International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 9, No.4 (July-2020)

http://dx.doi.org/10.12785/ijcds/090416

A Hybrid Face Identification System using a Compressed CNN in a Big Data Environment for Embedded Devices

Majdouline Meddad[1], Chouaib Moujahdi[2], Mounia Mikram[1,3] and Mohammed Rziza[1]

[1] *LRIT Laboratory, Associated Unit to CNRST (URAC 29), Rabat IT Center, Faculty of Sciences, Mohammed V University in Rabat, Morocco*

[2]*Scientific Institute of Rabat, Mohammed V University in Rabat, Morocco*

[3]*School of Information Science in Rabat, Morocco*

Received 25 Nov. 2019, Revised 12 Jun. 2020, Accepted 23 Jun. 2020, Published 1 Jul. 2020

Abstract: Convolutional Neural Networks have proved an excellent efficiency in several modern applications, for example, systems of face identification, like VGGFace and DeepFace, have achieved an excellent performance. However, these models still require huge memory for computation that involves an expensive computation cost especially for applications that use huge databases and that are running in embedded devices. To deal with this problem, we propose in this paper a hybrid identification system that is based on the compression of the VGGFace model for the feature extraction step, and on the indexation and the parallelization for the identification task. The proposed system has been evaluated in term of the Rank-1 prediction, the time of identification and the speed-up using two public face databases. Our experimental results illustrate the ability of the proposed system to preserve the performance while keeping a reasonable time of identification compared to two face identification systems based, respectively, on the original VGGFace model and on the Inception V3 model.
The Compressed Model can be downloaded from this Github link: https://github.com/Majdouline-Meddad/CompressedVGGFace

Keywords: Face Identification, Deep Learning, VGGFace, CNN compression, Inception V3, Big Data, Embedded Devices.

1. INTRODUCTION

Convolutional neural networks furnish a powerful framework to deal with unstructured data such as images, speeches to automatically extract pertinent features. These networks achieved significant successes in various fields, including clustering and object detection, classification, semantic segmentation and face identification.

Although the achievements in the literature are brilliant in face identification like VGGFace [1], DeepFace [2] and others. This CNN models still require huge memory for computation that involves an expensive computation cost especially for applications that use huge databases and that are running in embedded devices.

In general, to deal with this problem, only the parameter sharing among all spatial locations, which is an intrinsic technique of CNNs, is used to decrease the number of parameters.

Among the applications that use massive databases, namely in literature scalability database [3][71], there are face identification systems like Indian Aadhaar that comprises millions of identities. These becomes more challenging in the case of identification systems that are running on embedded devices. In fact, we cannot run the current CNN architecture in such devices because of their very large sizes (for example, VGGFACE size is 553Mo and DEEPFace size is 488Mo).

In this paper, we are concerned with scalability and the performance issues in embedded devices. In this context, we have built a hybrid system that combines three categories that deal with this problem: compression, indexation and parallelization approaches.

For the compression part of our model we have compressed the VGGFace [1] by a factor of 5639x which reduce the size from 566475ko to 312ko. For indexation approaches, we have used the LBG algorithm [4]. For the parallelization, we have used several workers to parallelize the identification tasks [5].

The rest of the paper is organized as follows. Section 2 presents the related works. Section 3 presents our methodology to build our proposed hybrid system. Our experimental results and comparisons are presented in Section 4. Our conclusions and perspectives for future work are provided in Section 5.

E-mail: majdouline.meddad@um5s.net.ma, chouaib.moujahdi@um5.ac.ma, mikram@ieee.org, rziza@fsr.ac.ma

2. RELATED WORKS

The scalability issue in embedded devices for identification systems have given rise to new problems and challenges related to the performance and execution time. Due to these problems, there are currently many research efforts underway to make such systems faster and efficient. We believe that proposed solutions in the literature can be divided into two main categories: Software solution and Hardware Solution. Each category can be further divided into several sub-categories (see Fig.1). The main idea of software solution is the optimization of techniques that can prompt the execution time in order to be used in embedded devices. For Hardware approaches, their key idea is to create chipsets to compress and distribute learning among different memories (S-Ram and D-Ram).

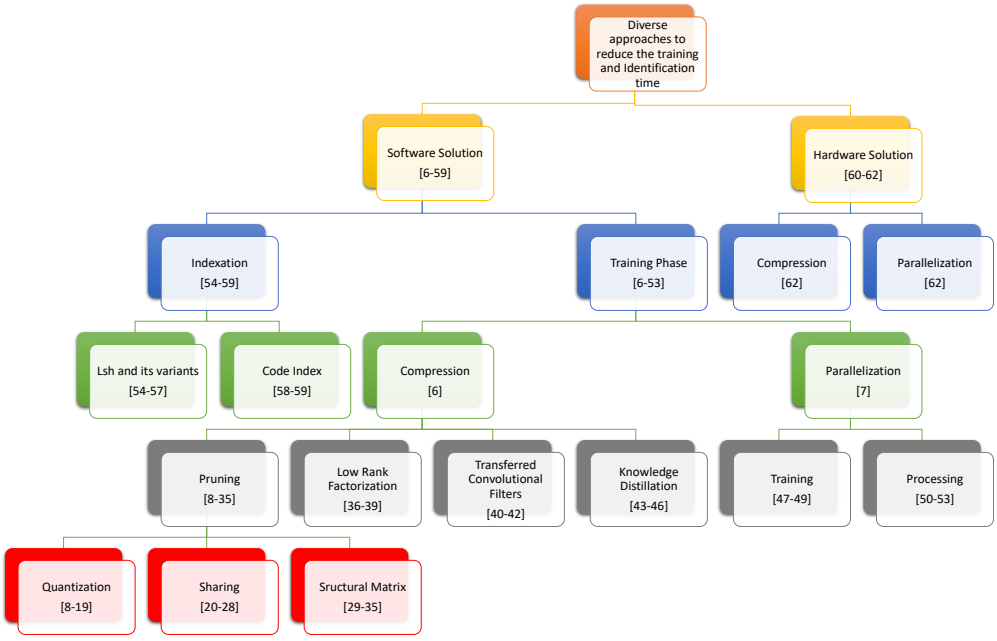

Figure 1 . Diverse approaches to reduce the training and identification time

Software solution can be divided into two sub-categories: Training phase approaches and indexation approaches The key idea of training phase approaches, is to compress or distribute learning algorithms that take a considerable time to find the right weights and biases in order to optimize the model. For indexation approaches, their main idea is to reduce the list of potential identities that will allow the test images to be matched to a smaller number of identities. Indexation approaches take place usually during the identification phase.

Training phase approaches can be divided into two sub-categories: compression approaches [6] and parallelization approaches [7]. Compression approaches present the different techniques that aim to squeeze and speed up the developed models. For parallelization approaches, they use the big data technologies and the cloud computing platforms to deal with the tremendous amount of data.

Compressing approaches can be divided as well into four categories: Pruning approaches [8-35], Low-Rank Factorization approaches [36-39], Transferred Convolutional filters approaches [40-42] and Knowledge Distillation approaches [43-46]. For pruning approaches, the main idea is to remove parameters that are not substantial to the model performance. For Low-Rank Factorization approaches, the general idea is using Tensor/Matrix decomposition to estimate informative parameters and speed up the computation. For Transferred Convolutional filters, the main idea is to use compressing CNN models by the equivariant group theory that claims that both convolutional weight sharing and translation invariant property are important for good predictive performance. For Knowledge Distillation approaches, the general idea is to get shallower model by compressing wide and deep ones, where the compressed model imitate the function learned by the complex one by shifting knowledge from a large model to small one.

