A Fresh Look at Real-Time Computation in Generic Recurrent Neural Circuits*

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

August 20, 2002

Abstract

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 stereotypical recurrent circuits of integrate-and-fire neurons in real-time. We propose a new 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.

1 Introduction

Diverse real-time information processing 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 a model that could explain the potentially universal computational capabilities of such recurrent circuits of neurons has been missing. Common models for the organization of computations, such as for example Turing machines or attractor neural networks, are not suitable since cortical 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

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

computing") or within a short time window ("real-time computing"). Furthermore biological data prove that cortical microcircuits can support several real-time computational tasks in parallel, a fact that is inconsistent with most modeling approaches. In addition the components of biological neural microcircuits, neurons and synapses, are highly diverse [3] 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, they 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 present a new 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 New Conceptual Framework for Real-Time Neural Computation

We view a computation as a process that assigns to any input from some domain D some output 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 [14]. Perhaps the most well-known computational model is the Turing machine. In this case the domain D and range R are sets consisting 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 (according to a well-established mathematical definition, see [15]) 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.

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 [2]). 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 [16]), 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 [16] and [5]). 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

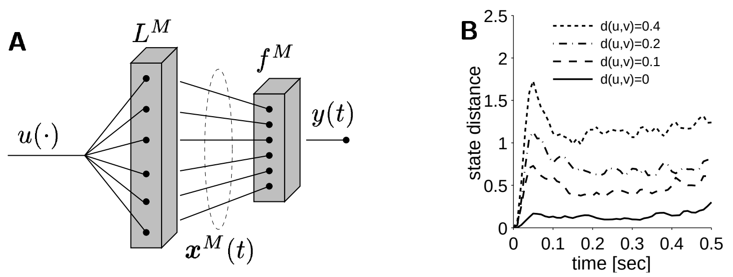

Figure 1: **A** Structure of a Liquid State Machine (LSM). **B** Separation property of a generic neural microcircuit. Plotted on the y-axis is the average value of $\|\boldsymbol{x}_u^M(t) - \boldsymbol{x}_v^M(t)\|$, where $\|\cdot\|$ denotes the Euclidean norm, and $\boldsymbol{x}_u^M(t)$, $\boldsymbol{x}_v^M(t)$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

specific tasks, whereas always *the same* 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 Fig. 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 that 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 repre-

[1]A closely related computational model was studied in [9].

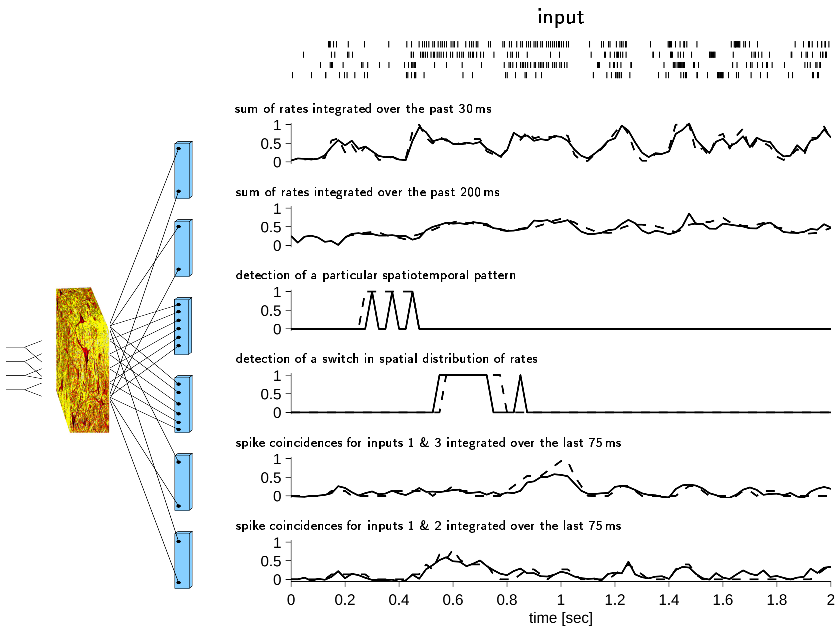

Figure 2: Multi-tasking in real-time. 4 input spike trains are injected into a generic neural microcircuit model consisting of 270 leaky integrate-and-fire neurons located on a $15 \times 6 \times 3$ grid. For each of the 6 readout modules we plotted the target function (dashed line) and the actual firing activity (solid line) on the same time scale as the input to indicate that these are 6 *real-time* computations that are supported in parallel by a single neural microcircuit model. Results shown are for a novel input that was not drawn from the same distribution as the training examples.[2]

sents 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 Fig. 2. The generic microcircuit model was drawn from the distribution discussed in section 3. In this case 6 different readout pools were trained to extract completely different types of information (see [11] for details) from the input stream $u(\cdot)$, which require different integration times stretching from 30 to 200 ms. The computations shown are for a novel input that did not occur during training[2], showing that each readout module has learned to execute its task for quite general circuit inputs. Since the neurons in the readout pools were modeled with a biologically realistic short time constant of just 30 ms, the temporally integrated information had to be contained at any instance t in the current firing state $x(t)$ of the recurrent

[2] 150 training examples were drawn randomly from a distribution that generated Poisson spike trains with a time varying firing rate. On this background activity 4 different patterns had been superimposed (always in the same order during training): rate switch to inputs 1 and 3, a burst pattern, rate switch to inputs 1 and 2, and finally a spatio temporal spike pattern. The results shown are for a test input that was not generated by the same distribution as the training examples and where the superimposed patterns never occurred in this order and at these time points for any training input (see [11] for details).

