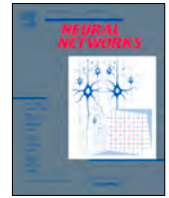




HOKKAIDO UNIVERSITY

Title	Learning long-term motor timing/patterns on an orthogonal basis in random neural networks
Author(s)	Kawai, Yuji; Park, Jihoon; Tsuda, Ichiro et al.
Citation	Neural Networks, 163, 298-311 https://doi.org/10.1016/j.neunet.2023.04.006
Issue Date	2023-04-12
Doc URL	https://hdl.handle.net/2115/98682
Rights	©2023 The Author(s).
Rights(URL)	https://creativecommons.org/licenses/by/4.0/
Type	journal article
File Information	Neural Networks_163_2023.pdf





Learning long-term motor timing/patterns on an orthogonal basis in random neural networks

Yuji Kawai ^{a,*}, Jihoon Park ^{a,b}, Ichiro Tsuda ^c, Minoru Asada ^{a,b,c,d}

^a Symbiotic Intelligent Systems Research Center, Institute for Open and Transdisciplinary Research Initiatives, Osaka University, 1–1 Yamadaoka, Suita, Osaka 565–0871, Japan

^b Center for Information and Neural Networks, National Institute of Information and Communications Technology, 1–4 Yamadaoka, Suita, Osaka 565–0871, Japan

^c Chubu University Academy of Emerging Sciences/Center for Mathematical Science and Artificial Intelligence, Chubu University, 1200 Matsumoto-cho, Kasugai, Aichi 487–8501, Japan

^d International Professional University of Technology in Osaka, 3–3–1 Umeda, Kita-ku, Osaka 530–0001, Japan

ARTICLE INFO

Article history:

Received 22 July 2022

Received in revised form 15 March 2023

Accepted 5 April 2023

Available online 12 April 2023

Keywords:

Reservoir computing

Motor timing

Random neural network

Orthogonal basis

Modular neural network

ABSTRACT

The ability of the brain to generate complex spatiotemporal patterns with specific timings is essential for motor learning and temporal processing. An approach that can model this function, using the spontaneous activity of a random neural network (RNN), is associated with orbital instability. We propose a simple system that learns an arbitrary time series as the linear sum of stable trajectories produced by several small network modules. New finding in computer experiments is that the trajectories of the module outputs are orthogonal to each other. They created a dynamic orthogonal basis acquiring a high representational capacity, which enabled the system to learn the timing of extremely long intervals, such as tens of seconds for a millisecond computation unit, and also the complex time series of Lorenz attractors. This self-sustained system satisfies the stability and orthogonality requirements and thus provides a new neurocomputing framework and perspective for the neural mechanisms of motor learning.

© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The brain generates a large variety of spatiotemporal motor patterns based on both spontaneous and input-driven activity, which is an essential ability to accomplish various tasks, such as motor learning including motor timing. Motor timing is a task of actively generating motions, such as pressing a lever or blinking, at a given interval ranging from tens of milliseconds to tens of seconds after an onset cue, for which the basal ganglia and cerebellum are mainly responsible (Paton & Buonomano, 2018; Petter, Lusk, Hesslow, & Meck, 2016). Although whether motor timing relies on dedicated (specialized) neural mechanisms (Church, Mech, & Gibbon, 1994; Miall, 1989) or the intrinsic (general) ability of neural networks (Buonomano & Laje, 2011; Laje & Buonomano, 2013) is an ongoing debate (Buhusi & Meck, 2005; Ivry & Schlerf, 2008; Mauk & Buonomano, 2004), we take the latter position because the involved brain areas also contribute to general motor pattern learning.

To accomplish the motor timing task using a biologically plausible RNN, that is, a *reservoir computing* approach (Jaeger, 2001;

Jaeger & Haas, 2004; Lukoševičius & Jaeger, 2009), the neural activity induced by the onset signal must be sustained within the networks during the interval without any inputs (Buonomano & Laje, 2011; Laje & Buonomano, 2013; Renart, 2013). In addition, a set of time series of neural activity should have sufficient representational capacity to synthesize arbitrary time series. In this respect, the production of an orthogonal basis of neural activity in that learning process brings about high efficiency of reservoir computers. If large recurrent weights change the RNN into the high-gain regime, a neural unit spontaneously activates itself even without any external input. However, the self-sustained neural activity shows chaoticity that implies orbital instability in which slight differences in initial conditions are magnified exponentially in time, and noise can also be magnified (for an exceptional case, see Galatolo, Monge, & Nisoli, 2020; Matsumoto & Tsuda, 1983) (Sompolsky, Crisanti, & Sommers, 1988), resulting in unsuccessful learning. In contrast, the activity in low-gain RNNs decays to zero in the absence of an input.

