A Testbed for Neural-Network Models Capable of Integrating Information in Time

Stefano Zappacosta, Stefano Nolfi, and Gianluca Baldassarre*

LARAL-ISTC-CNR Laboratory of Autonomous Robotics and Artificial Life,
Istituto di Scienze e Tecnologie della Cognizione, Consiglio Nazionale delle Ricerche
Via San Martino della Battaglia 44, I-00185 Roma, Italy
{stefano.zappacosta, stefano.nolfi, gianluca.baldassarre}@istc.cnr.it
http://laral.istc.cnr.it

Abstract. This paper presents a set of techniques that allow generating a class of testbeds that can be used to test recurrent neural networks' capabilities of integrating information in time. In particular, the testbeds allow evaluating the capability of such models, and possibly other architectures and algorithms, of (a) categorizing different time series, (b) anticipating future signal levels on the basis of past ones, and (c) functioning robustly with respect to noise and other systematic random variations of the temporal and spatial properties of the input time series. The paper also presents a number of analysis tools that can be used to understand the functioning and organization of the dynamical internal representations that recurrent neural networks develop to acquire the aforementioned capabilities, including periodicity, repetitions, spikes, and levels and rates of change of input signals. The utility of the proposed testbeds is illustrated by testing and studying the capacity of Elman neural networks to predict and categorize different signals in two exemplary tasks.

Keywords: Testbed, Time Series, Waves, Time Information Integration, Signal Processing, Recurrent Neural Networks, Passive and Active Perception, Dynamical Systems, Analysis of Internal Representations, Attractors.

1 Introduction

The capability of integrating information in time is a critical functionality, which lies at the core of the functioning of several anticipatory learning systems. For example, consider a rat sampling the profile of an object with its whiskers [14], an organism scanning the environment with its eyes [15], or a robot moving in

* This research has been supported by the European Integrated Project "ICEA – Integrating Cognition, Emotion and Autonomy", contract no. FP6-IST-027819-IP, and by the European Specific Targeted Research or innovation Project "MindRACES – from Reactive to Anticipatory Cognitive Embodied Systems", contract no. FP6-511931-STREP.

M.V. Butz et al. (Eds.): ABiALS 2006, LNAI 4520, pp. 189–217, 2007.

an office and sampling the walls with proximity sensors [16]. Imagine they have to categorize the object or the portion of the environment they are experiencing, or to predict future sensations on the basis of past ones. In all these examples, the systems need to integrate information in time. That is, they need to capture signal regularities that manifest in time in the form of periodicity, repetitions, spikes, numbers, rates of change, levels of signals, etc.

Given the importance of integrating information in time for anticipatory systems, artificial intelligence has proposed a number of models that possess such capabilities. This paper focuses in particular on recurrent neural-network models, but the testbeds, and some of the analysis tools it proposes, are also applicable to other models. The neural networks relevant for the topic tackled here are based on recurrent architectures [4], [5], [6], and [20], as these allow systems to compare signals in time, for example, by counting signals' duration, by accumulating evidence in favor of different options, by synchronizing internal dynamics with perception dynamics, etc. Section 3 briefly reviews few important examples of these neural networks, namely *Elman neural networks* ([7]: these networks are based on a hidden unit layer with a memory of the past), *echo state networks* ([11]: these networks are provided with a layer of fixed recurrent connections that provides a "reservoir" of various dynamics), *leaky integrator networks* ([21]: these networks are based on neurons with an internal memory), and *long short-term memory networks* ([10]: these networks are based on special neurons with a self-recurrent connection and gated input and output channels).

The ways recurrent neural networks integrate information in time is particularly interesting for two reasons. First, these networks exploit internal dynamical processes such as fixed-point attractors, limit cycles, chaotic attractors, etc., to "get in resonance" and synchronize with the dynamics of stimuli and so perform the integration. Second, if one assumes that neural networks of real brains exploit similar dynamics on the basis of their omnipresent recurrent connections, one can hope to understand how real organism integrate information in time by studying artificial recurrent neural networks.

Given the interest of the aforementioned models, the ABiALS community (Adaptive Behavior in Adaptive Learning Systems) has a great need of identifying a number of specific testbeds to compare the models and, given their different features, to understand how they self-organize to solve different tasks. Indeed, such models should be compared both on the basis of their capabilities of mimicking real systems and on the basis of more general criteria such as scalability properties, computational power, and robustness against noise. The first of these two "checks" should be accomplished on the basis of the evaluation of the general biological plausibility of the models and by comparing the model's behavior and functioning with data of real systems, provided by behavior and brain sciences. The second check should be based on standard testbeds developed and circulated within the community, such as those proposed here.

The *testbed* presented in this paper, which is actually a set of techniques that can be used to produce a *class of testbeds* with particular features, allows testing two anticipatory capabilities of anticipatory systems:

(1) The capacity of categorizing different signals perceived in time.
(2) The capacity of predicting future signal values on the basis of past ones.

In this respect, this paper proposes some techniques to generate different signals with various time regularities to systematically test the models' capabilities of integrating information in time. The paper also proposes some techniques to test the *robustness* of such capabilities with respect to:

(1) Noise of the signal level and of the signal speed.
(2) Biased expansions and compressions of the signal in duration.
(3) Biased variations of the phase of periodic signals.
(4) Biased variations of the signal amplitude levels.
(5) Biased expansions and compressions of the signal levels.

The paper also presents a number of techniques that can be used to analyze the internal representations that the models develop to solve the various tasks. The understanding of such representations is rather challenging given the complex dynamical systems nature of the considered models. Notwithstanding these difficulties, we believe that these studies are necessary to understand the detailed mechanisms underlying the information integration in time that such systems exhibit.

The functioning of the testbed and the analysis techniques are illustrated through some experiments using Elman neural networks. These experiments represent the preliminary investigations of a research agenda directed to investigate how recurrent neural networks internally self-organize and form abstract dynamical representations in order to integrate information in time (see also the European Projects "ICEA" and "MindRACES", which provided funding for this research).

The rest of the paper is organized as follows. Section 2 presents the testbed, in particular, the type of time series it generates, the anticipatory capabilities it allows to test, the type of noise and signal variations it allows to create, the measures of performance it uses, and some techniques that can be used to analyze the emergent internal organization of the tested models. Section 3 presents a brief review of dynamical neural networks that might be tested and compared with the testbed presented. Section 4 gives some examples, based on Elman neural networks, that illustrate the functioning of the testbed and the analysis techniques. Finally, Section 5 draws conclusions and indicates future work.

2 Testbed Description

As mentioned in the introduction, the *testbed* presented here is actually a set of techniques that can be used to generate a class of different testbeds having certain properties. These testbeds have the following features:

(a) They involve problems where the model to be tested receives a one-variable input time series (henceforth called *signal* or *input time series*).

Table 1. Summary of the testbed's features

Possible different types of input signals

Type	Properties of signal that can be manipulated
Wall profile	Levels, linear changes, sudden changes of levels (steps)
Object	Non-linear changes, derivatives, discontinuities, sudden changes of levels (steps)

Tasks that can be used to test the systems' capabilities

Task	Metrics to measure the capability
Prediction	Mean square errors between predicted and actual input pattern, capacity to reproduce the signal for several steps by using the prediction as input
Categorization	Percent of correct categorizations after the pattern is perceived for a certain time

(b) They allow testing the models' capacity of both categorizing different signals and predicting future signal values based on past ones.
(c) They allow testing the robustness of these capacities with respect to noise and various systematic random transformations of the signal.

The features of the testbed(s), the possible variations of the input signal that can be used to test the robustness of the models, and the analysis techniques are summarized respectively in Table 1, Table 2, and Table 3. Note that, as the implementation of the testbed is quite straightforward, the software used to implement it can be easily re-generated by the reader. Moreover, the implementation of the analysis techniques suggested in the paper can be found in any standard statistical analysis package, for example Matlab^{TM}, which was used to carry out the results' analysis illustrated in the paper. Next, all the features and techniques are analyzed in detail.

2.1 Input Time Series

The testbed proposes two alternative techniques for generating the input time series. These two techniques allow manipulating different aspects of the input time series with different implementation ease (see Table 1). Nevertheless, notice that there is a precise correspondence between the input time series that can be generated in two ways, as further illustrated below.

The first technique involves an automaton that travels along a "wall" that has a certain profile, formed by a sequence of segments. An example of this is given in the top-left graph of Figure 1 that shows a "punctiform automaton" that moves at a distance R from a wall having a saw-like profile. The signal samples (shown in the bottom-left graph of the figure) encode the distance between the automaton's central point and the intersection between the sensor ray and the segments composing the wall. Notice that this technique allows generating

Table 2. Summary of the variations of the input time series that can be used to test the robustness of the systems' capabilities

Types of noise

Source of noise	Description
Signal noise	White noise added to signal
Step noise	White noise added to size of automaton's translation movement

Systematic random variations of signal

Element varied	Description
Phase of signal	The phase of the signal is set randomly in different wall/object presentations
Period of signal	The step size of the automaton (and hence its speed) is set randomly in different wall/object presentations so as to have compressions/expansions of the signal
Signal level	The distance of the automaton from the wall/object is set randomly in different wall/object presentations
Signal range	The distance of the automaton from the wall/object and the size of the object are multiplied by a random parameter in different wall/object presentations

Table 3. Summary of the techniques that can be used to analyze the internal dynamics of systems

Technique	Aspects investigated
Cross-correlograms	Time correlations between different variables
Phase space analysis	Identification of different types of attractors
Ad-hoc input time series	Systems' reaction to different signal features
Hinton plot	Roles of different connection weights
Targeted lesions	Roles of different connection weights

signals that correspond to rather complex "objects", as shown by the right graph of Figure 1.

