Unpublished manuscript prepared for the Dahlem Workshop on
Microcircuits : The Interface between Neurons and Global Brain Function

Theory of the Computational Function of Microcircuit Dynamics

Wolfgang Maass[+], Henry Markram*

[+] Institute for Theoretical Computer Science, Technische Universität Graz,
A-8010 Graz, Austria
maass@igi.tugraz.at
* Brain Mind Institute, Ecole Polytechnique Federale de Lausanne,
CH-1015 Lausanne, Switzerland
henry.markram@epfl.ch

ABSTRACT

We discuss models for computation in cortical microcircuits in the light of general computational theories and biological data.

Keywords: neural microcircuit, computational power, computational models, learning, canonical circuit, offline computing, online computing.

1. INTRODUCTION

The neocortex has enabled a quantum leap in the ability of mammals to adapt to a rapidly changing environment by supporting the emergence of sophisticated cognitive functions. Higher cognitive function depends critically on the ability of a system to predict future events in order to generate appropriate and intelligent responses in a rapidly changing environment. The neocortex solves this computational challenge by transforming multi-sensory information in parallel and in real-time into a multi-dimensional output using information that was processed and stored at almost any combination of time points in the past. A central challenge is therefore to understand how the neocortex is designed to solve this task and what kind of theoretical framework could explain the computational principles that it uses to solve this task.

The neocortex is characterized by precise topographic maps where information from the senses are mapped onto different regions of the neocortex and, different regions of the neocortex are mapped back onto sub-cortical brain regions and onto effector organs that drive the body. Different regions of the neocortex are also intricately mapped onto each other to fuse all the modalities into a coherent perception. While these topographical maps onto, between and from the neocortex specify precisely the primary function of each cortical region, they also mean that all functions of the neocortex are interlinked. It is the manner in which these functions are interlinked that forms the cognitive architectures which allow the neocortex to construct integrated high dimensional sensory-motor models to simulate and predict future events.

The principle of precise topographic mapping combined with massive recurrent links is applied in the neocortex not only between brain areas, but down to the most detailed level of the circuit design. The neocortex is for example, arranged into vertical layers, each with unique afferent and efferent connectivity, which allows different brain regions to map onto different layers of the same column of neocortex. The non-specific thalamus is mapped onto layer I,

association regions onto layers II and III, specific thalamus onto layer IV, higher association areas and specific thalamus are mapped onto layer V, and multiple brain regions as well as specific thalamic nuclei are mapped onto layer VI. In terms of output, layer II/II neurons provide the main output to association cortical regions, layer V provides the main output to sub-cortical regions and contra-lateral hemispheres and layer VI provides the main output to thalamus and brain regions specializing to process different modalities. While each layer is specialized to primarily process specific input and generate a specific output, the layers are also highly interconnected [Thomson and Morris, 2002], indicating that information from multiple brain regions are interlinked in columns of neurons. It is the manner in which these layers are mapped onto each other that govern how sensory input, ongoing activity, and output are coordinated to create a coherent perception and appropriate response.

Within neocortical layers, neurons are also intricately mapped onto each other, where the anatomical and physiological properties as well as probabilities of connections between neurons are unique for each type of pre and postsynaptic neuron combination [Gupta et al., 2000]. The computational advantage of such a specific design of this recurrent microcircuit must be extremely powerful compared to a mere random recurrent network because this intricate design is duplicated and applied throughout the neocortex and to all mammalian species. Remarkable stereotypy exists in terms of morphology of cells (morphological stereotypy), electrical behavior of cells (electrophysiological stereotypy), positioning of cells (spatial stereotypy), patterning of the anatomical and physiological properties of synaptic connections between neighboring cells (local synaptic stereotypy), and in terms of long range afferent and efferent connectivities (distal connection stereotypy) [Silberberg et al., 2002]. However, while there clearly exists a unique design with a considerable degree of stereotypy across different regions, ages and species, variations are found that seem to be adaptations to the specific requirements of the brain region or species.

The stereotypy of the neocortical microcircuit led many early anatomists to propose that neocortical neurons could be anatomically grouped into columns, roughly the diameter of the spread of the basal axonal and dendritic arbors of pyramidal neurons (about 500μm) [see Silberberg et al., 2002] Indeed, more than 80% of the synapses of neocortical interneurons are devoted to interconnecting neurons within a diameter of 500μm. Experiments in the late 50's and early 60's further indicated that the neocortical sheet may also be regarded functionally as being composed of small repeating columns of several thousand cells of about 500μm in diameter. Such functional modules have now been observed in many neocortical areas and in many species. However, more recent experiments revealed multiple overlying functional columns, which indicates that the notion of a cortical column as a set anatomical entity is not correct. Functional modules are overlaid [Swindale et al., 2000] such that a group of neurons collaborating to perform one operation may not necessarily collaborate to perform a different operation. The dense local connectivity within a diameter of 500μm, embedded in a continuous sheet of interconnected neurons, therefore seems to impart the neocortex with the remarkable ability of allowing multiple functional columns to form and overlay dynamically on the same cortical sheet. The properties of the local as well as afferent and efferent connectivity and the microcircuit continuity enable a functional neocortical column to potentially form at any point in the neocortex.

In summary, the neocortex is composed of heterogeneous microcircuits that differ with age, across brain regions and across species. Nevertheless, many properties of the microcircuit are stereotypical, suggesting that neocortical microcircuits are merely variations of a common microcircuit template. Such a template could sub-serve the impressive computational capability of the neocortex, and diversification could allow microcircuits to meet the specific requirements demanded by different neocortical areas, environmental conditions and species adaptations. The theoretical question is how can such a generic template display sufficiently powerful and versatile information processing capabilities.

Some basic concepts that are useful for a more rigorous discussion of this question in the context of computation theory are introduced in section 2. In section 3 we will review a few existing hypotheses regarding the computational function of microcircuit dynamics in the light of various computational theories. Consequences for future research will be discussed in section 4.