Several methods have been proposed to suppress the chaoticity of RNNs (Deng & Zhang, 2007; Kawai, Park, & Asada, 2019; Yamaguti & Tsuda, 2021), such as applying output feedback to units (Boccaletti, Grebogi, Lai, Mancini, & Maza, 2000; Sussillo & Abbott, 2009). These models, which rely on the feedback or input signals to stabilize neural networks or to sustain neural network

* Corresponding author.

E-mail address: kawai@otri.osaka-u.ac.jp (Y. Kawai).

activity, cannot learn the motor timing because of an interval with little, or no feedback or input. Laje and Buonomano (2013) proposed an *innate training* approach as a model of motor timing, where each unit in a high-gain RNN was trained to output its own previous noiseless trajectory even when their initial conditions were different, and noise was added. This training results in the robust generation of stable complex neural activity, which enables the network to learn the timing of a few seconds. However, this approach lacks biological plausibility in terms of how the neural system obtains, stores, and learns its noiseless trajectories. In addition, it is difficult to learn time series longer than tens of seconds because the duration that can suppress orbital instability is limited. Furthermore, the computational cost of innate training is very high compared to that of the standard reservoir computing approach because it modulates the recurrent weights.

Therefore, instead of an approach that suppresses chaos with training, we adopt an approach that places the RNN in an intermediate state between a stable equilibrium state converging to zero and a chaotic state. Some seminal studies on the dynamics of RNNs have reported that for a finite network size N , there is a parameter region in which the network activity exhibits stable intermediate dynamics, that is, limit cycles and tori, and such regions expand with a decrease in N (Doyon, Cessac, Quoy, & Samuelides, 1994; Sompolinsky et al., 1988). Hence, non-chaotic stable self-sustained trajectories can be obtained when N is small, and the network gain g is large. However, such small-sized networks have fewer degrees of freedom in their activity, and their unit outputs tend to be more similar to each other, that is, they are less orthogonal.

In this study, to overcome the trade-off between orbital stability and orthogonality, we propose a simple and computationally powerful system in which several small RNNs are connected as modules independently and in parallel. This configuration enables the system to achieve both stability and orthogonality. First, we conform the trade-off in a single RNN and then evaluate the stability and orthogonality of the modular RNNs. The local Lyapunov exponents of the module outputs and the inner products between the time series of the module outputs are computed to confirm the system's low instability and its high orthogonality, respectively. Because the proposed system creates a dynamical orthogonal basis, we refer to it as a *reservoir of basal dynamics* (reBASICS). We then evaluate the learning performance of reBASICS in the two motor timing tasks and verify that it can generate a more complex time series, such as the Lorenz system. Finally, we show that feeding the system multiple inputs allows the network activity to vary flexibly in response to the inputs, resulting in an enhanced computational capability.

Although reBASICS can learn complex chaotic time series, it does not represent the dynamical system behind the time series but reproduces the time series with a combination of the orthogonal bases. Therefore, the generalization capability and noise robustness of reBASICS are limited. Nevertheless, because of its simplicity and high purity, reBASICS is highly interpretable and extensible; therefore, it provides a new perspective on the neural mechanisms of motor pattern learning and also a neurocomputational framework.

2. Methods

2.1. reBASICS

Fig. 1 shows the configuration of reBASICS. A module consists of N randomly connected firing-rate units (see Section 2.1.1). All units in a module receive an input signal and inputs from other units in the same module and then update their states. Recurrent connectivity is drawn from a standard normal distribution with

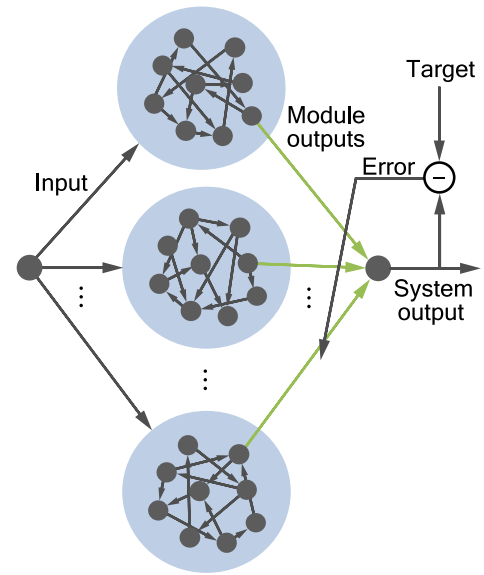


Fig. 1. Reservoir of basal dynamics (reBASICS). reBASICS consists of a number of RNN modules. The system output is synthesized by the module outputs, and the output (readout) weights are modulated with recursive least squares (green arrows).

