The 2nd International Conference on Neural Networks and Artificial Intelligence, October 2-5, 2001, Minsk, Belarus, pp. 136-142

APPLICATION OF RECIRCULATION NEURAL NETWORK AND PRINCIPAL COMPONENT ANALYSIS FOR FACE RECOGNITION

Dmitry Bryliuk and Valery Starovoitov

Institute of Engineering Cybernetics, Laboratory of Image Processing and Recognition
Surganov str., 6, 220012 Minsk, BELARUS
E-mail: bdv78@mail.ru, valerys@newman.bas-net.by

Keywords:
Principal component analysis, recirculation neural network, adaptive learning rate, face recognition

ABSTRACT

In this paper we describe our study of the recirculation neural network (RNN) for calculation principal components from a large set of face images. A new simple and fast algorithm was used to train RNN. Obtained principal components are used for face recognition. We also explore the ability of the RNN to reconstruct face images. Advantages of the proposed approach are shown.

1. INTRODUCTION

In many applications, particularly in pattern and image recognition, there is a need for dimensionality reduction of pattern description. One of the approaches of extraction of statistically independent features is the principal component analysis. In the classical approach it is required to solve large matrix equations to obtain eigenvectors. Instead, in neural networks, eigenvector-like weight matrices are formed on neurons during a training process.

There are several preceding works in this area. In [1] a recirculation neural network was used to extract principal components for a set of faces. At the input of the neural network was an image of a whole face. Then another neural network was used to classify output of hidden units of recirculation network according to their sex and emotional state and to classify to which person belongs image of face. It worth to remark that the used face database was simple (no 3D pose variation, no extremal changing in illumination conditions). We tested this approach on a more complex and largest ORL database (www.cam-orl.co.uk/facedatabase.html). Also, they used sigmoid activation functions. As it was remarked in [2] it is better to use an activation function with the output range [-1; 1]. Then, in [1] a constant learning rate was used. Due to this fact, when the value of learning rate was high, the output of hidden units was binary-like (i.e. about 0 or about 1). When the learning rate was small, the training process required more time.

A similar approach was used in [3] to compress a single image. A sliding window was moved over parts of image, and these parts were presented to the neural network. Advantage of the approach is a novel training algorithm. There is an adaptive learning rate and separate training of each layer. This algorithm requires only few training steps (from 5 to 20) to achieve the minimal root mean square error of image reconstruction. Instead in other approaches, such as standard backpropagation or cumulative delta-rule, it is required much more training steps.

In our work we has combined mentioned approaches. We have applied a recirculation neural network with adaptive learning rate to extract a compressed representation of a set of face images from the ORL database. Then Euclidean distance was used to calculate similarity between training and test faces.

2. THEORETICAL BACKGROUND

A recirculation neural network is based on multilayer perceptron, and it has one hidden layer (Figure 1). The number of units in the input and output layers is n and equals to the number of

pixels in the input image. The number of units in the hidden layer is m, where $m<<n$, and m is equal to the number of principal components which is predefined.