It should be noted that, Pruning approaches can be divided into three categories: Quantization approaches, sharing approaches and Structural Matrix approaches. For Quantification approaches, they try either to compress the network by reducing the number of bits required to represent each weight or to directly learn activation binary weights during the model training. For Sharing approaches, faced the major problem which is decreasing the complexity of a model and handle the overfitting issue. For Structural Matrix approaches, the major idea is to prune parameters by imposing the input x layer as a

parameterized structural matrix which can represented with much few parameters.

For quantification approaches, we can cite Gong et al. [8] and Wu et al. [9] that have used k-means scalar quantization to the values of the parameters. Vanhoucke et al. [10] claims that 8-bit quantization of the parameters can outcome a significant speed up with decreasing loss accuracy. Gupta et al. [11] applied 16-bit fixed point representation in stochastic based CNN training, which safely minimized float point operations and memory usage with much less loss in classification accuracy. Han et al. [12] proposed a method that quantizes the link weights, using the sharing of weight and thereafter-applied Huffman coding, to additional reduce the rate of memory usage. Choi et al. [13] proved as well that hessian weight could be used to adjust the prominent network parameter. In addition, many approaches are used to each present each weight by 1-bit such as BinaryConnect[14], BinaryNet [15], XNORNetwork [16]. Merolla et al. [17] proved that specific weight distortions including binary weights could be resilient while training a network with back-propagation. Sang et al. [18] propose to compress the Alex-net and VGG-16 models in three steps: firstly, prune the neural network model to remove redundant connections while keeping important connections. Secondly, quantize weights in three steps: 1) identify the shared weights that fall in the same cluster by using K-means clustering. 2) generate a codebook then quantize the weights with codebook generated before. 3) retrain the code Book. Thirdly, apply the Huffman coding to have more compressing and to take more advantage of the biases distribution. Iandola et al. [19] proposed a SqueezeNet method to compress Alex-net Model by using 8-bit/6-bit quantization, 100%/50%/33% sparsity and fire module which contains 1x1 filters in his squeeze part (squeeze convolutional layer) to replace 3x3 filters then feeding into expanding part that contains a mix of 3x3 and 1x1 convolutional filters that help to limit the number of the input image. The main drawback of these approaches category is that the accuracy of the binary nets is significantly good only when we are dealing with a thin CNNs model.

For Sharing approaches, many works are proposed to reduce the non-crucial parameters of a model such as Biased Weight Decay [20], Optimal Brain Damage [21] and the Optimal Brain Surgeon [22], which minimize the number of connections, based on Hessian loss function, and perform high accuracy. Srinivas and Babu [23] proposed a data-free pruning method to remove the redundancy among neurons. Chen et al. [24] proposed a method for parameter sharing based on creating a HashedNets model to group weights into buckets using a low-cost hash function. Han et al. [12] proposed a deep compression method by deleting redundant connections and quantizing the weights, then use Huffman Coding to encode the quantized weights. Ullrich et al. [25] proposed a simple regularization method based on soft weight sharing, which uses pruning and quantization in one simple retraining procedure. Lebedev et al. [26] imposed some sparsity constraints on the convolutional filters to realize structural brain damage method to reduce the number of connections. Li et al. [27] use L1-norm regularizers to prune and select less influencing filters. Ioffe and Szegedy [28] proposed a method to perform the normalization for each training mini-batch and make it a part of the model architecture. The main drawback of these approaches is that require, first, a manual setup sensitivity of layers and second, more iterations to converge the model while using regularizers.

For Structural matrix approaches, Cheng et al. [29][30] proposed an efficient and simple method, based on circulant projections, that presere competitive error rates. Yang et al. [31] introduced a novel adaptive FastFood transform to reparametrize the matrix-vector multiplication of fully connected layers. The work in Sindhwani et al. [32] showed an effective new notion of parsimony in the theory of structured matrices that can be extended into diverse other structured classes, including multi-level Toeplitz-like [33] matrices and block related to multi-dimensional convolution [34]. Moczulski et al. [35] proposed a general structured efficient linear layer module applied of CNNs to reduce the number of parameters and operations. The main drawback of these approaches is hard to find a structural matrix because there is no theoretical way to derive it out.

For Low-Rank approaches, we can cite Jaderberg et al. [36] and Yuri et al. [37] introduced a proposition to use different tensor decomposition schemes to speed up the training. Lebedev et al. [38] proposed Canonical Polyadic (CP) decomposition for the kernel tensors that uses nonlinear least squares. Tai et al. [39] introduced a new algorithm for training low-rank constrained CNNs from scratch by computing the low-rank tensor decomposition and using Batch Normalization to transform the activation of the internal hidden units. The main drawback of these approaches category is that the implementation is not easy to be performed since it involves some decomposition operation that are computationally expensive. In addition, these approaches perform low-rank approximation layer by layer, which make them unable to perform a global compression of parameters.

For Transferred Convolutional filters approaches, Cohen et al. [40] proposed equivariant group theory to transfer convolution filters and compress CNN models. Li et al. [41] and Zhai et al. [42] introduced a convolutional layer from a set of base filters. We think that the main drawback of this category is that these methods cannot achieve competitive performance in wide and deep architectures (i.e; Googlenet, Residual Net). In addition, the transfer assumptions are sometimes too strong to guide the learning, which makes the result unstable.

For knowledge distillation approaches, we can cite Bucilua et al. [43] that have exploited knowledge transfer to compress the model by training classifiers by using pseudo-data labeled, and then reproduce the output of the original larger network. Hinton et al. [44] introduced a knowledge distillation compression framework based on the student and teacher paradigm that makes the training of deep network easier and more efficient. Luo et al. [45] introduced the knowledge by using the neurons in the higher hidden layer, which preserves as much information as the label probabilities. Chen et al. [46] proved that instantaneously transfer of knowledge from a previous network to each new deeper or wider network accelerates the experimentation process by using the concept of function-preserving transformation. The main drawback of Knowledge distillation approaches is that can be applied only during the classification task with a softmax loss function. Moreover, the model assumptions are sometimes too strict to make a competitive performance with other types of approaches.

For parallelization approaches (see Fig.1), they can be divided into two sub-categories: Training approaches [47-49] and processing parallelization approaches [50-53], For training approaches, Deep Neural Network (DNN) is slow in computation particularly when the size of data is wide which requires the use of cloud computing platforms to speed up the training. For processing parallelization approaches, the main issue treated by this approach is that a single computer can process a large-scale of information in few time such as a video analysis, therefore, perform a parallel-distributed process using the computational resources (Big Data technologies) in a cloud platform.

For training approaches, Sun et al. [47] implemented and designed Map-Reduced neural network algorithm on large-scale data for handwriting character recognition. Basit et al. [48] proposed to improve the accuracy of a naive model by implementing two methods: applied elastic distortion to the training input and implemented distributed computing method by mapping the training dataset onto various machines, then the training process take place simultaneously in Hadoop platform. Liu et al. [49] introduced three parallel neural network methods (MRBPNN 1, MRBPNN 2 and MRBPNN 3) based on the Map-Reduce computing model technique to deal with data-intensive scenarios respectively in the condition of the size classification dataset (testing data to be classified), the size of training dataset, the size of neurons.

