Reservoir Computing for Sensory Prediction and Classification in Adaptive Agents

Cornelius Weber[1], Kazuhiro Masui[2], Norbert Michael Mayer[2], Jochen Triesch[1], and Minoru Asada[2]

[1] Frankfurt Institute for Advanced Studies, Johann Wolfgang Goethe University, Frankfurt am Main, Germany

[2] JST ERATO Asada Project, Dept. of Adaptive Machine Systems, Graduate School of Engineering, Osaka University, Japan

Abstract. Artificial neural networks are an in silico laboratory for studying the dynamics of the brain. In recurrent networks, the units' activations are recurrently fed back into the network. Thereby complex network dynamics emerge that extend over longer time scales than the individual units' activation time constants. The recurrent echo-state networks with their fixed connection weights acquire an internal representation that uniquely depends on the input history, but not on the initial state of the network. We present echo-state networks as models of sensory systems and sketch two examples of their usage in learning agents. The first example is gesture classification from moving camera images, and the second is a conceptual account of timing. Furthermore, we review a recent idea of self-prediction augmenting an echo-state network. The weights self-predicting the internal state filter out external noise, and improve the network performance significantly. Together, this chapter presents exciting new developments in the field of reservoir computing.

1 Introduction

Artificial neural networks is the branch of machine learning that most directly takes inspiration from the brain. Since the brain is far too complicated and too large to understand and to simulate in detail, existing artificial neural networks represent simplified models of isolated aspects of brain function. Systems models cannot integrate all the diversity of neuronal cell types and their variety in functionality. Rather it is of interest, what is the computational goal of a neuronal structure.

Current models of sensory systems use various concepts to define the goals of learning. Accordingly, sensory systems constitute generative models, or internal representations of the environment, and they use sparse-, independent- and maximum entropy codes. In visual and auditory systems, internal representations are of much higher dimensionality than the input; e.g. the visual cortex has many more neurons than the eye. The methodology of choice to learn internal sensory representations is unsupervised learning. Intuitively the goals of unsupervised learning express that neural activity in the sensory system should be as variable as possible in order to use the resources optimally, and at the same time must reflect the information of the environment in a reproducible manner.

The goals of motor systems are very different from those of sensory systems. The outputs are of much lower dimensionality than the inputs, e.g. there are fewer body and limb muscles than there are neurons in the motor cortex. Coordinated motor patterns result in actions, and their goal is often to seek a reward. The methodology of choice to learn action strategies is reinforcement learning [21] in which an agent learns by trial and error to perform an action to receive a reward. This yields a powerful method to develop goal-directed action strategies.

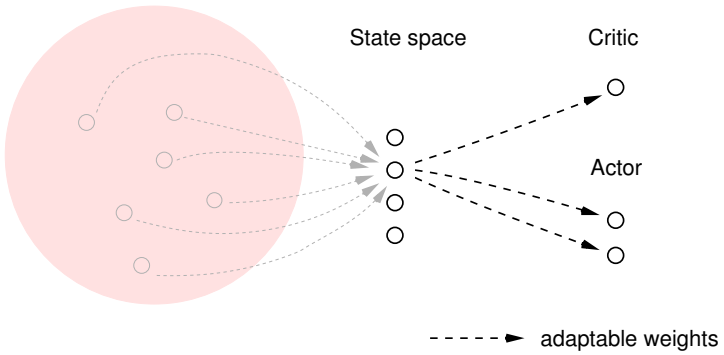

Figure 1: Architecture for reinforcement learning. A single active unit in the state space encodes the current situation of the agent. The critic unit encodes the value of the current state. One of the actor units is activated to encode the action the agent is going to take. The critic and the actor units receive connections from all state space units, but only one unit's connections are shown for clarity. The faint structure on the left denotes sensory preprocessing. This is required to obtain a usable state space encoding from raw sensory input. Our central tenet is that reservoir computing, or an ESN, is a viable model of sensory pre-processing.

How do motor systems interact with sensory systems? In [23] we draw a picture that the motor system is directly subject to the top-level goals of an animal, a human, or generally, an agent. Subject to reinforcement learning (RL), the motor system needs to implement a compatible neural architecture and representation. Figure 1 depicts an architecture commonly used for RL. It consists of an input layer that represents the state of the agent by the activation of one computational unit while all other units are silent. An output layer represents a chosen action by a corresponding unit. A critic guides learning. Given the low dimensionality of muscles, the output layer is not a problem. The high dimensionality of sensory input, however, leads to a "curse of dimensionality" problem: in high dimensions, not every state can be represented by a dedicated unit of the state space.

A solution to this problem may be that — guided by unsupervised learning — the sensory system maps the raw sensory stimulus into a low-dimensional

state space in which states are action-relevant. Ideally, two different sensory states would only be distinguished if they required different actions. So if two situations require the same action, then they need not be distinguished in the first place because that would cause unnecessary load to the decision system. In order for the sensory system to "know" which situations to distinguish and which ones not, there must be feedback from the motor system.

Algorithms dealing with sensory environments that are too large to define a state space directly are reviewed in [25]. Some methods reduce (and preprocess) the sensory space to an action-relevant subspace, or, in an alternative view, enhance the sub-part of the sensory space that is task-relevant.

One method is by directing limited sensory system capabilities, such as the focus of attention, or the center of foveated vision, to task-relevant environmental features [24][1]. Acquiring task-relevant features leads to "consistent" internal states, i.e. states that can be used to assign a value by the critic (cf. Fig. 1). "Inconsistent" internal states are those that cannot distinguish between different relevant environmental states ("perceptual aliasing"). They can be identified in that actions taken from them lead to inconsistent values (e.g. a given action from a given state yields conflicting values when repeated). A sensor-action strategy that directs the sensors (or the attention) to relevant environmental states can therefore also be based on a strategy of maximizing the state value, reminiscent of the motor action strategy.