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 capacity of the primary engine of the neural computation, the circuit state $x(t)$, in terms of such simple 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 information. 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 some neural readout can be trained to recover it.

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. Parameters were chosen to fit data from microcircuits in rat somatosensory cortex (based on [3], [13] and unpublished data from the Markram Lab).[3]

The "liquid state" $x(t)$ of the recurrent circuit consisting of n neurons was modeled by an

[3]*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 $I_b = 13.5\,\text{nA}$, 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 [11] 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 [13], 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 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 $\tau_s = 3\,\text{ms}$ ($\tau_s = 6\,\text{ms}$) 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 [11] 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].

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. Its value at time t for each of the n neurons defines the liquid state $x(t)$.

Readout elements used in the simulations for Fig. 2 consisted of 51 integrate-and-fire neurons (without lateral connections). A variation of the delta learning rule (for perceptrons) was applied to scale the synapses of these readout neurons: the p-delta rule introduced in [1]. The p-delta rule is a generalization of the delta rule that trains a population of perceptrons to adopt a given population response (in terms of the number of perceptrons that are above threshold), requiring very little overhead communication. Although this online learning rule is of a biologically realistic flavor (being distributed and requiring very little overhead computation and communication), it is not claimed to be biologically realistic. The main purpose of its application was to demonstrate that the corresponding information was in fact contained in the current state $x(t)$ of the generic neural microcircuit model, and that it is accessible to very simple models for neural readouts. Usually one achieves about the same readout performance with a single readout neuron and synaptic weights computed like for support vector machines (SVMs) with linear kernels.

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 analyzing parallel real-time computing on analog input streams. Furthermore we present a theoretical result which implies that within this framework the computational unit 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 Fig. 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 $(Bu)(t) \neq (Bv)(t)$.[4] There exist completely different classes $\mathcal{B}$ of filters that satisfy

[4]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 $u(\cdot)$, $v(\cdot)$.

this point-wise separation property: $\mathcal{B} = \{\text{all delay lines}\}$, $\mathcal{B} = \{\text{all linear filters}\}$, and perhaps biologically more relevant $\mathcal{B} = \{\text{models for dynamic synapses}\}$ (see [12]).

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 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 $\mathcal{F} = \{\text{feedforward sigmoidal neural nets}\}$, and according to [1] also $\mathcal{F} = \{\text{pools of spiking neurons with analog output in space rate coding}\}$.

A rigorous mathematical theorem [11], 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 a 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 assign for each time t and state $x(t) = (Lu)(t)$ the target output $y(t)$. We refer to the online available paper [11] for details.

For physical implementations of LSMs it makes more sense to study 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. It turns out that the difference between the liquid states scales roughly proportionally to the difference between the two input histories. 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 used several tasks, among them a well-studied computational benchmark task for which data had been made publicly available [6]: the speech recognition task considered in [7] and [8].

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[5] signaling onset, peak, or offset of activity in a particular frequency band. A network was presented in [8] that could solve this task with an error[6] 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 [7].[7] 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 task 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_w which received inputs from the 135 I&F neurons in the circuit were optimized (like 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 [8] for the same task[8]. 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. In fact, the best one among the 50 randomly

[5]The network constructed in [8] required that each spike train contained at most one spike.

[6]The error (or "recognition score") S for a particular word w was defined in [8] 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 [8] and of competing networks of other researchers can be found at [6]. 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.

[7]The network constructed in [8] 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" [8]. 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.

[8]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.

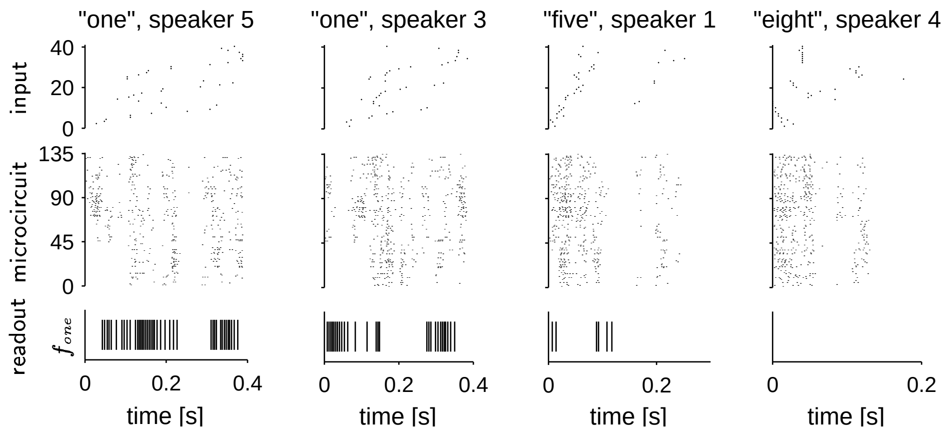

Figure 3: Application of our generic neural microcircuit model to the speech recognition from [8]. Top row: input spike patterns. Second row: spiking response of the 135 I&F neurons in the neural microcircuit model. Third now: output of an I&F neuron that was trained to fire as soon as possible when 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 them all as equivalent if they arise while the word "one" is spoken (see [11] for more material on such equivalence classes defined by readout neurons).

