Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations

Wolfgang Maass+, Thomas Natschläger+ & Henry Markram*

+Institute for Theoretical Computer Science, Technische Universität Graz; A-8010 Graz, Austria
* Department of Neurobiology, The Weizmann Institute for Science, Rehovot, 76100, Israel

Wolfgang Maass & Thomas Natschlaeger
Institute for Theoretical Computer Science,
Technische Universitaet Graz
Inffeldgasse 16b, A-8010 Graz, Austria
Tel: +43 316 873-5811
Fax: +43 316 873-5805
Email: maass@igi.tu-graz.ac.at, tnatschl@igi.tu-graz.ac.at

Henry Markram
Department of Neurobiology,
The Weizmann Institute for Science,
Rehovot, 76100, Israel
Tel:+972-89343179
Fax:+972-89316573
Email:Henry.Markram@weizmann.ac.il

Address for Correspondence:
Wolfgang Maass

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 framework for neural computation that provides an alternative to previous approaches based on attractor neural networks. It is shown that the inherent transient dynamics of the high-dimensional dynamical system formed by a neural circuit may serve as a universal source of information about past stimuli, from which readout neurons can extract particular aspects needed for diverse tasks in real-time. Stable internal states are not required for giving a stable output, since transient internal states can be transformed by readout neurons into stable target outputs due to the high dimensionality of the dynamical system. Our approach is based on a rigorous computational model, the liquid state machine, that unlike Turing machines, does not require sequential transitions between discrete internal states. Like the Turing machine paradigm it allows for universal computational power under idealized conditions, but for real-time processing of time-varying input. The resulting new framework for neural computation has novel implications for the interpretation of neural coding, for the design of experiments and data-analysis in neurophysiology, and for neuromorphic engineering.

Introduction

Intricate topographically organized feed-forward pathways project rapidly changing spatio-temporal information about the environment into the neocortex. This information is processed by extremely complex but surprisingly stereotypic microcircuits that can perform a wide spectrum of tasks (Shepherd, 1988, Douglas et al, 1998, von Melchner et al., 2000). The microcircuit features that enable this seemingly universal computational power, is a mystery. One particular feature, the multiple recurrent loops that form an immensely complicated network using as many as 80% of all the synapses within a functional neocortical column, has presented an intractable problem both for computational models inspired by current artificial computing machinery (Savage, 1998), and for attractor neural network models. The difficulty to understand computations within recurrent networks comes from the fact that their dynamics takes on a life of its own when challenged with rapidly changing inputs. This is particularly true for the high dimensional dynamical system formed by a neural circuit, where each neuron and each synapse adds degrees of freedom to the dynamics of the system. The most common approach for modeling computing in recurrent neural circuits has been to try to take control of their high dimensional dynamics, or to work only with a subset of specific states such as attractor states (Hertz et al., 1991). Attractor neural networks constitute a well-studied computational model, but they need to have a very large set of attractors in order to store salient information on past inputs (for example 1024 attractors in order to store 10 bits), and they are less suitable for real-time computing because of the time required for convergence to an attractor. Such limitations on recurrent networks of neurons seems counter to their seemingly inevitable or “natural” capability to continuously absorb information as a function of time.

Here we show how salient information about present and past stimuli can easily be extracted from the high dimensional dynamic state that is supported by recurrent microcircuits, in real-time. We present the mathematical framework for a computational model that does not require convergence to stable internal states or attractors (even if they do occur), since it holds past inputs in the perturbations of a dynamical system, i.e. in the continuous trajectory of transient internal states. We show that multiple perceptron-like readout elements can be trained to perform different tasks on the same trajectories and that once trained, each readout element can perform the task on novel inputs, because each readout can learn to define its own notion of equivalence of dynamical states within the system. This most unexpected finding of “readout-assigned equivalent states of a dynamical system” explains how invariant readout is possible despite the fact that the recurrent network may never re-visit the same state, and how recurrent microcircuits can enable real-time computation without relying on stable states. The computer simulations discussed in this article show that on the basis of this new paradigm one can now train a stereotypical recurrent network of integrate-and-fire neurons to carry out basically any real-time computation on spike trains, in fact several such real-time computations in parallel.

Computing without Stable States

As an illustration for our general approach towards real-time computing consider a series of transient perturbations caused in a liquid by a sequence of external disturbances ("inputs"), such as wind, sound, or sequences of pebbles dropped into a liquid. Viewed as an attractor neural network, the liquid has only one attractor state – the resting state – and may therefore seem useless for computational purposes. However, the perturbed state of the liquid, *at any moment in time*, represents present as well as past inputs, potentially providing the information needed for an analysis of various aspects of the environment. In order for such a liquid to serve as a source of salient information about present and past stimuli without relying on stable states, the perturbations must be sensitive to saliently different inputs but non-chaotic. The manner in which perturbations are formed and maintained would vary for different types of liquids and would determine how useful the perturbations are for such “retrograde analysis”. Limitations on the computational capabilities of liquids are imposed by their time-constant for relaxation, and the strictly local interactions as well as the homogeneity of the elements of a liquid. Neural microcircuits, however, appear to be “ideal liquids” for computing on perturbations because of the large diversity of their elements, neurons and synapses (see (Gupta et al., 2000), and well as the large variety of mechanisms and time constants characterizing their interactions, involving recurrent connections on multiple spatial scales ("loops within loops").

The foundation for our analysis of computations without stable states is a rigorous computational model: the *liquid state machine*. Two macroscopic properties emerge from our theoretical analysis and computer simulations as necessary and sufficient conditions for powerful real-time computing on perturbations: a *separation property, SP*, and an *approximation property, AP*.

