FACE RECOGNITION BY COUNTER-PROPAGATION NETWORKS

Juan Moreno[1], Miguel A. Fernandez[2],
Francisco J. Gomez[2] & Antonio Fernandez-Caballero[2]
[1]Departamento de Informatica, E.U. de Ingenieria Tecnica Industrial,
Universidad de Castilla-La Mancha, 45071 – Toledo, España
Mail: jmoreno@inf-cr.uclm.es
[2]Departamento de Informatica, Escuela Politecnica Superior,
Universidad de Castilla-La Mancha, 02071 – Albacete, España
Mail: {miki, fgomez, caballer}@info-ab.uclm.es

Abstract

The functionality of counter-propagation nets applied to face recognition is presented in this paper. The chosen procedure basically transforms the image into a vector of numbers and passes them to the net's input layer. The tested input vector is either a vector of the image grey levels or a vector of the histograms of the image's rows. The counter-propagation networks obtain an excellent result of over an 80% of guesses of face recognition in both studied cases.

1. INTRODUCTION

This paper aims to check the efficiency of the counter-propagation nets (CPNs) in face recognition. To do this, a simulator of a CPN has been software implemented. Our simulator incorporates a learning process to remember the faces of 16 different persons. Later on it receives as input 6 variations of the learned faces, providing as output a code representing one of the learned faces.

Hecht-Nielsen [1] synthesised the CPN by combining a structure known as competitive net with *Grossberg*'s *outstar* structure [2], obtaining this way the so called counter-propagation net [3] [4]. The general operation for these kinds of nets is as shown in figure 1.

Given a group of vectors (x_1,y_1), (x_2,y_2), ...,(x_L,y_L), the net is able to learn how to associate a vector X of the input layer with a vector Y of the output layer. If the relationship among X and Y can be defined by means of a continuous function Ω, such that $Y = \Omega(X)$, then the net will be able to learn how to approximate that correspondence for all value of X in the interval specified by the set of training vectors.

As you may appreciate on figure 1 the architecture consists of three layers. Layer 1 units receive as input a face image. They then process the image either as a vector of grey levels or as a vector of the image's row histograms. Afterwards they normalise this vector. All units of layer 2 receive the same normalised input vector, and calculate their output intensity thanks to a learned weight vector and to the specific vector got as input. The unit of maximal output intensity sends a *1* to layer 3 while the rest send a *0*. Finally, layer 3 selects the output image of among all the learned ones.

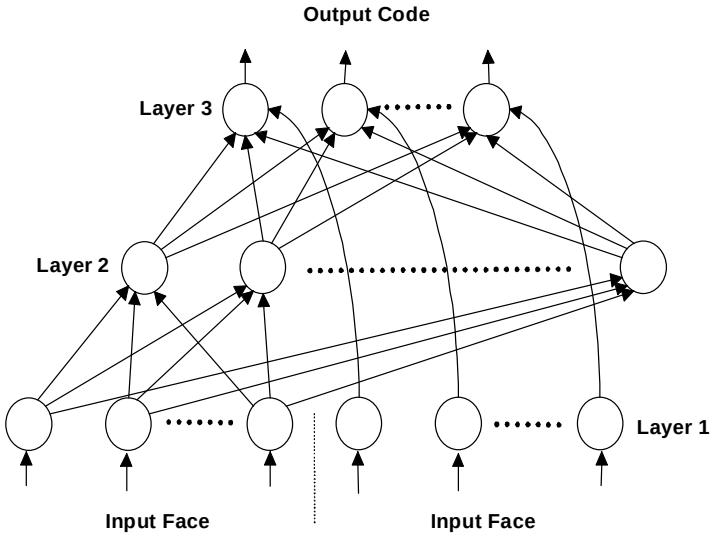

Figure 1: The CPN structure

2. THE CPN DESCRIPTION

2.1. Layer 1

Let us consider layer 1 on figure 1. The total intensity of the input vector is governed by equation $I = \sum_i I_i$ In close relationship to each I_i we shall define a magnitude as the one shown in equation 2.1. Vector $(\vartheta_1, \vartheta_2, \cdots \vartheta_n)^t$ is called a

reflectance plot as in *Freeman and Skapura* [5]. Observe that this plot is normalised, that is to say $\sum \partial_i = 1$.

