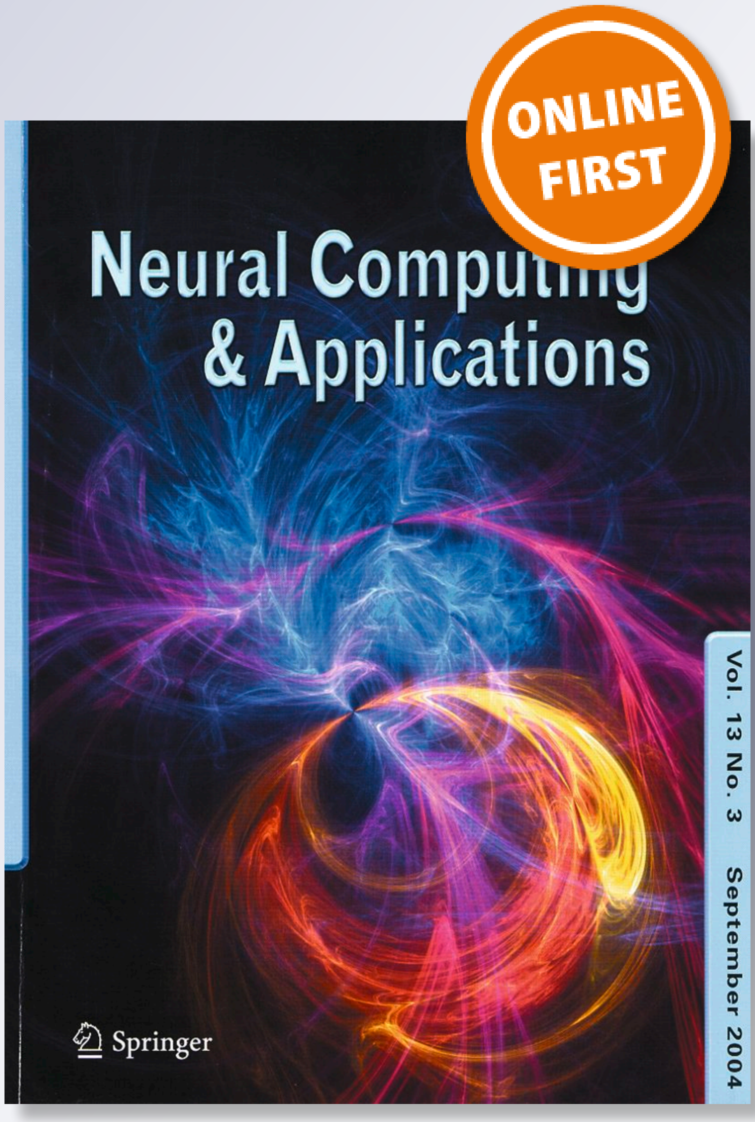
ONLINE FIRST
Neural Computing & Applications
Vol. 13 No. 3 September 2004
Springer

Neural Comput & Applic
DOI 10.1007/s00521-012-1129-5

ORIGINAL ARTICLE

Compressing arrays of classifiers using Volterra-neural network: application to face recognition

M. Rubiolo · G. Stegmayer · D. Milone

Received: 11 January 2012 / Accepted: 3 August 2012

Abstract Model compression is required when large models are used, for example, for a classification task, but there are transmission, space, time, or computing constraints that have to be fulfilled. Multilayer perceptron (MLP) models have been traditionally used as classifiers. Depending on the problem, they may need a large number of parameters (neuron functions, weights, and bias) to obtain an acceptable performance. This work proposes a technique to compress an array of MLPs, through the weights of a Volterra-neural network (Volterra-NN), maintaining its classification performance. It will be shown that several MLP topologies can be well-compressed into the first-, second-, and third-order (Volterra-NN) outputs. The obtained results show that these outputs can be used to build an array of (Volterra-NN) that needs significantly less parameters than the original array of MLPs, furthermore having the same high accuracy. The Volterra-NN compression capabilities were tested for solving a face recognition problem. Experimental results are presented on two well-known face databases: ORL and FERET.

Keywords Model compression · Array of neural networks · Volterra-neural network · Face recognition

Electronic supplementary material The online version of this article (doi:10.1007/s00521-012-1129-5) contains supplementary material, which is available to authorized users.

M. Rubiolo (✉) · G. Stegmayer
CONICET, CIDISI-UTN-FRSF, Lavaise 610,
3000 Santa Fe, Argentina
e-mail: mrubiolo@gmail.com; mrubiolo@santafe-conicet.gov.ar

D. Milone
CONICET, SINC(i)-FICH-UNL, Ciudad Universitaria,
RN 168 Km. 472.4, 3000 Santa Fe, Argentina

1 Introduction

The purpose of model compression is to find a fast and compact model to approximate a function learned by, for example, a classifier. Moreover, it is desirable to achieve this without significant loss in performance [3]. Often the best performing models that use supervised learning are combinations of complex and large classifiers. However, in some situations, it is not enough for a classifier to be highly accurate; it also has to meet some requirements regarding on-line execution time, storage space, and limited computational power [26].

It is well-known that multilayer perceptron (MLP) models are usually considered as a powerful classification model. However, they easily become large models just by the addition of neurons, for example, in the hidden layer, which has a direct effect over the number of model weights. Similarly, the introduction of more information to the model, that is to say more input variables in order to better learn the training data and to improve its classification ability, can cause an increment of the model complexity.

It has been recently shown that a Volterra model can be extracted from the parameters of a trained neural network (NN) [23]. The Volterra model is formed by Volterra kernels, which are associated with the parameters of the trained NN. This has proven to be particularly useful for reproducing the nonlinear and dynamic behavior of new wireless communications devices [18]. Moreover, in Rubiolo et al. [21], the Volterra kernels extraction procedure has been used to build a Volterra-neural network (Volterra-NN) model, which is a compressed version of a trained MLP model over a very simple classification task, maintaining the same recognition rate than the original MLP model but with fewer parameters. In this work, we propose that, since it is possible to obtain a Volterra-NN model from a single trained MLP model, it is also possible to compress an array of MLPs (*a*MLP) using Volterra-NNs. That is to say, the same methodology applied to build a Volterra-NN model from a single MLP can be used to obtain an array of Volterra-NN models from an *a*MLP.

In fact, arrays and ensembles of neural classifiers have proven to reach significant better results in classification problems than single models [7, 20], in particular for the task of face recognition (FR) [26]. In these systems, the first step consists of face detection through image processing techniques. Secondly, a feature extraction method is applied to extract useful information of the face. Finally, this information is used in a classifier for recognizing faces [27]. In Capello et al. [4], an *a*MLP model was proposed for the classification task within a FR system. Specifically, an array of NNs have been used for classification, consisting of one MLP for each subject (valid or authorized person), with a final decision made over the network outputs of the complete array. The classification was performed by the maximum output calculation among all the networks outputs. This configuration has achieved significant improvements over the performance of a classic MLP. However, the use of this new neural configuration implies much more parameters, and therefore, a larger and more complex FR system. Due to the fact that FR models should be able to run on small processing devices such as mobile phones, ipods, and security cams, or to be transmitted online, the model must have an appropriate size in order to adjust to these requirements.

This work presents a novel approach for compressing a face recognition model based on a novel application of the Volterra-NN method to an array of MLPs. It will be shown how an *a*MLP model that has learnt a classification problem with a certain (high) accuracy can be compressed into a more compact representation using a Volterra-NN model. This novel representation involves less parameters, maintaining, however, a high recognition accuracy. Two different face recognition databases have been used to show the effectiveness of the proposed method.

The paper is organized as follows. Section 2 explains in detail the proposed Volterra-NN model and its use as a classifier in a face recognition system. The materials and methods used in the study are presented on Sect. 3. Results of the model evaluation through two public face databases as well as a discussion of the experiments are shown in Sect. 4. Finally, Sect. 5 presents the conclusions and future work.

2 Volterra-neural network for neural networks compression

The Volterra series and Volterra theorem was developed in 1887 by Vito Volterra. It is a model for representing nonlinear dynamic behavior frequently used in system identification [25]. In Stegmayer and Chiotti [23], several formulas for the extraction of Volterra weights, independently of the neural model topology, number of variables involved in the problem, and nonlinearity of the system have been presented. The equations are based on architectures having an hyperbolic tangent activation functions in the hidden nodes, trained with a classical backpropagation algorithm [16], for multi-input, multi-output systems. The MLP model is trained using the available training data and, after that, the Volterra weights are obtained from the trained network parameters.

This section presents the Volterra-NN model for *a*MLP model compression, extending the simple algorithms for classification proposed in Rubiolo et al. [21]. The following subsection presents the neural model used for face recognition; after that, some basics concepts on Volterra models necessary to understand the proposed approach are explained. Finally, it is shown how the Volterra-NN can perform as a classifier when an *a*MLP is used.