The second technique of generating the input time series involves an automaton that follows a circular path around an object, and perceives its profile with a proximity sensor. An example of this is given in Figure 2 where the automaton moves around a cross-shaped object following a circle with radius R while its sensor detects the distance to the object at each step. Notice that this technique allows generating signals that correspond to rather complex "walls", as shown in Figure 2.

From an implementation point of view, the general idea behind the way of generating the input signal with the two techniques is that the walls or objects are composed by a set of segments. In this way, the signals are easily obtained on the basis of the computation of the distance between the automaton's central

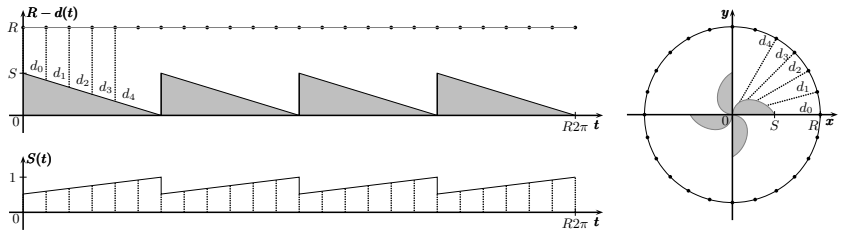


Fig. 1. Top left: profile of the wall perceived by the automaton, and positions from which the automaton senses it (dots along the straight trajectory followed by the automaton, marked by the light gray line). Right: equivalent setting showing an object, and the automaton circulating around it, that generates the same sensor reading as the wall setting. The curve of the object has been obtained with a very dense sampling of the wall. Bottom left: automaton's sensor reading, $S(t)$ normalized in $[0, 1]$, equal for the two settings.

point and the (closest) intersection between the sensor's ray and the segments composing the object. The two techniques give the researcher the possibility of generating a great number of signals having different time regularities, as shown in the examples reported in Section 4 and indicated in Table 1, with computationally rather easy effort. Moreover, the analysis of results can be aided by the fact that the first technique allows plotting the signal produced by walls in terms of objects, performing a dense sampling of the latter (see Figure 1, right). Vice versa, the second technique allows plotting object signals in terms of walls (see Figure 2, top right).

2.2 Metrics for Measuring Prediction and Categorization Capabilities

The testbed allows testing models' capabilities of prediction and/or categorization. The prediction task consists of producing an output at time t that matches the value of the signal that will be perceived at time $t+1$. This capability requires the integration of past signal values to capture the regularities of the signal in time.

Regarding the metrics that can be used to measure prediction performance, an immediate way of measuring the model's capacity to predict is to compute the quadratic error E of prediction with respect to the next input, averaged over the duration of the test:

$$E = \sum_{t=1}^{T-1} \left(P(t) - S(t+1)\right)^2 \tag{1}$$

where T is the duration of the test, $P(t)$ is the value of the prediction, and $S(t)$ is the value of the input unit activated by the automaton's sensor.

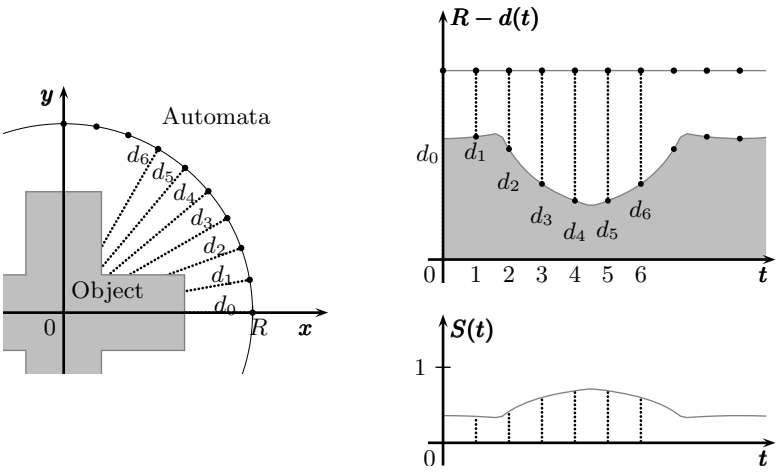


Fig. 2. Left: time series of the distances $d_0, d_1, d_2, \ldots$ (dotted lines) detected by the testbed's automaton (represented by the dots on the circumference) while moving around a cross-shaped object. Top right: the corresponding wall-setup; the curve of the wall has been obtained with a very dense sampling of the object. Bottom right: intensity of the normalized signal $S(t)$ detected by the automaton at each time step, equal for both setups.

Here "categorization" is referred to the models' capacity of distinguishing between several different signals. To this purpose, the models should have some output units whose activation can be trained in a supervised fashion and can be interpreted as the category assigned to the perceived signal. For example, in the tasks considered in Section 4, the automaton experiences two/three different objects/signals and has to categorize them with two/three units using a local code: the unit with the highest activation corresponds to the chosen category.

Performance of categorization can be measured as the percentage of time steps in which the categorization of the current perceived object produced by the automaton is correct. As the signals last more than one step, it has to be decided when to detect the categorization answer of the network. For example, in the examples shown in Section 4, this detection is done at the end of the pattern presentation. Nevertheless, dependent to the task, other points in time may be preferred.

Another way to measure the accuracy of the systems' prediction capabilities is to use prediction at time t as a new self-generated input pattern for time $t + 1$, to use the latter to produce a new prediction at $t + 1$ and again use it as input pattern for $t + 2$, and so on in a cyclical way. By monitoring how the patterns so generated diverge from those of the original input time series (i.e. by measuring the mismatch of the self-generated and real time series at a given time step in the future), it is possible to evaluate the goodness of the systems' prediction capabilities (cf. [22], where the quality of the prediction capability

was measured not in terms of this mismatch, which can be used only when the input time series does not depend on the system's actions, but in terms of the capacity of the self-generated time series to produce accurate behavior).

2.3 Test of Robustness vs. Noise and Systematic Transformations of the Input Signal

Another important feature of the testbed is the possibility of determining the robustness of the model's prediction and categorization capabilities mentioned in Section 2.2. In particular, the testbed allows the evaluation of performance changes when the input time series is modified on the basis of step by step noise or on the basis of systematic random variations applied to the signal.

There are two types of step-by-step noise that can be applied to the simulated automaton (see examples in Figure 3):

1. *Translation noise.* This noise affects the size of translation of the automaton along the circular or linear trajectory it follows. This noise is set as a percent p of the automaton step size s: the noise is obtained adding a random number uniformly distributed over $[-ps, ps]$ to each step of the automaton. Notice that this noise can be cumulative, so that a certain long distance might be covered by the automaton with a different number of steps: this effect is particularly important as it can produce an overall random compression or expansion of the signal duration.
2. *Sensor noise.* This noise affects the automaton's sensor readings. The size of this noise is set as a percent of the original signal, similar to the translation noise.

The testbed allows setting four possible types of random systematic transformations to affect the signal in a biased way. We have seen in Section 2.1 that the input time series can be represented as a wave signal represented in an $x - y$ plot where the x-axis corresponds to time whereas the y-axis corresponds to the

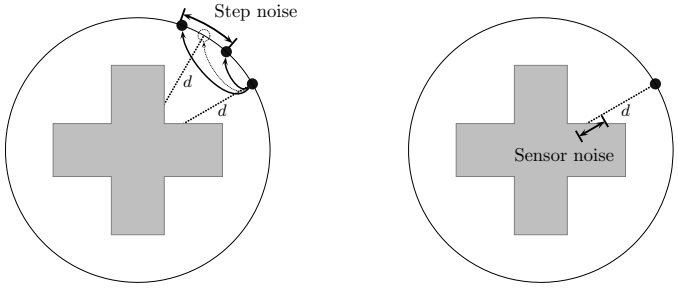


Fig. 3. Left: first type of noise affecting the step-size of the automaton, and hence the time regularity of the input time series. Right: second type of noise affecting the automaton's sensor reading.

signal level. The four random systematic transformations correspond to *linear transformations* of the wave signal with respect to the two axes. As we show below, these transformations also have an interpretation in terms of specific manipulations of the relation existing between the automaton and the object (or wall) it perceives. Referring to the plot that represents the wave signal, we can formally define the transformation as follows, which are now illustrated in detail.

$$x \longmapsto a_x x + b_x \tag{2}$$

$$y \longmapsto a_y y + b_y \tag{3}$$

where a_x, b_x, a_y, and b_y are numerical coefficients. Each of these coefficients, when different from zero, causes one of the following four transformations:

1. *Compression/expansion of the signal time duration.* The coefficient a_x of the mapping in (2) sets the duration of the signal that corresponds to the speed of the automaton. In the testbed this parameter is set in terms of the number of steps that the automaton takes to sense the whole input time series (i.e. to complete a whole circle around the object or to complete one wall profile). The left graph of Figure 4 shows the effects of this transformation with respect to the original wave signal generated by the cross-shaped object reported in Figure 3.
2. *Phase of the signal.* The coefficient b_x of the mapping in (2) sets the start (phase or shift) of the signal and corresponds to initial position of the automaton with respect to the object or the wall. The right graph of Figure 4 shows the effects of this transformation.