generated circuits achieved an error of 0.013, which is just 8.6% of the error achieved by any of the networks constructed in [8] and the associated international competition.[9]

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 [8] 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 Fig. 2 in [8]), 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. In fact, as illustrated in Fig. 3, one can even train the readout neurons quite successfully to provide provisional answers long before the input pattern has ended (thereby implementing an "anytime" algorithm).

We also compared the noise robustness of the generic neural microcircuit models with that of [8], which had been constructed to facilitate robustness with regard to linear time warping of the input pattern. Since no benchmark input date were available to calculate this noise robustness we

[9]These results are presented in order to give an idea of the performance that can be achieved with generic microcircuit models. But one should keep in mind that the network of [8] was constructed to store a spike pattern (consisting of one spike per channel), and to check whether a new input pattern resulted from a linear time warp of the stored pattern. Consequently this constructed circuit could be initialized with just one sample of each spoken word. Generic neural microcircuits have no built in bias towards which aspect of an input pattern is currently viewed as salient, and which apects are viewed as noise. Therefore we had to use 300 samples of spoken words to train the generic microcircuit, and thereby convey implicitly the information which aspect of the input was irrelevant for the classification of spoken words.

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[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 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 [8] 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 [8].

The performance of the same generic neural microcircuit model on completely different computational tasks (recall of information from preceding input segments, movement prediction and estimation of the direction of movement of extended moving objects) turned out to be also quite remarkable, see the online available reports [11] and [10]. Hence this microcircuit model appears to have quite universal capabilities for real-time computation on time-varying inputs.

6 Discussion

We have presented a new conceptual framework for analyzing computations in generic neural microcircuit models that satisfies the biological constraints listed in section 1. Thus one can now

[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 $f = 2\,\text{Hz}$, K randomly drawn from [0.5,2], φ randomly drawn from $[0, 2\pi]$ and B chosen such that $g(0) = 0$.

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 on these models. Furthermore, our new conceptual framework for analyzing computations in neural circuits not only provides a solid theoretical support for their seemingly universal capabilities for real-time computing, but also throws new light on classical concepts such as neural coding. In addition it provides new hypotheses regarding the functional role of highly recurrent neural circuits as general purpose temporal integrators (analog fading memory) and simultaneously as high dimensional nonlinear kernels to facilitate linear readout (see [4]).

Apparently neural circuits are not perfectly constructed according to a precise masterplan, but rather have emerged through a partially stochastic evolutionary and developmental process. The theoretical results discussed in this article may be viewed as first steps towards a theory of computation with such partially unknown (random) circuits.

Whereas the approach discussed in this article has primarily been developed to model the organization of information processing at the microcircuit level of neural systems, it also suggests new ways of thinking about cognitive processes in larger neural systems. Obviously many cognitive processes are not modeled well by traditional computational models such as Turing machines or inference-based algorithms from Artificial Intelligence, since these approaches are not suitable for real-time processing of complex time-varying information. The alternative model suggested by our approach is to replace time-consuming offline algorithms by anytime readouts from a suitably adapted analog fading memory. Finally, since in contrast to virtually all other computational models the circuit models that we consider have no preferred direction of information processing, they offer an ideal platform for investigating the interaction of bottom-up and top-down processing of information in neural systems.

References

[1] P. Auer, H. Burgsteiner, and W. Maass. Reducing communication for distributed learning in neural networks. In *Proc. ICANN'2002*. Springer-Verlag, 2002. Online available as #127 from http://www.igi.tugraz.at/maass/publications.html.

[2] 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.

[3] 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.

[4] S. Häusler, H. Markram, and W. Maass. Observations on low dimensional readouts from the complex high dimensional dynamics of neural microcircuits. 2002. submitted for publication.

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

[6] 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`.

[7] 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.

[8] 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.

[9] H. Jäger. The "echo state" approach to analyzing and training recurrent neural networks. 2001. submitted for publication.

[10] R. A. Legenstein, H. Markram, and W. Maass. Input prediction and autonomous movement analysis in recurrent circuits of spiking neurons. *submitted for publication*, 2002. Online available as #140 from http://www.igi.tugraz.at/maass/publications.html.

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

[12] 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.

[13] 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.

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

[15] R. I. Soare. *Recursively enumerable sets and degrees: A Study of Computable Functions and Computably Enumerable Sets.* Springer Verlag, 1987.

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