2. BASIC CONCEPTS FROM THE THEORY OF COMPUTING

We will define in this section a few basic concepts that are useful for highlighting characteristic differences among competing models for cortical computation. We refer to [Savage, 1998] for details regarding general computational models. Details on the computational power of artificial neural networks can be found in [Sima and Orponen, 2003]. An elementary introduction to artificial neural networks is given in [Tsodyks, 2002].

2.1 Input/output conventions

If all of the input for a computation is available when the computation begins, one talks about *batch input*. In contrast *online computations* receive a possibly "never-ending" stream of inputs and need to be able to integrate information from previously received input segments with information from the current input segment. If the output of a computation has to be delivered by a specific deadline (e.g. within 100 ms after a specific input segment has arrived) one calls this *real-time computing*. If the computational machinery can even be prompted at any time to provide its current best guess of a suitable output, without a prespecified schedule for output demands as in real-time computing, then it implements an *anytime algorithm*. If there exists no deadline for delivering the output of a computation one calls this *offline computing*[1].

2.2 Programs, learning, and the difference between computing and learning

The *program* of a computational model specifies which algorithm is applied to what type of input, and where and when the output of a computation is provided. One usually contrasts fully programmed computational models (such as Turing machines or cellular automata) with models that are able to learn (such as multi-layer perceptrons with backprop). One usually says that a machine (or organism) *learns* when information contained in current inputs affects the way in which it processes future inputs. But of course the way in which current inputs can change future processing has to be pre-specified by a *learning algorithm*. Thus a computational device that learns also requires a program, but a program on a higher level that specifies the organization of learning for the system, i.e., how the handling of future inputs is affected by current or preceding inputs. Even "self-organizing" computational models require such higher level program (which might of course be encoded through distributed local operating rules).

---

[1] Offline computing in combination with batch input is the most common mode considered in computational complexity theory, where upper bounds are provided for the number of computation steps that a specific algorithm needs on a specific computational model for completing its computation for any input of length n. Such upper bounds are usually expressed by terms that may for example have the form $c_1 \cdot n^d + c_2$ with arbitrary constants c_1, c_2 (abbreviated $O(n^d)$) that absorb all finite size effects into the constants c_1 and c_2. In the more general parts of computational complexity theory one focuses on the exponent d. For example the best-known complexity class P consists of all families of computational problems for which there exists a (deterministic) algorithm and some finite exponent d so that any instance of this computational problem of any length n can be solved on a Turing machine within time $O(n^d)$ by this algorithm. Although this complexity class P is frequently viewed as a characterization of the universe of computationally solvable problems, one has to be careful in applications of these asymptotic concepts to scenarios such as computations in the brain, where it is clear that only inputs of length $n \leq n_0$ (for some specific finite bound n_0) can occur, and finite size effects that determine for example the size of the previously mentioned constants c_1 and c_2 may become quite relevant. It should also be mentioned that the class P as well as most other concepts from classical computational complexity theory are only meaningful for offline computations on batch input.

The preceding definitions are only meaningful for computations on batch inputs. The analysis of learning machines (or organisms that learn) becomes conceptually more difficult when one looks at online computing, since there one cannot talk about "current" and "future" inputs, one just has one (virtually) endless input stream. One could of course talk about current and future segments of this input stream, but even very simple devices for online computing such as linear filters or finite automata have the property that current input segments will influence the way in which future input segments will be processed, without any "learning" or "adaptation" being involved. Phrased differently, for any model for online computing that integrates information from several preceding input segments for its current output (i.e., for any model which is capable of *temporal integration* of information) it becomes difficult or even impossible to conceptually distinguish computing from learning. The "difference" becomes usually a matter of perspective, where for example online computations that take place on larger time scales, or which involve higher levels of computational organization, might be referred to as learning.

2.3 Internal states and transitions between states

The internal state of a computational model (or of an autonomously learning system) at time t should contain all information that would have to be stored if one interrupts its processing at time t for a while and then wants to restart it and continue its operation at a later time $t+\Delta$ based on the information contained in this stored information – as if there had been no interruption. Such internal states can be digital or analog, and can form finite or infinitely large sets of (possible) internal states. A finite automaton (or finite state machine) is characterized by the fact that it only has a finite state set, whereas a Turing machine (which is basically just a finite automaton together with a read/write tape) has an infinite set of digital states, since also its current tape inscription (whose length is finite at any given time t, but may grow with t) is part of its current internal state.

Transitions between internal states can be deterministic or stochastic. In the deterministic case a program determines transitions between internal states (in dependence of the current state and the current input). In the case of stochastic computations a program specifies the probability that a particular state s is assumed at time $t+1$, given the state assumed at time t and the current input.

For computational models that work in continuous time one commonly writes the program in the form of a differential equation. The resulting computational models are referred to as *dynamical systems*.

2.4 Computational goals

Computational goals may be very specific, e.g. to multiply two numbers, or more general, such as predicting future inputs, or surviving for a long time in a complex real environment with limited resources. Very specific computational goals like in the first example are characteristic for our current use of digital computers, whereas more general computational goals are potentially more characteristic for the brain.

2.5 Examples of computational models

Turing machines are computational models that are particularly suited for studying deterministic offline computations on digital batch input in discrete time with a number of internal states that is finite at any time t, but may grow unboundedly with t. Turing machines are *universal* in the sense that one has not been able to come up with any digital computation that cannot be carried out by a Turing machine.

Digital or analog feedforward circuits (e.g. feedforward Boolean circuits or feedforward neural nets) constitute another class of standard models for offline computing. But they can also be used for real-time computing (in the form of pipelining) since their computation time on any (batch-) input is limited by the depth of the circuit. But they can only be used for those

computations on sequences of inputs where no temporal integration of information from several successively presented batch inputs is required.