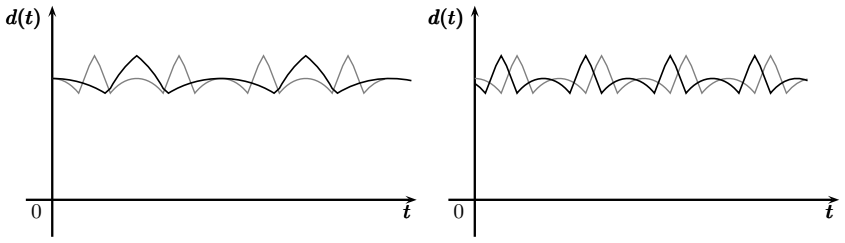


Fig. 4. Examples of random systematic transformations affecting the time variable of the input signal: $x \longmapsto \frac{1}{2}x$ (left), and $x \longmapsto x + 3$ (right)

3. *Compression/expansion of the signal level.* The coefficient a_y of the mapping in (3) sets the expansion/compression on the y-axis of the signal level. The left graph of Figure 5 shows the effects of this transformation with respect to the original signal generated by the cross-shaped object of Figure 3.
4. *Absolute signal level.* The coefficient b_y of the mapping in (3) sets the absolute position on the y-axis of the signal level. The right graph of Figure 5 shows the effects of this transformation, which corresponds to a variation of the distance of the automaton from the center of the object.

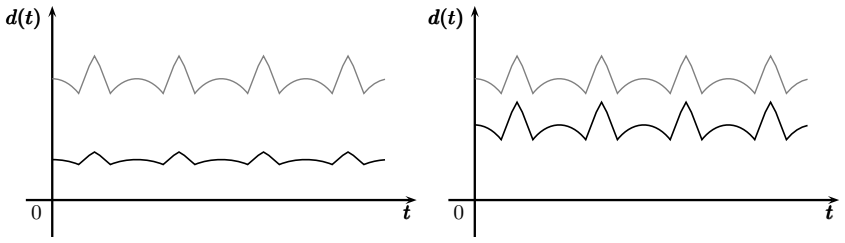


Fig. 5. Linear transformations in the space variable (distance): $y \longmapsto \frac{1}{3}y$ on the left, and $y \longmapsto y - 0.5$ on the right

Notice that the third transformation implies a compression/expansion of both the object's size and the distance of the automaton from it, which is a rather uncommon situation in real experiments (e.g. with real robots). Nevertheless, notice that if it is combined with the fourth transformation it corresponds to a variation of the size of the object.

2.4 Analysis Techniques

This section proposes a number of techniques that can be used to analyze the functioning of the recurrent neural-network models tested on the testbed. The analysis tools described here are particularly important as the functioning and internal representations autonomously developed by dynamical neural networks are particularly difficult to be understood. The analysis tools are now presented and the reader is referred to Section 4 for some examples of them.

1. *Cross-correlograms.* One way to understand the functioning of the tested models is to study the *time correlations* existing between some variables of the models, such as the activation of hidden and output units, and between such variables and the input time series. Cross-correlograms and other statistical techniques directed to detect and represent correlations between time series can be used for this purpose (see Figure 11 and Figure 12 for some examples).
2. *Phase space analysis.* Given the dynamical nature of the problems tackled, and of the neural networks tested, the analysis of the *system's activity trajectory within the state space of selected variables* might shed light on the mechanisms that allow the learner to solve tasks. In particular, the identification of limit cycles, fixed point attractors, and chaotic attractors might allow understanding the properties of the solution and the properties emerging from the coupling between the dynamical process occurring within the neural controller and the dynamical process relative to the automaton/environmental interactions (see Figure 20 for an example).
3. *Special input time series.* Particular aspects of the behavior and functioning of the models might be analyzed by studying how they react to *special input*

time series. These special input time series might be directly obtained by simplifying the ones used during training so as to isolate few features of interest (see Figure 14 and Figure 15 vs. Figure 10 for some examples).

4. *Hinton plot.* The study of the models' *connection weights* is of crucial importance because, after training, the models' performances depend on them (and the architecture). In this regard, a key tool to understand the role played by different weights is the *Hinton plot*, which allows representing the weights' intensities and signs in a visually comprehensive graph (see Figure 13 for an example).
5. *Targeted lesions.* The role of the models' components, either connection weights or units, might be understood by *lesioning* them and by observing how the performance accuracy of the models is disrupted. This can be done by setting connection weights to zero or by clumping the activation of the neurons of interest to zero (see Section 4.2 for some examples).

This list of analysis tools is of course non exhaustive. The use of these analysis tools, and how they can be used in a complementary fashion, is exemplified in Section 4.

3 Neural Networks for Integrating Information in Time

As mentioned in the introduction, dynamical neural networks are one of the most interesting classes of models that are capable of integrating information in time. Their integration capabilities are also supported by a wide literature that analyzes their relation with statistical algorithms for time series analysis (see [3], [4], [5], [8], and [19]). This section illustrates few important examples of these models. They have been selected for various reasons:

(1) they are widely used within the neural-network community;
(2) the testbed presented here has been originally developed as a tool to test and compare them within the research thread mentioned in the introduction;
(3) they allow the reader to envisage the models' properties that might be analyzed with the tools presented here;
(4) in the future the testbed will be used to systematically compare them.

3.1 Elman Neural Networks

The first neural architecture considered here is the *Elman neural network* (see [2], [7] and [16]). This is a feed-forward network with three layers: an input layer, a hidden layer, and an output layer (Figure 6). The core of this type of network is the presence of recurrences in the hidden layer that implies that, at each time step, the activations of hidden units depend on the activations of the same units at the previous time step.

This architecture allows the network to store information from the past so that the network is capable of detecting periodicity and regularities of the input patterns in time. In particular, if we let N_I, N_H, and N_O denote the number

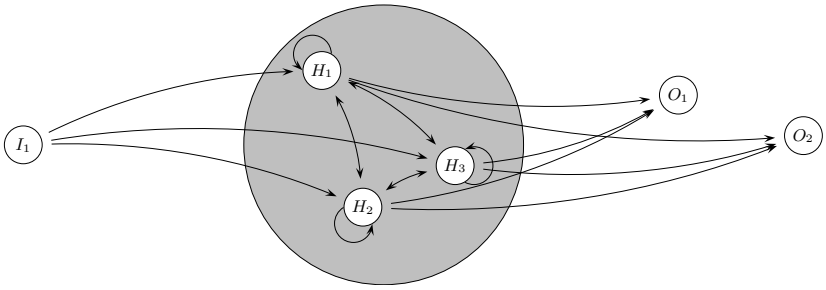


Fig. 6. Architecture of an Elman network with 1 input unit, 3 hidden units and 2 output units

of units in the input, hidden and output layer, respectively, then the input to the N_H hidden units will be formed not only by the N_I units activated by the current input, but also by further N_H units encoding the activation of the hidden units at the previous time step (these play the role of a *memory buffer*).

The activation function of the hidden and the output layer units is the logistic function:

$$\Phi_\sigma(x) = \frac{1}{1 + \mathrm{e}^{\sigma x}} \ , \tag{4}$$

where σ is a "temperature" parameter (set to 1 in the experiments herein). The learning rule used is the error back-propagation algorithm (see [17] or [20]).

3.2 Echo State Neural Networks

The second neural architectures considered, called *echo state neural networks* (Figure 7), are only briefly reviewed here (see [11] and [13] for a detailed description and literature review). The important aspect of these networks is the presence of a hidden layer, called *dynamical reservoir*, formed by linear or sigmoid units which have hard-wired recurrent connections (connections in solid black in Figure 7). These connections, that form a W matrix, are initially set randomly, and then are normalized with the highest eigenvalue of the matrix so that W has a spectral radius slightly smaller than 1. This setting of the weights implies that the hidden neurons do not produce a chaotic behavior, do not explode or do not saturate on maximum or minimum values. Moreover, it implies that the variation in the activation state of the hidden neurons produced by transient inputs tends to slowly decay after the end of the stimulation.

If the set of hidden units is large enough, the dynamical reservoir is capable of producing a large number of dynamics. The units of the reservoir are connected to the output units in a simple linear fashion (the output units also feedback to the reservoir units with random connections – hence the term "echo" in the name: the signals vehiculated by these connections contribute to modulate the reservoir's dynamics). During training, the weights that connect the reservoir units with the output units are updated with a supervised algorithm to reproduce

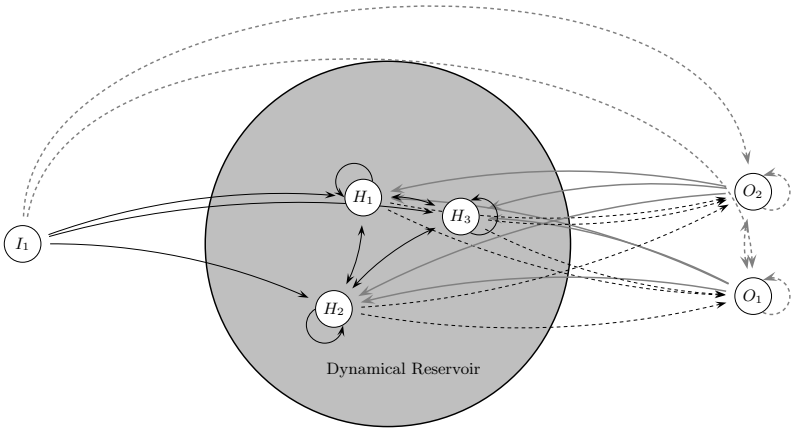


Fig. 7. Architecture of an Echo State Network with 1 input unit, 3 hidden units, and 2 output units

a target output signal (e.g., a periodic signal). This training leads the hidden-output weights to "select" few relevant dynamics from the internal reservoir among the possible ones, and allows the output units to learn to reproduce, in principle, any desired output signal having some correlation with the input signal.

