Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning

Gregor M. Hoerzer, Robert Legenstein and Wolfgang Maass

Institute for Theoretical Computer Science, Graz University of Technology, Graz, Austria

Address correspondence to Wolfgang Maass. Email: maass@igi.tugraz.at

This paper addresses the question how generic microcircuits of neurons in different parts of the cortex can attain and maintain different computational specializations. We show that if stochastic variations in the dynamics of local microcircuits are correlated with signals related to functional improvements of the brain (e.g. in the control of behavior), the computational operation of these microcircuits can become optimized for specific tasks such as the generation of specific periodic signals and task-dependent routing of information. Furthermore, we show that working memory can autonomously emerge through reward-modulated Hebbian learning, if needed for specific tasks. Altogether, our results suggest that reward-modulated synaptic plasticity can not only optimize the network parameters for specific computational tasks, but also initiate a functional rewiring that re-programs microcircuits, thereby generating diverse computational functions in different generic cortical microcircuits. On a more general level, this work provides a new perspective for a standard model for computations in generic cortical microcircuits (liquid computing model). It shows that the arguably most problematic assumption of this model, the postulate of a teacher that trains neural readouts through supervised learning, can be eliminated. We show that generic networks of neurons can learn numerous biologically relevant computations through trial and error.

Keywords: cortical microcircuit model, cortical plasticity, pattern generation, working memory

Introduction

Generic networks of neurons in different locations of the cortex perform a large variety of different computations and pattern generation tasks. It has remained an open problem how these diverse computational functions are attained and maintained, in spite of the generic laminar architecture of cortical microcircuits in different cortical areas. A large body of experimental work suggests that cortical function is adapted during learning in order to optimize performance. Populations of neurons in the lateral prefrontal cortex change their response properties during a delayed matching-to-sample task that involved noise-degraded visual stimuli in a way that correlates with performance improvements (Rainer and Miller 2000). Repeated training of working memory tasks can improve performance (Klingberg et al. 2002) and this improvement can transfer to tasks that were not a part of the training program (Klingberg et al. 2002). Functional magnetic resonance imaging studies showed that such training is accompanied by increased activity in the prefrontal and parietal cortex (Olesen et al. 2003). Functional adaptation of neurons in the motor cortex has been demonstrated in experiments where the neural readout in a brain–computer interface was perturbed during cursor control in a 3-dimensional virtual reality environment (Jarosiewicz et al. 2008). Finally, the classical experiments by Fetz and Baker (1973) showed that neural activity in cortical circuits of primates adapted to specific tasks, even in quite unnatural settings, such as a task where increased activity of single neurons in the motor cortex leads to reward.

It has already been shown that networks of spiking neurons with stereotypical connection probabilities can in principle support a large variety of computational tasks (Maass et al. 2002; Haeusler and Maass 2007; Haeusler et al. 2008; Buonomano and Maass 2009). In these models, termed liquid computing models, the synaptic weights of readout neurons that project to other circuits or areas are modulated by synaptic plasticity. The aim of synaptic adaptations is to approximate a desired output signal with the actual output of these readout neurons. Since these synaptic plasticity rules require knowledge of the desired output signal, this learning process is referred to as supervised learning, where a postulated supervisor or teacher supports the learning process. Furthermore, it has been shown that if such supervised learning is applied to readout neurons whose axons are also connected to other neurons within the network, such generic networks can also learn to generate periodic patterns and working memory (Jaeger and Haas 2004; Maass et al. 2007; Sussillo and Abbott 2009). These results imply that, in order to “program” the network to carry out a specific type of computation, it suffices to assign suitable values to the synaptic weights of some neurons. Furthermore, local synaptic learning rules are able to produce suitable weight settings, provided that the desired output (target output) of each readout neuron is provided at any moment during training by a teacher or supervisor. But for most concrete tasks, the assumption of such a teacher signal requires that there already exist some other neuron or network that is able to perform the desired computational task. Hence, this set-up is more suitable for duplicating a computational function, rather than for explaining how it could emerge in the first place.

We begin in this paper a new chapter in liquid computing theory, by investigating what computational properties can emerge in this approach if one eliminates all supervised learning; that is, all learning rules that require a teacher that tells a neuron how it should respond at any given time. We show that, for a large class of biologically relevant computational tasks, a teacher signal is not needed. It can be replaced by a biologically more realistic signal that assumes a high value if the average performance has recently increased and a low value otherwise. In fact, we show that several different tasks can be learned by different readouts simultaneously, based on such a relatively uninformative global feedback signal about average performance improvements. This set-up requires each neuron to explore different output values for the same network inputs. Hence, our model is not only consistent with
experimentally observed stochastic features of neuronal responses (trial-to-trial variability), but requires these stochastic features for its function. In this article, for the sake of simplicity, we are discussing only rate models for generic microcircuits. Mean field models predict that sufficiently large populations of spiking neurons will behave very similar to these rate-based models (see Discussion).

The learning rule that we apply is a variation of the exploratory Hebbian (EH) rule from Legenstein et al. (2010). The EH rule is a 3-factor learning rule (Kandel and Tauc 1965a, 1965b; Bailey et al. 2000; Fiete and Seung 2006; Fremaux et al. 2010; Pawlak et al. 2010) that depends on—besides pre- and postsynaptic neural activity—a modulatory third factor. A large set of experimental data shows that neuromodulators, such as dopamine, implement such a third factor in biological networks of neurons (Reynolds et al. 2001; Reynolds and Wickens 2002; Pawlak et al. 2010), but also a modulatory signal in the form of synaptic input can influence the amplitude of the backpropagating action potential of a neuron, and thereby the learning rate of spike-timing dependent plasticity (Waters and Helmchen 2004; Sjöström and Häusser 2006). In this paper, we employ a variation of the EH rule that imposes extremely weak demands on the information provided by this third factor. Rather than assuming that it provides information regarding how much—and in which direction—the current system response deviates from some hypothetical target response, we only assume here a 2-valued global third factor $M(t)$. This global signal informs all local plasticity mechanisms whether the system performance has recently improved. We investigate in this paper to what extent this—arguably the least informative performance-related third factor that one can conceive—suffices for installing in generic recurrent networks of neurons different, task-dependent, computational organizations. We consider 4 different types of computational tasks:

1. Periodic pattern generation.
2. Learning a rule, which requires storage of specific information in working memory, and application of this stored information for online computation on complex analog input signals.
4. Nonlinear analog computations on complex input signals.

In spite of the heterogeneity of these 4 computational tasks, we show that they can all be learned by the same generic neural circuit, requiring only the previously described weak information about recent performance improvement. There exists substantial evidence that neural networks of primates and other animals can carry out these tasks (see Discussion). But it has remained an open problem how biological neural networks could acquire these specific computational capabilities. In summary, the results of this work provide a new model for the emergence of diverse complex computations in biological neural systems.

The remainder of this paper is structured as follows. We first introduce the generic microcircuit model and the reward-modulated plasticity rule used in section Materials and Methods. In Results, we then show how this network can autonomously learn to generate a complex periodic pattern. The network performance is compared with the performance of previously studied models, and the robustness of network function to various types of noise perturbations is demonstrated. We then test the same network on a completely different task that requires the emergence of 2 independent working memory stores. In a final simulation task, we show that the same network can acquire the ability to dynamically route information in a state-dependent manner. Several predictions of our model are discussed in Discussion.

Materials and Methods

Network Architecture

We employed a generic network model consisting of $N$ sparsely recurrently connected neurons (with a connection probability of 0.1). We refer to these neurons as the network neurons in the following. The recurrent network model is generic in the sense that it is not designed for a particular computational function. Instead, connections within the network are randomly drawn such that network neurons are sparsely connected by excitatory and inhibitory synapses (cf. Supplementary Methods). Similar network models have been previously used to model the dynamics of recurrent biological networks of neurons (Amari 1972; Hopfield 1984; Haykin 1999; Sussillo and Abbott 2009). If necessary for the computational task, some network neurons receive in addition projections from external input streams $u_i(t)$. Specific computational functions are acquired through synaptic modification of the weights from neurons in the network to so-called readout neurons, which could represent, for example, layer 5 pyramidal cells (Maass et al. 2002; Jaeger 2003). These readout neurons can also feed back their activity into the recurrent neural network (Jaeger and Haas 2004; Maass et al. 2007). Figures 1, 4, and 3A show the basic network topology without and with external input streams, respectively.