$$\vartheta = I_i \left(\sum_i I_i \right)^{-1} \qquad (2.1)$$

The reflectance plot is independent of the total intensity of the corresponding input plot. For example, the reflectance plot corresponding to a person's face image is independent of the person's image brightness. Equation 2.2 represents layer 1 output, where $0 < x_i(0) < B$, $A, B > 0$.

$$\dot{x}_i = -Ax_i + (B - x_i)I_i - x_i \sum_{k \neq i} I_k \qquad (2.2)$$

The processing elements quickly reach a balance state once input vector *X* is applied $(\dot{x} = 0)$ [5]. When software simulating the CPN you can simplify the program just by normalising the input vectors. Our simulator normalises according to equation 2.3.

$$x_i = I_i / \left(\sqrt{\sum_n I_i^2} \right) \qquad (2.3)$$

2.2. Layer 2

Layer 2 is what we know as a competitive network. It is also often called the hidden layer. It consists of a series of processing elements called *instar*s [2][5]. Input vector *I* and weight vector *w* have been normalised in this case. The *instar*'s output is governed by equation 2.4, where *neta=I*w* and *a,b>0*.

$$\dot{y} = -ay + b * neta \qquad (2.4)$$

The *instar* units reach the balance value whenever $y^{eq} = -ay + b * neta$. Values *a* and *b* have been given the same value in simulation, so they are just eliminated of the formula.

The competitive net classifies any input vector. That particular *instar* with the greatest output value is the winner of the competition, and it will be the only one

offering a non null output. So the winner will send a value of *1* to the *outstar*, and the rest of the competitive net's *instar*s will send the value *0*.

2.3. Layer 3

Layer 3 consists of some processing elements called *outstar*s. Each *outstar* takes an output value according to equation 2.5, where w_i^{eq} is the weight value found during the learning phase [1] [2].

$$\dot{y}'_i = -ay'_i + c\, w_i^{eq} \qquad (2.5)$$

The *outstar* quickly reaches a balance value equal to the weight value existing in the connection coming from the competitive net's winner unit. An easy way to understand this processing is to realise that the *outstar*'s balance output is equal to equation 2.6, where z_j is the input received from the corresponding *instar* of the competitive net.

$$y_k^{'eq} = \sum_j w_{kj} z_j \qquad (2.6)$$

Moreover, since $z_j = 0$ unless $j = i$, equation 2.7 represents the *outstar*'s output.

$$y_k^{'eq} = w_{ki} z_i = w_{ki} \qquad (2.7)$$

In our particular simulation, the *outstar*'s output is actually obtained by means of equation 2.7.

3. OUR PARTICULAR CPN IMPLEMENTATION

As commented before, our system carries out a learning phase. The CPN memorises the faces of the series 1 of the 16 different persons of figures 2 and 3. In a second step, all faces of figures 2 and 3 are shown to the system. Later on, the system returns as a result a code representing each of the learned images. In the example shown in this paper, our application worked with 256 grey level images of 112 lines per 92 columns each.

The outline of our CPN is the following one:

- Layer 1 consists of 112*92 nodes in input vector *x* corresponding to the input vector length. With regard to inputs *y*, we use 4 nodes corresponding to the 4 binary values composing the code for each face to learn.

- The hidden layer, that's to say, the competitive net, is formed of 16 *instar* nodes, where each *instar* learns one single face.

- Layer 3, the output layer, consists of 4 nodes, each one learning a binary number that identifies one of the components of the output code. The output of this layer will be the selected face.

We have chosen the counter-propagation net to implement this system due to a couple of strong reasons. First there is our previous experience in using it for the detection of vectors. In *Moreno et al.* [6] [7] some good results were obtained. So we have continued looking for even better results. We have also kept in mind that the greater the number of the vector's co-ordinates the better the classes should be distributed in space. And finally, the counter-propagation nets use different learning algorithms for each layer, allowing the net to be trained very quickly.

To explain why we have chosen the possibility to work with histograms of image rows, think about the problem that occurs to the CPN when the face doesn't appear exactly in the same position as in the learned image. Using histograms solves problem related to this fact.