a gain parameter g , which controls the chaoticity of the network activity. Self-sustained limit cycle activity is generated by setting $g \geq 1$ in each module, and a small N of the modules can suppress the chaoticity of their activity. The output signal of a unit that is randomly chosen from each module is considered to be the module output, and the system output is obtained as a linear sum (readout) of the module outputs (green arrows in Fig. 1). To minimize the error between the system output and target signal in an online manner, only the readout weights are trained with recursive least squares (Haykin, 2009).

reBASICS synthesizes target time series from many limit cycles. Ideally, the time series of module outputs should be orthogonal to each other, i.e., create an orthogonal basis to replicate an arbitrary time series. This mechanism is similar to the Fourier transform (for differentiable time series) and also to the wavelet transform (for time series containing nondifferentiable parts), which use trigonometric functions as orthogonal bases to synthesize time series. In reBASICS, the randomness of recurrent weights in the modules is expected to contribute to the generation of diverse orthogonal trajectories.

2.1.1. Network dynamics

The dynamics of a state $x_{k,i}(t)$ of unit i ($i = 1, 2, \dots, N$) in module k ($k = 1, 2, \dots, M$) at time t are given as a firing-rate model using the hyperbolic tangent activation function:

$$\tau \frac{dx_{k,i}(t)}{dt} = -x_{k,i}(t) + \sum_{j=1}^N W_{k,ij}^{\text{Rec}} r_{k,j}(t) + W_{k,i}^{\text{In}} u(t) + I_{k,i}^{\text{noise}}, \quad (1)$$

$$r_{k,i}(t) = \tanh(x_{k,i}(t)), \quad (2)$$

where τ denotes a time constant; N denotes the network size, that is, the number of units in a module; $u(t)$ denotes input signal; in the following, i and j denote units, k denotes a module; $W_{k,ij}^{\text{Rec}}$, $W_{k,i}^{\text{In}}$, $I_{k,i}^{\text{noise}}$, $r_{k,i}(t)$ denote recurrent weights from j to i , input weights to i , the noise added to i , and the firing rate of i , respectively.

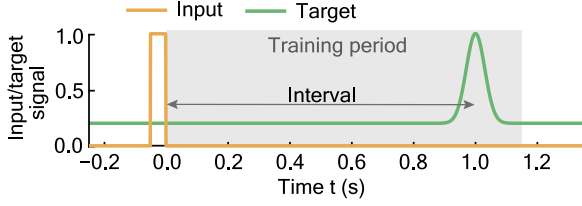


Fig. 2. Motor timing task. In this task, the input pulse of magnitude 5 is fed into the system from -51 to 0 ms (yellow; one fifth of the input magnitude is shown). The target signal is a single Gaussian pulse with a peak after a certain interval from the end of the input pulse (green). The shaded area indicates the training period.

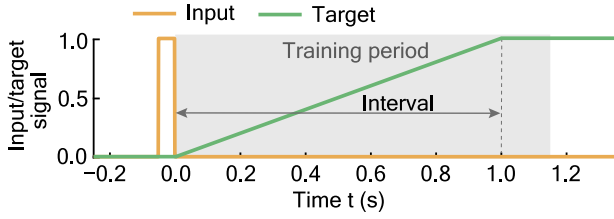


Fig. 3. Ramping motor timing task. The input pulse of magnitude 5 is fed into the system from -51 to 0 ms (yellow; one fifth of the input magnitude is shown). The target signal is a ramp function increasing from 0 to 1 during an interval and 1 after the interval (green). The shaded area indicates the training period.