Finite automata are computational models with a fixed finite set of internal states that are suitable for online computations in discrete time, in fact they are perfectly suited for real-time computing. Their current state can hold information about current and past inputs, and their state transitions can be deterministic or stochastic. Cellular automata are ensembles of infinitely many identical copies of some finite automaton that are located on the nodes of some infinite graph where every node has the same finite number of neighbors (e.g. a 2-dimensional grid). At every discrete time step each automaton changes its state and determines the output to its direct neighbors in dependence of the inputs that it has received from its direct neighbors at the end of the preceding time step.[2] The input to a cellular automaton is commonly encoded in the initial states of the individual finite automata. It is well known that every Turing machine (hence any currently existing digital computer) can be simulated by some cellular automaton.[3]

Artificial neural networks are also usually considered only with discrete time, but with analog internal states (so that even a single internal state may contain infinitely many bits of information). Both deterministic and stochastic state transitions are considered. Feedforward neural nets are primarily used for offline computing since they cannot integrate information from successive inputs, but recurrent neural nets are suitable for offline and online computations. Although the learning capability of artificial neural networks is viewed as one of their essential properties, the organization of learning for neural networks is usually not controlled by the network itself, but by an external supervisor.[4] From that perspective neural network models are far away from being autonomously learning systems.

Genetic (or evolutionary) algorithms are programs for computational models with stochastic state transitions in discrete time whose computational goal is the generation of formal objects ("agents") that have high "fitness" according to a fitness-function that is part of the program.

3. OPTIONS FOR UNDERSTANDING THE COMPUTATIONAL FUNCTION OF MICROCIRCUIT DYNAMICS

Obviously the computational function of the brain is to enable an autonomous system to survive in the real world. Often the computational function of the brain is seen more narrowly, separating conceptually computing from *learning*. But we have seen in the preceding section that such distinction is not very meaningful for analyzing online computations on input streams (only for isolated computations on batch inputs)[5]. Hence one needs to understand learning as an integral part of the computational function of neural microcircuits.

We will review in the subsequent subsections three categories of computational models for neural microcircuits that differ with regard to the type of computation which is supported by these models (offline versus online computing) and with regard to the circuit structures on which they focus (feedforward versus recurrent circuits). Apart from that the models that we review also differ with regard to their underlying assumption about the computational specialization of

[2] The well-known "Game of Life" is an example for such cellular automaton.

[3] To be precise, this holds only for a cellular automaton consisting of infinitely many cells. A cellular automaton consisting of just finitely many cells can only simulate those Turing machine computations that can be carried out within a corresponding space bound. Basic results from computational complexity theory imply that the computational power of Turing machines (and hence also of cellular automata) does not saturate at any finite space bound.

[4] For example the selection of training examples, of a learning rate, of the time points when learning is turned on and off are typically taken over by a human supervisor, and are not carried out by the "learning device" (e.g. a multi-layer perceptron) itself. In that sense a multi-layer perceptron with backprop cannot be viewed as an *autonomously learning machine.*

[5] Separate trials in neurophysiological experiments often aim at testing an organism on artificially created batch inputs, and proceed on the assumption that one can ignore the fact that from the perspective of the organism they still constitute segments of one continuous input stream.

cortical microcircuits (do there exist cortical microcircuits that carry out just one particular computation?) and with regard to the underlying assumption how the computational function is "programmed" into cortical microcircuits (is their computational function directly genetically programmed, or acquired by learning in the course of genetically programmed developmental procedures and fine-tuned by synaptic plasticity throughout adulthood?).

3.1 Microcircuits as modules that compute stereotypical basis functions for computations on batch input

One can compute any Boolean function (i.e., any function $f:\{0,1\}^m \to \{0,1\}^n$, for arbitrary $m,n \in \mathbf{N}$)[6] on a feedforward circuit consisting of units or subcircuits that compute certain stereotypical *basis functions*. For example it suffices to have subcircuits that compute an OR of 2 input bits in conjunction with subcircuits that compute a negation. It even suffices to iterate stereotypical copies of a single subcircuit, e.g. of a subcircuit that computes (x_1 AND NOT x_2). From a mathematical point of view there is nothing special about such basis functions, and there exist many different sets of basis functions that are complete in the sense that all Boolean functions can be generated by them.

For analog computations it is more meaningful to look at ways of approximating (rather than computing) arbitrary given functions $f:[-B,B]^m \to [-B,B]^n$ from real numbers into real numbers by circuits that are composed from stereotypical subcircuits that compute suitable basis functions. Again there exist many different sets of basis functions that are complete in the sense that any continuous function $f:[-B,B]^m \to [-B,B]^n$ can be approximated arbitrarily closely[7] through suitable combinations of such basis functions. Since continuous functions can be approximated arbitrarily closely by polynomials (on any bounded domain $[-B,B]^m$), it suffices for example to choose addition and multiplication of real numbers (in combination with real or rational constants) as basis functions. The universal approximation theorem from artificial neural networks states that one can also choose as basis functions sigmoidal gates applied to weighted sums of the inputs x_i, i.e. functions of the form $\sigma\left(\sum_{i=1}^{k} w_i x_i\right)$ with $\sigma(x) := 1/(1+e^{-x})$. In fact, one can use here instead of the sigmoidal function σ almost any nonlinear function $h:\mathbf{R} \to \mathbf{R}$. As an alternative it suffices to use a single application of a winner-take-all like nonlinearity in combination with subcircuits that compute just linear weighted sums [Maass, 2000]. Thus we see that also for the computation of analog functions there exists many different sets of basis functions that are complete.

A tempting hypothesis regarding the computational role of cortical microcircuits is that there exist genetically programmed stereotypical microcircuits that compute certain basis functions. Numerous ways in which circuits of neurons can potentially compute a complete set of Boolean basis functions have been proposed (see e.g. [Shepherd and Koch, 1998]). For example a single shunting synapse can in principle compute the Boolean function (x_i AND NOT x_2), which forms a complete basis. Also many possible ways in which single neurons or circuits of neurons can potentially compute basis functions for *analog* computing (e.g. addition and

---

[6] $\{0,1\}^m$ is the set of all bit strings of length m. $[-B,B]$ is the set of all real number with absolute value bounded by B.

