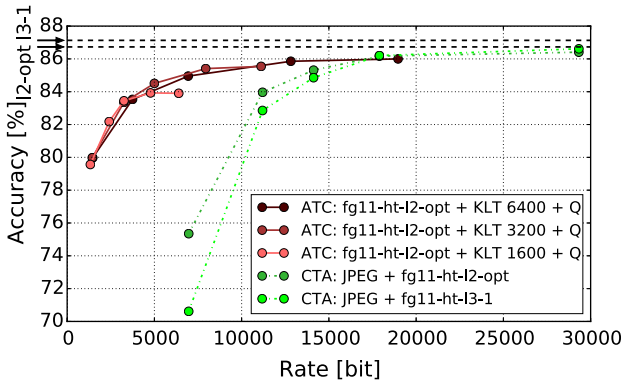

Fig. 6. Rate-accuracy curves. ATC vs. CTA on the whole *PubFig83* dataset. Results are the average over a 10-fold cross validation procedure.

requires at least approximately 7000 bits/image. On the other hand, when transmission resources are severely constrained, ATC is the only viable option, and achieves good performance even at low bitrates. Considering the case of $K = 1600$ and $\Delta = 0.03$, it is possible to achieve a task accuracy equal to 82.18% (4.95 percentage points less than the *fg11-ht-l3-1* instance), requiring as few as 2410 bits/image (99.7% and 76.3% bitrate reduction with respect to uncompressed data and the CTA paradigm, respectively, at the same target accuracy).

5. Conclusions

We propose a rate-energy-accuracy optimized convolutional architecture for face recognition operating in distributed devices. The proposed architecture requires on average 94% less energy as compared to the baseline architecture in order to extract features from an input image on a low-power ARM-based Raspberry Pi computer. Also, it achieves a recognition rate of 85.41% (only 1.72% less than the uncompressed baseline) while requiring as few as 7379 bits per image.

Our results clearly show that the proposed ATC approach is highly competitive with CTA in a distributed face recognition setup when the main constraint is represented by the limited bandwidth. In terms of overall energy requirement, CTA benefits from a highly optimized implementation of JPEG that requires as few as 11 ms on the Raspberry Pi device, while our ATC approach requires 1.06 s. Even though such ATC requirement might already suit a number distributed face recognition applications, as future research we plan to investigate faster implementations of convolutional architectures, possibly relying on dedicated hardware for some operations such as the filtering. In addition, future work will aim at extending the optimized model to other applications such as object classification and pedestrian detection.

Finally, all the source code for the developed methods is freely available at www.greeneyesproject.eu.

Acknowledgments

The project GreenEyes acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open Grant No.: 296676. We also thank the financial support of the Brazilian National Research Council (CNPq) and the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES), through the DeepEyes project, and of the São Paulo Research Foundation (FAPESP) through Grant No. 2013/11359-0.