2.1 Neural models for classification

Arrays and ensembles of single MLP networks have proven to reach significant better results in classification problems than single models [1, 2, 4]. The classifier model is an array of MLPs where there is one MLP model for each class k to be identified, with $k = 1 \ldots K$, being K the total number of classes. Therefore, the *a*MLP model is formed by K networks like the one shown in Fig. 1. Each network output takes a value of 1 if the class is identified or 0 otherwise. The first layer of each MLP in the *a*MLP is a set of N_I input neurons, where each input is an eigenspace vector and there are N_H hidden neurons. When a picture has to be classified, its projected eigenspace vector is used as input for the classifier, that is to say, it is presented to all the k networks defined for the *a*MLP model, and the maximum output obtained among all network outputs is assigned as class label. If a pattern of the kth-class has been presented to the model, a value of (near) 1 is expected at the kth network.

The application of the Volterra weights extraction method for model compression of a classical MLP classifier was presented in Rubiolo et al. [21]. It was shown how a MLP model that has learnt a classification problem with a certain (high) accuracy can be compressed into a more compact representation using a Volterra model and its parameters, named Volterra weights. Several MLP topologies can be well-compressed into the first-, second-, and third-order Volterra weights, which can be used to build a Volterra model that needs less parameters than the MLP model and, at the same time, has a similar high accuracy. This work proposes to apply and extend the compression

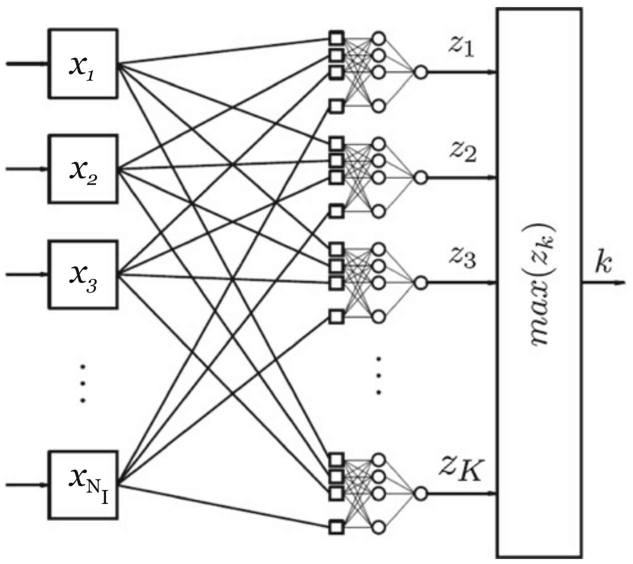

Fig. 1 Array of MLPs (*a*MLP) model for classification

procedure based on Volterra models to an array of MLPs to solve a face recognition problem.

The first stage of a complete automatic FR system is face detection. Once detected, the face must be represented through an appropriate feature extraction method. A global representation can be done through a well-known technique such us the eigenfaces [24] by applying principal component analysis (PCA) for dimensionality reduction [11], which is the most widely used feature extraction method for face recognition [14]. The final stage consists of the classification carried out by using an appropriate classifier, which be a very simple method, such us the Euclidean distance or k-nearest-neighbors, or an array or ensemble solution based on hundred of weak classifiers, being MLP is one of the most popular models [12, 17]. The next subsection presents the details of the new algorithm proposed for extraction of the Volterra weights from an array of MLPs used for classification, as well as the procedure used to obtain different-order Volterra-NN outputs.

Algorithm 1 Volterra weights extraction

```
   Data:
      D: training data
   Results:
      $v^{(0)}$: 0-order Volterra weight
      $\mathbf{v}^{(1)}$: $1^{st}$-order Volterra weigths
      $\mathbf{v}^{(2)}$: $2^{nd}$-order Volterra weigths
      $\mathbf{v}^{(3)}$: $3^{rd}$-order Volterra weigths
1  begin
2      $M \leftarrow$ train MLP model with $D$
3      $v^{(0)} \leftarrow$ calculate zero-order Volterra weight from $M$ using (1)
4      for $1 \le i \le N_I$ do
5          $v_i^{(1)} \leftarrow$ calculate first-order Volterra weight from $M$ using (2)
6      Assign each $v_i^{(1)}$ extracted to $\mathbf{v}^{(1)}$
7      for $1 \le i \le N_I$ do
8          for $1 \le j \le N_I$ do
9              $v_{i,j}^{(2)} \leftarrow$ calculate second-order Volterra weight from $M$ using (3)
10     Assign each $v_{i,j}^{(2)}$ extracted to $\mathbf{v}^{(2)}$
11     for $1 \le i \le N_I$ do
12         for $1 \le j \le N_I$ do
13             for $1 \le k \le N_I$ do
14                 $v_{i,j,k}^{(3)} \leftarrow$ calculate third-order Volterra weight from $M$ using (4)
15     Assign each $v_{i,j,k}^{(3)}$ extracted to $\mathbf{v}^{(3)}$
16 end
```

2.2 Volterra-NN model forward computation

Stegmayer and Chiotti [23] have derived equations that allow the calculation of any Volterra kernel order using the weights and hidden neurons bias values of a NN that has been trained on a problem using hyperbolic tangent hidden functions. The following formulas, instead, allow the calculus of zero, first-, second-, and third-order Volterra kernels from a trained MLP having sigmoidal hidden units:

$$h_0 = b_o + \sum_{h=1}^{N_H} w_h^2 \frac{1}{(1+e^{-b_h})} \tag{1}$$

$$h_1(\cdot) = \sum_{h=1}^{N_H} w_h^2 w_{h,i}^1 \frac{e^{-b_h}}{(1+e^{-b_h})^2} \tag{2}$$

$$h_2(\cdot) = \sum_{h=1}^{N_H} w_h^2 w_{h,i}^1 w_{h,j}^1 \frac{\frac{e^{-b_h}(e^{-b_h}-1)}{(1+e^{-b_h})^3}}{2!} \tag{3}$$

$$h_3(\cdot) = \sum_{h=1}^{N_H} w_h^2 w_{h,i}^1 w_{h,j}^1 w_{h,k}^1 \frac{\frac{-e^{-b_h}(-e^{-2b_h}+4e^{-1}-1)}{(1+e^{-b_h})^4}}{3!} \tag{4}$$

where N_H is the number of hidden neurons, N_I is the number of input neurons, and w_h and b_h are the weight and the bias associated with a hidden sigmoidal neuron, respectively; for $i,j,k = [1,\cdots,N_I]$. These formulas are easilly extended to any kernel order. Due to space restrictions, only up to third order is shown. To simplify the notation, the Volterra weights will be defined from now on as follows: $v^{(0)} = h_0, v_i^{(1)} = h_1(\cdot), v_{i,j}^{(2)} = h_2(\cdot)$ and $v_{i,j,k}^{(3)} = h_3(\cdot)$. Algorithm 1 shows in detail the Volterra weights extraction procedure, in which it is possible to compute the zero, first-, second-, and third-order Volterra weights. The input is the training data, and the outputs are the Volterra weights. The first step consists of training a MLP classifier model with the training data D as it is possible to see in line 2. From the trained neural model M, the Volterra weights can be calculated. According to (1), the zero-order Volterra weight is obtained (line 3). Similarly, by applying (2), (3), and (4), it is possible to compute the first (line 9), second (line 8),and third-order (line 7) Volterra weights, respectively.

The different-order Volterra-NN (V-NN) models that can compress a MLP are depicted graphically in Fig. 2. The boxes represent the input variables (in the example, x_1 and x_2), and the arrows that join them symbolize their product with their corresponding Volterra weights.

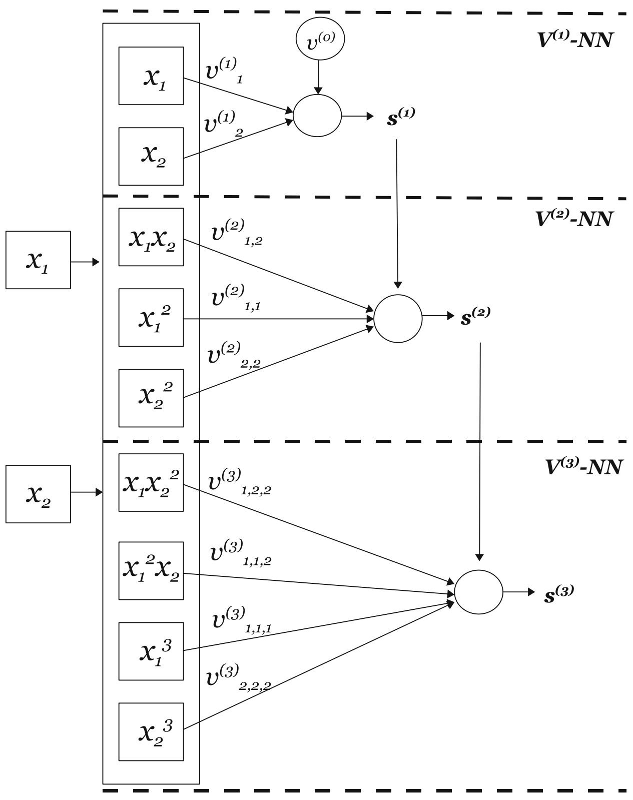

Fig. 2 Example of the topology of a Volterra-NN model for 2 input variables and three V-NN outputs

The white circles act as summarizing all the products between input variables and Volterra weights, plus the $(n-1)$-order V-NN model ($V^{(n-1)} - NN$). As a result, a new nth-order Volterra model ($V^{(n)} - NN$) is obtained. It is important to highlight that the different cross-products combinations between the input variables are shown in the figure inside a rectangular box, in an effort to clarify the process for obtaining the different Volterra outputs. This V-NN model can be similarly applied to any number of inputs.