For processing parallelization, Yamamoto et al. [50] have introduced the process of converting the video frame to grey scale images by using Map-Reduce to extract some features from the video images. Bao et al. [51] compared Sun Grid Engine (SGE) and Map-Reduce performance in terms of processing duration in medical application.

Chebbi et al. [52] presented an approach for processing and storing large satellite images by using OTB remote sensing processing tools integrated into Hadoop Map-Reduce technique. Manojbhai et al. [53] introduced a model parallel execution that uses Hadoop Map-Reduce technique for tumor image pattern similarity. The main drawback of all parallelization approaches is that clustering management is hard and if the master node fails all the processing will break down. Moreover, Map-Reduce processes large datasets consumes more time to execute tasks. In addition, in Map-Reduce, data is processed and distributed over the cluster that increases time and reduces processing speed.

For Indexing approaches (see Fig.1), they can be divided into two sub-categories: LSH and its variant approaches [54-57], code index approaches [58-59]. For Locality Sensitive Hashing (LSH) and its variant approaches, a specific algorithm is used to hash similar input to the same buckets that are smaller of the number of input terms. For code index approaches, a specific method to create a code index to reduce the list of potential identities during classification.

For LSH and its variant, Indyk and Motwani [54] proposed the Locality Sensitive Hashing (LSH) algorithm and applied it to sub-linear time similarity searching context. In [55-57] a variant of LSH algorithm is proposed to reduce the target identities list. The main drawback of LSH approaches and its variant is that these algorithms it required quite a lot of memory storage to achieve a good performance.

For code index approaches, Ratha et al. [58] proposed different method for different biometric traits. For fingerprint recognition, they proposed to create a hash table of minutiae by putting the same minutiae in one row (code index) of the hash table and computing the most recurrent identities of each minutia to get the identity of the test template. For iris recognition, they proposed to compute the distance of a part of the iris code with all small part of the template (code index) database then compute the rotation, thereafter the distance between the whole iris code of the test template and selected images from the database is calculated. Aglika and Ross proposed in [59] a method to calculate a code index to select reference images and compare them with the test template for multimodal biometric databases. We believe that the main drawback of the hash table/code index approaches is that we are not sure that the right identities are selected during the first step of selection, which can influence the performance considerably.

For Hardware Solution [60-62], they can be divided in three categories: Compression, parallelization and hybrid approaches [62] that reduces time by distributing tasks overall memories or processing elements.

For Hardware Solution, in this context, Chen et al. [60] designed a materiel accelerator for large-scale DNNs and CNNs that take in consideration the impact of memory, performance and energy. Chen et al. [61] proposed a custom multi-ship machine learning architecture with a speedup of 450.65x and reduce energy 150.31x.

For hybrid approach, several categories of approaches described above can be combined. For example, Han et al. [62] proposed an Efficient Inference Engine that firstly prune the Alex-net model in a factor of 10x then reduce the weight sharing to only 4-bits per weight then fetch the small model to SRAM instead of DRAM to give a 120x energy saving and interleaving the matrix of weights over multiple processing elements for distributing the matrix-vector computation.

In this paper, we propose a hybrid method that combine the compress approach, parallelization and indexing techniques to choose the reference identities and uses as well the big data technologies to parallelize the comparison of identities to get as result the identity of the person in little time. Next section describes the proposed method.

3. METHODOLOGY

In this section, we present a hybrid approach for face identification, based on compression, parallelization and indexation. The main idea of the proposed approach is compressing a deep CNN model then extract discriminant features from it and distribute or/and indexed feature vectors afterward to speed up the identification time.

We firstly, input the test template in the CNN model to extract the pertinent features then we identify the identity of the entered image by choosing one of the three methods, which are Indexing Method, Processing Parallelization and Indexing with Parallelization Method. For feature extraction, we have compressed a VGGFace model [1] to extract discriminant features (see Figure.2).

The compressed VGGFace uses 8 layers instead of 16 as shown in Fig.2. We have passed from layer to another by adding respectively 2, 4, or multiplying by 6 the number of filters instead of multiplying by 2 and 8.

Our compressed VGGFace model for facial identification is implemented using a deep CNN architecture with eight layers comprising five convolutional layers and three fully connected layers. The choice of this network model is defined by the nature of the employed datasets and the peculiarities of the classification problem to solve. We also consider the network as a solution to reduce the problem of overfitting. Unlike the network architecture proposed in [1], During training, we preprocess the input face images by resizing them into 60x60 pixels before being fed to the network.

Furthermore, the image is passed through the eight-layer CNN-based architecture as follows:
The first convolutional layer learned 8, 3 x 3 kernels and a stride of 1 x 1, then followed by an activation layer (rectified linear unit (ReLU)) and a max-pooling layer. The second, third and fourth and fifth series of convolutional layers applied the same structure only with different filter sizes. Second convolutional layer learns 10, 5 x 5 filters, third is near identical to the second convolutional layers but with an increase in the number of filters to 12. The fourth convolutional layer set has a

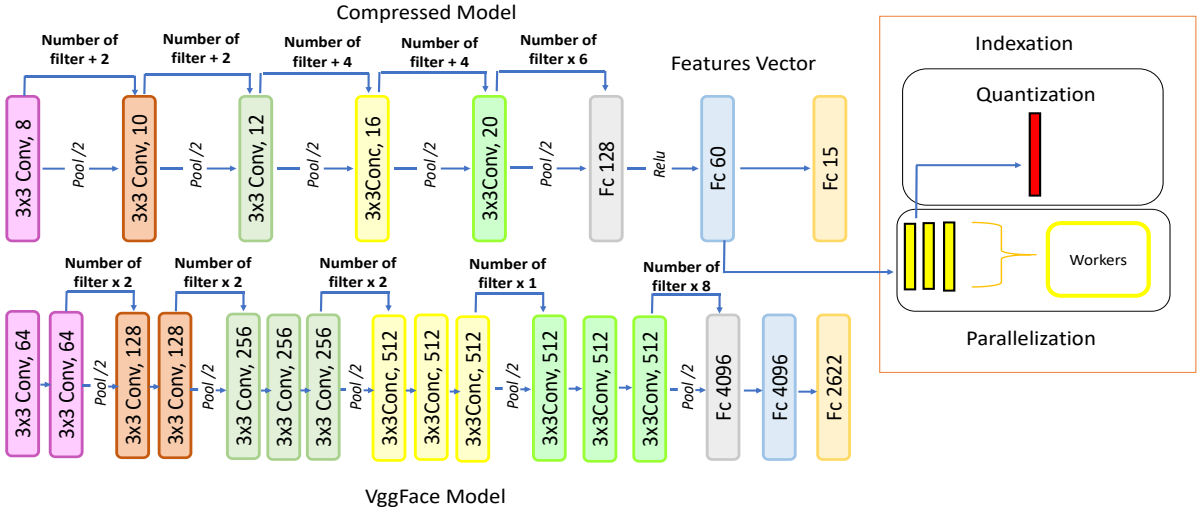

Figure 2 . Compressed CNN Model based on VGGFace Model

filter of 16 and the fifth convolutional layer set has a filter of 20. We have applied a Relu function and max polling after each convolutional layer to down sample the maps.