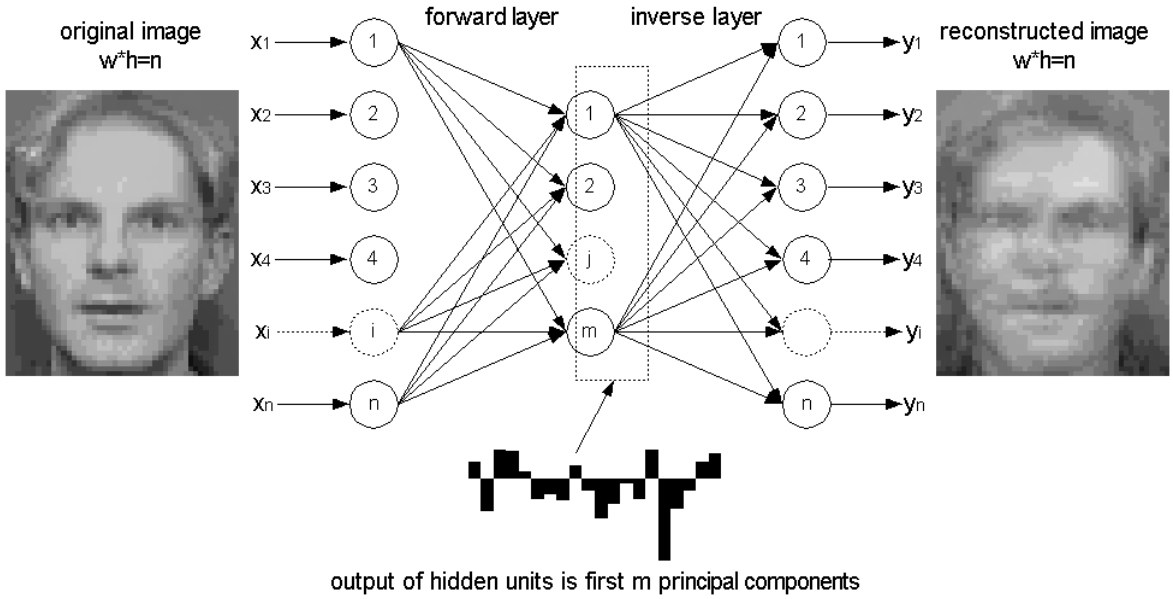

Figure 1. Architecture of recirculation neural network.

During the training process a neural network is learned to compress and reconstruct input images through the small number of hidden units, and the output of the hidden units is a compressed representation of the image. In the case of linear activation functions it is proved that when a network has the global minimum of the reconstruction error, the values formed on the output of hidden units are exactly the same as first principal components obtained by solving matrix equations [3]. For the non-linear activation functions the output from the hidden units resembles principal components and represents the input image better than linear network [3].

The RNN has an advantage in comparison with solving matrix equations. There is no need to calculate all eigenvectors when you need only a small number of the first eigenvectors. For the less number of principal components it is required less time to train network. Also due to the similarity of images there can be problems when solving matrix equations. RNN have no such drawback.

In our study we used a hyperbolical tangent as an activation function. It has output range [-1; 1] and a simple calculation of derivatives. The input of RNN must have a zero mean, and must be normalized or appropriately scaled to avoid paralysis of network. For each of our experiment a scale factor is given in tables.

The forward step, which calculates an output of the network is:

$$y_{ki} = x_i, k = 0,$$

$$y_{ki} = \tanh(\sum_{j=1}^{p} y_{k-1,j} w_{kij}), k = 1..L,$$

where k is an index of current layer, k increases from 0 to L, p is a number of neurons in a previous ($k-1$) layer, i is an index of neuron in current layer, j is an index of neuron in previous layer, x_i is an input sample element, y_{ki} is an output of layer k (or input for layer $k+1$), w_{kij} is a weight from neuron j in layer $k-1$ to neuron i in layer k, L is an index of last layer (i.e. 2 here).

When the network is trained, the output of the first layer gives compressed representation of the input image, it is the first m principal components. Due to the fact of a random initialization of the network the components have different random orders for each training process. To compress an image it is required to put the image in the input layer and calculate the output of the hidden units. To reconstruct an image by its principal components it is required to put the output of the hidden units and calculate the output of the last (output) layer.

To train a network we apply an error correction procedure, it is called backpropagation. Its first step is calculation of error, i.e. propagation of error from the output layer to input layer through the hidden layers:

$$\delta_{ki} = (y_{ki} - t_i) \cdot (1 - y_{ki}^2), k = L,$$

$$\delta_{ki} = (\sum_{j=1}^{q} \delta_{k+1,j} w_{k+1,ji}) \cdot (1 - y_{ki}^2), k = (L-1)..1,$$

where q is a number of neurons in layer $k+1$, δ_{ki} is an error correction coefficient, t_i is the desired output of the network, for RNN $t_i = y_{0,i}$. Note that index of layer k decreases from L to 0.

The second step is a weight correction:

$$w_{kij}(t+1) = w_{kij}(t) - \alpha(t)\delta_{ki} y_{k-1,j}, k = 1..L,$$

where $\alpha(t)$ is learning rate, t is a number of iteration. For a classical backpropagation it is fixed. There are also a number of heuristical approaches where the learning rate varies from big values on the beginning to the lower values on the next steps.

The main advantage of Golovko's approach is the **adaptive** learning rate, which is calculated for each layer at each iteration in order to made the best step minimizing the root mean square error of the network. Rewritten in δ-notation it is:

$$\alpha(t) = \frac{\sum_{i=1}^{r} \frac{\delta_{ki}^2}{1 - y_{ki}^2}}{\left(1 + \sum_{j=1}^{p} y_{k-1,j}^2\right) \cdot \left(\sum_{i=1}^{r} \delta_{ki}^2\right)},$$

where r is a number of neurons in layer k. Note that $\frac{\delta_{ki}^2}{1 - y_{ki}^2} = 0$, when $1 - y_{ki}^2 = 0$, and $\alpha(t) = 0$, when $\sum_{i=1}^{r} \delta_{ki}^2 = 0$. In the original γ-notation by Golovko there is no division by zero.