The output of the second-order V-NN model ($V^{(1)} - NN$) is obtained analytically by adding the 0-order Volterra weight $v^{(0)}$ to the product between each input variable (x_1 and x_2) and their corresponding first-order Volterra weights,

$$s^{(1)} = v^{(0)} + v^{(1)}_1 x_1 + v^{(1)}_2 x_2. \tag{5}$$

The second-order V-NN model ($V^{(2)} - NN$) output is obtained by adding $s^{(1)}$ together with the products among x_1^2, x_2^2, the cross-product $x_1\ x_2$ and their corresponding second-order Volterra weights, resulting in:

$$s^{(2)} = s^{(1)} + v^{(2)}_{1,1} x_1^2 + v^{(2)}_{2,2} x_2^2 + v^{(2)}_{1,2} x_1 x_2. \tag{6}$$

The $V^{(3)} - NN$ model output is obtained similarly:

Algorithm 2 3rd-Order Volterra-NN outputs forward computation

```
Data:
    $\mathbf{x}$: input point to classify
    $K$: number of elements in the array
    $\mathbf{v}^{(0)}$: 0-order Volterra weights for the array
    $\mathbf{v}^{(1)}$: array $1^{st}$-order Volterra weights
    $\mathbf{v}^{(2)}$: array $2^{nd}$-order Volterra weights
    $\mathbf{v}^{(3)}$: array $3^{rd}$-order Volterra weights
Results:
    $\mathbf{s}^{(1)}$: $V^{(1)} - NN$ outputs for the array
    $\mathbf{s}^{(2)}$: $V^{(2)} - NN$ outputs for the array
    $\mathbf{s}^{(3)}$: $V^{(3)} - NN$ outputs for the array
1  begin
2      for each element in the array do
3          $s^{(1)} \leftarrow v^{(0)}$
4          for $1 \le i \le N_I$ do
5              $s^{(1)} = s^{(1)} + v^{(1)}_i x_i$
6          $s^{(2)} \leftarrow s^{(1)}$
7          for $1 \le i \le N_I$ do
8              for $1 \le j \le N_I$ do
9                  $s^{(2)} = s^{(2)} + v^{(2)}_{i,j} x_i x_j$
10         $s^{(3)} \leftarrow s^{(2)}$
11         for $1 \le i \le N_I$ do
12             for $1 \le j \le N_I$ do
13                 for $1 \le k \le N_I$ do
14                     $s^{(3)} = s^{(3)} + v^{(3)}_{i,j,k} x_i x_j x_k$
15         Assign each $s^{(1)}$ calculated to $\mathbf{s}^{(1)}$
16         Assign each $s^{(2)}$ calculated to $\mathbf{s}^{(2)}$
17         Assign each $s^{(3)}$ calculated to $\mathbf{s}^{(3)}$
18     Determine upper and lower thresholds for each array element (class)
19 end
```

$$s^{(3)} = s^{(2)} + v^{(3)}_{1,1,1} x_1^3 + v^{(3)}_{2,2,2} x_2^3 + v^{(3)}_{1,1,2} x_1^2 x_2 + v^{(3)}_{1,2,2} x_1 x_2^2. \tag{7}$$

Similarly, higher order V-NN outputs can be calculated. As can be seen, each high order Volterra model includes its own parameters (Volterra weights) plus the lower order ones. The new algorithm for an array of V-NN models forward computation is presented in detail in Algorithm 2. It receives a point from where the number of input variables N_I is determined, the number of elements in the array, and the Volterra weights for each element in the array, obtained by using Algorithm 1. The output of this algorithm is an array of third-order ($V^{(3)} - NN$) Volterra outputs, but it is also possible to obtain only the first-order ($V^{(1)} - NN$) (lines 3 to 5) and second-order ($V^{(2)} - NN$) outputs (lines 7 to 10), thus providing different compressed versions of the original aMLP classifier. The next subsection shows how the compressed model can be used as a classifier.

2.3 Volterra-NN as a classifier

The Volterra-NN model can be applied to a classification problem in which different classes can be recognized from the output signal. When an array of MLP models is compressed by using V-NN, it is possible to identify only a class from each of the output signals. That is to say, each V-NN model is specialized on a class and the output signal analysis determines whether the pattern is part of the class.

During the training phase, data are shown in an ordered way, showing first those pattern belonging to the class associated with the model. In this way, the model output can be considered as a signal having two levels, with a bound between levels corresponding to the class limit [13]. For instance, if we have a three classes problem, the first model is trained with data where the patterns corresponding to the first class are activated (they have a 1 as target) and other patterns, which do not represent the first class, are not activated (they have a 0 as target). Similarly, training data for learning the second and third classes are presented to their corresponding models.

Figure 3 shows the output signal analysis that has to be performed for determining upper and lower thresholds for each class. It is possible to see the output signal corresponding to the MLP (full line) and V-NN (dotted line) models, both for the training data. The lower and the upper threshold (dashed lines) of the class are measured in order to determine the output-signal level change for the V-NN model, considering the values obtained for the training set. These changes (thresholds) will allow us to identify a new data point received for classification as belonging (or not) to the class. The membership of a new pattern (test point) to the class is identified by analyzing whether the output associated with this pattern has a value between the lower and upper class thresholds.

Since we are analyzing the output of an *a*MLP model, *a* different signals, each one corresponding to a class, must be considered in order to identify which class is activated as a result. First of all, each individual signal must be evaluated as it was presented in the above paragraph. Secondly, it is necessary to discover whether more than one class model was activated for each pattern. If this situation occurs, a *max* criteria is applied. That is to say, the higher value of the activated classes for each pattern will be stated as the winner class.

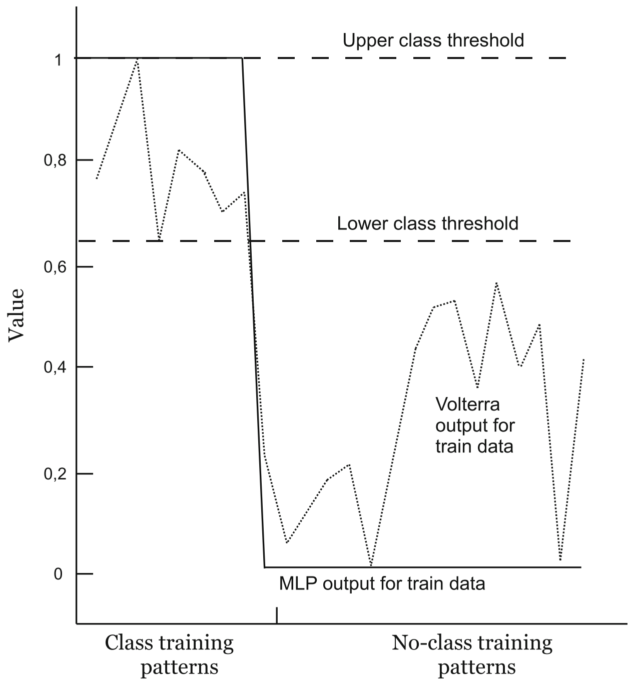

Fig. 3 Model output signal analysis for classification. *Full* and *dotted lines*: output signal corresponding to the MLP and V-NN models, respectively, for training data. *Dashed lines*: class thresholds

3 Materials and methods

3.1 Face databases

Two independent face databases have been used in this study. They provide typical experimental setups for face recognition. They are explained in detail in the following paragraphs.

3.1.1 AT&T Laboratories: ORL

The first experiments were done by using the AT&T Laboratories Cambridge ORL Database of Faces[1] since it is widely used in the face recognition literature [14]. In this database, there are 10 different images of each of 40 persons of different gender, ethnic background, and age (see Fig. 4). For some subjects, the images had been taken at different times, varying the lighting, facial expressions (open/closed eyes, smiling/not smiling), and facial details (glasses/no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). The complete database contains 400 grayscale face images of size 92 $\times$ 112 pixels. From this dataset, three subject were used in this study having ten pictures associated with each one.

Training a distinct classifier for each class (in this case, subject) requires sufficient training data per class. However, in face recognition tasks, it is common to have a small number of pictures per person. In fact, since different data partitions are used to train and validate the classifier, two pictures for subject are available for testing. Hence, noise addition to the test dataset has been performed in order to enlarge it and to obtain more test samples to prove the performance of the Volterra-NN model. The noise used was *Gaussian noise* with a mean of 0 and a variance between 0.01 and 0.1. After this procedure, a total of 66 patterns per class have been obtained for the testing dataset.