In our model, the state $x_j(t)$ of neuron $j$ represents its membrane potential at the soma at time $t$, resulting from excitatory and inhibitory synaptic inputs (Hopfield 1984; Haykin 1999). The firing rate of the $j$th neuron at time $t$ is given by

$$r_j(t) = \tan h(x_j(t)) + e_{stat}(t),$$

where $e_{stat}(t)$ models zero-mean noise on the firing rate of the neuron (see Supplementary Methods for details on noise statistics). The network dynamics is given by

$$\tau \dot{x}_i(t) = -x_i(t) + \lambda \sum_{j=1}^{N} w_{ij}^{rec} r_j(t) + \sum_{j=1}^{M} w_{ij}^{in} u_j(t) + \sum_{k=1}^{L} w_{ik}^{fb} z_k(t),$$

where $\tau$ is the membrane time constant. The parameters $w_{ij}^{rec}$, $w_{ij}^{in}$, and $w_{ik}^{fb}$ denote the synaptic weights for recurrent connections within the network, connections from inputs to the network, and feedback connections from readout neurons to the network neurons, respectively.

Different dynamic regimes, from ordered to chaotic, can be accomplished by scaling the recurrent synaptic connections through the parameter $\lambda$. We choose a value of $\lambda$ so that, as in Sussillo and Abbott (2009), the dynamics of the recurrent network tend to be in the chaotic regime prior to learning. During learning, the readout neurons drive the network activities into a nonchaotic regime via the feedback pathway. Supplementary Figure 1 illustrates the contributions of input connections, recurrent connections, and feedback connections to the membrane potential of network neurons.

We assemble the firing activities (rates) of the network neurons at time $t$ into a column vector $r(t)$. Assembling the synaptic weights of connections from these neurons to a readout neuron $i$ in a corresponding column vector $w$, this readout neuron $i$ computes the function

$$\hat{z}_i(t) = w_i^T r(t) + b_i.$$

Downloaded from http://cercor.oxfordjournals.org/ at Max Planck Institut on November 12, 2012
Here, the bias $b_i$ models the baseline activity (spontaneous activity) of the neuron. Our tasks do not necessitate the bias term, but we chose to adopt it for consistency with Legenstein et al. (2010). Reward-modulated Hebbian learning of readout weights requires the readout neurons to be noisy (Legenstein et al. 2010). Therefore, the output of the $i$th readout neuron is modeled by

$$z_i(t) = \hat{z}_i(t) + \xi_i(t),$$

where $\xi_i(t)$ models zero-mean exploration noise on the firing rate of

Figure 1. Emergence of periodic activity through reward-modulated learning. (A) A recurrent network receives feedback from a readout neuron, but no other inputs. Circuit neurons are indicated as open circles in the gray schematic network. The readout neuron (right, with red incoming arrows indicating connections from circuit neurons) is trained to produce a specific periodic trajectory composed of 4 sinusoids with an overall period of 1 s. Red connections are subject to reward-modulated plasticity. (B) Beginning of learning (first quarter period shown). The dashed blue line indicates the onset of learning. After less than 50 ms, the readout output without exploration noise $\hat{z}(t)$ (red) approximately follows the target function $f(t)$ (black), a prerequisite for learning with direct readout feedback (i.e., without teacher forcing). The actual feedback signal $z(t)$ (light red) that is provided to the network includes the exploration noise $\xi(t)$, which is the driving force of learning. (C) The beginning of the testing interval where synaptic weights remain fixed (the last second of learning and first 2 s of testing shown). The dashed green line indicates the beginning of the testing period. After a short learning time (400 s for the presented example), the readout continues to approximately produce the target function without further weight adaptation. (D) During the testing interval, the readout output shows a small drift from the target function due to the fact that the frequencies of the oscillatory components are not learned perfectly (cf. also panel H). (E and F) Emergence of a stable periodic pattern of the network state through learning. Outputs of 5 randomly chosen units of the network are shown. The dashed blue line in panel E indicates the onset of learning. While the network produces spontaneous activity before learning, a stable periodic pattern emerges shortly after the onset of learning due to the drive of the feedback loop of the readout, with a rich set of diverse activity patterns across neurons within the network. The dashed green line in panel F indicates the beginning of the testing interval. After sufficiently long learning, the stable periodic pattern keeps being produced during the testing interval. (G) Comparison of the average test error for 3 different learning rules and varying learning times. While the nonlocal RLS-based FORCE rule (green) performs best, the reward-modulated Hebbian learning rule (red) performs similar to the local LMS-based FORCE rule (blue). (H) Variability of the frequency components of the trained signal. The average amplitude spectrum across the testing interval of all simulation trials with a learning time of 400 s is shown, together with the mean and standard deviation of the peak frequencies across successful trials (MSE < 0.01). Target frequency components are shown in red. While the precision of the frequency components is high in general, the small deviations lead to the drift depicted in panel D.
readout neuron $i$. We did not apply the tanh function for readout neurons to match the model of Sussillo and Abbott (2009). In some cases, this noisy readout signal is also fed back to the network neurons. One can apply exploration noise $\xi(t)$ either only during learning (then $z_i(t) = \tilde{z}_i(t)$ during a subsequent testing period), or during learning and testing, yielding a similar performance.

**Reward-Modulated Learning Rule**

In contrast to Sussillo and Abbott (2009), who use a fully supervised online learning rule to train the network, we investigate the capabilities of a biologically more plausible reward-modulated online learning rule. The supervisor is replaced by a binary signal $M(t)$ that communicates whether the average performance of all readouts (which are generally required to carry out different computational tasks) has recently increased. But it does not provide information on the sign and magnitude of the error, nor on which readouts contributed how much to a recent change in the performance. This single modulatory signal, which could be viewed, for example, as an abstract model for the phasic output of dopaminergic systems in the brain (Schultz et al. 1997; Schultz 2007), modulates synaptic plasticity of all synapses of all readout neurons in our simple model.

More precisely, the modulatory signal $M(t)$ is defined by

$$M(t) = \begin{cases} 1 & \text{if } P(t) > \bar{P}(t) \\ 0 & \text{if } P(t) \leq \bar{P}(t) \end{cases},$$

(5)

where $P(t)$ is the current performance of the system and $\bar{P}(t)$ is a low-pass filtered version of $P(t)$ that reflects its recent performance (see Supplementary Methods for the implementation of the low-pass filter).

The current system performance $P(t)$ is given by the sum of the mean squared errors (MSEs) of all readout neurons:

$$P(t) = -\sum_{i=1}^{N} \left[ z_i(t) - f_i(t) \right]^2,$$

(6)

where $f_i(t)$ is the target output of readout neuron $i$ for its assigned computational task. It is important to note that the values $f_i(t)$ of the target signals are not communicated to the neural network. This is appropriate, since their values may be unknown to a biological organism, while the existence of solutions $f_i(t)$ is assured through evolution.

We employ a variant of the EH rule proposed by Legenstein et al. (2010), where the weight change $w_i(t)$ of readout neuron $i$ at time $t$ is given by

$$\Delta w_i(t) = \eta(t)[z_i(t) - f_i(t)]M(t)x(t).$$

(7)

Here, $\eta(t)$ is a small learning rate that can either be constant, or decay over time, such that learning saturates as learning progresses. We use a decaying learning rate in all simulations reported in this paper. See Supplementary Results for additional simulation results with constant learning rates and for simulations used to determine the choice of the learning rate. Here $z_i(t)$ is a low-pass filtered version of the noisy output $z_i(t)$ of the readout (cf. Supplementary Methods). If one assumes that $z_i(t)$ changes only slightly within the time scale of the filter, and the noise is only weakly correlated over time, then the term $z_i(t) - \tilde{z}_i(t)$ approximates the noise $\xi(t)$. Therefore, in contrast to previously proposed rules (Fiete and Seung 2006), the rule does not need explicit information on the exploration noise, but instead estimates the noise autonomously (Legenstein et al. 2010).

This learning rule is Hebbian, since it uses the correlation of changes in the postsynaptic activity and the activity of the presynaptic neuron. It belongs to the category of 3-factor or reward-modulated Hebbian learning rules (Bailey et al. 2000; Legenstein et al. 2008; Fremaux et al. 2010; Pawlak et al. 2010), because this correlation is multiplied with the modulatory signal $M(t)$. Since our goal was to demonstrate learning in a minimal model, we used in our simulations a variant of the original EH rule, where the modulatory signal assumes only 2 possible values. From a biological perspective, this synaptic plasticity rule only distinguishes between a high and a low modulatory signal (that could, for example, be a high or low concentration of some neuromodulator). When compared with the original formulation in Legenstein et al. (2010), where $M(t)$ was chosen to be $P(t) - \bar{P}(t)$, this signal is much less informative. It does not communicate the magnitude of the performance change from the recent past but only whether it has improved at all. Additional simulations with the analog modulatory signal $M(t) = P(t) - \bar{P}(t)$ indicate that the performance of the 2-valued signal is comparable with the analog one; see Supplementary Results.

We point out that the exploration noise $\xi(t)$ is the driving force of learning. Without perturbations of the readouts’ output, no learning would take place.

