RECURRENT AND CONCURRENT NEURAL NETWORKS FOR OBJECTS RECOGNITION

Federico Cecconi
Laboratory of Artificial Life and Robotics
Institute of Cognitive Sciences and Technologies CNR
Via S. Martino della Battaglia 44,
ROME, Italy
email: federico.cecconi@istc.cnr.it

Marco Campenní
Laboratory of Artificial Life and Robotics
Institute of Cognitive Sciences and Technologies CNR
Via S. Martino della Battaglia 44,
ROME, Italy

ABSTRACT
A system based on a neural network framework is considered. We used two neural networks, an Elman network [1][2] and a Kohonen (concurrent) network [3], for a categorization task. The input of the system are objects derived from three general prototypes: circle, square, polygon. We varied the size and orientation of the objects in a continuous way. The system is trained using a new algorithm, based on recurrent version of backpropagation and Kohonen rule. The system achieves the capacity to predict the shape of the objects with a remarkable generalization [4]. We compare our results with the results using a classical Elman network. The model is implemented by a Matlab/Simulink environment.

KEY WORDS
Artificial neural networks; recurrent neural networks; object recognition; artificial vision; autonomous robotics.

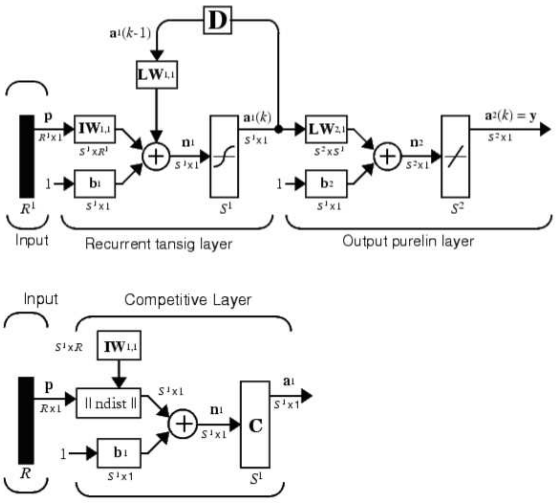

Figure 1. Examples of Elman (above) and Kohonen (below) network.

1 Introduction

One important task in artificial vision (and more in general in the autonomous robotics field) is the capacity of a system to predict the shape (the contour) of objects with a reasonable degree of generalization [5][6][7]. Our model is based on artificial neural networks: they are bio-inspired networks of neuron-like system that work together to carry out intelligent computing task. There are many different kinds of neural networks: we use Elman networks and Kohonen networks.

The Elman network commonly is a two-layer network with feedback from the first-layer output to the first layer input. (see first schema in figure 1). The Elman network differs from conventional two-layer networks in that the first layer has a recurrent connection (in figure 1 D is the feedback). The delay in this connection stores values from the previous time step, which can be used in the current time step. This recurrent connection allows the Elman network to both detect and generate time-varying patterns.

Another kind of neural network is competitive network, in our model Kohonen network (see the second schema in figure 1). As usual, inputs are applied to the input layer, and outputs from the output layer nodes is considered. A node with its weight vector which is closest to the vector of inputs is declared the winner, and only its weights are adjusted by the Kohonen rule. This process is then repeated for each input vector, over and over, for a number (usually large) of cycles. The Kohonen learning rule is $\Delta\, IW_{ij} = \eta\Lambda(i, i^*)(\xi_j - IW_{ij})$, IW is the weight matrix, i and j are input and output nodes, i^* is the winner, η is a learning parameter. Λ is a neighborhood function that returns 1 for $i = i^*$ and falls off with distance $|r_i - r_{i^*}|$ between units i and i^* in the output array. The Kohonen network is a classifier: it divide the inputs into different output categories.

A typical task in which we can test neural network ability is perception. We identify two different kinds of perception: active perception and passive perception [8][9][10][11]. In passive perception the stimulus is presented as a pattern of inputs, 'hic et nunc'; there is no real interaction with the input. Vice versa in active

perception there is an interaction between the subject and the environment; the stimulus is acquired step-by-step, during the time [12][13][14]. A connected tasks for every action of perception, passive or active, is prediction [15]. A typical example of neural network application regarding the problem of perception and prediction is given in [16]. These authors explore the possibility of providing robots with an `inner world` based on internal simulation of perception, rather than an explicit representational world model. Recurrent neural network is evolved to control collision-free corridor following behavior in a simulated Khepera robot and it predicts the next time step's sensory input as accurately as possible.