[7] This means that for any given ε there exist such approximating function C so that $\| f(\underline{x}) - C(\underline{x}) \| \leq \varepsilon$ for any $\underline{x} \in [-B,B]^m$, where $\| f(\underline{x}) - C(\underline{x}) \|$ denotes the Euclidean distance between the points $f(\underline{x})$ and $C(\underline{x})$ in the n-dimensional space $\mathbf{R}^n$.

multiplication) have been collected in Table 1.2 of [Shepherd and Koch, 1998] (which is reproduced in this article as Table 1) and in ch. 21 of [Koch, 1999].

Biophysical mechanism	Neuronal operation	Example of computation	Time scale
Action potential initiation	Threshold, one-bit analog-to-digital converter		0.5–5 msec
Action potentials in dendritic spines	Binary OR, AND, AND-NOT gate	[a]	0.1–5 msec
Nonlinear interaction between excitatory and inhibitory synapses	Analog AND-NOT veto operation	Retinal directional selectivity[b]	2–20 msec
Spine–triadic synaptic circuit	Temporal differentiation high-pass filter	Contrast gain control in the LGN[c]	1–5 msec
Reciprocal synapses	Negative feedback	Lateral inhibition in olfactory bulb[d]	1–5 msec
Low, threshold calcium current (I_T)	Triggers oscillations	Gating of sensory information in thalamic cells[e]	5–15 Hz
NMDA receptor	AND-NOT gate	Associative LTP[f]	0.1–0.5 sec
Transient potassium current (I_A)	Temporal delay	Escape reflex circuit in *Tritonia*[g]	10–400 msec
Regulation of potassium currents (I_M, I_{AHP}) via neurotransmitter	Gain control	Spike frequency accommodation in sympathetic ganglion[h] and hippocampal pyramidal cells[i]	0.1–2 sec
Long-distance action of neurotransmitters	Routing and addressing of information	[j]	1–100 sec
Dendritic spines	Postsynaptic modification of functional connectivity	Memory storage[k]	∞

Table 1: Potential computational functions of various biophysical mechanisms in neural circuits (from p. 21 of G.M. Shepherd. 1990. *The Synaptic Organization of the Brain*. New York: Oxford University Press. Reprinted with friendly permission of Oxford University Press USA).

A possible way in which circuits of neurons could implement a sigmoidal gate has been proposed in [Maass, 1997]. In [Pouget and Sejnowski, 1997] products of a Gaussian function (e.g. of the retinal location of a cue) and a sigmoidal function (e.g. of eye position) are proposed as basis functions for sensory-motor transformations. They prove that these basis functions are complete in the sense that weighted sums of such functions can approximate any desired continuous sensory-motor transformation. It is argued that the outputs of some parietal neurons can be approximated quite well by such basis functions. But it is not known how exactly the computation of these basis functions is neurally implemented. A more general perspective of this basis function approach is given in [Salinas and Sejnowski, 2001].

Most approaches based on static basis functions do not provide good models for the biologically more realistic case of online computing on time-varying input streams. There also do not exist good models for explaining how the composition of basis functions to larger computational modules is organized or learnt by neural systems.

3.2 Microcircuits as dynamical systems whose input is encoded in the initial state

Complementary approaches towards understanding the role of stereotypical cortical microcircuits for cortical computation emphasize that these microcircuits are synaptically connected in a highly recurrent manner, not in the way of a feedforward circuit. Therefore we will focus now on recurrent circuits. But we discuss in this subsection only approaches that focus on offline computations, i.e. computations on batch input that is encoded in the initial state of the system. Turing machines fall into this category, but also cellular automata, recurrent neural networks (e.g. Hopfield nets, attractor neural networks) and other more general dynamical systems are traditionally considered in this offline computational mode.[8] In fact it appears that the majority of computational models currently considered in computational neuroscience fall into this category.

Recurrent circuits can be composed from the same basis functions as feedforward circuits (see section 3.1). But a new problem arises. In a feedforward circuit the computational control that determines which subcircuits are activated at any given moment is implicitly encoded by the underlying wiring diagram, which is in that case a directed acyclic graph. Computations in recurrent circuits that implement Turing machines, cellular automata, or recurrent neural nets in discrete time require a central clock, in conjunction with a protocol that determines which subcircuit is active at which clock tick. Although such computational organization is theoretically also possible for a circuit consisting of integrate-and-fire neurons [Maass, 1996], the biologically more realistic case is obviously that of computing in continuous time with the interaction between computing elements described by differential equations. In this way one arrives at special cases of dynamical systems. The (batch-) input is traditionally encoded in the initial state of the dynamical system, and one usually waits until it has converged to an attractor, which could be a fixed point, a limit cycle, or a "strange attractor" as considered in chaos theory. Even dynamical systems that are composed of very simple dynamic units (e.g. sigmoidal gates) can have a very complicated dynamics, and it is not clear what particular contribution should be expected from cortical microcircuits if they are viewed as implementations of such dynamic units. [Freeman, 1975] proposed to use a conceptual framework by Ketchalsky for classifying components of recurrent circuits (K0, KI, KII sets, etc.), but the nature of nonlinear recurrent circuits tends to be in the way of any meaningful decomposition into simpler components.

Synfire chains [Abeles, 1991] have been proposed as potential dynamic modules of complex recurrent circuits, that inject a particular timing structure into the recurrent circuit. This is theoretically possible since a synfire chain (in its most basic form) is a feedforward circuit whose subsets of neurons are activated in a sequential manner. But no significant computations can be carried out by a single synfire chain. It has been conjectured that interactions between several overlapping synfire chains may attain significant computational power, but this has not yet been demonstrated (without postulating an "intelligent" higher order structure). Obviously one needs to find principles by which a network of synfire chains could be autonomously created, structured, and updated. Another interesting open problem is how such network could learn to process *time-varying* input streams.