In all our simulations, we used $N = 1000$ neurons in the recurrent network. These and other basic network parameters were chosen such that they correspond to the values used in Sussillo and Abbott (2009). We fixed suitable values for the other parameters that we kept for all simulations. In other words, we did not perform a parameter search in order to find the set-up with optimal performance for each individual simulation task (see Supplementary Methods for details on the parameter setting).

**Results**

**Autonomous Learning of Periodic Pattern Generation**

Biological neural networks produce many different types of rhythmic activities for various purposes, such as muscle activations, breathing, or locomotion. Sussillo and Abbott (2009) showed that a desired rhythmic activity can be acquired by generic recurrent neural circuits through supervised learning, where the desired output of each readout is provided at any moment during learning by a teacher or supervisor. The existence of such a teacher signal implies that some other neuron or network exists that is able to perform the task. Hence, this set-up cannot explain how the computational function could emerge in the first place. We therefore studied whether such tasks can also be learned autonomously without a teacher. We replaced the teacher signal by a modulatory signal $M(t)$ that indicates whether the performance of the neural circuit for the considered task recently increased; see Equation (5) in Materials and Methods. The supervised learning rule used in Sussillo and Abbott (2009) was replaced by reward-modulated Hebbian learning, that is, by Hebbian synaptic plasticity that is modulated by the modulatory signal $M(t)$; see Equation (7) in Materials and Methods. We simulated a network that receives no inputs besides the feedback projections from a single readout neuron (Fig. 1A). The task of the readout neuron was to produce a specific periodic trajectory and to repeat this periodic trajectory in a stable manner.

Since the actual output of the readout and not the target signal is fed back into the network during learning, the readout output has to resemble the target computational function already shortly after the beginning of learning (Sussillo and Abbott 2009). Figure 1B shows a representative example of the readout activity at the onset of the learning procedure. Within less than 50 ms, the readout is able to adapt its activity in order to reach the desired target, and to approximately follow the target function henceforth. The goal of learning is to find a set of time-independent weights such that the system is able to keep producing the target function when the learning mechanism is switched off after an appropriate learning time. Figure 1C shows that this goal is accomplished by reward-modulated Hebbian learning on the synapses from the
network to the readout neuron. After a learning time of 400 s, which corresponds to 400 oscillation cycles of the periodic trajectory, the readout keeps producing the desired trajectory in spite of the lack of any further weight adaptation. Here, no exploration noise was applied during testing. However, the performance is similar when exploration noise is applied during testing as well (cf. Supplementary Fig. 2A,B). After learning, one can usually observe a small drift from the desired trajectory over time, as depicted in Figure 1D. This drift arises due to the fact that the oscillation length of the frequency components of the readout output is not perfectly matched to the frequency components of the desired target signal. Such drift is to be expected irrespective of the applied learning mechanism. However, the difference in the oscillation length is very small (Fig. 1H). If an animal reproduces a periodic pattern, for example, for locomotion, performance depends on how well the shape of the pattern is reproduced, but the cumulative effects of the drift can generally be ignored. We therefore corrected for the drift in the subsequent performance evaluations. This was done by cutting the readout’s output during the testing interval into successive time slices of 1 s and by calculating the minimum MSE between each time slice and circularly shifted versions of a 1-oscillation cycle slice of the target pattern (Supplementary Methods).

The rhythmic activity of the readout, which drives the network via the feedback pathway, has a strong influence on the internal network dynamics. Figure 1E,F shows a subset of 5 random units within the network at the onset of learning and at the transition from learning to testing, respectively. Before the system starts with the learning process, the network exhibits chaotic dynamics and produces rich spontaneous activity. Shortly after the onset of learning, a stable periodic pattern emerges due to the driving force of the feedback loop (panel E). This stable periodic pattern persists during the testing interval when there is no further weight adaptation (panel F). A certain level of chaoticity—which we regulate by the parameter $\lambda$ that scales the weights of the recurrent network—is necessary for an accurate performance of the system. Initial chaotic dynamics are needed because the network has to initially produce sufficiently rich dynamics to properly generate the target function. On the other hand, if the chaoticity exceeds a certain level, the drive from the feedback loop is too weak to drive the network dynamics into a stable regime (Supplementary Fig. 2D; see also Supplementary Fig. 1 for the contributions of input connections, recurrent connections, and feedback connections to the membrane potential of network neurons). This is consistent with the results of Sussillo and Abbott (2009).

To investigate the learning time needed such that the network reliably autonomously reproduces the oscillatory pattern, we conducted 50 independent simulation trials with different learning times from 10 to 400 s. Each simulation trial consisted of a learning interval of varying length and a subsequent testing interval of 500 s. Moreover, in order to evaluate whether the elimination of the teacher leads to a significant decrease in performance, we conducted the same simulations with systems employing 2 different FORCE learning rules and compared the performance of the systems (cf. Supplementary Methods for a brief description of the FORCE learning rules). The supervised FORCE learning rules have previously been tested for readout training on similar tasks (Sussillo and Abbott 2009). Figure 1G shows the result of this comparison. The recursive least squares (RLS)-based FORCE rule (green) performs best, leading to a good approximation of the target signal after learning for as few as 10 s (which corresponds to 10 oscillation cycles of the target pattern). This is not surprising since the RLS-based rule uses nonlocal information about the correlations between all pairs of inputs to the readout to adapt the individual synapses. However, this approach seems to be problematic from the point of biological plausibility. The reward-modulated Hebbian learning rule (red) performs similarly to the local least mean squares (LMS)-based FORCE rule (blue) that still requires full knowledge of the desired output signal. With both of these local learning rules, the network needs to learn for approximately 100 s before a good performance level is reached. The desired trajectory was stably produced until the end of the testing interval in the majority of simulation trials (cf. Supplementary Fig. 2C). A comparison of the performance distributions across simulation trials showed that the performance of our learning rule did not differ significantly from the performance of the LMS FORCE rule for learning times of $>100$ s. The performance of our learning rule remained approximately constant across learning times $>150$ s (i.e., there was no significant difference between the performances at any 2 training times $>150$ s; nonparametric Wilcoxon rank-sum tests, significance level: 5%). For the comparison between the LMS FORCE rule and the EH rule, learning rates were independently set to good values that were obtained by a brute force search (see Supplementary Results for the choice of the learning rates for the 2 rules).

Biological networks of neurons have to be able to operate under the influence of substantial amounts of noise and other perturbances. Therefore, we have tested the noise robustness of the model. The results show that the system, which has learned without a supervisor, is indeed robust to noisy perturbations of the network state and the readout output as well as to long clamping of the readout output after learning. Additionally, substantial amounts of noise can also be applied to network neurons during learning (Supplementary Results).

**Necessary Conditions**

To investigate the limits of this approach, we tested the system behavior under various conditions. We performed simulations where we varied some of the system parameters and the properties of the target pattern. Three factors were varied concurrently, leading to a total of 45 parameter settings for which the network was tested (Fig. 2). Specifically, we investigated the influence of the following network and target pattern properties on the ability of the network to generate a periodic target pattern:

1. Frequency components of the target pattern (columns in Fig. 2A);
2. Update interval of the weights and the modulatory signal $M(t)$ (rows in Fig. 2A); and
3. Time constant of the exploration noise (x-axis of histograms in Fig. 2A).

The frequency components and therefore the rate of change of the target pattern are an important factor for the difficulty of the task. If the target signal changes too fast, the readout is not able to adapt its output quickly enough to follow the
target pattern during learning. Additionally, in order to assure that temporal averages are estimated with sufficient accuracy in the EH rule, the noise-free readout activation should change slowly on the time scale of these averages (see Legenstein et al. 2010 for details). Each column of Figure 2A corresponds to simulations with a set of given frequency components. The periodic target pattern in the above simulations had a length of 1 s with frequency components between 1 and 4 Hz (middle column). We performed simulations where this target pattern was scaled to lengths of 0.5 and 2 s,
network time constant. Higher frequency components can be strongly improved if the network time constant $\tau$ is reduced (cf. Supplementary Results).

For additional simulation results with different target patterns, see Supplementary Results.

**Autonomous Learning of a Computational Rule and of the Working Memory that it Requires**

Many cognitive operations of the brain require a working memory, where specific task-relevant information is stored for intervals up to a few seconds. Neuronal correlates of working memory have been observed, for example, in single neuron recordings from the prefrontal cortex of macaque monkeys during visual working memory in delayed matching-to-sample tasks (Fuster and Alexander 1971; Goldman-Rakic 1995; Miller et al. 1996; Bernacchia et al. 2011). In these experiments, it was observed that prefrontal cortex neurons hold information of previously observed stimuli by a persistent increase or decrease of their firing rates for a time interval in the range of seconds.