SP addresses the amount of separation between the trajectories of internal states of the system that are caused by two different inputs as a function of time (in the case of a physical liquid, *SP* could reflect the difference between the wave patterns resulting from different sequences of disturbances).

AP addresses the resolution and recoding capabilities of the readout mechanisms - more precisely its capability to distinguish and transform different internal states of the liquid into given target outputs (whereas *SP* depends mostly on the complexity of the liquid, *AP* depends mostly on the adaptability of the readout mechanism to the required task).

Liquid State Machines

Like the Turing machine (Savage, 1998), the model of a liquid state machine (LSM) is based on a rigorous mathematical framework that guarantees, under idealized conditions, *universal computational power*. Turing machines, however, have universal computational power for off-line computation on (static) discrete inputs, while LSMs have universal computational power for real-time computing with fading memory on analog functions in continuous time. The input function $u(\cdot)$ can be a continuous sequence of disturbances, and the target output can be some chosen function $y(\cdot)$ of time that provides a real-time analysis of this sequence. In order for a machine M to map input functions of time $u(\cdot)$ to output functions $y(\cdot)$ of time, we assume that it contains some "liquid" that generates, at every time t, an internal state $x^M(t)$, which constitutes its current response to preceding perturbations, i.e., to preceding inputs $u(s)$ for $s \le t$ (Figure 1A). In mathematical terms, the liquid of M is some operator or filter[1] L^M that maps input functions $u(\cdot)$ on functions $x^M(\cdot)$ representing the trajectory of the internal state of M:

$$x^M(t) = (L^M u)(t) \ .$$

The second component of M is a memory-less readout map f^M that transforms, at every time t, the current internal state $x^M(t)$ of the liquid into the output

$$y(t) = f^M(x^M(t)) \, .$$

Since the readout map can be memory-less[2], all information about inputs $u(s)$ from preceding time points $s \le t$ that is needed to produce the output $y(t)$ at time t has to be contained in the *current* internal state $x^M(t)$. Models for computation that have originated in computer science, store such information in stable states (for example in memory buffers). We argue, however, that this is not necessary since large computational power on functions of time can also be realized if the liquid does not have multiple stable states. In fact, it can even be achieved in a system where all memory traces are decaying. Instead of worrying about the code and location where information about past inputs is stored, and how long a stored item

[1] Functions F that map input functions of time $u(\cdot)$ on output functions $y(\cdot)$ of time are usually called operators in mathematics, but are commonly referred to as filters in engineering and neuroscience. We use the term *filter* in the following, and we write $(Fu)(t)$ for the output of the filter F at time t when F is applied to the input function $u(\cdot)$. Formally, such filter F is a map from U^n into $(\mathbf{R}^{\mathbf{R}})^k$, where $\mathbf{R}^{\mathbf{R}}$ is the set of all real-valued functions of time, $(\mathbf{R}^{\mathbf{R}})^k$ is the set of vectors consisting of k such functions of time, U is some subset of $\mathbf{R}^{\mathbf{R}}$, and U^n is the set of vectors consisting of n functions of time in U.

[2] The term "memory-less" refers to the fact that the readout map f^M is not required to retain any memory of previous states $x^M(s)$, $s < t$, of the liquid. However, in a biological context, the readout map will in general be subject to plasticity, and thereby contain at least long-term memory and possibly short-term memory. The differentiation into a memory-less readout map and a liquid that serves as a memory device, is made for conceptual clarification, but is not essential to the model.

can be retrieved, it is enough to address the separation question: For which later time points t will any two significantly different input functions of time $u(\cdot)$ and $v(\cdot)$ cause significantly different internal states $x_u^M(t)$ and $x_v^M(t)$ of the system. Good separation capability, in combination with an adequate readout map f^M, allows us to discard the requirement of storing bits "until further notice" in stable states of the computational system.

Universal Computational Power on Time Varying Inputs

One may argue that a class of machines has *universal power for computations with fading memory on functions of time* if any filter F, i.e., any map from functions of time $u(\cdot)$ to functions of time $y(\cdot)$, that is time invariant[3] and has fading memory[4], can be approximated by machines from this class, to any degree of precision. Arguably, these filters F include all maps from input functions of time to output functions of time that a behaving organism might want to compute.

A mathematical theorem (see Appendix A) guarantees that LSMs have this universal computational power regardless of specific structure or implementation, provided that two abstract properties are met: the class of basis filters from which the liquid filters L^M are composed satisfies the point-wise separation property and the class of functions from which the readout maps f^M are drawn, satisfies the approximation property. This theorem implies that there are no serious a priori limits for the computational power of LSMs on continuous functions of time, and thereby provides a theoretical foundation for the new approach towards modeling neural information processing. The mathematical theory can also be extended to cover computation on spike trains (discrete events in continuous time) as inputs. Here the ith

[3] A filter F is called *time invariant* if any temporal shift of the input function $u(\cdot)$ by some amount t_0 causes a temporal shift of the output function $y = Fu$ by the same amount t_0, i.e., $(Fu^{t_0})(t) = (Fu)(t + t_0)$ for all $t, t_0 \in \mathbf{R}$, where $u^{t_0}(t) := u(t + t_0)$. Note that if U is closed under temporal shifts, then a time invariant filter $F : U^n \to (\mathbf{R}^{\mathbf{R}})^k$ can be identified uniquely by the values $y(0) = (Fu)(0)$ of its output functions $y(\cdot)$ at time 0.