In a neural network simulation we can distinguish two different steps; typically there is a learning phase and then a test phase.

During the learning phase the weights in the system will be modified. For Elman network, a pattern is presented at the inputs units; the pattern will be transformed in its passage through the layers of the networks until it reaches the output layer. The outputs of the networks as they are now compared with the outputs as they ideally would have been if this pattern were correctly classified. On the basis of this comparison all the connection weights are modified a little bit. The differences between the actual outputs and the idealized outputs are propagated back from the top layer to lower layers in order to be used at these layers to modify connection weights. For Kohonen network, we use the Kohonen learning rule (see section 3 for details).

Vice versa during the test phase a new pattern of inputs is presented to the system. The weights are not adjusted; the system 'manipulates' the information and tries to solve the problem.

The paper is divided in three parts:

- The description of the model is in section 2. We describe the features of the objects, the structure of the system, the learning algorithm and the parameters. We show that the input of the system is a periodic signal and we analyze the spectrum by a DFT tool.
- In the section 3 we describe a new algorithm to classify the objects.
- In the sections 4 and 5 we describe the results of the simulations and we sketch some general conclusions.

2 The model

2.1 The objects

We use three prototypes to create the objects. We call them `square`, `circle` and `polygon` prototype (see table 1).

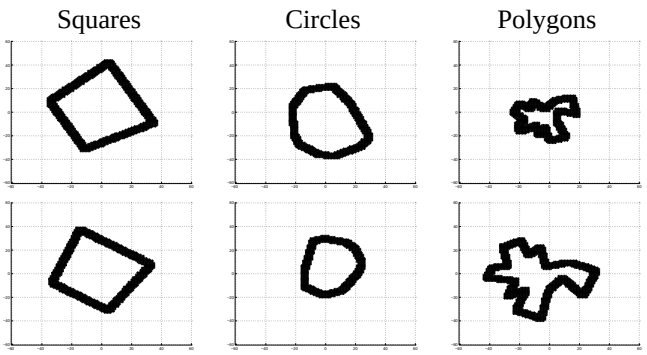

Table 1. Examples of input figures. We use three prototypes, with a variable orientation and size. Two figures from `square` prototype (left), two from `circle` prototype (center) , at the end two figure from `polygon` prototype (right)

We put the figure at the center of a square lattice with side $L = 121$ point. The simulation is divided into discrete n-time steps. An automa goes round the object, keeping constant the distance from the center of lattice and the angular velocity, $v = 0.0349\ rad/sec$ (see figure 2). During one step, we say $n = 0$, the automa store $\kappa_i = 10$ distances from the positions $p^i_{n=0}$, $i = [p, p+\rho, p+2\rho, \ldots, p+\kappa_i\rho]$ to the nearest point on the contour of the figure (obviously coming from a position to another the automa covers always an angle $\rho = 0.0349\ rad$; the angular velocity is constant). In other words, the automa receives as input a vector $\overrightarrow{V}\ \lceil\kappa_i\rfloor$ of sampled distances. The rate of sampling is not constant: in fact, for small figure, the rate decreases (this is an another consequence of an constant angular velocity). However, this information is not available to the system in an explicit way.

The our implicit categorization task consists in prediction of the next $\kappa_o = 10$ distances. The system will compute an output vector $\overrightarrow{O}\ \lceil\kappa_0\rfloor$. The difference between prediction and real distances is the error of the system. Then the system goes to the next step $n = 1$, and restart the sampling-prediction procedure. During the teaching phase the system runs on 60 objects (see section 3 for details). During the test phase we used new objects; from the prototypes we have generated new instances with different orientation and size. So, our task is a generalization task: we are not interested in the capacity of the system to store all 60 object's shapes. We study the capability of the system to extract some general features, and to predict the shapes using, for example, the frequency of changes in the input vector.

