Model for a flexible motor memory based on a self-active recurrent neural network

Kim Joris Boström[a,b], Heiko Wagner[a,b,c], Markus Prieske[a,b], Marc de Lussanet[a,c]

[a] *Motion Science, Westfälische Wilhelms-University of Münster, Horstmarer Landweg 62b, 48149 Münster, Germany*
[b] *Center for Nonlinear Science (CeNoS), 48149 Münster, Germany*
[c] *Otto Creutzfeldt Center for Cognitive and Behavioral Neuroscience (OCC), 48149 Münster, Germany*

Abstract

Using recent recurrent network architecture based on the reservoir computing approach, we propose and numerically simulate a model that is focused on the aspects of a flexible motor memory for the storage of elementary movement patterns into the synaptic weights of a neural network, so that the patterns can be retrieved at any time by simple static commands. The resulting motor memory is flexible in that it is capable to continuously modulate the stored patterns. The modulation consists in an approximately linear inter- and extrapolation, generating a large space of possible movements that have not been learned before. A recurrent network of thousand neurons is trained in a manner that corresponds to a realistic exercising scenario, with experimentally measured muscular activations and with kinetic data representing proprioceptive feedback. The network is "self-active" in that it maintains recurrent flow of activation even in the absence of input, a feature that resembles the "resting-state activity" found in the human and animal brain. The model involves the concept of "neural outsourcing" which amounts to the permanent shifting of computational load from higher to lower-level neural structures, which might help to explain why humans are able to execute learned skills in a fluent and flexible manner without the need for attention

Email addresses: mail@kim-bostroem.de (Kim Joris Boström), heiko.wagner@uni-muenster.de (Heiko Wagner), m_prie03@uni-muenster.de (Markus Prieske), lussanet@psy.uni-muenster.de (Marc de Lussanet)

to the details of the movement.

Keywords: Neural Networks, Motor memory, Motor Control, Motor Learning, Reservoir Computing

1. Introduction

Deenah the dancer works on a new choreography. She closes her eyes and concentrates on the moves she is going to perform. When she opens her eyes again, she performs her first slow moves. From time to time she takes a look at the large mirror on the wall and corrects a posture or stops to repeat a particular movement. Deenah teaches herself the new choreography by executing it over and over. At the end of the day she performs hundreds of delicate movements by heart, in correct order, fast, fluent and filled with just the right amount of emotional expression, as though her body moved by itself.

This is just an imagined scenario, but it exemplifies in a paradigmatic manner a remarkable phenomenon: When we repeat movements over and over, even if they are highly complicated and initially attract all of our attention, then the movements will become more and more fluent and precise, and eventually we are able to perform them without paying attention to the details. Although we still perform the movements *deliberately*, there is a sense in which they have become *automatic*, but not so far as to merely reproducing the exact learned movements in a robotic fashion. Rather, we are able to continuously recombine and modulate the learned movements in a flexible manner, so that they fit to our intentions and to the demands of the environment.

Theories of motor learning rely on the concept of a *motor memory*. How does the brain, and probably also the spinal chord, store and recall movements or, more generally, *dynamical information*? There is a large amount of literature on proposed solutions to the issue of storing and retrieving time-sensitive information in a biological system, and the review of them all would go beyond the scope of this paper. Many of the proposed models are designed to deal with the problem of how a set of temporally ordered discrete "items" is learned and recalled (Atkinson & Shiffrin, 1968; Grossberg, 1978; Grossberg & Paine, 2000; Rhodes et al., 2004; Grossberg & Pearson, 2008). In the case of *serial recall*, the temporal order of the items needs to be preserved, while in the case of *free recall* the temporal order needs not be preserved.

There are models for short-term and long-term memory, and they qualitatively capture the phenomenon that, for example, it is easier to recall items from the beginning and the end of a learned sequence (*primacy effect* and *recency effect*, respectively). A powerful and physiologically plausible model that realizes both serial and free recall is the LIST PARSE model proposed by Grossberg & Pearson (2008), which encodes a discrete temporal sequence by "an analog spatial pattern of activation that evolves in parallel across a network of content-adressable cells" (ibid., p. 683).

Moreover, there is a conception in motor learning theory called *motor chunking* (Book, 1908; Verwey & Dronkert, 1996; Grossberg & Pearson, 2008; Wymbs et al., 2012). The idea is that movements are internally segregated into *chunks* which are stored and recalled individually to improve the efficiency of memory usage and information processing. The motor chunks act like words of a vocabulary composed of individual *motor primitives* as letters, and their combination results in a broad range of possible movements. One part of the motor system would *parse* every planned movement for primitives, while another part would *concatenate* these primitives into chunks, and both processes complement each other to achieve optimal efficiency (Wymbs et al., 2012).

Most models of sequential learning focus on the storage and retrieval of a discrete sequence of *items*, whereas a movement, on the other hand, is a continuous dynamical function. Even given that movements are parsed into sequences of chunks and then stored, the question remains how the individual chunks are stored as continuous dynamical functions within the nervous system. In the present study we would like to address this latter question, so our proposed model is compatible with any model of a discrete sequential memory, as the former would represent a supplement to the latter. Specifically, we wish to put forward the concept of a *flexible motor memory*, that is, a neural mechanism to store, recall and modulate elementary motor primitives for the generation of complex movements. The process of storing motor primitives in the nervous system, according to our view, is realized during the repeated execution of movements, and it involves a higher-level system that trains a lower-level system to learn individual movement patterns as motor primitives, so that after learning the higher-level system may recall, combine and modulate the learned patterns from the lower-level system, resulting in a broad range of possible movements that considerably exceeds the learned range.

We implement these theoretical concepts in a numerical model that relies

on recent recurrent network architecture exploiting certain novel and unique features outlined below. The intention of our study is to investigate the biological relevance of these features for the computations that occur in the central nervous system. To better understand the underlying principles, the precise anatomy of individual neuronal structures has to be simplified and idealized. The resulting model involves an idealized network that is trained with experimentally measured electromyographic (EMG) data representing the muscular activations of a biological system. We will discuss some interesting aspects of the results of our simulations in the light of their possible functional role and their physiological plausibility.

2. Theoretical concepts

We conceptualize that lower-level neural structures are enacted by higher-level structures for the execution of movement patterns. The lower-level structures learn these patterns during their repeated execution, so that eventually the lower-level structures are able to generate the learned patterns by themselves in response to drastically reduced abstract commands from the higher-level structures. This amounts to a permanent shifting of computational load from the higher level to the lower one, a process that we denote as *neural outsourcing*. After the learning, the lower-level structures are able to *generalize* the learned patterns in such a way that they can interpolate and extrapolate the learned patterns within and beyond the learned range, respectively. As already mentioned, a system that displays such a behavior is what we denote as a *flexible motor memory*. In our model, movement patterns in the form of periodic continuous dynamical functions are stored in the synaptic weights of a recurrent neural network. This raises the question how *dynamical* information can possibly be stored into *static* synaptic weights. The key to understanding how this may be possible is to conceive of a neural network not as a mere input-output device but rather as a *dynamical system* that evolves on its own. In the flexible motor memory that we suggest, movements are not stored as fixed "movies" that are merely replayed but rather as dynamical properties of a recurrent neural network. When the network is trained to respond dynamically in a specific way in the presence of a specific static input, then this potentially realizes a flexible memory for movement patterns.

Such a behavior cannot be achieved by a feedforward network, that is, by a network where the information flows in only one direction from input to

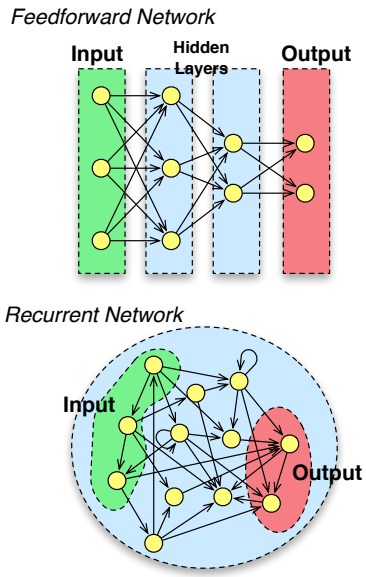

Figure 1: Top: Feedforward network. There are dedicated input and output layers of neurons with intermediate "hidden" layers of neurons. Bottom: Recurrent network. There are no layers, all neurons may be connected to each other including themselves, and every neuron can potentially act as input or output.

output, crossing a set of "hidden" layers (Figure 1). A feedforward network is in fact just an input-output device that maps static input to static output,

$$x \mapsto y, \tag{1}$$

so that in order to get dynamical output also the input already has to be dynamic. Only recurrent networks, that is, networks where every neuron may be connected to every other neuron, are capable of mapping static inputs to dynamic outputs,

$$x \mapsto y(t). \tag{2}$$

This is because recurrent networks have the potential to be *self-active*, that is, they may admit recurrent flows of activation even in the absence of input, a feature that strikingly resembles the *resting-state activity* of the brain (Biswal et al., 1995; Laufs et al., 2003; Smith, 2012). Hence, recurrent networks are capable of generating dynamical output by themselves, $0 \mapsto y(t)$, which implies that they are more than just input-output devices. Moreover, these networks are capable of mapping dynamic input to static or dynamic output, $x(t) \mapsto y$ and $x(t) \mapsto y(t)$, respectively, which in effect may realize real-time temporal pattern recognition and response modulation. Altogether one may say that recurrent neural networks are potentially powerful generators of complex and flexible dynamical behavior.

