Available online at www.sciencedirect.com
SCIENCE DIRECT
Journal of Physiology - Paris 98 (2004) 315–330
Journal of Physiology Paris
www.elsevier.com/locate/jphysparis

Fading memory and kernel properties of generic cortical microcircuit models ☆

Wolfgang Maass [a,*], Thomas Natschläger [a], Henry Markram [b]

[a] Institute for Theoretical Computer Science, Technische Universitaet Graz, A-8010 Graz, Austria
[b] Brain Mind Institute, EPFL, Lausanne, Switzerland

Abstract

It is quite difficult to construct circuits of spiking neurons that can carry out complex computational tasks. On the other hand even randomly connected circuits of spiking neurons can in principle be used for complex computational tasks such as time-warp invariant speech recognition. This is possible because such circuits have an inherent tendency to integrate incoming information in such a way that simple linear readouts can be trained to transform the current circuit activity into the target output for a very large number of computational tasks. Consequently we propose to analyze circuits of spiking neurons in terms of their roles as analog fading memory and non-linear kernels, rather than as implementations of specific computational operations and algorithms. This article is a sequel to [W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations, Neural Comput. 14 (11) (2002) 2531–2560, Online available as #130 from: <http://www.igi.tugraz.at/maass/publications.html>], and contains new results about the performance of generic neural microcircuit models for the recognition of speech that is subject to linear and non-linear time-warps, as well as for computations on time-varying firing rates. These computations rely, apart from general properties of generic neural microcircuit models, just on capabilities of simple linear readouts trained by linear regression. This article also provides detailed data on the fading memory property of generic neural microcircuit models, and a quick review of other new results on the computational power of such circuits of spiking neurons.

Keywords: Spiking neurons; Computational power; Neural circuits; Computational models; Analog memory; Non-linear kernels; Speech processing; Linear regression

1. Introduction

Diverse computational tasks are carried out by neural microcircuits in the cerebral cortex whose anatomical and physiological structure is quite similar in many brain areas and species. However, it is difficult to explain the potentially universal computational capabilities of such recurrent circuits of neurons. Common models for the organization of computations, such as for example Turing machines or attractor neural networks, are less suitable for modeling computations in cortical microcircuits, since these microcircuits carry out computations on continuous streams of inputs. Another difference between Turing machines and attractor neural networks on one hand (which both use widely varying task-dependent computation times until they provide an output) and many cortical computations on the other hand, is that the latter often have to provide an output within a specific and rather short deadline. Hence cortical microcircuits have to be able to support *real-time computing*. Furthermore one may argue that some aspects of computations in the brain are reminiscent of *anytime algorithms*, which (unlike Turing machines or attractor neural networks) can be prompted at any time to return their current best possible answer, which will in

☆ The work was partially supported by the Austrian Science Fund FWF, project #P15386, and PASCAL project #IST2002-506778 of the EU.
* Corresponding author.
E-mail addresses: maass@igi.tugraz.at (W. Maass), thomas.natschlaeger@scch.at (T. Natschläger), henry.markram@epfl.ch (H. Markram).

0928-4257/$ - see front matter © 2005 Published by Elsevier Ltd.
doi:10.1016/j.jphysparis.2005.09.020

general improve if the algorithm is allowed to run longer (we refer to [30] for more precise definitions of various notions from computation theory that are relevant for analyzing brain-style computing). Furthermore biological data suggest that cortical microcircuits can support several computational tasks in parallel, a hypothesis that is inconsistent with most modeling approaches. Another difference between cortical microcircuits and classical computational models is that the components of biological neural microcircuits, neurons and synapses, are highly diverse [11] and exhibit complex dynamical responses on several temporal scales [23]. This makes them unsuitable as building blocks of computational models that require simple uniform components, such as virtually all models inspired by computer science, statistical physics, or artificial neural nets. Furthermore, neurons are connected by highly recurrent circuitry (''loops within loops''), which makes it particularly difficult to use such circuits for robust implementations of specific computational tasks. Finally, computations in most computational models are partitioned into discrete steps, each of which require convergence to some stable internal state, whereas the dynamics of cortical microcircuits appears to be continuously changing. Hence, one needs a model for using continuous perturbations in inhomogeneous dynamical systems in order to carry out real-time computations on continuous input streams.

In this article we explore the computational power of generic circuits of spiking neurons in the light of a recently proposed new model, the liquid state machine [31]. To make this article self-contained, we give a quick review of that approach in Sections 2–4, providing also some new perspectives compared with [31]. In Section 2 a new demonstration is given for the possibility to multiplex diverse computations on Poisson spike trains with time-varying firing rates, using linear readouts trained by linear regression. Section 5 contains new results on the performance of generic neural microcircuit models for the recognition of spoken words that are subject to linear and non-linear time warps, as well as an account of other applications of such circuits for non-trivial computational tasks. Section 6 sums up our view of the fundamental computational properties of generic neural microcircuit models (fading memory and kernel capabilities), and examines these properties in detail for some basic circuit models. Section 7 contains pointers to software for simulating neural microcircuits and evaluating their computational power.

2. A conceptual framework for real-time neural computation

A computation is a process that assigns to inputs from some domain D certain outputs from some range R, thereby computing a function from D into R. Obviously any systematic discussion of computations requires a mathematical or conceptual framework, i.e., a computational model [38]. Perhaps the most well-known computational model is the Turing machine [42,41,38]. In this case the domain D and range R are sets of finite character strings. This computational model is universal (for deterministic offline digital computation) in the sense that every deterministic digital function that is computable at all (according to a well-established mathematical definition, see [41]) can be computed by some Turing machine. Before a Turing machine gives its output, it goes through a series of internal computation steps, the number of which depends on the specific input and the difficulty of the computational task (therefore it is called an ''offline computation''). This may not be inadequate for modeling human reasoning about chess end games, but most cognitive tasks are closer related to real-time computations on continuous input streams, where online responses are needed within specific (typically very short) time windows, regardless of the complexity of the input. In this case the domain D and range R consist of time-varying functions $u(\cdot)$, $y(\cdot)$ (with analog inputs and outputs), rather than of static character strings.

If one excites a sufficiently complex recurrent circuit (or other medium) with a continuous input stream $u(s)$, and looks at a later time $t > s$ at the current internal state $x(t)$ of the circuit, then $x(t)$ is likely to hold a substantial amount of information about recent inputs $u(s)$. In [7] it was demonstrated that even a bucket of water is a sufficiently complex dynamical system to exhibit such property. For the case of neural circuit models this was first demonstrated in [3]. Pairs of pulses and other simple temporal patterns were injected as input $u(s)$ into a randomly connected network of integrate-and-fire neurons, and it was shown that readouts from the network can be trained to transform such temporal information into a spatial code (place code). We as human observers may not be able to understand the ''code'' by which this information about $u(s)$ is encoded in the current circuit state $x(t)$, but that is obviously not essential. Essential is whether a readout neuron that has to extract such information at time t for a specific task can accomplish this.[1] But this amounts to a classical pattern recognition problem, since the temporal dynamics of the input stream $u(s)$ has been transformed by the recurrent circuit into a high-dimensional spatial pattern $x(t)$.[2] This pattern classification problem tends to be relatively easy to learn, even by a memoryless readout, provided the desired information is present in the circuit state $x(t)$. Furthermore, if the recurrent neural circuit is sufficiently large, it may support this learn-

[1] We will refer to such neuron as a *readout neuron*. Such readout neuron receives synaptic inputs from neurons in the circuit under consideration (due to the small size of circuits considered in our simulations we always assumed there that a readout neuron receives synaptic inputs from all neurons in the circuit). One can think of it either as a neuron at a different location in the brain to which numerous axons from the microcircuit converge, or of a neuron that sits within this microcircuit and has projections to targets outside of the circuit (the potential computational role of axon collaterals that deliver the same synaptic output also to neurons within the microcircuit is discussed in [26]).

[2] If one lets t vary, then the spatio-temporal pattern $x(t)$ may of course contain even more information about preceding inputs.

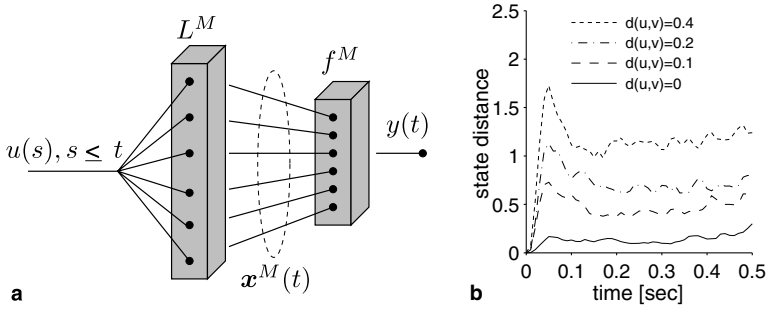

Fig. 1. (a) Structure of a liquid state machine (LSM). (b) Separation property of a generic neural microcircuit. Plotted on the y-axis is the average value of $\|\boldsymbol{x}_u^M(t) - \boldsymbol{x}_v^M(t)\|$, where $\|\cdot\|$ denotes the Euclidean norm, and $\boldsymbol{x}_u^M(t)$, $\boldsymbol{x}_v^M(t)$ denote the liquid states at time t for Poisson spike trains u and v as inputs. $d(u,v)$ is defined as distance (L_2-norm) between low-pass filtered versions of u and v, see Section 4 for details.

ing task by acting like a kernel for support vector machines (see [43]), which presents a large number of non-linear combinations of components of the preceding input stream to the readout. Such non-linear projection of the original input stream $u(\cdot)$ into a high-dimensional space tends to facilitate the extraction of information about this input stream at later times t, since it boosts the power of *linear* readouts for classification and regression tasks. Linear readouts are not only better models for the readout capabilities of a biological neuron than for example multi-layer-perceptrons, but their training is much easier and robust because it cannot get stuck in local minima of the error function (see [43,14]). These considerations suggest new hypotheses regarding the computational function of generic recurrent neural circuits: to serve as general-purpose temporal integrators, and simultaneously as kernels (i.e., non-linear projections into a higher dimensional space) to facilitate subsequent linear readout of information whenever it is needed. Note that in all experiments described in this article only the readouts were trained for specific tasks, whereas always a *fixed* recurrent circuit can be used for generating $x(t)$.