3.3 Microcircuits as generic modules for online computing in dynamical systems

We now consider models for the computational function of cortical microcircuits that allow them to carry out online computations on complex input streams (which require temporal integration of information). Finite automata are capable of carrying out such computations, although only on digital inputs in discrete time. To implement an arbitrary finite automaton it suffices to combine a feedforward Boolean circuit that computes the transition function between states with

[8] The latter models can also be used for online computing, but such less traditional uses will be discussed in the next subsection.

computational units that act as registers for storing and retrieving information. One possible neural implementation of such registers was proposed in [Douglas et al., 1995], using hysteretic effects in recurrent circuits. Altogether the main weakness of finite automata as conceptual framework for neural computation is their strongly digital flavor (discrete time and discrete states), which makes learning or adaptation (that usually involves in neural systems gradient descent in one form or another, except for genetic algorithms) less powerful in this context.

Another attractive conceptual framework for the analysis of neural computation on online input streams is provided by linear and nonlinear filters. Numerous biophysical mechanisms that implement specific linear and nonlinear filters have been identified (see Table 1, as well as [Shepherd and Koch, 1998] and [Koch, 1999]). [Marmarelis and Marmarelis, 1978] had introduced into neurophysiology a number of useful techniques for modeling the input/output behavior of black-box neural circuits by linear and nonlinear filters. A more recent account is given in [Rieke et al., 1977]. These techniques rely on Volterra series (or equivalently Wiener series) as mathematical framework for modeling online computations of neural systems on continuous input streams or spike trains as inputs. Volterra series model the output of a system at time t by a finite or infinite sum of terms of the form $\int_0^\infty \ldots \int_0^\infty h_d(\tau_1,\ldots,\tau_d) \cdot u(t-\tau_1) \cdot \ldots \cdot u(t-\tau_d)\, d\tau_1 \ldots d\tau_d$, where some integral kernel h_d is applied to products of degree d of the input stream $u(\cdot)$ at various time points $t-\tau_i$ back in the past. Usually only linear (d =1) or quadratic (d =2) filters are used for modeling specific neural systems, since too many data would be needed to fit higher order kernel functions h_d. Not all possible computations on input streams $u(\cdot)$ can be modeled by Volterra series (of any degree), since any Volterra series (with convergent integral terms) has automatically a fading memory, where features of the input streams $u(\cdot)$ at any specific time point in the past have decreasing influence on the current output at time t when t grows. In addition a Volterra series can only model outputs that depend in a smooth manner on the input stream $u(\cdot)$. Thus they can for example model spike output only in the form of smoothly varying firing rates or firing probabilities. On the other hand this mathematical framework imposes no constraint on how slowly the memory fades, and how fast the smoothly varying output changes its value.

This conceptual framework of filters, which has traditionally been used primarily for analyzing signal processing rather than computing, was recently used in [Maass et al., 2002, Maass et al., 2004b] as basis for a new approach towards understanding the computational function of microcircuit dynamics. The liquid state machine was introduced as a generalization of the model of a finite automaton to continuous time and continuous ("liquid") states (see Fig. 1).

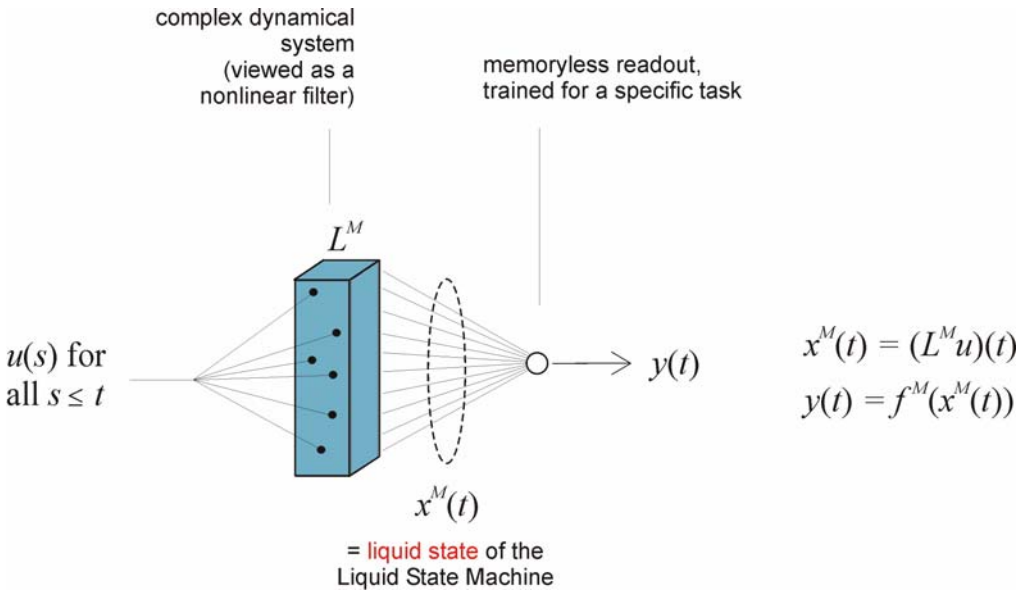

Fig. 1: Structure of a liquid state machine, which transforms input streams $u(\cdot)$ into output streams $y(\cdot)$.

In order to make this model better accessible to learning than the finite automaton, it was postulated that the liquid states (i.e., that part of the current state of the circuit that is expressed in its spiking activity and therefore "visible" for readout neurons) generated by a neural microcircuit should contain sufficient information about the recent input stream $u(\cdot)$ that other neurons ("readout neurons") just have to learn to *select* and *recombine* those parts of the information stream contained in the time-varying liquid state that are useful for their specific computational task. It is natural to apply here a variant of the basis function idea from section 3.1, and look for possible characterizations of sets of *basis filters* (that could for example be implemented by specific components of cortical microcircuits) that endow the resulting liquid states with the capacity to absorb enough information about the input stream $u(\cdot)$. A mathematical theorem (see [Maass et al., 2002]) guarantees that sufficient *diversity* of the basis filters implemented by the components of a neural microcircuit, e.g. neurons or dynamic synapses with sufficiently diverse time constants (see [Maass and Sontag, 2000]), suffices to endow the resulting liquid state machine with the capability to approximate in principle any input/output behavior that could potentially be approximated by a (finite or infinite) Volterra series. The available amount of diversity in a microcircuit can be measured indirectly via its separation property (see [Maass et al., 2002]).

