Computational Models for Generic Cortical Microcircuits

Wolfgang Maass, Thomas Natschläger
Institute for Theoretical Computer Science
Technische Universitaet Graz
A-8010 Graz, Austria
{maass, tnatschl}@igi.tu-graz.ac.at

Henry Markram
Brain Mind Institute
EPFL, Lausanne
Switzerland
henry.markram@epfl.ch

June 10, 2003

1 Introduction

A key challenge for neural modeling is to explain how a continuous stream of multi-modal input from a rapidly changing environment can be processed by neural microcircuits (columns, minicolumns, etc.) in the cerebral cortex whose anatomical and physiological structure is quite similar in many brain areas and species. However, a model that could explain the potentially universal computational capabilities of such microcircuits has been missing. We propose a computational model that does not require a task-dependent construction of neural circuits. Instead it is based on principles of high dimensional dynamical systems in combination with statistical learning theory, and can be implemented on generic evolved or found recurrent circuitry. This new approach towards understanding neural computation on the micro-level also suggests new ways of modeling cognitive processing in larger neural systems. In particular it questions traditional ways of thinking about neural coding.

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. Often there is no time to wait until a computation has converged, the results are needed instantly ("anytime computing") or within a short time window ("real-time computing"). Furthermore biological data suggest that cortical microcircuits can support several real-time computational tasks in parallel, a hypothesis that is inconsistent with most modeling approaches. In addition the components of biological neural microcircuits, neurons and synapses, are highly diverse [5] and exhibit complex dynamical responses on several temporal scales. This makes them completely 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 requires 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 chapter we present a conceptual framework for the organization of computations in cortical microcircuits that is not only compatible with all these constraints, but actually requires these biologically realistic features of neural computation. Furthermore, like Turing machines, this conceptual approach is supported by theoretical results that prove the universality of the computational model, but for the biologically more relevant case of real-time computing on continuous input streams.

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 [22]. Perhaps the most well-known computational model is the Turing machine. 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 [24]) 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. We propose here an alternative computational model that is more adequate for analyzing parallel real-time computations on analog input streams, such as those occurring in generic cognitive information processing tasks. Furthermore, we present a theoretical result which implies that within this framework the computational units of a powerful computational system can be quite arbitrary, provided that sufficiently diverse units are available (see the separation property and approximation property discussed in section 4). It also is not necessary to *construct* circuits to achieve substantial computational power. Instead sufficiently large and complex "found" circuits tend to have already large computational power for real-time computing, provided that the reservoir from which their units are chosen is sufficiently diverse.

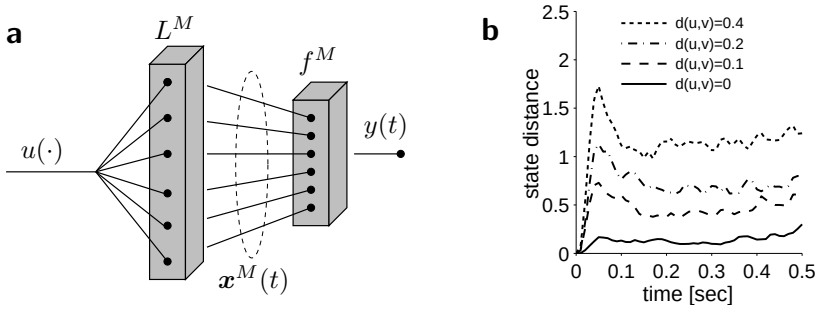

Figure 1: **a)** Structure of a Liquid State Machine (LSM). **b)** Separation property of a generic neural microcircuit. Plotted on the y-axis is the average value of $\|\boldsymbol{x}_u^M(t) - \boldsymbol{x}_v^M(t)\|$, where $\|\cdot\|$ denotes the Euclidean norm, and $\boldsymbol{x}_u^M(t)$, $\boldsymbol{x}_v^M(t)$ 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.

Our approach is based on the following observations. 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 $u(s)$ (for the case of neural circuit models this was first demonstrated by [4]). 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. 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 $x(t)$. 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 learning task by acting like a kernel for support vector machines (see [25]), which presents a large number of nonlinear combinations of components of the preceding input stream to the readout. Such nonlinear 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 [25] and [7]). 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., nonlinear 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 introduce the abstract model of a Liquid State Machine (LSM), see Figure 1a. As the name indicates, this model has some weak resemblance to a finite state

machine. 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 nonlinear 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 $y(\cdot)$.[1]

A recurrently connected microcircuit could be viewed in a first approximation 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 4 spike trains is indicated in Figure 2. The generic microcircuit model consisting of 270 neurons was drawn from the distribution discussed in section 3. In this case 7 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 biologically realistic short 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 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 fashion that looks like "noise" to the untrained observer, and that traditionally discussed "neural codes" might capture only specific aspects of the actually encoded informa-

[1] A closely related computational model was studied in [11].

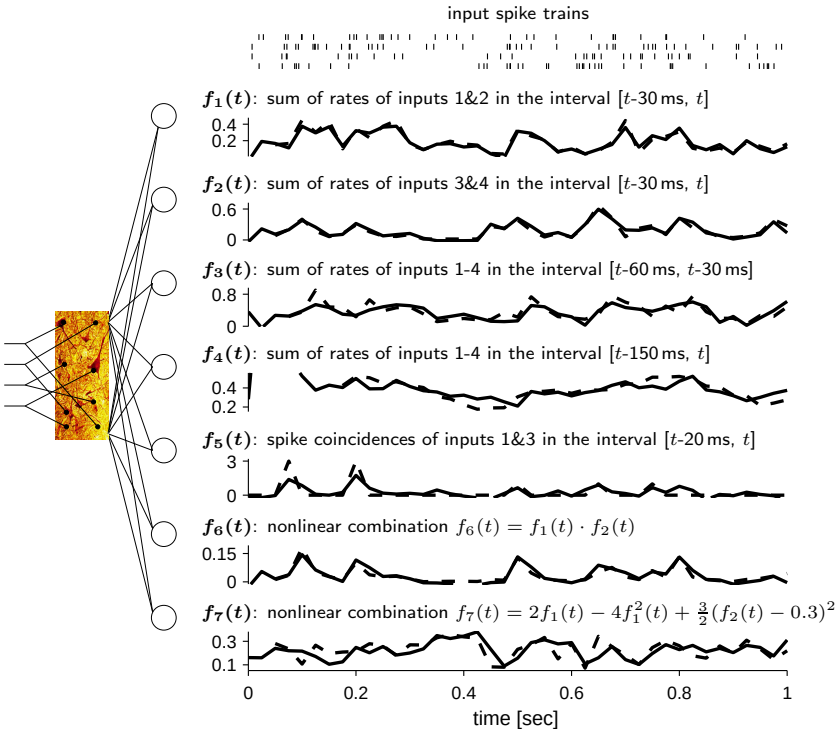

Figure 2: Multi-tasking in real-time. Input spike trains were randomly generated in such a way that at any time t the input contained no information about preceding input more than 30 ms ago. Firing rates $r(t)$ were randomly drawn from the uniform distribution over [0 Hz, 80 Hz] every 30 ms, and input 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 4 input spike trains the target (dashed curves) and actual outputs (solid curves) of 7 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&2 during the preceding 30 ms (f_1), the firing rate of spike trains 3&4 (f_2), the sum of f_1 and f_2 in an earlier time interval [t-60 ms,t-30 ms] (f_3) and during the interval [t-150 ms,t] (f_4), spike coincidences between inputs 1&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 ms,t]), a simple nonlinear combinations f_6 and a randomly chosen complex nonlinear combination f_7 of earlier described values. Since that all readouts were linear units, these nonlinear 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 to f_7 were 0.91, 0.92, 0.79, 0.75, 0.68, 0.87, and 0.65.

tion. 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.