With adaptive learning rate there is no need to choose the rate manually when it can be too small or too big. The training process converged quite fast and stable. For simplicity we did not use sectioning training, see details in [3].

Before the training process a weights are initialized with random values in range [-0.01; 0.01].

Training process consists from the sequence of training steps. This process stops after given number of steps or when error of network is less then required.

At each training step all images from the training set is presented to the network in random order. For each image forward step calculated and then the procedure of error correction is applied.

All this formulas can be successfully applied to train any multilayer perceptron.

3. EXPERIMENTAL RESULTS

In our experiments we investigated application of the recirculation neural network for reconstruction and recognition of face images. For an unknown image and training images a compressed representation is calculated. Recognition was performed by calculation Euclidean distance between points, which coordinates were the outputs of hidden units. The set of distances from the unknown image to all images from training set is calculated. An image with minimal distance to the unknown image considered as recognized. Note, any image from the training set did not compare with itself, because in this case the distance equals zero. Also we have tried some other metrics, but their performance was comparable or worse.

We did not use centers of clusters for a particular person. The faces from the database have some degree of variation in pose, scale and expression. In [4] it has been shown that for the head rotation from left to right coordinates of principal components formed trajectory, called eigensignature. And such eigensignature is unique for each face. In the case of other variations (head tilt, scale, expression) the principal components of each face is formed a kind of a hypersurface, not a cluster. The principal components of training faces lies on such hypersurfaces. And it is more correct to calculate distance for the each training image instead of the distance to the center of cluster.

Images from ORL database (400 images, 40 person, 10 images per person with slightly different scale, pose and expression) were randomly divided in training and test set. 5 images of each person for training, 5 for testing. If otherwise is not mentioned, division was the same for a sequence of experiments. For our experiments we have varied the following factors.

1) Number of training cycles.
2) Number of units in hidden layer (e.g. number of principal components).
3) Resolution of images: ORL/1 (92x112, original size), ORL/2 (46x56, one half), ORL/4 (23x28, one quarter from the original size).
4) Different random divisions of the database into training and test sets.

The time of neural network learning is shown for all experiments. Experiments were performed on Pentium 166MHz computer.

Our experimental results are collected in tables. Training and test rmse means root mean square error of reconstruction images from training and test sets. Training step is a number of network training cycles. Training time is a network training time between the previous and the current rows in table. Training and test errors are a percent of misclassified images. Due to the fact that images from the training set were not compared with itself, the recognition rate for the training set is always lower.

3.1. Experiment 1

In this experiment we varied image scale and number of training steps. The number of hidden units was 20 and fixed. Results are given in Table 1.

As it can be seen from Table 1, the reconstruction error jumps down during the first 10-20 steps and practically did not change during the next steps. The recognition rate has the same trend. The reconstruction error and the recognition rate have a little hesitation because order of samples for learning is random at each step. The average recognition rate for test set is about 93%.

Despite the recognition rate at step 0 (after initialization of the network) is high, the distinction between classes is very fuzzy, and distances between classes are small.

For each scale of image size the training and test sets has different division, but it is hard to say that the recognition rate depends on the image size.

Figure 2 illustrates how the reconstruction of image changes on each step. Also the variation of the principal components is shown under each image.

Figure 3 shows input and output weights. They are similar to eigenvectors. In some places black color is changed to white and contrary, because weights are not appropriately scaled when drawn.

Figure 4 shows compressed representation and reconstruction of images from the training and test sets.

3.2. Experiment 2

In this experiment we varied the number of principal components. The results of this experiment are presented in Tables 1 and 2. Table 2 is a summary for different number of principal components and fixed number of training steps. As it can be seen from the tables, reconstruction error has a strong dependence of the number of principal components. Recognition rate also depends on a number of principal components, but the dependence is weaker. Training time is proportional to the number of hidden units.

The 2nd International Conference on Neural Networks and Artificial Intelligence, October 2-5, 2001, Minsk, Belarus, pp. 136-142

3.3. Experiment 3

In this experiment we tested a random division of the image database into the training and test sets. The image size and the number of principal components are fixed.