Another potential computational function of microcircuit dynamics arises if one considers such liquid state machine from the perspective of an (approximately) linear readout neuron, for which it should become feasible to *learn* to select and recombine those aspects of liquid states that may be needed for specific tasks (e.g. smooth eye pursuit, or classification and prediction of dynamic visual scenes). A microcircuit can boost the capability of any linear readout by adding a certain redundancy to the information contained in its stream of liquid states, precomputing for example also nonlinear combinations of salient time-varying variables (analogously as in the special case of gain fields), see Fig. 2. Very recently in [Maass et al., 2004a] a general quantitative method has been developed to evaluate such *kernel capability* of neural microcircuits (where the notion of a *kernel* is used here in the sense of support vector machines in machine learning, it goes back

to the idea of a fixed nonlinear preprocessing proposed already in the 1950's by Rosenblatt's work on perceptrons, but applied here to a time-varying context).

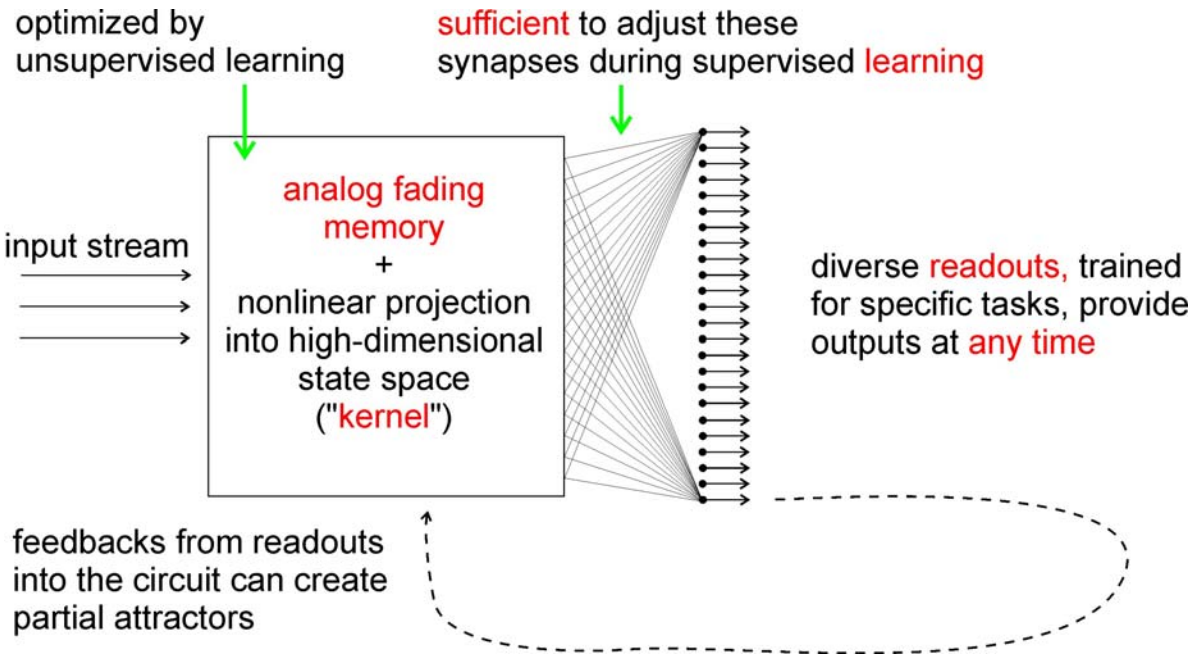

Fig. 2: A system-oriented interpretation of the computational function of microcircuit dynamics.

From this perspective a cortical microcircuit is not viewed as an implementation of a single computation, but as a more universal computational device that can simultaneously support a large number of different computations. An example is given in Fig. 3, where 7 different linear readouts from a generic neural microcircuit model consisting of 270 neurons had been trained to output at any time the result of 7 different computational operations on information provided to the circuit in the form of 4 spike trains (a sample is shown at the top of Fig. 3). After training the weights of these linear readouts had been fixed. The results shown in Fig. 3 are for new input spike trains that had never before been injected into the circuit, thereby demonstrating good generalization capability of this simple learning scheme (see [Maass et al., 2002] for details).

Several experimental studies in the group of Yves Fregnac have shown that neurons can in fact be trained via current injections (even in adult animals in vivo) to read out particular aspects of the "liquid state" represented by the current firing activity of presynaptic neurons (see e.g. [Debanne et al, 1998]). But it has remained open by which principles readout neurons can be trained autonomously within a neural system to perform such task. This is least dubious in the case of prediction learning, where the arrival of the next input could provide such current injection into a readout neuron that learns to perform such prediction task as in the experiments of [Debanne et al., 1998]. Another quite realistic scenario is the case of association learning, where a "teaching current" could be injected into a readout neuron by projection neurons from other microcircuits in cortical or subcortical structure that are involved in processing inputs from other sensory modalities, or which signal external or internal rewards.