In order to analyze the potential capabilities of this approach, we have introduced in [31] the abstract model of a liquid state machine (LSM), see Fig. 1a. As the name indicates, this model has some weak resemblance to a finite state machine (see [38]). But whereas the finite state set and the transition function of a finite state machine have to be custom designed for each particular computational task (since they contain its "program"), a liquid state machine might be viewed as a universal finite state machine whose "liquid" high-dimensional analog state $x(t)$ changes continuously over time. Furthermore if this analog state $x(t)$ is sufficiently high-dimensional and its dynamics is sufficiently complex, then the states and transition functions of many concrete finite state machines F are virtually contained in it. But fortunately it is in general not necessary to reconstruct F from the dynamics of an LSM, since the readout can be trained to recover from $x(t)$ directly the information contained in the corresponding state of a finite state machine F, even if the liquid state $x(t)$ is corrupted by some—not too large—amount of noise.

Formally, an LSM M consists of a filter L^M (i.e., a function that maps input streams $u(\cdot)$ onto streams $x(\cdot)$, where $x(t)$ may depend not just on $u(t)$, but in a quite arbitrary non-linear fashion also on previous inputs $u(s)$; formally: $x(t) = (L^M u)(t)$), and a memoryless readout function f^M that maps at any time t the filter output $x(t)$ (i.e., the "liquid state") into some target output $y(t)$ (only these readout functions are trained for specific tasks in the following). Altogether an LSM computes a filter that maps $u(\cdot)$ onto $y(\cdot)$.

A recurrently connected microcircuit could be viewed in a first approximation[3] as an implementation of such general purpose filter L^M (for example some unbiased analog memory), from which different readout neurons extract and recombine diverse components of the information which was contained in the preceding input $u(\cdot)$. If a target output $y(t)$ assumes analog values, one can use instead of a single readout neuron a pool of readout neurons whose firing activity at time t represents the value $y(t)$ in space-rate-coding. In reality these readout neurons are not memoryless, but their membrane time constant is substantially shorter than the time range over which integration of information is required for most cognitive tasks. An example where the circuit input $u(\cdot)$ consists of four spike trains is indicated in Fig. 2. The generic microcircuit model consisting of 270 neurons was drawn from the distribution discussed in Section 3. In this case seven different linear readout neurons were trained to extract completely different types of information from the input stream $u(\cdot)$, which require different integration times stretching from 30 to 150 ms. The computations shown are for a novel input that did not occur during training, showing that each readout module has learned to execute its task for quite general circuit inputs. Since the readouts were modeled by linear neurons with a time constant of just 30 ms for the integration of spikes, additional temporally integrated information had to be contained at any instance t in the current firing state $x(t)$ of the recurrent circuit (its "liquid state"), see Section 3

[3] Specific cortical microcircuits may of course have more specialized feature detectors embedded into the filter L^M, e.g. simple cells in primary visual cortex.

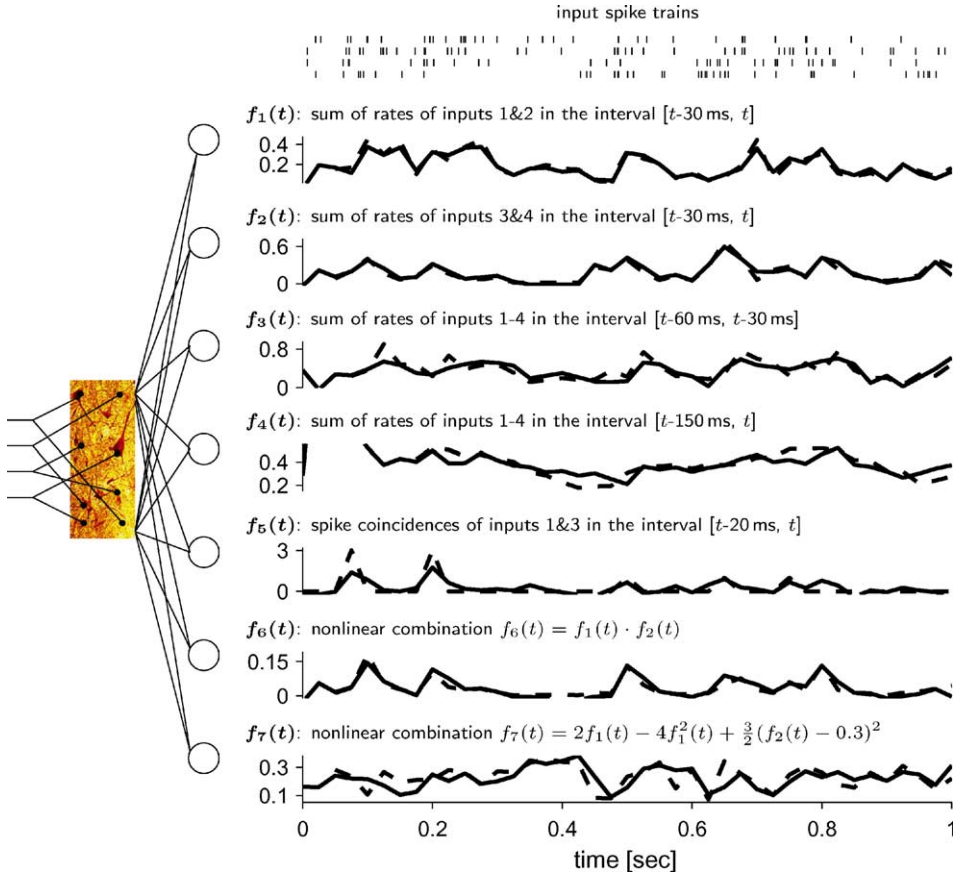

Fig. 2. Multi-tasking in real-time. Input spike trains were randomly generated in such a way that at any time t the input contained no information about input rates that were used more than 30 ms ago. Firing rates $r(t)$ were randomly drawn from the uniform distribution over [0 Hz, 80 Hz] every 30 ms, and input spike trains 1 and 2 were generated for the present 30 ms time segment as independent Poisson spike trains with this firing rate $r(t)$. This process was repeated (with independent drawings of $r(t)$ and Poisson spike trains) for each 30 ms time segment. Spike trains 3 and 4 were generated in the same way, but with independent drawings of another firing rate $\tilde{r}(t)$ every 30 ms. The results shown in this figure are for test data, that were never before shown to the circuit. Below the four input spike trains the target (dashed curves) and actual outputs (solid curves) of seven linear readout neurons are shown in real-time (on the same time axis). Targets were to output every 30 ms the actual firing rate (rates are normalized to a maximum rate of 80 Hz) of spike trains 1 and 2 during the preceding 30 ms (f_1), the firing rate of spike trains 3 and 4 (f_2), the sum of f_1 and f_2 in an earlier time interval $[t - 60\text{ ms}, t - 30\text{ ms}]$ (f_3) and during the interval $[t - 150\text{ ms}, t]$ (f_4), spike coincidences between inputs 1 and 3 ($f_5(t)$ is defined as the number of spikes which are accompanied by a spike in the other spike train within 5 ms during the interval $[t - 20\text{ ms}, t]$), a simple non-linear combinations f_6 and a randomly chosen complex non-linear combination f_7 of earlier described values. Since that all readouts were linear units, these non-linear combinations are computed implicitly within the generic microcircuit model. Average correlation coefficients between targets and outputs for 200 test inputs of length 1 s for f_1–f_7 were 0.91, 0.92, 0.79, 0.75, 0.68, 0.87, and 0.65.

for details. Whereas the information extracted by some of the readouts can be described in terms of commonly discussed schemes for "neural codes", it appears to be hopeless to capture the dynamics or the information content of the primary engine of the neural computation, the circuit state $x(t)$, in terms of such coding schemes. This view suggests that salient information may be encoded in the very high-dimensional transient states of neural circuits in a way that cannot be easily described as rate code, temporal code, or population code (see [5] for a review of frequently discussed neural coding schemes). Furthermore, the concept of "neural coding" suggests an agreement between "encoder" (the neural circuit) and "decoder" (a neural readout) which is not really needed, as long as the information is encoded in a way so that a generic neural readout can be trained to recover it.