The composition of the first fully connected layer, contains 128 neurons, followed by a ReLU [63]. In the second fully connected layer, contains 60 neurons. In the final fully connected layer the output from the previous layer of size 60 features are densely mapped to 15 neurons for facial classification. A softmax with cross-entropy loss function is adopted to obtain a probability for each class. More details of the network architecture are given in Figure 2.

We get the feature vector 60-D by extracting the 7th layer of compressed model after training the model for 500000 iterations, and applying the Softmax of the 8th layer to compute loss, and get the accuracy of the model.
It is important to understand that like in any other neural network, a convolutional neural network also has the input data x which is an image and weight filters (i.e., W). Once the weights and the input image is convolved, we get the weighted output W * x and then we add the bias b as introduced in the equation (1), we use the convolution in five first layers with different number of filters that are respectively (8,10,12,16,20) with 3 as height and 3 as weight (3x3).

$$S = \sum_j W_{i,j} * x_j + b_i \quad (1)$$

We use the max pooling after each convolution layer to subsampling the size of the feature map to half and reduce the number of parameters. Max pooling is the most known sample-based discretization process. The objective is to reduce an input representation (image, hidden-layer output matrix, etc.) to reduce process the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

The output of max pooling is fed to one of the key functions, which is the RELU activation function (REctified Linear Unit). Relu function helps better train/ learn the model weights for the generalized case and add non-linearity to the learning model as introduced in equation (2). We use the Relu function after each max-pooling layer and at the first fully connected layer (6th layer) of a vector of 128 numbers.

$$Y = \mathrm{Max}(0, S) \quad (2)$$

To get the probability of each identity in the last layer of fully connected layer we use Softmax function which is a function that takes as input a vector of S real numbers, and normalizes it into a probability distribution consisting of S probabilities. That is, after applying softmax, each component will be in the interval (0,1) (3). We use a Softmax function in the last fully connected layer of 15 numbers (identity number) to classify a training image and get the performance of the model in each iteration.

$$Y = \frac{e^{S_i}}{\sum_j e^{S_j}} \quad (3)$$

To get the best performance with less error, we define the cost value. It gives the information about how far our model is from the desired value [64]. In other words, an error value should be minimized. For our compressed model, we have used Adam Optimizer with learning rate equal to 10-4, the batch size was set to 100 for both training and validation sets. The training not regularized by weight decay like VGGFace and without dropout regularization for the first two fully connected layers (dropout ratio set to 0.5) like VGGFace because the compressed model doesn't overfit. To do this, we have calculated the loss function by using cross entropy loss defined in (4).

$$L = -\sum_j y'^{(j)} \log \sigma(y)^{(j)} \quad (4)$$

-where "y" is the output of the last layer of the network (8 th layer).
-"y'" is the label of the one hot vector (putting bit 1 in the correct label and 0 for others).
-"$\cdot_{(j)}$ " denotes jth dimension of a given vector.
-"$\sigma(.)$" denotes probability estimate

After getting the best model for facial identification system, we extract features from the seventh layer of the compressed model than to get the identity, we should compare the feature vector of probe image to whole gallery features vectors in the database and choose the minimum distance between two vectors.

Our compressed model has a compression rate of 5639,28X. The VGGFace model is compressed from 145 million parameters (145002878) to 25713 parameters (26 K) and we have used 8 layers instead of 16 layers (see Table. I).

In the case of large scale databases, this comparison is time-consuming, because of that we propose three methods to reduce the identification time per probe, which are Indexing Approach, Processing Parallelization Approach and the combination of parallelization and indexation Approaches.

The main idea of indexing approach is to reduce the reference list that is compared with test template by comparing the quantized vectors of training templates with the test template then comparing it with the

TABLE I. CNN CONFIGURATIONS (SHOWN IN COLUMNS). THE CONVOLUTIONAL LAYER PARAMETERS ARE DENOTED AS "CONV(KERNEL SIZE)-(NUMBER OF CHANNEL)".

Compressed Model	Number of parameters	VGGFACE	Number of parameters	Compression rate
8 weight layers	25713	16 weight layers	145002878	5639,28X
Input (60 x 60 RGB image)		Input (224 x 224 RGB image)		
Conv3-8	224	Conv3-64 Conv3-64	38720	172,85X
MaxPool				
Conv3-10	730	Conv3-128 Conv3-128	221440	303,34X
MaxPool				
Conv3-12	1092	Conv3-256 Conv3-256 Conv3-256	1475328	1351,03X
MaxPool				
Conv3-16	1744	Conv3-512 Conv3-512 Conv3-512	5899776	3382,89X
MaxPool				
Conv3-20	2900	Conv3-512 Conv3-512 Conv3-512	7079424	2441,18X
MaxPool				
FC-128	10368	FC-4096	102764544	9911,70X
FC-60	7740	FC-4096	16781312	2168,12X
FC-15	915	FC-2622	10742334	11740,25X

completely training templates of reference identities (see figure 3).

The main idea of processing parallelization approach is to reduce the identification time by parallelizing the identities to each worker of a cluster (see fig.4).

The main idea of indexing with parallelization approach is to reduce the reference identities by using indexing approach while parallelizing these identities to each worker of a cluster by using processing parallelization approach.

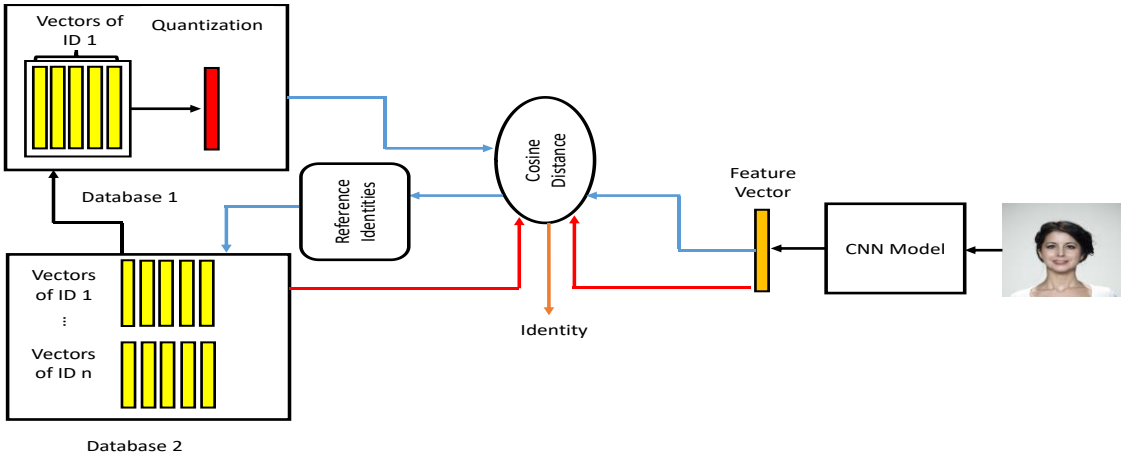

Figure 3 . Indexing Approach for identification phase

For Indexing Approach, we firstly create the first database from the second one (see figure 5), we take the whole feature vectors of each identity and we quantize them by using the Linde Buzo and Gray LBG algorithm [4] to get one quantized vector that represents well the identity.