We consider that a set of 16 faces is a significant sample to check the validity of the counter-propagation net for these kinds of problem statements, although we are certain that with a greater number of faces the CPN would arise a worse behaviour.

Before closing this section, let's comment that for each input value *i* an input intensity value of *i+1* has been chosen, so that when *i=0* an input value different from *0* is provided.

4. THE CPN LEARNING PHASE

The CPN learning phase is carried out at a layer by layer basis. Let's then focus on learning by the same basis.

Layer 1 doesn't incorporate any learning at all, since it has only to normalise the input values.

Consider again that, as it has already been said, layer 2 is formed by some elements called *instar*s. Each *instar* learns the weight vector *w*. The learning phase as described in *Moreno et al.* [6] [7] is carried out starting from the initial weight vector *w* that goes evolving according to differential equation 4.1, where *y* is the output value, and *c, d > 0*.

$$\dot{w} = -cw + dIy \qquad (4.1)$$

The mission of any *instar* is to memorise an input vector, providing a greater intensity output the more the input vector resembles the learned vector. When software simulating, it is possible to simplify the *instar*'s learning phase by assigning the weight values of vector *w* directly starting from the values of the already normalised vector y .

Finally, the learning phase [6] [7] of each *outstar* of layer 3 evolves according to equation 4.2, where parameters *a, b, c>0* and the value of $neta_i$ is calculated as previously described.

$$\dot{y}'_i = -ay'_i + by_i + c * neta_i \qquad (4.2)$$

To the effects of the digital simulation, this learning may be approximated assigning the weight values of the *outstar* directly from the input vectors.

5. RESULTS

Firstly, we are thankful to *The ORL Database of Faces* (*www.cam-orl.co.uk/facedatabase.html*) for their courtesy in the public offer of images of figures 2 and 3 used in our tests.

The simulator first carries out the learning phase of the diverse layers. And next it sends to the input layers of both nets all the faces to be tested (images 1 to 16 of figures 2 and 3), one by one.

Two kinds of tests have been performed: (a) passing each image as a vector of grey levels, (b) passing each image as a vector of the image rows histograms. The results obtained for both (a) and (b) are excellent, in both cases over an 80% of guessed faces.

Table 1 shows the guessed images for case (a). The global success is of an 83.3%. One of the detected problems appears when the input image takes a different brightness from the learned image, despite the normalisation carried out at layer 1.

Some examples of it are images 1 (series 2 and 4), 15 (series 2 and 5) and 16 (series 2 and 3). Another less important problem is the presence or absence of glasses. See image 4 (series 5 and 6), for example. Note nevertheless that for this same image the absence of glasses in series 4 doesn't suppose any problem. See also that in images 7 and 13 this doesn't suppose any problem.

Table 2 offers a view of the images correctly guessed for case (b). The global success in this case is of an 86.6%. In this series the previously described problem of the glasses is completely solved. The change in brightness between the learned image and the input image still continues causing confusions in the detection of the images, as shown in images 3 (series 6), 9 (series 4,5,y 6) and 16 (series 2, 3, 4 and 5).

Images	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Series 1	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Series 2		✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		
Series 3	✔	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔	✔	✔	✔	
Series 4		✔	✔	✔	✔	✔	✔	✔	✔			✔	✔	✔	✔	✔
Series 5	✔	✔			✔	✔	✔	✔	✔			✔	✔	✔		✔
Series 6	✔		✔		✔	✔	✔	✔		✔	✔	✔	✔	✔	✔	✔

Table 1: Results for the CPN using vectors of grey levels

Images	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Series 1	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Series 2	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔		
Series 3	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔	✔	
Series 4	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔		✔	✔	✔	
Series 5	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔		✔	✔		
Series 6	✔	✔		✔	✔	✔	✔	✔		✔	✔	✔	✔	✔	✔	✔