[4] *Fading memory* (see ref [19]) is a continuity property of filters F that demands that for any input function $u(\cdot) \in U^n$ the output $(Fu)(0)$ can be approximated by the outputs $(Fv)(0)$ for any other input functions $v(\cdot) \in U^n$ that approximate $u(\cdot)$ on a sufficiently long time interval $[-T, 0]$. Formally one defines that $F : U^n \to (\mathbf{R}^{\mathbf{R}})^k$ has fading memory if for every $u \in U^n$ and every $e > 0$ there exist $d > 0$ and $T > 0$ so that $\|(Fv)(0) - (Fu)(0)\| < e$ for all $v \in U^n$ with $\|u(t) - v(t)\| < d$ for all $t \in [-T, 0]$. Informally a filter F has fading memory if the most significant bits of its current output value $(Fu)(0)$ depend just on the most significant bits of the values of its input function $u(\cdot)$ in some finite time interval $[-T, 0]$ going back into the past. Thus, in order to compute the most significant bits of $(Fu)(0)$ it is not necessary to know the *precise* value of the input function $u(s)$ for any time s, and it is also not necessary to know *anything* about the values of $u(\cdot)$ for more than a finite time interval back into the past. Note that a filter that has fading memory is automatically causal.

component $u_i(\cdot)$ of the input $u(\cdot)$ is a function that assumes only the values 0 and 1, with $u_i(t)=1$ if the i^{th} preceding neuron fires at time t. Thus $u_i(\cdot)$ is not a continuous function but a sequence of point events. Theorem 2 in Appendix A provides the theoretical foundation to approximate any biologically relevant computation on spike trains by LSMs.

LSM Model for Neural Microcircuits

In order to test the applicability of this conceptual framework to computation in neural microcircuits, we carried out computer simulations where a generic recurrent network of integrate-and-fire neurons was employed as the ”liquid” L^M (see Appendix B). The input to this network was via a single or several input spike trains, which diverged to inject current into 30% randomly chosen ”liquid neurons”. The amplitudes of the input synapses were chosen from a Gaussian distribution, so that each neuron in the liquid received a slightly different input (a form of topographic injection). A readout map f^{M} was implemented by another population P of integrate-and-fire neurons (referred to as "readout neurons") that received input from all the "liquid neurons", but had no lateral or recurrent connections[5]. The vector of currents injected into the readout neurons at time *t* by the firing activity of the liquid neurons, represented the state of the liquid $x^M(t)$ at time t. The current firing activity *p(t)* of the population *P*, that is the fraction of neurons in *P* firing during a time bin of 20ms, was interpreted as the analog output of f^{M}. The class of such readout maps was recently shown to satisfy the approximation property (Maass, 2000, Auer et al., 2001). The readout neurons were trained to perform a specific task by adjusting the strengths of synapses projected onto them from the liquid neurons using a perceptron-like local learning rule (Auer et al., 2001). The final learned state of the readout neurons enables them to take particular weighted sums of the current outputs of the liquid neurons and generate a response that approximates the target function. Perturbations resulting from different input streams can be separated by perceptrons (or populations of perceptrons, implemented by integrate-and-fire neurons), because by activating the recurrent neural circuitry these inputs are projected nonlinearly into a much higher dimensional space. This effect of nonlinear projections of data into high-dimensional spaces also provides the basis for one of the most successful approaches in modern machine learning, support vector machines (Vapnik, 1998).

In order to illustrate the separation of responses in the liquid, different input spike trains $u(\cdot)$ and $v(\cdot)$ were presented to the liquid with the same initial states. These two spike trains generated different trajectories $x_u^M(\cdot)$ and $x_v^M(\cdot)$ of internal states of the liquid (Figure 1B). The separation property *SP* of this liquid is represented by curves (Figure 1C) that show, for various fixed time points t after the onset of the input, the average distance

[5] For conceptual purposes we separate the “liquid” and “readout” elements in this paper, although dual liquid-readout functions can also be implemented.

$\left\| x_u^M(t) - x_v^M(t) \right\|$ of liquid states resulting from two different input spike trains u and v (plotted as a function of the distance $d(u,v)$[6] between u and v). Note the apparent absence of chaotic effects for this recurrent microcircuit with intermediate connection lengths.

Computations on Spike Trains in Generic Neural Circuits

As a first test of its computational power this simple generic circuit was applied to a previously considered classification task (Hopfield & Brody, 2001), where spoken words are represented by noise-corrupted spatio-temporal spike patterns over a rather long time interval (40-channel spike patterns over 0.5s). This classification task had been solved in (Hopfield & Brody, 2001) by a network of neurons designed for this task (relying on unknown mechanisms that could provide smooth decays of firing activity over longer time periods), whose structure limited its classification power to spike trains consisting of a single spike per channel. We found that the same, but also a more general version of the problem, allowing several spikes per channel, can be solved by any generic recurrent network as described in the previous section (Figure 1 D).

In order to explore the limits of this simple implementation of a LSM for computing on time-varying input, we chose another classification task where all information of the input is contained in the interspike intervals of a single input spike train. In this test, two randomly generated Poisson spike trains were chosen as template patterns, labeled 1 and 0. Other spike trains were generated by jittering each spike in the templates (more precisely: each spike was moved by an amount drawn from a Gaussian distribution with mean 0 and a SD that we refer to as "jitter"). The readout was trained to generate as target output *y(t)* the constant label 1 or 0 of the template from which the noise-corrupted spike pattern had been generated. Giving a constant output for a time-varying input is a serious challenge for a LSM, since it cannot rely on attractor states, and the memory-less readout has to transform the transient and continuously changing states of the liquid into a stable output.