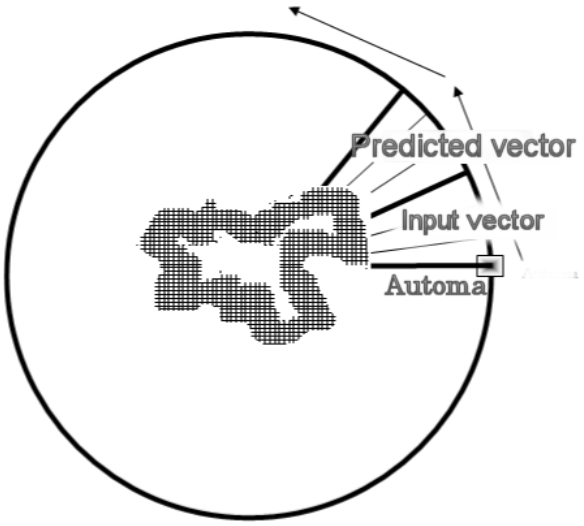

Figure 2. An automa goes around the objects, sampling distance from its contour. The task is prediction of following $K_o = 10$ distances, stored in a O vector

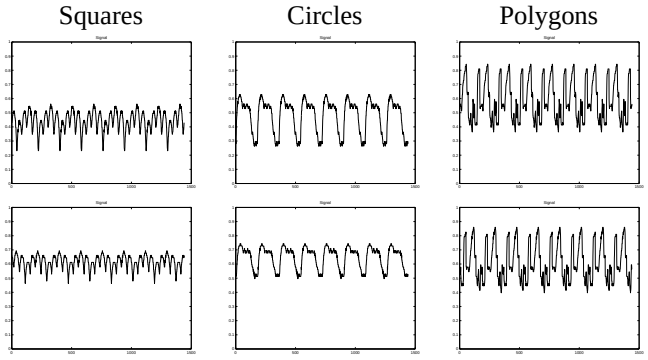

Table 2. Examples of periodic signals, obtained sampling two *square* object, two *circle* and two *polygon*

2.2 Analysis of the contour of the objects. DFT results

The table 2 shows the sampled distances on a) two `square` objects, b) two `circle`, c) two `polygon`. These curves can be interpreted as a periodic signal. The form of signals derives from the geometrical features of the objects, for example the size, or the prototype. We have analyzed the spectrum of these signals by DTF (Discrete Fourier Transform). As a matter a fact, it is difficult to identify the frequency components by looking at the original signals. Converting to the frequency domain, the discrete Fourier transform is found by taking the 512-point fast Fourier transform (FFT): we show that there are three typical patterns for three different prototypes (in the table 3): a) two bars for `square`, b) one bar for `circle`, c) many bars for `polygon` with height decreasing in function of the frequency.

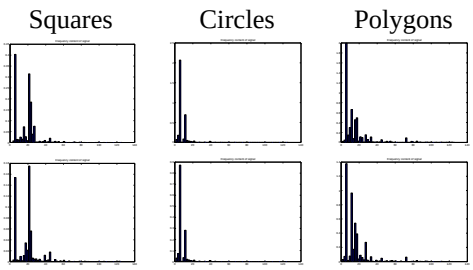

Table 3. Spectrum of the periodic signals from table 2

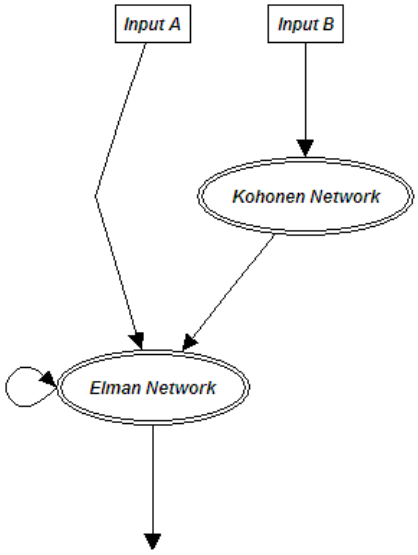

Figure 3. The system scheme. $InputA$ is the vector $\vec{V}\lceil\kappa_i\rfloor$; the elements are the distances between automa and the object. $InputB$ is a vector with size $\kappa_i - 1$; the elements are the discrete derivative of $\vec{V}\lceil\kappa_i\rfloor$ (see section 3)

2.3 System structure

Neural networks have been shown to be successful as predictive tools in a variety of tasks, predicting the time at which an event will occur, or predicting the level of some outcome. In our task, two main factors have an influence: a) the prediction is a vector; it is not sufficient to predict the following value in a function; b) the regularities of the objects are not explicit (see DFT result). Then, the system have to extract the regularities.