3.3 Leaky Integrator Neural Networks

The third type of neural network considered here is formed by *leaky integrator neurons*. The core property of these neurons is that their activation potential depends not only on the input from other internal and external neurons, but also on own previous activation potential (see [1], [18] and [21]). Let $u_i(t)$ denote the i-th unit's potential at time t, $I_i(t)$ its external input, $h_i(t)$ its resting level, Φ_σ is the sigmoid activation function in (4), $u_j(t)$ the activation potential of another j-th unit of the network, and w_{ij} the weight from the unit j to the unit i; the dynamics of $u_i(t)$ is governed by the following dynamic equation:

$$\tau \dot{u}_i(t) = -u_i(t) + I_i(t) + h_i + \sum_j w_{ij}\, \Phi_\sigma\left(u_j(t)\right) \ . \tag{5}$$

Equation (5) implies that in absence of external inputs I_i, and with zero connection weights w_{ij}, the neuron exponentially relaxes to the resting state h_i with a rate equal to $-\frac{1}{\tau}$ (hence the name "leaky"). If Δt denotes the integration time step, then (5) has a discrete version with the following form (that can also be used to numerically integrate equation (5) in the simulations):

$$u_i(t+\Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) u_i(t) + \frac{\Delta t}{\tau} \left(I_i(t) + h_i + \sum_j w_{ij}\, \Phi_\sigma \left(u_j(t)\right) \right) \quad . \tag{6}$$

This form highlights that the activation potential of leaky neurons approaches the sum of the resting level, external input, and input from other neurons, on the basis of a "partial adjustment mechanism". This implies that leaky neurons have a ready available "internal" memory of the past that can be exploited by the whole neural network to integrate information in time.

3.4 Long Short-Term Memory Neural Networks

The last types of neural networks reviewed here are the *Long Short-Term Memory Networks* (see [9], [10] and [12]). They have been introduced to extend the memory capacity of standard recurrent neural networks, in particular, they have been shown to efficiently solve many tasks involving integration of information in time that are unlearnable for other neural networks (e.g. the recognition of temporally very long extended patterns in noisy input sequences, the recognition of the temporal order of widely separated events in noisy input streams, or the stable generation of precisely timed rhythms).

The key feature of these neural networks resides in the special type of neurons that form them, which are characterized by a self-recurrent connection and gates that exert multiplicative effects on the input and output channels (Figure 8). The functioning of one neuron of this type can be described as follows:

$$y_i(t) = \Phi_\sigma \left(\sum_j w_{ij}^{go} u_j(t) \right) \Phi_\sigma \left(u_i(t)\right) \ , \tag{7}$$

$$u_i(t) = u_i(t-1) + \Phi_\sigma \left(\sum_j w_{ij}^{gi} u_j(t) \right) \Phi_\sigma \left(\sum_j w_{ij} u_j(t) \right) \ , \tag{8}$$

where $y_i(t)$ and $u_i(t)$ are the i-th unit's activation and action potential at time t, respectively, w_{ij} is the weight from the unit j to the unit i, w_{ij}^{gi} and w_{ij}^{go} are the weights from the units j to the input and output gates, respectively, and Φ_σ is the sigmoid activation function of Equation (4).

These features allow neural networks formed by several of these special neurons to produce highly complex dynamics. The networks so formed can be trained on the basis of supervised learning algorithms.

4 Examples of Applications

This section illustrates the potential of the testbed by testing an Elman neural network with two specific tasks. In the first task, the automaton perceives two walls with different profiles, while following a linear trajectory, while in the second task the automaton senses three different objects while following a circular

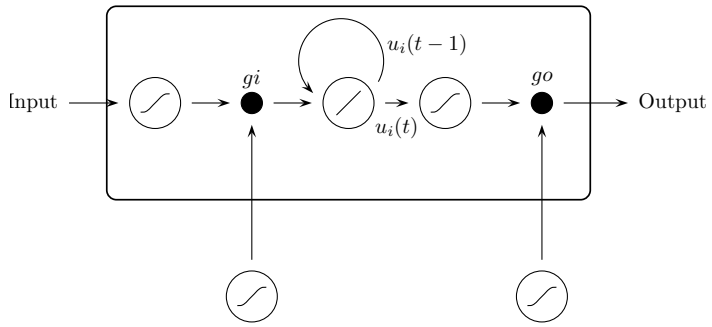


Fig. 8. Structure of a single neuron of a long short-term memory neural network

trajectory. In both cases the automaton's goal is to categorize the perceived signal and to predict, at each step, the signal level at the next step on the basis of the previously experienced signal levels. Note that the experiments reported here also represent the initial work of a research agenda directed to understand the internal dynamical mechanisms exploited by recurrent neural networks to capture regularities in time.

4.1 Wall Task: Experimental Setup

In this task the automaton moves along a straight trajectory along a wall which can have one of two different profiles, shown in Figure 9. The various settings of the experimental setup can be summarized as follows:

1. *Wall profiles.* The two possible wall profiles (Figure 9) had "hollows" with same depth (this caused the normalized automaton's sensor reading return 1, so this portion of the two walls was ambiguous for the automaton), and "humps" with different heights (these caused a normalized sensor reading equal to 0.36 and 0.68 respectively for the first and second profile).
2. *Model.* The tested model was an Elman neural network with $N_I = 1$ input units, $N_H = 2$ hidden units, and $N_O = 3$ output units. The input unit was activated by the sensor reading normalized in $[0, 1]$. The first output unit was devoted to predict the next input pattern while the other two output units were devoted to encode the categories of the two wall profiles. Such categories were locally encoded as $\{1, 0\}$ and $\{0, 1\}$ respectively for the two profiles.
3. *Training.* During training, the walls were repeatedly presented one by one to the automaton. The wall used in each presentation was randomly chosen, and at each presentation the automaton performed a whole circle around it. Training lasted $1,000,000$ presentations, and used a $\lambda = 0.005$ learning rate. For each time step, the teaching input was formed by the next input pattern (i.e. the value of the signal at time $t + 1$) and by the binary value that encoded the category of the current wall.

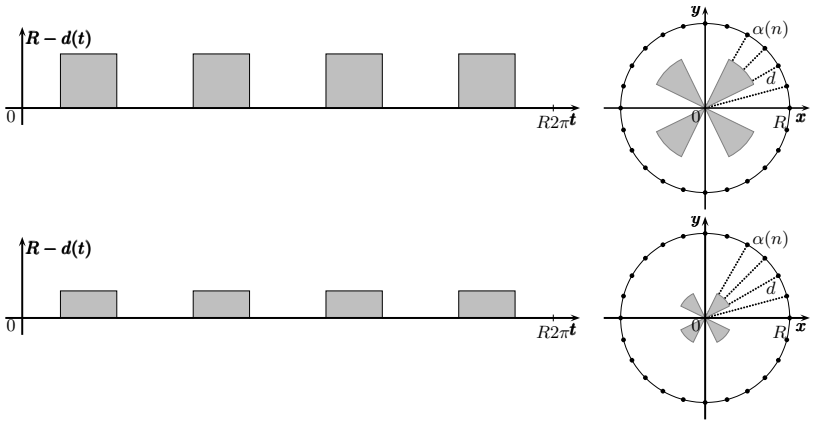


Fig. 9. The two wall profiles used in the wall task (left) and the two equivalent objects in the corresponding object task (right)

4. *Steps (Angular Speed).* The automaton covered a single lap around the patter in a number of steps denoted by $\#Steps$. In the experiment this parameter was randomly assigned one of the following values in every object presentations (and kept constant during each presentation):

$$\#Steps \in \{16, 24, 32, \ldots, 128\} \ , \tag{9}$$

   The angle of the trajectory covered by one step of the automaton, that is, its angular speed, depended on the total number of steps of a lap, and was equal to $2\pi/\#Steps$.
5. *Starting Point.* This was the angle of the circular trajectory where the automaton started to perceive the wall. Let $\alpha(n)$ denote the angle at the n-th step. For any $n = 0, \ldots, \#Steps$:

$$\alpha(n) \in \left\{0, \frac{2\pi}{\#Steps}, \frac{4\pi}{\#Steps}, \ldots, (\#Steps - 1)\frac{2\pi}{\#Steps}\right\} \ . \tag{10}$$

   The starting point $\alpha(0)$ was set randomly at each presentation of the input time series.
6. *Compression/Expansion of the Signal Level (Height of Profiles).* The size of the maximum height of the two walls was kept constant: 1.6 and 0.8 for profiles 1 and 2, respectively.
7. *Absolute Signal Level (Radius/Distance from Walls).* The parameter of the distance from the walls hollows, denoted by R, was set to 2.5.
8. *Noise.* The two sources of noise illustrated in Section 2.3 were both set to 5%.