Another method dealing with a large sensory input dimensionality is to increase the sensory system resolution where different sensory states are mapped to the same internal representation. This can be done by growing a classification tree that obtains new branches whenever a new distinction, i.e. a new class, has to be introduced (CS-QL and G-algorithms in [25]).

Sensory and motor systems are even more interrelated if one considers that the goals of actions — the external rewards — are perceived via the sensory system. The goals may even lie solely within the sensory system: for example, one may orient the eyes to a visual target for sole visual recognition of the object but without aiming at any further action. Moreover, perception itself can be regarded as an active process: selecting a particular position in the visual field by "covertly" attending to it shares properties of "overtly" attending to that position by moving the eyes into that direction.

Several combined models take account of the fact that sensory and motor systems are tightly interconnected [19, 10, 2]. In these models, sensory and motor systems form loops in that the motor output is recurrently fed back toward the sensory representations (together with the "real" sensory input), however, they do not depict a unified theory of both systems. A step toward modeling goal-directedness in a sensory system is demonstrated in a model of "attention-gated reinforcement learning" [20], however, this model does not learn action strategies as in reinforcement learning.

The high-level goals of unsupervised learning described above coincide with

[1] This leads to *deictic representations* in which *markers* are attached to these sensation- or action-relevant environmental features.

the idea of echo-state networks and the related liquid state machines [16], which are both more generally addressed by the term reservoir computing. Without bothering about learning, the design principle of the "echo-state condition" for recurrent neural networks facilitates rich and complex neural activations, reminiscent of maximum entropy coding. This ensures that the information of an input data sequence is captured for a long history time in the network's activations. Only (a) linear readout unit(s) may be trained, for example in order to perform a classification or prediction task based on the reservoir's state.

Recent experiments show that the visual cortex of cats can be used as a reservoir for advanced visual processing and classification tasks [18]. Thus, one way to approach the sensor processing in the brain can be to look at it as an echo state network, at least to the extent that the effective recurrent connectivity of cortical tissue seems to obey the below described echo state condition.

Section 2 introduces the Echo State Network (ESN). Section 3 shows by example that the ESN can classify a real-world stimulus. The result, a vector with one-of-n active unit, is ready to be used for the state space input of a reinforcement learner. Section 4 conceptually shows how an ESN can perform timing. This skill may be adopted by another neural structure that receives ESN input. Section 5 shows how feedback can improve the robustness of an ESN. Chapter 6 concludes with a discussion.

2 Standard Echo State Network

We orient the definitions on those in [12, 13]. The definition of the echo-state condition outlined by Jaeger in [12] is also described in the following in a slightly more compact form.

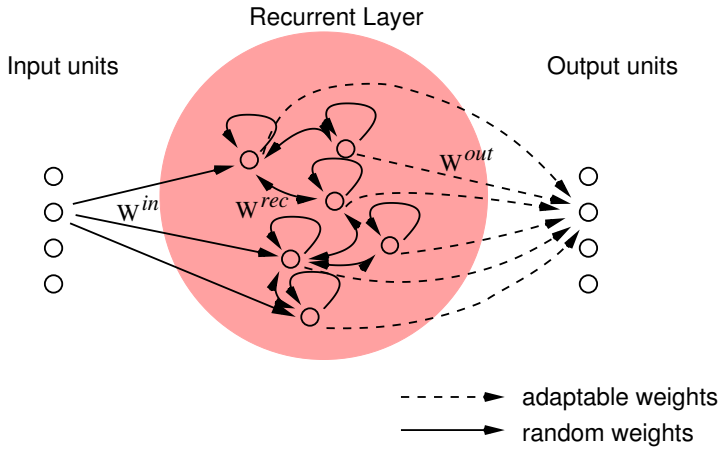

Figure 2: Principle architecture of ESN networks. Of the three different weight types, one example weight each is labeled.

Consider a time-discrete recursive function $\mathbf{x}_{t+1} = F(\mathbf{x}_t, \mathbf{u}_t)$ that is defined at least on a compact region of the vector-space $\mathbf{x} \in R^n$ and where $\mathbf{x}_t$ are to be interpreted as internal states and $\mathbf{u}_t$ is some external input sequence, i.e. the stimulus.

The definition of the echo-state condition is the following: Assume an infinite stimulus sequence: $\bar{\mathbf{u}}^{\infty} = \mathbf{u}_0, \mathbf{u}_1, \ldots$ and two random initial internal states of the system $\mathbf{x}_0$ and $\mathbf{x'}_0$. To both initial states $\mathbf{x}_0$ and $\mathbf{x'}_0$ the sequences $\bar{\mathbf{x}}^{\infty} = \mathbf{x}_0, \mathbf{x}_1, \ldots$ and $\bar{\mathbf{x'}}^{\infty} = \mathbf{x'}_0, \mathbf{x'}_1, \ldots$ can be assigned.

$$\mathbf{x}_{t+1} = F(\mathbf{x}_t, \mathbf{u}_t) \tag{1}$$

$$\mathbf{x'}_{t+1} = F(\mathbf{x'}_t, \mathbf{u}_t) \tag{2}$$

Then the system $F(.)$ fulfills the echo-state condition if independent from the set $\mathbf{u}_t$ and for any ($\mathbf{x}_0$,$\mathbf{y}_0$) and all real values $\epsilon > 0$ there exists a $\delta(\epsilon)$ for which the square Euclidean distance

$$d(\mathbf{x}_t, \mathbf{x'}_t) \leq \epsilon \tag{3}$$

for all $t \geq \delta$.