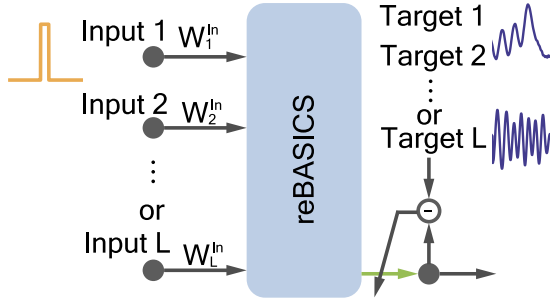


Fig. 4. Learning of multiple input-target pairs. reBASICS learns L input-target pairs one by one, where the readout weights are shared. The input pulse is common to the target signals, but their random input weights are different for each target signal.

$W_{k,ij}^{\text{Rec}}$ and $W_{k,i}^{\text{In}}$ are independently drawn from a standard normal distribution $\mathcal{N}(0, 1)$ as follows:

$$W_{k,ij}^{\text{Rec}} \sim \mathcal{N}\left(0, \frac{g}{\sqrt{pN}}\right), \quad (3)$$

$$W_{k,i}^{\text{In}} \sim \mathcal{N}(0, 1), \quad (4)$$

where g denotes a scaling coefficient of recurrent weights and p denotes the connection probability in units within a module. Although reBASICS sets $g \geq 1$ to autonomously generate activity, using a small N enables its modules to output stable orbits, such as limit cycles and tori, which are an intermediate state between convergence to a zero fixed point and chaotic behavior (Doyon et al., 1994; Sompolinsky et al., 1988). $W_{k,ij}^{\text{Rec}}$ (Eq. (3)) has a value

with probability p ; otherwise, it is set to zero. In addition, when $i = j$, $W_{k,ij}^{\text{Rec}} = 0$ to remove self-recurrent connections. There are no connections between the modules. The noise is drawn from a standard normal distribution with an amplitude I_0 :

$$I_{k,i}^{\text{noise}}(t) \sim \mathcal{N}(0, I_0). \quad (5)$$

The system output is obtained as a linear sum of the trajectories of the module outputs $r_k(t)$, which is the firing rate of a unit randomly chosen from each module.

$$y(t) = \sum_{k=1}^M W_k^{\text{Out}}(t) r_k(t), \quad (6)$$

where $W_k^{\text{Out}}(t)$ denotes the readout weights trained using recursive least squares (Haykin, 2009). The initial value of $W_k^{\text{Out}}(t)$ is zero. Note that the recurrent weights of the modules whose outputs converge to a fixed point are resampled, because such trajectories are useless for learning. That is, $W_{k,ij}^{\text{Rec}}$ are drawn from Eq. (3) again if the difference between the maximum and minimum values of the module outputs from 5 s to the learning end time is less than 0.01.

2.1.2. Learning rule

Recursive least squares modulate $W_k^{\text{Out}}(t)$ to minimize the error $e(t)$ between $y(t)$ and a target signal, $d(t)$, in an online manner. This learning rule is well known as an online learning method and has been used in existing reservoir computing methods including innate training (Inoue, Nakajima, & Kuniyoshi, 2020; Laje & Buonomano, 2013; Sussillo & Abbott, 2009). At time t , $W_k^{\text{Out}}(t)$ is updated as follows:

$$W_k^{\text{Out}}(t) = W_k^{\text{Out}}(t - \Delta t) - e(t) \sum_{l=1}^M P_{kl}(t) r_l(t), \quad (7)$$

$$e(t) = \sum_{k=1}^M W_k^{\text{Out}}(t - \Delta t) r_k(t) - d(t), \quad (8)$$

where $P_{kl}(t)$ ($\mathbf{P}(t)$, an $M \times M$ matrix) corresponds to the running estimate of the inverse sample covariance matrix of $r_k(t)$ (Haykin, 2009), and is updated according to

$$P_{kl}(t) = P_{kl}(t - \Delta t) - \frac{\sum_{m=1}^M \sum_{n=1}^M P_{km}(t - \Delta t) r_m(t) r_n(t) P_{nl}(t - \Delta t)}{1 + \sum_{m=1}^M \sum_{n=1}^M r_m(t) P_{mn}(t - \Delta t) r_n(t)}. \quad (9)$$

The initial value of $\mathbf{P}(t)$ is given as $\mathbf{P}(0) = (1/\alpha)\mathbf{I}$, where $\mathbf{I}$ denotes an identity matrix and α is a constant. Eq. (7) can be regarded as a type of delta rule which is the product of the error and module outputs. However, P_{lk} plays the role of multiple adaptive learning rate, rather than a fixed learning rate. See 5.6 in Haykin (2009) for the derivation of Eqs. (7)–(9).