We tested whether such memory-dependent processing can emerge in our model through reward-modulated learning. We designed a task where good performance could only be achieved when the network state retained specific information about the input history. The task required the output value $z_1(t)$ of the first readout neuron to be high if target value $f_2(t)$ when—among the 2 input streams $u_{on,1}(t)$ and $u_{off,1}(t)$—the most recent high activity had occurred in stream $u_{on,1}(t)$; otherwise the readout value $z_1(t)$ was required to be low. The complexity of the task was increased substantially by adding a second readout neuron $z_2(t)$ that was expected to learn independently the corresponding task for 2 further input streams $u_{on,2}(t)$ and $u_{off,2}(t)$. The performance $P(t)$ of the whole system was given by the sum of the MSEs of the 2 readouts from their (implicit) target values; see Equation (6). As in the preceding simulation task, the network only received information whether this performance had recently improved through the modulatory signal $M(t)$. Thus, it neither received information about the target value of any of the readouts, nor noise correlations are beneficial for the learning process. With increasing frequencies in the target pattern, the system is likely to fail to reach appropriate performance levels if the exploration noise is temporally correlated (upper right panel). For additional simulation results with different target patterns, see Supplementary Results.
In summary, these simulations show that a generic neural circuit can learn a computational rule and simultaneously hold information about recently observed inputs for at least several seconds. This learning took place without any supervision, just from information about recent changes in global system performance.

**Emergence of Context-Dependent Switchable Routing of Information**

It has frequently been conjectured that networks of neurons in the brain are able to route information in a task- and context-dependent manner between relevant brain areas, giving rise to “effective connectivity” as opposed to “structural” network connectivity. But it has remained an open problem how this can be achieved (or even be learnt). We show in our last task that the same generic neural circuit, with the same general purpose reward-modulated Hebbian learning rule as in our preceding simulation tasks, can achieve this.
The network (Fig. 5A) receives here 4 input streams \( u_3(t), ..., u_4(t) \). The first 2 play the same role for the computational task of the first readout \( z_1 \) as before: They represent a switchable state (context), henceforth denoted by state “on” or “off.” The first readout has to learn to maintain a working memory of this context. The other 2 inputs \( u_3(t) \) and \( u_4(t) \) are 2 independently generated generic time-varying analog signals (Supplementary Methods). We require that the network learns to route the signal \( u_3(t) \) to the second readout \( z_2 \) if the network is in state “on”, and to route the signal \( u_4(t) \) to this second readout \( z_2 \) if the network is in state “off.” To test whether the network can simultaneously learn to carry out a demanding state-independent computation on the same 2 input signals \( u_3(t) \) and \( u_4(t) \), the global performance measure \( P(t) \) also included a third readout \( z_3 \) that was required to learn a complex nonlinear online computation on these 2 input signals, and to output \( f_3(t) = 0.5[u_3(t)^2 + u_4(t)^2 + u_3(t) u_4(t)] \). While the routing task could have been performed similarly without this third computation, we introduced it in order to make the task even more difficult and to show that, despite the feedback from the memory providing readout, the network state was still high-dimensional enough to be able to carry out such computation. Altogether, the performance measure \( P(t) \) was defined as the sum of the MSEs of the 3 readouts from their (implicit) target values; see Equation (6).

As before, the network (or more precisely: the learning rules for the synapses of the 3 readouts) received through the global signal \( M(t) \) only the information on whether this composite performance function \( P(t) \) had recently improved.

Figure 5B, C shows the readout outputs of the network at the end of a 500-s testing interval, after 500 s of learning. The first trace in Figure 5B shows an example in which the transition of the memory unit from the “on” state to the “off” state is correctly executed. At the time of this state transition, the routing unit also changes its output from approximately representing the input \( u_3(t) \) (blue) to representing the input \( u_4(t) \) (green). The third readout is not affected by the state switch and correctly computes the nonlinear function of both of these inputs throughout. This shows that a generic neural circuit is able to learn to perform concurrently complex memory-dependent and memory-independent nonlinear computations. Figure 5C shows an example from the same trial, but approximately 50 s earlier, where a switch of the first readout to the “on” state failed. The second readout behaves as expected: Since the first readout is in the wrong state, the second readout also represents the “wrong” input \( u_3(t) \) (green) instead of \( u_4(t) \) (blue). As soon as the first readout accurately switches to the “on” state, the second readout also switches to representing \( u_4(t) \) (blue). The correlation coefficient of the second readout with its target function is nevertheless altogether \( 0.8032 \pm 0.0087 \) and \( 0.8704 \pm 0.0033 \) based on the actual state of the first readout, averaged over 50 simulation trials with 500 s of testing. The first readout represents an incorrect state for \( 5.75 \pm 0.38 \% \) of the whole testing
time (50 simulation trials with 500 s of testing). The additional readout unit, which computes the nonlinear function of the inputs $u_3(t)$ and $u_4(t)$, remains largely unaffected by the wrong state of the first readout (correlation coefficient: $0.9327 \pm 0.0022$ with its target function $f_3(t)$, average over 50 simulation trials with 500 s of testing).

**Analysis of Emerging Computational Mechanisms**

We analyzed the behavior of the network in the switchable routing task with the help of a PCA of the network activity vectors $r(t)$ during the test epoch. Figure 6A shows that the network dynamics resides in a higher-dimensional space than in the working memory task analyzed above. This is consistent with the demands of the task. Two effective dimensions are sufficient to keep the 2 items of the former task in working memory. In addition to those required for the 1-item memory, the switchable routing task demands dimensions for the routing of information, as well as for the nonlinear online processing task. The reward-modulated Hebbian learning rule thus autonomously adapts the dimensionality of the network dynamics to the task at hand (note that exactly the same network parameters were used for both simulations). This is also apparent in Figure 6B. Here, 2 attractor regions can be identified that are visited according to the actual activity of the memory readout. These regions, however, occupy a larger volume of the state space (compare with Fig. 4C), since

**Figure 6.** Analysis of network dynamics in the switchable routing task. (A) Percentage of variance that can be explained by the first 1–10 principal components (#PCs) after learning. (B) First 3 principal components of the network trajectory after learning. The dynamics visits 2 subregions depending on the state of the memory readout (red: $z_1(t) < -0.3$; blue: $z_1(t) > 0.3$) and transitions between these regions (gray). (C) The histogram of correlation coefficients $cc_{Mem}$ between network neuron activities and the target function of the memory readout. (D) Neuron activity $r_i(t)$ of the neuron with maximal $cc_{Mem}$ (indicated by the black bar in panel C and arrow). The neuron is highly active whenever the memory readout should be active (indicated by gray shaded areas). (E) Pulse inputs $u_1$ (red) and $u_2$ (blue). (F) The histogram of correlation coefficients $cc_{u_1}$ between network neuron activities and pulse input $u_1$. (G) Neuron activity $r_i(t)$ of the neuron with maximal $cc_{u_1}$. The neuron exhibits peaks in its activity when a pulse in $u_1$ appears (compare with panel E). (H) The histogram of correlation coefficients $cc_{switched \ u_3}$ between network neuron activities and the target function of the routing readout at times when analog input $u_2$ should be routed to the readout. (I) Neuron activity $r_i(t)$ of a neuron with large $cc_{switched \ u_3}$ (black trace) along with analog inputs $u_3$ (red) and $u_4$ (blue). The neuron follows $u_3$ (red trace) when $u_3$ should be routed to the routing readout (indicated by gray shaded areas) and is preferentially highly active at other times.
additional computations have to be performed within the subspaces. Such attracting subspaces were termed high-dimensional attractors in Maass et al. (2007).

To further elucidate the computational mechanisms that emerged autonomously by reward-modulated Hebbian learning of readout weights, we analyzed the dynamics of neurons in the recurrent network after learning. Figure 6C shows the histogram of correlation coefficients $cc_{Mem}$ between network neurons’ activities and the target function for readout $z_1$ (the memory readout). As exemplified in panel D, neurons with large $cc_{Mem}$ showed sustained activity during “on” states. These neurons had on average strong excitatory recurrent weights to each other (Supplementary Fig. 5A). They also had on average strong inhibitory recurrent weights to neurons that were anticorrelated with the memory target (Supplementary Fig. 5B). Interestingly, we observed no correlation between $cc_{Mem}$ of a network neuron and the weight of its projection to the memory readout. However, when we considered only neurons with $cc_{Mem} > 0.6$, a positive correlation was significant (Supplementary Fig. 5C). Additionally, neurons with large $cc_{Mem}$ tended to receive strong feedback from the readout (Supplementary Fig. 5D). These findings indicate that the working memory benefited from recurrent connections in the network. The readout utilized such network neurons, but in a complex way. Successively reducing the feedback weights from the readout during the test interval leads to an amplification of the readout output, rather than to a reduction (Supplementary Fig. 6). This indicates a regulatory role of the readout, rather than just a boosting of activity through positive feedback.