Table 2: Results for the CPN using vectors of row histograms of the images

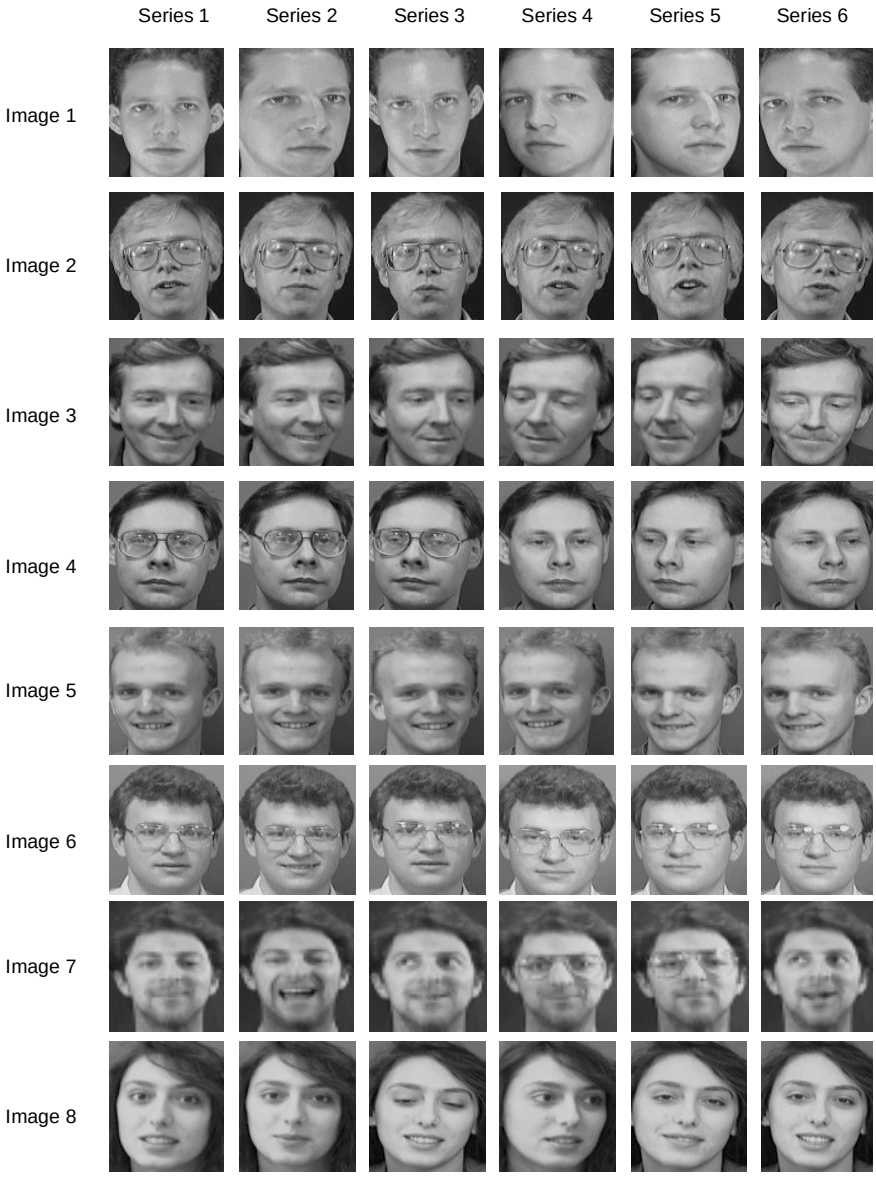

Figure 2: Series of faces (images 1-8)