As it can be seen from Table 3 the recognition rate depends on the quality of the training set. It means if the unknown image of any class has no similar image (with same pose or scale) in the training set, the system may fail (see next experiment).

3.4. Experiment 4

In this experiment we manually analyzed how system made errors. The recognition is hard when in the training set there is no image of required person under particular conditions: pose, lighting, scale, emotional expression. In this cases system may respond to the similar conditions of images such as pose, scale etc. Also system can misclassify visually similar images, such as s28 and s37, s38/9 and s23/10.

Examples of the correct and wrong classification with calculated distances are given in Table 4.

3.5. Experiment 5

The recirculation neural network was trained once using only 5 images from 20 persons. Then database was divided into equal training and test set. For each division the recognition rate was taken. The recognition rate for such sequences of divisions was little worse then in previous experiments, and was about 90%.

4. CONCLUSION

We made a sequence of experiments. Recirculation neural network was applied for extracting principal components from the whole images from ORL database. Then such components were used for image reconstruction and recognition. Average recognition rate was about 92%.

There are several advantages of our approach. The recirculation neural network is a good tool for the principal components extraction. The advantage of RNN is that training time linearly depends on the number of required principal components. Application of adaptive learning rate allows achieving optimal reconstruction error in a few steps (about 10-20). The output of hidden units was continuous in range [-1; 1]. There are no difficulties when images from training set are very similar. The principal components extracted for RNN is well suited for the image reconstruction and recognition.

For preliminary experiments it is worth to use images with reduced size, less number of principal components (10-20) and less training steps (10-20). For a final application one can use bigger number of the components (30-40 and more), and more training steps.

For the future application of principal components in recognition we need a mechanism, which maximizes interclass distances and minimizes intraclass distances. Because the distance, based on the Euclidean metric gives mixed and not distinct measure between classes (see Table 4 for example).

For a practical application, such as an access control, we need more extensive training set, or use an approach proposed in [4,5].

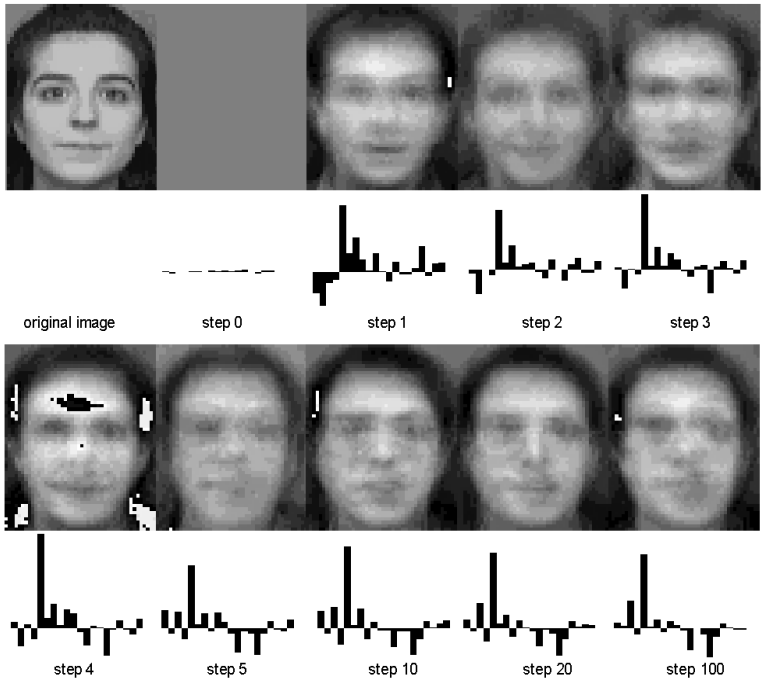

Figure 2. Reconstruction of the top-left image and outputs of the 20 hidden units at different training steps applied to the ORL/2 image set (see other details in Table 1).