How did the pulse input $u_1$ influence the working memory? Neurons with large $cc_{Mem}$ had on average no preference for input channel $u_1$ which initiates the persistent activity of readout $z_1$. Instead, we found a number of neurons in the network that were correlated with the pulse input $u_1$; see Figure 6F (and analogous neurons for $u_2$, not shown). These correlations $cc_{u1}$ were not particularly strong. But as shown in Figure 6G for the most strongly correlated neuron, these neurons had peak activities whenever $u_1$ was active (compare with panel E), resulting from strong synaptic inputs from $u_1$ (Supplementary Fig. 5E). The direct influence of such neurons on neurons with large $cc_{Mem}$ was presumably weak, as their weights to these neurons were small on average (Supplementary Fig. 5F). We thus hypothesized that the readout utilized these neurons to switch itself and therefore the whole attractor. Surprisingly, we found only a slight correlation between $cc_{u1}$ of a network neuron and its weight to the readout neuron (Supplementary Fig. 5G). To further elaborate on this point, we performed control simulations where we set all weights in the recurrent network to zero. We found that no attractor was attained. Instead, the memory readout mimicked $u_1$ and $u_2$. This shows that some disynaptic connections—from the switching inputs to the network, and then directly to the readout—substantially participated in switching the readout.

Finally, we investigated the question of how the network achieved switchable routing. We found neurons that were significantly correlated with the target function of the routing readout (maximal $cc_{rout} = 0.67$). These neurons were utilized by readout $z_2$ (the routing readout), since such neurons had preferentially strong projections to it (Supplementary Fig. 5H). Additionally, a number of neuron activities had an even higher correlation coefficient $cc_{switched u_3}$ with the target of the switchable routing task if only times were considered when $u_3$ should be routed to the output (Fig. 6I). Similar neurons were found for $u_4$. An example neuron with large $cc_{switched u_3}$ is shown in Figure 6I. It is also evident that this neuron was not correlated with $u_4$ (blue trace). As expected, neurons with large $cc_{switched u_3}$ received strong input from $u_3$ (Supplementary Fig. 5I). Such neurons, which follow the desired analog signal in the corresponding memory states and which are indifferent to the analog input in the other memory states, are presumably valuable presynaptic partners for the readout neuron. Consistent with this view, such neurons showed preferentially strong projections to the routing readout (Supplementary Fig. 5J).

In summary, this brief analysis shows that surprisingly complex computational mechanisms can be induced in a generic recurrent circuit by adapting the synaptic efficacies to readout neurons with a reward-modulated Hebbian learning rule.

Discussion

We have shown that heterogeneous and specialized computational functions can emerge through synaptic plasticity in generic sparsely connected recurrent networks of neurons. Furthermore, different computational structures can emerge simultaneously, even without an instructive teacher signal that tells each neuron what it should compute. Finally, we have shown that neural circuits need to receive very little information about the target output of each readout neuron: It suffices if they all receive a single global signal, which informs them whether their combined average performance has recently improved, or not. Arguably, this is the least informative performance-related signal that one can possibly conceive. We conjecture that if one reduces the information content of this global signal even further, then goal-directed learning is no longer possible.

A common feature of the 4 computational tasks considered in this paper is that there exists substantial evidence that neural networks of primates and other animals can carry out these tasks. Primates can learn to generate an immense variety of periodic movements, but it is not known how these capabilities are acquired and stored. Our results imply that, in principle, no specialized genetically encoded neural networks are required for this. This is of interest, because to the best of our knowledge no evidence for the existence of the latter has been found in primates. Furthermore, there exists a large body of experimental evidence that primates (Rodriguez and Paule 2009) and rodents (Rich and Shapiro 2009; Durstewitz et al. 2010) can learn rules for behaving that depend on specific cues, in a way that is likely to be rewarded. To accomplish such behavior, they must be able to keep relevant cues in working memory for extended periods of time. But it has remained an open problem how such rules are learned, and how working memory is implemented in neural networks of the brain. One remarkable recent experimental study (Bernacchia et al. 2011) suggests that working memory is implemented through heterogeneous neural subpopulations with different temporal responses, in a manner similar to that which emerges through reward-modulated learning in our study. A number of researchers have postulated that context- and task-dependent routing of information is required in the
brain for language understanding (Dominey et al. 2006), and for many other tasks where abstract knowledge needs to be applied for processing sensory input streams (Olshausen et al. 1993). Previous models for flexible routing of information in networks of neurons (Anderson and Van Essen 1987; Olshausen et al. 1995; von der Malsburg 2003; Zylberberg et al. 2010) required specially constructed neural networks. Recently, Vogels and Abbott (2009) proposed a model that is based on balanced excitation and inhibition. Additionally, several models have been introduced where synchronous activity is utilized for information routing in neural networks (Crick and Koch 1990; Salinas and Sejnowski 2001; Fries 2005; Akam and Kullmann 2010). Our work proposes a new model that relies neither on assumptions about synchronous network activity nor explicitly on balanced synaptic input. According to our model, generic networks of neurons can learn to route information as needed for specific computational tasks. Finally, it had already been shown in Maass et al. (2002) that specific analog computations on complex input signals can be learned by the readouts from a generic recurrent network of neurons through supervised learning. We show here that this learning also succeeds with a biologically more realistic feedback signal that does not require a teacher. 

Some specific conditions on the structure and dynamics of networks of neurons have to be met in order to make this learning result possible. One is that the output of neural circuits needs to exhibit trial-to-trial variability (i.e., stochasticity). Without that, no exploration of possibly better network specialization is possible. Another condition is that neural circuits need to have some basic features, which enable them to acquire virtually arbitrary computational specializations by just changing the synaptic weights of a few neurons. The underlying theory (Maass et al. 2007) guarantees only that this can be achieved if readout neurons can compute arbitrary continuous functions (without memory). However, if the neural circuit provides sufficiently rich generic nonlinear preprocessing (see the kernel property of Legenstein and Maass 2007), linear readouts tend to suffice (Maass et al. 2002; Legenstein and Maass 2007). Finally, for some of the desired computations (those that require a nonfading memory), it is necessary that readout neurons whose synaptic inputs are subject to synaptic plasticity also project their output back into the circuit.

In this paper, we considered only rate models for generic cortical microcircuits. Similar network models have been previously used to model the dynamics of recurrent biological networks of neurons (Amari 1972; Hopfield 1984; Haykin 1999; Sussillo and Abbott 2009). It has been shown in computer simulations that some related computational tasks can be learned by networks of integrate-and-fire neurons through supervised learning (Maass et al. 2007). This indicates that the general set-up is in principle compatible with networks of spiking neurons, although the network size will have to be increased considerably for comparable performance. Reward-modulated Hebbian learning rules similar to the EH rule used in this paper have also been formulated for spiking neurons (Fiete and Seung 2006). Such spike-based learning rules could be employed by spiking readout neurons. One major difference between the EH rule [Equation (7)] and related reward-modulated Hebbian learning rules is the estimate of the exploration noise $\xi_i(t)$ with the help of a low-pass filtered version $\bar{z}_i(t)$ of the current firing rate $z_i(t)$. The low-pass filtered version of the firing activity could be implemented in a biological neuron through its molecular machinery. For example, the intracellular concentration of $\text{Ca}^{2+}$ of a neuron is related to its recent firing history. This is also demonstrated by the fact that many pyramidal neurons exhibit spike frequency adaptation. Our model also predicts that information about recent increases or decreases in the firing rate is available to the molecular machinery that implements synaptic plasticity. But how this could be implemented is a wide open question, as are most questions regarding the molecular mechanisms that implement synaptic plasticity, and their exact time courses.

**Experimentally Testable Predictions of Our Model**

A primary prediction of our model is that no drastic differences in the structure of cortical microcircuits that perform different types of computations (periodic pattern generations, working memory, switchable routing of information, and nonlinear online computations) should be expected. Furthermore, our model predicts that a large number of different brain areas can learn to carry out these computations. In fact, several studies indicate that network function is only partially genetically predetermined, for example, in networks that generate periodic patterns for locomotion (Marder and Gaubard 2006) and in the auditory cortex (von Melchner et al. 2000). Other predictions of our model concern the organization of learning, more precisely the information provided by modulatory signals that gate synaptic plasticity.

Many studies indicate that the frontal cortex contains neurons that are sensitive to errors (Riddervikhof et al. 2004) as well as neurons that track past or future performance (Hasegawa et al. 2000). In traditional error-based learning approaches involving multiple readouts, each readout unit was supervised individually by providing it with its exact error (or even with its target value). According to our model, it is not necessary that an individual modulatory signal is provided to each readout. Instead, we use the same modulatory signal for all readouts, indicating only whether the collective performance has increased due to random noise perturbations. Consider, for example, a task where the overall error is given by the combined error in several subtasks. Our model suggests that such tasks can in principle be learned, even if only the overall error but not the individual error signals can be extracted from the sensory information. The fact that animals are able to learn motor tasks that demand the coordination of many muscle activations or motor synergies indicates that this is indeed possible.