[1] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

Fig. 4 Samples face images from the AT&T Laboratories Cambridge ORL Database of faces [14]

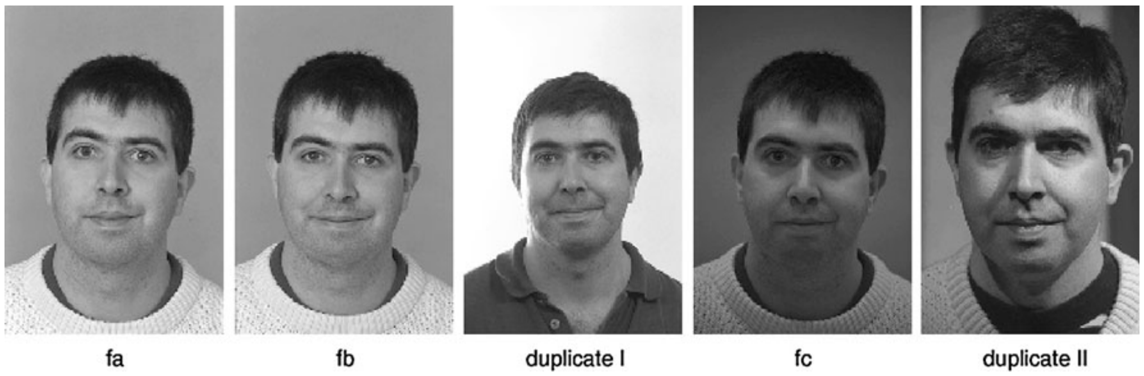

Fig. 5 The facial recognition technology (FERET) database [19]. Examples of different categories of probes (images). The *duplicate I* image was taken within 1 year of the *fa* image, and the *duplicate II* and *fa* images were taken at least 1 year apart

3.1.2 FERET

The facial recognition technology (FERET) database [19] ran from 1993 through 1997[2], sponsored by the Department of Defense's Counterdrug Technology Development Program through the Defense Advanced Research Products Agency (DARPA). The aim was to develop automatic face recognition capabilities that could be employed to assist security, intelligence, and law enforcement personnel in the performance of their duties. The FERET image corpus was assembled to support government monitored testing and evaluation of face recognition algorithms using standardized tests and procedures. The final corpus consists of 14,051 eight-bit grayscale images of human heads with views ranging from frontal to left and right profiles.

The facial images were collected in 15 sessions between August 1993 and July 1996. Collection sessions lasted one or two days. To maintain a degree of consistency throughout the database, the same physical setup and location was used in each photography session. Images of an individual were acquired in sets of 5 to 11 images. Two frontal views were taken (fa and fb); a different facial expression was requested for the second frontal image. For 200 sets of images, a third frontal image was taken with a different camera and different lighting (fc image). The remaining images were collected at various angles between right and left profile (see Fig. 5). To add variations to the database, a second set of images was taken, for which the subjects were asked to put on their glasses and/or pull their hair back. The set of images referred to as a duplicate indicates a second set of images of a person that was taken on a later date, resulting in variations in scale, pose, expression, and illumination of the face. Similarly to the previous dataset, three subject were used in the study having ten pictures associated with each one, adding gaussian noise to the pictures to increase the number of training/testing patterns.

3.2 Classifier architecture

Figure 6 shows the NN topology used in this study, an array of MLP models, each one associated with a subject to recognize. Several MLP topologies have been evaluated from where the corresponding Volterra weights have been extracted after training. For simplicity in the analysis of the results, in particular which respects the performance of the V-NN compression capabilities, only three classes ($k = 3$) of each database are presented in the tables.

[2] http://www.itl.nist.gov/iad/humanid/feret.

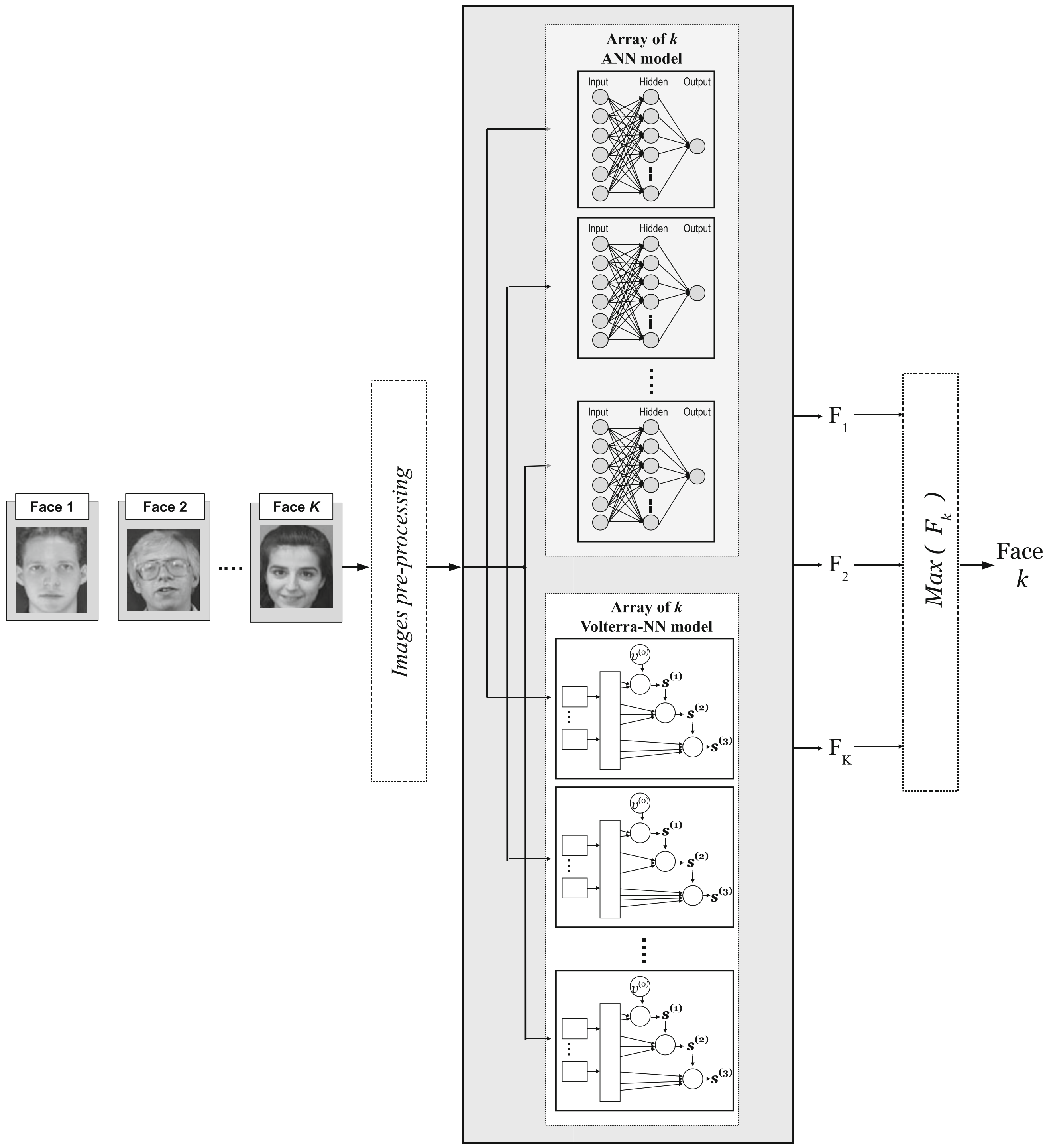

Fig. 6 *a*MLP model (*upper part* of the classifier) for face recognition problem, which can be compressed into an array of Volterra-NN models (*lower part* of the classifier)

The full results obtained on both face databases can be found as supplementary material.

In face recognition problems, the training and test datasets generally have high dimensionality due to the pictures size. Therefore, an appropriate feature extraction method is needed. A global representation can be done using a widely used technique such us the eigenfaces [24] by applying principal component analysis (PCA) for dimensionality reduction [11]. Although PCA can highly reduces the feature space, the application of a technique called *Scree Test* [10] further reduces the dimensionality of the features space focusing only on those most representative eigenfaces. Scree Test is applied for determining the number of principal components of the training and test datasets, according to the percentage of variability of the data to be shown. This technique allows obtaining the different components in an orderly way according to the variability of the data, so the first principal components represent the data of more variability.

Regardless of the data that are modeled with PCA, it is common to use a value of 85 % of the total variance of the space of characteristics for identification [10]. Applying Scree Test in the ORL dataset, the 85 % of the variance of each training set is represented by 11 eigenfaces, so each MLP model consists of 11 input neurons. As regards hidden neurons number, a simple heuristic will be adopted in which the number of neurons N_H are 11, 22, and 33, and an output that takes a value of 1 if the subject is recognized. Considering the case of the FERET dataset, the 85 % of the variance of each training set is represented by 9 eigenfaces. Therefore, each MLP model consists of 9 input neurons, a number of hidden neuron that can be 9,18, or 27, and also only one output. For each input signal arriving at the array, a V-NN output of a certain order can be obtained. For example, to obtain a first-order $V^{(1)} - NN$ output, the maximum $s^{(1)}$ from the array is selected.