The ESN is designed to fulfil the echo-state condition. The ESN consists of three layers (cf. Fig. 2). The first layer is the input. Here the stimulus is presented to the network. The subsequent hidden layer is recurrently connected. The final layer is the output layer, which is trained to reproduce the teacher's output. The network dynamics is defined for discrete time-steps t.

The equation of the dynamics of the hidden layer is

$$\mathbf{x}_{\mathsf{lin},t+1} = \mathbf{W}^{rec}\mathbf{x}_t + \mathbf{W}^{in}\mathbf{u}_t \tag{4}$$

$$\mathbf{x}_{t+1} = \mathsf{tanh}\left(\mathbf{x}_{\mathsf{lin},t+1}\right) \tag{5}$$

$$\mathbf{y}_{t+1} = \mathbf{W}^{out}\mathbf{x}_{t+1} \tag{6}$$

where the vectors $\mathbf{u}_t$,$\mathbf{x}_t$, $\mathbf{y}_t$, are the activations of the input-, hidden- and output layer neurons, respectively, and $\mathbf{W}^{in}$, $\mathbf{W}^{rec}$, $\mathbf{W}^{out}$ are the matrices of the respective synaptic weight factors.

As mentioned above, in the classic ESN approach learning is restricted to the connections $\mathbf{W}^{out}$ between the hidden and the output layer. These weights are set by solving a system of equations:

$$\mathbf{W}^{out}\mathbf{x}_t = \mathbf{y}_t^{\mathsf{teach}} \tag{7}$$

This is obtained from writing all data points into vectors $\mathbf{x}_t$ and $\mathbf{y}_t^{\mathsf{teach}}$ (for sufficiently long time series the solution is over-defined). All other connections are chosen random. In order to fulfill the echo-state condition the connectivity matrix $\mathbf{W}^{rec}$ of the weights of the hidden layer should meet the following requirements:

- C1: Necessary for the echo state condition is that the spectral radius (maximum of absolute values of eigenvalues) of $\mathbf{W}^{rec}$ is below one.

- C2: Sufficient is that the biggest singular value of $\mathbf{W}^{rec}$ is smaller than one [2].

For all matrices $\mathbf{W}^{rec}$ that fulfill requirement 1, but do not fulfill requirement 2 no general rule is known whether or not the network meets the echo-state condition. Thus, for these matrices there is a critical regime border. In the following, a network that meets both requirements is called a sub-critical network. All connectivity matrices $\mathbf{W}^{rec}$ can easily be transformed to matrices that fulfill the second or both requirements by multiplying an appropriate scalar prefactor to the matrix.

If both conditions are met, we get an exponential convergence law (cf. Eq. 3):

$$d(\mathbf{x}_t, \mathbf{x'}_t) \leq a\eta^t, \tag{8}$$

with a constant a and $\eta < 1$. Without training, the reservoir units do not perform any systematic processing of the input. For example, there is no discernible hierarchical processing as in the visual system. One important remaining function of the reservoir is redundancy reduction: even if the input is fairly monotonic in time or if input components are dependent, the reservoir activations will nevertheless have a rich spatio-temporal structure. This rich structure is suitable for separation into classes by the readout units. In summary, the echo-state network is a model of a sensory system that is ready-to-use after very simple training.

Online Learning by using the Recursive Least Square Method One simple way of of adapting the weights $\mathbf{W}^{out}$ in the output layer is to use the Recursive Least Square Method (RLS) [7]. The RLS adaptive algorithm minimizes the square error between desired signals and unknown system's outputs. The parameter $0 < \lambda \leq 1$ is known as the forgetting factor. When $\lambda < 1$, the weighting factors give more weight to the recent samples of the error estimates (and thus to the recent samples of the observed data) compared to the old ones. In other words, the choice of a small λ results in a scheme, which puts more emphasis on the recent samples of the observed data and tends to forget the past.

We give a brief summary of the standard iterative RLS algorithm. At each iteration n an input vector $\mathbf{x}(n)$ denotes the recurrent layer's activations. The teacher signal vector is $\mathbf{y}^{\mathsf{teach}}(n)$. We start with an initial guess of the weights $\mathbf{W}^{out}$ and an auxiliary square matrix $\mathbf{\Psi}_\lambda$. These two matrices are updated at iteration n based on their previous values $\mathbf{W}^{out}(n-1)$ and $\mathbf{\Psi}_\lambda(n-1)$. Each iteration computes the output vector $\mathbf{y}(n)$, the updated weights $\mathbf{W}^{out}(n)$, and the updated matrix $\mathbf{\Psi}_\lambda(n)$. An iteration consists of the following steps.

1. Computation of a *gain vector* $\mathbf{k}$ (which has the dimensionality of $\mathbf{x}$):

[2] A closer sufficient condition has been found in [3]. It is slightly better than the one outlined here, but still leaves a gap to the necessary condition.

$$\mathbf{v}(n) \;=\; \mathbf{\Psi}_\lambda(n-1)\,\mathbf{x}(n) \tag{9}$$

$$\mathbf{k} \;=\; \frac{1}{\lambda \;+\; \mathbf{x}^T(n)\,\mathbf{v}(n)}\,\mathbf{v}(n) \tag{10}$$

2. Filtering:

$$\mathbf{y}(n) \;=\; \mathbf{W}^{out}(n-1)\,\mathbf{x}(n) \tag{11}$$

3. Error estimation:

$$\mathbf{e}(n) \;=\; \mathbf{y}^{\mathsf{teach}}(n) \;-\; \mathbf{y}(n) \tag{12}$$

4. Adaptation of the output weights:

$$\mathbf{W}^{out}(n) \;=\; \mathbf{W}^{out}(n-1) \;+\; \mathbf{e}(n)\,\mathbf{k}^T(n) \tag{13}$$

5. $\mathbf{\Psi}_\lambda(n)$ update:

$$\mathbf{\Psi}_\lambda(n) \;=\; \lambda^{-1}(\mathbf{\Psi}_\lambda(n-1) \;-\; \mathbf{k}(n)\,\mathbf{x}^T(n)\,\mathbf{\Psi}_\lambda(n-1)). \tag{14}$$

Echo State Networks for Prediction As an ansatz to understand how the formation of sensory representations and the formation of action strategies are related, we consider an echo state network as a simplified sensory representation. Next we will present several approaches that relate to this concept. In the following, we use a variant of this method to classify gestures.