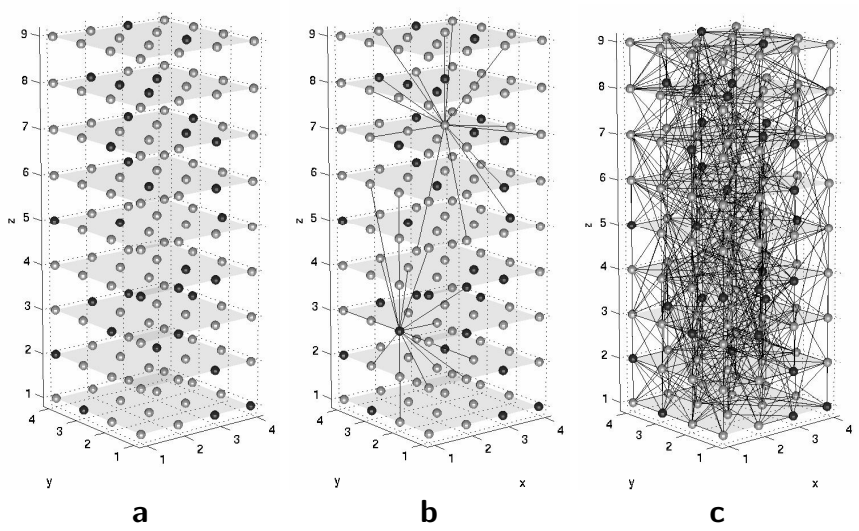

Figure 3: Construction of a generic neural microcircuit model, as used for all computer simulations discussed in this chapter (only the number of neurons varied). **a)** A given number of neurons is arranged on the nodes of a 3D grid. 20% of the neurons, marked in black, are randomly selected to be inhibitory. **b)** Randomly chosen postsynaptic targets are shown for two of the neurons. The underlying distribution favors local connections (see footnote 2 for details). **c)** Connectivity graph of a generic neural microcircuit model (for $\lambda = 2$, see footnote 2). This figure was prepared by Christian Naeger.

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 Figure 3). Parameters of neurons and synapses were chosen to fit data from microcircuits in rat somatosensory cortex (based on [5], [19] and unpublished data from the Markram Lab).[2] Random circuits were

[2] *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 nonspecific background current $I_b = 13.5$ nA, input resistance 1 MΩ. *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 [16] 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). In the case of a synaptic connection from a to b we modeled the synaptic dynamics according to the model proposed in [19], 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. Depending on whether a and b were excitatory (E) or inhibitory (I), the mean values of these three parameters (with D,F expressed in seconds, s) were chosen to be .5, 1.1, .05 (EE), .05, .125, 1.2 (EI), .25, .7, .02 (IE), .32, .144, .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

constructed with sparse, primarily local connectivity (see Figure 3), both to fit anatomical data and to avoid chaotic effects.

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 Figure 5). As readout units we used simply linear neurons, 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 propose here an alternative computational theory that is more adequate for parallel real-time computing on analog input streams. Furthermore we present a theoretical result which implies that within this framework the computational units of a powerful computational system can be quite arbitrary, provided that sufficiently diverse units are available (see the separation property and approximation property discussed below). It also is not necessary to *construct* circuits to achieve substantial computational power. Instead sufficiently large and complex "found" circuits (such as the generic circuit used as the main building block for Figure 2) tend to have already large computational power, provided that the reservoir from which their units are chosen is sufficiently diverse.

Consider a class $\mathcal{B}$ 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 $\mathcal{B}$ has the *point-wise separation property* if for any two input functions $u(\cdot), v(\cdot)$ with $u(s) \neq v(s)$ for some $s \leq t$ there exists some $B \in \mathcal{B}$ with $(Bu)(t) \neq (Bv)(t)$.[3] There exist completely different classes $\mathcal{B}$ of filters that satisfy this point-wise separation property: $\mathcal{B} = \{$all delay lines$\}$, $\mathcal{B} = \{$all linear filters$\}$, and perhaps biologically more relevant $\mathcal{B} = \{$models for dynamic synapses$\}$ (see [17]).

The complementary requirement that is demanded from the class $\mathcal{F}$ 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

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. We have shown in [16] that without synaptic dynamics the computational power of these microcircuit models decays significantly. 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].

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

any closed and bounded domain one can approximate h on this domain with any desired degree of precision by some $f \in \mathcal{F}$. Examples for such classes are $\mathcal{F} = \{$feedforward sigmoidal neural nets$\}$, and according to [3] also $\mathcal{F} = \{$pools of spiking neurons with analog output in space rate coding$\}$.

A rigorous mathematical theorem [16], states that for *any* class $\mathcal{B}$ of filters that satisfies the point-wise separation property and for *any* class $\mathcal{F}$ 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 an LSM M whose filter L^M is composed of finitely many filters in $\mathcal{B}$, and whose readout map f^M is chosen from the class $\mathcal{F}$. 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 [16] 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 Figure 1b for the case where $u(\cdot), v(\cdot)$ are Poisson spike trains and L is a generic neural microcircuit model. 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). 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 new standard test for their computational capabilities.

5 A Generic Neural Microcircuit on the Computational Test Stand

The theoretical results sketched in the preceding section implies 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.

5.1 Speech Recognition

One well-studied computational benchmark task for which data had been made publicly available [8] is the speech recognition task considered in [9] and [10]. The dataset consists of 500 input files: the words "zero", "one", ..., "nine" are spoken by 5 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[4] signaling onset, peak, or offset of activity in a particular frequency band (see top of Figure 4). A network was presented in [10] that could solve this task with an error[5] 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 [9].[6] 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.