2.2. Simulation procedure

The numerical solutions of Eq. (1) are obtained with a simulation step size of 1 ms. During the training period, the recursive least squares is applied once every two steps, and Δt in Eqs. (7)–(9) is set to 2 ms. The simulation begins at time -250 ms (-2050 ms in the simulation in Fig. 5) with a uniform random value in the range $[-1, 1]$ as the initial state of each unit. The input $u(t)$ is a single pulse with a magnitude of five between -51 and 0 ms, and zero in other periods (yellow line in Fig. 2). This pulse input is used to reset the phases of limit cycles of modules, that is, to eliminate the initial value dependence (see Fig. 6(a)).

Unless otherwise noted, the parameter values in Table 1 are used in all simulations. To establish a valid point of reference with

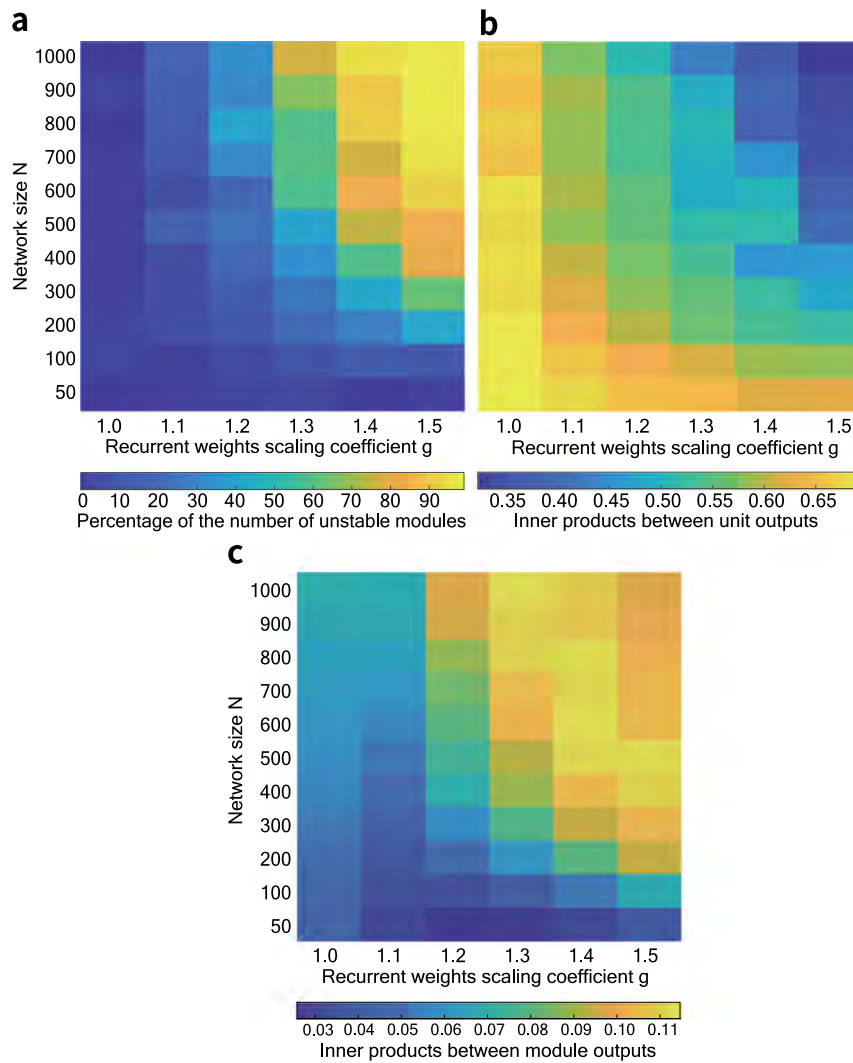


Fig. 5. Instability and orthogonality. (a) Colors indicate the instability of a single module; percentage of modules with orbital instability for 100 modules. The instability is defined by a local Lyapunov exponent of 0.05 or greater. (b) Colors indicate the orthogonality of a single module; averaged absolute inner products between *unit outputs* in a module, which were averaged over 100 modules. A low value (blue) indicates a high degree of orthogonality. (c) Colors indicate the orthogonality of reBASICS with 400 modules, with averaged absolute inner products between *module outputs* (averaged over 10 systems). In panels b and c, time series to compute the orthogonality were acquired from 1 to 10 s after the input pulse. Note that the color scale of panel c (absolute inner products between outputs in 400 modules) is much smaller than that of panel b (those between unit outputs in a single module). In all cases including panels a and b, modules whose outputs converged to a fixed point were excluded from the analysis.

a comparison method (innate training Laje & Buonomano, 2013), the parameter M , p , τ , I_0 , and α were matched in our study. N and g were set empirically based on the experimental result shown in Fig. 5 to satisfy both orbital stability and orthogonality.