3 Classification of Gestures using Echo State Networks

The ESN is used as a gesture recognizer, demonstrating that it can classify a continuous-valued input vector that arrives as a time sequence as belonging to one of several classes.

A traditional approach to classifying time sequence data is to introduce delayed weights such as in Time Delay Neural Networks. These are of feedforward type. Their delayed weights introduce a delay of $1, 2, \ldots, n$ time steps between a value arriving at an input node and the weighted signal arriving at the receiving unit. For longer time periods this is not biologically plausible since axons and dendrites are relatively fast. In the ESN all weights transfer information from one iteration time step to the immediately following time step. It is only through the recurrent computations that the network stores a history of activities.

When tracking a plain hand, it is unreliable to distinguish the hand region and face region because of similarity of their color. Therefore, we use a color ball to solve above in this experiment. One gesture pattern is given and the computer

samples tracking data by Camshift. Camshift is part of the OpenCV library [1]. It is an algorithm for tracking colored objects. For each video frame, the raw image is converted to a color probability distribution. Color distributions derived from video image sequences change over time, so the algorithm was modified to adapt dynamically to the probability distribution it is tracking. The current size and location of the tracked object are reported and used to set the size and location of the search window in the next video image.

Sampled data are the velocity (dx, dy) of the object's center and the changeable moving angle θ defined below.

$$dx(t) = x(t) - x(t-1) \tag{15}$$

$$dy(t) = y(t) - y(t-1) \tag{16}$$

$$\theta(t) = \arccos \frac{-\{dx(t-1)dx(t)+dy(t-1)dy(t)\}}{\sqrt{dx^2(t-1)+dy^2(t-1)}\sqrt{dx^2(t)+dy^2(t)}} \tag{17}$$

Each data is normalized between -1 and 1. The sampled data length is 200 steps for one gesture. The number of output nodes corresponds to the number of gesture patterns. Each output node works as a gesture recognizer. The weights $\mathbf{W}^{out}$ of the ESN are trained after all gesture data are sampled with batch learning. The learning and recognition system is shown in Fig. 3.

For the task a network with a hidden layer of 100 neurons and a random, orthonormal matrix with all eigenvalues being 0.8 was used. Each gesture was represented as one bin of the output, resulting in a 5-dimensional output vector. Training was done as described in Section 2. During the test phase the gesture corresponding to the maximum output bin was assumed to be recognized.

We trained five gestures as shown in Fig. 4: *(i)* horizontal movement, *(ii)* vertical movement, *(iii)* clockwise movement, *(iv)* anticlockwise movement, and *(v)* figure 8-shape movement.

One gesture is iterated until the end of sampling data. Input data which is intended as gesture or not, is given to the ESN that has already finished learning sequentially and the network's output nodes return their values. The maximum value of each node is recognized as the corresponding gesture. Table 1 summarizes the recognition results. The 8-shape figure is hard to distinguish, because its movement pattern overlaps with those of the clockwise and anticlockwise movements.

Input Gesture	Training Rate(%)	Test Recognition Rate(%)
Horizontal	99	91
Vertical	99	85
Clockwise	99	87
Anticlockwise	94.5	96
8-shape	84	65

Table 1: Recognition rates for each gesture for training and test data.

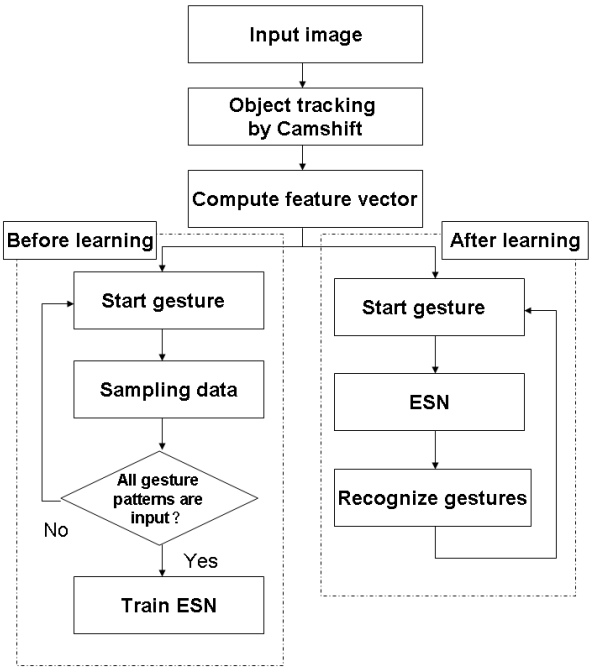

Figure 3: Gesture recognition system using an ESN.

4 Echo State Networks for Timing

Dopamine neurons in the midbrain are important for reward learning in such structures in the brain to which they project dopamine. They become active during unexpected rewards, or during the occurrence of reward-predicting ("conditioned") stimuli. They do not react to expected rewards: if one second after a conditioned stimulus a reward is persistently given, they continue firing at their resting firing rate during reward delivery. However, if the reward is unexpectantly retained, then they will briefly stop firing at the expected time of reward delivery [11]. This demonstrates their ability to measure time over a relatively long time scale of approximately one second. How can these dopamine neurons perform such timing without change of firing within this timing period?