Movement-related rhythmic activity patterns related to jaw and tongue movements have been found in the primary motor cortex of primates (Yao et al. 2002), which has been shown to be involved in learning of fine motor skills (Molina-Luna et al. 2009). Traces of persistent memory, reflected in sustained firing activity of single neurons in response to specific visual stimuli, have been recorded in the prefrontal cortex (Fuster and Alexander 1971; Goldman-Rakic 1995; Miller et al. 1996). Primary motor cortex and prefrontal cortex receive input projections from midbrain dopaminergic neurons, and the release of dopamine from such projections has been related to the expression of synaptic plasticity in this area (Molina-Luna et al. 2009; Hosp et al. 2011). Therefore, the entrainment of such movement-related activity patterns in...
the primary motor cortex and persistent memory traces in the prefrontal cortex may also be guided by modulatory input from midbrain neuromodulatory signals. Assuming that the proposed tasks are performed using such a learning mechanism, our model predicts that the synaptic adaptations that keep the desired trajectories stable during learning depend on the availability of global signals, such as specific neuromodulators. Without the presence of such signals, adaptation would not be possible. This is consistent with studies showing that both working memory performance and motor skill learning are impaired if dopaminergic system of the brain is degenerated, as in patients with Parkinson's disease (Durstewitz and Seamans 2002; Doyon 2008; Molina-Luna et al. 2009), and also with studies showing that working memory performance is impaired if dopaminergic input to the prefrontal cortex is blocked (Durstewitz and Seamans 2002). Moreover, our simulation results are also consistent with results indicating that dopaminergic signaling in the primary motor cortex is involved in learning new motor skills, but not in executing a previously learned skill (Molina-Luna et al. 2009; Hosp et al. 2011).

Conclusion
In summary, we have shown how diverse computational functions, such as periodic pattern generation, memory-dependent computations, and state-dependent routing of information, could be attained and maintained by generic cortical microcircuits via a biologically plausible 3-factor learning rule. It suffices that local synaptic learning rules receive a global modulatory signal that transmits some minimal information about global changes in task performance. Our results suggest that neuronal variability plays a crucial role in this learning process. It enables generic networks of neurons to learn important computational tasks without any supervisor or teacher (as postulated in previous work on the liquid computing model), simply through trial and error.

Supplementary Material
Supplementary material can be found at: http://www.cercor.oxfordjournals.org/

Notes
We also thank 2 anonymous reviewers for fruitful suggestions and Andrew Whitford for his helpful comments on the manuscript. Conflict of Interest: None declared.

Funding
This work was supported by the ORGANIC project (#FP7-231267), the Brain-i-Nets project (#FP7-243914), and the AMARSi project (#FP7-248311) of the European Union.

References
Emergence of Complex Computational Structures


Supplementary Materials for:

Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning

Gregor M. Hoerzer¹, Robert Legenstein¹, Wolfgang Maass¹,²

¹ Institute for Theoretical Computer Science
Graz University of Technology
A-8010 Graz, Austria
Tel.: +43 316 873-5822
Fax: +43 316 873-5805
[gregor,legi,maass]@igi.tugraz.at
² Corresponding author: maass@igi.tugraz.at

Contents:

1. Supplementary Methods
2. Supplementary Results
3. Supplementary Figures
1. Supplementary Methods

Recurrent neural network model and readout plasticity

For the recurrent network, we used a set of $N=1000$ leaky integrator neurons that were sparsely connected in a recurrent fashion.

Connection probabilities and weight distributions: Neurons in the recurrent network were connected with a probability $p=0.1$. In other words, the strength $W_{ij}^\text{rec}$ of each recurrent connection was set to zero with probability $(1-p)$. The weights of the network connectivity matrix $W^\text{rec}$ were drawn from a Gaussian distribution with zero mean and a variance of $1/(pN)$. The input weights $W^{\text{in}}$ and feedback weights $W^{\text{fb}}$ were drawn from a uniform distribution in the interval $[-1, 1]$. The output weights $w_i$, which we adapted during learning, were initialized to zero in our simulations. In principle, they can also be set to nonzero values. However, these values need to be relatively small in order to avoid that the readout output is too sensitive to noise on the network state because large values need to cancel each other out in order to obtain a small readout output. Note also that while the readouts with feedback in our simulations receive input and also project their output to all neurons within the network, each readout neuron could in principle also be substituted by multiple neurons with sparse connectivity.

In our simulations, we used the following parameter settings: network size $N=1000$ units, internal connectivity $p=0.1$, time constant $\tau = 50\text{ms}$ for the first and $\tau = 10\text{ms}$ for the other simulation tasks, chaoticity level $\lambda = 1.5$ for the first simulation task (without input) and $\lambda = 1.2$ for the other (partly input driven) simulation tasks, simulation time step $\Delta t = 1\text{ms}$.

Noise distributions: The noise $\xi_i^\text{state}(t)$ in the firing rate $r_i(t) = \tanh(x_i(t)) + \xi_i^\text{state}(t)$ of neuron $i$ was drawn from a uniform distribution in the interval $[-\theta^\text{state}, \theta^\text{state}]$ at each time step, with $\theta^\text{state} =0.05$. Since the sigmoidal $\tanh(\cdot)$ function nonlinearly maps the neuron's state $x_i(t)$ onto the interval $[-1, 1]$, the neuronal output $r_i(t)$ assumed values in the interval $[-1 - \theta^\text{state}, 1 + \theta^\text{state}]$. The exploration noise $\xi(t)$ was drawn independently for each readout and time step from a uniform distribution in the interval $[-0.5, 0.5]$.

In a subset of simulations, we used temporally correlated exploration noise. In these simulations, the actual noise at time $t$ was given by
\[ \xi(t) = \left(1 - \frac{\Delta t}{\tau_{\text{noise}}} \right) \xi(t - \Delta t) + \left(\frac{\Delta t}{\tau_{\text{noise}}} \right) X(t), \]  

(S1)

where \( \Delta t \) is the simulation time step and \( \tau_{\text{noise}} \) is the time constant of the temporal correlation. It was chosen between 1ms (corresponding to no noise correlation since \( \Delta t =1\text{ms} \)) and 5ms. \( X \) denotes a random variable that is drawn independently in every time step from a uniform distribution over the interval \([-0.5, 0.5]\). The initial value was also drawn from this distribution.

**Decaying learning rate:** We used a slowly linearly decaying learning rate of

\[ \eta(t) = \frac{\eta_{\text{init}}}{1 + \frac{t}{T}} \]  

(S2)

with initial learning rate \( \eta_{\text{init}} \) and a decay constant \( T=20\text{s} \). This linear decay ensures that adaptations take place throughout the whole learning interval and satisfies the convergence conditions for stochastic approximation (Robbins and Monro, 1951). The initial learning rate was \( \eta_{\text{init}} = 0.0005 \) for the reward modulated Hebbian learning rule, and the initial learning rate \( \eta_{\text{init}} = 0.0001 \) for the LMS-based FORCE rule (for information on the choice of the learning rate see Supplementary Results below). Additional simulations were performed with a constant learning rate (see Supplementary Results).

**Filter operations on readout output and performance:** For the reward-modulated Hebbian learning rule, the running average of the noisy readout output was given by

\[ \bar{z}_i(t) = \left(1 - \frac{\Delta t}{\tau_{\text{avg}}} \right) \bar{z}_i(t - \Delta t) + \left(\frac{\Delta t}{\tau_{\text{avg}}} \right) z_i(t), \]  

(S3)

and the running average of the overall performance was given by

\[ \bar{P}(t) = \left(1 - \frac{\Delta t}{\tau_{\text{avg}}} \right) \bar{P}(t - \Delta t) + \left(\frac{\Delta t}{\tau_{\text{avg}}} \right) P(t), \]  

(S4)

where \( \Delta t \) is the simulation time step. We used \( \tau_{\text{avg}}=5\text{ms} \) in all simulations except for those simulations where \( \tau_{\text{avg}} \) was varied.