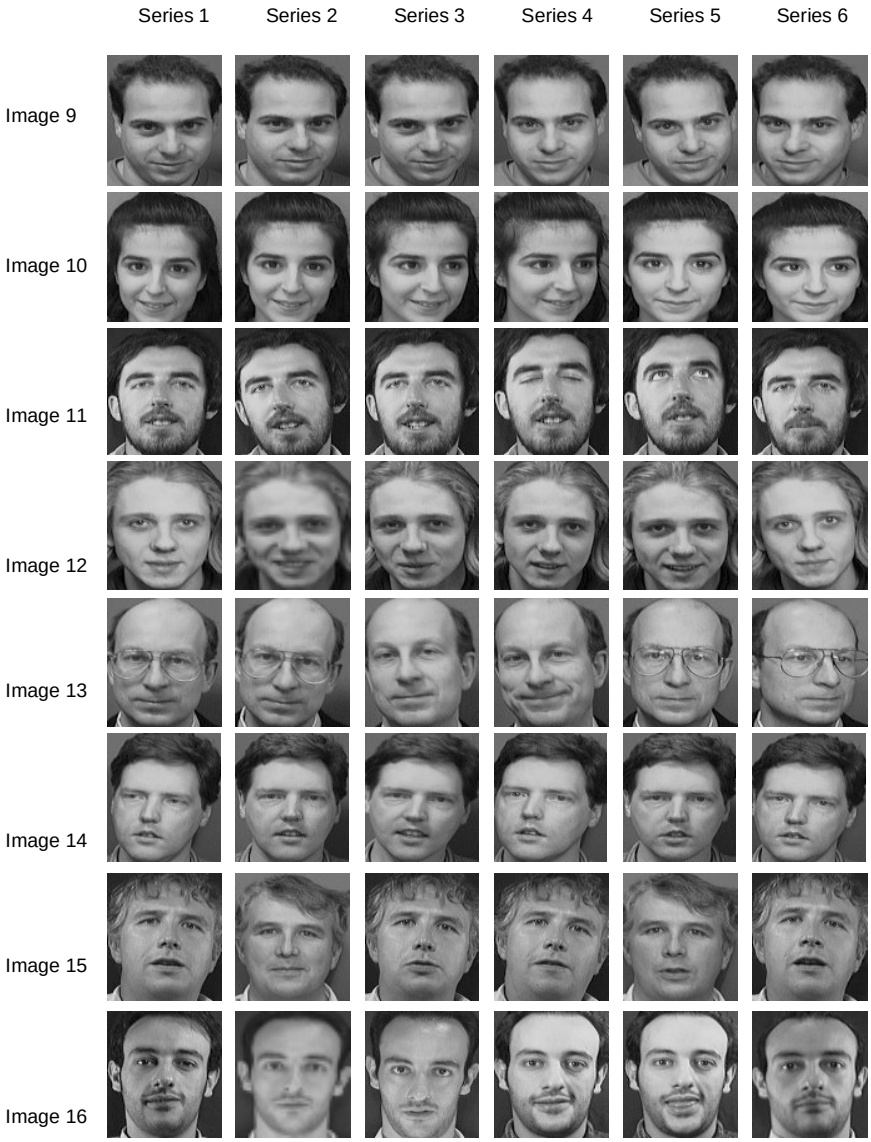

Figure 3: Series of faces (images 9-16)

6. CONCLUSIONS

The CPN is a good tool for face recognition, as demonstrated by means of the obtained results, transforming the image into a vector of grey levels, as well as taking the image as a vector of image rows histograms.

The difference of brightness between the learned image and the tested one may cause confusions in occasions, although the normalisation phase of layer 1 avoids the problem in many cases.

Transforming the images into a vector of image rows histograms enhances the results of the previous case. This is due to the fact that the input is the row histogram and the relative position of the face in the image doesn't alter the result.

References

[1] R. Hecht-Nielsen, *Neurocomputing,* Addison-Wesley, Reading MA, 1990.

[2] S. Grossberg, *Studies of Mind and Brain. Boston Studies in the Philosophy of Science,* vol. 7, D. Reidel Publishing Company, Boston, 1982.

[3] R. Hecht-Nielsen, "Counterpropagation Networks", *Applied Optics*, vol. 26, no. 23**,** pp. 4979-4984, 1987.

[4] R. Hecht-Nielsen, "Counterpropagation Networks", in *Proceedings of the IEEE First International Conference on Neural Networks*, II-19-II-32, Piscataway, NJ, 1987.

[5] J.A. Freeman and D.M. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley, Reading, MA, 1991.

[6] J. Moreno, G. Sebastian, M.A. Fernandez & A. Fernandez-Caballero, "A Telephone Number Corrector using a Counterpropagation Network", in *Proceedings of the Fifth International Conference on Neural Information Processing ICONIP´98,* vol 2, pp. 1168-1171 Kitakyushu, Japan, 1998.

[7] J. Moreno, G. Sebastian, M.A. Fernandez & A. Fernandez-Caballero, "Comparison of counter-propagation and spatio-temporal nets in the detection of sequences of numbers", in *Proceedings of the Eighth Turkish Symposium on Artificial Intelligence and Neural Networks TAINN'99,* pp. 145-153, 1999.