The performance of the LSM after training was evaluated on inputs from the same distribution, but with an example that the liquid had not seen before. Figure 2 A and B show that even a very simple LSM, employing a randomly connected circuit of neurons as its liquid, is able to perform this task (for jitter = 4 ms) with a fairly high degree of accuracy. This demonstrates that recurrent connections between integrate-and-fire neurons endow such circuit with the capability to hold and integrate information from past input segments over

[6] In order to define the distance $d(u,v)$ between two spike trains u and v we replaced each spike by a Gaussian $\exp(-(t/t)^2)$ for τ = 5ms (to be precise, u and v are convolved with the Gaussian kernel $\exp(-(t/t)^2)$) and defined $d(u,v)$ as the distance of the resulting two continuous functions in the L_2-norm (divided by the maximal lengths 0.5 s of the spike trains u and v).

several hundred ms.[7] Since information from the input has to be integrated over a few hundred ms before an accurate classification of the input spike train is possible, a time delay is involved to generate an internal state that contains enough information. Hence the certainty at time *t*, defined as average correctness of the readout during the time interval [0, *t*], where

$$\text{correctness at time } s = 1 - |\text{target output } y(s) - \text{readout activity } p(s)| ,$$

increases with time *t*.

In this LSM, the certainty level reached was not maximal. Since the approximation property *AP* was already optimal (increasing the number of neurons in the readout module did not increase the performance any further; not shown), the limitation in performance lay in the separation property *SP*. This illustrates that the liquid needs to be sufficiently complex to hold the details required for the particular task, but should reduce information that is not relevant to the task (the spike time jitter in our example). *SP* can be engineered in many ways such as incorporating neuron diversity, implementing specific synaptic architectures, altering microcircuit connectivity and so on. In this simulation, a parameter λ regulated the average number of connections and the average spatial length of connections (see Appendix B). Increasing λ homogenized the microcircuit that facilitated chaotic behavior, decreased the performance, and increased the latency before reaching a given level of certainty in the task (Figure 2 C and D). For this classification task therefore, the "ideal liquid" is a microcircuit that has in addition to local connections also a few long-range connections, thereby interpolating between the customarily considered extremes of strictly total connectivity (like in a cellular automaton) on one hand, and the locality-ignoring global connectivity of a Hopfield net on the other hand.

Adding Computational Power

An interesting and powerful feature of LSMs is that *SP* can also be improved by simply adding more low performing liquids as additional "columns". Figure 3A shows the performance of the LSM for the case of high jitter (8 ms) with a single column, and Figure 3B shows the enhanced performance in this task when multiple columns are incorporated. Each column was a randomly connected circuit as described in Appendix B, with the same distribution of synaptic connections from the input and to the readout neurons (no connections between the columns). The performance difference between altering the internal microcircuitry of a single column and adding independent low-performing columns can be observed. Improving *SP* by altering the intrinsic microcircuitry of a single column increases sensitivity for the task, but also increases sensitivity to noise (Figure 3C). On the other hand,

[7] Control experiments (not shown) prove that this capability is provided by the recurrent connections even if all synapses are chosen to be linear, and hence cannot contribute to the temporal integration of information. The neurons themselves were simulated with rather short membrane time constants of 30 ms.

increasing the number of low-performing columns in the liquid also improves *SP* (and hence performance in the task), but maintains generalization power (Figure 3C). This suggests that competition for the best computational performance with a minimal number of neurons may arrive at a balance between the intrinsic complexity of the microcircuitry and the number of repeating columns depending on the task.

The benefits of using multiple columns can be understood more clearly by examining the characteristic curves for the *SP* of the liquids. This curve should be high for input distances $d(u,v)$ that are typical for input spike trains u and v from different classes, but low for values typical for the distance between spike trains from the same class - achieved by a liquid consisting of multiple columns. On the other hand increasing λ in a single column raises this characteristic curve for most values of $d(u,v)$ in a rather uniform manner (Figure 3D).

Parallel Computing in Real-Time on Novel Inputs

Since the liquid of the LSM does not have to be trained for a particular task, it supports parallel computing in real-time. This was demonstrated by a test in which multiple spike trains were injected into the liquid and multiple readout neurons were trained to perform different tasks in parallel. We added 6 readout modules to a liquid consisting of 2 columns with different values of λ[8]. Each of the 6 readout modules was trained independently for a completely different task. We focused here on tasks that require diverse and rapidly changing analog output responses $y(t)$. Figure 4 shows that each of the tasks can be performed in real-time with high accuracy, in spite of noise injected into the liquid neurons (an additional current drawn independently every 0.2 ms from a Gaussian distribution with mean 0 and SD 0.01 nA). Once trained on randomly drawn multiple input spike trains, all readout modules proceed accurately, simultaneously and in real-time on new inputs drawn from the same distribution.

Readout-Assigned Equivalent States of a Dynamical System

Real-time computation on novel inputs implies that the readout must be able to generate an invariant or appropriately scaled response for any input even though the liquid state may never repeat. Indeed, Figure 4 shows that the dynamics of readout pools do become quite independent from the dynamics of the liquid even though the liquid neurons are the only

[8] In order to combine high sensitivity with good generalization performance we chose here a liquid consisting of two columns as before, one with λ=2, the other with λ=8 and the interval [14.0 14.5] for the uniform distribution of the nonspecific background current I_b.

source of input. To examine the underlying mechanism for this relatively independent readout response, we re-examined the readout pool in Figure 4, for a particularly simple input and target output. The rate of the input Poisson train was held constant (Figure 5A), and the target output was the sum of these rates (i.e. a constant target). The liquid response (Figure 5B) and the average rate of all the liquid neurons (Figure 5C) shows that the liquid is not simply driven by the input, but rather takes on a dynamic of its own. Despite this dynamic response, after training the resulting average membrane potential of the readout neurons is almost constant when the target readout-output is constant (Figure 5D & E). The stability of the readout response does not simply come about because the readout only samples a few "unusual" liquid neurons as shown by the distribution of synaptic weights onto a sample readout neuron (Figure 5F). Since the synaptic weights do not change after learning, this indicates that the readout neurons have learned to define a notion of equivalence for dynamic states of the liquid. Indeed, equivalence classes are an inevitable consequence of collapsing the high dimensional space of liquid states into a single dimension, but what is surprising is that the equivalence classes are meaningful in terms of the task, allowing invariant and appropriately scaled readout responses and therefore real-time computation on *novel inputs.* Furthermore, while the input rate may contain salient information that is constant for a particular readout element, it may not be for another (e.g. one that extracts the derivatives of inputs, see Fig.4), indicating that equivalence classes and dynamic stability exist purely from the perspective of the readout elements.