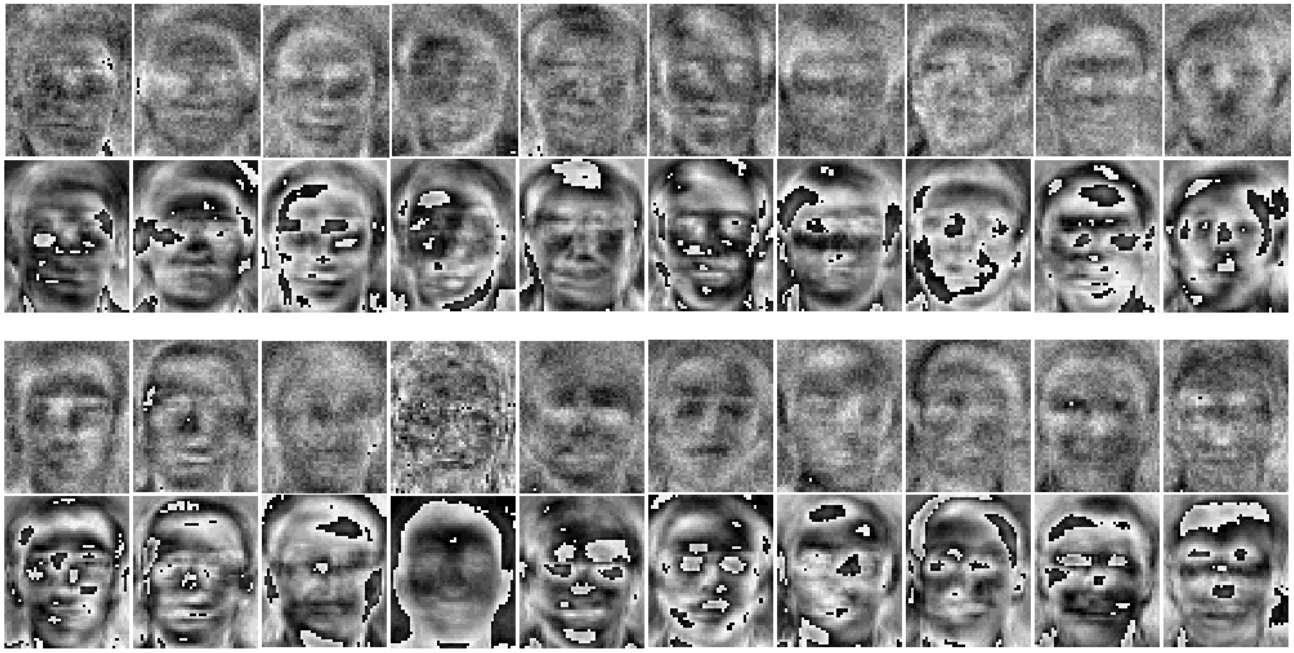

Figure 3. Weights of input and output layers show similarity to eigenvectors. First and third rows are representations of the input weights, and second and fourth rows of output weights, respectively.