An f-level x-dimensional quantizer is a mapping q, that assigns to each input vector, $s = (s_0, \dots, s_{x-1})$, a reproduction vector, $\hat{s} = q(s)$, drawn from a finite reproduction alphabet, $A = \{y_i;\ i = 1, \dots, f\}$.

The quantizer q is completely described by the reproduction alphabet (or codebook) A together with the partition, $S = \{S_i; i = 1, \dots, f\}$, of the input vector space into the sets $S_i = \{s: q(s) = y_i\}$ of input vectors mapping into the i_{th} reproduction vector (or codeword), such quantizers are also called block quantizers, vector quantizers, and block source codes.

Firstly, we get the feature vector of the training images by input them in the CNN model (Compressed VGGFace/ VGGFace) of each identity then we compute the distance between each identity (quantized Vector of each identity).

We select N identities with a minimum cosine distance equation (5) afterwards as reference identities.

$$Distance(X,Y) = 1 - \frac{X.Y}{\|X\|.\|Y\|} = 1 - \frac{\sum_{i=1}^{S} X_i Y_i}{\sum_{i=1}^{S}\sqrt{X_i^2}\sum_{i=1}^{S}\sqrt{Y_i^2}} \quad (5)$$

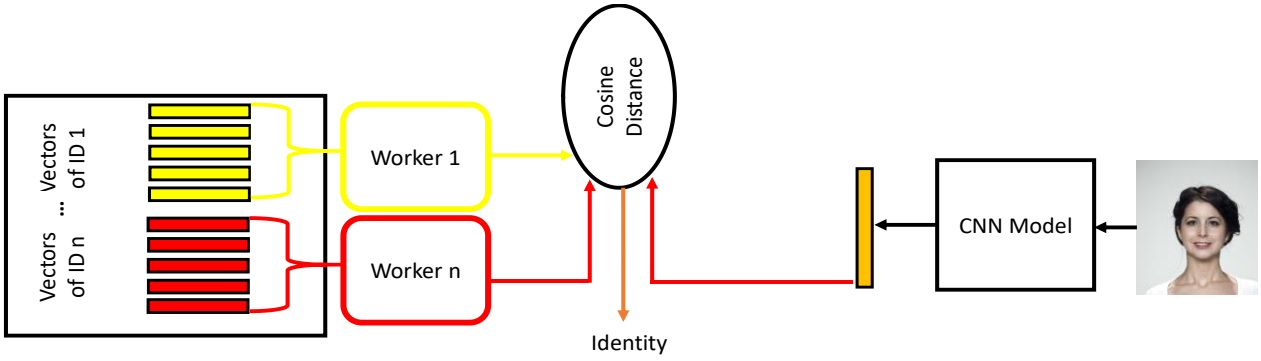

Figure 4 . Processing Distribution Approach for identification phase

Secondly, we compute the distance between the feature vector of the probe with hole feature vectors of reference identities (N=5/38 identities with minimum cosine distance) from Database 2 as shown in Fig.3 then we take the identity of a minimum distance between two vectors.

For processing parallelization Approach, we extract a feature vector from CNN Model, then each worker takes the whole feature vectors of one identity and compute the cosine distance between the training templates of this identity and test template then we take the minimum cosine distance with his/her identity as shown in Fig.5.

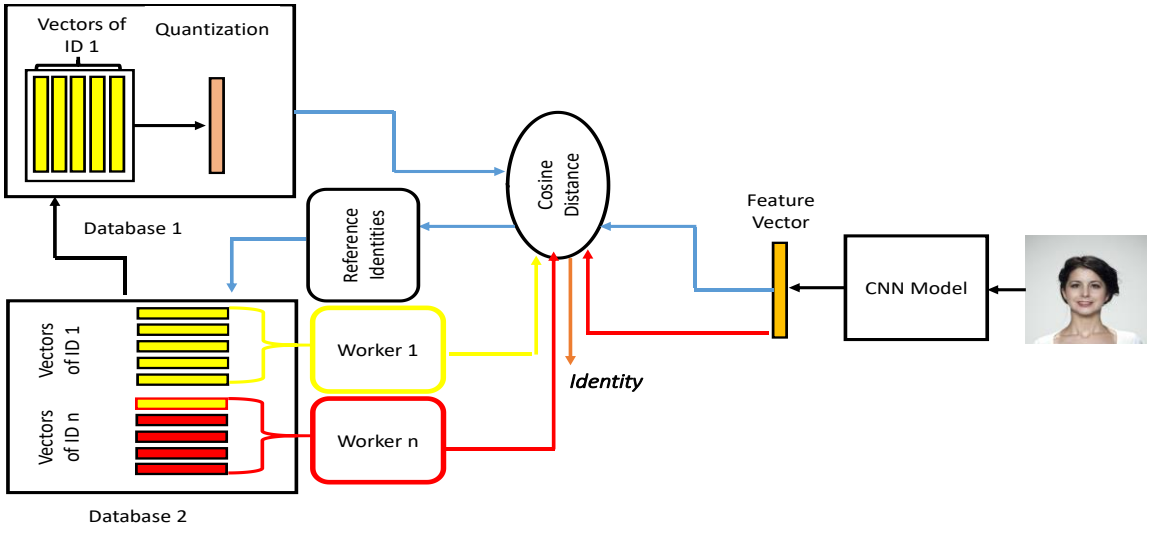

Figure 5. Hybrid proposed model architecture

Finally, we choose the minimum distance got from all workers with the equivalent identity as the identity of the test template.

For parallelization and indexation approach, we compute a quantized vector by using LBG algorithm then we compute the minimum cosine distance to get the N nearest identities of the test template, then each worker takes the hole training templates of one identity and compute the cosine distance (5) with the test template as shown in FIG.5.

Finally, each worker gives the minimum distance with his/her identity, then we choose the minimum distance with his/her identity as the identity of the test template.
In this case, we use the indexing method in the first part of the identification system then processing parallelization method in the second part of the system.

4. EXPERIMENTAL RESULTS

In this section, we evaluate the accuracy and identification time of several identification systems that use different CNN Models: our proposed hybrid model, VGGFace [1] and Inception v3 [65] is a CNN model contains 48 layers and can classify 1000 object categories such as animals, pencil, mouse.

We have used the Extended Yale faces B database [66] for training and evaluation the compressed model and we have also evaluate it by vidTIMIT database [67] for our experimentation.

a) Database Description and Evaluation

The vidTIMIT database [67] is contain video and audio recordings of 43 identities that used in many topics such as automatic lip reading, multi-view face recognition, multi-modal speech recognition and person identification, for each identity the person is moving his/her head to the left, right, back to the center, up, then down and finally return to the center. Thus, the used database comprises 102004 images that we have split into two parts as described in Table II.

TABLE II. VIDTIMIT DATABASE DISTRIBUTION

	Training	***Test***
Number of human subject	43	43
Number of images	1700	(3-2596)
Total	73100	28904

The extensive Yale Face database [68] contains 16128 images of 28 identities with 9 poses and 64 lighting conditions [69]. In our case, we are only interested in 15 identities and we have applied 17 transformations (e.g., Noise, Rotation, etc) for each identity to enhance the intra/inter-subject variations to make the database larger and the recognition task harder. Thus, the used database contains 81000 images. We have divides the database into two parts as described in Table.III.