Summary

We introduce the liquid state machine, a new paradigm for neural computation that is based on clear mathematical principles. This model emphasizes the importance of perturbations for real-time computing, since even without stable states or attractors the separation property and the approximation property may endow an analog computational system with virtually unlimited computational power on time-varying inputs. We demonstrated the computational universality of generic recurrent circuits of integrate-and-fire neurons (arbitrary connection structure), if viewed as special cases of LSMs. This provides the first stable and complete method for using recurrent networks of integrate-and-fire neurons to learn to carry out complex real-time computations on trains of spikes. Furthermore this approach provides possible explanations not only for the computational role of the highly recurrent connectivity structure of neural circuits, but also for their characteristic distribution of connection lengths, which places their connectivity structure between the extremes of strictly local connectivity and uniform global connectivity that are usually addressed in theoretical studies. Finally, we reveal a most unexpected and remarkable principle that readout elements can establish their own equivalence relationships on high-dimensional transient states of a dynamical system, making it possible to generate stable and appropriately scaled output responses even if the internal state never converges to an attractor state.

The LSM provides a complete framework for temporal processing and recurrent computation and therefore novel potential insight into the complex structure and function of microcircuits, for example the computational role of the complexity and diversity of neurons and synapses (see for example (Gupta et al., 2000)), which presents an obstacle for virtually any other modeling approach. While there are many plausible models for spatial aspects of neural computation, a biologically realistic framework for modeling temporal aspects of neural computation has been missing. In contrast to models inspired by computer science, the liquid state machine does not try to reduce these temporal aspects to transitions between stable states or limit cycles, and it does not require delay lines or buffers. Instead it proposes that the trajectory of internal states of a recurrent neural circuit provides a raw, unbiased, and universal source of temporally integrated information, from which specific readout elements can extract specific information about past inputs for their individual task. Hence the notorious trial-to-trial stimulus response variations in single and populations of neurons observed experimentally, may reflect an accumulation of information from previous inputs in the trajectory of internal states, rather than noise (see also (Arieli et al., 1996)). This would imply that averaging over trials or binning, peels out most of the information processed by recurrent microcircuits and leaves mostly topographic information.

This approach also offers an alternative to models for the computational organisation of perception that are not consistent with biological data. Instead of scattering all information about sensory input by recoding it through feedforward processing as output vector of an ensemble of general purpose feature detectors with fixed receptive fields (thereby creating the "binding problem"), it proposes that information could also be assembled and preserved in the trajectories of very high dimensional dynamical systems, from which multiple readout modules extract the information needed for specific tasks in an adaptive manner. Many of the particular neural coding features that have been demonstrated *in vivo* could therefore reflect specific readouts, whose ensemble does not necessarily capture the actual current state of the computation within the neural system. Hence the new conceptual framework presented in this article suggests to complement the experimental investigation of neural coding by a systematic investigation of the trajectories of internal states of neural systems, for example with regard to their separation property. Alternate experimental approaches may include, for example recordings of trajectories of states of internal microcircuits, which are compared on one hand with inputs to the circuit, and on the other hand with responses of different readout projections. In addition artificial readouts may be trained to extract information from previously not interpretable multi-unit recordings.

The liquid computing framework may provide a starting point for a new generation of cortical models that link LSM “columns” to form cortical areas where neighboring columns read out different aspects of another column and where each of the stereotypic columns serve both liquid and readout functions. In fact, the classification of neurons into liquid- and readout neurons is primarily made for conceptual reasons. Another conceptual simplification was made by restricting plasticity to synapses onto readout neurons, while future, more realistic learning algorithms, could allow simultaneous adaptation of synapses within microcircuits and those projecting out of microcircuits, to allow adaptation to a changing environment and a changing liquid. Furthermore, algorithms for unsupervised learning and reinforcement learning can be explored in the context of this new approach.

Finally, the general conceptual framework for computing on perturbations could also inspire novel approaches towards neuromorphic engineering. In addition to building LSMs from silicon neurons, one could pursue a different strategy in order to create a new generation of inexpensive low-power devices for parallel computation in real-time. A wide variety of materials may potentially enable inexpensive implementation of liquid modules with suitable separation properties, to which a variety of readout devices may be attached to execute multiple tasks.

Acknowledgement

We thank Wulfram Gerstner, Alan Murray, Misha Tsodyks, Thomas Poggio, Lee Segal, Tali Tishby, Idan Segev, Phil Goodman & Mark Pinsky for their comments on the ms. The work was supported by project P12153 of the Austrian Science Fond, the NeuroCOLT project of the EU, the Office of Naval Research, HFSP, Dolfi & Ebner Center and the Edith Blum Foundation. HM is the incumbent of the Diller Family Chair in Neuroscience.

References

Arieli, A., Sterkin, A., Grinvald, A., Aertsen, A. (1996) *Science* 273, 1868-1871.