The model parameters (weights and biases) are initialized with random values uniformly distributed between 0 and 1. Neurons in the hidden and output layers have sigmoid activation functions. All the MLPs are trained with the Levenberg-Marquardt algorithm [15] in order to guarantee a fast convergence. To avoid overfitting, a k-fold cross-validation procedure [9] has been used, using the standard setup of splitting the available images of each person into 80 % of each class data for training and 20 % for testing. The complete dataset has been randomly split into $k = 3$ mutually exclusive subsets of equal size, repeating the experiments three times in each fold. The cross-validation estimate of the overall accuracy of a model has been calculated by simply averaging the accuracy measures over the test datasets. In each experiment and repetition, MLP and V-NN models having less than 100 % classification rate for each class of the training dataset have not been considered in this study. This restriction was imposed to the classifier performance in order to be able to measure precisely any loss of accuracy originated by the proposed Volterra-NN compression method.

3.3 Performance measures

This subsection presents two classical measures for comparing models performance. Besides, a new measure for trade-off analysis and final model selection is proposed.

3.3.1 Recognition rate and space saving rate

For comparing each output from the proposed V-NN models against the corresponding MLP classifier, two performance measures are used: recognition rate (RR) and data space savings (SS), adapted here for an aMLP classifier. In classification problems, the primary source of performance measurements is the overall accuracy of a classifier estimated through the classification or recognition rate [6]. For measuring compression, data compression ratio can be used to quantify the reduction in data representation size produced by a compression algorithm. However, the space saving measure is given here instead, defined as the reduction in size relative to the uncompressed space [22], often reported as a percentage, which gives a better idea of compression power. For both cases, the greater the rate, the better the result.

To estimate how much is the compression level obtained, SS is calculated as the relation between the number of parameters needed for a Volterra-NN output (the Volterra weights) and the number of parameters of each corresponding MLP architecture (the weights and biases). It is well-known that for a MLP model, the number of model parameters can be calculated as follows:

$$P_{\mathrm{MLP}} = N_I \times N_H + N_{BH} + N_H \times N_O + N_{BO}, \tag{8}$$

where N_I, N_H, and N_O are the number of neurons at input, hidden, and output layers, respectively, and N_{BH} and N_{BO} are the number of bias for each neuron in the hidden and output layers, respectively. In an array of MLP models, these measures are affected by the array length a. Hence, the number of model parameters for an aMLP can be calculated as

$$P_{a\mathrm{MLP}} = a \times P_{\mathrm{MLP}}. \tag{9}$$

From each MLP that is part of the array of MLPs, a Volterra-NN output (of a particular order) can be extracted: $s^{(1)}$ that includes only the zero and first-order Volterra weights, $s^{(2)}$ includes up to the second-order Volterra weights, and $s^{(3)}$ that corresponds to a third-order model output. The following equations show the number of parameters necessary to build each of these outputs, for a given training set.

For the $V^{(1)} - NN$ model output, we have

$$P_{s^{(1)}} = N_I. \tag{10}$$

In this case, the number of parameters necessary to build $s^{(1)}$ is each first-order Volterra weight that multiplies each input variable.

For the second-order model $V^{(2)} - NN$, we have the following output

$$P_{s^{(2)}} = P_{s^{(1)}} + N_I^2 - \frac{N_I^2 - N_I}{2}. \tag{11}$$

The number of parameters necessary to build $s^{(1)}$ is summed up together with the number of parameters necessary for $s^{(2)}$ (the second-order Volterra weights). There is a second-order weight for each input variable squared, plus the cross-products between each input variable. With respect to these last ones, since the symmetrical weights are equivalent, they are considered only once. That is why half of the cross-product weights are counted.

The number of parameters necessary to build $V^{(3)} - NN$ is

$$P_{s^{(3)}} = P_{s^{(2)}} + N_I^2, \tag{12}$$

that is to say, the number of weights associated with $s^{(2)}$ plus the product of each third-order weight corresponding to each input variable, counting only once the symmetrical weights.

As Volterra-NN can be applied to the compression of an *a*MLP model, the output now will be an array of V-NN outputs of different order. That is to say, for example, for each MLP inside the array, three different-order V-NN outputs can be obtained. In this case, the number of V-NN models parameters has to be redefined as

$$P_{as^{(i)}} = a \times P_{s^{(i)}}, \tag{13}$$

where $i = 1, 2, 3$. Therefore, the space saving measure SS is calculated as

$$\mathrm{SS} = 1 - \frac{P_{as^{(i)}}}{P_{\ aMLP}}. \tag{14}$$

3.3.2 *Model selection: measure for trade-off solution indication*

Different solutions can be evaluated by calculating the performance measures presented above. A V-NN model output can have a value associated with the recognition rate or to the space saving measures. However, sometimes these values can differ from each other having even completely opposite meaning and the problem of how determining the best set of parameters arises. For instance, one solution can have a high performance as regards RR but a very low SS rate, or vice versa.

It is possible to consider the error for both measures as a point in a coordinate axis graph, where the X axis represents $e_s = 1 - \mathrm{SS}$, and the Y axes is associated with $e_r = 1 - \mathrm{RR}$. Therefore, a point can be defined as a pair $[e_s, e_r]$, being [0,0] the optimum. Figure 7 shows examples of models and their associated errors in this new space. Let us suppose a model 1 that has SS = 0.9 and RR = 0.7 and an error to the optimum represented in the point $M_1 =$ [0.1,0.3]. For model 2, where SS = 0.7 and RR = 0.9, the error is $M_2 =$ [0.3,0.1]. Model 3, in which SS = 0.3 and RR = 0.95, has an error $M_3 =$ [0.7,0.05], and a model 4, with SS = 0.95 and RR = 0.3, has an error $M_4 =$ [0.05,0.7].

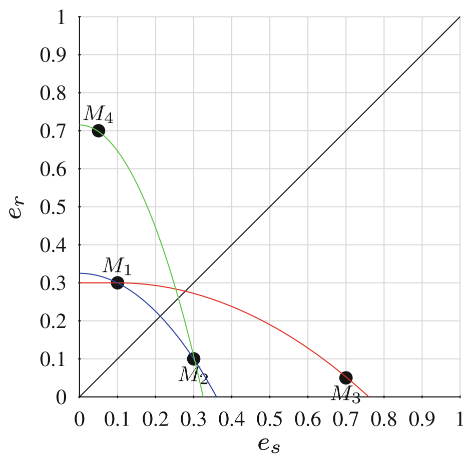

Fig. 7 Trade-off measure determination by analyzing the possible solutions of compressing a model

Considering these errors, a trade-off solution could be found as the minimum Euclidean distance from the point $[e_s, e_r]$ to the optimum [0,0]. However, when calculating these distances, it is possible to have cases with the same result, but opposite values. For example, points M_1 and M_2 are equidistant from the main diagonal and have the same Euclidean distance to the optimum; similarly to any other models that could be located on the blue parabola depicted in Fig. 7. However, M_2 is a better classifier than M_1, while M_1 is better compressed than the first one. Hence, it is necessary to discriminate which of the two points is the best solution for the problem under study.

Since FR is a classification problem, where it is reasonably to think that a better RR would be considered more important than SS, it is possible to infer that the trade-off solution must always be above the diagonal of the coordinate axes graph, because in a classification problem, it could be more important to have a high recognition rate than a high space savings rate. But, in some applications, it can be needed a better compressed model than an excellent classifier; in that case, SS would be more important than RR.

In order to evaluate which is the trade-off solution (more adequate model) for a problem, in the family of solutions that arose from the experiences, we define a new measure ∂ that represents a trade-off between RR and SS as follows:

$$\partial = \sqrt{(\gamma e_r)^2 + ((1-\gamma)e_s)^2}, \tag{15}$$

where γ is a regularization parameter. Precisely, to be able to perform the discrimination needed above, we propose to modify the distance calculation including a regularization parameter γ. In order to consider γ as a weight that allows us to select between two models, its value can be modified according to whether a better compressor or a better classifier is needed, with $\gamma \epsilon [0, \ldots, 1]$.

When $\gamma = 0.5$, both e_r and e_s are considered with the same importance; the preference for RR is emphasized with a $\gamma > 0{,}5$ (for example, red line in the figure), while the preference for SS is obtained choosing a $\gamma < 0.5$ (for example, green line in the figure). Applying (15) to each model M_k, presented in Fig. 7 and considering RR as a preference over SS with, for example $\gamma = 0.8$, we obtain $\eth_{M_1} = 0.241, \eth_{M_2} = 0.1, \eth_{M_3} = 0.146$, and $\eth_{M_4} = 0.56$. Now it is possible to conclude, without any doubt, that the best compromise solution is M_2 for the presented example, since it has the minimum distance to the optimum, according to the new distance calculation proposed.

4 Results and discussion

The experimental results obtained on two well-known face recognition problems are shown in this section. First of all, the results on class by class recognition rates are presented. Then, global RR and SS values for all the models obtained are shown. At last, global results using the new measure for model selection are presented.

4.1 Recognition performance for each class

From each $a\mathrm{MLP}_{I,H,O}$ model considered in this study, their corresponding zero-order, first-order, second-order, and third-order Volterra weights have been extracted according to Algorithm 1 and their corresponding array of Volterra-NN outputs $s^{(1)}$, $s^{(2)}$ and $s^{(3)}$ have been obtained using Algorithm 2. Table 1 presents the results obtained for the models recognition rate (RR_i) for each class $i = 1, 2, 3$ over the face recognition task, calculated over the ORL Database test set, whereas Table 2 shows similar results but calculated over the FERET database.