TABLE III. DATABASE DISTRIBUTION OF AUGMENTATED YALE

	Training	***Test***
Number of human subject	15	15
Number of images	201	99
Number of transformations	17	17
Total	54270	26730

To evaluate our system, we have used the Closed-set Identification scenario [70] where only templates of enrolled identities are used for test.

The performance of the evaluation is calculated by the Rank-1 prediction, which is the right identity that belong with first minimum cosine distance. We have also calculated the speed-up of each system following four scenarios: 1) original systems, 2) original systems with indexation, 3) original systems with parallelization, 4) original system with indexation and parallelization.

The vidTimit database, is used in our experimentation. This database contains several multi-view images per identity with fixed background which makes the identification task not complicated.

To make the identification harder we have used an augmented Extended Yale Face B that contain different transformations (Scale, Rotation, Noise…).
The features and databases can be downloaded from this Github link: https://github.com/Majdouline-Meddad/CompressedVGGFace

b) Results and Discussion

For the first set of experiments with the VidTimit database as shown in Table IV (i.e., yellow columns), the identification time per image using VGGFace is equal to 2046s with a Rank-1 Prediction of 100%, and it is equal to 315s using the Inception V3 Model with a Rank-1 Prediction of 100% as well. However, after applying the compression for the VGGFace, the identification time per image becomes 11s with a Rank-1 Prediction of 99.9%, which means that the compression gives a gain of 186X the identification time using VGGFace, while keeping an excellent performance.

After applying the indexation approach with N=38 on the three original models (i.e., green columns), the identification time per image using the VGGFace Model becomes 1773s with a speed-up of 14.02% and a performance of 100%. For the Inception V3 Model, the identification time becomes 268s with a speed-up of 16.06% and a performance of 100% as well. Then, the identification time using the compressed VGGFace becomes 9s (i.e., gain of 197x) with a speed-up of 18.18% and a performance of 99.30%. These results prove that the

indexation approach decrease the execution time of the three models while keeping the same performance without indexation.

TABLE IV. OVERALL PERFORMANCE COMPARISON BETWEEN THE DIFFERENT CNN MODELS WITH THEIR IDENTIFICATION TIME

	VidTIMIT Database			*Extended Yale B Database*		
	Identification time per image	***Rank-1 Prediction***	***Speed-UP***	***Identification time per image***	***Rank-1 Prediction***	***Speed-UP***
Compressed VGGFace	11s	99,90%	0%	4,35s	96,48%	0%
Compressed VGGFace with indexing method	9s	99,30%	18,18%	1,45s	96,50 %	66.66%
Compressed VGGFace with parallelization method	5s	99,90%	45,45%	1.37s	96,48%	68,50%
Hybrid Approach	3s	99,30%	72,72%	1,12s	96,50 %	74.25%
Inception V3	315s	100%	0%	9,14s	74,10 %	0%
Inception V3 with indexing method	268s	100%	16,06%	3,41s	74,03%	62,69%
Inception V3 with parallelization method	278s	100%	11,74%	3,91s	74,10 %	57,22%
Inception V3 with indexing+distribution method	165s	100%	38,25%	2,87s	74,03 %	68,60%
VGGFACE	2046s	100%	0%	950s	99,82%	0%
VGGFACE with indexing method	1773s	100%	14.02%	362s	99,79%	61.89%
VGGFACE with parallelization method	1860s	100%	9,55%	450s	99,82%	52,63%
VGGFACE with indexation and distribution method	1415s	100%	32,42%	320s	99,79%	66,31%

After applying the parallelization approach with a number of thread equal to 4 on the three original models (i.e., blue columns), the identification time per image using the VGGFace Model becomes 1860s with a speed-up of 9.55% and always a performance equal to 100%. For the inception V3 Model, the identification time becomes 278s with a speed-up of 11,74% and a performance of 100%. Then, for the compressed VGGFace, the identification time decreased to 5s (i.e., gain of 372x) with a speed-up of 45.45% and a performance of 99.90%. We can conclude these results that the parallelization decrease as well the identification time, hence the idea to combine the indexation and parallelization.

Finally, after applying the indexation and the parallelization on the three models (i.e., brown columns), the identification time per image using the VGGFace Model becomes 1415s with a speed-up of 32.42% and a performance equal to 100%. For the Inception V3 Model, the identification time becomes 165s with a speed-up of 38,28% and a performance of 100%. Then for our hybrid approach, the identification time decreased to 3s (i.e., gain of 471x) with a speed-up of 72.72% and a performance of 99.30%. The results prove that the proposed approach significantly decrease the execution time while keeping an excellent performance.

For the second set of experiments with the Extended yale Face B database, as shown in Table IV, the identification time per image using the VGGFace is equal to 950s with a Rank-1 Prediction of 99.82%. For the

inception V3 Model, the identification time is equal to 9.14s and a performance of 74.10%. Then, for our compressed VGGFace, the identification time decreased to 4.35s with a speed-up while keeping a performance of 96.48%. We can conclude from these results that, in the presence of intra/inter-subject variations, the Inception V3 Model lost the performance while the VGGFace and the compressed VGGFace were able to conserve it.

After applying the indexation approach with N=5 on the three original models, the identification time per image using VGGFace Model becomes 362s with a speed-up of 61,89% and a performance of 99.79%. For the inception V3 Model, the identification time becomes 3.41s with a speed-p of 62,69% and a stabilized performance at 74.03%. Then, the identification time using the compressed VGGFace becomes 1.45s (i.e., gain of 249.65x) with speed-up of 66.66% and a performance of 96.46%. These results confirm, like the first set of experiments, that the indexation approach decrease the execution time of the three models.

After applying the parallelization approach with a number of thread equal to 4 on the three original models, the identification time per image using VGGFace Model becomes 450s with a speed-up of 52.63% and always the same performance of 99.82%. For the Inception V3 Model, the identification time becomes 3.91s with a speed-up of 57.22% and a performance of 74.10%. Then, for the compressed VGGFace, the identification time decreased to 3.91s (i.e., gain of 328.46x) with a speed-up of 68.50% and a performance of 96.48%. These resultas prove again that the parallelization decrease as well the identification time.

Finally, after applying the indexation and the parallelization on the three models, the identification time per image using the VGGFace Model becomes 320s with a speed-up of 66.31% and a performance equal to 99.79%. For the Inception V3 Model, the identification time becomes 2.87s with a speed-up of 68.60% and a performance of 74.03%. Then, for out hybrid approach, the identification time decreased to 1.12s (i.e., gain of 378.88x) with a speed-up of 74.25% and a performance of 96.50%. These results prove that the proposed approach significantly decrease the execution time while keeping an excellent performance even in the presence of intra/inter-subject variations.

We would like to mention that the results shown in Table.IV are get from a laptop i5 with 2.4HZ of the CPU frequency and 4 Go of Ram Memory.