Auer, P., Burgsteiner, H., & Maass, W. (2001) *The p-delta rule for parallel perceptrons*. Submitted for publication, online available at http://www.igi.TUGraz.at/maass/p_delta_learning.pdf.

Boyd, S., & Chua, L.O. (1985). Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE Trans. on Circuits and Systems*, 32, 1150-1161.

Douglas, R., Martin, K. (1998). Neocortex. In: *The Synaptic Organization of the Brain*, G.M. Shepherd, Ed. (Oxford University Press), 459-509.

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

Hertz, J., Krogh, A., & Palmer, R.G. (1991). *Introduction to the Theory of Neural Computation*. (Addison-Wesley, Redwood City, Ca).

Hopfield, J.J. & Brody, C.D. (2001). What is a moment? Transient synchrony as a collective mechanism for spatio-temporal integration. *Proc. Natl. Acad. Sci.*, USA, 89(3), 1282.

Maass, W. (2000). On the computational power of winner-take-all. *Neural Computation*, 12(11):2519-2536.

Maass, W., & Sontag, E.D. (2000). Neural systems as nonlinear filters. *Neural Computation,* 12(8):1743-1772

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

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

Shepherd, G.M. (1988). A basic circuit for cortical organization. In: *Perspectives in Memory Research*, M. Gazzaniga, Ed. (MIT Press), 93-134.

Tsodyks, M., Uziel, A., & Markram, H. (2000). *J. Neuroscience*, Vol. 20 RC50.

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

von Melchner, L., Pallas, S.L., & Sur, M. (2000). *Nature*, 2000 Apr 20; 404:871-6.

Appendix A: Mathematical Theory

We say that a class CB of filters has the *point-wise separation property* with regard to input functions from U^n if for any two functions $u(\cdot), v(\cdot) \in U^n$ with $u(s) \neq v(s)$ for some $s \leq 0$ there exists some filter $B \in CB$ that separates $u(\cdot)$ and $v(\cdot)$, i.e., $(Bu)(0) \neq (Bv)(0)$. Note that it is *not* required that there exists a filter $B \in CB$ with $(Bu)(0) \neq (Bv)(0)$ for any two functions $u(\cdot), v(\cdot) \in U^n$ with $u(s) \neq v(s)$ for some $s \leq 0$. Simple examples for classes CB of filters that have this property are the class of all delay filters $u(\cdot) \mapsto u^{t_0}(\cdot)$ (for $t_0 \in \mathbf{R}$), the class of all linear filters with impulse responses of the form $h(t) = e^{-at}$ with $a > 0$, and the class of filters defined by standard models for dynamic synapses, see (Maass and Sontag, 2000). A liquid filter L^M of a LSM M is said to be *composed* of filters from CB if there are finitely many filters $B_1, \ldots, B_m$ in CB – to which we refer as basis filters in this context – so that $(L^M u)(t) = \langle (B_1 u)(t), \ldots, (B_m u)(t) \rangle$ for all $t \in \mathbf{R}$ and all input functions $u(\cdot)$ in U^n.

A class CF of functions has the *approximation property* if for any $m \in \mathrm{N}$, any compact (i.e., closed and bounded) set $X \subseteq \mathbf{R}^m$, any continuous function $h : X \to \mathbf{R}$ and any given $\rho > 0$ there exists some f in CF so that that $|h(x) - f(x)| \leq \rho$ for all $x \in X$. The definition for the case of functions with multi-dimensional output is analogous.

Theorem 1: Consider a space U^n of input functions where $U = \{u : \mathbf{R} \to [-B, B] : |u(t) - u(s)| \leq B' \cdot |t - s| \text{ for all } t, s \in \mathbf{R}\}$ for some $B, B' > 0$ (thus U is a class of uniformly bounded and Lipschitz-continuous functions). Assume that CB is some arbitrary class of time invariant filters with fading memory that has the point-wise separation property. Furthermore, assume that CF is some arbitrary class of functions that satisfies the approximation property. Then any given time invariant filter F that has fading memory can be approximated by LSMs with liquid filters L^M composed from basis filters in CB and readout maps f^M chosen from CF. More precisely: For every $\varepsilon > 0$ there exist $m \in \mathrm{N}$, $B_1, \ldots, B_m \in CB$ and $f^M \in CF$ so that the output $y(\cdot)$ of the liquid state machine M with liquid filter L^M composed of $B_1, \ldots, B_m$, i.e., $(L^M u)(t) = \langle (B_1 u)(t), \ldots, (B_m u)(t) \rangle$, and readout map f^M satisfies for all $u(\cdot) \in U^n$ and all $t \in \mathbf{R}$ $\|(Fu)(t) - y(t)\| \leq \varepsilon$.

The *proof* of this theorem follows from the Stone-Weierstrass Approximation Theorem, similarly as the proof of Theorem 1 in Boyd & Chua (1985).One can easily show that the inverse of Theorem 1 also holds: If the functions in CF are continuous, then any filter F that can be approximated by the liquid state machines considered in Theorem 1 is time invariant and has fading memory. In combination with Theorem 1, this provides a complete characterization of the computational power of LSMs.