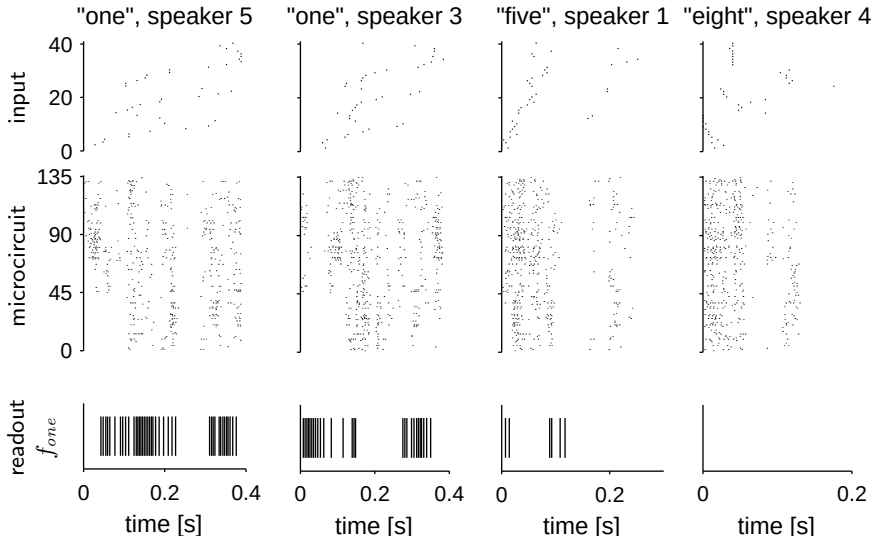

Figure 4: Application of our generic neural microcircuit model to the speech recognition from [10]. Top row: input spike patterns. Second row: spiking response of the 135 I&F neurons in the neural microcircuit model. Third row: output of an I&F neuron that was trained to fire as soon as possible when 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 learnt to view most of them as equivalent if they arise while the word "one" is spoken (see [16] for more material on such equivalence classes defined by readout neurons).

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

[4]The network constructed in [10] required that each spike train contained at most one spike.

[5]The error (or "recognition score") S for a particular word w was defined in [10] by $S = \frac{N_{fp}}{N_{cp}} + \frac{N_{fn}}{N_{cn}}$, 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 [10] and of competing networks of other researchers can be found at [8]. 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.

[6]The network constructed in [10] 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" [10]. 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.

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 Figure 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_w which received inputs from the 135 I&F neurons in the circuit were optimized (like for SVMs with linear kernels) to fire whenever the input encoded the spoken word w. Hence the whole circuit consisted of 145 I&F neurons, less than $1/30^{th}$ of the size of the network constructed in [10] for the same task[7]. 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"), 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 [10] and the associated international competition.

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 [10] 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 (450 ms in the example of Figure 2 in [10]), 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 [10], which had been constructed to facilitate robustness with regard to linear time warping 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 *nonlinear* time warp of a spatio-temporal pattern (it is not known whether this could also be achieved by a constructed circuit). For the case of nonlinear (sinusoidal) time warp[8] an average (50 microcircuits) error of 0.2 is achieved (error of the best circuit: 0.02). This demonstrates that it is

[7] If one assumes that each of the 800 "large" pools of neurons in that network would consist of just 5 neurons, it contains together with the α and β-neurons 5600 neurons.

[8] 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 $g(0) = 0$.

not really necessary to build noise robustness explicitly into the circuit. A randomly generated 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 [10] 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 [10].

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.[9] 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 [10] 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 5, 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 Figure 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, Figure 4 showed already that the firing activity of readout neuron 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 Figure 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 Figure 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 Figure 5). Since the synaptic weights do not change after learning, this indicates that the readout neurons have learned to define a notion of equivalence

[9] It turns out that the speech classification task from [10] 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 [10]. Therefore we consider in the following the much harder task of *anytime speech recognition*.

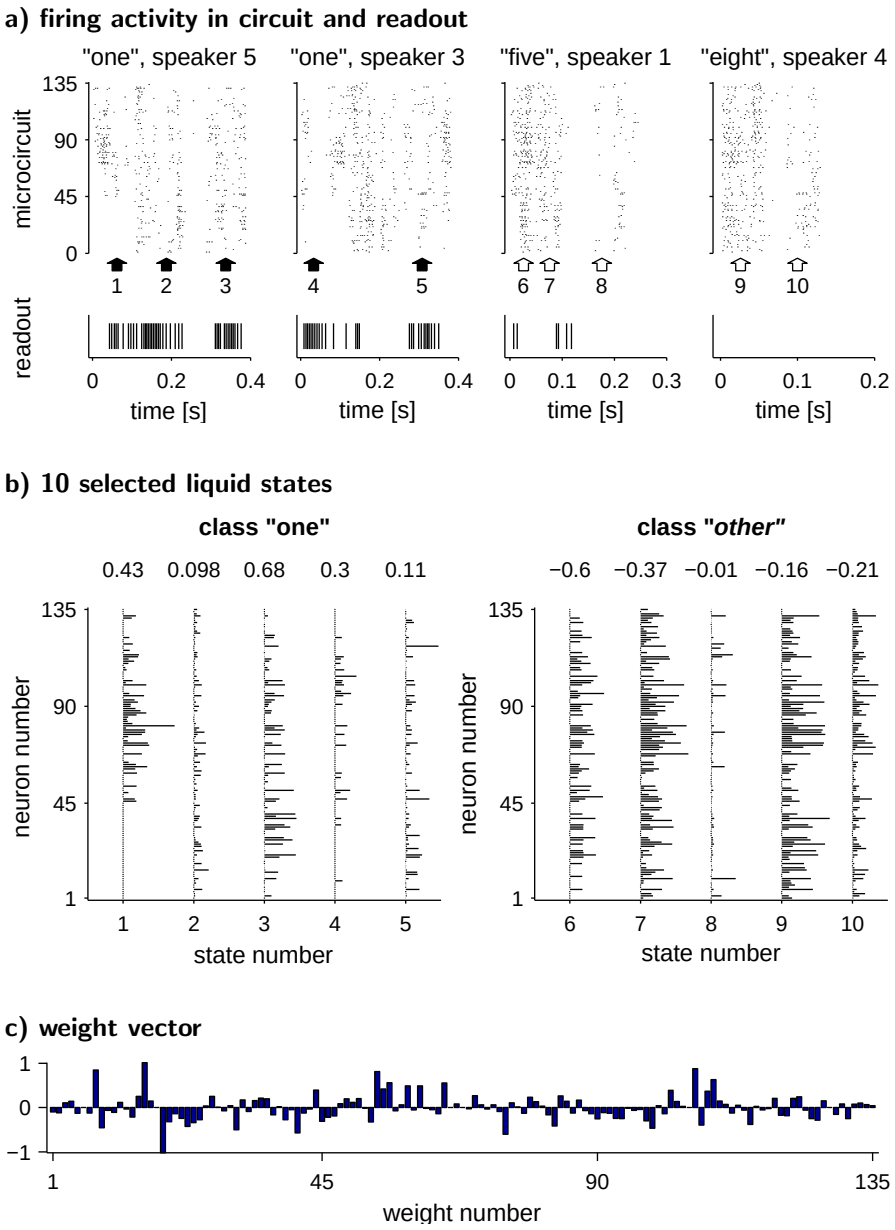

Figure 5: Readout defined equivalence classes of liquid states. **a)** The firing activity of the microcircuit for the speech recognition task from Figure 4 is reexamined. **b)** The liquid state $x(t)$ is plotted for 10 randomly chosen time points t (see arrowheads in panel a). The target output of the readout neuron is 1 for the first 5 liquid states, and 0 for the other 5 liquid states. Nevertheless the 5 liquid states in each of the 2 equivalence classes are highly diverse. But by multiplying these liquid state vectors with the weight vector of the linear readout (see panel c), the weighted sums yields the values shown above the liquid state vectors, which are separated by the threshold 0 of the readout (and by the firing threshold of the corresponding leaky integrate-and-fire neuron whose output spike trains are shown in panel a). **c)** The weight vector of the linear readout.