4.2 Wall Task: Results

The training of the system was rather successful. At the end of training, the neural network shows a rather good categorization ability. More precisely, the

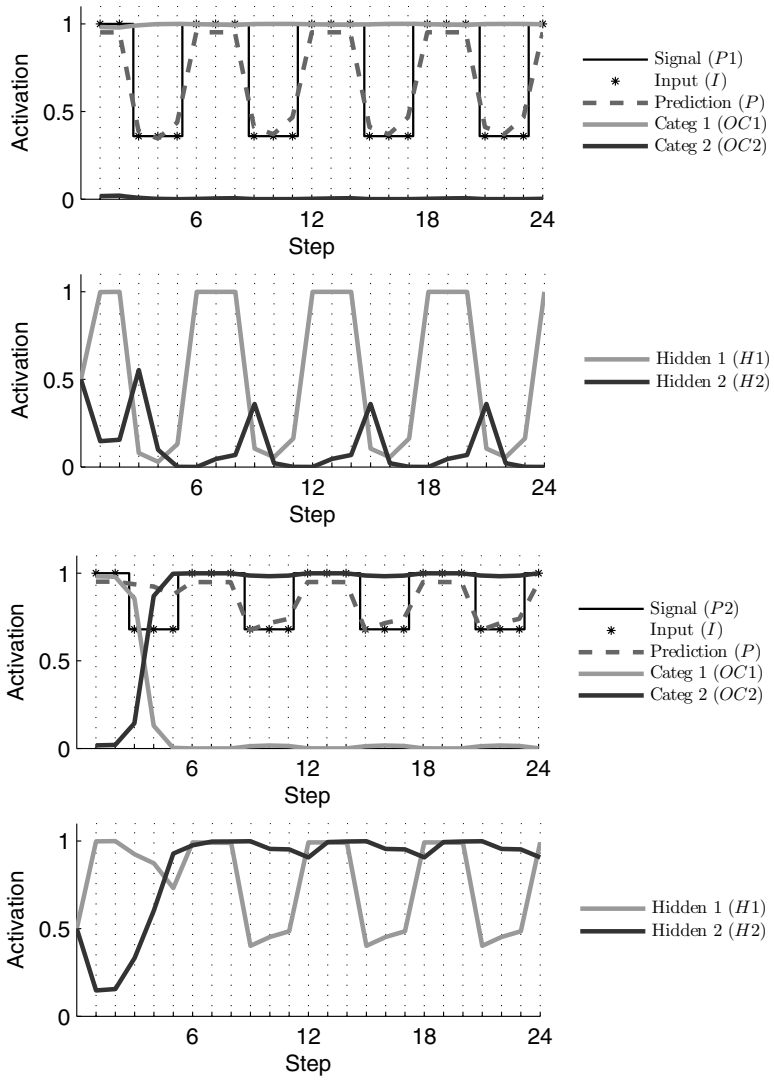
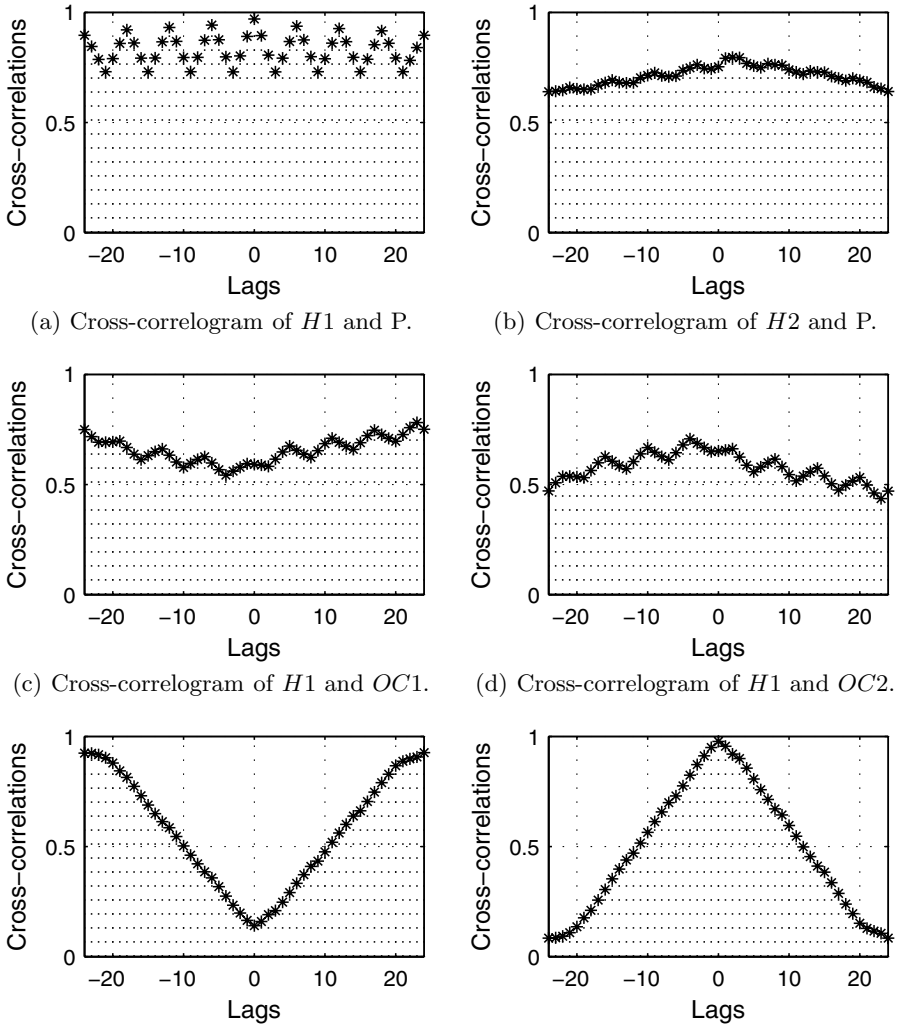


Fig. 10. Activations of the output units (first and third graph from top) and hidden units (second and fourth graph from top) when the automaton perceives the signal from the wall profile 1 and 2 (respectively first/second and /third/foruth graphs from top). Stars in the first and third graph indicate the actual sensor's readings (noise has been switched off to ease the analysis of results), whereas the continuous black lines show the wall's profiles obtained with a very dense sensor reading.

network produces the right categorization output after few steps, and after that, keeps producing the same categorization output even during the phase in which the signal is ambiguous, that is, when scanning a hollow with a value equal to 1



(a) Cross-correlogram of $H1$ and P.

(b) Cross-correlogram of $H2$ and P.

(c) Cross-correlogram of $H1$ and $OC1$.

(d) Cross-correlogram of $H1$ and $OC2$.

(e) Cross-correlogram of $H2$ and $OC1$.

(f) Cross-correlogram of $H2$ and $OC2$.

Fig. 11. Cross-correlograms between different variables of the models (see text) when the model perceives a repeated sequence of 1000 $P1/P2$ wall profiles, in an alternate fashion, with $\alpha(0) = 0$

(as shown by the thin black curves in Figure 10). With respect to the prediction capability, however, the network simply predicts that the signal at time $t + 1$ will be identical to the signal at time t. Although this simple strategy allows the network to produce the right answer in most of the cases, it fails to predict correctly in the cases in which the value of the signal suddenly varies from time t to $t + 1$ (Figure 10). Indeed, exactly predicting this sudden change is not possible due to the noise affecting the automaton's step-size.

The capabilities of the model are robust with respect to signal's random systematic transformations of the first type indicated in (2) (recall that these transformations are related to variations of the initial position of the automaton with respect to the object and to the step size): none of them prevents the system's capabilities to emerge.

In order to understand in detail how the system performs prediction and categorization, a test was run where the system was presented for 1000 times, in an alternate way, the two wall profiles, each time with $\alpha(0) = 0$ (that is, the automaton is placed at the beginning of the input time series). The data collected in this test were used to build cross-correlograms capturing correlations within couples of time series related to various variables of the network, namely, the input value, the hidden units' activations, and the output units' activations. Let us denote with I the input unit's activation (i.e. the perceived signal), with $H1$ and $H2$ the two hidden units' activation, with OP the activation of the output unit devoted to prediction, with $OC1$ and $OC2$ the activation of the two output units devoted to categorization, with $OD1$ and $OD2$ the desired output for the two categorization units, and with $P1$ and $P2$ the two wall profiles.

The cross-correlograms between the hidden and the output units' activation, reported in Figure 11, give important indications on the role played by the hidden units in the model's responses:

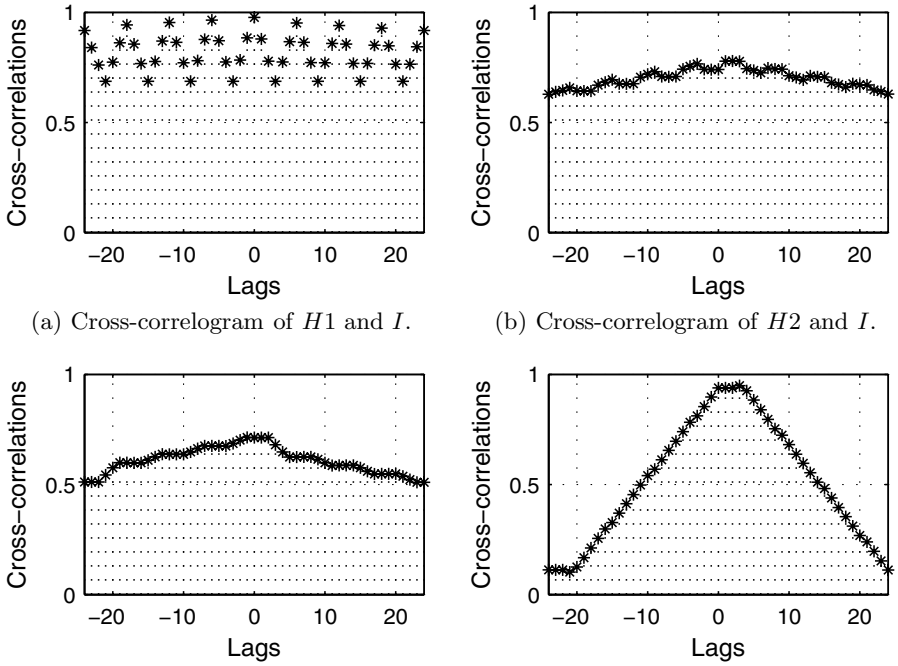
1. The comparison of cross-correlograms of Figure 11(a) and 11(b), related to the correlation between $H1/H2$ and OP, indicate that $H1$ has a strong correlation with OP whereas $H2$ has an almost null correlation with it.
2. The cross-correlograms of Figure 11(c) and 11(d) indicate that $H1$ has a very low anti-correlation with $OC1$ and a very low correlation with $OC2$.
3. The cross-correlograms of Figure 11(e) and 11(f) indicate that $H2$ has a strong anti-correlation with $OC1$ and a strong correlation with $OC2$.