5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new hybrid face identification system based on the compression of the VGGFace model for the feature extraction step, and on the indexation and parallelization for the identification task, the proposed system has been evaluated according to the time of identification per image, the speed-up and the Rank-1 accuracy. We have compared the proposed model with the original VGGFace model and the Inception V3 Model. Our experimental results indicate that the proposed system decrease the identification time and conserves an excellent performance, which make it suitable for embedded devices. In our future work, we plan to create a compressed CNN model for medical image analysis.

ACKNOWLEDGMENT

Majdouline Meddad acknowledges the financial support of the "Centre National pour la Recherche Scientifique et Technique" CNRST, Morocco.

REFERENCES

[1] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks For Large-Scale Image Recognition. ICLR 2015, 2015.

[2] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In CVPR, pages 1701–1708, 2014

[3] Avita Katal, Mohammad Wazid, R. H. Goudar, Big data: Issues, challenges,tools and Good practices. 2013 Sixth International Conference on Contemporary Computing (IC3), 2013.

[4] YOSEPH LINDE, Andres Buzo, Robert M. Gray, An Algorithm for Vector Quantizer Design. IEEE TRANSACTIONS ON COMMUNICATIONS, 1980.

[5] Desai Devanshi Manojbhai, Kodrani Kajal Pradipkumar, R. Raj Amenakshi, Big Image Analysis for Identifying Tumor Pattern Similarities. International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), 2016.

[6] Yu Cheng, Duo Wang, Pan Zhou, Tao Zhang, A Survey of Model Compression and Acceleration for Deep Neural Networks. IEEE SIGNAL PROCESSING MAGAZINE, SPECIAL ISSUE ON DEEP LEARNING FOR IMAGE UNDERSTANDING, 2019.

[7] TAL BEN-NUN, Torsten Hoefler, Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. arXiv:1802.09941v2 [cs.LG], 2018.

[8] Y. Gong, Liu Liu, Ming Yang, Lubomir Bourdev ,Compressing deep convolutional networks using vector quantization. CoRR, vol. abs/1412.6115, 2014.

[9] Y. W. Q. H. Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, Jian Cheng, Quantized convolutional neural networks for mobile devices. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[10] V. Vanhoucke, Andrew Senior, Mark Z. Mao , Improving the speed of neural networks on cpus. Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011, 2011.

[11] S. Gupta, Ankur Agrawal, Kailash Gopalakrishnan, Pritish Narayanan, Deep learning with limited numerical precision. Proceedings of the 32Nd International Conference on International Conference on Machine Learning, 2015.

[12] S. Han, Huizi Mao, William J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. International Conference on Learning Representations (ICLR), 2016.

[13] Y. Choi, Mostafa El-Khamy, Jungwon Lee, Towards the limit of network quantization. CoRR, vol. abs/1612.01543, 2016.

[14] M.Courbariaux,YoshuaBengio, Jean-Pierre David, Binaryconnect: Training deep neural networks with binary weights during propagations. Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, 2015.

[15] M.Courbariaux, Itay Hubara, Daniel oudry, Ran El-Yaniv, Yoshua Bengio, Binarized neural network: Training deep neural networks with weights and activations constrained to +1 or -1. CoRR, vol. abs/1602.02830, 2016.

[16] M. Rastegari, Vicente Ordonez, Joseph Redmon, Ali Farhadi ,Xnor-net: Imagenet classification using binary convolutional neural networks. ECCV, 2016.

[17] Lu. Hou, Quanming Yao, James T. Kwok, Loss-aware binarization of deep networks. CoRR, vol. abs/1611.01600, 2016.

[18] S. Han, H. Mao, and W. Dally. Deep compression: Compressing DNNs with pruning, trained quantization and huffman coding. arxiv:1510.00149v3, 2015.

[19] Forrest N. Iandola1, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer. SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE. arxiv: 1602.07360, 2016.

[20] S. J. Hanson, Lorien Y. Pratt, Comparing biases for minimal network construction with back-propagation. in Advances in Neural Information Processing Systems 1, D. S. Touretzky, Ed., 1989.

[21] Y. L. Cun, J. S. Denker, and S. A. Solla, Advances in neural information processing systems 2. D. S. Touretzky, Ed., 1990.

[22] B. Hassibi, David G. Stork, Second order derivatives for network pruning: Optimal brain surgeon. Advances in Neural Information Processing Systems 5, 1993.

[23] S. Srinivas, R. Venkatesh Babu, Data-free parameter pruning for deep neural networks. in Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015.

[24] W. Chen, James T.Wilson, Stephen Tyree, Kilian Q. Weinberger, Yixin Chen, Compressing neural networks with the hashing trick. JMLR Workshop and Conference Proceedings, 2015.

[25] K. Ullrich, Edward Meeds, Max Welling, Soft weight-sharing for neural network compression. CoRR, vol. abs/1702.04008, 2017.

[26] V. Lebedev, Victor Lempitsky, Fast convnets using group-wise brain damage. 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, 2016.

[27] Hao. Li, Asim Kadav, Igor Durdanovic, Hanan Samet, Hans Peter Graf, Pruning filters for efficient convnets. CoRR, vol. abs/1608.08710, 2016.

[28] Sergey Ioffe, Christian Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning, 2015.

[29] Yu. Cheng, Felix X. Yu, Rogerio S. Feris, Sanjiv Kumar, Alok Choudhary, Shih-Fu Chang, An exploration of parameter redundancy in deep networks with circulant projections. International Conference on Computer Vision(ICCV), 2015.

[30] Y. Cheng, Felix X Yu, Rogerio S. Feris, Sanjiv Kumar, Alok Choudhary, S. Chang Fast neural networks with circulant projections. CoRR, vol. abs/1502.03436, 2015.

[31] Z. Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, Ziyu Wang, Deep fried convnets. International Conference on Computer Vision (ICCV), 2015.

[32] V. Sindhwani, Tara N. Sainath, Sanjiv Kumar, Structured transforms for small-footprint deep learning. in Advances in Neural Information Processing Systems 28, 2015.

[33] J. Chun, T. Kailath, Generalized Displacement Structure for Block- Toeplitz, Toeplitz-block, and Toeplitz-derived Matrices. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991.

[34] M. V. Rakhuba, I. V. Oseledets, Fast multidimensional convolution in low-rank tensor formats via cross approximation. SIAM J. Scientific Computing, 2015.

[35] M. Moczulski, Misha Denil, Jeremy Appleyard, Nando de Freitas, Acdc: A structured efficient linear layer. International Conference on Learning Representations (ICLR), 2016.

[36] M. Jaderberg, Andrea Vedaldi, Andrew Zisserman, Speeding up convolutional neural networks with low rank expansions. Proceedings of the British Machine Vision Conference. BMVA Press, 2014.

[37] Yury Vizilter, Vladimir Gorbatsevich , Andrey Vorotnikov, Nikita Kostromov, Real-Time Face Identification via Multi-convolutional Neural Network and Boosted Hashing Forest, Deep Learning for Biometrics. Advances in Computer Vision and Pattern Recognition, 2017.

[38] V. Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, Victor Lempitsky , Speeding-up convolutional neural networks using fine-tuned cpdecomposition. CoRR, vol. abs/1412.6553, 2014.

[39] C. Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, Weinan E, Convolutional neural networks with low-rank regularization. vol. abs/1511.06067, 2015.

[40] T. S. Cohen, Max Welling Group equivariant convolutional networks. arXiv preprint arXiv:1602.07576, 2016.