To address these problems, we used an integrated approach (see figure 3).

The input is divided into two parts: $InputA$, that contains the vector of the distances; $InputB$, that contains a vector $\vec{Diff}\lceil\kappa_i - 1\rfloor$. We build $\vec{Diff}$ computing the discrete derivative of $InputA$ (see section 3).

The $InputA$ is processed directly by the Elman

network. The $InputB$ is processed by the Kohonen network: the output is a classification of the $InputB$ in 16 patterns, that classifies different $\vec{Diff}$. For example the Kohonen network can identify flat slope (the distance remain constant during the κ_i sampling) with pattern 0010, and the bell curve slope (the distance grows and decreases during the κ_i sampling) with the pattern 1010.

Concluding, the Elman network receives two different inputs: $InputA$, without preprocessing, informs the Elman network about the dynamics of the distances; $InputB$, with derivative preprocessing, informs the network about the actual state of the distances. Naturally the output of the system is the vector $\vec{O}\ \lceil \kappa_0 \rfloor$, with the predicted distances.

3 Learning algorithm and parameters

Elman network is a recurrent network (see figure 1). As we have already said, the delay in this connection stores values from the previous time step, which can be used in the current time step. Thus, even if two Elman networks, with the same weights and biases, are given identical inputs at a given time step, their outputs can be different due to different feedback states. Because the network can store information for future reference, it is able to learn temporal patterns as well as spatial patterns. The Elman network can be trained to respond to, and to generate, both kinds of patterns. The teaching function is a classical backpropagation rule. We use 25 hidden units, a $logsig$ transfer function $f(n) = 1/(1+e(-n))$ where n is the net input. We train the network for 30 epochs for each input pattern. We use 3600 input patterns and we move the automa around the object for 10 step. The learning rate is 0.8.

The Kohonen network is a competitive network (see figure 1). $\|NDist\|$ accepts the input vector $\vec{p}$ and the input weight matrix $IW^{1,1}$, and produces a vector having S^1 elements. The elements are the negative of the distances between the input vector and formed from the rows of the input weight matrix. The C transfer function accepts a net input vector for a layer and returns neuron outputs of 0 for all neurons except for the winner, the neuron associated with the most positive element of net input n_1. The weights of the winning neuron (a row of the input weight matrix) are adjusted with the Kohonen learning rule. Supposing that the i_{th} neuron wins, the elements i_{th} of the row of the input weight matrix are adjusted as shown below:

$$dW = \alpha(P - W),\ if P \neq 0,\ 0\ otherwise \qquad (1)$$

with α learning rate, neuron's input P.

We used a novel teaching algorithm: we teach Kohonen network to discriminate on $10,000$ $\vec{Diff}$, obtained sampling 60 objects. $\vec{Diff}$ is a discrete derivative of $\vec{V}$.

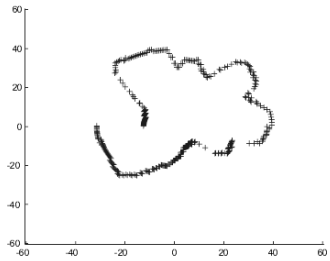

Figure 4. How the automa see an object

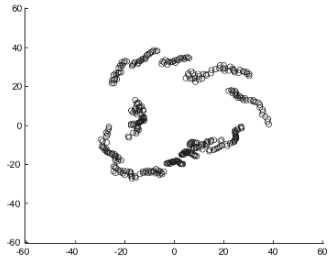

Figure 5. How the automa predict an object

Each element of $\vec{Diff}$ is obtained by the difference between $\vec{V}_i$ and $\vec{V}_{i+1}$. For example, if the vector for the distances is

$$\langle 0.5, 0.5, 0.5, 0.1, 0.1, 0.5, 0.5, 0.5, 0.5, 0.5 \rangle$$

,$\vec{Diff}$ will be

$$\langle 0, 0, 0.5, 0, -0.5, 0, 0, 0, 0 \rangle$$

Kohonen network has 4 neurons for output. So, the $10,000$ $\vec{Diff}$ will be categorized into 16 pattern.