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 Figure 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

This section reports results of joint work with Robert Legenstein [13], [15]. The general setup of this simulated vision task is illustrated in Figure 6. Moving objects, a ball or a bar, are presented to an 8 x 8 array of sensors (panel a). The time course of activations of 8 randomly selected sensors, resulting from a typical movement of the ball, is shown in panel b. Corresponding functions of time, but for all 64 sensors, are projected as 64 dimensional input by a topographic map into a generic recurrent circuit of spiking neurons. This circuit with randomly chosen sparse connections had been chosen in the same way as the circuits for the preceding tasks, except that it was somewhat larger (768 neurons) to accommodate the 64 input channels.[10] The resulting firing activity of all 768 integrate-and-fire neurons in the recurrent circuit is shown in panel c. Panel d of Figure 6 shows the target output for 8 of the 102 readout pools. These 8 readout pools have the task to predict the output that the 8 sensors shown in panel b will produce 50 ms later. Hence their target output (dashed line) is formally the same function as shown in panel b, but shifted by 50 ms to the left. The solid lines in panel d show the actual output of the corresponding readout pools after unsupervised learning. Thus in each row of panel d the difference between the dashed and predicted line is the prediction error of the corresponding readout pool.

The diversity of object movements that are presented to the 64 sensors is indicated in Figure 7. Any straight line that crosses the marked horizontal or vertical line segments of length 4 in the middle of the 8 x 8 field may occur as trajectory for the center of an object. Training and test examples are drawn randomly from this - in principle infinite - set of trajectories, each with a movement speed that was drawn independently from a uniform distribution over the interval from 30 to 50 units per second (unit = side length of a unit square). Shown in Figure 7 are 20 trajectories that were randomly drawn from this distribution. Any such movement is carried out by an independently drawn object type (ball or bar), where bars were assumed to be oriented vertically to their direction of movement. Besides movements on straight lines one could train the same circuit just as well for predicting nonlinear movements, since nothing in the circuit was specialized for predicting linear movements.

36 readout pools were trained to predict for any such object movement the sensor activations of the 6 x 6 sensors in the interior of the 8 x 8 array 25 ms into the future. Further 36 readout pools were independently trained

[10] A 16 x 16 x 3 neuronal sheet was divided into 64 2 x 2 x 3 input regions. Each sensor injected input into 60 % randomly chosen neurons in the associated input region. Together they formed a topographic map for the 8 x 8 array of sensors.

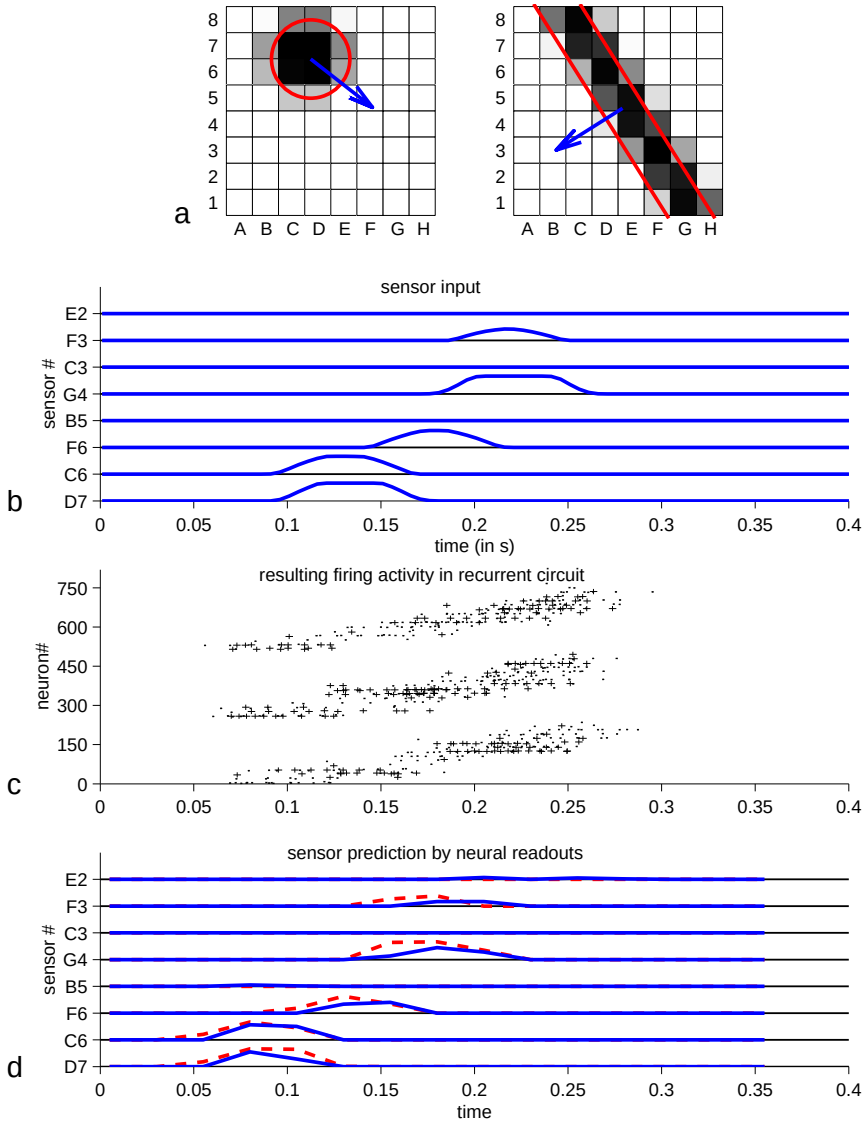

Figure 6: The prediction task. **a)** Typical movements of objects over a 8 x 8 sensor field. **b)** Time course of activation of 8 randomly selected sensors caused by the movement of the ball indicated on the l.h.s. of panel a. **c)** Resulting firing times of 768 integrate-and-fire neurons in the recurrent circuit of integrate-and-fire neurons (firing of inhibitory neurons marked by +). The neurons in the 16 x 16 x 3 array were numbered layer by layer. Hence the 3 clusters in the spike raster result from concurrent activity in the 3 layers of the circuit. **d)** Prediction targets (dashed lines) and actual predictions (solid lines) for the 8 sensors from panel b. (Predictions were sampled every 25 ms, solid curves result from linear interpolation.)

to predict their activation 50 ms into the future, showing that the prediction span can basically be chosen arbitrarily. At any time t (sampled every 25 ms from 0 to 400 ms) one uses for each of the 72 readout pools that predict sensory input ΔT into the future the actual activation of the corresponding sensor at time $t + \Delta T$ as target value ("correction") for the learning rule. The 72 readout pools for short-term movement prediction were trained by