An essentially equivalent framework for real-time computing has been introduced independently by Jäger (see [19,20]) and applied to artificial neural network models. In this approach randomly connected recurrent networks of sigmoidal neurons with synchronous updates in discrete time are employed instead of biologically motivated neural microcircuit models. The connectivity is kept rather sparse (e.g. 1% in [20]) in order to achieve that the network decomposes into many loosely coupled subcircuits, thereby establishing a richly structured reservoir of excitable dynamics. The random assignment of weights to the edges of this recurrent network is chosen in such a way that the largest absolute value of any eigenvalue of the weight matrix is less than 1. This entails that the recurrent circuit is in a regime where the impact of the initial condition of the network fades away over time. This property is called

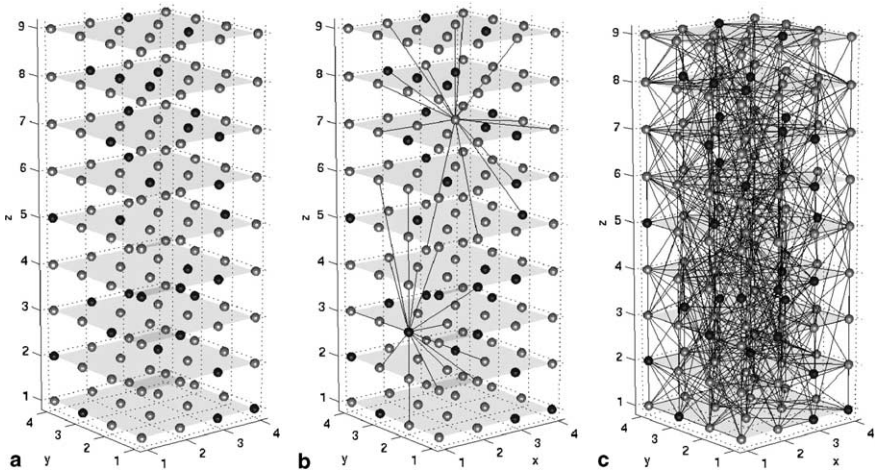

Fig. 3. Construction of a generic neural microcircuit model, as used for all computer simulations discussed in this article (only the number of neurons varied). (a) A given number of neurons is arranged on the nodes of a 3D grid. Twenty percentage of the neurons, marked in black, are randomly selected to be inhibitory. (b) Randomly chosen postsynaptic targets are shown for two of the neurons. The underlying distribution favors local connections. (c) Connectivity graph of a generic neural microcircuit model (for $\lambda = 2$, see footnote 2). This figure was prepared by Christian Naeger.

the echo state property in [19,20]. It is essentially equivalent to the fading memory property of circuits considered in [31]. It is closely related to a dynamic regime of dynamical systems which is commonly referred to as dissipative or sub-chaotic in dynamical systems theory. Although the latter concepts were traditionally only applied to autonomous systems without online input, they can also be applied for analyzing computation-relevant dynamic regimes of dynamical systems that receive online input streams (see [2,27]).

3. The generic neural microcircuit model

We used a randomly connected circuit consisting of leaky integrate-and-fire (I&F) neurons, 20% of which were randomly chosen to be inhibitory, as generic neural microcircuit model. Best performance was achieved if the connection probability was higher for neurons with a shorter distance between their somata (see Fig. 3). Parameters of neurons and synapses were chosen to fit data from microcircuits in rat somatosensory cortex (based on [11,34]). Random circuits were constructed with sparse, primarily local connectivity (see Fig. 3), both to fit anatomical data and to avoid chaotic effects.[4] We refer to Appendix A for detailed information about the circuit models used for the computer simulations discussed in this article.

The "liquid state" $x(t)$ of the recurrent circuit consisting of n neurons was modeled by an n-dimensional vector consisting of the current firing activity of these n neurons. To reflect the membrane time constant of the readout neurons a low pass filter with a time constant of 30 ms was applied to the spike trains generated by the neurons in the recurrent microcircuit. The output of this low pass filter applied separately to each of the n neurons, defines the liquid state $x(t)$. Such low pass filtering of the n spike trains is necessary for the relatively small circuits that we simulate, since at many time points t no or just very few neurons in the circuit fire (see top of Fig. 5). As readout units we used simply linear neurons (i.e., neurons that compute $\mathbf{w} \cdot \mathbf{x}(t)$) whose weights $\mathbf{w}$ were trained by linear regression (unless stated otherwise).

4. Towards a non-Turing theory for real-time neural computation

Whereas the famous results of Turing have shown that one can construct Turing machines that are universal for digital sequential offline computing, we have proposed in [31] an alternative computational theory that appears to be more adequate for parallel real-time computing on analog input streams. We quickly review here some basic concepts of this theory which are needed in the following sections.

Consider a class $\mathscr{B}$ of basis filters B (that may for example consist of the components that are available for building filters L^M of LSMs). We say that this class $\mathscr{B}$ has the *point-wise separation property* if for any two input functions $u(\cdot)$, $v(\cdot)$ with $u(s) \neq v(s)$ for some $s \leqslant t$ there exists some $B \in \mathscr{B}$ with $(Bu)(t) \neq (Bv)(t)$.[5] There exist completely different classes $\mathscr{B}$ of filters that satisfy this point-wise separation property: $\mathscr{B} = \{\text{all delay lines}\}$, $\mathscr{B} = \{\text{all linear}$

[4] A typical chaotic effect is that the current circuit state $x(t)$ also depends on circuit inputs $u(s)$ for $s \ll t$, so that a relatively small fraction of the information capacity of $x(t)$ is left for information about any particular inputs $u(s)$. This is disadvantageous for online computing if the target output at time t depends primarily on $u(s)$ for particular times $s < t$, e.g. on the most recent inputs $u(s)$ (see [2] for a discussion of chaos and edge-of-chaos in dynamical systems with online inputs).

[5] Note that it is *not* required that there exists a single $B \in \mathscr{B}$ which achieves this separation for any two different input histories $u(\cdot)$, $v(\cdot)$.

filters}, and perhaps biologically more relevant $\mathscr{B}$ = {models for dynamic synapses} (see [32]).

The complementary requirement that is demanded from the class $\mathscr{F}$ of functions from which the readout maps f^M are to be picked is the well-known *universal approximation property*: for any continuous function h and any closed and bounded domain one can approximate h on this domain with any desired degree of precision by some $f \in \mathscr{F}$. Examples for such classes are $\mathscr{F}$ = {feedforward sigmoidal neural nets}, and according to [1] also $\mathscr{F}$ = {pools of spiking neurons with analog output in space rate coding}.

A rigorous mathematical theorem [31], states that for *any* class $\mathscr{B}$ of filters that satisfies the point-wise separation property and for *any* class $\mathscr{F}$ of functions that satisfies the universal approximation property one can approximate any given real-time computation on time-varying inputs with fading memory (and hence any biologically relevant real-time computation) by a LSM M whose filter L^M is composed of finitely many filters in $\mathscr{B}$, and whose readout map f^M is chosen from the class $\mathscr{F}$. This theoretical result supports the following pragmatic procedure: In order to implement a given real-time computation with fading memory it suffices to take a filter L whose dynamics is "sufficiently complex", and train a "sufficiently flexible" readout to transform at any time t the current state $x(t) = (Lu)(t)$ into the target output $y(t)$. In principle a memoryless readout can do this, without knowledge of the current time t, provided that states $x(t)$ and $x(t')$ that require different outputs $y(t)$ and $y(t')$ are sufficiently distinct. We refer to [31] for details.

For physical implementations of LSMs it makes more sense to analyze instead of the theoretically relevant point-wise separation property the following quantitative separation property as a test for the computational capability of a filter L: How different are the liquid states $x_u(t) = (Lu)(t)$ and $x_v(t) = (Lv)(t)$ for two different input histories $u(\cdot)$, $v(\cdot)$? This is evaluated in Fig. 1b for the case where $u(\cdot)$, $v(\cdot)$ are Poisson spike trains and L is a generic neural microcircuit model as specified in Section 3. It turns out that the difference between the liquid states scales roughly proportionally to the difference between the two input histories (thereby showing that the circuit dynamic is not chaotic, or more precisely, that this simple test does not detect chaotic effects). This appears to be desirable from the practical point of view since it implies that saliently different input histories can be distinguished more easily and in a more noise robust fashion by the readout. We propose to use such evaluation of the separation capability of neural microcircuits as a test for their temporal integration capability.

5. Real-time cognition based on neural microcircuit models

The theoretical results sketched in the preceding section imply that there are no strong a priori limitations for the power of neural microcircuits for real-time computing with fading memory, provided they are sufficiently large and their components are sufficiently heterogeneous. In order to evaluate this somewhat surprising theoretical prediction, we tested it on several benchmark tasks that all involve real-time processing of time-varying inputs.

5.1. Speech recognition

One well-studied computational benchmark task for which data had been made publicly available [15] is the speech recognition task considered in [16] and [17]. The dataset consists of 500 input files: the words "zero", "one",…,"nine" are spoken by five different (female) speakers, 10 times by each speaker. The task was to construct a network of I&F neurons that could recognize each of the 10 spoken words w. Each of the 500 input files had been encoded in the form of 40 spike trains, with at most one spike per spike train[6] signaling onset, peak, or offset of activity in a particular frequency band (see top of Fig. 4). A network was presented in [17] that could solve this task with an error[7] of 0.15 for recognizing the pattern "one". No better result had been achieved by any competing networks constructed during a widely publicized internet competition [15–17].[8] A particular achievement of this network (resulting from the smoothly and linearly decaying firing activity of the 800 pools of neurons) is that it is robust with regard to linear time-warping of the input spike pattern.

We tested our generic neural microcircuit model on the same task (in fact on exactly the same 500 input files). A randomly chosen subset of 300 input files was used for training, the other 200 for testing. The generic neural microcircuit model was drawn from the distribution described in Section 3, hence from the same distribution as the circuit drawn for the completely different tasks discussed in Fig. 2, with randomly connected I&F neurons located on the integer points of a $15 \times 3 \times 3$ column. The synaptic weights of 10 readout neurons $f_0,\ldots,f_9$ which received inputs from the 135 I&F neurons in the circuit

[6] The network constructed in [17] required that each spike train contained at most one spike (see also [15]).

[7] The error (or "recognition score") S for a particular word w was defined in [15] by $S = \frac{N_{\mathrm{fp}}}{N_{\mathrm{cp}}} + \frac{N_{\mathrm{fn}}}{N_{\mathrm{cn}}}$, where N_{fp} (N_{cp}) is the number of false (correct) positives and N_{fn} and N_{cn} are the numbers of false and correct negatives. We use the same definition of error to facilitate comparison of results. The recognition scores of the network constructed in [16,17] and of competing networks of other researchers can be found at [15]. For the competition the networks were allowed to be constructed especially for their task, but only one single pattern for each word could be used for setting the synaptic weights.