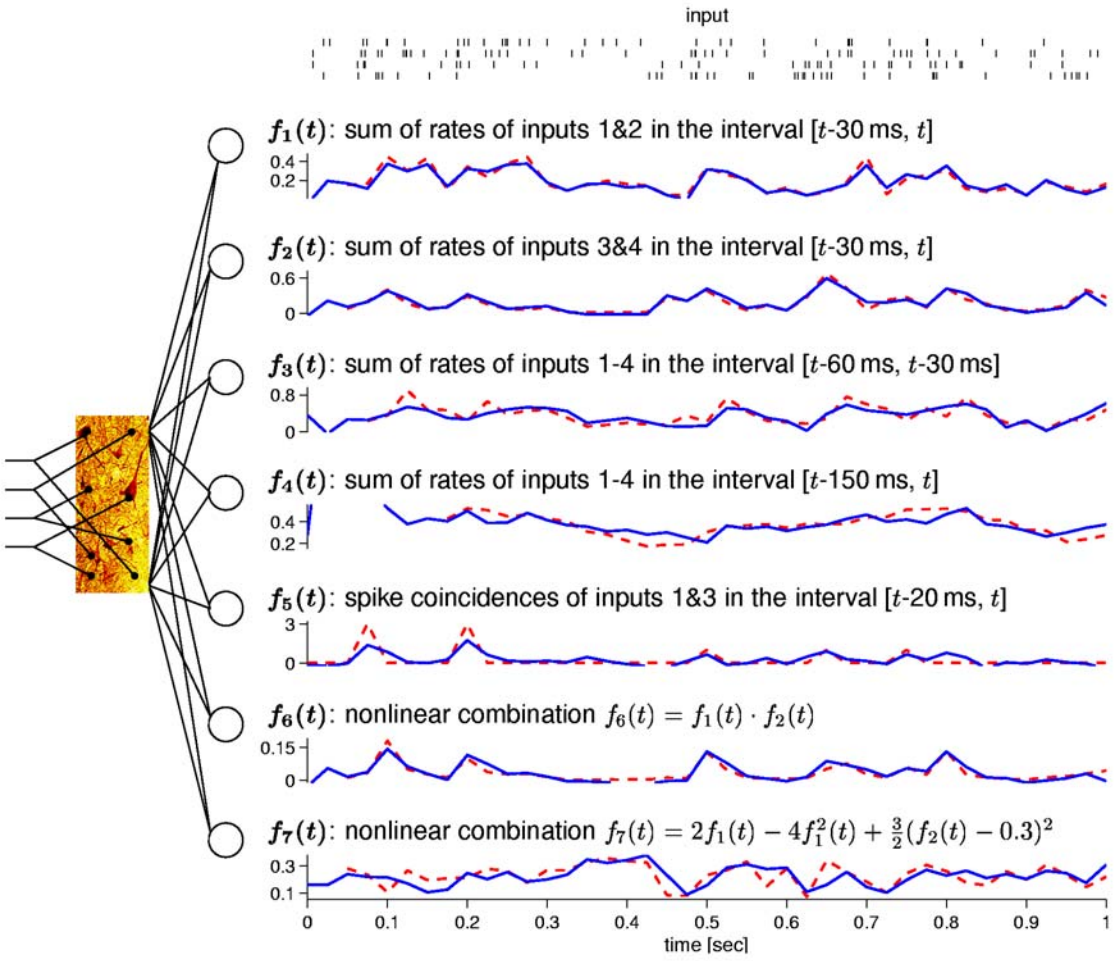

Fig. 3: Multi-tasking in real-time. Input spike trains were randomly generated in such a way that at any time t the input contained no information about input rates that were used more than 30 ms ago. Firing rates $r(t)$ were randomly drawn from the uniform distribution over [0 Hz, 80 Hz] every 30 ms, and input spike trains 1 and 2 were generated for the present 30 ms time segment as independent Poisson spike trains with this firing rate $r(t)$. This process was repeated (with independent drawings of $r(t)$ and Poisson spike trains) for each 30 ms time segment. Spike trains 3 and 4 were generated in the same way, but with independent drawings of another firing rate $\tilde{r}(t)$ every 30 ms. The results shown in this figure are for test data, that were never before shown to the circuit. Below the 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.

An attractive feature of this computational framework is that it produces a possible explanation for the *anytime computing capabilities* of neural systems, since readouts can learn to transform at any moment in time the currently provided liquid state into the best guess for a decision or a parameter value that is needed for their specific task. Another interesting aspect is that this approach is compatible with biological data regarding oscillations that are superimposed on sensory inputs [Kaske and Maass, 2004].

This approach also provides a computational explanation for the large scale architecture of the brain, where sensory inputs and internal outputs are not spread out uniformly all over the brain (see section 1). Rather, each brain area is characterized by the specific set of information streams that it receives. This feature is a prerequisite for online computing with dynamical systems, since different input streams that converge onto a single microcircuit all have an influence on its internal dynamics, thereby facilitating computations that depend on segments of all these information streams. Some other input stream that is not relevant for these computations would influence the microcircuit dynamics as well, but would represent a huge source of noise from the perspective of these computations, in particular blowing up and possibly interleaving the classes of equivalent circuit states (see [Maass et al., 2003], section 5) that a readout neuron has to learn to distinguish.

An essentially equivalent computational framework to liquid computing, echo state networks, has been developed independently in an engineering context [Jäger, 2002], and currently surpasses all other known methods for various time series prediction and adaptive nonlinear filtering tasks [Jäger and Haas, 2004]. Other recent work relates these approaches to earlier work by Chris Langton et al. on computation on the edge of chaos in dynamical systems, but applied now to anytime computing on continuous input streams rather than to offline computations on static batch input [Bertschinger and Natschläger, 2004]. It is argued there that neither the "ordered" nor the "chaotic" regime of recurrent circuits (where recurrent circuits are viewed here as special cases of dynamical systems) are well-suited for computing, but rather the regime in between (the "edge of chaos"). This has been demonstrated in [Bertschinger and Natschläger, 2004] for synchronized recurrent circuits consisting of threshold gates. Results of [Maass et al., 2004a] suggest that for more realistic models of neural microcircuits the "edge of chaos" becomes harder to conceptualize, and it is proposed there to measure the computational power and generalization capability of neural microcircuits instead by a quantitative measure of its kernel quality and a direct estimate of its generalization capability (VC-dimension).