Altogether, these data corroborate the suggestion that $H1$ mainly underlies the model's prediction capability, whereas $H2$ mainly underlies its categorization capability, and that a high and low activation of the latter tends to cause the model to categorize the input pattern respectively as $P2$ and $P1$.

These interpretations are further corroborated by the cross-correlograms related to the hidden units, the input unit, and the second wall category ($OD2$; the cross-correlograms with $OD1$ give similar information), reported in Figure 12, which show that:

1. The comparison of cross-correlograms of Figure 12(a) and 12(b), related to the correlation between $H1/H2$ and I, indicates that $H1$ has a strong correlation with I whereas $H2$ has an almost null correlation with it.
2. The comparison of cross-correlograms of Figure 12(c) and 12(d), related to the correlation between $H1/H2$ and $OD2$, indicate that $H1$ has no correlation with $OD2$ whereas $H2$ has a strong correlation with it.

Figure 13 reports the values of the model's weights that emerged with training. The analysis of the weights between the hidden and output units confirms the



(a) Cross-correlogram of $H1$ and I.

(b) Cross-correlogram of $H2$ and I.

(c) Cross-correlogram of $H1$ and $OD2$.

(d) Cross-correlogram of $H2$ and $OD2$.

Fig. 12. Cross-correlograms between hidden units' activation and the input signal or desired output of the second categorization output unit, when the model perceives a repeated sequence of 1000 $P1/P2$ wall profiles, in an alternate fashion, with $\alpha(0) = 0$

indications given by the cross-correlograms, and also allows formulating a more detailed explanation of the functioning of the system:

1. The weights from $H1$ and $H2$ to OP show that P depends only on $H1$, as the weight of the connection from $H1$ is positive (positive correlation) whereas the weight from $H2$ is close to zero (no correlation). Indeed, lesioning this weight, that is setting it to zero, has no effect on prediction performance (data not reported).
2. The high weights from $H2$ to $OC1$ and $OC2$ confirm that this hidden unit greatly contributes to determine the category of the wall profile, namely $P1$ when it is low and $P2$ when it is high. $H1$ also partially contributes to the categorization as its weights to $OC1$ and $OC2$ are different from zero (its high activation tends to cause a categorization of the input signal as $P2$).
3. The analysis of the weights between the input and the memory units on one side, and the output units on the other side, give other important indications on how the system solves the task.
4. Considering the connections to $H1$, the positive connection weight between I and $H1$ implies that $H1$ implements the prediction capabilities by "relaying"

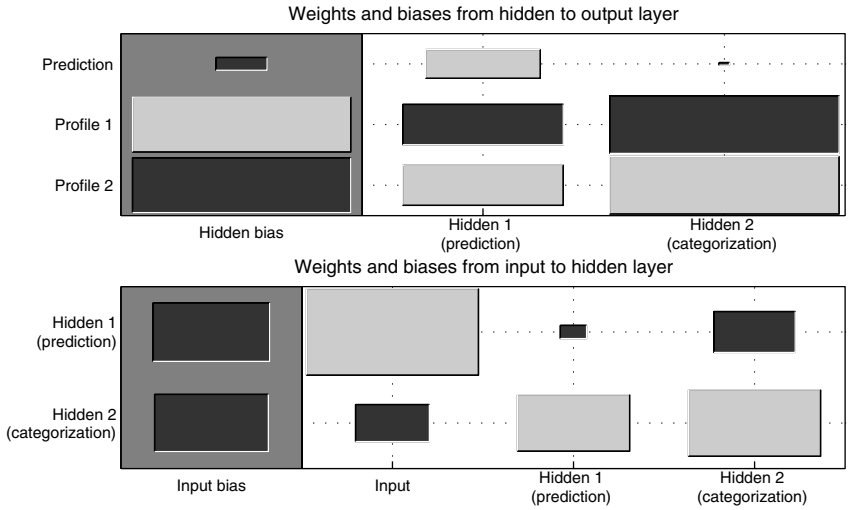


Fig. 13. Hinton plot of model's connection weights after training. The black and gray squares respectively correspond to negative and positive values of the weights, whereas their size is proportional to the weights' absolute value.

the input signal: the model tends to return a high or low prediction value respectively for high or low input signal levels. Note that this implies that the model is not capable of returning an accurate prediction when the signal suddenly changes level one step in advance, as shown in Figure 10. The weights from $H1$ to itself do not play an important role. Indeed, lesioning them does not impair performance (data not shown).

5. Considering the connections to $H2$, the positive self-connection of $H2$ indicates that it has a strong inertia. The positive connection from $H1$ (that, as we have seen, positively correlates with the input pattern and generates P) indicates that $H2$'s categorization capacity strongly depends on $H1$'s activation. Lesioning the connection between the input and $H2$ indicates that it is also important for categorization (data not reported).

A further refinement of these interpretations is furnished by two other experiments where the input signal to the system is handcrafted in order to highlight particular aspects of its internal dynamics. In particular, Figure 14 shows the dynamics of the model's hidden and output units' activation when it first perceives a signal of 1 and then of 0.68 (recall that the level of the signal has been normalized in the range $[0, 1]$), whereas Figure 15 shows the dynamics of the same variables when the system first perceives a signal of 0.36 and then of 0.68.

With respect to the prediction capability, these figures confirm that prediction capability ($H1$) relies in part on categorization ($H2$). In fact, if $H2$ gives the category $P1$, then $H1$ gives $P = 1$ with both $I = 1$ (Figure 14) and $I = 0.68$ (Figure 15: note how after the signal abruptly changes, the prediction makes a

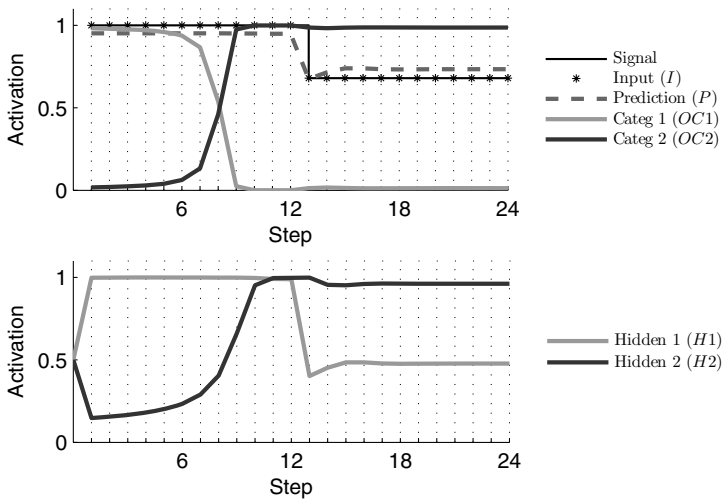


Fig. 14. Activations of the hidden and output units when the automaton first perceives a signal of 1 and then of 0.68

mistake for about six cycles because the activation of $H2$ is incorrectly categorizing the input as $P1$). On the other hand, if $H2$ gives the category $P2$, $H1$ gives $P = 1$ with $I = 1$ (Figure 14), but it gives $P = 0.68$ with $I = 0.68$ (Figure 14 and 15).

With respect to the categorization capability, it is interesting to see how the system can solve the $I = 1$ ambiguity. $H2$ moves slowly toward 1 (that implies $P2$) both when $I = 1$ (Figure 14) and when $I = 0.68$ (Figure 15), whereas it stays at 0 when $I = 0.36$ (Figure 15). This implies that $H2$ has the value of 1, (corresponding to $P2$) as a fixed-point attractor value when $I > 0.36$ or so, and 0 (corresponding to $P1$) when $I = 0.36$. For this reason, when the signal level $I = 1$ has been preceded by a signal $I = 0.36$ (corresponding to $P1$), $H2$ approaches 1 ($P2$) only slowly and so continues to give $P1$ for some time until the system perceives $I = 0.36$ again.

4.3 Three Objects Task: Experimental Setup

In this task, the automaton moves along a circular trajectory around three different "objects" and at each step detects the distance from them (see Fig. 16 and compare with [2]).

The various settings of the experimental setup can be summarized as follows:

1. *Objects.* The three objects are illustrated in Figure 17: a "square", a "thick cross" and a "thin cross". The number of objects is denoted with N_P. Notice that, given the shape of the objects, during one lap around an object the automaton experienced a signal formed by four succeeding equal waves, similarly to the wall task.