In order to extend Theorem 1 to the case where the inputs are finite or infinite spike trains, rather than continuous functions of time, one needs to consider an appropriate notion of fading memory for filters on spike trains. The traditional definition, given in 1, is not suitable. If $u(\cdot)$ and $v(\cdot)$ are functions with values in {0, 1} that represent spike trains and if $\delta \leq 1$, then the condition $\| u(t) - v(t) \| < \delta$ is too strong: it would require that $u(t) = v(t)$. Hence we define for the case where the domain U consists of spike trains (i. e. $0-1$ valued functions) that a filter $F : U^n \rightarrow (\mathbf{R}^{\mathbf{R}})^k$ has *fading memory on spike trains* if for every $u = \langle u_1, ..., u_n \rangle \in U^n$ and every $\varepsilon > 0$ there exist $\delta > 0$ and $m \in \mathbb{N}$ so that $\| (Fv)(0) - (Fu)(0) \| < \varepsilon$ for all $v = \langle v_1, ..., v_n \rangle \in U^n$ with the property that for i = 1,...,n the last m spikes in v_i (before time 0) each have a distance of at most δ from the corresponding ones among the last m spikes in u_i. Intuitively this says that a filter has fading memory on spike trains if the most significant bits of the filter output can already be determined from the approximate times of the last few spikes in the input spike train. For this notion of fading memory on spike trains one can prove:

Theorem 2: Consider the space U^n of input functions where U is the class of spike trains with some minimal distance Δ between successive spikes (e. g., Δ = 1 ms). Assume that CB is some arbitrary class of time invariant filters with fading memory on spike trains that has the point-wise separation property. Furthermore, assume that CF is some arbitrary class of functions that satisfies the approximation property. Then any given time invariant filter F that has fading memory on spike trains can be approximated by liquid state machines with liquid filters L^M composed from basis filters in CB and readout maps f^M chosen from CF.

The *proof* for Theorem 2 is obtained by showing that for all filters F fading memory on spike trains is equivalent to continuity with regard to a suitable metric on spike trains that turns the domain of spike trains into a *compact* metric space. Hence, one can apply the Stone-Weierstrass Approximation Theorem also to this case of computations on spike trains.

Appendix B: Details of the Computer Simulation

We used a randomly connected circuit consisting of 135 integrate-and-fire neurons, 20% of which were randomly chosen to be inhibitory, as a single "column" of neural circuitry

(Tsodyks et al., 2000). Neuron parameters: membrane time constant 30ms, absolute refractory period 3ms (excitatory neurons), 2ms (inhibitory neurons), threshold 15mV (for a resting membrane potential assumed to be 0), reset voltage 13.5mV, constant nonspecific background current I_b independently chosen for each neuron from the interval [14.975nA, 15.025nA], 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 $C \cdot e^{-(D(a,b)/l)^2}$, where l is a parameter which controls both the average number of connections and the average distance between neurons that are synaptically connected. We assumed that the 135 neurons were located on the integer points of a 15×3×3 column 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 Markram et al., (1998), 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, b were excitatory (E) or inhibitory (I), the mean values of these three parameters (with D, F expressed in second, s) were chosen to be .5, 1.1, .05 (EE), .05, .125, 1.2 (EI), .25, .7, .02 (IE), .32, .144, .06 (II). The scaling parameter A (in nA) was chosen to be 1.2 (EE), 1.6 (EI), -3.0 (IE), -2.8 (II). In the case of input synapses the parameter A had a value of 0.1 nA. The SD of each parameter was chosen to be 50 % of its mean (with negative values replaced by values chosen from an appropriate uniform distribution). The postsynaptic current was modeled as an exponential decay $\exp(-t/\tau_s)$ with τ_s=3ms (τ_s=6ms) for excitatory (inhibitory) synapses. The transmission delays between liquid neurons were chosen uniformly to be 1.5 ms (EE), and 0.8 for the other connections. For each simulation, except the ones discussed in Figure 1B, the initial conditions were identical for all trials.

Readout elements were made of 40 to 80 integrate-and-fire neurons (unconnected). A variation of the perceptron learning rule (the delta rule, see (Hertz et al., 1991)) was applied to scale the synapses of these readout neurons: the p-delta rule discussed in (Auer et al., 2001). 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. This rule, which formally requires to adjust the weights and the threshold of perceptrons, was applied in such a manner that the background current of an integrate-and-fire neuron is adjusted instead of the threshold of a perceptron (while the firing threshold was kept constant at 15mV). In Fig. 3,4,5 the readout neurons are not fully modeled as integrate and fire neurons, but just as perceptrons (with a low pass filter in front that transforms synaptic currents into PSPs, time constant 30 ms); in order to save computation time. In this case the "membrane potential" of each perceptron is checked every 20 ms, and it is said to "fire" at this time point if this "membrane potential" is currently above the 15 mV threshold. No refractory effects are modeled, and no reset after firing. The percentage of readout neurons that fire during a 20 ms time bin is interpreted as the current output of this readout module (assuming values in [0 , 1]).