There are several variants of predictive models that can explain how time can be measured. One model [5] accounts for reward prediction in simplified experimental settings that can be decomposed into two phases: the inter-stimulus interval (ISI) between the reward-predictive cue and the reward, and the inter-

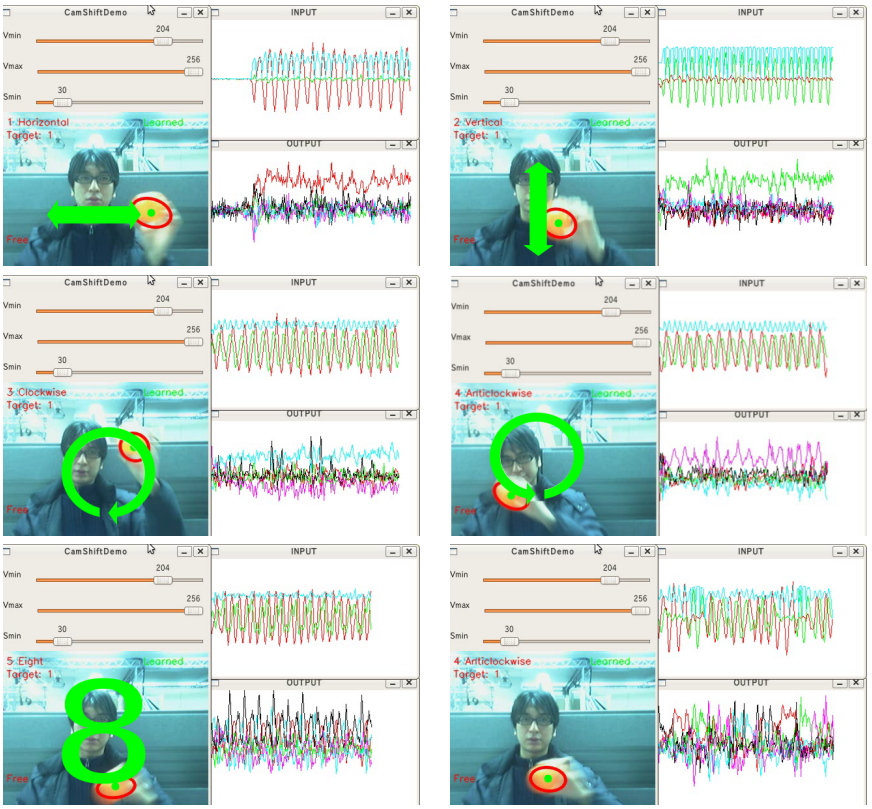

Figure 4: By using ESN it is possible to recognize gestures in real-time. The six screen shots each show the response of the ESN to a gesture in the real time test phase. Each left window displays an image that was fetched by a USB camera. Each upper right window displays input data (red = dx; green = dy; cyan = θ). Each lower right window displays the ESN's output data (red = horizontal; green = vertical; cyan = clockwise; magenta = anti-clockwise; black = 8-shape movement). The lowest screenshot on the right side depicts the reaction of the system to an arbitrary, unspecific gesture.

trial interval (ITI) between the reward and the start of the next experiment (signalled by the reward-predictive cue). However, in this setup, the timing duration and the variance is defined directly by a Gaussian function, hence the model does not explain how timing may be implemented by a neural substrate.

Another model implements neuronal units with different temporal response properties each, setting up a "spectrum" of varying responses [9]. It is doubtful though whether the time constants of individual neurons (typically in the order of tens of milliseconds) can explain delays that last seconds. However, the idea of introducing heterogeneous neurons with differential temporal properties may

also benefit other neural network methodologies such as reservoir computing.

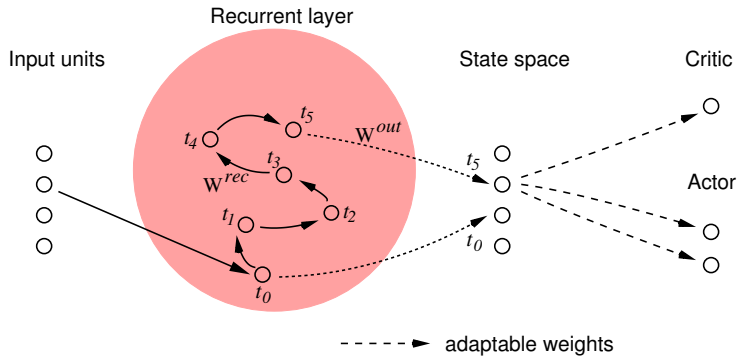

Figure 5: An ESN used for timing. A packet of neural activity starts travelling at time t_0 at the corresponding unit and arrives at t_5 at another unit.

Figure 5 diplays the idea, how one or two neurons in the state space perform timing over a longer time span without changing their firing properties during that time span. The simple idea is that another neural structure, a recurrent layer, represents a slowly moving packet of neural activation. This packet moves along the neurons labeled according to the time that they become active, $t_0, t_1, \ldots$. Only a subset of the recurrent layer neurons, those active at t_0 and at t_5 in the figure, are connected to the state space units. Hence the recurrent layer can activate the state space units at arbitrary time (or silence them via inhibitory connections).

One issue that needs to be resolved is how to implement a moving packet of activation on the recurrent layer. The activations are shaped by the recurrent weights $\mathbf{W}^{rec}$. A general ESN with random weights will produce ongoing activation change even when the input does not change, however, it is unlikely to create a regular pattern. A custom-tailored attractor network can easily produce a moving activity hill of which the shape and speed can be well determined [26]. Calcium waves that emerge intrinsically in the cortex [8] may provide target signals for supervised learning of such moving activity patterns.