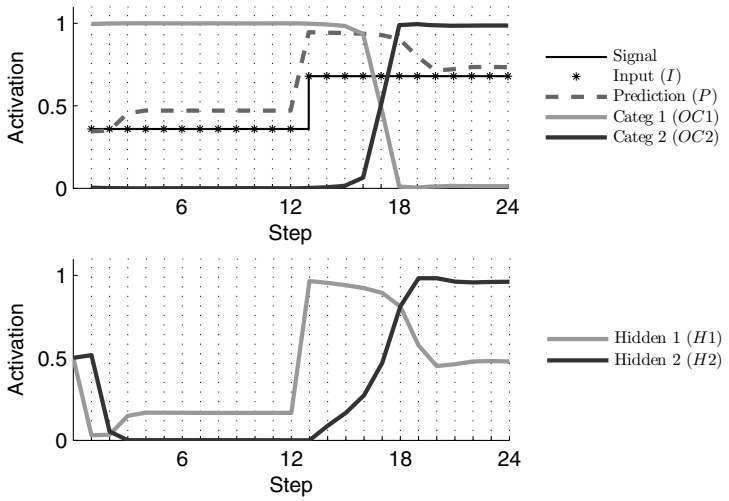


Fig. 15. Activations of the hidden and output units when the automaton first perceives a signal of 0.36 and then of 0.68

2. *Model.* The tested model was an Elman neural network with $N_I = 1$ input unit, $N_H = 3$ hidden units, and $N_O = 4$ output units. The input unit was activated by the sensor reading normalized in $[0, 1]$. The first output unit was devoted to predict the next input pattern while the remaining three output units were devoted to encode the categories of the three signal patterns. Such categories were locally encoded as $\{1, 0, 0\}$, $\{0, 1, 0\}$, and $\{0, 0, 1\}$) respectively for the three objects.
3. *Training.* Training was performed as in the wall task.
4. *Steps (Speed).* The number of steps *#Steps* the automaton took to scan an object's profile during a presentation was randomly varied from presentation to presentation as in the wall task.
5. *Starting Point.* The starting point $\alpha(0)$ was randomly varied from presentation to presentation as in the wall task.
6. *Compression/Expansion of the Signal Level (Object Size).* The size of the object is denoted by S (this represents half of the longest arm of the cross objects and half the size of the square's side) and half of the length of the shortest arm of the crosses is denoted with T. In the majority of experiments reported below, the size of the objects was set at fixed values, whereas in few other experiments it was randomly varied at each presentation (but kept fixed within it) within the range $[0.5, 1.8]$ (recall that S is related with the parameter a_y of equation 3).
7. *Absolute Signal Level (Radius).* In the majority of experiments reported below the radius R of the circular trajectory followed by the automaton around the objects was set at fixed values, whereas in some variants of the experiment it was randomly varied at each presentation (but kept fixed within it)

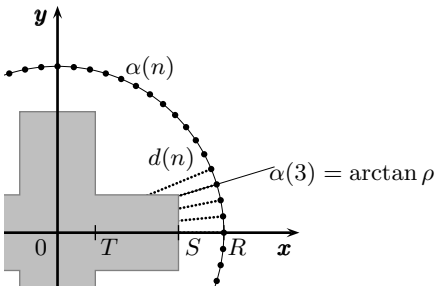


Fig. 16. Important parameters of the three object recognition task: the dots represent different positions in space occupied by the automaton from which it detects a certain distance d from the object. R is the automaton's distance from the object's center, S is half of the size of the object's longest axis, and T is half of the object's shortest axis.

in the range $[2, 3]$ (recall that R is linearly related to the parameter b_y of equation 3).

8. *Noise.* The two sources of noise illustrated in Section 2.3 were both set to 5%.

Note that, since the setting implies that $0 \leq T \leq S \leq R$, the ratio $\rho = \frac{T}{S}$ has the following restriction: $\rho \in [0, 1]$. Also note that parameter ρ uniquely identifies the three objects (see Figure 16).

4.4 Three Objects Task: Results

With the two sources of noise on and the four random systematic transformations off, the Elman network achieves a satisfying performance both for the prediction and for the categorization tasks. As far as the prediction capability is concerned, as in the wall task, the system adopts a strategy of input repetition. Nevertheless, the network had low quadratic errors when tested with various step sizes (see Figure 18): the graph shows that the error decreases as the steps grow.

A further analysis of the system's prediction capabilities was obtained by presenting the patterns to the model for some time, and then by forcing the network to use its prediction as the next self-generated input. In this experiment, the quality of the prediction signal rapidly deteriorates (constant output), hence confirming that the system directly repeats the input as prediction.

The model also shows to be robust with respect to signal's random systematic transformations of the first type indicated in (2). In particular, the model is robust with respect to variations of the initial position of the automaton with respect to the object (corresponding to $\alpha(0)$), which does not deteriorate performance (data not shown). Moreover, and surprisingly, the model has an even higher performance when trained with step sizes that vary randomly between the objects' presentations. This can be seen by comparing the model trained in two different conditions:

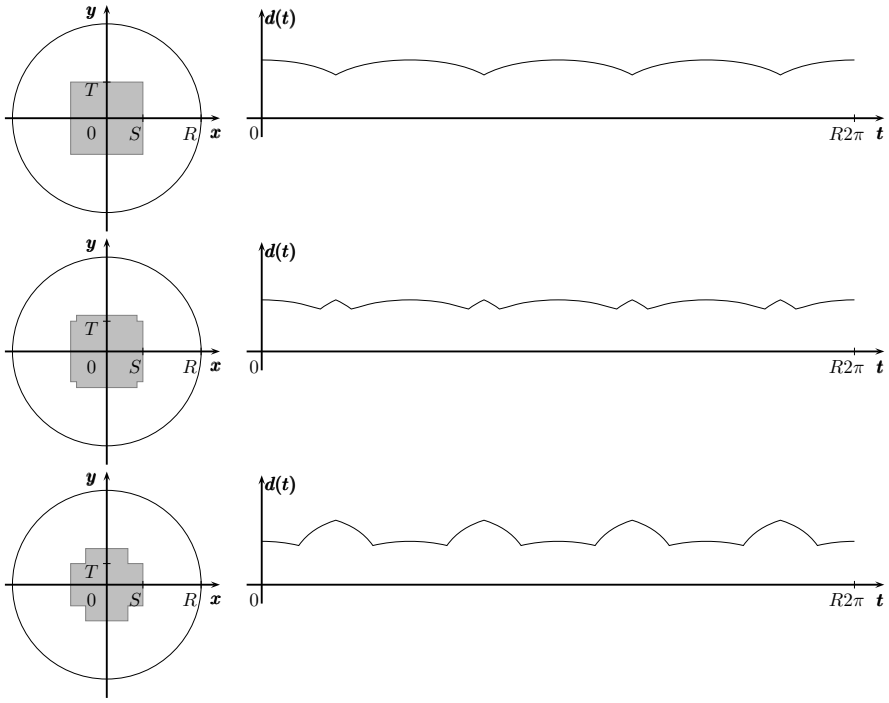


Fig. 17. Left graphs: the three crosses used in the object task, characterized by $(\rho_1, S_1) = (1, 0.95)$, $(\rho_2, S_2) = (0.70, 1.13)$, and $(\rho_3, S_3) = (0.41, 1.35)$: these values were set to similar maximum and minimum values of the sensor's reading (in few experiments was randomly varied). The radius R was set to 2.5. Right graphs: the sensor readings caused by the three objects.

1. In each presentation $\#Steps$ is randomly set within the values indicated in (9), and the object approach angle $\alpha(0)$ is randomly set in the range indicated in (10).
2. The $\#Steps$ is set at the same value used in the test of performance, whereas the object approach angle $\alpha(0)$ is randomly set as in the previous condition.

The white and black bars of the histogram reported in Figure 19, which refer to the two training conditions, respectively, indicate that the performance of the model is generally higher when it is trained with varying step sizes. Further experiments with one square and three different crosses corroborate this result (data not reported).

Other tests showed that the signal's random systematic transformations of the second type, indicated in (3), completely disrupt the performance of the algorithm (data not reported). This result suggests that the capacity of the model both to categorize the object and to predict the next input heavily relies on the absolute and relative levels of the signals in time.