**Definitions of target functions for the simulations**

The periodic function which had to be produced in the first simulation task was

\[ f(t) = 1.3/1.5\sin(2\pi1t) + 1.3/3\sin(2\pi2t) + 1.3/9\sin(2\pi3t) + 1.3/3\sin(2\pi4t). \]  

(S5)
The signal of the pulse input streams of the second and third simulation task were smoothed 100ms pulses which were generated with an average rate of 0.5Hz. More precisely, two variables $\hat{u}_{on,i}(t)$ and $\hat{u}_{off,i}(t)$ that indicated the start of a pulse in the on and off stream respectively, were independently set to 1 with a probability of 0.0005, and to 0 otherwise in each time step $\Delta t$. The input streams $u_{on,i}(t)$ and $u_{off,i}(t)$ were then smoothed pulses with an amplitude of 0.4 and a duration of 100ms (before smoothing). The smoothing filter time constant was 50ms. More formally, $u_{on,i}$ was given by $u_{on,i} = \frac{1}{\sigma_u} (\Theta_0 \circ (\hat{u}_{on,i} * h)) * g$, where $\circ$ denotes the convolution operation (we used a discrete convolution where the functions $g$ and $h$ were discretized with time-steps $\Delta t$), $\circ$ denotes function composition, and $\Theta_x(s)$ is the Heaviside function that is 0 for $s<0$, $x$ for $s=0$, and 1 otherwise. $h(s) = \Theta_1(s) - \Theta_1(s-100ms)$ is a 100ms pulse, and $g(s) = \exp(-s/\tau_L) \Theta_1(s)$ is an exponential smoothing function with time constant $\tau_L = 50ms$. The constant $\sigma_u$ scales the pulses to an amplitude of 0.4. $u_{off,i}$ was produced in the same way from $\hat{u}_{off,i}$.

The target values for the memory units were -0.5 for the ``off'' state and 0.5 for the ``on'' state:

$$f_i(t) = \begin{cases} 
0.5 & \text{if } \hat{u}_{on,i}(t) = 1, \\
-0.5 & \text{if } \hat{u}_{off,i}(t) = 1, \\
\hat{f}_i(t - \Delta t) & \text{otherwise.}
\end{cases} \quad (S6)$$

The actual target function $f_i = \frac{1}{\sigma_f} \hat{f}_i * g$ was then a smoothed version of $\hat{f}_i(t)$ with the exponential filter $g$ as given above and a time constant $\tau_L$ of 50ms. Here, $\sigma_f$ denotes a constant scaling factor that rescaled the function such that the values of $f_i(t)$ ranged between -0.5 and 0.5 again after the application of the smoothing filter.

The values of the two independent temporally correlated analog inputs $u_3(t)$ and $u_4(t)$ in the third simulation task were each drawn from a uniform distribution in each time step, filtered with a time constant of 0.5s and then scaled to have a standard deviation of 0.25. Bias values of 0.3 and 0.15 were added to these inputs, respectively.

The target function $f_2(t)$ was then defined as

$$f_2(t) = \begin{cases} 
u_3(t) & \text{if } \hat{f}_1(t) > 0, \\
u_4(t) & \text{otherwise.}
\end{cases} \quad (S7)$$
Comparison to FORCE Learning

Because we compare the performance of the reward modulated learning rule to the performance of the supervised FORCE learning rules proposed by (Sussillo and Abbott, 2009), we state them here in short. In contrast to the learning rule used here, the two variants of the FORCE learning mechanism use the exact error \( e_i(t) = \hat{z}_i(t) - f_i(t) \) of the \( i \)-th readout to update its weights \( w_i \). In the local least mean squares (LMS) based FORCE rule, readout weights are adapted according to

\[
\Delta w_i(t) = -\eta(t)e_i(t)r(t). \tag{S8}
\]

The more powerful recursive least squares (RLS) based FORCE rule is defined as

\[
\Delta w_i(t) = -e_i(t)C(t)r(t), \tag{S9}
\]

the matrix \( C(t) \) being a running estimate of the inverse of the correlation matrix of the network output \( r(t) \) plus a regularization term (cf. Haykin, 2001; Sussillo and Abbott, 2009 for details). Therefore, the RLS-based FORCE rule uses information about the activation of all synapses to the readout neuron to modify the learning rate of an individual synapse. This procedure makes the learning rule more powerful than the simple LMS-based rule, but at the expense of locality and simplicity of the rule, which involves some rather complicated computations to generate the matrix \( C(t) \).

For consistency with (Sussillo and Abbott, 2009) we did not apply exploration noise during learning of the FORCE-trained systems. However, control simulations with noise showed that this noise does not significantly change the system behavior (data not shown).

Performance evaluation

To evaluate the performance of the trained system for different parameter settings in the periodic trajectory production task, we used the following procedure. Since there are no input signals in this simulation task (cf. Figure 1A), the readout has no reference during the testing period, and small errors in the frequencies of the trained signal components therefore lead to a varying shift between the target and the readout output over time, i.e. the actual oscillation cycle length of the output of the readout is slightly longer or shorter than the one of the target signal (cf. Figures 1D and 1H). To see whether the shape of the target signal is nevertheless accurately reproduced by the readout's
output, we cut the readout's output during the testing interval into successive time slices of one second, which corresponds to the oscillation cycle length of the target function $f(t)$. Then, we calculated the minimum mean squared error (MSE) between each time slice and circularly shifted versions of a one-cycle slice of the target signal, instead of just calculating the MSE between the target function $f(t)$ and the readout's output $\hat{z}(t)$ directly.

In the comparative simulations employing the FORCE learning procedures, we conducted simulations with and without adding exploration noise during learning. The traces shown in figure 1E correspond to results without added exploration noise during learning. In any case, the error $e_i$ of the $i$-th readout that was used for weight adaptation was based on the readout output without exploration noise in these simulations.

To evaluate the performance of the memory units in the "persistent memory" and the "switchable routing" task, we calculated the percentage of time steps during the testing interval in which the absolute difference between the target function and the output of the readout neuron exceeded the threshold 0.5 (half the difference between the two target values 0.5 and -0.5 for the two different states).
2. Supplementary Results

The supplementary results section discusses only those results that have not been discussed in the main article. Since we arranged the figures in the order in which they are referenced in the main text and also grouped panels that are referenced in the main text together with non-referenced panels that show similar types of simulation results into single figures, only a subset of the figure panels are discussed here.

Additional tests for noise robustness of the system

To test the robustness of the system, we perturbed the network in different ways during learning and testing. In order to observe the response of the system to different amounts of noise on the network state during learning, we performed simulations with state noise levels $\theta^{\text{stat}}$ of 0.0, 0.05 (as in the original simulations), 0.1, 0.2, 0.3, 0.4, 0.5, 0.75 and 1.0 applied during the learning period. During testing, we always used the original state noise level of 0.05 in order to keep the results comparable. The learning time was 400s in all simulations. The mean squared error (MSE) was calculated across the whole testing period of 500s as described in the performance evaluation section above. Based on the MSE, we calculated median values across 50 trials per state noise level. The results are shown in Supplementary Figure S2E. Our simulations revealed that the system is very robust against noise perturbations of the network state during learning, leading to significant decreases in system performance only for surprisingly high amounts of state noise.

To further test the robustness of the system to unexpected perturbations during the testing interval, we applied the following perturbations and observed the system response. First, the system was perturbed for one second by noise in the feedback signal of the same amplitude as the exploration noise applied during learning. Second, it was perturbed with state noise on the network output $r(t)$ that was ten times stronger than the state noise usually used in the simulations. Figures S2F and S2G show that the trained system is robust against perturbations of the feedback signal (panel F) that affect the readout only via the network, and perturbations of the network output (panel G), which directly affect the input to the readout neuron itself. While perturbations of the feedback signal have no visible effect on the output of the readout since they are filtered by the slow dynamics of the network, the strong perturbations of the network output do have a visible effect on the readout output. However, while this type of noise lets the readout fluctuate around the desired trajectory during the time interval in which the system is perturbed, the desired trajectory is
restored shortly after the noise is removed again. Interestingly, the network is able to restore the desired trajectory even after extremely severe perturbations of the readout output. In a third perturbation simulation, we clamped the readout output such that it remained constant for 0.5s. After unclamping of the readout output, the system is able to recover the desired trajectory within a few oscillation cycles (see panel H of Figure S2).

**Choice of the learning rate for the EH rule and LMS FORCE**

In order to find an appropriate learning rate, we simulated the system with different learning rates $\eta$ between 0.001 and 0.00001 for both the LMS FORCE (Supplementary Figure S4A, left panel) and the reward modulated Hebbian learning method (right panel). The results indicate that the system performed best using a learning rate of $\eta=0.0001$ for the LMS FORCE rule and $\eta=0.0005$ for the reward modulated Hebbian learning rule. Performance increased for decreasing learning rates up to the learning rate used for our comparison between our learning rule and the FORCE rules. For even lower learning rates, the system failed to adapt quickly enough in order to follow the target trajectory appropriately after the beginning of the learning process, leading to significantly decreased performance for too small learning rates. We also performed simulations with and without ‘exploration noise’ applied during learning for the LMS FORCE rule (which is not needed for this kind of learning). The resulting performances were similar in both cases.