Another issue is to train the readout weights $\mathbf{W}^{out}$. The dashed weights in Fig. 5 depict a possible solution in which two units of the state space are specifically activated at the time steps t_0 and t_5. How can such a solution be obtained by learning? Standard ESN learning (cf. Eq. 7) works only if the state space units' activations were given at any time. This is not the case, but we can characterize and further constrain the properties of these activations. The task here is to be coding-efficient in the state space: different units should only be recruited if either an action is to be taken or if the value of a state changes. Elsewise a given unit may remain active for some time. This information is not in the reservoir, but is available only in a top-down stream from the critic and actor.

Learning rules that feed back over several layers, as required here, are rare in reinforcement learning. "Attention-gated reinforcement learning" [20] is one of them, but it does not deal with time delays. We propose to give a reinforcement signal if the value function changes drastically during a task (i.e. at time step t_5 in the figure) to learn the connections $\mathbf{W}^{out}$ of the recurrent layer. Hence, only when task-relevant progress is made the recurrent layer should trigger a state change in the state space layer. By unfolding time in the recurrent layer the algorithm could thereby learn to deal with delays.

These considerations lead to a direction of possible future research. While it is simple to design a solution to the timing problem by hand, we do not know of a coherent framework that links reinforcement learning and ESMs, or reinforcement learning and unsupervised learning. Inspiration from neuroscience may lead to a better understanding.

5 Optimization of the Reservoir

Initially, the ESNs have been proposed for a random connectivity. It is still an open question which type of reservoir is optimal to preserve the information of the stimulus statistics in the best way. The representation needs to be sufficiently complex to handle a lot of information. However, a downside to the capability of handling complexity is the susceptibility to noise in the input. A predictive attractor network has been proposed for denoising neural representations in visual cortex [22]. Accordingly, the recurrent network activations render the response properties of V1 neurons largely independent of visual contrast. As other forms of adaptation, spike timing dependent plasticity (STDP) and intrinsic plasticity (IP) shape the network structure and dynamics in ways that allow the discovery of patterns in input time series and lead to good performance in time series prediction [14].

We suggest here a way of prediction that resembles in effect a Kalman approach, and can be seen as one possibility of denoising the input. We briefly introduce the approach, for a detailed outline please refer [17].

In the following, self-prediction was implemented into the modified ESN in the following way. The update rule of the recurrent layer is now (cf. Eqs. (4-5))

$$\mathbf{x}_{\mathsf{lin},t+1} = \left((1-\alpha)\mathbf{W}^{rec} + \alpha\mathbf{W}^{pred}\right)\mathbf{x}_t + (1-\alpha)\mathbf{W}^{in}\mathbf{u}_t \quad (18)$$

$$\mathbf{x}_{t+1} = \mathsf{tanh}\left(\mathbf{x}_{\mathsf{lin},t+1}\right) \quad (19)$$

where α is a constant parameter. Hence, not only the connections $\mathbf{W}^{out}$ to the output $\mathbf{y}_t^{\mathsf{teach}}$ must be learned but also recurrent connections $\mathbf{W}^{pred}$ predicting the linear response of the neurons $\mathbf{x}_{\mathsf{lin},t+1}$ (cf. Eq. 4) in the *next* time step. Thus, the teaching signal is

$$\{\mathbf{y}_t^{\mathsf{teach}}\ ,\ \mathbf{x}_{\mathsf{lin},t+1}\}. \quad (20)$$

In analogy to Eq. (7), the weights can be calculated by solving a system of linear equations

$$\mathbf{W}^{out}\mathbf{x}_t = \mathbf{y}_t^{\mathsf{teach}} \quad (21)$$

$$\mathbf{W}^{pred}\mathbf{x}_t \quad = \quad \mathbf{x}_{\mathsf{lin},t+1} \tag{22}$$

for all times t of the teaching period (except the initial transient period).

If the quality of the prediction is sufficient, $\mathbf{W}^{pred}\mathbf{x}_t$ should be close to $\mathbf{W}^{rec}\mathbf{x}_t + \mathbf{W}^{in}\mathbf{u}_t$. In these situations, the dynamics of the network remains almost unaffected by the value of α, as one can see in Eq. (18). In situations where prediction is poor, α is expected to determine the degree to which the dynamics are modulated by the internal model of the system, versus the actual observed data.

If α is equal to 1, the network becomes independent from the input. Of course, in the present context, this makes only sense in the case of periodic or at least quasi-periodic output signal. In this case, the network becomes an autonomous pattern generator. A value of α being 0 results in the modified network reverting to the original ESN. Other values of α result in some mixture of the two. The dynamics of the system are identical in both the initial network and the network after implementing the prediction. From a formal point of view merely the weights of the network are changed. Thus, one can write Eq. (18) as:

$$\mathbf{x}_{\mathsf{lin},t+1} \quad = \quad \mathbf{W}^{rec}_{\mathsf{new}}\mathbf{x}_t + \mathbf{W}^{in}_{\mathsf{new}}\mathbf{u}_t \tag{23}$$

where $\mathbf{W}^{rec}_{\mathsf{new}} = (1-\alpha)\mathbf{W}^{rec} + \alpha\mathbf{W}^{pred}$ and $\mathbf{W}^{in}_{\mathsf{new}} = (1-\alpha)\mathbf{W}^{in}$. In this way the equation is formally equivalent to the equation for the ESN (cf. Eq. 4), except that $\mathbf{W}^{rec}$ is replaced by $\mathbf{W}^{rec}_{\mathsf{new}}$ and $\mathbf{W}^{in}$ by $\mathbf{W}^{in}_{\mathsf{new}}$. For sufficiently high values of α the echo-state condition is no longer fulfilled, in particular if α is equal to 1. For the simulation details, we refer to the initial publication [17].