3.4 Other approaches

There exist various other approaches for explaining or modeling biological neural computation that have not yet given rise to specific hypotheses regarding the computational role of cortical microcircuits. A number of interesting brain theories are based on information theoretic concepts, such as redundancy reduction, the information bottleneck method, and theories of optimal neural coding. Another family of models for biological neural computation uses generative models, which are based on the hypothesis that the large scale computational goal of cortical computation is to reconstruct sensory inputs, and to "explain" them as being generated by independent components or factors that can be learnt in an unsupervised manner from the statistics of the inputs. An attractive feature of this approach is that it gives rise to autonomously learning systems. However in a concrete biological context it is hard to demonstrate or even make precise the theoretically very attractive goal of reconstructing the input in terms of internally represented "hidden" sources. For example top down connections to primary sensory cortices appear to contribute to purpose-related interpretations of raw sensory input, rather than to the reconstruction of a (usually dynamically varying) sensory input, see e.g. ch. 45 of [Chalupa and Werner, 2004] for the case of visual input. Unfortunately the biologically more

realistic goal of predicting future inputs (rather than reconstructing preceding inputs) has not yet given rise to an equally attractive theoretical approach.

4. DISCUSSION

This short survey shows that there exist drastic differences regarding theories on the computational function of cortical microcircuits. To find the most appropriate computational theory for a neural microcircuit one first needs to decide whether this circuit is designed to carry out offline computations on batch inputs or online computations on input streams. One also needs to decide to what extent this circuit is genetically programmed to carry out one specific computational operation, or whether it is designed to provide numerous and diverse "neural users" with a suitably processed amalgamation of the input streams that enter this microcircuit, where the specific output delivered to any particular "neural user" is to some degree shaped by learning.

The computational function of various salient aspects of cortical microcircuits is presently still not known, especially the computational role of specific cortical layers with particular types of neurons that are connected by particular types of synapses with particular probabilities with other specific types of neurons on specific layers. Another big open problem is the organization of learning in cortical microcircuits, e.g. we need to know how the plasticity of its synapses is gated, and what other processes regulate the computational function of its neurons in dependence of their individual history and the statistics of their inputs.

REFERENCES

Abeles, M. 1991. *Corticonics: Neural Circuits of the Cerebral Cortex*, Cambridge University Press: Cambridge.

Bertschinger, N., and Natschläger, T. 2004. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7), 1413-1436.

Chalupa, L.M., and Werner, J.S. 2004. The Visual Neurosciences, Cambridge: MIT-Press.

Debanne, D., Shulz, D. E., and Fregnac, Y. 1998. Activity dependent regulation of on- and off-responses in cat visual cortical receptive fields. *Journal of Physiology*, 508:523-548.

Douglas, R.J., Koch, C., Mahowald, M., Martin, K.A., and Suarez, H.H. 1995. Recurrent excitation in neocortical circuits. *Science* 269(5226):981-985.

Freeman, W.J. 1975. *Mass Action in the Nervous System*, New York: Academic Press.

Gupta A., Wang Y., and Markram H. 2000. Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex. *Science* 287(5451):273-8.

Jäger, H. 2002. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. *GMD Report* 159, German National Research Center for Information Technology.

Jäger, H., and Haas, H. 2004. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* 304(5667):78-80.

Kaske, A., and W. Maass. 2004. A model for the interaction of oscillations and pattern generation with real-time computing in generic neural microcircuit models, *submitted for publication*.

Koch, C. 1999. *Biophysics of Computation: Information Processing in Single Neurons*. New York: Oxford University Press.

Maass, W. 1996. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8(1):1-40.
Online available as # 75 from http://www.igi.tugraz.at/maass/publications.html

Maass, W. 1997. Fast sigmoidal networks via spiking neurons. *Neural Computation*, 9:279-304.
Online available as # 82 from http://www.igi.tugraz.at/maass/publications.html

Maass, W. 2000. On the computational power of winner-take-all. *Neural Computation*, 12(11):2519-2536.
Online available as # 113 from http://www.igi.tugraz.at/maass/publications.html

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

Maass, W., Legenstein, R., and Bertschinger, N. 2005. Methods for estimating the computational power and generalization capability of neural microcircuits. In *Proc. of NIPS 2004, Advances in Neural Information Processing Systems* 17, MIT Press.
Online available as # 160 from http://www.igi.tugraz.at/maass/publications.html

Maass, W., Natschläger, T., and Markram, H. 2004. Computational models for generic cortical microcircuits. In J. Feng, editor, *Computational Neuroscience: A Comprehensive Approach*, chapter 18, 575-605. Boca Raton: Chapman & Hall/CRC.
Online available as # 149 from http://www.igi.tugraz.at/maass/publications.html

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

Marmarelis, P.Z., and Marmarelis, V.Z. 1978. *Analysis of Physiological Systems: The White-Noise Approach*. New York: Plenum Press.

Pouget, A., and Sejnowski, T.J. 1997. Spatial transformation in the parietal cortex using basis functions. *Journal of Cognitive Neuroscience* 9: 222-237.

Rieke, F., Warland, D., de Ruyter van Steveninck, R., and Bialek, W. 1997. *Spikes: Exploring the Neural Code*. Cambridge: MIT Press.

Salinas, E., and Sejnowski, T.J. 2001. Gain modulation in the central nervous system: where behavior, neurophysiology, and computation meet. *Neuroscientist* 7(5):430-440.

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

Shepherd, G.M., and Koch, C. 1998. Introduction to synaptic circuits. In: *The Synaptic Organization of the Brain*, 4th ed., G. M. Shepherd, ed., New York: Oxford University Press.

Silberberg G., Gupta A., and Markram H. 2002. Stereotypy in neocortical microcircuits. *Trends Neurosci*. 25(5):227-30.

Sima, J., and Orponen, P. 2003. General-purpose computation with neural networks: a survey of complexity theoretic results. *Neural Computation* 15(12):2727-2778.

Swindale N.V., Shoham D., Grinvald A., Bonhoeffer T., and Hubener M. 2000. Visual cortex maps are optimized for uniform coverage. *Nat. Neurosci*. 3(8):822-6.

Thomson A.M., Morris O.T. 2002. Selectivity in the inter-laminar connections made by neocortical neurons. *J. Neurocytol*. 31(3-5):239-46.

Tsodyks, M. 2002. Neural circuits: models of emergent functions. In: *International Encyclopedia of the Social & Behavioral Sciences*, Elsevier Science Ltd., http://www.iesbs.com