Artificial recurrent neural networks are sometimes used as analytical tools to identify and characterize input-output relationships and activity patterns of physiological systems (vocal tract: Burrows & Niranjan, 1995; muscle activity: Draye et al., 1997, Cheron et al., 2007). However, this is not the purpose of our use of an artificial neural network in the present study. Rather, we intend the artificial network to serve as a simplified and idealized model for a biological network of neurons.

The notorious problem with recurrent networks is that in general they are enormously difficult to train. The training is traditionally done by a numerical method called *error backpropagation* (Rumelhart et al., 1985) which is physiologically highly implausible and computationally so demanding that only small networks (tens of neurons) can be trained on available hardware. Only by the advent of *reservoir computing* it became possible to train much bigger recurrent networks (thousands or even hundreds of thousands of neurons) in reasonable time on available hardware ("echo state networks": Jaeger, 2001; "liquid state machines": Maass et al., 2002; see Lukoševičius & Jaeger, 2009, for a review). Reservoir networks are numerically much more effi-

cient since only the output weights are adapted during the learning process. Their unique features and their numerical efficiency make reservoir networks ideal for application in motor learning theory, and they have recently been used to simulate central pattern generators in robot locomotion (Wyffels & Schrauwen, 2009; Jaeger et al., 2012).

The basic idea of reservoir computing is to conceive of a recurrent network not as a fixed program-driven machine but as a continuous dynamical system that responds to an input signal similar to how a water surface responds to the impact of a stone. The information about the input, such as the size and weight of the stone, the velocity and angle of impact, is contained in the dynamical response of the water surface. If certain points on the surface are read out in an adequate manner, the water can effectively perform almost arbitrary "calculations" on these values[1]. A *reservoir*, in the sense of reservoir computing, consists of a large heap of neurons that are randomly connected with each other, and it is only their output weights that define the relevant response to the input. Learning takes place while feeding the network with slight variations of the target function. The network then starts to reverberate the target function, and the output weights are adapted so that a temporarily stable resonance state is reached.

For our model we chose a further development of the reservoir computing approach, proposed by Sussillo & Abbott (2009): the FORCE (**F**irst-**O**rder **R**educed and **C**ontrolled **E**rror) network architecture (Figure 2). It offers two advantages against other types of reservoir networks. First, the learning takes place *online*, that is, the target functions are trained one after the other in subsequent "lessons", whereas the network training algorithms of other architectures involve *offline* learning, that is, they process all input/output pairs in parallel and not in sequence. This is not a plausible, yet not even a physically possible strategy in the case of motor learning: Humans and other biological systems cannot learn more than one movement sequence at a time. Second, FORCE networks bring their output close to the target *from the beginning on*, while traditional learning algorithms do so in small steps only. For the case of motor learning the latter would mean that the initial movements would appear erratic until they slowly become more and more targeted. However, humans do obviously *not* flail around their limbs

[1]This has literally been demonstrated by having a bucket of water performing pattern recognition of spoken words (Fernando & Sojakka, 2003).

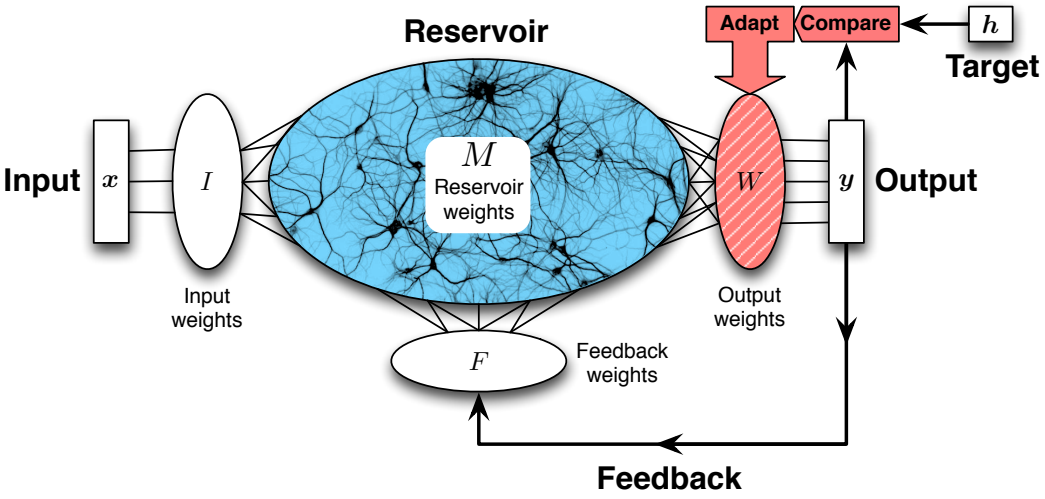

Figure 2: Scheme of the FORCE learning network used in this study. During learning the output weights are quickly and strongly modified so that the output closely (but not exactly) matches the target function. The output is fed back, causing the network to resonate with itself. As learning proceeds, the weight update rate is minimized so that after successful learning the network generates the target function even with constant output weights.

erratically when they exercise a particular movement. They start off with targeted movements and then *improve* their performance in terms of fluency, speed, and precision (Wolpert et al., 2011). Altogether, the FORCE network architecture meets the basic requirements of human motor learning.

In the present study, we demonstrate another remarkable capacity of the FORCE network: Without any modification or optimization of the learning process, the network was able to *morph*, that is, to inter- and extrapolate between the learned patterns in an approximately linear fashion. This morphing capacity of the network might help to explain how to overcome an obvious limitation of the human motor system: It is impossible to learn *all* variations of a particular movement. Instead, the system would only have to repeatedly execute some particular movement, parts of which would be stored as motor primitives and later be morphed into a broad range of new movements. In view of the morphing capacities of the network, few discrete points in the continuous space of possible movements would suffice to be learned by repeated execution, and then the system would be able to freely interpolate between them and even extrapolate beyond them. This concep-

tion constitutes an extension to the idea of fixed motor programs or program schemes to generate a continuum of movements, as for example in the schema theory of Schmidt (1975), and it fits well to certain theories of motor control, notably the *differential learning approach* (Schöllhorn, 1999), that postulate a benefit of learning *variations* of a movement pattern instead of learning only the exact pattern. For if the system learns variations of a movement pattern, it would afterwards be capable of morphing between the variations.

Lastly, besides muscular activations we also had the network learn a kinetic feature of the generated movement, in this case exemplarily the elbow joint angle of the moved arm. The motivation was that if the executive system is able to store and recall muscular activation patterns generating a particular movement, then it could also simultaneously store and recall sensory feedback signals provided by the proprioceptive system during the execution of that very movement, similar to an *efference copy*. These signals could later be used as a cheap "as-if-simulation" of the expected sensory consequences of the movement. In contrast to a *full* internal simulation, which would be very demanding on the neural resources, a simple recall of memory traces stemming from the proprioceptive system would be much cheaper and can as well be used to predict what it would be like if the movement was executed. The so predicted proprioception may then be compared to the actual proprioception, which might yield valuable information for higher-level systems (e.g. to correct postures or to adapt perception) and also for lower-level regulative systems (e.g. to induce pain or to modulate reflexes). The joint angle measured by our external camera system during the experiment served to represent proprioceptive information. It is relevant to remark that in our simulations this "proprioceptive" information was also morphed correctly, that is, the predicted joint angle time course changed its frequency in accordance to that of the muscular activation patterns. In this sense, thus, the network may act as a flexible and computationally cheap *simulation engine* to predict the effects of planned movements.