[8] The network constructed in [16,17] transformed the 40 input spike trains into linearly decaying input currents from 800 pools, each consisting of a "large set of closely similar unsynchronized neurons" [17]. Each of the 800 currents was delivered to a separate pair of neurons consisting of an excitatory "α-neuron" and an inhibitory "β-neuron". To accomplish the particular recognition task some of the synapses between α-neurons (β-neurons) are set to have equal weights, the others are set to zero.

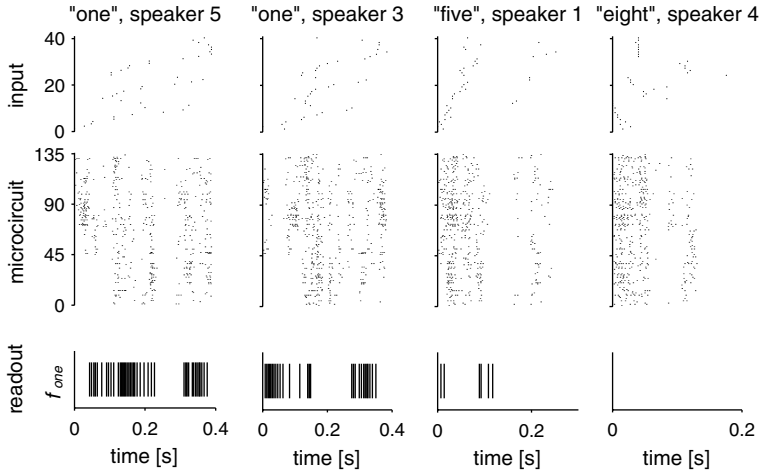

Fig. 4. Application of our generic neural microcircuit model to the speech recognition from [17]. Top row: Input spike patterns. Second row: Spiking response of the 135 I&F neurons in the neural microcircuit model. Third row: Output of an I&F neuron that was trained to fire as soon as possible when the word "one" was spoken, and as little as possible else. Although the "liquid state" presented to this readout neuron changes continuously, the readout neuron has learned to view most of them as equivalent if they arise while the word "one" is spoken (see [31] for more material on such equivalence classes defined by readout neurons).

were optimized (by linear regression) to fire whenever the input encoded the corresponding spoken word "zero", ..., "nine". Hence the whole circuit consisted of 145 I&F neurons, less than 1/30th of the size of the network constructed in [17] for the same task.[9] Nevertheless the average error achieved after training by these randomly generated generic microcircuit models was 0.14 (measured in the same way, for the same word "one" on which the analysis in [17] had focused), hence slightly better than that of the 30 times larger network custom designed for this task. The score given is the average for 50 randomly drawn generic microcircuit models. It is about the same as the error achieved by any of the networks constructed in [17] or in the associated international competition (see [15]).

The comparison of the two different approaches also provides a nice illustration of the difference between offline computing and real-time computing. Whereas the network of [16,17] implements an algorithm that needs a few hundred ms of processing time between the end of the input pattern and the answer to the classification task (250 ms in the example of Fig. 2 in [17]), the readout neurons from the generic neural microcircuit were trained to provide their answer (through firing or non-firing) immediately when the input pattern ended.

We also compared the noise robustness of the generic neural microcircuit models with that of [16,17], which had been constructed to facilitate robustness with regard to linear time warping (i.e., to a linear transformation of the time scale) of the input pattern. Since no benchmark input data were available to calculate this noise robustness we constructed such data by creating as templates 10 patterns consisting each of 40 randomly drawn Poisson spike trains at 4 Hz over 0.5 s. Noisy variations of these templates were created by first multiplying their time scale with a randomly drawn factor from $[1/3, 3]$ (thereby allowing for a 9-fold time warp), and subsequently dislocating each spike by an amount drawn independently from a Gaussian distribution with mean 0 and SD 32 ms. These spike patterns were given as inputs to the same generic neural microcircuit models consisting of 135 I&F neurons as discussed before. Ten readout neurons were trained (with 1000 randomly drawn training examples) to recognize which of the 10 templates had been used to generate a particular input (analogously as for the word recognition task). On 500 novel test examples (drawn from same distributions) they achieved an error of 0.09 (average performance of 30 randomly generated microcircuit models). The best one of 30 randomly generated circuits achieved an error of just 0.005. Furthermore it turned out that the generic microcircuit can just as well be trained to be robust with regard to *non-linear* time warps, i.e., non-linear transformations of the time scale, of a spatio-temporal pattern (it is not known whether this could also be achieved by a constructed circuit). For the case of non-linear (sinusoidal) time warp[10] an average (50 microcircuits) error of 0.2 is achieved (error of the best circuit: 0.02). This demonstrates that it is not really necessary to build noise robustness explicitly into the circuit. A randomly generated

[9] If one assumes that each of the 800 "large" pools of neurons in that network would consist of just five neurons, it contains together with the α- and β-neurons 5600 neurons. Actually, in order to achieve the required smooth linear decay of firing activity in each of the 800 pools of neurons, one probably has to make each of these 800 pools by several orders of magnitude larger than 5. Unfortunately, to the best of ones knowledge, a full realization of the circuit proposed in [17] with spiking neurons has so far not yet been carried out.

[10] A spike at time t was transformed into a spike at time $t' = g(t) : B + K \cdot (t + 1/(2\pi f) \cdot \sin(2\pi f t + \varphi))$ with $f = 2$ Hz, K randomly drawn from $[0.5, 2]$, φ randomly drawn from $[0, 2\pi]$ and B chosen such that $g(0) = 0$.

microcircuit model can easily be trained to have at least the same noise robustness as a circuit especially constructed to achieve that. In fact, it can also be trained to be robust with regard to types of noise that are very hard to handle with constructed circuits.

This test had implicitly demonstrated another point. Whereas the network of [16,17] was only able to classify spike patterns consisting of at most one spike per spike train, a generic neural microcircuit model can classify spike patterns without that restriction. It can for example also classify the original version of the speech data encoded into onsets, peaks, and offsets in various frequency bands, before all except the first events of each kind were artificially removed to fit the requirements of the network from [16,17].

We have also tested the generic neural microcircuit model on a much harder speech recognition task: to recognize the spoken word not only in real-time right after the word has been spoken, but even earlier when the word is still spoken. It turns out that the speech classification task from [15–17] is in a sense too easy for a generic neural microcircuit. If one injects the input spike trains that encode the spoken word directly into the 10 readout neurons (each of which is trained to recognize one of the 10 spoken words) one also gets a classification score that is almost as good as that of the network from [16,17]. Therefore we consider in the following the much harder task of *anytime speech recognition*. More precisely, each of the 10 readout neurons is trained to recognize the spoken word at any multiple of 20 ms during the 500 ms interval while the word is still spoken (''anytime speech recognition''). Obviously the network from [16,17] is not capable to do this. But also the trivial generic microcircuit model where the input spike trains are injected directly into the readout neurons perform poorly on this anytime speech classification task: it has an error score of 3.4 (computed as described in footnote 7, but every 20 ms). In contrast a generic neural microcircuit model consisting of 135 neurons it achieves a score of 1.4 for this anytime speech classification task (see Fig. 4 for a sample result).

One is easily led to believe that readout neurons from a neural microcircuit can give a stable output only if the firing activity (or more abstractly: the state of the dynamical system defined by this microcircuit) has reached an attractor. But this line of reasoning underestimates the capabilities of a neural readout from high-dimensional dynamical systems: even if the neural readout is just modeled by a perceptron, it can easily be trained to recognize completely different states of the dynamical system as being equivalent, and to give the same response. Indeed, Fig. 4 showed already that the firing activity of readout neurons can become quite independent from the dynamics of the microcircuit, even though the microcircuit neurons are their only source of input. To examine the underlying mechanism for the possibility of relatively independent readout response, we re-examined the readout from Fig. 4. Whereas the firing activity within the circuit was highly dynamic, the firing activity of the readout neurons was quite stable after training. The stability of the readout response does not simply come about because the spiking activity in the circuit becomes rather stable, thereby causing quite similar liquid states (see Fig. 5). It also does not come about because the readout only samples a few ''unusual'' liquid neurons as shown by the distribution of synaptic weights onto a sample readout neuron (bottom of Fig. 5). Since the synaptic weights do not change after learning, this indicates that the readout neurons have learned to define a notion of equivalence for dynamic states of the microcircuit. Indeed, equivalence classes are an inevitable consequence of collapsing the high-dimensional space of microcircuit states into a single dimension, but what is surprising is that the equivalence classes are meaningful in terms of the task, allowing invariant and appropriately scaled readout responses and therefore real-time computation on *novel inputs*. Furthermore, while the input may contain salient information that is constant for a particular readout element, it may not be for another (see for example Fig. 2), indicating that equivalence classes and dynamic stability exist purely from the perspective of the readout elements.

5.2. Predicting movements and solving the aperture problem

The same computational strategy can of course also be applied to other types of real-time cognition tasks that require temporal integration of information. In [24,28] it has been applied to a real-time analysis of motion in a simulated vision task. In particular it was shown that simple readouts from a generic neural microcircuit model can be trained in an unsupervised manner to predict the continuation of movements, whereas other readouts from the same neural microcircuit model can simultaneously be trained to estimate the direction of movement (thereby solving the aperture problem, see [33]), and to estimate the shape and velocity of the moving object. Since all these computations are carried out by readouts from the same generic microcircuit model, this approach provides a paradigm for the puzzling capability of cortical microcircuits to participate simultaneously in several different cognitive tasks.