Figure 6 displays the output of the trained network with respect to a target signal for various levels of α. It can be seen that the unmodified ESN is most sensitive to noise in the input signal. As expected, the case of $\alpha = 1.0$ results in output that well reflects the structure of the input signal, since the network dynamics have been optimized to reflect the dynamics of the target signal. However, since this network receives no input it can no longer synchronize with the target signal. Best performance was observed for quite high values of α, corresponding to a network with dynamics that are a mixture of both echo states (that are a function of the input history) and self prediction states (that are a function of network's model of the input history).

Figure 7 displays the mean square error relative to signal amplitude for varying values of α. As α increases, so does the correct performance of the network up to approximately a value of 0.97, at which point performance decreases sharply. In this case of large α, the network dynamics is governed primarily by the network model, with minimal correction provided by the input signal.

These preliminary results indicate that the inclusion of self prediction may improve the performance of an ESN performing a function mapping task in the presence of additive noise. We emphasize that the present results should not be taken as conclusive. However, the inclusion of self prediction modules may represent an interesting direction for extending the basic ESN architecture.

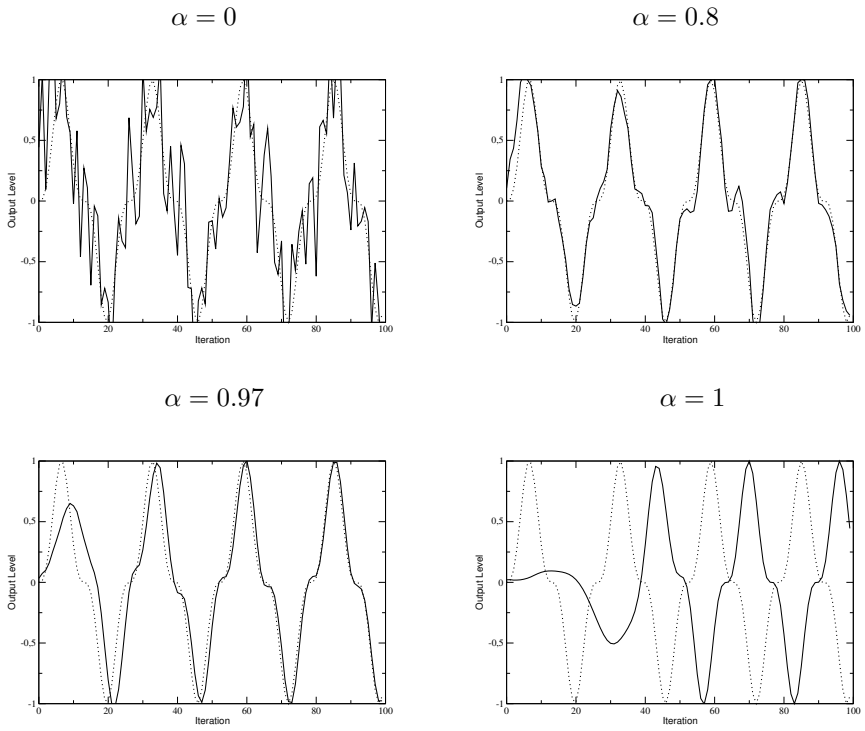

Figure 6: Effect of varying α for the modified ESN operating on input data with 0.015% relative error. Solid line shows network output, dashed line shows the target output signal synchronous to the network input. The trained pattern is $sin^3(0.24t)$, the input signal is $sin(0.24t)$. $\alpha = 0$ is a standard ESN.

6 Discussion

Echo-state networks have recently emerged as a paradigm for studying recurrent dynamics of fixed-weight neural networks, and they have a couple of applications.

By making use of the temporal dynamics they are predestined to measure time over periods longer than individual units' time constants, be it only one of several functions they are able to perform. The brain with its fast and simple neurons can so represent slow and complex events in the "external" world. A simple example of a reservoir that computes time is that of a line attractor which produces a moving hill of activity [26]. Autonomously generated, periodic activity fluctuations are also generated by predictive recurrent connections in the absense of input ($\alpha = 1$, Fig. 6). A relation to models of predictive horizontal connections in visual cortex [22] indicates that ESNs may be implemented in the brain within the horizontal connections that are abundant in all cortical

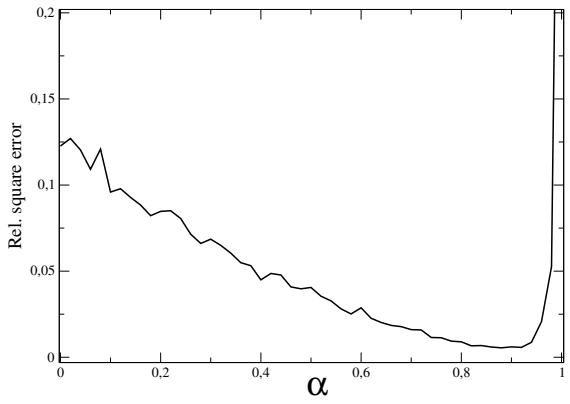

Figure 7: Network error to a noisy input signal: The network error depends on the value of α. For high values of α the network error increases again due to the lacking synchronization of the output signal to the input signal.

areas.

Besides representing the world, the brain also performs goal-directed behavior. An example tailored to reinforcement learning is an echo state network that captures a dynamical system with temporally dependent rewards such that the goodness value (Q-function) of every state can be retrieved [4]. In a more general view, a reservoir represents the real world, predicting a future expected state, hence acts as a forward model.