3. Methods

3.1. Numerical simulations

We implemented a FORCE network analog to the one proposed by Sussillo & Abbott (2009). The differential equations for a network of N reservoir neurons receiving a K-dimensional input and yielding an L-dimensional output are given by

$$\tau \dot{q}_n(t) = -q_n(t) + \sum_{k=1}^{K} I_{nk} x_k(t) + \sum_{m=1}^{N} M_{nm} r_m(t) + \sum_{l=1}^{L} F_{nl} y_l(t) \quad (3)$$

$$r_n(t) = \tanh(q_n(t)) \quad (4)$$

$$y_l(t) = \sum_{m=1}^{N} W_{lm} r_m(t) \quad (5)$$

where τ is a global time constant that has been set to $\tau = 0.01$ simulated seconds[2], $n = 1, \ldots, N$ is the index of the network neuron, $q_n(t)$ is the internal excitation state of the n-th neuron, $r_n(t)$ is the output firing rate of the n-th neuron with negative rates corresponding to inhibition, $x_k(t)$ is the k-th component of the input vector $\boldsymbol{x}(t)$, $y_l(t)$ is the l-th component of the output vector $\boldsymbol{y}(t)$, M_{nm} is the synaptic weight connecting the m-th with the n-th neuron, I_{nk} is the synaptic weight connecting the k-th input with the n-th neuron, F_{nl} is the synaptic weight for the feedback from the l-th output to the n-th neuron, W_{lj} is the synaptic weight connecting the n-th neuron with the l-th output. Note that (3) is a nonlinear differential equation because of the nonlinearity of the hyperbolic tangent in (4). The hyperbolic tangent restricts the output to the interval $[-1, 1]$ in a smooth sigmoid fashion and is both numerically efficient to calculate and biologically plausible enough within the range of precision considered here; in opting for the hyperbolic tangent we follow Sussillo & Abbott (2009), but some other sigmoid function, e.g. the logistic function, would also do.

FORCE networks are non-spiking neural networks, in contrast to other variants of reservoir networks such as *Liquid State Machines*, so only the firing rates of the neurons are calculated, which considerably increases computational efficiency. The differential equations were numerically solved by the Euler method,

$$q_n(t + \delta t) = q_n(t) + \delta t \cdot \dot{q}_n(t), \quad (6)$$

with a simulation time step of $\delta t = 0.001$ simulated seconds, and a random initial value of $q_n(0)$ taken from a zero-centered Gaussian distribution with standard deviation of 0.5. Since we have a large recurrent net-

[2]Here we follow Sussillo & Abbott (2009). The global time constant τ governs how fast the system responds to changes. The higher τ, the slower the response and thus the smoother the system trajectory.

work described by 1000 coupled nonlinear differential equations, more advanced integration methods than the Euler method would result in a drastically higher computational demand. The temporal discretization yields time steps $t_i = i \cdot \delta t$, so that temporal integrations are performed by replacing $\int dt\, f(t) \to \sum_i \delta t\, f(t_i)$.

The number N of neurons in the reservoir can and should be rather high because it limits the capacity of the network to learn multiple complex dynamical functions, but it should not be too high to avoid unnecessary computational demands. We have set N to 1000 which turned out to be an adequate value (see Discussion). The input dimension was set to $K = 3$ (this value is more or less arbitrary, but too low a value would compromise the separability of input vectors in input space, resulting in a bad morphing performance, see below), and the output dimension was set to $L = 5$ (corresponding to four muscle activations and one joint angle, see below). I_{nk} and F_{nl} are random matrices with components taken from a uniform distribution over the interval $[-1, 1]$. M_{nm} is a sparse random matrix[3] with a sparseness of $p = 0.01$ and non-zero weight values taken from a zero-centered Gaussian distribution with a standard deviation of

$$\sigma_M = \frac{g}{\sqrt{pN}}, \tag{7}$$

where g governs the strength of the recurrent flow in the reservoir. For $g < 1$, the network activity is damped, and all activity caused by prior input decays so that the network returns to a default idle state. Such behavior is one of the defining properties of echo state networks (the "echo state property", see Jaeger, 2001, pp 6) and of liquid state machines (the "time invariance" and "fading memory" preconditions, see Maass et al., 2002, p 7). On the other hand, FORCE networks, like the one we used, potentially exhibit and maintain self-activity (in a more general context denoted as "chaotic behavior") by allowing $g > 1$, which is responsible for the capacity of the network to generate dynamical output from static or even zero input. Interestingly, the number of cycles required to train a network to generate periodic target functions *drops* considerably as a function of g (Sussillo & Abbott, 2009, p 550), so self-activity in fact yields a benefit for the training performance. If

[3]The usage of a sparse matrix M_{mn} is both numerically more efficient and biologically more plausible, since in a biological neural network each neuron is only connected to a proper subset of other neurons (cf. Ioannides, 2007).

g is *too* high, however, the training fails to "control the chaos" and learning does not succeed any more. For our implementation we chose $g = 1.5$, which is well in the self-active regime but not too far out to push the network out of control. The scaling factor $1/\sqrt{pN}$ takes care that the total weight of outgoing connections per neuron is independent from the number of outgoing connections per neuron: There are N neurons in the reservoir, and each neuron is connected on average with pN other neurons; the sum of all weights of outgoing connections per neuron is a random variable whose variance equals the sum of the variances of the individual connections, so $\sigma_{tot}^2 = pN \cdot \sigma_M^2$, and this number is independent of the number pN of outgoing connections exactly if σ_M is chosen to be proportional to $1/\sqrt{pN}$ as in Equation 7.

The network is trained by injecting a K-dimensional input function $\boldsymbol{x}(t)$, an associated L-dimensional target function $\boldsymbol{h}(t)$, and by modifying the synaptic weights of the output neurons according to the delta rule

$$W_{ln}(t + \Delta t) = W_{ln}(t) + E_l(t) \sum_{m=1}^{N} P_{lm}(t) r_m(t), \tag{8}$$

where Δt is the learning time step set to $\Delta t = 2\delta t$, and where $E_l(t) = h_l(t) - y_l(t)$ is the output error, and where $P_{lm}(t)$ is a running estimate of the regularized inverse of the correlation matrix of the output vector, given by

$$P_{lm}(t + \Delta t) = P_{lm}(t) + \frac{\sum_{ij} P_{li}(t) r_i(t + \Delta t) r_j(t + \Delta t) P_{jm}(t)}{\sum_{ij} r_i(t + \Delta t) P_{ij}(t) r_j(t + \Delta t) + 1}, \tag{9}$$

with the initial condition $P_{lm}(0) = 1/\alpha$ and the regularization constant set to $\alpha = 1$. The update of W realizes the learning process. When learning is switched off (during the "validation phase"), the delta rule is no longer applied, the matrix W stays constant and the network freely evolves in response to the input. Learning took place *online*, that is, the target functions were trained one after the other in subsequent "lessons". One lesson consisted of feeding the network subsequently with the pairs $(\boldsymbol{x}_i, \boldsymbol{h}_i(t))$ of input vector and its associated target function, for a certain number of periods. If T_i is the duration of one period of the target function $\boldsymbol{h}_i(t)$, and n_i is the number of repetitions per lesson, then each lesson took $T = \sum_i n_i T_i$ simulated seconds (abbreviated as "sims"). For our calculations there were only two input/output pairs, the number of repetitions was globally set to $n = 6$ and the number of lessons to 8, so each lesson starts with feeding the pair $(\boldsymbol{x}_1, \boldsymbol{h}_1(t))$

for $6T_1$ sims and then feeding the next pair $(\boldsymbol{x}_2, \boldsymbol{h}_2(t))$ for $6T_2$ sims, where T_1, T_2 are the period durations of $\boldsymbol{h}_1(t), \boldsymbol{h}_2(t)$, respectively. An entire lesson thus took $T = 6T_1 + 6T_2$ sims. The *learning activity* was measured by the weight update rate defined by

$$\mathrm{WUP}(t) = \|\dot{W}(t)\| = \frac{\|W(t) - W(t - \Delta t)\|}{\Delta t}, \tag{10}$$

where the matrix norm is defined by $\|A\| = \sqrt{\max(\mathrm{eig}(A^T A))}$ with $\mathrm{eig}(A)$ being the set of eigenvalues of a matrix A. The *learning error* was measured by the distance between output and target (root mean square, RMS) averaged over the previous T sims, where T is the duration of one lesson,

$$\mathrm{RMS}(t) = \sqrt{\frac{1}{T} \int_{t-T}^{t} dt' \, \|\boldsymbol{y}(t') - \boldsymbol{h}(t')\|^2}, \tag{11}$$

and where for $t < T$ the constant T in the above formula is replaced by t with $t = 0$ at the beginning of the training phase.

As for the modulation of the stored movement patterns, we restrict our considerations here to the case of a simple linear morphing between stored patterns. Other modulations, e.g. nonlinear morphing, are certainly also possible and may further enhance the system's capacity to generate complex dynamics. The investigation of such enhanced modulation may be the topic of a future study. Here, the morphing took place by first training the network with two input/output pairs $(\boldsymbol{x}_1, \boldsymbol{h}_1(t))$ and $(\boldsymbol{x}_2, \boldsymbol{h}_2(t))$ and then feeding the network with the morphed input

$$\boldsymbol{x}(\lambda) = \boldsymbol{x}_1 + \lambda(\boldsymbol{x}_2 - \boldsymbol{x}_1), \tag{12}$$

with $\lambda = \lambda(t)$ continuously increasing from -0.25 up to 1.25 in 30 sims. All simulations and calculations have been carried out using MATLAB 7.11.0.584 (R2010b) on either a MacBook Pro with 2.5 GHz Intel Core i5 processor and 4 GB RAM running MacOS 10.6, or on a PC laptop with Intel Core i3-2310M CPU 2.1 GHz processor and 3 GB RAM running Windows 7.