5.3. Motor control

In [21] this approach was applied to a movement generation task, more precisely to a reaching task for a 2-joint non-linear arm model. In this case the readouts from the generic neural microcircuit model provide at any moment in time the torques for the joints (in the case of a model for a robot arm) or the input to muscles (in the case of a model for arm control by cortical motor neurons in primates) that activate the arm model. It turns out that proprioceptive or sensory feedback is essential for good performance in arm reaching tasks, even if this feedback is significantly delayed. In fact, the neural control of the arm movements works best in this model if the feedback delays lie in the biologically realistic range of 100–

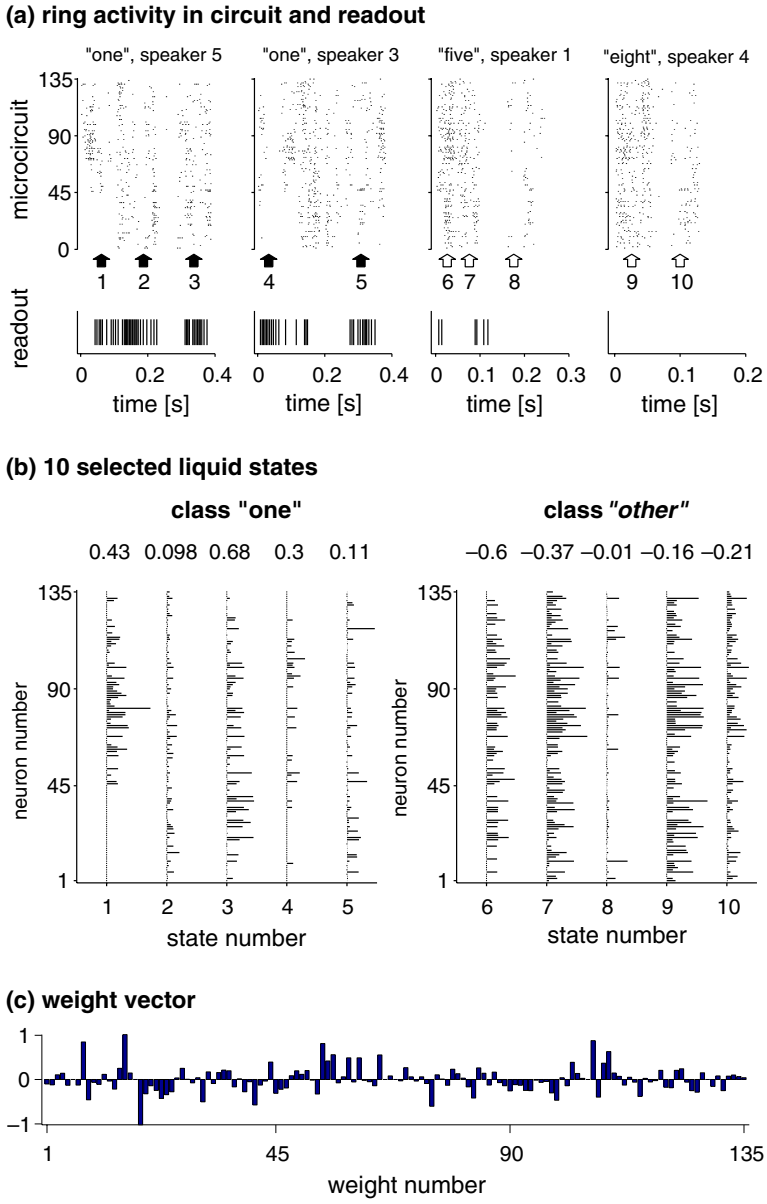

Fig. 5. Readout defined equivalence classes of liquid states. (a) The firing activity of the microcircuit for the speech recognition task from Fig. 4 is reexamined. (b) The liquid state $x(t)$ is plotted for 10 randomly chosen time points t (see arrowheads in panel a). The target output of the readout neuron is 1 for the first five liquid states, and 0 for the other five liquid states. Nevertheless the five liquid states in each of the two equivalence classes are highly diverse. But by multiplying these liquid state vectors with the weight vector of the linear readout (see panel c), one gets the values shown above the liquid state vectors. These weighted sums are separated by the threshold 0. (c) The weight vector of the linear readout.

200 ms. A possible explanation is that feedbacks with such delays complement the inherent fading memory capability of the neural circuit (i.e., the range of delays of its "internal feedbacks") particularly well.

5.4. Multiplexing pattern generation and online computing

In [22] it is demonstrated that under small changes of their connectivity structure (it suffices for example to let the probability of a synaptic connection from an inhibitory neuron a to a neuron b drop somewhat faster—as a function of the distance between both neurons—than for the case of an excitatory neuron a) generic neural microcircuit models can generate periodic patterns, as known from circuits in spinal cord of various species. Furthermore it is shown that such periodic pattern generation can be multiplexed with online processing of additional spike input, representing for example sensory or proprioceptive feedback, within the same circuit.

6. Temporal integration and kernel function of neural microcircuit models

In Section 2 we have proposed that the computational role of generic cortical microcircuits can be understood in

terms of two computational processes that are in a sense complementary, but are both implemented and intertwined in a neural microcircuit:

1. temporal integration of continuously entering information ("analog fading memory"),
2. creation of diverse non-linear combinations of components of such information to enhance the capabilities of linear readouts to extract non-linear combinations of pieces of information for diverse tasks ("kernel function").

These two computational processes may be viewed as being complementary since the first one is primarily concerned with the storage of information, whereas the second one is concerned with the fusion of information. The results reported in the preceding section suggest that both of these general computational processes are supported by generic cortical microcircuit models, since all of the benchmark problems that we discussed require temporal integration of information. Furthermore for all the considered computational tasks it sufficed to train *linear* readouts to transform liquid states into target outputs (although the target function to be computed was highly non-linear in the inputs). In this section we provide a more quantitative analysis regarding the implementation of the two computational processes 1 and 2 in generic neural microcircuit models.

6.1. Temporal integration in neural microcircuit models

In order to evaluate the temporal integration capability we considered two input distributions. These input distributions were chosen so that the mutual information (and hence also the correlation) between different segments of the input stream have value 0. Hence all temporal integration of information from earlier input segments has to be carried out by the microcircuit model, since the input itself does not provide any clues about its past. We first consider a distribution of input spike trains where every 30 ms a new firing rate $r(t)$ is chosen from the uniform distribution over the interval from 0 to 80 Hz (first row in Fig. 6). Then the spikes in each of the concurrent input spike trains are generated during each 30 ms segment by a Poisson distribution with this current rate $r(t)$ (second row in Fig. 6). Due to random fluctuation the actual sum of firing rates $r_{\text{measured}}(t)$ (plotted as dashed line in the first row) represented by these four input spike trains varies around the intended firing rate $r(t)$. $r_{\text{measured}}(t)$ is calculated as the average firing frequency in the interval $[t - 30\text{ ms}, t]$. Third row of Fig. 6 shows that the autocorrelation of both $r(t)$ and $r_{\text{measured}}(t)$ vanishes after 30 ms.

Various readout neurons, that all received the same input from the microcircuit model, had been trained by linear regression to output at various times t (more precisely: at all multiples of 30 ms) the value of $r_{\text{measured}}(t)$, $r_{\text{measured}}(t - 30\text{ ms})$, $r_{\text{measured}}(t - 60\text{ ms})$,

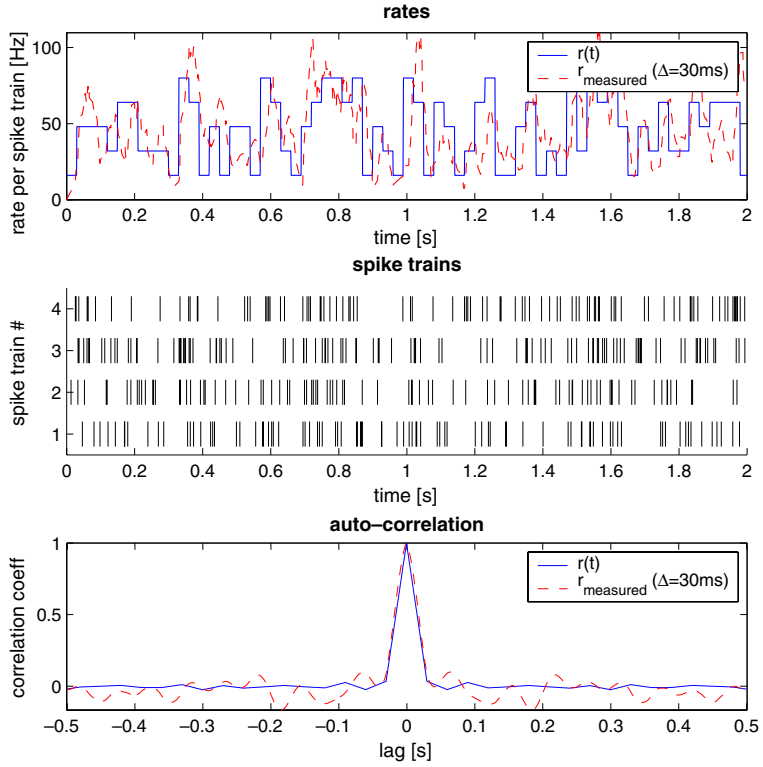

Fig. 6. Input distribution used to determine the "memory curves" for firing rates. Input spike trains (second row) are generated as Poisson spike trains with a randomly drawn rate $r(t)$ (see text for details). First row shows intended and measured firing rate $r(t)$. Last row shows autocorrelation for both.