[41] H. Li, Wanli Ouyang, Xiaogang Wang, Multi-bias non-linear activation in deep neural networks. arXiv preprint arXiv:1604.00676, 2016.

[42] S. Zhai, Yu Cheng, Weining Lu, Zhongfei Zhang, Doubly convolutional neural networks. Advances In Neural Information Processing Systems, 2016.

[43] C. Bucila, Rich Caruana, Alexandru Niculescu-Mizil, Model compression. in Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006.

[44] G. E. Hinton, Oriol Vinyals, Jeff Dean ,Distilling the knowledge in a neural network. CoRR, vol. abs/1503.02531, 2015.

[45] P. Luo, Zhenyao Zhu1*, Ziwei Liu1, Xiaogang Wang2,3, and Xiaoou Tang1, Face model compression by distilling knowledge from neurons. Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[46] T. Chen, Ian Goodfellow, Jonathon Shlens , Net2net: Accelerating learning via knowledge transfer. CoRR, vol. abs/1511.05641, 2015.

[47] Kairan Sun, Xu Wei, Gengtao Jia, Risheng Wang, Ruizhi Li, Large-scale Artificial Neural Network: MapReduce-based Deep Learning. Project Report for Cloud Computing and Storage. Dept. of Electrical and Computer Engineering, University of Florida, 2015.

[48] [45] Nada Basit, Yutong Zhang, Hao Wu, Haoran Liu, Jieming Bin, Yijun He, Abdeltawab M. Hendawi, MapReduce-based Deep Learning With Handwritten Digit Recognition Case Study. 2016 IEEE International Conference on Big Data (Big Data), 2016.

[49] [46] Yang Liu, Jie Yang, Yuan Huang, Lixiong Xu, Siguang Li, and Man Qi , MapReduce Based Parallel Neural Networks in Enabling Large Scale Machine Learning . Hindawi Publishing Corporation Computational Intelligence and Neuroscience, 2015.

[50] Muneto Yamamoto, Kunihiko Kaneko, Parallel Image Database Processing With Mapreduce and Performance Evaluation in Pseudo Distributed mode. International Journal of Electronic Commerce Studies, 2012.

[51] [48] Shunxing Bao, Weitendorf FD, Plassard AJ, Huo Y, Gokhale A, Landman BA.Theoretical and Empirical Comparison of Big Data Image Processing with Apache Hadoop and Sun Grid Engine .Medical Imaging 2017: Imaging Informatics for Healthcare Research and Applications, 2017.

[52] I. Chebbi, W. Boulila, I. R. Farah, Improvement of Satellite Image Classification : Approach Based on Hadoop/Map Reduce, 2nd International Conference on Advanced Technologies for Signal and Image Processing - ATSIP'2016, 2016.

[53] Desai Devanshi Manojbhai, Kodrani Kajal Pradipkumar, R. Raj Amenakshi, Big Image Analysis for Identifying Tumor Pattern Similarities. International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), 2016.

[54] Piotr Indyk, RAJEEV MoTwani, Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. Proceedings of the 30th Symposium on Theory of Computing, 1998.

[55] Aristides Gionis, Piotr Indyk, Rajeev Motwani, Similarity Search in High Dimensions via Hashing. Proceedings of the 25th VLDB conference, 1999.

[56] Naser Damer, Philipp Terhörst, Andreas Braun, Arjan Kuijper, Fingerprint And Iris Multi-Biometric Data Indexing and Retrieval. 2018 21st International Conference on Information Fusion (FUSION) , 2018.

[57] M. Bawa, Tyson Condie, Prasanna Ganesan, Lsh forest: Self-tuning indexes for similarity search. Proceedings of the 14th International Conference on World Wide Web, 2005.

[58] N. K. Ratha, Jonathan Connell, S. Pankanti, Big Data Approach to biometric-based identity analytics . IBM Jounal of Research and Development, 2015.

[59] Aglika Gyaourova , Arun Ross,. A Coding Scheme for Indexing Multimodal Biometric Databases Proc. of IEEE Computer Society Workshop on Biometrics at the Computer Vision and Pattern Recogniton (CVPR) conference, 2009.

[60] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning, in ASPLOS, 2014.

[61] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam. Dadiannao: A machine-learning supercomputer, in MICRO, 2014.

[62] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram Mark A. Horowitz, William J. Dally. EIE: Efficient Inference Engine on Compressed Deep Neural Network, arxiv:1602.01528v2, 2016.

[63] Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In NIPS, pp. 1106–1114, 2012.

[64] Aamir Khan, Hasan Farooq, Principal Analysis-Linear Discriminant Analysis Feature Extraction for Pattern Recognition. IJCSI International Journal of Computer Science Issues, 2011.

[65] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna Rethinking the Inception Architecture for Computer Vision. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[66] K.C. Lee, J. Ho, D. Kriegman: Extended Yale Face B. http://vision.ucsd.edu/ leekc/ExtYaleDatabase/ExtYaleB.html, 2016.

[67] C. Sanderson and B.C. Lovell. Multi-Region Probabilistic Histograms for Robust and Scalable Identity Inference. Lecture Notes in Computer Science (LNCS), Vol. 5558, pp. 199-208, 2009.

[68] Georghiades A.S., Belhumeur P.N., Kriegman, D.J.: 'From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose'.IEEE Transaction on Pattern Analysis and Machine Intelligence, 2001.

[69] K.C. Lee, J. Ho, D. Kriegman: 'Acquiring Linear Subspaces for Face Recognition under Variable Lighting'.IEEE Transaction on Pattern Analysis and Machine Intelligence, 2005.

[70] Stan Z.Li, Anil. K. Jain, Handbook of face recognition. Springer Science+Business Media, 2005.

[71] Amit B, Chinmay C, Anand K, Debabrata B, Emerging trends in IoT and big data analytics for biomedical and health care technologies, Elsevier: Handbook of Data Science Approaches for Biomedical Engineering, Ch. 5, 2019.

Majdouline Meddad is a Ph.D Student at the IT and Telecommunications Research Laboratory of Mohamed V University in Rabat, Morocco. Her research field is the artificial intelligence. The current work focuses on biometrics and compressing of convolutional neural network models.

Chouaib Moujahdi is an Associate Professor at the Scientific Institute of Rabat. He received both the Master's and the Ph.D. degrees in Computer Science and Telecommunications from Mohammed V University. He was a Fulbright visiting student at University of Nevada - Reno between 2012 and 2014. His research interests include Biometrics and Pattern Recognition. Current work focuses on "Biometric Security Protection" and on "Automated Plant Species Identification".

Mounia Mikram is an Associate Professor of computer sciences and mathematics at the School of Information Sciences, Rabat since 2010. She received her master degree from Mohammed V University Rabat (2003) and her PhD degree from Mohammed V University, Rabat, and Bordeaux I University (2008). Her research interests include pattern recognition, computer vision, and biometrics security systems and artificial intelligence.

Mohammed Rziza is Received his national Doctorate in engineering sciences, image processing specialty, from the Faculty of Science of the Mohammed V-Agdal University, Rabat, Morocco, in 2002. He joined the Faculty of Science, Rabat, Morocco, in 2003, as an assistant professor. Since 1997, he is a member of the GSCM group. His research interests include image processing, pattern recognition, and stereovision.