3.2. Experimental data

The elementary movement patterns were represented by periodic target functions that our network had to learn; they have been chosen to correspond to selected muscle activations and kinetics of a human being who

moves one arm up and down. The muscle activations were based on experimentally measured surface electromyograms (EMGs) of one healthy and physically active male (29 years of age, 2.06 m height, 130 kg body mass, left-handed). Bipolar surface EMGs (5-700Hz; Biovision, Wehrheim, Germany) were taken from four muscles of the left arm that are involved in the movement of the elbow joint and that are accessible by surface electrodes: *m. brachioradialis*, *m. biceps brachii*, *m. triceps brachii caput longum* and *m. triceps brachii caput laterale*. EMGs were sampled at 2000 Hz (DAQCard-AI-16E-4: 12 bit, National Instruments, USA) and preamplified (bipolar 2500 times). Electrodes were positioned according to international established recommendations (Hermens et al., 1999). Disposable Ag–AgCl electrodes (H93SG, Arbo®, Germany) with a circular uptake area of 1 cm diameter and an inter-electrode distance of 2.5 cm were used. Prior to electrode application, the skin was shaved and cleaned with a special purification paste (EPICONT, Marquette Hellige GmbH Freiburg, Germany). The kinetics of the arm were measured with a three camera 3D motion capture system (200 Hz, Oqus, Qualisys, Gothenburg, Sweden). The arm was marked (reflective markers with 18 mm diameter) at the acromion, the elbow joint, and the wrist. From this the elbow joint angle was calculated using the Qualisys Track Manager. First, the activation of the elbow flexors and elbow extensors were measured during maximum voluntary contraction (MVC) for ten seconds. The subject was standing in an upright position with the elbow flexed at about 90°. Then, cyclic arm movement at three different frequencies (1 Hz, 1.5 Hz, and 2 Hz) and three different amplitudes (low, mean, large) were performed for 30 seconds, respectively. The instances of maximum elbow angles were then determined to yield the mean trajectories of the elbow kinematics and the EMGs. In a final step, the EMG data were downscaled to match the 200 Hz frequency of the kinematic data, then the data were centered, rectified, and filtered by a 4th-order zero-phase high-pass Butterworth filter with 20 Hz cutoff frequency, all cycles of one period were averaged, and lastly windows of 10 samples from the start region and the end region were cross-faded into each other to obtain smoothly periodic target functions.

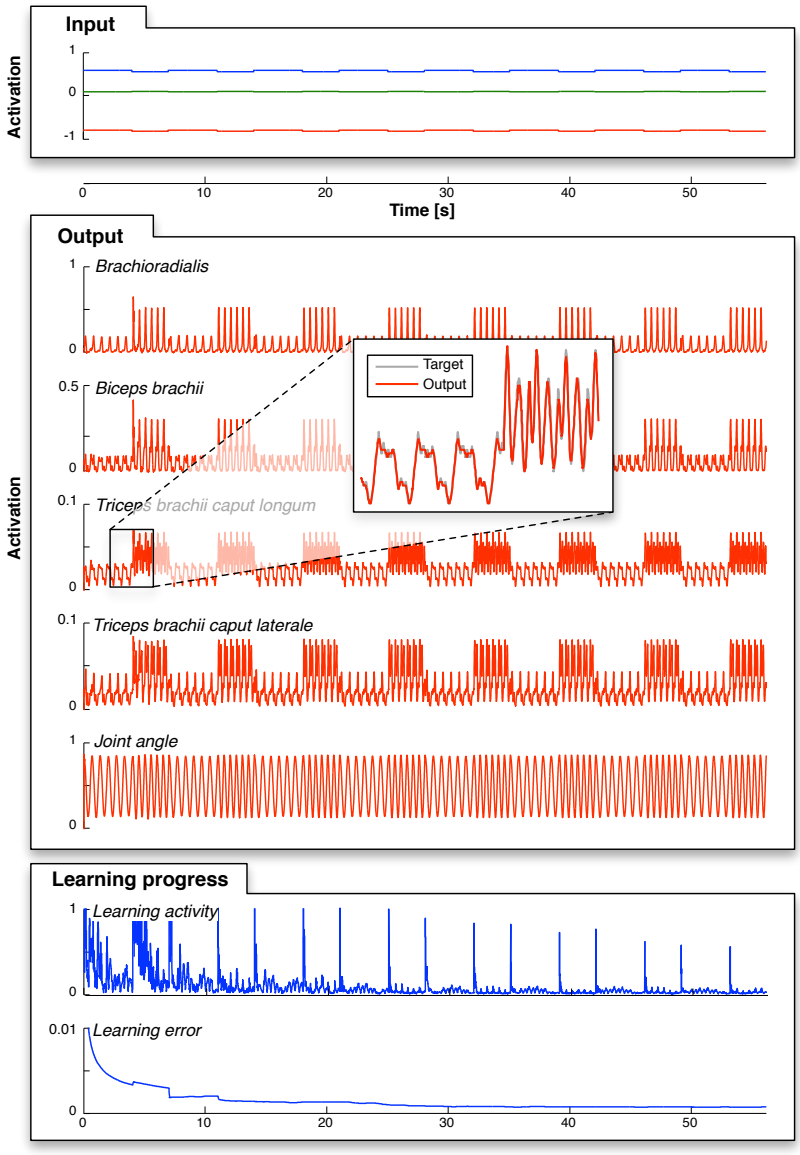

Figure 3: Training phase of the network. The top box shows the input of the network, which is an alternating set of three static firing rates. The middle box shows the output of the network, which corresponds to the muscular activations of four arm muscles and the joint angle between the upper and lower arm. The target functions (measured data) are not shown here since at the chosen scale they are virtually indistinguishable from the network output. By exemplarily zooming in, the inset shows how close the network output is to the target from the beginning on. The bottom box shows the learning progress of the network, with the update rate of the synaptic weights representing the learning activity and the distance between output and target representing the learning error.

4. Results

4.1. Training

On each run of the training sequence, the network weights were initialized with random values, as were the initial activation values of the network neurons. Each training sequence consisted of a fixed number of repetitions, and on almost every run of the training sequence the network was able to learn the generation of the target output in response to the training sequence to a satisfying degree of accuracy (see Figure 3 for a representative outcome of a successful training sequence). FORCE networks closely match their output to the target almost immediately, so the learning error in terms of the average distance between output and target (Equation 11) starts with an already small value which further decreases during training. The learning activity was measured by the update rate of the output weights of the network (Equation 10), and these weights are the only ones that are modified during learning. As can be seen in Figure 4, the network starts with a relatively strong learning activity which rapidly decreases during learning. Every time the input vector switches, the network responds with a sharp increase of learning activity which then rapidly decreases again. As a requirement for successful learning we found that the actual values of the components of the input vectors turned out to be irrelevant, except that their *overall strength* in terms of their Euclidean norm would have to be modest in order for the learning to succeed. We thus chose the strength of the input vectors to be equal to unity throughout the simulations. Also, we got best results with target functions that remained positive and did not exceed unity, which is readily fulfilled by having the EMG data normalized to maximum voluntary contraction.

4.2. Validation

During the validation phase the learning was switched off, the network weights were thus remaining constant and the system freely evolved in response to the input. Since the neuronal activity was randomly initialized, and also the time points of the switching between the two different input vectors were at random, each run of the validation phase was an independent realization with a different output. If the previous learning phase had successfully finished, the network was able to satisfyingly reproduce the learned patterns when the corresponding static input was given. In Figure 4 a representative output of a validation run is shown. The network behaved like a dynamical

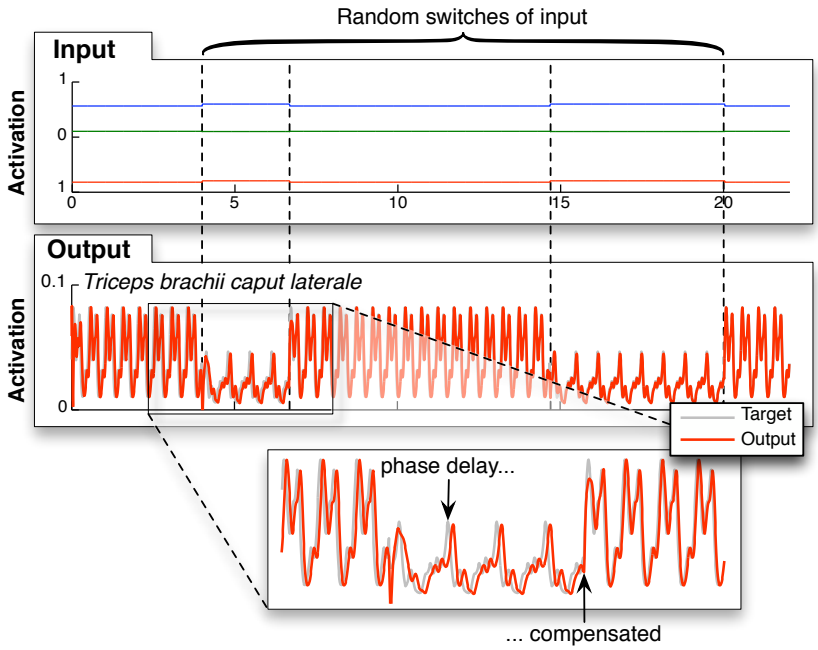

Figure 4: Validation phase of the network. The network weights are no longer modified and the network freely evolves while dynamically responding to the input signal that is switched between the two learned values at random time points. Here, only one output function corresponding to the activation of the *triceps brachii caput laterale* is exemplarily shown, with a zoomed selection displayed in the inset. It can be seen that the network acts like a dynamical system that "swings" into the adequate learned pattern as soon as the input switches. Sometimes this swinging introduces phase delays which may be compensated on a later swinging (see inset). The phase delays are due to the training input being constant and thus carrying no phase information.