In Table 1, it is possible to see that, when the aMLP classifier has 11 hidden neurons ($3\mathrm{MLP}_{11,11,1}$), the RR values regarding class 1 (RR_1) are between 91 and 94 % for all $3\text{V-NN}_{s^{(1)}}$, $3\text{V-NN}_{s^{(2)}}$ and $3\text{V-NN}_{s^{(3)}}$; for the second class (RR_2), these values vary from 90 to 94 %, and they are between 91 % and approximately 97 % for the third class (RR_3). As regards the second FR problem (Table 2) when the aMLP classifier has 9 hidden neurons ($3\mathrm{MLP}_{9,9,1}$), the RR values are similarly high for class 1, 2, and 3.

Focusing on Table 1, it can be seen that a worse RR rate is obtained for the classification problem if $3\text{V-NN}_{s^{(3)}}$ is used instead of the original aMLP, specially for class 2. This lower RR is, still, of 90 %. Furthermore, the $3\text{V-NN}_{s^{(1)}}$ output requires a significant lower number of parameters than the original $3\mathrm{MLP}_{11,11,1}$ classifier (a relation of 1:12). Similar conclusions can be drawn from Table 2, in which the FERET Database is used: the simplest V-NN output is, at the same time, the best one. But if we take into account that the three $3\mathrm{MLP}_{I,H,1}$-model topologies in both cases, with $N_I = 11$, $N_H = 11, 22, 33$ for ORL and $N_I = 9$, $N_H = 9, 18, 27$ for FERET, are solutions to the same problem, actually it is not necessary

Table 1 Recognition rates (RR_i) for each class $i = 1, 2, 3$ for three aV-NN ($a = 3$) classifiers and their corresponding first-order $s^{(1)}$, second-order $s^{(2)}$, and third-order $s^{(3)}$ outputs, for the AT&T Laboratories Cambridge ORL Database of Faces

RR_i (%)	$3\mathrm{MLP}_{11,11,1}$			$3\mathrm{MLP}_{11,22,1}$			$3\mathrm{MLP}_{11,33,1}$		
	RR_1	RR_2	RR_3	RR_1	RR_2	RR_3	RR_1	RR_2	RR_3
$a\mathrm{MLP}_{I,H,O}$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
$3\text{V-NN}_{s^{(1)}}$	94.28	93.94	**97.47**	92.76	92.25	91.92	97.14	91.92	94.28
$3\text{V-NN}_{s^{(2)}}$	91.92	92.09	91.25	92.76	91.24	**94.27**	95.29	94.44	90.57
$3\text{V-NN}_{s^{(3)}}$	91.58	90.23	91.58	90.23	89.06	88.89	86.19	90.74	**93.26**

Bold numbers highlight the best value of RR for each order V-NN output

Table 2 Recognition rates (RR_i) for each class $i = 1, 2, 3$ for three aV-NN ($a = 3$) classifiers and their corresponding first-order $s^{(1)}$, second-order $s^{(2)}$, and third-order $s^{(3)}$ outputs, for the facial recognition technology (FERET) database

RR_i (%)	$3\mathrm{MLP}_{9,9,1}$			$3\mathrm{MLP}_{9,18,1}$			$3\mathrm{MLP}_{9,27,1}$		
	RR_1	RR_2	RR_3	RR_1	RR_2	RR_3	RR_1	RR_2	RR_3
$a\mathrm{MLP}_{I,H,O}$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
$3\text{V-NN}_{s^{(1)}}$	94.11	94.27	95.62	86.70	95.62	93.94	88.38	91.41	**96.63**
$3\text{V-NN}_{s^{(2)}}$	91.08	88.38	91.58	88.89	87.88	94.11	92.25	90.07	**95.29**
$3\text{V-NN}_{s^{(3)}}$	90.40	85.35	**93.10**	89.56	85.52	85.86	91.92	84.34	85.01

Bold numbers highlight the best value of RR for each order V-NN output to consider just only one topology as the best solution. That is to say, there are many possible solutions to the same problem and it is possible to obtain the best according to the goals that are followed. For instance, the best recognition rate in ORL Database case is obtained by the first-order Volterra-NN output for $3\text{MLP}_{11,33,1}$, which is approximately as high as 97.50 %, while the best recognition rate in FERET Database case is obtained by the first-order Volterra-NN output for $3\text{MLP}_{9,27,1}$, achieving an RR of 96.63 %.

Tables 3 and 4 show the space savings rate for the models discussed in this paper and the mean values of the recognition rate for these models. The upper part of each table shows the number of parameters and global RR measure for the three aMLP topologies. The second part of the table shows parameters and performance measure values related to the Volterra-NN outputs. First of all, the number of parameters needed for each neural classifier architecture ($P_{a\text{MLP}}$) involved in this study is shown at the first row, while the number of Volterra weights needed for each Volterra-NN model ($P_{as^{(i)}}$) is shown in the first column. The values that fill the RR and SS columns are the average recognition rate and space saving capabilities, respectively, for each combination between a MLP classifier and the corresponding Volterra-NN output.

As stated before (Eqs. 8, 9), the number of parameters required for the neural classifier depends not only on N_I but also on the number of neurons at the hidden layer N_H, as well as in the number of elements in the array. Therefore, considering the ORL Database, the number of parameters for $3\text{MLP}_{11,11,1}$ is 432, for $3\text{MLP}_{11,22,1}$ it is 861, and the $3\text{MLP}_{11,33,1}$ model has a total of 1,290 parameters (see Table 3). Regarding the FERET Database, the number of parameters for $3\text{MLP}_{9,9,1}$ is 300 and for $3\text{MLP}_{9,18,1}$ is 597, and the $3\text{MLP}_{9,27,1}$ model has a total of 894 parameters (see Table 4).

4.2 Recognition rate and space saving rate

In the case of the different-order Volterra-NN outputs, the number of parameters only depends on the number of inputs, as it is possible to see in (10), (11), and (12). However, for an array, this number must be multiplied by the array size. Therefore, the number of parameters for $3\text{V-NN}_{s^{(1)}}$ is 33, and the number of parameters needed for $3\text{V-NN}_{s^{(2)}}$ and $3\text{V-NN}_{s^{(3)}}$ are 231 and 594, respectively. In the FERET Database, the number of parameters for $3\text{V-NN}_{s^{(1)}}$ is 27, and the number of parameters needed for $3\text{V-NN}_{s^{(2)}}$ and $3\text{V-NN}_{s^{(3)}}$ are 162 and 405, respectively.

Focusing on Table 3, it is possible to see that when using $3\text{V-NN}_{s^{(1)}}$ instead of the $3\text{MLP}_{11,11,1}$ classifier, a global recognition rate for the face recognition problem of 95.23 % can be obtained and a compression or space saving rate of 92.36 % can be achieved. Almost half of this space saving rate is obtained if the 3V- output is used, and there is no compression when using $3\text{V-NN}_{s^{(3)}}$.

Table 3 Global recognition rate (RR) and space saving(SS) comparison for three aV-NN ($a = 3$) classifiers and their corresponding first-order $s^{(1)}$, second-order $s^{(2)}$, and third-order $s^{(3)}$ outputs, for the AT&T Laboratories Cambridge ORL Database of Faces

		$3\text{MLP}_{11,11,1}$		$3\text{MLP}_{11,22,1}$		$3\text{MLP}_{11,33,1}$	
$P_{a\text{MLP}_{I,H,O}} \rightarrow$		432		861		1,290	
RR(%) $\rightarrow$		100.00		100.00		100.00	
3V-NN	$P_{as^{(i)}}$	RR (%)	SS (%)	RR (%)	SS (%)	RR (%)	SS (%)
$3\text{V-NN}_{s^{(1)}}$	33	95.23	92.36	92.31	96.16	94.44	97.44
$3\text{V-NN}_{s^{(2)}}$	231	91.75	46.53	92.76	73.17	93.43	82.09
$3\text{V-NN}_{s^{(3)}}$	594	91.13	−37.50	89.39	31.01	90.07	53.95

Table 4 Global recognition rate (RR) and space saving (SS) comparison for three aV-NN ($a = 3$) classifiers and their corresponding first-order $s^{(1)}$, second-order $s^{(2)}$, and third-order $s^{(3)}$ outputs, for the facial recognition technology (FERET) database

		$3\text{MLP}_{9,9,1}$		$3\text{MLP}_{9,18,1}$		$3\text{MLP}_{9,27,1}$	
$P_{a\text{MLP}_{I,H,O}} \rightarrow$		300		597		894	
RR(%) $\rightarrow$		100.00		100.00		100.00	
3V-NN	$P_{as^{(i)}}$	RR (%)	SS (%)	RR (%)	SS (%)	RR (%)	SS (%)
$3\text{V-NN}_{s^{(1)}}$	27	94.67	91.00	92.09	95.48	92.14	96.98
$3\text{V-NN}_{s^{(2)}}$	162	90.35	46.00	90.29	72.86	92.54	81.88
$3\text{V-NN}_{s^{(3)}}$	405	89.62	−35.00	86.98	32.16	87.09	54.70

For the $3\mathrm{MLP}_{11,22,1}$ model, the compression achieved by the Volterra-NN models is higher because of the amount of parameters associated with this model; in this case, the SS value achieves a maximum of 96.16 % when the smallest possible number of parameters is considered (a first-order Volterra-NN output). A similar case is related to $3\mathrm{MLP}_{11,33,1}$ model, where this trend is also verified. The results obtained with the other dataset are quite similar (Table 4). For instance, a global recognition rate of 94.67 % can be obtained and a compression or space saving rate of 91 % can be achieved when the $3\text{V-NN}_{s(1)}$ is used instead of the $3\mathrm{MLP}_{9,9,1}$ classifier.