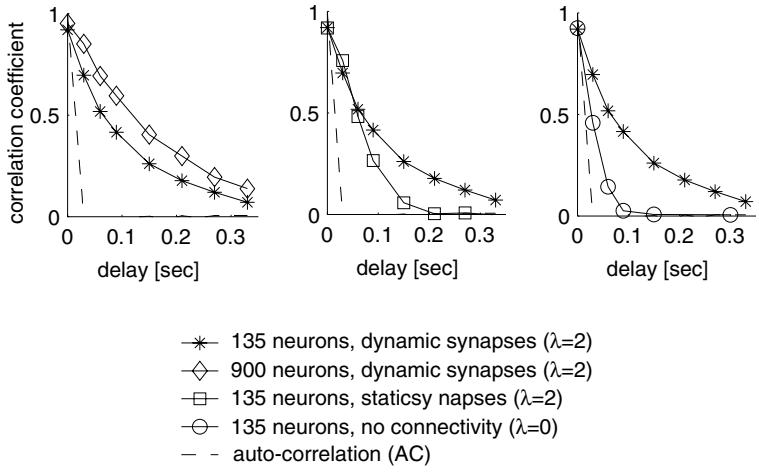

Fig. 7. Memory curves for firing rates in a generic neural microcircuit model (only performance for test data is shown). (a) Performance improves with circuit size. (b) Dynamic synapses are essential for longer recall. (c) Hidden neurons improve recall performance. In each panel the bold solid line is for a generic neural microcircuit model (discussed in Section 3) consisting of 135 neurons. All readouts were linear, trained by linear regression with 500 combinations of input spike trains (1000 in the case of the liquid with 900 neurons) of length 2 s to produce every 30 ms the desired output.

$r_{\text{measured}}(t - 90\text{ ms})$, etc. Fig. 7a shows (on test data not used for training) the correlation coefficients achieved between the target value and actual output value for eight such readouts, for the case of two generic microcircuit models consisting of 135 and 900 neurons (both with the same distance-dependent connection probability with $\lambda = 2$ discussed in Section 3). Fig. 7b shows that dynamic synapses are essential for this analog memory capability of the circuit, since the "memory curve" drops significantly faster if one uses instead static ("linear") synapses for connections within the microcircuit model. Fig. 7c shows that the intermediate "hidden" neurons in the microcircuit model are also essential for this task, since without them the memory performance also drops significantly (in the control case $\lambda = 0$ the readout receives synaptic input only from those neurons in the circuit into which one of the input spike trains is injected, hence no "hidden" neurons are involved).

It should be noted that these memory curves not only depend on the microcircuit model, but also on the diversity of input spike patterns that may have occurred in the input before, at, and after that time segment in the past from which one recalls information. Hence the recall of firing rates is particularly difficult, since there exists a huge number of diverse spike patterns that all represent the same firing rate. If one restricts the diversity of input patterns that may occur, substantially longer memory recall becomes possible, even with a fairly small circuit. In order to demonstrate this point eight randomly generated Poisson spike trains over 250 ms, or equivalently two Poisson spike trains over 1000 ms partitioned into four segments each (see top of Fig. 8), were chosen as template patterns. Then spike trains over 1000 ms were generated by choosing for each 250 ms segment one of the two templates for this segment, and by jittering each spike in the templates (more precisely: each spike was moved by an amount drawn from a Gaussian distribution with mean 0 and a SD that we refer to as "jitter", see bottom of Fig. 8). A typical spike train generated in this way is shown in the middle of Fig. 8. Because of the noisy dislocation of spikes it was very hard to recognize a specific template from a single interspike interval (and there were no spatial cues contained in this single channel input). Instead, a pattern formed by several interspike intervals had to be recognized and classified

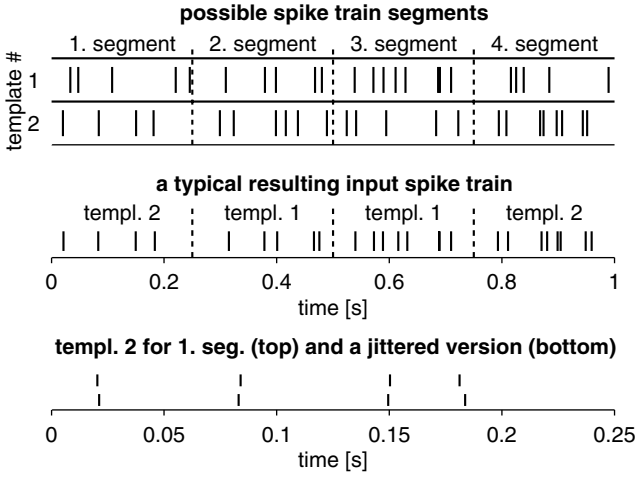

Fig. 8. Evaluating the fading memory of a generic neural microcircuit for spike patterns. In this classification task all spike trains are of length 1000 ms and consist of four segments of length 250 ms each. For each segment two templates were generated randomly (Poisson spike train with a frequency of 20 Hz); see upper traces. The actual input spike trains of length 1000 ms used for training and testing were generated by choosing for each segment one of the two associated templates, and then generating a noisy version by moving each spike by an amount drawn from a Gaussian distribution with mean 0 and a SD that we refer to as "jitter" (see lower trace for a visualization of 4 ms jitter). The task is to output with four different readouts at time $t = 1000$ ms for each of the preceding four input segments the number of the template from which the corresponding segment of the input was generated.

retrospectively. The performance of four readout neurons trained by linear regression to recall the number of the template from which the corresponding input segment had been generated is plotted in Fig. 9 (dashed line).

For comparison the memory curve for the recall of firing rates for the same temporal segments (i.e., for inputs generated as for Fig. 7, but with each randomly chosen target firing rate $r(t)$ held constant for 250 instead of 30 ms) is also plotted as thin solid line in Fig. 9, for the same generic microcircuit model consisting of 135 neurons (thick solid line is for the special case where just two firing rates were used, showing no significant difference). Fig. 9 shows that information about spike patterns of past inputs decays in a generic neural microcircuit model slower than information about firing rates of past inputs, even if just two possible firing rates may occur. One possible explanation is that the ensemble of liquid states reflecting preceding input spike trains that all represented the same firing rate forms a much more complicated equivalence class than liquid states resulting from jittered versions of a single spike pattern. This problem is amplified by the fact that information about earlier firing rates is "overwritten" with a much more diverse set of input patterns in subsequent input segments in the case of arbitrary Poisson inputs with randomly chosen rates. (The number of concurrent input spike trains that represent a given firing rate is less relevant for these memory curves; not shown.)

A theoretical analysis of memory retention in somewhat similar recurrent networks of sigmoidal neurons has been given in [18]. A theoretical analysis of fading memory for online input in recurrent networks of McCulloch–Pitts neurons is given in [2]. It is demonstrated there that it is important that the memory fades away, since otherwise the network has a chaotic dynamics and computations on specific segments of online input streams become impossible. Similar effects that also point to the "edge of chaos" as the most favorable parameter regime for online computing are demonstrated in [27] for recurrent circuits of spiking neurons.

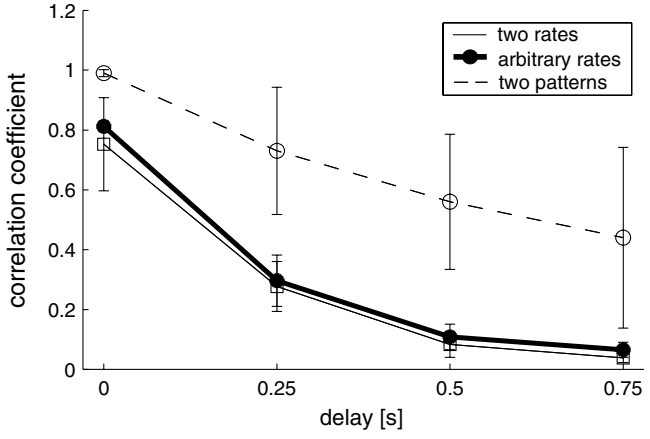

Fig. 9. Memory curves for spike patterns and firing rates (on test data). Dashed line: Correlation of trained linear readouts with the number of the templates used for generating the last input segment, and the segments that had ended 250 ms, 500 ms, and 750 ms ago (for the inputs discussed in Fig. 8). Solid lines: Correlation of trained linear readouts with the firing rates for the same time segments of length 250 ms that were used for the spike pattern classification task. Thick solid line is for the case where the ideal input firing rates can assume just 2 values (30 or 60 Hz), whereas the thin solid line is for the case where arbitrary firing rates between 0 and 80 Hz are randomly chosen. In either case the actual average input rates for the four time segments, which had to be recalled by the readouts, assumed of course a wider range.

6.2. Kernel function of neural microcircuit models

It is well-known (see [37,43,39]) that the power of linear readouts can be boosted by two types of preprocessing:

- computation of a large number of non-linear combinations of input components and features,
- projection of the input into a very high-dimensional space.

In machine learning both preprocessing steps are carried out simultaneously by a so-called kernel, that uses a mathematical trick (see [39,43]) to avoid explicit computations in high-dimensional spaces. In contrast, in our model for computation in neural microcircuits both operations of a kernel are physically implemented (by the microcircuit). The high-dimensional space into which the input is projected is the state space of the neural microcircuit (according to [6] a cubic millimeter of gray matter in the neocortex contains typically about 50 000 neurons). This implementation makes use of the fact that the precise mathematical formulas by which these non-linear combinations and high-dimensional projections are computed are less relevant. Hence these operations can be carried out by "found" neural circuits that have not been constructed for a particular task. The fact that the generic neural microcircuit models in our simulations automatically compute an abundance of non-linear combinations of input fragments can be seen from the fact that the target output values for the tasks considered in Figs. 2 and 4 are non-linear in the input, but are nevertheless approximated quite well by *linear* readouts from the current state of the neural microcircuit.

The capability of neural microcircuits to boost the power of linear readouts by projecting the input into higher dimensional spaces is further underlined by joint work with Häusler [13]. There the task to recover the number of the template spike pattern used to generate the second-to-last segment of the input spike train[11] was carried out by generic neural microcircuit models of different sizes, ranging from 12 to 784 neurons. In each case a perceptron was trained by the Δ-rule to classify at time 0 the template that had been used to generate the input in the time segment $[-500, -250$ ms$]$. The results of the computer simulations

[11] This is exactly the task of the second readout in the spike pattern classification task discussed in Figs. 8 and 9.