system that responds to piecewise static input and "swings" into the learned output pattern when the corresponding static input is given. This "swinging" sometimes introduced phase shifts which showed up when the actual network output was compared to the target. The phase shifts are due to the control input being constant and thus carrying no phase information. To obtain a phase-locked output, the control input would have to be designed with a peak at the beginning of each cycle during training (Sussillo & Abbott, 2009). The phase-shifting behavior, though irrelevant to our considerations here, makes it unfavorable to quantify the learning success by the usual RMS measure, because it would severely punish phase delays although, in view of the just mentioned lack of phase information in the control signal, these delays do not constitute an unacceptable deviation from the target function. Altogether, after successful training the network was able to generate the correct output patterns in response to the randomly switching input, while afflicted with the already mentioned occasional phase shifts after input switching.

4.3. Morphing

After having successfully learned the output of two sets of dynamical functions $\boldsymbol{h}_1(t), \boldsymbol{h}_2(t)$ in response to two static input vectors $\boldsymbol{x}_1, \boldsymbol{x}_2$, respectively, the network was fed with a morph of the two inputs (Equation 12). During a simulation time of 30 sims, the morphing parameter λ was linearly increased from -0.25 to 1.25. The output of the network was a nearly linear interpolation between the patterns $\boldsymbol{h}_1(t)$ and $\boldsymbol{h}_2(t)$ during the central time interval $[5, 25]$, and a nearly linear *extra*polation of the patterns in the remaining two intervals at the start and the end of the simulation (Figure 5). A closer inspection showed that several visible features of the learned patterns were linearly modulated, in the amplitude domain as well as in the frequency domain (Figure 6). The global amplitude was modulated, and so were the heights of in-between local peaks of the functions and also the frequency of the nearly sinusoidal function corresponding to the joint angle. As for the local peaks it is interesting to note that some of them were *in*creased and others were *de*creased, which indicates that the modulation was not just a simple global amplification or attenuation.

The morphing capacity of the network emerged without any modification of the algorithm. We generally obtained good morphing results when the initial (unmorphed) input vectors were close to each other in terms of their Euclidean distance. In order to test the induced hypothesis that the morphing quality depends on the distance of the initial input vectors, we carried

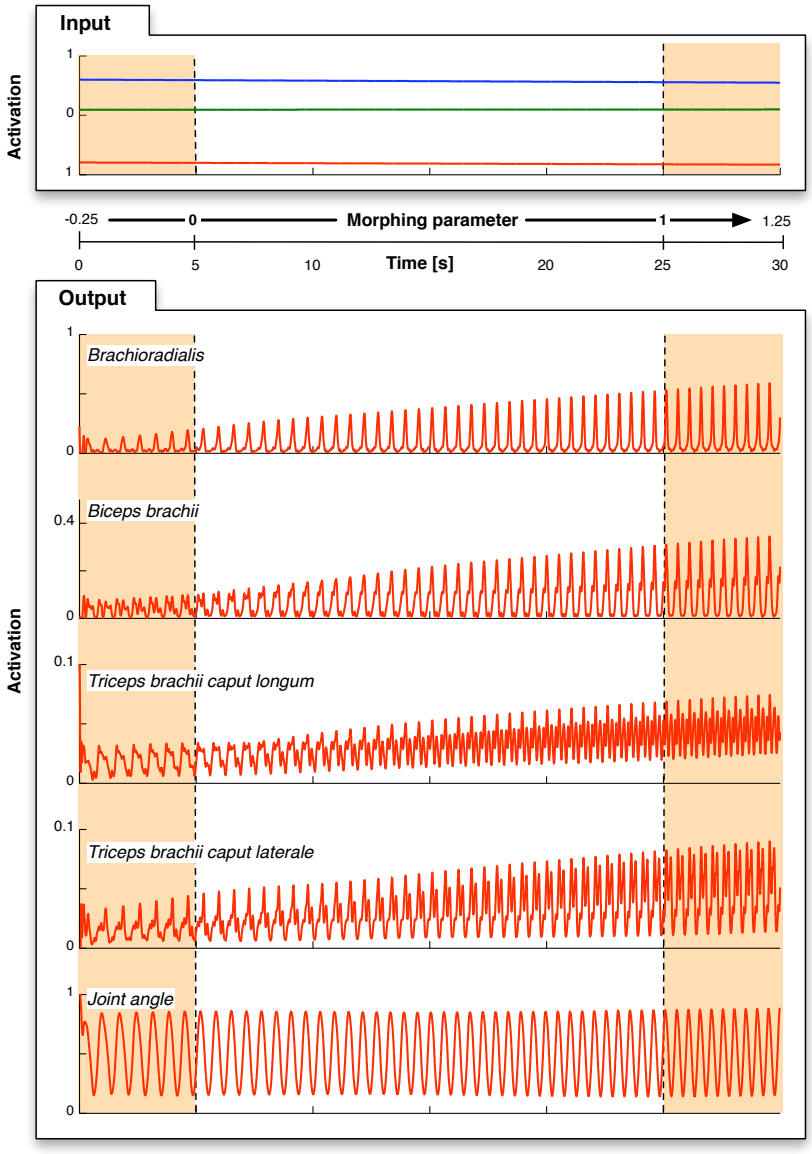

Figure 5: Demonstration of the "morphing" capacity of the network after learning. Interpolation takes place between the 5th and 25th second, when the morphing parameter λ of Equation 12 is linearly increased from 0 to 1. Extrapolation takes place in the orange colored vertical areas, during the first and last five seconds, where λ is linearly extended below 0 and above 1, respectively. The change of the input values (upper panel: red, green, blue lines) is hardly visible, because the two 3-dimensional input vectors used for learning are very close to each other in terms of their Euclidean distance, which turned out to be a necessary requirement for successful morphing.

out numerical simulations with different pairs of input vectors of varying distance. On each run of the simulation, the two input vectors were chosen from a unit sphere in the three-dimensional input space so that their total strength remained equal, and their mutual distance was varied by their enclosed angle from 0° to 180°; then the network was trained and afterwards the inputs were morphed. The morphing success was measured in terms of the root mean square (RMS) distance between the estimated frequency evolution of one of the network outputs (elbow angle) and the frequency evolution of a "perfect morph" obtained by an analytic linear superposition of the corresponding sinusoid target functions, so that a low RMS distance corresponded to a good morph. The results showed that the quality of the morph indeed increased with decreasing distance between the input vectors, down to a critical value (2°) under which the training, and thus also the morphing, failed (Figure 7). Most plausibly, this was because the input vectors became too close to each other, so that the network could not discriminate between them any more.

4.4. Neural activity

As has already been pointed out by Sussillo & Abbott (2009) in their presentation of the FORCE network architecture, the neurons in the reservoir show a chaotic spontaneous activity, which is typical for this kind of architecture because the synaptic weights admit undamped recurrent flows of activation even in the absence of input, a feature that distinguishes FORCE networks from other reservoir networks. We denote this feature as *self-activity* to better differentiate it from the more general and potentially misleading term of *chaotic behavior*. While the activity of individual neurons during idle input appears erratic and is of a considerably high amplitude, the total network output is similar to a low-level random noise floor. (See Figure 8 for the activity of four randomly selected neurons and one dimension of the network output). During training with a constant nonzero control input, the activity of the neurons becomes synchronized and yields a total output which is close to the periodic target function. After training, when the input is set to zero again, the neurons return to erratic activity and the total network output is noisy, but this time the amplitude of the output noise is evidently stronger and contains strong low-frequent components. As soon as the same constant input is applied as during training, the neurons start to synchronize again and the network output yields the learned periodic function. It should be remarked that we have not trained the network to yield zero output on zero input, which would correspond to the situation in the human and animal mo-