Also, representation and prediction of the external world subserve goal oriented behavior. Dopamine neurons in the midbrain become active when rewards, such as food, are obtained. They modulate learning in other structures so to facilitate actions leading to these rewards; hence they set the future goals. In learnt situations, dopamine neurons are predictive: if the reward is unexpectantly retained, then they will briefly pause their regular firing at the expected time of reward delivery. They can "measure" such delay times over a relatively long time scale of seconds; therefore, their clock is unlikely to be internal. We have proposed here that the cortex as their afferent network structure implements this clock. As a graphic example, a line attractor network that produces a moving hill of activity can capture time in the distance that the hill moves.

It has been proposed that unsupervised learning of world representation can be assigned to the cortex, while reinforcement learning of goal-directed actions can be assigned to the basal ganglia [6]. Both learning paradigms require specific architectures. Unsupervised learning seems to prefer recurrent architectures while reinforcement learning prefers feedforward architectures.

Earlier we have tried to understand the control system of an agent acting in an environment from the perspective of reinforcement learning in the basal ganglia [23]. It turns out that basal ganglia output feeds into the thalamus of the brain which is also the major input structure to the cortex. While basal ganglia output is traditionally regarded as action- or behavior selection, it can therefore also modify upcoming sensory input directly, such as by means of attention. This underlines that the sensory system of the brain is strongly dependent on the motor system. An implementation of feedback from the output units has been proposed in [15]. It remains to be explored further how different systems of the brain that compute different functions individually can be combined to subserve a common goal.

Acknowledgments We acknowledge financial support by the European Union through projects FP6-2005-015803 and MEXT-CT-2006-042484 and by the Hertie Foundation. A summer school project by Olav Krigolson, Luis Rivera, Isabella Cattinelli and Dimitri Ognibene contributed to some of the ideas and literature.

References

[1] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, Q2:15, 1998.

[2] J. Brown, D. Bullock, and S. Grossberg. How laminar frontal cortex and basal ganglia circuits interact to control planned and reactive saccades. *Neural Networks*, 17:471–510, 2004.

[3] M. Buechner and P. Young. A Tighter Bound for the Echo State Property. *IEEE Transaction on Neural Networks*, 17(3):820–824, 2006.

[4] K. Bush and C. Anderson. Modeling reward functions for incomplete state representations via echo state networks. In *Proc. IJCNN*, pages 2995–3000, 2005.

[5] N.D. Daw, A.C. Courville, and D.S. Touretzky. Dopamine and inference about timing. In *Proc. ICDL*, 2002.

[6] K. Doya. What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Networks*, 12:961–74, 1999.

[7] B. Farhang-Boroujeny. *Adaptive Filters*. Wiley, 1999.

[8] O. Garaschuk, J. Linn, J. Eilers, and A. Konnerth. Large-scale oscillatory calcium waves in the immature cortex. *Nat Neurosci*, 3:452–9, 2000.

[9] S. Grossberg and J.W.L. Merrill. A neural network model of adaptively timed reinforcement learning and hippocampal dynamics. *Cog Brain Res,*, 1:3–38, 1992.

[10] T.E. Hazy, M.J. Frank, and R.C. O'Reilly. Towards an executive without a homunculus: computational models of the prefrontal cortex/basal ganglia system. *Phil. Trans. R. Soc. B*, 2007.

[11] J.R. Hollerman and W. Schultz. Dopamine neurons report an error in the temporal prediction of reward during learning. *Nature Neurosci*, 1(4):304–9, 1998.

[12] H. Jaeger. The 'echo state' approach to analysing and training recurrent neural networks. In *GMD Report 148, GMD German National Research Insitute for Computer Science*, 2001.

[13] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In *Proc. of NIPS 2002*, 2003. AA14.

[14] A. Lazar, G. Pipa, and J. Triesch. Fading memory and time series prediction in recurrent networks with different forms of plasticity. *Neural Networks*, 20:312–22, 2007.

[15] W. Maass, P. Joshi, and E.D. Sontag. Principles of real-time computing with feedback applied to cortical microcircuit models. In B. Weiss, Y. Schölkopf and J. Platt, editors, *Advances in Neural Information Processing Systems, Vol. 18*, pages 835–42, 2006.

[16] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework on for neural computation based on perturbations. In *NeuroCOLT Technical Report*, 2001. NC-TR-01-113.

[17] N. Mayer and M. Browne. Echo state networks and self-prediction. *Lecture Notes in Computer Science*, 3141:40 – 47, 2004.

[18] D. Nikolic. The brain as a liquid state machine. 2006. NIPS 2006: Workshop on Echo State Networks and Liquid State Machines, published online.

[19] T.J. Prescott, T. Stafford, and K. Gurney. A robot model of the basal ganglia: Behavior and intrinsic processing. *Neural Networks*, 19:31–61, 2006.

[20] P.R. Roelfsema and A. van Ooyen. Attention-gated reinforcement learning of internal representations for classification. *Neur Comp*, 17:2176–214, 2005.

[21] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[22] C. Weber. Self-organization of orientation maps, lateral connections, and dynamic receptive fields in the primary visual cortex. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Proc. ICANN*, pages 1147–52. Springer-Verlag Berlin Heidelberg, 2001.

[23] C. Weber, M. Elshaw, S. Wermter, J. Triesch, and C. Willmot. *Reinforcement Learning: Theory and Applications*, chapter Reinforcement Learning Embedded in Brains and Robots. 2008.

[24] S.D. Whitehead and D.H. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83, 1991.

[25] S.D. Whitehead and L.J. Lin. Reinforcement learning of non-Markov decision processes. *Artificial Intelligence*, 73:271–306, 1995.

[26] K. Zhang. Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory. *J Neurosci*, 16:2112–26, 1996.