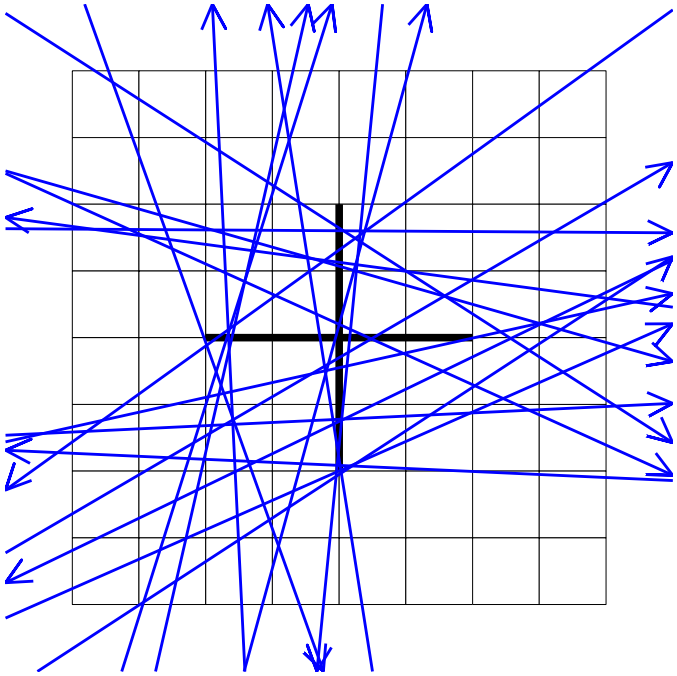

Figure 7: 20 typical trajectories of movements of the center of an object (ball or bar).

1500 randomly drawn examples of object movements. More precisely, they were trained to predict future sensor activation at any time (sampled every 25 ms) during the 400 ms time interval while the object (ball or bar) moved over the sensory field, each with another trajectory and speed.

Among the predictions of the 72 different readout pools on 300 novel test inputs there were for the 25 ms prediction 8.5 % false alarms (sensory activity erroneously predicted) and 4.8 % missed predictions of subsequent sensor activity. For those cases where a readout pool correctly predicted that a sensor will become active, the mean of the time period of its activation was predicted with an average error of 10.1 ms. For the 50 ms prediction there were for 300 novel test inputs 16.5 % false alarms, 4.6 % missed predictions of sensory activations, and an average 14.5 ms error in the prediction of the mean of the time interval of sensory activity.

One should keep in mind that movement prediction is actually a computationally quite difficult task, especially for a moving ball, since it requires context-dependent integration of information from past inputs over time and space. This computational problem is often referred to as the "aperture problem": from the perspective of a single sensor that is currently partially activated because the moving ball is covering part of its associated unit square (i.e., its "receptive field") it is impossible to predict whether this sensor will become more or less activated at the next movement (see [18]). In order to decide that question, one has to know whether the center of the ball is moving towards its receptive field, or is just passing it tangentially. To predict whether a sensor that is currently not activated will be activated 25 or 50 ms later, poses an even more difficult problem that requires not only information about the direction of the moving object, but also about its speed and

shape. Since there exists in this experiment no preprocessor that extracts these features, which are vital for a successful prediction, each readout pool that carries out prediction for a particular sensor has to extract on its own these relevant pieces of information from the raw and unfiltered information about the recent history of sensor activities, which are still "reverberating" in the recurrent circuit.

28 further readout pools were trained in a similar unsupervised manner (with 1000 training examples) to predict *where* the moving object is going to leave the sensor field. More precisely, they predict which of the 28 sensors on the perimeter are going to be activated by more than 50 % when the moving object leaves the 8 x 8 sensor field. This requires a prediction for a context-dependent time span into the future that varies by 66 % between instances of the task, due to the varying speeds of moving objects. We arranged that this prediction had to be made while the object crossed the central region of the 8 x 8 field, hence at a time when the current position of the moving object provided hardly any information about the location where it will leave the field, since all movements go through the mid area of the field. Therefore the tasks of these 28 readout neurons require the computation of the direction of movement of the object, and hence a computationally difficult disambiguation of the current sensory input. We refer to the discussion of the disambiguation problem of sequence prediction in [14] and [1]. The latter article discusses difficulties of disambiguation of movement prediction that arise already if one has just pointwise objects moving at a fixed speed, and just 2 of their possible trajectories cross. Obviously the disambiguation problem is substantially more severe in our case, since a virtually unlimited number of trajectories (see Figure 7) of different extended objects, moving at different speeds, crosses in the mid area of the sensor field. The disambiguation is provided in our case simply through the "context" established inside the recurrent circuit through the traces (or "reverberations") left by preceding sensor activations. Figure 6 shows in panel a a typical current position of the moving ball, as well as the sensors on the perimeter that are going to be active by ≥ 50 % when the object will finally leave the sensory field. In panel b the predictions of the corresponding 28 readout neurons (at the time when the object crosses the mid-area of the sensory field) is also indicated (striped squares). The prediction performance of these 28 readout neurons was evaluated as follows. We considered for each movement the line from that point on the opposite part of the perimeter, where the center of the ball had entered the sensory field, to the midpoint of the group of those sensors on the perimeter that were activated when the ball left the sensory field (dashed line). We compared this line with the line that started at the same point, but went to the midpoint of those sensor positions which were predicted by the 28 readout neurons to be activated when the ball left the sensory field (solid line). The angle between these two lines had an average value of 4.9 degrees for 100 randomly drawn novel test movements of the ball (each with an independently drawn trajectory and speed). Another readout pool was independently trained in a supervised manner to classify the moving object (ball or bar). It had an error of 0 % on 300 test examples of moving objects. The other readout pool that was trained in a supervised manner to estimate the speed of the moving bars and balls, which ranged from 30 to 50 units per second, made an average error of 1.48 units per second on 300 test examples. This shows that the same recurrent circuit that provides the input for the movement prediction can be used simultaneously by a basically unlimited