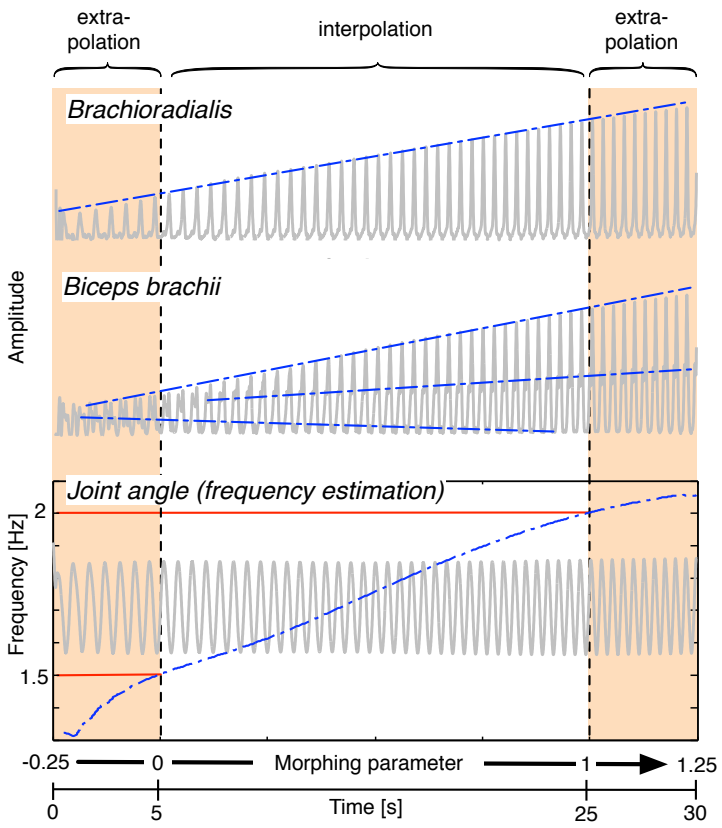

Figure 6: Closer look at selected morphs taken from Figure 5. The network interpolates and extrapolates several features of the learned functions (blue dotted lines) both in the amplitude domain (first and second panel) and the frequency domain (third panel; the blue frequency course has been numerically estimated from the data that are displayed in light gray for orientation only). The morphing of the features is fairly linear.

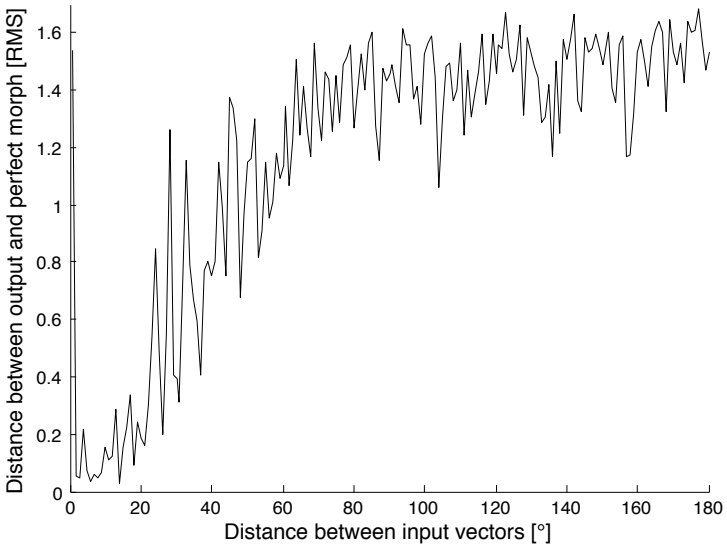

Figure 7: Distance (RMS measure, see text) between network output and a perfect linear morph of two particular target functions as a function of the distance between the two normalized input vectors (in angular degrees, see text). For each step on the x-axis, a numerical simulation was run.

tor system where it is desirable to suppress involuntary random movements. Although it is of course possible to train the network this way, we refrained from such a procedure to demonstrate what the network would do *by itself*, that is, without special training, in the absence of input. Such a scenario mimics periods of idle input in highly interconnected parts of the brain, not necessarily only in the motor system.

While it is not the focus of this study to analyze these phenomena in more detail, we believe they are worth reporting and we would suggest to dedicate future research on their investigation. A better understanding of this kind of behavior, which is characteristic for self-active recurrent networks (and for these networks only), may yield insights into the origin and meaning of the *resting-state activity* found in the brains of humans and animals during periods of idle tasking (cf. Biswal et al., 1995; Laufs et al., 2003; Mazzoni et al., 2007; Smith, 2012).

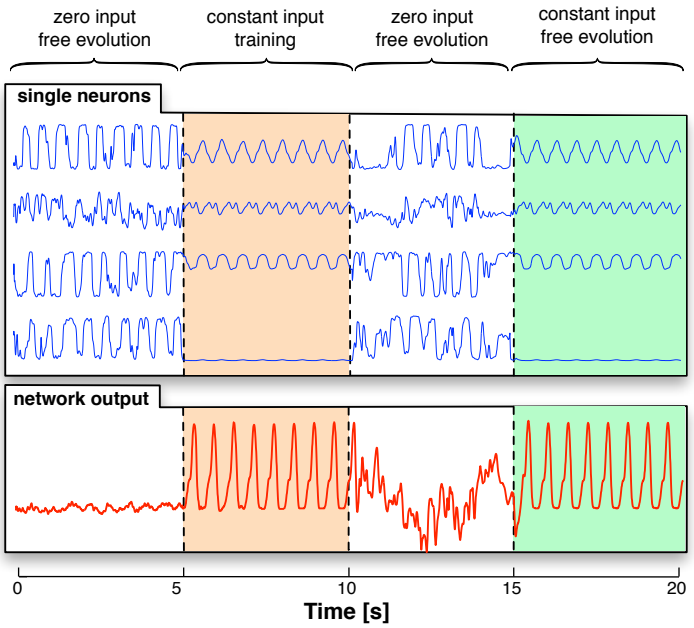

Figure 8: Neural activity of four randomly selected neurons (upper panel), and one dimension of overall network output (lower panel) during twenty simulated seconds. In the first five seconds there is no input and no training, so the network evolves freely, its neurons show erratic resting activity, and the total output is close to low-level random noise. During subsequent five seconds of training with a constant input, the neurons synchronize their activity and contribute to a network output which is close to the periodic target function. After the training and with idle input, the network evolves freely again, the neurons return to erratic resting activity, and the total output is noisy, although evidently of higher amplitude and with low-frequency components. In the final five seconds, when the same constant input as during training is applied, the network synchronizes again to yield an output close to the trained periodic function.

5. Discussion

Based on recent recurrent network architecture, we have proposed and numerically implemented a model for a flexible motor memory that is capable of storing elementary movement patterns as motor primitives into the static synaptic weights of the network. The model is capable of retrieving the stored primitives by simple static commands, and it is moreover capable of modulating them by linear inter- and extrapolation. Although we have concentrated so far on the linear combination of just two motor primitives, it is tempting to consider larger numbers of primitives. Consider M primitives $\boldsymbol{y}_i(t)$ which are stored in the network and retrieved by associated static inputs $\boldsymbol{x}_i$, where $i = 1, \ldots, M$. If the network is fed with a linear combination of inputs,

$$\boldsymbol{x} = \sum_{i=1}^{M} \lambda_i \boldsymbol{x}_i, \tag{13}$$

then it would be expected to generate an output approximately equal to a linear combination of the stored primitives,

$$\boldsymbol{y}(t) \approx \sum_{i=1}^{M} \lambda_i \boldsymbol{y}_i(t), \tag{14}$$

resulting in a large space of possible movements. Since the network is also capable of performing some limited *extra*polation, the weights λ_i might even go beyond the usual range of $[0, 1]$, resulting in an even larger space of possible movements. There might be other, nonlinear superpositions possible both in biological and artificial networks. However, the conception of linear superposition of primitive movement patterns fits well to established theories of motor control (Wolpert et al., 2011) and to empirical data (d'Avella et al., 2006).

Training a recurrent network of 1000 neurons even on a supercomputer in reasonable time was way out of reach before the advent of reservoir computing ten years ago, due to the lack of efficient training algorithms. Considering this, it is remarkable that the calculations for our model have been carried out on ordinary modern laptops, and the simulation time was about equal to real time: for example, 56 simulated seconds of network training took 74 seconds of real time. Also, it is amazing how fast the FORCE network is learning in terms of "lessons", that is, of subsequent presentation blocks of

input/output pairs. The learning of five periodic target functions in parallel, four of which were EMG signals of rather irregular shape, took only 8 lessons with 6 repetitions of each input/output pair per lesson.

As one can see in Figure 3, the learning activity abruptly increases when a new lesson starts, but then rapidly decreases again to an even lower level than before. In a manner of speaking, it looks as if the network was "surprised" by a new input/output relation and responds with a strong and broad burst of learning activity, until it "remembers" already learned input/output relations, becomes less and less "surprised" and responds with shorter and smaller bursts of learning activity. A similar bursting activity in response to novel stimuli is also known to occur in the biological brain (cf. Rutishauser et al., 2006).