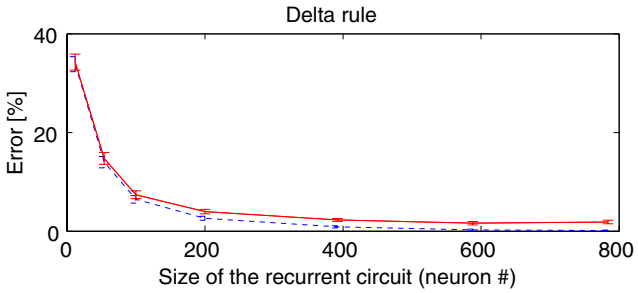

Fig. 10. The performance of a trained readout (perceptron trained by the Δ-rule) for microcircuit models of different sizes, but each time for the same input injected into the microcircuit and the same classification task for the readout. The error (plotted as percentage of incorrect classifications) decreases with growing circuit size, both on the training data (dashed line) and on new test data (solid line) generated by the same distribution.

reported in Fig. 10 show that the performance of such (thresholded) linear readout improves drastically with the size of the microcircuit into which the spike train is injected, and therefore with the dimension of the "liquid state" that is presented to the readout.

A quantitative test for evaluating the kernel function of simulated and real neural microcircuits is proposed in [27]. For an optimal kernel each of the 2^m possible dichotomies on an ensemble E consisting of m different input streams $I_1,\ldots,I_m$ to the circuit (kernel) can be implemented by a perceptron that receives as input the output of the circuit at some subsequent time t_0. This is equivalent to the condition that the m circuit output vectors $O_1,\ldots,O_m$ at time t_0 for the m different input streams $I_1,\ldots,I_m$ from the ensemble E are linearly independent. Hence it is proposed in [27] to view the rank of the matrix with columns $O_1,\ldots,O_m$ as a preliminary measure for the quality of a circuit (kernel). A more thorough computational analysis shows that a good kernel supports in addition generalization of the perceptron-output to noisy variations of the circuit inputs $I_1,\ldots,I_m$. This second kernel property can be captured quite well by a second quantitative measure for kernels that is based on VC-dimension theory (see [27,43]).

7. Software for evaluating the computational capabilities of neural microcircuit models

New software for the creation, fast simulation and computational evaluation of neural microcircuit models has recently been written by Natschläger, see [35]. This software, which has been made available for free use on WWW.LSM.TUGRAZ.AT, uses an efficient C^{++} kernel for the simulation of neural microcircuits.[12] But the construction and evaluation of these microcircuit models can be carried out conveniently in MATLAB. In particular the website contains MATLAB scripts that can be used for validating the results reported in this article. The object oriented style of the software makes it easy to change the microcircuit model or the computational tasks used for these tests.

[12] For example a neural microcircuit model consisting of a few hundred leaky integrate-and-fire neurons with up to 1000 dynamic synapses can be simulated in real-time on a current generation PC.

8. Discussion

We have shown that the inherent dynamics of cortical microcircuit models, which appears to be virtually impossible to understand in detail in terms of commonly discussed neural codes, nevertheless presents information about the recent past of its input stream in such a way that a single perceptron (or linear readout in the case where an analog output is needed) can immediately extract from it the "right answer". Traditional approaches towards producing the outputs of such complex computations in a computer usually rely on a sequential algorithm consisting of a sequence of computation steps involving elementary operations such as feature extraction, addition and multiplication of numbers, and "binding" of related pieces of information. The simulation results discussed in this article demonstrate that a completely different organization of such computations is possible, which does not require to implement these seemingly unavoidable elementary operations. Furthermore, this alternative computation style is supported by theoretical results (see Section 4), which suggest that it is in principle as powerful as von Neumann style computational models such as Turing machines, but more adequate for the type of real-time computing on analog input streams that is carried out by the nervous system.

This alternative conceptual framework places some basic concepts of computational neuroscience such as receptive fields, neural coding and binding into a new context, which is likely to have an impact on their meaning and relevance. Furthermore it suggests new experimental paradigms for investigating the computational role of cortical microcircuits. Instead of experiments on highly trained animals that aim at isolating neural correlates of conjectured elementary computational operations, the approach discussed in this article suggests experiments on naturally behaving animals that focus on the role of cortical microcircuits as general purpose temporal integrators (analog fading memory) and simultaneously as "universal" preprocessors for a large number of diverse readouts (for example as high-dimensional non-linear kernels to facilitate linear readouts). The underlying computational theory (and related experiments in machine learning) support the intuitively rather surprising finding that the precise details how these two tasks are carried out (e.g. how memories from different time windows are superimposed, or which non-linear combinations are produced in the kernel) are less relevant for the performance of the computational model, since a linear readout from a high-dimensional dynamical system can in general be trained to adjust to any particular way in which these two tasks are executed.

One characteristic feature of the model for neural computation discussed in this article is that it relies less on a specific circuit structure than other approaches, as long as the statistics of connection stays within a suitable range. Thus instead of a clever construction it requires sufficiently powerful learning algorithms that are able to train simple (for example: linear) readouts to select from the ongoing circuit dynamics the right mixture of components in order to produce a suitable output function. Although this amounts to a particular simple problem of supervised learning in computer simulations (linear regression for a linear readout), it is not clear how this learning problem is solved in biological organisms. A number of positive results in this direction have been achieved by Fregnac and his collaborators, see e.g. [8,9,40,4,10,36]. It has been demonstrated in these articles that cortical neurons can be trained by suitable pairing protocols in vitro and in vivo to "read out" particular features of their synaptic input. In [25] it is shown that some of these results can be explained on the basis of well-known data on spike-timing-dependent plasticity. If similar pairing protocols are executed autonomously within the nervous system, then this provides a mechanism for training neurons to read out information from the joint activity of their presynaptic neurons in much the same way as the linear readouts were trained in the simulations reported in this article to read out information from the "liquid state" of their presynaptic neurons.

Finally we would like to point out that the computational approach discussed in this article does not try to dispute the existence of more specific circuitry and feature detectors in many neural systems. Obviously such specific circuit components can be built into the here considered generic circuit models, and are likely to enhance their performance for various computational tasks. In fact, it is shown in [12] that the computational power of neural microcircuit models increases when one takes anatomical and physiological data on lamina-specific connections between neurons into account. Likewise we do not want to suggest that synaptic plasticity is restricted to the readouts from neural circuits. Supervised training of neurons within the circuit makes content-selective non-fading memory available for real-time computing [26]. In addition generic neural microcircuits are likely to perform better for a wide family of tasks if their internal dynamics has been adapted to the particular statistics of inputs which they receive. For example the neurons in the circuit may "self-organize" to represent independent components of the input dynamics, and/or they might want to move the dynamics of the circuit towards the "edge of chaos" (see [2,27]), a dynamic regime that is apparently associated with sparse and asynchronous firing. These are topics of current research.

Acknowledgments

The work was partially supported by the Austrian Science Fund FWF, project # P15386, and PASCAL, project # IST2002-506778, of the European Union. Numerous helpful comments from two anonymous referees are gratefully acknowledged.

Appendix A. Details of circuit models

A.1. Neuron parameters

Membrane time constant 30 ms, absolute refractory period 3 ms (excitatory neurons), 2 ms (inhibitory neurons), threshold 15 mV (for a resting membrane potential assumed to be 0), reset voltage 13.5 mV, constant non-specific background current $I_b = 13.5$ nA, input resistance 1 MΩ.

A.2. Connectivity structure

The probability of a synaptic connection from neuron a to neuron b (as well as that of a synaptic connection from neuron b to neuron a) was defined as $C \cdot \exp(-D^2(a,b)/\lambda^2)$, where λ is a parameter which controls both the average number of connections and the average distance between neurons that are synaptically connected (we set $\lambda = 2$, see [31] for details). We assumed that the neurons were located on the integer points of a 3-dimensional grid in space, where $D(a,b)$ is the Euclidean distance between neurons a and b. Depending on whether a and b were excitatory (E) or inhibitory (I), the value of C was 0.3 (EE), 0.2 (EI), 0.4 (IE), 0.1 (II).

A.3. Synaptic dynamics

In the case of a synaptic connection from a to b we modeled the synaptic dynamics according to the model proposed in [34], with the synaptic parameters U (use), D (time constant for depression), F (time constant for facilitation) randomly chosen from Gaussian distributions that were based on empirically found data for such connections. The model predicts the amplitude A_k of the EPSC for the kth spike in a spike train with interspike intervals $\Delta_1, \Delta_2, \ldots, \Delta_{k-1}$ through the equations:

$$\begin{aligned} A_k &= w \cdot u_k \cdot R_k \\ u_k &= U + u_{k-1}(1-U)\exp(-\Delta_{k-1}/F) \\ R_k &= 1 + (R_{k-1} - u_{k-1}R_{k-1} - 1)\exp(-\Delta_{k-1}/D) \end{aligned} \quad (1)$$

with hidden dynamic variables $u \in [0,1]$ and $R \in [0,1]$ whose initial values for the first spike are $u_1 = U$ and $R_1 = 1$ (see [29] for a justification of this version of the equations, which corrects a small error in [34]).

A.4. Parameters for synaptic transmission

Depending on whether a and b were excitatory (E) or inhibitory (I), the mean values of the three parameters U, D, F (with D, F expressed in seconds) were chosen to be

0.5, 1.1, 0.05 (EE), 0.05, 0.125, 1.2 (EI), 0.25, 0.7, 0.02 (IE), 0.32, 0.144, 0.06 (II). The SD of each parameter was chosen to be 50% of its mean. The mean of the scaling parameter A (in nA) was chosen to be 30 (EE), 60 (EI), −19 (IE), −19 (II). In the case of input synapses the parameter A had a value of 18 nA if projecting onto a excitatory neuron and 9 nA if projecting onto an inhibitory neuron. The SD of the A parameter was chosen to be 100% of its mean and was drawn from a gamma distribution. The postsynaptic current was modeled as an exponential decay $\exp(-t/\tau_s)$ with $\tau_s = 3$ ms ($\tau_s = 6$ ms) for excitatory (inhibitory) synapses. The transmission delays between liquid neurons were chosen uniformly to be 1.5 ms (EE), and 0.8 ms for the other connections.