Figure Legends

Figure 1: **A**: Schema for a LSM. A function of time (time series) $u(\cdot)$ is injected as input into the liquid module L^M, creating at time t an internal liquid state $x^M(t)$, which is transformed by a memory-less readout map f^M to generate an output $y(t)$. **B**: Two-dimensional projection of the trajectory of the liquid state for two different input spike trains (solid and dashed lines) in the case where the liquid consists of a circuit of integrate-and-fire neurons (see Appendix B for details). $\mathrm{psc}_1(t)$ and $\mathrm{psc}_2(t)$ are the total input currents received by two readout neurons at time t. The arrow marks the onset of the deviation of the trajectories for the two stimuli (time t=0 is marked by a black bullet). Note that the full state of the liquid as a dynamical system has a much higher dimension (proportional to the number of neurons + number of synapses). The "liquid state" $x^M(t)$ discussed in our model just has to contain those components of the full state which can be accessed by the readout. **C**: Average distance of liquid states for two different input spike trains u and v at a fixed time t (t=20,100,200,500ms) after the onset of the input increases as a function of the distance *d(u,v)* between the two inputs (see 6). Plotted on the y-axis is the average value of $\left\| x_u^M(t) - x_v^M(t) \right\|$, where ||.|| denotes the Euclidean norm, and $x_u^M(t)$, $x_v^M(t)$ denote the liquid states at time t (defined as the 135-dimensional vectors of currents injected at time t by the 135 liquid neurons into the readout neurons) for input spike trains u and v . The average state distance for a given input distance *d'* is calculated as the average over 200 randomly generated pairs u and v such that $|d' - d(u,v)| < 0.01$. Parameters of the liquid: 1 column, degree of connectivity λ = 2, see Appendix B. **D**: Application of the LSM to a more difficult version of a well-studied classification task (Hopfield & Brody, 2001). Five randomly drawn patterns (called "one ", "two", ..), each consisting of 40 parallel Poisson spike trains over 0.5s, were chosen. Five readout modules, each consisting of 40 integrate-and-fire neurons, were trained to respond selectively to noisy versions of just one of the patterns (noise was injected by randomly moving each spike by an amount drawn independently from a Gaussian distribution with mean 0 and variance 6ms). The responses of the readout which had been trained[9] to detect pattern "one" is shown for previously not shown noisy versions of three of the patterns. Initial conditions of neurons in the liquid were chosen randomly for each trial (membrane voltage was initialized by a random value from the interval [13.5mV, 15mV]), and internal noise was added to the neurons to demonstrate the robustness of the network (at each simulation step, i.e. every 0.2 ms) Gaussian noise with mean 0 and std 0.2nA was added to the input current of each neuron).

[9] The familiar delta-rule was applied or not applied to each readout neuron, depending on whether the current firing activity in the readout pool was too high, too low, or about right, thus requiring at most two bits of global communication. The precise version of the learning rule was the p-delta rule that is discussed in Auer et al., (2001).

Figure 2: Classification of randomly perturbed spike trains. **A** and **B**: The raster plots show for two sample test spike trains generated from templates 1 and 0 (top traces) the spikes generated by the liquid and readout neurons, after the readout neurons had be trained on 40 randomly perturbed spike trains (jitter=4ms) to produce the label of the template as constant output $y(t)$. The correctness and certainty (= average correctness so far) are shown from the onset of the stimulus in the lower two graphs. **C**: Average certainty (at time t = 0.5s, calculated as average over 200 trials) depends on the parameter λ that controls the distribution of random connections. In each trial a randomly connected circuit was constructed for a given value of λ and a readout population of 50 neurons was trained to perform the classification task for perturbed versions (jitter=4ms) of two randomly chosen Poisson spike trains (mean rate 20Hz , length 0.5s) as template patterns. **D**: The average latency increases with increasing λ. Latency was calculated as the average of times l where certainty reaches 0.7 from below (average over 200 trials).

Figure 3: A: Classification of spike trains with high noise (8ms jitter) by a one column LSM with λ = 2. Note low certainty levels. **B**: Improved performance by a LSM with 4 columns (λ=2). **C**: Noise robustness. Average certainty at time t=0.5s decreases with increasing jitter. Decrease in certainty levels with jitter depends on the number of modules and λ. Several columns (not interconnected) with low λ (internal connectivity) yield a noise robust LSM. **D**: Separation property depends on the structure of the liquid. Average state distance (at time t=0.5s) calculated as described in Figure 1C. A column with high internal connectivity (high λ) tends to chaotic behavior, equally sensitive to small and large input differences d(u,v). On the other hand the characteristic curve for a liquid consisting of 4 columns with small λ is lower for values of $d(u,v)$ lying in the range of jittered versions u and v of the same spike train pattern ($d(u,v) \leq 0.1$ for jitter $\leq$ 8 ms) and higher for values of $d(u,v)$ in the range typical for spike trains u and v from different classes (mean: 0.22).

Figure 4: Multi-tasking in real-time. 4 input spike trains of length 1500ms are injected into a single liquid module consisting of 2 columns, which is connected to multiple readout modules. Each readout module is trained to extract information for a different task. For each readout module we plotted the target function (solid line) and population response of the readout module (dashed line). The tasks assigned to the 6 readout modules were the following: Represent the *sum of rates:* at time t, output the sum of firing rates of all 4 inputs within the last 50ms. Represent a *switch in spatial distribution of rates:* output a high value if a specific input pattern occurs where the rate in input spike trains 1 and 2 goes up and simultaneously the rate in input spike trains 3 and 4 goes down, otherwise remain low. Represent the *derivative of the sum of rates:* output 1 if the sum of rates in the 4 inputs is increasing, output 0 if this sum is decreasing. Represent the *integral of the sum of rates:* at time t, output the total activity in all 4 inputs integrated over the last 250ms. Represent the

firing correlation over the last 100 ms: at time *t*, output a maximal value of 1 if at least 3 of the 4 inputs had a spike (no spike) within a time bin of 5 ms. Represent *pattern detection with memory:* output a high value if a specific spatio-temporal burst pattern appeared in the input during the last 100ms. Results shown are for a novel input drawn from the same distribution as the training inputs.

Figure 5: Readout assigned equivalent states of a dynamical system. A LSM (liquid as in Figure 2, with 80 readout neurons) was trained to output the sum of rates of the input spike trains (as in Figure 4). **A**: The input to this LSM are four Poisson spike trains with a rate that has a constant value of 100Hz. **B**: Raster plot of the 135 liquid neurons. **C**: Normalized population rate of the liquid; bin-size 20ms. **D**: Average membrane potential of a readout neuron (dashed line is the firing threshold). **E**: Readout response (solid line) and target response (dashed line). **F**: Weight distribution of a single readout neuron.