**Simulations with constant learning rate and analog modulatory signal**

We consistently used decaying learning rates and a binary modulatory signal for the results reported in the main text. Here we report simulations with constant learning rates and an analog modulatory signal for the weight updates. The comparison shows that the system performs equally well in all three conditions. In our simulations with constant learning rate, we did not decay the initial learning rate over time. For the analog performance-signal case, we used $P - \bar{P}$ directly instead of using the binary signal $M(t)$ in the weight update rule (7) together with a decaying learning rate. We performed simulations with $\eta=0.0005$ and $\eta=0.001$ for both conditions. Supplementary Figure S4B shows the simulation results, indicating that the performance does not vary significantly across the three conditions (decaying learning rate and binary modulatory signal, constant learning rate and binary modulatory signal, decaying learning rate and analog modulatory signal).
**Simulations with different target patterns**

In order to investigate a possible relation of the necessary learning time and the complexity of the target pattern, we performed simulations with target patterns with different numbers of superimposed frequency components (again, 50 simulation trials per learning time and pattern type). We used the same amplitudes for the frequency components as in the original target pattern, but left out some components for the simulations with lower numbers of frequency components. Supplementary Figure S4C shows the results, indicating that a simple sinusoid can be learned very quickly. For the target patterns with two to four superimposed sinusoidal components, the learning time was similar to each other. For five superimposed sinusoidal components (1-5Hz, in 1Hz steps), the system failed to reach the same level of performance as for the less complex target patterns.

Furthermore, we compared the results of the reward modulated Hebbian learning rule with the LMS FORCE rule for different target patterns. These patterns had the same number of superimposed components as the original target pattern (1-4Hz, in 1Hz steps), but with different amplitudes of the frequency components. This leads to target patterns with different shapes (see Supplementary Figure S4D). Our results indicate that performance can in general be different across different target patterns, but is nevertheless similar for both learning rules.

**References**


3. Supplementary Figures

A

B

C

D
Supplementary Figure S1: Contributions of recurrent connections, input connections, and feedback connections to the membrane potential of network neurons. Shown are contributions for two tasks considered in this article, a periodic pattern generation task (A, B) and a working memory task (C,D). A: Histogram of the number of neurons with a given feedback weight (bottom) and a given total weight from recurrent connections (top). The total weight is the sum over all incoming weights from recurrent network neurons (this includes the scaling factor $\lambda$). B: Histogram of the contributions of feedback connections (bottom) and recurrent connections (top) to the membrane potential $x_i(t)$ of network neurons. Probability of occurrence was computed over all time-steps and all network neurons in a 5 sec. test period after 500 sec. learning. C, D: Same as A and B respectively, but for a working memory task. In this task, input is provided to network neurons. The middle histograms show the number of neurons with a given weight from one of the input connections (C) and the contributions of input connections to the neuron’s membrane potential $x_i(t)$ (D). The sparse activity of pulse inputs in the working memory task is reflected in a sparse distribution in this histogram.
Supplementary Figure S2: Relation between chaoticity of the network, network performance, and robustness of the system against perturbations. **A:** System performance with exploration noise also applied during testing. Example trace of the readout output at the beginning of the testing period after 400s of learning. Although the feedback from the readout to the network is noisy also during testing, the readout keeps producing the desired trajectory. **B:** Comparison of the mean squared error (MSE) during a testing period of 500s without (left) and with (right) added exploration noise applied during testing (average over 50 independent simulations, 400s of learning). The system performance is similar in both cases. **C:** Distribution of the MSE across simulation trials within the last second at the end of a 500s testing interval (log scale) for the reward modulated Hebbian learning rule after 400s of learning (cf. Figure 1G, red trace). The network keeps producing the target function properly until the end of testing in most of the trials. **D:** Performance of the network after 400s of learning for different values of $\lambda$. The network needs a certain level of chaoticity, leading to rich enough dynamics, in order to generate the target function without further input. The system performs best for $\lambda$ values between 1.3 and 1.5. **E:** Median MSE for different state noise levels applied during learning. The value 1.0 was omitted here because it was much larger than the other noise levels (value: 0.1393). We compared neighboring noise levels for significant differences in performance distribution across trials. Only the distributions of the two largest levels (0.75 and 1.0) were significantly larger than performance of the the next lower level (as indicated by the bar with the star at for the level of 0.75). (nonparametric Wilcoxon rank sum test, significance level: 5%). **F, G:** Noise robustness of the trained trajectory. Uniformly distributed noise (of the same amplitude as the exploration noise used during learning) was added to the feedback from the readout $\hat{z}(t)$ (panel F, noise added in the shaded 1s interval) as well as to the network output $r(t)$, which represents the input to the readout unit (panel G, noise added in the shaded 1s interval). Noise on the readout output does not produce any visible deviations of the readout output. Noise on the network output directly affects the input to the readout and produces deviations that fluctuate around the desired trajectory. The desired trajectory is quickly restored after the noise is removed again. **H:** The readout output was clamped to a constant value for a time interval of 0.5s. After unclamping the output of the readout, the system is able to restore the desired trajectory within a few oscillation cycles.
Supplementary Figure S3: Simulations using different time constants for the network dynamics. A: Given that weight updates are fast (1ms) and the exploration noise is not temporally correlated, the system is also able to perform well for patterns with frequency components of up to 8Hz if the time constant of the network dynamics is reduced to $\tau=10$ms (cf. Figure 2A, upper right and middle right panels). For slower weight updates and for temporally correlated exploration noise, the system fails to produce the target function. B: Representative example traces of the readout activity (black) that show the system behavior in simulation trials with varying performance during the testing period and target output (red). The subpanel indices (a) to (c) are being indicated by the indices with arrows in panel A.
Supplementary Figure S4: Additional simulations. A: Comparison of different learning rates for the LMS FORCE (left) and Reward-modulated Hebbian (right) learning rules. We chose a learning rate of $\eta=0.0001$ (left, red trace) for the LMS FORCE rule and $\eta=0.0005$ (right, green trace) for the reward modulated Hebbian rule for our comparison between the learning rules, showing the respective best performance across different values of $\eta$. In general, the reward modulated Hebbian rule needed higher values of $\eta$ in order to perform properly. B: Simulations with analog reward and constant learning rate $\eta$, for two different learning rates each. In all cases, the system was able to perform the task properly if the system was trained for a sufficiently long time. C: Simulations with target patterns with different number of superimposed frequency components. While the system is able to learn a simple 1Hz oscillation within a few seconds of learning, it needs to be trained for a longer, but approximately similar time for two to four superimposed frequency components. For the simulations with five superimposed frequency components, the system in general showed decreased performance. D: Simulations with other target patterns with both LMS Force learning and our reward modulated Hebbian learning rule. For both patterns A and B, while having different performances for the different patterns, both learning rules performed similar for learning times larger than 75s in all but one cases (nonparametric Wilcoxon rank sum tests, significance level: 5%, exception: pattern B, 300s learning time, $p=0.0172$).
Supplementary Figure S5: Correlations between network neuron behavior and weight parameters in the switchable routing task. A: The correlation coefficient a neuron’s activity with the target function of the memory readout $cc_{Mem}$ is correlated with the mean weight of this neuron to neurons with high $cc_{Mem}$. B: The correlation coefficient a neuron’s activity with the target function of the memory readout $cc_{Mem}$ is anti-correlated with the mean weight of this neuron to neurons with large negative $cc_{Mem}$. C: The weight of a neuron to the memory readout is significantly correlated with its $cc_{Mem}$ for neurons with $|cc_{Mem}| > 0.6$. No such correlation is seen if all neurons are taken into account. D: $cc_{Mem}$ is strongly correlated with the feedback weight of the memory readout to this neuron. E: The weight that a neuron receives from pulse input $u_1$ is correlated with the correlation coefficient $cc_{u1}$ between this neuron’s activity and $u_1$. F: $cc_{u1}$ is slightly correlated with the mean weight of this neuron to neurons with large $cc_{u1}$. G: The correlation between the weight of a neuron to the memory readout and its $cc_{u1}$ is not significant. H: The correlation coefficient $cc_{routing}$ of a neuron’s activity with the target function for the routing readout is correlated with this neuron’s weight to the routing readout. I: $cc_{switchedu3}$ (the correlation coefficient of a neuron’s activity with the target function for the routing readout at times when $u_3$ should be routed to the readout) is strongly correlated with the weight from $u_3$ to this neuron. J: $cc_{switchedu3}$ is correlated with this neuron’s weight to the routing readout.
Supplementary Figure S6: Output of the memory unit in the switchable routing task with feedback weights linearly decaying to zero after learning. While the system is able to keep the memory states stable for a surprisingly long time despite the decaying feedback, the amplitude of the output is amplified rather than reduced, indicating a regulatory role of the feedback from the memory readout.