4 Experimental results

We test the performance of the systems *with objects that it has never seen before* (of course, coming from one of the three prototypes). We plot the real distances between automa and objects (in table 4 the continuous line), together with the predicted distances (in table 4 line with circle). We find a good correspondence between real and predicted distances, but we also find the presence of a systematic error (indicated in the figures by small arrows). In figure 4 we draw how the automa sees an object (a polygon), plotting the distances. In figure 5 we draw how the automa *predicts* the object

We have computed ς, the average of the absolute difference between output of the system at step n and the real distances at step $n+1$.The table 5 shows the results.

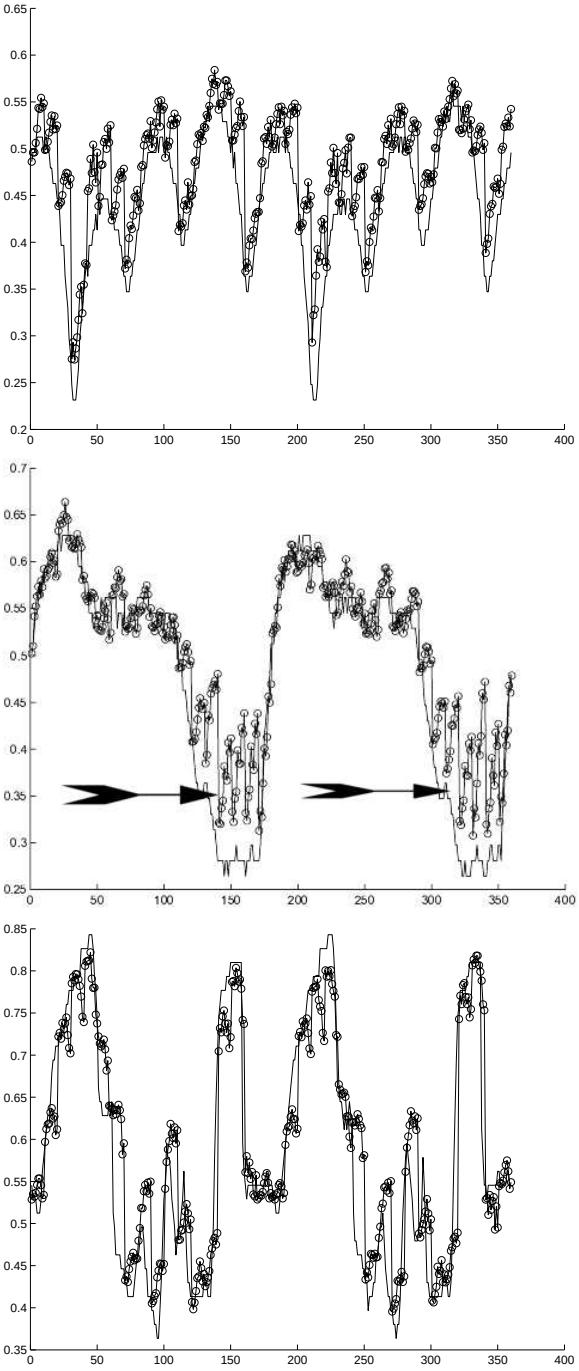

Table 4. The real and the predicted distances for one `square`, one `circle`, and one `polygon`

System structure	ς
Elman + Kohonen (our system)	0.0311
Elman	0.0416
Kohonen	0.354
Simple FF Network	0.421

Table 5. ς, the average of the absolute difference between output of the system at step n and the real distances at step $n+1$

5 Conclusions and future work

This paper describe a two-neural-network-based system for object categorization. We use an Elman recurrent network and a Kohonen concurrent network. An automa goes around the objects, sampling distances from surface of the object to the automa: the inputs of the system are the distance between automa and object. The task is the prediction of next distances.