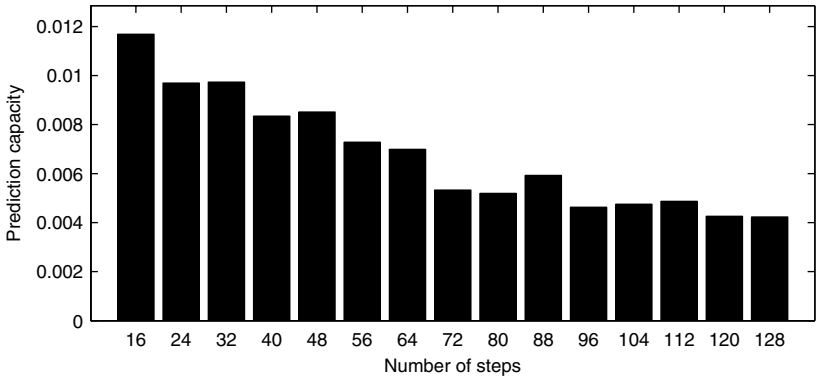


Fig. 18. Mean square error of the prediction unit with respect to *#Steps*

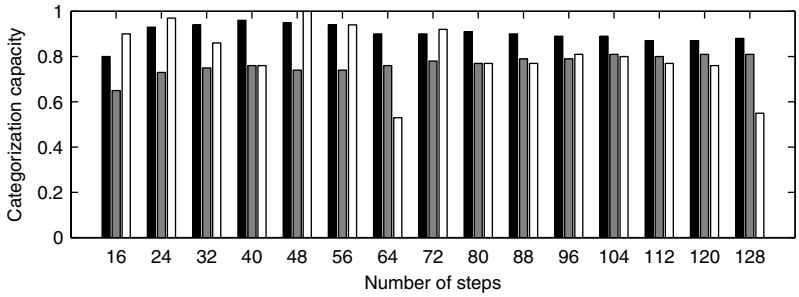


Fig. 19. Categorization performance of the model (y-axis) when it is tested with various step sizes (x-axis). The black bars refer to a model trained, with varying step size, to categorize the objects and to predict the next input. The gray bars refer to a model trained, with varying step size, only to categorize. The white bars refer to different models trained to categorize and to predict with a step size equal to the one used in the performance tests, reported on the x-axis.

Another interesting result is that training the prediction capability of the model improves the model's capacity to categorize the objects. With this respect the gray histogram bars reported in Figure 19 show that the performance of the model deteriorates if the system is not trained to predict the next input. This result indicates that training the model prediction capability likely leads the system to develop internal representations that aid the categorization capability. Further analyses of the model's internal representations should be carried out to understand this outcome in further depth. These analyzes might be aided by some of the investigation methods suggested in Section 2.4. For example, Figure 20 shows the dynamics of the model's three hidden units when the system perceives the three objects. The graphs of the figure show that the model's object categorization and prediction capability is based on three different limit-cycle attractors: the model's internal state converges to a different limit-cycle attractor

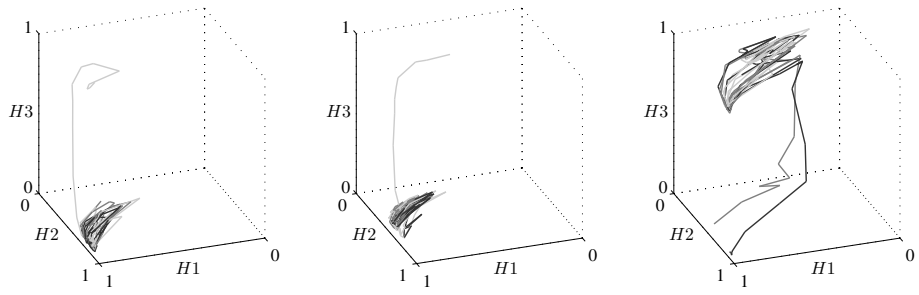


Fig. 20. Dynamics and attractors of the model's hidden units activations. Each of the three graphs represents the "history" of the three hidden units' activation when the model categorizes and predicts one of the three objects (respectively the square, the thick cross and the thin cross). Each graph reports the outcome of the experiment run in three different conditions: (a) the model categorizes the object after it has perceived the square (black line); (b) the model categorizes the object after it has perceived the thick cross (gray line); (c) the model categorizes the object after it has perceived the thin cross (light gray line).

in order to categorize the different objects. Notice that when the system starts to perceive an object after it has perceived a different object, its internal state takes some time to settle to the limit-cycle attractor of the current object as its internal memory needs to synchronize with the dynamics of the new input time series. After the state has settled to the attractor corresponding to the object, then it follows a cyclic trajectory within it in order to predict the next input pattern.

5 Conclusions and Future Work

This paper presented a testbed that can be used to evaluate the capabilities of recurrent neural networks (and similar models) of integrating information in time, in particular the capabilities of categorizing different signals, of predicting future signals on the basis of past ones, and of doing so in the face of noise and systematic variations of the input signal. The paper also illustrated the potentialities of the testbed by exemplifying its functioning with two tests involving simple recurrent Elman networks engaged in solving two different prediction and categorization tasks.

The added value of the paper is manyfold. First, it highlights the need of building standard testbeds, metrics, and analysis tools to compare, and build taxonomies of, the increasing number of models proposed within the literature of the ABiALS community. Second, it presents a specific testbed that allows testing models' capabilities of categorization and anticipation. Third, it shows the potential utility of developing and using testbeds by showing some results obtained by applying the testbed proposed here to the study of the functioning

of Elman neural networks. These applications showed that, even if a detailed understanding of the functioning of recurrent neural networks is very difficult, the dynamical principles that might underlie their capacity of integrating information in time are particularly interesting and make it worth designing and implementing testbeds and analysis tools, as those proposed here.

Future developments of this research will follow two main directions. On the one hand, it will continue to carry out systematic studies, in line with the preliminary experiments presented in Section 4, to understand the exact mechanisms that are developed by recurrent networks to integrate information in time, such as the formation of cyclic or fixed point attractors, units with progressive increases or decreases of activation, hierarchical abstract representations, etc. On the other hand, it will use the testbed to compare the capacities of the four neural networks described in Section 3 to capture different time regularities. This comparison could be important to highlight which particular features of temporal signals can be best integrated in time by the different models, and hence which types of tasks are more suitable for them.

References

1. Amari, S.I.: Dynamics of pattern formation in lateral-inhibition type neural fields. Biological Cybernetics 27, 77–87 (1977)
2. Cecconi, F., Campenní, M.: Recurrent and concurrent neural networks for objects recognition. In: Deved, V. (ed.): Proceedings of the International Conference on Artificial Intelligence and Applications (IASTED 2006), Innsbruck, Austria, pp. 216–221 IASTED/ACTA Press (2006)
3. Chakraborty, K., Mehrotra, K., Mohan, C.K., Ranka, S.: Forecasting the behavior of multivariate time series using neural networks. Neural Networks 5, 961–970 (1992)
4. Chappelier, J.C., Grumbach, A.: Time in neural networks. ACM SIGART Bulletin 5, 3–11 (1994)
5. Dorffner, G.: Neural networks for time series processing. Neural Network World 6, 447–468 (1996)
6. Doya, K.: Recurrent networks: learning algorithms. In: Arbib, M.A. (ed.) The Handbook of Brain Theory and Neural Networks, Second edn. pp. 955–960. The MIT Press, Cambridge, MA, USA (2003)
7. Elman, J.L.: Finding structure in time. Cognitive Science 14, 179–211 (1990)
8. Hellström, T., Holmström, K.: Predicting the stock market. Research and Reports Opuscula ISRN HEV-BIB-OP–26-SE, Department of Mathematics and Physics, Mälardalen University, Västerås, Sweden (1998)
9. Hochreiter, S., Schmidhuber, J.: Bridging long time lags by weight guessing and "Long Short-Term Memory". In: Silva, F.L., Principe, J.C., Almeida, L.B. (eds.) Spatiotemporal models in biological and artificial systems. Frontiers in Artificial Intelligence and Applications, vol. 37, pp. 65–72. IOS Press, Amsterdam (1996)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9, 1735–1780 (1997)
11. Jaeger, H.: Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the "echo state network". Gesellschaft für Mathematik und Datenverarbeitung Report 159, German National Research Center for Information Technology (2002)

12. Klapper-Rybicka, M., Schraudolph, N.N., Schmidhuber, J.: Unsupervised learning in LSTM recurrent neural networks. In: Dorffner, G., Bischof, H., Hornik, K. (eds.) ICANN 2001. LNCS, vol. 2130, pp. 684–691. Springer Verlag, Heidelberg (2001)
13. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14, 2531–2560 (2002)
14. Mitchinson, B., Pearson, M., Melhuish, C., Prescott, T.J.: A model of sensorimotor coordination in the rat whisker system. In: Nolfi, S., Baldassarre, G., Calabretta, R., Hallam, J.C.T., Marocco, D., Meyer, J.-A., Miglino, O., Parisi, D. (eds.) From animals to animals SAB 2006. Number 4095 in Lecture Notes in Artificial Intelligence, pp. 77–88. Springer-Verlag, Heidelberg (2006)
15. Nolfi, S., Marocco, D.: Evolving robots able to integrate sensory-motor information over time. Theory in Biosciences 120, 287–310 (2001)
16. Nolfi, S., Tani, J.: Extracting regularities in space and time through a cascade of prediction networks: The case of a mobile robot navigating in a structured environment. Connection Science 11, 129–152 (1999)
17. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323, 533–536 (1986)
18. Schöner, G., Kelso, J.A.S.: Dynamic pattern generation in behavioral and neural systems. Science 239, 1513–1520 (1988)
19. Ulbricht, C., Dorffner, G., Canu, S., Guillemyn, D., Marijuán, G., Olarte, J., Rodríguez, C., Martín, I.: Mechanisms for handling sequences with neural networks. In: Dagli, C.H. (ed.): Intelligent Engineering Systems through Artificial Neural Networks (ANNIE 1992) New York, NY, USA, vol. 2, pp. 273–278 ASME Press (1992)
20. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation 1, 270–280 (1989)
21. Wilson, H.R., Cowan, J.D.: Excitatory and inhibitory interactions in localized populations of model neurons. Biophysical Journal 12, 1–24 (1972)
22. Ziemke, T., Jirenhedb, D.A., Hesslow, G.: Internal simulation of perception: a minimal neuro-robotic model. Neurocomputing 68, 85–104 (2005)