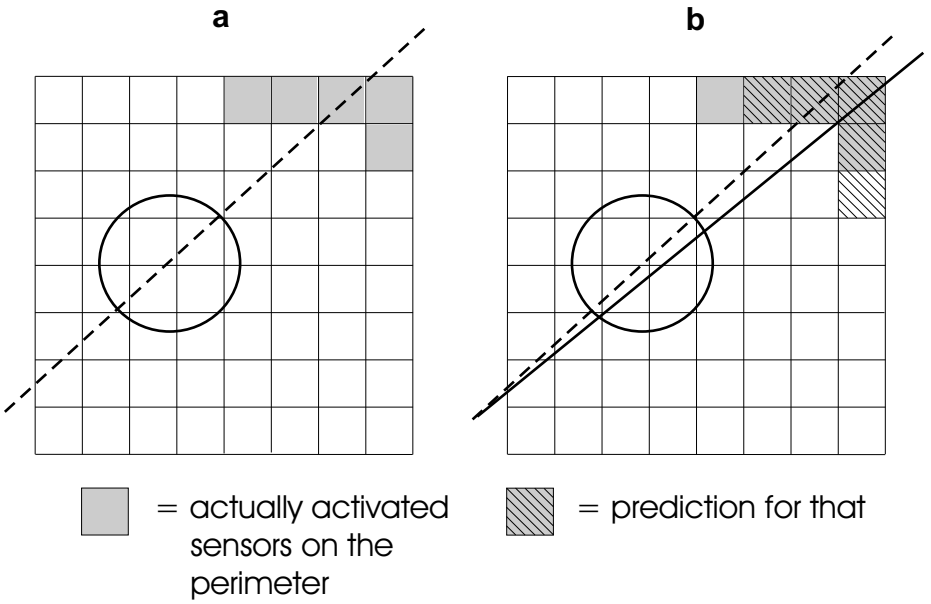

Figure 8: Computation of movement direction. Dashed line is the trajectory of a moving ball. Sensors on the perimeter that will be activated by ≥ 50 % when the moving ball leaves the sensor field are marked in panel a. Sensors marked by stripes in panel b indicate a typical prediction of sensors on the perimeter that are going to be activated by ≥ 50 %, when the ball will leave the sensor field (time span into the future varies for this prediction between 100 and 150 ms, depending on the speed and angle of the object movement). Solid line in panel b represents the estimated direction of ball movement resulting from this prediction (its right end point is the average of sensors positions on the perimeter that are predicted to become ≥ 50 % activated). The angle between the dashed and solid line (average value 4.9 for test movements) is the error of this particular computation of movement direction by the simulated neural circuit.

number of other readouts, that are trained to extract completely different information about the visual input. We refer to [13] and [15] for details. Currently similar methods are applied to real-time processing of input from infra-red sensors of a mobile robot.

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 complementary computational perspectives:

1. temporal integration of continuously entering information ("analog fading memory")

2. creation of diverse nonlinear combinations of components of such information to enhance the capabilities of linear readouts to extract nonlinear combinations of pieces of information for diverse tasks ("kernel function").

The results reported in the preceding section demonstrate implicitly that both of these computational functions are supported by generic cortical microcircuit models, since all of the benchmark problems that we discussed require temporal integration of information. Furthermore, for all of these computational tasks it sufficed to train *linear* readouts to transform liquid states into target outputs (although the target function to be computed was highly nonlinear in the inputs). In this section we provide a more quantitative analysis of these two complementary computational functions.

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 circuit 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 Figure 9). 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 Figure 9). Due to random fluctuation the actual sum of firing rates $r_{measured}(t)$ (plotted as dashed line in the first row) represented by these 4 input spike trains varies around the intended firing rate $r(t)$. $r_{measured}(t)$ is calculated as the average firing frequency in the interval $[t - 30\,\text{ms}, t]$. Third row of Figure 9 shows that the autocorrelation of both $r(t)$ and $r_{measured}(t)$ 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 $r_{measured}(t)$, $r_{measured}(t{-}30ms)$, $r_{measured}(t{-}60ms)$, $r_{measured}(t{-}90ms)$, etc. Figure 10a shows (on test data not used for training) the correlation coefficients achieved between the target value and actual output value for 8 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). Figure 10b 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. Figure 10c 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.

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 8 randomly generated Poisson spike trains over 250 ms, or equivalently 2 Poisson spike trains over 1000 ms partitioned into 4 segments each (see top of

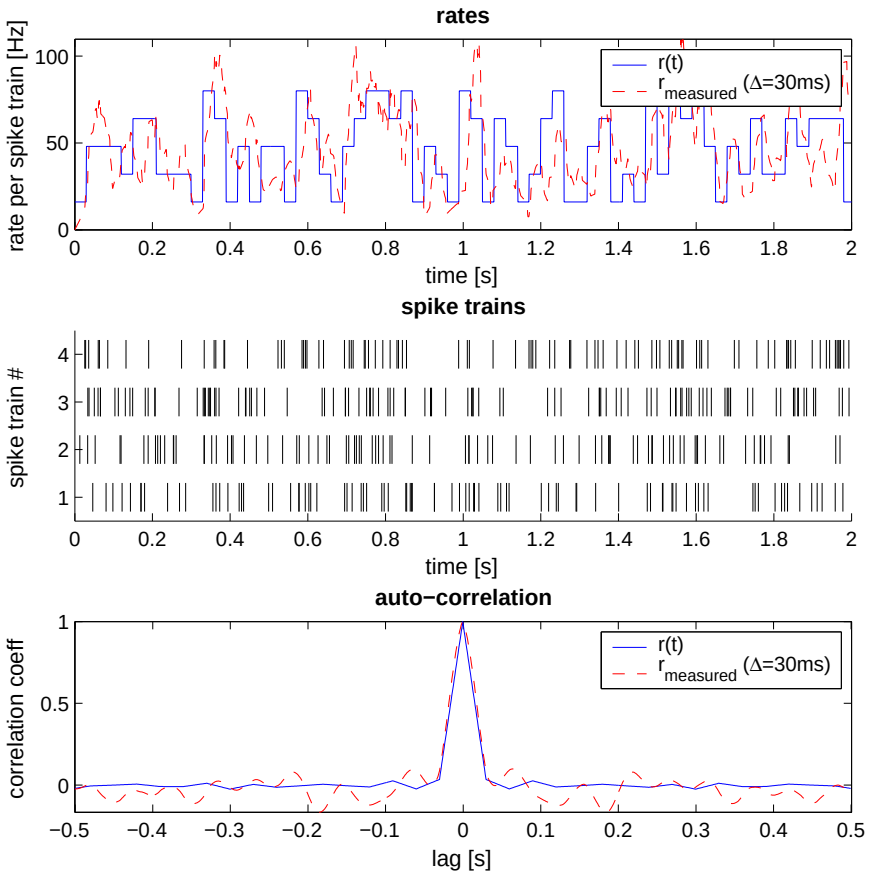

Figure 9: Input distribution used to determine the "memory curves" for firing rates. Input spike trains (second row) are generated as Poisson spike trains with a randomly drawn rate $r(t)$. The rate $r(t)$ is chosen every 30 ms from the uniform distribution over the interval from 0 to 80 Hz (first row, sold line). Due to random fluctuation the actual sum of firing rates $r_{measured}(t)$ (first row, dashed line) represented by these 4 input spike trains varies around the intended firing rate $r(t)$. $r_{measured}(t)$ is calculated as the average firing frequency in the interval $[t-30\,\text{ms}, t]$. The third row shows that the autocorrelation of both $r(t)$ and $r_{measured}(t)$ vanishes after 30 ms.

Figure 11), 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 Figure 11). A typical spike train generated in this way is shown in the middle of Figure 11. Because of the noisy dislocation of spikes it was impossible 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 retrospectively. The performance of 4 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 Figure 12 (thin line).