We can sketch some general conclusions; (a) We study a task of active perception: the input is not a single pattern, but is a sequence of values. The sequence can be analyzed by a classical tools (for example DFT, see paragraph 2.2). From this sequence, the system extracts the internal features (more than DFT results) to compute the prediction task; (b) The system categorizes the object correctly (see tables 4 and 5). The prediction of the next vector of distances is rather precise. The system is very simple, with a reduced number of components, and the learning phase is executed off line. These features of our system are the main difference with the PCNN approach [11]. (c) During the test phase, the system see new objects. So, we can say that the system generalize the internal features.

Nevertheless the networks carry out a good prediction performance, we observe systematic errors, for example in the prediction of the circles (see arrows in table 4). We guess that the problem originates from the fixed path (a circle) that the automa follows during the categorization task. In these months, we have begun to download the system in a Khepera robot [17]. We could evolve an control system so that the Khepera approaches to the object, and then it keeps on following the contour. We are studying like integrating the networks of prediction with this new control system.

6 Acknowldgements

The research has been supported by the ECAGENTS project founded by the Future and Emerging Technologies programme (IST-FET) of the European Community under EU RD contract IST-2003-1940.

References

[1] J. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[2] Federico Cecconi and Domenico Parisi. Networks that learn to predict where the food is and also to eat it. *Proc. IJCNN International Joint Conference on Neural Networks*, Washington, USA, 1989.

[3] T. Kohonen. *Self-Organizing Maps.* (Springer, Berlin, 1995).

[4] L.A. Machowski and T. Marwala. Representing and classifying 2d shapes of real-world objects using neural networks. *Proc. IEEE International Conf. on Systems, Man and Cybernetics*, The Hague, The Netherlands, 2004,7:6366-6372.

[5] B. Sun and F. Takeda. Research on the neural network with rbf for currency recognition. *Proc. IASTED International Conf. on Neural Networks and Computational Intelligence*, Grindelwald, Switzerland, 2004, 201-205.

[6] C.-B. Liu and N. Ahuja. A model for dynamic shape and its applications. *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, USA, 2004, 2:129-134.

[7] Pramath R. Sinha and Ruzena K. Bajcsy. Robotic exploration of surfaces and its application to legged locomotion. *Proc. IEEE International Conf. on Robotics and Automation*, Nice, France, 1992, 221-226.

[8] Jeffrey A. Fayman, Ehud Rivlin, and Henrik I. Christensen. System for active vision driven robotics. *Proc. IEEE, International Conf. on Robotics and Automation*, Minneapolis, Minnesota, USA, 1996, 3:1986-1992.

[9] N.J.T. Thomas. Are theories of imagery theories of imagination? an active perception approach to conscious mental content. *Cognitive Science*, 23(2):207–245, 1999.

[10] Daesik Jang and Hyung-Il Choi. Moving object tracking using active models. *Proc. IEEE International Conf. on Image Processing*, Genova, Italy, 1998, 3:648-652.

[11] R. Eckhorn, H. J. Reitboeck, M. Arndt, and P. W. Dicke. Feature linking via synchronization among distributed assemblies: Simulation of results from cat cortex. *Neural Computation*, 2:293–307, 1990.

[12] M. Shanahan. Perception as abduction: Turning sensor data into meaningful representation. *Cognitive Science*, 29(1):103–134, 2005.

[13] J. Tani and S. Nolfi. Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems. *Neural Networks*, 12:1131–1141, 1999.

[14] G. Cannata and E. Grosso. Active eye-head control. *Proc. IEEE International Conf. on Robotics and Automation*, San Diego, USA, 1994, 4:2837-2843.

[15] A. Chella, M. Frixione, and S. Gaglio. A cognitive architecture for artificial vision. *Artificial Intelligence*, 89:73–111, 1997.

[16] T. Ziemke, D.A. Jirenhed, and G. Hesslow. Internal simulation of perception: a minimal neuro-robotic model. 2005.

[17] KTeam S.A. *Khepera user manual*. (http://www.k-team.com/download/khepera.html, 2005).