From both Tables 3 and 4, it is possible to conclude that an array of MLPs for classification can be well-compressed by using Volterra-NN model, and this compression rate can be, in some cases, even higher than 90 %. Moreover, very high recognition rates are achieved. In fact, the *a*MLP model architecture can be more and more complex and can have many more hidden units, but the number of weights needed to build each Volterra-NN output will remain the same while the number of input variables of the problems is the same, and this will certainly be reflected in even higher space saving rates.

Furthermore, in some cases, these high compression rates have not to be payed by lower model classification rates. For example, for the $3\mathrm{MLP}_{11,11,1}$ model in the ORL Database, the best RR value is related to the $3\text{V-NN}_{s(1)}$ output, achieving 95.23 %, and having a space saving rate of more than 92 %. The exactly same case can be seen in the FERET Database, in which the $3\mathrm{MLP}_{9,9,1}$ model achieves an RR of 94.67 %, having a space saving rate of 91 %.

Considering the $3\mathrm{MLP}_{11,22,1}$ case, in Table 3, the best RR value is associated with the $3\text{V-NN}_{s(2)}$ output, which has only a SS of 73.17 %. But if a better compression is necessary for this topology, it is possible to choose the $3\text{V-NN}_{s(1)}$ which has a SS of 96.16 % and preserves a very high RR value of 92.31 %. Even though the differences are minimum between $3\text{V-NN}_{s(1)}$ and $3\text{V-NN}_{s(2)}$ outputs regarding the second model in both tables, in Table 4, the results are slightly different if it is considered the model $3\mathrm{MLP}_{9,18,1}$. Here, the best RR and SS are obtained in the same case, achieving a space saving of 95.48 % preserving the 92.09 % of recognition ability if the $3\text{V-NN}_{s(1)}$ output is chosen.

For the $3\mathrm{MLP}_{11,33,1}$ model, in Table 3, the best RR value is associated with the $3\text{V-NN}_{s(1)}$ Volterra-NN model and it is possible to achieve a SS of 97.44 %. This is a very interesting result, because with approximately less than 3 % of the parameters of the original *a*MLP classifier, the classification capacity of $3\text{V-NN}_{s(1)}$ is very high (94.44 %) and very close to the *a*MLP model. This particular case can be considered as the best overall result obtained in this study, where the $3\mathrm{MLP}_{11,33,1}$ classifier can be compressed in almost a 98 % using the corresponding $3\text{V-NN}_{s(1)}$ Volterra-NN output without significantly loosing recognition capability. Similar results can be highlighted in Table 4. Once again, a small difference of 0.4 % can be seen as regards the output ($3\text{V-NN}_{s(1)}$ or $3\text{V-NN}_{s(2)}$) that is related to the best RR obtained in the $3\mathrm{MLP}_{9,27,1}$ classifier. Despite this, it is possible to state that the best overall result obtained in this case is related to the $3\text{V-NN}_{s(1)}$ Volterra-NN output, in which the classifier can be compressed in almost a 97 %, maintaining a 92 % of recognition rate.

We have compared our proposal against two classical weight pruning algorithms (as simpler ways of compression) for MLP models, such as optimal brain damage (OBD) [5] and optimal brain surgeon (OBS) [8]. Two aspects have been compared: i) RR of each method while maintaining the same number of parameters and ii) SS considering approximately the same RR value for all the three methods. In ORL database case, for the 33 parameters used for $3\text{V-NN}_{s(1)}$ in order to achieve an RR of 95.23 %, OBS and OBD obtained RR values of 40.57 and 61.11 %, respectively (see Table 5). As regards FERET database, considering that $3\text{V-NN}_{s(1)}$ needs 27 parameters for an RR value of 94.67 %, OBD obtained an RR value of 38.38 %, and OBS achieved an RR of 40.24 % for the same number of parameters (see Table 7). With respect to compression, as it is possible to see in Tables 6 and 8, for maintaining an RR of 95 % in both databases, while V-NN needs approximately 30 parameters, OBS and OBD require approximately 200 parameters or more.

As a limitation of the *a*V-NN model, it can be noted that the model reduces its compression capability as long as the number of units in the hidden layer of the original MLP decreases. That is to say, as smaller the number of hidden neurons needed to solve the problem, less compression will be obtained by the *a*V-NN model.

From the previous paragraphs, it can be seen that it is necessary to look at both RR and SS tables, for each dataset, to select the best model compressed, which can be confusing. The next subsection will present this analysis in

Table 5 RR with the same SS in *a*V-NN, OBD, and OBS (ORL Database)

		$3\mathrm{MLP}_{11,11,1}$	
$P_{a\mathrm{MLP}_{I,H,O}}$ $\rightarrow$		432	
RR(%) $\rightarrow$		100.00	
	$P_{as(i)}$	RR (%)	SS (%)
$3\text{V-NN}^{(1)}$	33	95.23	92.36
OBD		40.57	
OBS		61.11	

Table 6 SS in *a*V-NN, OBD, and OBS, with the same RR (ORL Database)

		$3\text{MLP}_{11,11,1}$	
$P_{a\text{MLP}_{I,H,O}} \rightarrow$		432	
RR(%) $\rightarrow$		100.00	
	$P_{as^{(i)}}$	RR (%)	SS (%)
3V-NN$^{(1)}$	33		92.36
OBD	192	$\approx$ 95	58.44
OBS	285		34.03

Table 7 RR with the same SS in *a*V-NN, OBD, and OBS (FERET Database)

		$3\text{MLP}_{9,9,1}$	
$P_{a\text{MLP}_{I,H,O}} \rightarrow$		300	
RR(%) $\rightarrow$		100.00	
	$P_{as^{(i)}}$	RR (%)	SS (%)
3V-NN$^{(1)}$		94.67	
OBD	27	38.38	91.00
OBS		40.24	

Table 8 SS in *a*V-NN, OBD, and OBS, with the same RR (FERET Database)

		$3\text{MLP}_{11,11,1}$	
$P_{a\text{MLP}_{I,H,O}} \rightarrow$		300	
RR(%) $\rightarrow$		100.00	
	$P_{as^{(i)}}$	RR (%)	SS (%)
3V-NN$^{(1)}$	27		91.00
OBD	204	$\approx$95	32.00
OBS	198		34.00

a simplified manner, through the use of the proposed $\eth$ trade-off measure.

4.3 Global results for model selection

Tables 9 and 10 present the $\eth$ values for each 3V-NN, considering three different γ values. The rows group the three possible 3V-NN outputs ($s^{(1)}$, $s^{(2)}$, and $s^{(3)}$) for each 3MLP topologies that are shown, which vary in the number of hidden neurons inside the single MLP model. The three main columns represent the application of three different γ values, each one to emphasize the priority of SS over RR ($\gamma = 0.25$), RR over SS ($\gamma = 0.75$), or both of them equally measured ($\gamma = 0.5$).

Taking into account the results at the first column ($\gamma = 0.25$) of Table 9 on the one hand, it is possible to conclude that the best trade-off solution is reached by $3\text{V-NN}_{s^{(1)}}$ when the topology of the MLP model is $3\text{MLP}_{11,33,1}$, achieving the minimum $\eth$ value 0.024 (it is the best SS (97.44 %) and its RR is up to 94.44 %). On the other hand, focusing on the same column of Table 10, similar conclusions can be drawn due to the best trade-off solution is also reached by $3\text{V-NN}_{s^{(1)}}$, when the MLP model is $3\text{MLP}_{9,27,1}$, achieving the minimum $\eth$ value 0.030 (SS is 96.98 % while its RR is as higher as 92.14 %).

For the second column ($\gamma = 0.5$), the trade-off solution is associated with the same solution that in the previous case: $3\text{V-NN}_{s^{(1)}}$ for the MLP model $3\text{MLP}_{11,33,1}$ in Table 9 and for the MLP model $3\text{MLP}_{9,27,1}$ in Table 10. For this case, both measures at both experiences have high values and determine the trade-off solution when there is no priority criteria between RR and SS rates.

In the third column, in which $\gamma = 0.75$, the $3\text{V-NN}_{s^{(1)}}$ model for the $3\text{MLP}_{11,11,1}$ model is the trade-off solution at Table 9, achieving the minimum $\eth$ value 0.041. Its RR value is the best one (95.23 %), and its SS is up to 92.36 %. In Table 10, the $3\text{V-NN}_{s^{(1)}}$ model for the $3\text{MLP}_{9,9,1}$ model is also the trade-off solution for the third column, with an RR value of 94.67 % and a SS of 91 %.