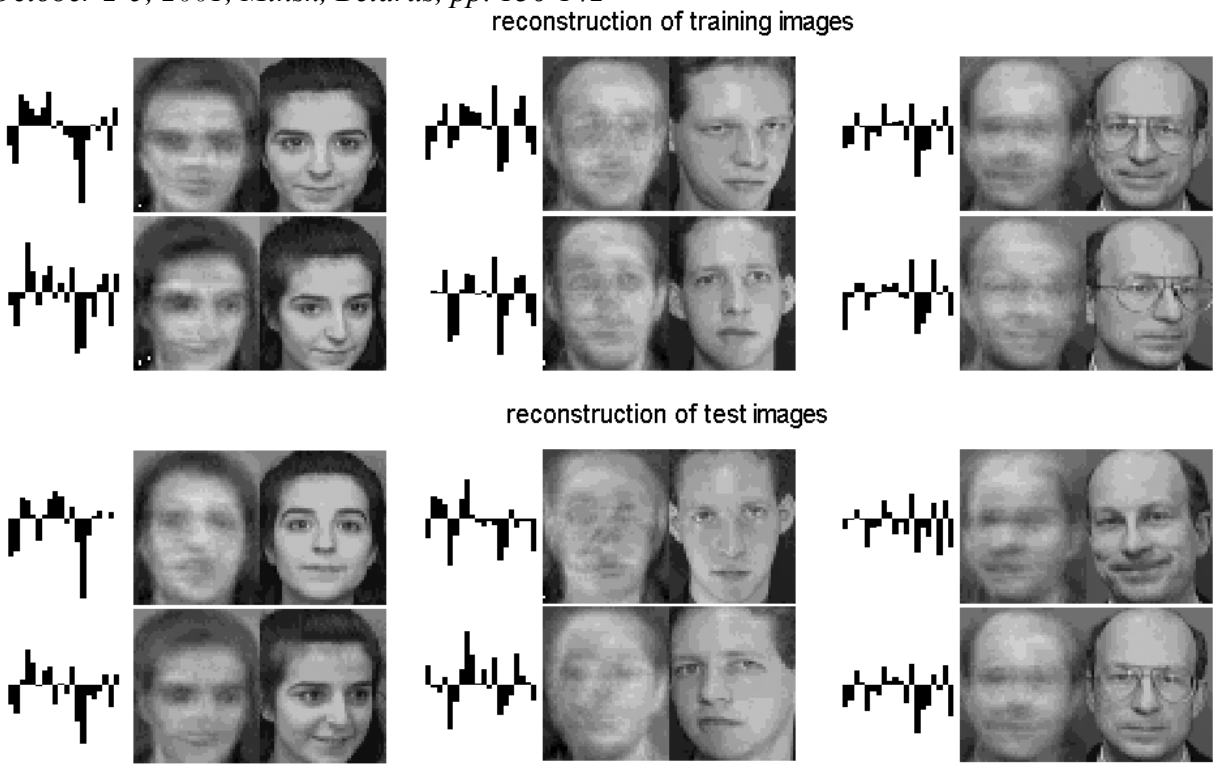

Figure 4. Reconstruction of training and test images with their compressed representation (output of 20 hidden units).

training step	training time, sec	training rmse	test rmse	training error,%	test error,%
ORL/4, 20 hidden units				scale = 1280	
5	1	0,0863	0,0968	15	5
10	4	0,0653	0,0772	12,5	4
20	18	0,0636	0,0756	13	4
ORL/2, 20 hidden units, other division				scale = 2560	
0	0	0,5212	0,5181	23	22
1	6	0,1769	0,1855	23	18,5
2	6	0,1483	0,1593	20	13,5
3	7	0,1329	0,1454	14,5	10
4	7	0,1457	0,1564	13	8
5	6	0,1090	0,1219	13,5	9
10	41	0,0848	0,1028	10,5	6,5
20	66	0,0802	0,0989	9	7
100	655	0,0811	0,1000	10	5,5
ORL/1, 20 hidden units, other division				scale = 6400	
5	30	0,0859	0,0946	17	6
10	151	0,0688	0,0803	14	5
20	301	0,0628	0,0759	12,5	6,5

Table 1. Test of variations in image size and number of training cycles.

ORL/2, 100 training cycles				scale = 2560	
number of hidden units	training time, sec	training rmse	test rmse	training error,%	test error,%
10	377	0,1326	0,1451	12,5	6
20	655	0,0811	0,1000	10	5,5
30	960	0,0699	0,0919	8,5	4
40	1240	0,0504	0,0758	8	3,5

Table 2. Summary results for different number of hidden units.

ORL/2, 20 hidden units, 20 training cycles				scale = 2560	
Division	training time, sec	training rmse	test rmse	training error,%	test error,%
1	132	0,0966	0,1130	12,5	7
2		0,0830	0,0976	7,5	8,5
3		0,0882	0,1034	9	8,5
4		0,0829	0,0980	7,5	8,5
5		0,0844	0,1014	7	10

Table 3. Recognition errors for several divisions into training and test sets.

correct recognition			**wrong recognition**		
image	name	distance	image	name	distance
	s1/2	unknown image		s1/6	unknown image
	s1/5	0.3182		s4/6	0.3161
	s13/6	0.3551		s1/1	0.3357
	s18/5	0.3944		s12/1	0.3361
	s13/5	0.3974		s13/7	0.3425
	s5/3	0.3984		s5/5	0.3432

Table 4. Example of recognition. Left part is correct recognition, right part is wrong recognition. Upper row presents two unknown images. Next images are closest to the unknown images in terms of the Euclidean distance.

5. REFERENCES

1. D. Valentin, H. Abdi, A. O'Toole, and G. Cottrel, "Connectionist models of face processing: A survey," Pattern Recognition, vol. 27, pp. 1209-1230, 1994.
2. A.I. Wasserman, "Neural Computing: Theory and Practice," Van Nostrand Reinhold, New York, 1989.
3. V. Golovko and V. Gladyschyk, "Recirculation neural network training for image processing," Advanced Computer Systems, pp. 73-78, 1999.
4. D.B. Graham and N.M. Allinson "Face recognition using virtual parametric eigenspace signatures," Image Processing and its Applications, pp. 106-110, 1997.
5. T. Vetter and T. Poggio, "Linear Object Classes and Image Synthesis From a Single Example Image," IEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no 7, pp. 733-742, 1997.