One may wonder why the network does not *overwrite* the already learned patterns on entering the next learning lesson. It seems that there is just enough abstract space spanned by 1000 neurons and their interconnections to store the dynamical information needed for the given learning task with the probability of overwriting already stored information being sufficiently small. The phenomenon of non-overwriting already stored information is very interesting in the context of biological learning as it would make it unnecessary to allocate new memory space for each learned movement. There would be no need for a special location for each particular stored movement; rather, all movements could be stored within one and the same neural network across its synaptic weights in a non-local fashion.

A biological network of thousand neurons, to give an impression, would represent about ten times the size of a cortical column, which would amount to about 75x75 μm of cortical surface (Mountcastle, 1997). In the cerebellum, the same amount of neurons would occupy a volume of below 0.001 mm^3 (Lange, 1975). When we reduced the network size to 500 neurons, the training, validation, and morphing performance dropped considerably, although this could be compensated partly (but not fully) by increasing the number of repetitions and lessons. The actual number 1000 is of no fundamental significance, though, as we cannot expect that a real-world human motor system is exactly designed like a FORCE network. In particular, the synaptic weights in a biological network are most probably not fixed and random but are also subject to adaptation and optimization. Note that the assumption of a completely random reservoir network is the *weakest possible assumption*. Any additional structure might further enhance the capacities of the network to generate complex dynamical behavior. For example, it

is a topic in contemporary research to implement self-organizing dynamics (Lazar et al., 2009) or other optimizations (Wyffels et al., 2008; Wyffels & Schrauwen, 2009; Jaeger et al., 2012) into recurrent neural networks. Also, it would be interesting to implement a *small-world topology* (Watts & Strogatz, 1998) which has been shown to match the connection characteristics of the biological brain (Ioannides, 2007).

Which anatomical structures might come into question as functional realizers of our model? The selection of adequate motor behavior involves conscious decisions and is realized by cortical structures, particularly involving the prefrontal cortex (Koechlin et al., 1999; Haynes et al., 2007) and the supplementary motor area (Libet, 1985; Soon et al., 2008), the latter being the one that generates pre-conscious readiness potentials. The activity of these regions would result in an abstract action command that is sent to the reservoir in our model to constitute the *input*. The *output* of the reservoir directly goes to the muscles. Now where does the *target function* come from? A good candidate would be the joint system of *premotor and motor cortex*. The premotor cortex is a region that prepares goal-specific motor actions and hosts the mirror neuron system (Rizzolatti & Craighero, 2004). It is a region, thus, where both the generation and recognition of goal-specific movement patterns is performed. The *motor cortex* has a roughly somatotopic organization (Penfield & Boldrey, 1937; Schieber, 2001) and further processes the signals from the premotor cortex into concrete motor commands dedicated to individual muscles. According to our model, these motor commands would not directly activate the muscles. Rather, the motor commands would be sent to the reservoir as a *target function*, and the reservoir would then generate the final muscle activation signals by closely matching its output to the received target function. This way, the reservoir would be able to learn the association of each target function with the corresponding abstract action command which it receives as input, and eventually it would be able to generate the correct target function by itself in response to the received action command.

What, lastly, might be the anatomical location of the *reservoir*? It is known that electrical stimulation of the motor cortex causes monkeys to make coordinated, complex movements (Graziano et al., 2002). This does not imply, however, that the motor cortex must also be the place where the patterns are stored. According to our model, complex movement patterns are recalled in functionally lower-level structures by the same higher-level action commands that once (repeatedly) triggered the movement. Hence it

would be expected that by stimulating regions in the motor cortex, complex movement patterns stored in the lower-level structures, wherever they are located, are accidentally triggered. We believe that there is not one unique location for the *reservoir*, but rather that throughout the entire central nervous system there are regions that functionally correspond to a reservoir in the sense of our model. The already mentioned cortical regions involved in the generation of movement patterns might themselves include substructures that store these patterns. Another place might be the *cerebellum* which is assumed to be responsible for the modulation of movement patterns and to be actively involved in motor learning in a manner similar to that envisaged in our model (Marr, 1969; Albus, 1971; Thach et al., 1992; Contreras-Vidal et al., 1997; Boyden et al., 2004; Wolpert et al., 2011). The capacities of the cerebellum, though, are not restricted to motor tasks but extend also to cognitive, emotional, and language functions (Leiner et al., 1993; Timmann et al., 2010). According to the modern view, the cerebellum is involved in all sorts of tasks that require supervised learning, while the basal ganglia and the cerebral cortex are involved in reinforcement learning and unsupervised learning, respectively (Doya, 2000). Since our model involves *supervised* learning, this may point to the cerebellum rather than to cortical structures. Also, the *spinal cord* may host reservoirs which then would take the role of *central pattern generators* (CPGs). It is known that animal locomotion is to a large extent directly caused by activity of CPGs (Brown, 1911; Grillner, 2006; Ijspeert, 2008), and the same may also hold for humans (Duysens & Van de Crommert, 1998; Dietz, 2003). Moreover, there is evidence that the CPGs are subject to neuroplasticity (Raineteau & Schwab, 2001; Scivoletto et al., 2007); they are able to re-learn movement patterns, for instance with the help of robotic-assisted locomotor training, so that patients with incomplete spinal injury may learn to walk again (Mehrholz et al., 2008).

It should be mentioned that network-based theories of motor learning are certainly not new. There is the pioneering work of Marr (1969) and Albus (1971) who independently proposed a theory of the cerebellum as a movement pattern generation and recognition device; Albus specifically presented a *Perceptron* network architecture as a model for the cerebellum. Contreras-Vidal et al. (1997) proposed an integrative neural model of cerebellar learning for arm-movement control which involves cortex, cerebellum and central pattern generators. As a further development, Grossberg & Pearson (2008) proposed the LIST PARSE model as a unified model of motor learning and motor working memory which involves many cortical structures such as the

prefrontal cortex, the sensory cortices and the amygdala. Sussillo & Abbott (2009), the inventors of the FORCE network, applied their techniques to the kinetics, though not the dynamics, of human movements. A FORCE network has also been used to realize a brain-machine interface (BMI) decoder for reaching movements of monkeys (Sussillo et al., 2012). Wyffels & Schrauwen (2009) and Jaeger et al. (2012) modeled central pattern generators as reservoir networks, but they put their results more into the context of robot locomotion. Our model of a flexible motor memory differs from the mentioned models in that it is intended to be a simplified and idealized model for a small-scale biological network of neurons to store and recall movement patterns, and in that it involves a self-active recurrent neural network that is capable of generating previously unlearned dynamical output from linearly superposed static input without additional specific optimization.

So far, our model is *open-loop* only, as there is no feedback from the body and its environment that affects the system's behavior. An enhanced model that would enable closed-loop control would require a numerical simulation of the interaction between network, body, and environment. It would be a promising matter of future research to investigate how a suitably enhanced model of a flexible motor memory performs when endowed with a properly simulated body and environment, thus when it becomes an *embodied* system. The gained insights might lead to concrete applications in robotics, prosthetics, and rehabilitation from stroke or spinal chord injury.

6. Acknowledgements

We thank the reviewers for valuable comments that helped to further improve this work which is supported by the german Federal Ministry of Education and Research (BMBF) [01EC1003A].

Albus, J. S. (1971). A theory of cerebellar function. *Mathematical Biosciences*, *10*, 25–61.

Atkinson, R., & Shiffrin, R. (1968). Human memory: A proposed system and its control processes. *The psychology of learning and motivation: Advances in research and theory*, *2*, 89–195.

Biswal, B., Zerrin Yetkin, F., Haughton, V. M., & Hyde, J. S. (1995). Functional connectivity in the motor cortex of resting human brain using echo-planar mri. *Magnetic Resonance in Medicine*, *34*, 537–541.

Book, W. F. (1908). *The psychology of skill, with special reference to its acquisition in typewriting*. Missoula : University of Montana.

Boyden, E. S., Katoh, A., & Raymond, J. L. (2004). Cerebellum-dependent learning: the role of multiple plasticity mechanisms. *Annual Review of Neuroscience*, *27*, 581–609.

Brown, T. (1911). The intrinsic factors in the act of progression in the mammal. *Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character*, *84*, 308–319.

Burrows, T., & Niranjan, M. (1995). Vocal tract modelling with recurrent neural networks. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on* (pp. 3315 –3318 vol.5). volume 5.

Cheron, G., Cebolla, A. M., Bengoetxea, A., Leurs, F., & Dan, B. (2007). Recognition of the physiological actions of the triphasic emg pattern by a dynamic recurrent neural network. *Neurosci Lett*, *414*, 192–6.

Contreras-Vidal, J. L., Grossberg, S., & Bullock, D. (1997). A neural model of cerebellar learning for arm movement control: cortico-spino-cerebellar dynamics. *Learn Mem*, *3*, 475–502.

d'Avella, A., Portone, A., Fernandez, L., & Lacquaniti, F. (2006). Control of fast-reaching movements by muscle synergy combinations. *The Journal of Neuroscience*, *26*, 7791–7810.

Dietz, V. (2003). Spinal cord pattern generators for locomotion. *Clinical Neurophysiology*, *114*, 1379 – 1389.