Table 9 Measure for trade-off solution $\eth$ comparison for 3Volterra-NN models corresponding to a 3MLP classifier, for the AT&T Laboratories Cambridge ORL Database of Faces

$\eth$	3V-NN	$\gamma_1 = 0.25$	$\gamma_2 = 0.5$	$\gamma_3 = 0.75$
	$3\text{V-NN}_{s^{(1)}}$	0.059	0.045	**0.041**
$3\text{MLP}_{11,11,1}$	$3\text{V-NN}_{s^{(2)}}$	0.402	0.271	0.147
	$3\text{V-NN}_{s^{(3)}}$	1.031	0.689	0.350
	$3\text{V-NN}_{s^{(1)}}$	0.035	0.043	0.058
$3\text{MLP}_{11,22,1}$	$3\text{V-NN}_{s^{(2)}}$	0.202	0.139	0.086
	$3\text{V-NN}_{s^{(3)}}$	0.518	0.349	0.190
	$3\text{V-NN}_{s^{(1)}}$	**0.024**	**0.031**	0.042
$3\text{MLP}_{11,33,1}$	$3\text{V-NN}_{s^{(2)}}$	0.135	0.095	0.067
	$3\text{V-NN}_{s^{(3)}}$	0.346	0.236	0.137

The best value for each model is highlighted in bold

It is necessary to note that the last $\eth$ in Table 9 has a very close value ($\eth = 0.042$) for $3\text{V-NN}_{s^{(1)}}$ when the topology of the MLP model is $3\text{MLP}_{11,33,1}$. But, in this last case, SS is higher than RR and this situation is penalized by the value of γ, because of the priorities in model selection. It is important to note that for all previously options, $3\text{V-NN}_{s^{(1)}}$ is the best trade-off model, which requires fewer parameters to be represented and therefore is the smallest one.

Another important point to be highlighted is that the model selection step is a necessary task that has to be

Table 10 Measure for trade-off solution ∂ comparison for *3*Volterra-NN models corresponding to a *3*MLP classifier, for the facial recognition technology (FERET) database

∂	*3*V-NN	$\gamma_1 = 0.25$	$\gamma_2 = 0.5$	$\gamma_3 = 0.75$
	$3\text{V-NN}_{s(1)}$	0.069	0.052	**0.046**
$3\text{MLP}_{9,9,1}$	$3\text{V-NN}_{s(2)}$	0.406	0.274	0.153
	$3\text{V-NN}_{s(3)}$	1.013	0.677	0.346
	$3\text{V-NN}_{s(1)}$	0.039	0.046	0.060
$3\text{MLP}_{9,18,1}$	$3\text{V-NN}_{s(2)}$	0.205	0.144	0.100
	$3\text{V-NN}_{s(3)}$	0.510	0.345	0.196
	$3\text{V-NN}_{s(1)}$	**0.030**	**0.042**	0.059
$3\text{MLP}_{9,27,1}$	$3\text{V-NN}_{s(2)}$	0.137	0.098	0.072
	$3\text{V-NN}_{s(3)}$	0.341	0.236	0.149

The best value for each model is highlighted in bold

performed in order to be certain of which model and parameters are the most suited to a dataset. In this sense, the ∂ measure can help in the comparisons among models. Furthermore, this measure could help finding the best possible classifier for each class by combining different-order V-NN outputs, obtained from different neural topologies and configurations.

Finally, from the analysis of Tables 9 and 10, it can be seen that consistent results are obtained with respect to the detailed analysis performed on Tables 3 and 4, achieved, however, in a more compact and simpler way, thanks to the new proposed trade-off measure.

5 Conclusions and future work

This paper has shown a method to obtain a compact representation of an array of MLPs using the different-order *a*V-NN model outputs. Two algorithms that implement the proposed approach have been explained. Algorithm 1 allowed extracting the Volterra kernels from the MLP parameters (after training). Algorithm 2 allowed obtaining the third-order ($V^{(3)} - NN$) Volterra output by using the Volterra weights when a new data point to be classified is received. The *a*V-NN has been tested on a face recognition task, obtaining almost the same accuracy than three different configurations of arrays of MLP classifiers. They have been significantly compressed into less parameters. Experimental results have demonstrated the capabilities of the proposed V-NN model to compress a solution to the face recognition problem with very high recognition and space savings rates. Furthermore, a new trade-off measure for model selection was proposed, allowing to consider in a simple manner a priority of one rate over the other one. This new measure ∂ allowed us to evaluate different solutions in a compact way by only considering one value, which arose from the relationship between the recognition rate and the space savings. This measure was useful for indicating one of the obtained outputs as the best one, considering both rates at the same time but with a different weights.

Future work involves further application of the proposed method to more complex problems, involving more classes for classification and data having an evolution on time. Besides, the V-NN compression capabilities could be tested on ensembles of different kinds of NN models. These different NN classifier topologies should be further studied in order to establish their impact on the compression capabilities offered by the proposed Volterra-NN model approach.

References

1. Aitkenhead MJ, McDonald AJS (2003) A neural network face recognition system. Eng Appl Artif Intell 16(3):167–176
2. Bianchini M, Maggini M, Sarti L, Scarselli F (2005) Recursive neural networks learn to localize faces. Pattern Recognit Lett 26(12):1885–1895
3. Buciluǎ C, Caruana R, Niculescu-Mizil A (2006) Model compression. In: KDD'06: proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 535–541
4. Capello D, Martinez C, Milone D, Stegmayer G (2009) Array of multilayer perceptrons with no-class resampling training fr face recognition. Revista Iberoamericana de Inteligencia Artificial 13(44):5–13
5. Cun YL, Denker JS, Solla SA (1990) Optimal brain damage. In: Touretzky DS (ed) Advances in neural information processing systems. Morgan Kaufmann, Los Altos, pp 598–605
6. Duda R, Hart P (2003) Pattern classification and scene analysis. Wiley, London
7. Dzeroski S, Zenko B (2004) Is combining classifiers with stacking better than selecting the best one? Mach Learn 54:255–273
8. Hassibi B, Stork DG, Com SCR (1993) Second order derivatives for network pruning: optimal brain surgeon. In: Hanson SJ, Cowan JD, Giles CL (eds) Advances in neural information processing systems 5. Morgan Kaufmann, Los Altos, pp 164–171
9. Haykin S (1999) Neural networks: a comprehensive foundation. Prentice-Hall, Englewood Cliffs
10. Jackson J (1991) A user's guide to principal components. Wiley series in probability and mathematical statistics: applied probability and statistics. Wiley, London. http://books.google.com.ar/books?id=qhQYvH8CFQQC
11. Kirby M, Sirovich L (1990) Application of the Karhunen–Loeve procedure for the characterization of human faces. IEEE Trans Pattern Anal Mach Intell 12(1):103–108
12. Kong S, Heo J, Abidi B, Palk J, Abidi M (2005) Recent advances in visual and infrared face recognition-a review. Comput Vis Image Underst 97(1):103–135
13. Korenberg M, David R, Hunter I, Solomon J (2001) Parallel cascade identification and its application to protein family prediction. J Biotechnol 91:35–47
14. Li S, Jain A (eds) (2004) Handbook of face recognition. Springer, Berlin
15. Madsen K, Nielsen HB, Tingleff O (2004) Methods for nonlinear least squares problems

16. Marquardt D (1963) An algorithm for least-squares estimation of nonlinear parameters. SIAM J Appl Math 11(2):431–441
17. Martinez A, Kak A (2001) Pca versus lda. IEEE Trans Pattern Anal Mach Intell 23(2):228–233
18. Orengo G, Colantonio P, Serino A, Giannini F, Stegmayer G, Pirola M, Ghione G (2007) Neural networks and Volterra-series for time-domain pa behavioral models. Int J RF Microw CAD Eng 17(2):160–168
19. Phillips PJ, Moon H, Rizvi SA, Rauss PJ (2000) The feret evaluation methodology for face-recognition algorithms. IEEE Trans Pattern Anal Mach Intell 22:1090–1104
20. Rahman A, Verma B (2011) Novel layered clustering-based approach for generating ensemble of classifiers. IEEE Trans Neural Netw 22(5):781–792
21. Rubiolo M, Stegmayer G, Milone D (2010) Compressing a neural network classifier using a volterra-neural network model. In: IEEE international joint conference on neural networks (IJCNN), Barcelona, Spain, pp 1–7
22. Salomon D (2007) Data compression: the complete reference. Springer, Berlin
23. Stegmayer G, Chiotti O (2009) Volterra NN-based behavioral model for new wireless communications devices. Neural Comput Appl 18:283–291
24. Turk M, Pentland A (1991) Eigenfaces for recognition. J Cogn Neurosci 3(1):72–86
25. Volterra V (1959) Theory of functionals and integral and integro-differential equations. Dover, New York
26. Zhang D, Wangmeng Z (2007) Computational intelligence-based biometric technologies. IEEE Comput Intell Mag 2(2):26–36
27. Zhao W, Chellappa R, Phillips P, Rosenfeld A (2003) Face recognition: a literature survey. ACM Comput Surv 35(4):399–458