References

[1] O. Barkan, J. Weill, L. Wolf, H. Aronowitz, Fast high dimensional vector multiplication face recognition, in: 2013 IEEE International Conference on Computer Vision (ICCV), December 2013, pp. 1960–1967.
[2] L. Baroffio, M. Cesana, A. Redondi, M. Tagliasacchi, S. Tubaro, Coding visual features extracted from video sequences, IEEE Trans. Image Process. (TIP) 23 (5) (2014) 2262–2276.
[3] W.W. Bledsoe, The Model Method in Facial Recognition, Panoramic Research, Inc., 1966.
[4] D. Chen, X. Cao, F. Wen, J. Sun, Blessing of dimensionality: high-dimensional feature and its efficient compression for face verification, in: CVPR, IEEE, 2013, pp. 3025–3032.
[5] G. Chiachia, A.X. Falcao, N. Pinto, A. Rocha, D. Cox, Learning person-specific representations from faces in the wild, IEEE Trans. Inf. Forensics Secur. 9 (12) (2014) 2089–2099.
[6] D. Cox, N. Pinto, Beyond simple features: a large-scale feature search approach to unconstrained face recognition, in: IEEE Intl. Conference on Automatic Face Gesture Recognition and Workshops (FG), 2011, pp. 8–15.
[7] G. Dahl, D. Yu, L. Deng, A. Acero, Context-dependent pre-trained deep neural networks for large vocabulary speech recognition, IEEE Trans. Audio Speech Lang. Process. (TASL) 20 (1) (2012) 30–42.
[8] M. Gunther, A. Costa-Pazo, C. Ding, E. Boutellaa, G. Chiachia, H. Zhang, M. de Assis Angeloni, V. Struc, E. Khoury, E. Vazquez-Fernandez, D. Tao, M. Bengherabi, D. Cox, S. Kiranyaz, T. de Freitas Pereira, J. Zganec-Gros, E. Argones-Rua, N. Pinto, M. Gabbouj, F. Simoes, S. Dobrisek, D. Gonzalez-Jimenez, A. Rocha, M. Neto, N. Pavesic, A. Falcao, R. Violato, S. Marcel, The 2013 face recognition evaluation in mobile environment, in: Intl. Conference on Biometrics (ICB), June 2013, pp. 1–7.
[9] G. Huang, H. Lee, E. Learned-Miller, Learning hierarchical representations for face verification with convolutional deep belief networks, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2012, pp. 2518–2525.
[10] A. Jain, R. Bolle, S. Pankanti, Introduction to biometrics, IEEE Biometric Consortium (1998).
[11] A.K. Jain, P. Flynn, A.A. Ross (Eds.), Handbook of Biometrics, Springer, 2008.
[12] A.K. Jain, A. Ross, S. Prabhakar, An introduction to biometric recognition, IEEE Trans. Circ. Syst. Video Technol. 14 (1) (2004) 4–20.
[13] T. Kanade, Picture Processing System by Computer Complex and Recognition of Human Faces, Ph.D. Thesis, Kyoto University, November 1973.
[14] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.
[15] Y. Lecun, Y. Bengio, The Handbook of Brain Theory and Neural Networks, The MIT Press, 1995. Ch. Convolutional Networks for Images, Speech and Time Series, pp. 255–258.
[16] S.Z. Li, A.K. Jain (Eds.), Handbook of Face Recognition, first ed., Springer, 2005.
[17] M. Makar, S.S. Tsai, V. Chandreasekhar, D. Chen, B. Girod, Interframe coding of canonical patches for low bit-rate mobile augmented reality, Int. J. Semantic Comput. 07 (01) (2013) 5–24.
[18] P. Phillips, J. Beveridge, B. Draper, G. Givens, A. O'Toole, D. Bolme, J. Dunlop, Y. M. Lui, H. Sahibzada, S. Weimer, An introduction to the good, the bad, amp; the ugly face recognition challenge problem, in: 2011 IEEE Intl. Conference on Automatic Face Gesture Recognition (FG), March 2011, pp. 346–353.
[19] N. Pinto, Z. Stone, T. Zickler, D. Cox, Scaling up biologically-inspired computer vision: a case study in unconstrained face recognition on facebook, in: 2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2011, pp. 35–42.
[20] A. Redondi, L. Baroffio, J. Ascenso, M. Cesana, M. Tagliasacchi, Rate-accuracy optimization of binary descriptors, in: IEEE Intl. Conference on Image Processing (ICIP), Melbourne, Australia, September 2013, pp. 2910–2914.
[21] A. Redondi, L. Baroffio, M. Cesana, M. Tagliasacchi, Compress-then-analyze vs. analyze-then-compress: two paradigms for image analysis in visual sensor networks, in: IEEE Intl. Workshop on Multimedia Signal Processing (MMSP), September 2013, pp. 278–282.
[22] P. Rotshtein, R.N.A. Henson, A. Treves, J. Driver, R.J. Dolan, Morphing Marilyn into maggie dissociates physical and identity face representations in the brain, Nat. Neurosci. 8 (1) (2005) 107–113, http://dx.doi.org/10.1038/nn1370.
[23] L. Sirovich, M. Kirby, Low-dimensional procedure for the characterization of human faces, J. Opt. Soc. Am. A 4 (3) (1987) 519–524.
[24] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, Deepface: closing the gap to human-level performance in face verification, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014, pp. 1701–1708.
[25] M. Turk, A. Pentland, Face recognition using eigenfaces, in: Proceedings CVPR '91., IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991, pp. 586–591.