For comparison the memory curve for the recall of firing rates for the same

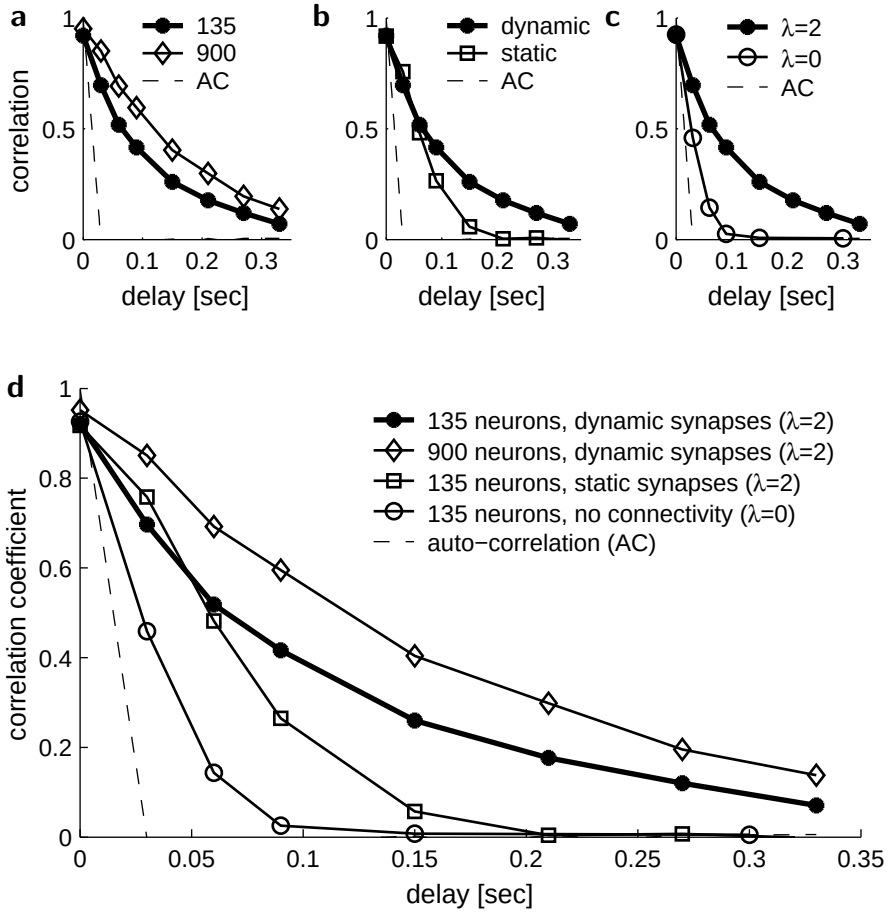

Figure 10: Memory curves for firing rates in a generic neural microcircuit model. **a)** Performance improves with circuit size. **b)** Dynamic synapses are essential for longer recall. **c)** Hidden neurons in a recurrent circuit improve recall performance (in the control case $\lambda = 0$ the readout 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). **d)** All curves from panels a to c in one diagram for better comparison. In each panel the bold solid line is for a generic neural microcircuit model (discussed in section 3) consisting of 135 neurons with sparse local connectivity ($\lambda = 2$) employing dynamic synapses. 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.

temporal segments (i.e., for inputs generated as for Figure 10, but with each randomly chosen target firing rate $r(t)$ held constant for 250 instead of 30 ms) is plotted as thin line in Figure 12, both for the same generic microcircuit model consisting of 135 neurons. Figure 12 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

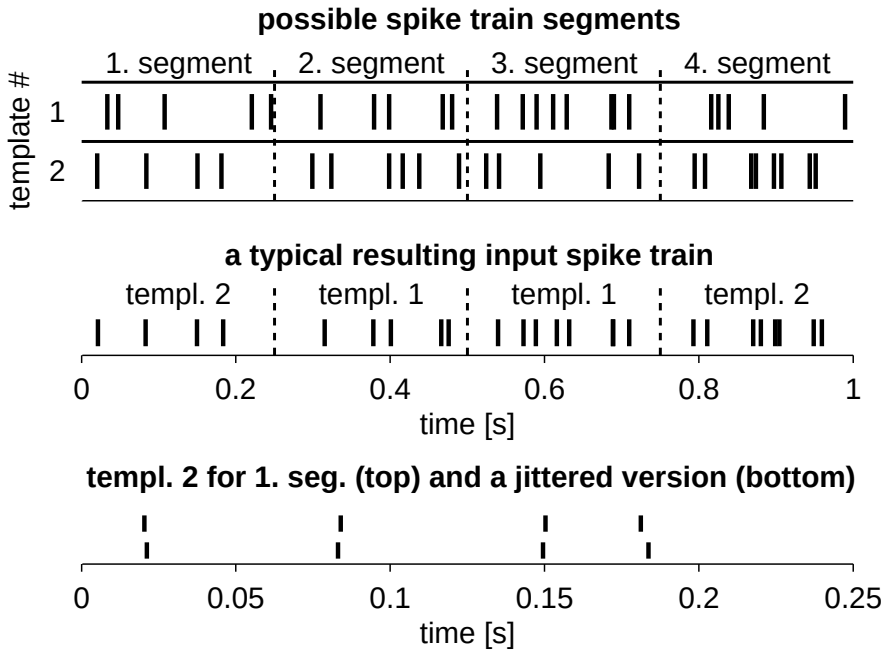

Figure 11: Evaluating the fading memory of a generic neural microcircuit for spike patterns. In this classification task all spike trains are of length 1000 ms and consist of 4 segments of length 250 ms each. For each segment 2 templates were generated randomly (Poisson spike train with a frequency of 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 the jitter with an SD of 4 ms). The task is to output with 4 different readouts at time $t = 1000$ ms for each of the preceding 4 input segments the number of the template from which the corresponding segment of the input was generated.

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 [12].

6.2 Kernel Function of Neural Microcircuit Models

It is well-known (see [21],[25], [23]) that the power of linear readouts can be boosted by two types of preprocessing:

- computation of a large number of nonlinear combinations of input components and features

- projection of the input into a very high dimensional space

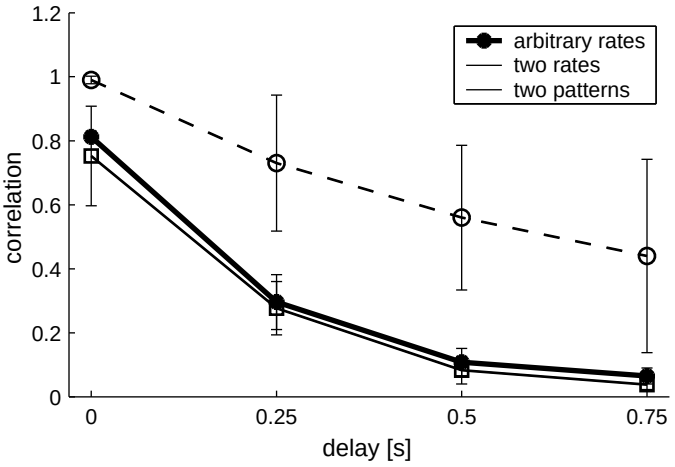

Figure 12: Memory curves for spike patterns and firing rates. Dashed line: correlation of trained linear readouts with the number of the templates used for generating the last input segment, and the segments that had ended 250 ms, 500 ms, and 750 ms ago (for the inputs discussed in Figure 11). Solid 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 4 time segments, which had to be recalled by the readouts, assumed of course a wider range.