Doya, K. (2000). Complementary roles of basal ganglia and cerebellum in learning and motor control. *Current Opinion in Neurobiology*, *10*, 732 – 739.

Draye, J. P., Cheron, G., Bourgeois, M., Pavisic, D., & Libert, G. (1997). Improved identification of complex temporal systems with dynamic recurrent neural networks. application to the identification of electromyography and human arm trajectory relationship. *Journal of Intelligent Systems Special Issue on Neural Networks Applications*, *7*, 83–102.

Duysens, J., & Van de Crommert, H. W. A. A. (1998). Neural control of locomotion; part 1: The central pattern generator from cats to humans. *Gait & Posture*, *7*, 131–141.

Fernando, C., & Sojakka, S. (2003). Pattern recognition in a bucket. In W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, & J. T. Kim (Eds.), *Advances in Artificial Life* chapter 63. (pp. 588–597). Berlin, Heidelberg: Springer Berlin / Heidelberg volume 2801 of *Lecture Notes In Computer Science*.

Graziano, M. S. A., Taylor, C. S. R., Moore, T., & Cooke, D. F. (2002). The cortical control of movement revisited. *Neuron*, *36*, 349–362.

Grillner, S. (2006). Biological pattern generation: the cellular and computational logic of networks in motion. *Neuron*, *52*, 751–66.

Grossberg, S. (1978). Behavioral contrast in short term memory: Serial binary memory models or parallel continuous memory models? *Journal of Mathematical Psychology*, *17*, 199 – 219.

Grossberg, S., & Paine, R. (2000). A neural model of cortico-cerebellar interactions during attentive imitation and predictive learning of sequential handwriting movements. *Neural Networks*, *13*, 999 – 1046.

Grossberg, S., & Pearson, L. R. (2008). Laminar cortical dynamics of cognitive and motor working memory, sequence learning and performance: Toward a unified theory of how the cerebral cortex works. *Psychological Review*, *115*, 677 – 732.

Haynes, J.-D., Sakai, K., Rees, G., Gilbert, S., Frith, C., & Passingham, R. E. (2007). Reading hidden intentions in the human brain. *Curr Biol*, *17*, 323–328.

Hermens, H., Freriks, B., Merletti, R., Stegeman, D. F., Blok, J., Rau, G., & Disselhorst-Klug, C. (1999). *European Recommendations for Surface ElectroMyoGraphy, results of the SENIAM project.*. Roessingh: Roessingh Research and Development b.v.

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Netw*, *21*, 642–53.

Ioannides, A. A. (2007). Dynamic functional connectivity. *Curr Opin Neurobiol*, *17*, 161–70.

Jaeger, H. (2001). *The "echo state" approach to analysing and training recurrent neural networks*. GMD Report 148 German National Research Center for Information Technology.

Jaeger, H., Ajallooeian, M., Billard, A., Schack, T., Reinhart, F., & Wyffels, F. (2012). Technical report on dynamic extensibility methods: Amarsi deliverable d6.2.

Koechlin, E., Basso, G., Pietrini, P., Panzer, S., & Grafman, J. (1999). The role of the anterior prefrontal cortex in human cognition. *Nature*, *399*, 148–151.

Lange, W. (1975). Cell number and cell density in the cerebellar cortex of man and some other mammals. *Cell Tissue Res*, *157*, 115–24.

Laufs, H., Krakow, K., Sterzer, P., Eger, E., Beyerle, A., Salek-Haddadi, A., & Kleinschmidt, A. (2003). Electroencephalographic signatures of attentional and cognitive default modes in spontaneous brain activity fluctuations at rest. *Proceedings of the National Academy of Sciences*, *100*, 11053–11058.

Lazar, A., Pipa, G., & Triesch, J. (2009). Sorn: a self-organizing recurrent neural network. *Frontiers in Computational Neuroscience*, *3*.

Leiner, H. C., Leiner, A. L., & Dow, R. S. (1993). Cognitive and language functions of the human cerebellum. *Trends in Neurosciences*, *16*, 444 – 447.

Libet, B. (1985). Volitional processes (planned, spontaneous and conscious) in relation to the sma. *Behavioral and Brain Sciences*, *8*, 592–594.

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*, 127–149.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*, 2531–2560.

Marr, D. (1969). A theory of cerebellar cortex. *The journal of physiology*, *202*, 437.

Mazzoni, A., Broccard, F. D., Garcia-Perez, E., Bonifazi, P., Ruaro, M. E., & Torre, V. (2007). On the dynamics of the spontaneous activity in neuronal networks. *PLoS One*, *2*, e439.

Mehrholz, J., Kugler, J., & Pohl, M. (2008). Locomotor training for walking after spinal cord injury. *Cochrane Database Syst Rev*, *2*.

Mountcastle, V. B. (1997). The columnar organization of the neocortex. *Brain*, *120*, 701–722.

Penfield, W., & Boldrey, E. (1937). Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation. *Brain*, *60*, 389–443.

Raineteau, O., & Schwab, M. (2001). Plasticity of motor systems after incomplete spinal cord injury. *Nature Reviews Neuroscience*, *2*, 263–273.

Rhodes, B. J., Bullock, D., Verwey, W. B., Averbeck, B. B., & Page, M. P. A. (2004). Learning and production of movement sequences: behavioral, neurophysiological, and modeling perspectives. *Hum Mov Sci*, *23*, 699–746.

Rizzolatti, G., & Craighero, L. (2004). The mirror-neuron system. *Annu Rev Neurosci*, *27*, 169–92.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation.

Rutishauser, U., Mamelak, A. N., & Schuman, E. M. (2006). Single-trial learning of novel stimuli by individual neurons of the human hippocampus-amygdala complex. *Neuron*, *49*, 805 – 813.

Schieber, M. H. (2001). Constraints on somatotopic organization in the primary motor cortex. *Journal of Neurophysiology*, *86*, 2125–2143.

Schmidt, R. A. (1975). A schema theory of discrete motor skill learning. *Psychological Review*, *82*, 225–260.

Schöllhorn, W. (1999). Individualität - ein vernachlässigter Parameter? *Leistungssport*, *29*, 7–11.

Scivoletto, G., Ivanenko, Y., Morganti, B., Grasso, R., Zago, M., Lacquaniti, F., Ditunno, J., & Molinari, M. (2007). Review article: Plasticity of spinal centers in spinal cord injury patients: New concepts for gait evaluation and training. *Neurorehabilitation and neural repair*, *21*, 358.

Smith, K. (2012). Neuroscience: Idle minds. *Nature*, *489*, 356–8.

Soon, C. S., Brass, M., Heinze, H.-J., & Haynes, J.-D. (2008). Unconscious determinants of free decisions in the human brain. *Nat Neurosci*, *11*, 543–545.

Sussillo, D., & Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, *63*, 544–557.

Sussillo, D., Nuyujukian, P., Fan, J. M., Kao, J. C., Stavisky, S. D., Ryu, S., & Shenoy, K. (2012). A recurrent neural network for closed-loop intracortical brain–machine interface decoders. *Journal of Neural Engineering*, *9*, 026027.

Thach, W. T., Goodkin, H. P., & Keating, J. G. (1992). The cerebellum and the adaptive coordination of movement. *Annual Review of Neuroscience*, *15*, 403–442.

Timmann, D., Drepper, J., Frings, M., Maschke, M., Richter, S., Gerwig, M., & Kolb, F. (2010). The human cerebellum contributes to motor, emotional and cognitive associative learning. a review. *Cortex*, *46*, 845 – 857.

Verwey, W. B., & Dronkert, Y. (1996). Practicing a structured continuous key-pressing task: Motor chunking or rhythm consolidation? *Journal of Motor Behavior*, *28*, 71–79. PMID: 12529225.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*, 440–2.

Wolpert, D. M., Diedrichsen, J., & Flanagan, J. R. (2011). Principles of sensorimotor learning. *Nat Rev Neurosci*, *12*, 739–751.

826 Wyffels, F., & Schrauwen, B. (2009). Design of a central pattern generator
827 using reservoir computing for learning human motion. In *Proceedings of*
828 *the 2009 Advanced Technologies for Enhanced Quality of Life* AT-EQUAL
829 '09 (pp. 118–122). Washington, DC, USA: IEEE Computer Society.

830 Wyffels, F., Schrauwen, B., & Stroobandt, D. (2008). Stable output feedback
831 in reservoir computing using ridge regression. In V. Kurková, R. Neruda,
832 & J. Koutník (Eds.), *Artificial Neural Networks - ICANN 2008* (pp. 808–
833 817). Springer Berlin / Heidelberg volume 5163 of *Lecture Notes in Com-*
834 *puter Science*.

835 Wymbs, N. F., Bassett, D. S., Mucha, P. J., Porter, M. A., & Grafton,
836 S. T. (2012). Differential recruitment of the sensorimotor putamen and
837 frontoparietal cortex during motor chunking in humans. *Neuron*, *74*, 936
838 – 946.