A.5. Initial conditions

For each simulation, the initial conditions of each I&F neuron, i.e., the membrane voltage at time $t = 0$, were drawn randomly (uniform distribution) from the interval [13.5 mV, 15.0 mV].

References

[1] P. Auer, H. Burgsteiner, W. Maass, Reducing communication for distributed learning in neural networks, in: J.R. Dorronsoro (Ed.), Proceedings of the International Conference on Artificial Neural Networks—ICANN 2002, Lecture Notes in Computer Science, vol. 2415, Springer-Verlag, 2002, pp. 123–128, Online available as #127 from: <http://www.igi.tugraz.at/maass/publications.html>.

[2] N. Bertschinger, T. Natschläger, Real-time computation at the edge of chaos in recurrent neural networks, Neural Comput. 16 (7) (2004) 1413–1436.

[3] D.V. Buonomano, M.M. Merzenich, Temporal information transformed into a spatial code by a neural network with realistic properties, Science 267 (February) (1995) 1028–1030.

[4] D. Debanne, D.E. Shulz, Y. Fregnac, Activity-dependent regulation of "on" and "off" responses in cat visual cortical receptive fields, J. Physiol. 508 (1998) 523–548.

[5] R.C. deCharms, A. Zador, Neural representation and the cortical code, Ann. Rev. Neurosci. 23 (2000) 613–647.

[6] R. Douglas, H. Markram, K. Martin, Neocortex, in: G.M. Shepherd (Ed.), The Synaptic Organization of the Brain, Oxford University Press, 2002.

[7] C. Fernando, S. Sojakka, Pattern recognition in a bucket: a real liquid brain, in: Proceedings of ECAL, 2003, Online available from: <http://www.informatics.susx.ac.uk/users/ctf20/bucket.html>.

[8] Y. Fregnac, D. Shulz, S. Thorpe, E. Bienenstock, A cellular analogue of visual cortical plasticity, Nature 333 (6171) (1988) 367–370.

[9] Y. Fregnac, D. Shulz, S. Thorpe, E. Bienenstock, Cellular analogs of visual cortical epigenesis. i. Plasticity of orientation selectivity, J. Neurosci. 12 (4) (1992) 1280–1300.

[10] Y. Fregnac, D.E. Shulz, Activity-dependent regulation of receptive field properties of cat area 17 by supervised Hebbian learning, J. Neurobiol. 41 (1) (1999) 69–82.

[11] A. Gupta, Y. Wang, H. Markram, Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex, Science 287 (2000) 273–278.

[12] S. Häusler, W. Maass, A statistical analysis of information processing properties of lamina-specific cortical microcircuit models, submitted for publication, 2005, Online available as #162 from: <http://www.igi.tugraz.at/maass/publications.html>.

[13] S. Häusler, H. Markram, W. Maass, Perspectives of the high dimensional dynamics of neural microcircuits from the point of view of low dimensional readouts, Complexity (Special Issue on Complex Adaptive Systems) 8 (4) (2003) 39–50, Online available as #137 from: <http://www.igi.tugraz.at/maass/publications.html>.

[14] S. Haykin, Neural Networks: A Comprehensive Foundation, second ed., Prentice Hall, 1999.

[15] J. Hopfield, C. Brody, The Mus silicium (sonoran desert sand mouse) web page, Base: http://moment.princeton.edu/~mus/Organism, Dataset: Base+/Competition/digits_data.html, Scores: Base+/Docs/winners.html.

[16] J.J. Hopfield, C.D. Brody, What is a moment? "Cortical" sensory integration over a brief interval, Proc. Natl. Acad. Sci. USA 97 (25) (2000) 13919–13924.

[17] J.J. Hopfield, C.D. Brody, What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration, Proc. Natl. Acad. Sci. USA 98 (3) (2001) 1282–1287.

[18] H. Jäger, Short term memory in echo state networks, GMD Report 152, German National Research Center for Information Technology, 2002.

[19] H. Jäger, Adaptive nonlinear system identification with echo state networks, in: Advances in Neural Information Processing Systems 2002 (NIPS 2002), MIT-Press, 2003, pp. 609–616.

[20] H. Jäger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication, Science 304 (2004) 78–80.

[21] P. Joshi, W. Maass, Movement generation with circuits of spiking neurons, Neural Comput. 17 (8) (2005) 1715–1738, Online available as #158 from: <http://www.igi.tugraz.at/maass/publications.html>.

[22] A. Kaske, W. Maass, A model for the interaction of oscillations and pattern generation with real-time computing in generic neural microcircuit models, Neural Networks, in press, 2005, Online available as #156 from: <http://www.igi.tugraz.at/maass/publications.html>.

[23] C. Koch, Biophysics of Computation, Oxford University Press, 1999.

[24] R.A. Legenstein, H. Markram, W. Maass, Input prediction and autonomous movement analysis in recurrent circuits of spiking neurons, Rev. Neurosci. (Special Issue on Neuroinformatics of Neural and Artificial Computation) 14 (1–2) (2003) 5–19, Online available as #140 from: <http://www.igi.tugraz.at/maass/publications.html>.

[25] R.A. Legenstein, C. Näger, W. Maass, What can a neuron learn with spike-timing-dependent plasticity? Neural Comput. 17 (11) (2005) 2337–2382, Online available as #154 from: <http://www.igi.tugraz.at/maass/publications.html>.

[26] W. Maass, P. Joshi, E.D. Sontag, Principles of real-time computing with feedback applied to cortical microcircuit models, in: Advances in Neural Information Processing Systems, vol. 18, MIT Press, 2006, to appear. Online available as #164 from: <http://www.igi.tugraz.at/maass/publications.html>.

[27] W. Maass, R.A. Legenstein, N. Bertschinger, Methods for estimating the computational power and generalization capability of neural microcircuits, in: L.K. Saul, Y. Weiss, L. Bottou (Eds.), Advances in Neural Information Processing Systems, vol. 17, MIT Press, 2005, pp. 865–872, Online available as #160 from: <http://www.igi.tugraz.at/maass/publications.html>.

[28] W. Maass, R.A. Legenstein, H. Markram, A new approach towards vision suggested by biologically realistic neural microcircuit models, in: H.H. Buelthoff, S.W. Lee, T.A. Poggio, C. Wallraven (Eds.), Proceedings of the 2nd International Workshop on Biologically Motivated Computer Vision 2002, Lecture Notes in Computer Science, vol. 2525, Springer, Berlin, 2002, pp. 282–293, Online available as #146 from: <http://www.igi.tugraz.at/maass/publications.html>.

[29] W. Maass, H. Markram, Synapses as dynamic memory buffers, Neural Networks 15 (2002) 155–161, Online available as #119 from: <http://www.igi.tugraz.at/maass/publications.html>.

[30] W. Maass, H. Markram, Theory of the computational function of microcircuit dynamics, in: Proceedings of the 2004 Dahlem Workshop on Microcircuits, MIT Press, 2005, Online available as #157 from: <http://www.igi.tugraz.at/maass/publications.html>.

[31] W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations, Neural Comput. 14 (11) (2002) 2531–2560, Online available as #130 from: <http://www.igi.tugraz.at/maass/publications.html>.

[32] W. Maass, E.D. Sontag, Neural systems as nonlinear filters, Neural Comput. 12 (8) (2000) 1743–1772, Online available as #107 from: <http://www.igi.tugraz.at/maass/publications.html>.

[33] H.A. Mallot, Computational Vision, MIT-Press, Cambridge, MA, 2000.

[34] H. Markram, Y. Wang, M. Tsodyks, Differential signaling via the same axon of neocortical pyramidal neurons, Proc. Natl. Acad. Sci. 95 (1998) 5323–5328.

[35] T. Natschläger, H. Markram, W. Maass, Computer models and analysis tools for neural microcircuits, in: R. Kötter (Ed.), Neuroscience Databases. A Practical Guide, Kluwer Academic Publishers, Boston, 2003, pp. 123–138 (Chapter 9), Online available as #144 from: <http://www.igi.tugraz.at/maass/publications.html>.

[36] A. Rene, M. Pananceau, N. Huguet, P. Bandot, Y. Fregnac, An in vivo generalization of spike-timing-dependent plasticity to multiple pairs of pre-/post-synaptic spikes in adult visual cortex, Poster at the Ladislav Tauc Conference in Neurobiology, Gif sur Yvette, December 2003.

[37] J.F. Rosenblatt, Principles of Neurodynamics, Spartan Books, New York, 1962.

[38] J.E. Savage, Models of Computation: Exploring the Power of Computing, Addison-Wesley, Reading, MA, USA, 1998.

[39] B. Schölkopf, A.J. Smola, Learning with Kernels, MIT Press, Cambridge, 2002.

[40] D.E. Shulz, Y. Fregnac, Cellular analogs of visual cortical epigenesis. ii. Plasticity of binocular integration, J. Neurosci. 12 (4) (1992) 1301–1318.

[41] R.I. Soare, Recursively Enumerable Sets and Degrees: A Study of Computable Functions and Computably Generated Sets, Springer-Verlag, 1987.

[42] A.M. Turing, On computable numbers with an application to the Entscheidungs problem, Proc. London Math. Soc. 2 (1936) 230–265.

[43] V.N. Vapnik, Statistical Learning Theory, John Wiley, New York, 1998.