In machine learning both preprocessing steps are carried out simultaneously by a so-called kernel, that uses a mathematical trick 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 (a typical column consists of roughly 100,000 neurons). This implementation makes use of the fact that the precise mathematical formulas by which these nonlinear 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 nonlinear combinations of input fragments can be seen from the fact that the target output values for the tasks considered in Figures 2, 4, 6, 8 are nonlinear 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 Stefan Häusler [6]. 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

[11]This is exactly the task of the second readout in the spike pattern classification task discussed in Figures 11 and 12.

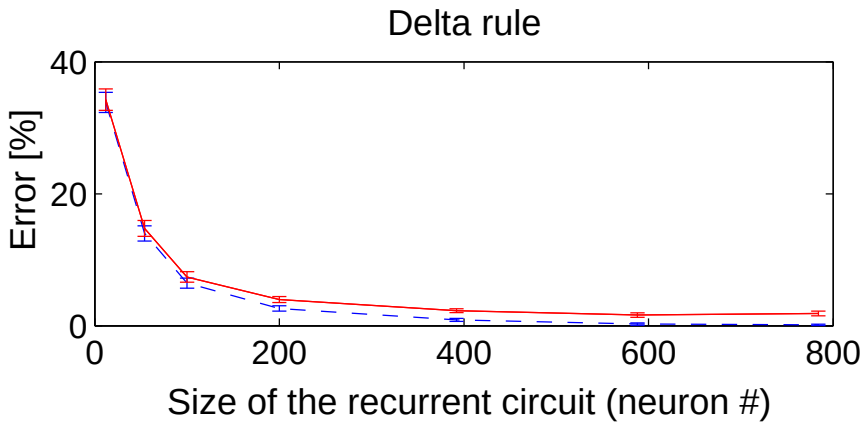

Figure 13: The performance of a trained readout (perceptron trained by the Δ-rule) for microcircuit models of different sizes, but each time for the same input injected into the microcircuit and the same classification task for the readout. The error decreases with growing circuit size, both on the training data (dashed line) and on new test data (solid line) generated by the same distribution.

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 reported in Figure 13 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.

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 Thomas Natschläger (with contributions by Christian Naeger), see [20]. 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 chapter. The object oriented style of the software makes it easy to change the microcircuit model or the computational tasks used for these tests.

8 Discussion

We have presented a conceptual framework for analyzing computations in generic neural microcircuit models that satisfies the biological constraints

[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.

listed in section 1. Thus one can now take computer models of neural microcircuits, that can be as realistic as one wants to, and use them not just for demonstrating dynamic effects such as synchronization or oscillations, but to really carry out demanding computations with these models. The somewhat surprising result is that the inherent dynamics of cortical microcircuit models, which appears to be virtually impossible to understand in detail for a human observer, 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 chapter 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.

Obviously this alternative conceptual framework relativizes some basic concepts of computational neuroscience such as receptive fields, neural coding and binding, or rather places them into a new context of computational organization. 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 chapter 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 high dimensional nonlinear kernels to facilitate linear readout. 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 nonlinear 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. Some evidence for temporal integration in cortical microcircuits has already been provided through experiments that demonstrate the dependence of the current dynamics of cortical areas on their initial state at the beginning of a trial, see e.g. [2]. Apparently this initial state contains information about preceding input to that cortical area. Our theoretical approach suggests further experiments that quantify the information about earlier inputs in the current state of neural microcircuits in vivo. It also suggests to explore in detail which of this information is read out by diverse readouts and projected to other brain areas.

The computational theory outlined in this chapter differs also in another aspect from previous theoretical work in computational neuroscience: instead of constructing hypothetical neural circuits for specific (typically simplified)

computational tasks, this theory proposes to take the existing cortical circuitry "off the shelf" and examine which adaptive principles may enable them to carry out those diverse and demanding real-time computations on continuous input streams that are characteristic for the astounding computational capabilities of the cortex.

The generic microcircuit models discussed in this chapter were relatively simple insofar as they did not yet take into account more specific anatomical and neurophysiological data regarding the distribution of specific types of neurons in specific cortical layers, and known details regarding their specific connection patterns and regularization mechanisms to improve their performance (work in progress). But obviously these more detailed models can be analyzed in the same way, and it will be quite interesting to compare their computational power with that of the simpler models discussed in this chapter.

Acknowledgement: The work was partially supported by the Austrian Science Fond FWF, project # P15386.

References

[1] L. F. Abbott and K. I. Blum. Functional significance of long-term potentiation for sequence learning and prediction. *Cerebral Cortex*, 6:406–416, 1996.

[2] A. Arieli, A. Sterkin, A. Grinvald, and A. Aertsen. Dynamics of ongoing activity: explanation of the large variability in evoked cortical responses. *Science*, 273:1868–1871, 1996.

[3] P. Auer, H. Burgsteiner, and W. Maass. Reducing communication for distributed learning in neural networks. In José R. Dorronsoro, editor, *Proc. of the International Conference on Artificial Neural Networks – ICANN 2002*, volume 2415 of *Lecture Notes in Computer Science*, pages 123–128. Springer-Verlag, 2002. Online available as #127 from http://www.igi.tugraz.at/maass/publications.html.

[4] D. V. Buonomano and M. M. Merzenich. Temporal information transformed into a spatial code by a neural network with realistic properties. *Science*, 267:1028–1030, Feb. 1995.

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

[6] S. Häusler, H. Markram, and 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*, 2003. in press. Online available as # 137 from http://www.igi.tugraz.at/maass/publications.html.

[7] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1999.

[8] J. Hopfield and 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`.

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

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

[11] H. Jäger. The "echo state" approach to analyzing and training recurrent neural networks. GMD Report 148, German National Research Center for Information Technology, 2001.

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

[13] R. A. Legenstein, H. Markram, and W. Maass. Input prediction and autonomous movement analysis in recurrent circuits of spiking neurons. *Reviews in the Neurosciences (Special Issue on Neural and Artificial Computation*, 2003. in press. Online available as #140 from http://www.igi.tugraz.at/maass/publications.html.

[14] W. B. Levy. A sequence predicting CA3 is a flexible associator that learns and uses context to solve hippocampal-like tasks. *Hippocampus*, 6:579–590, 1996.

[15] W. Maass, R. A. Legenstein, and H. Markram. A new approach towards vision suggested by biologically realistic neural microcircuit models. In H. H. Buelthoff, S. W. Lee, T. A. Poggio, and C. Wallraven, editors, *Biologically Motivated Computer Vision, Proc. of the Second International Workshop, BMCV 2002, Tübingen, Germany, November 22–24, 2002*, volume 2525 of *Lecture Notes in Computer Science*, pages 282–293. Springer (Berlin), 2002. in press. Online available as #146 from http://www.igi.tugraz.at/maass/publications.html.

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

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

[18] H. A. Mallot. *Computational Vision*. MIT Press, Cambridge, MA, 2000.

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

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

[21] J. F. Rosenblatt. *Principles of Neurodynamics*. Spartan Books (New York), 1962.

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

[23] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press (Cambridge), 2002.

[24] R. I. Soare. *Recursively Enumerable Sets and Degrees: A Study of Computable Functions and Computably Generated Sets*. Springer Verlag (Berlin), 1987.

[25] V. N. Vapnik. *Statistical Learning Theory*. John Wiley (New York), 1998.