For all tasks, to evaluate the learning performance, we used the squared Pearson correlation coefficient R^2 between the system output $y(t)$ and target $d(t)$ from 1 ms to the end of a task, which was averaged over 10 test trials and then over 10 different networks.

2.2.1. Setting for motor timing task

To learn the motor timing, the system receives an input pulse and outputs a single Gaussian pulse with a peak after a specific interval from the end of the input pulse (green line in Fig. 2). The magnitude and standard deviation of the Gaussian pulse are one

Table 1
Parameter values.

Parameter	Description	Value
M	Number of modules	800
N	Network size (the number of units) of a module	100
g	Scaling coefficient of recurrent weights	1.2
p	Connection probability	0.1
τ	Time constant	10 (ms)
I_0	Noise amplitude	0.001
α	Initial value for recursive least squares	1.0

and 30 ms, respectively. The interval is set from 1 to 120 s in the experiments. The training period is defined as the interval plus 150 ms from time 0, which is indicated by the shaded area in Fig. 2. The reBASICS is trained on 10 trials (a trial is a run to the

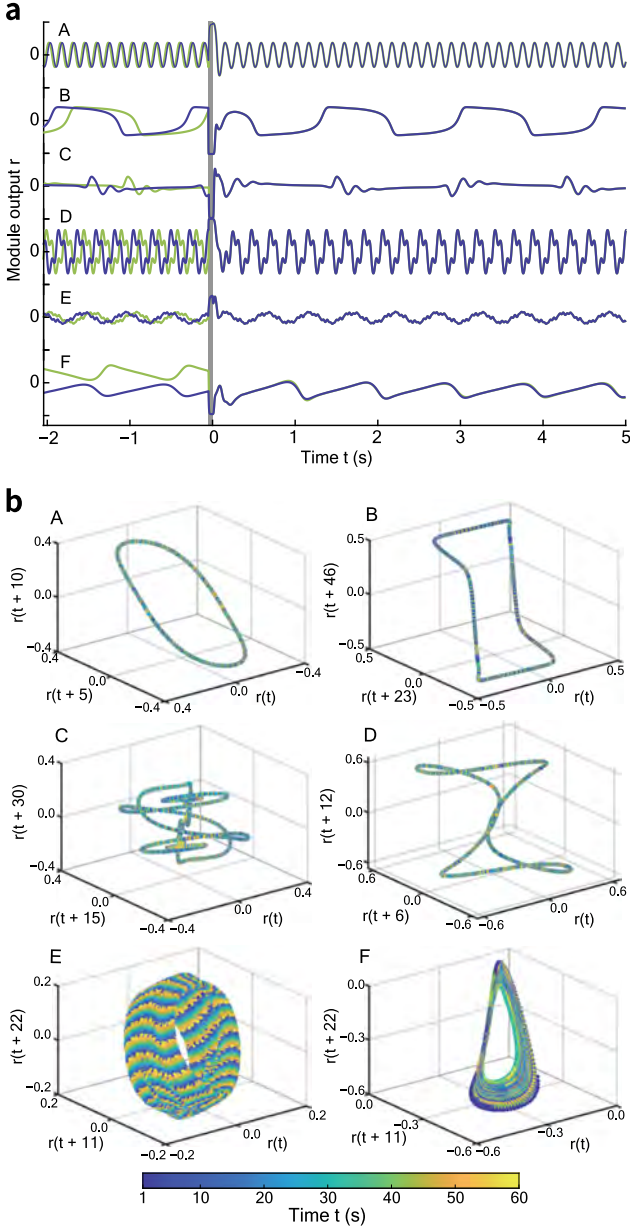


Fig. 6. reBASICS and its stable trajectories. (a) Time series of six sample module outputs in reBASICS with 800 modules from -2.05 to 5 s. The dark blue and green curves indicate two runs with different initial conditions. Input pulse was fed into the modules between -51 to 0 ms (vertical gray bar). Before the input pulse, in the most cases, the two time series had different phases, whereas after the input pulse, they traced very similar paths while showing the orbital stability. (b) The time series of panel a are represented in three-dimensional time-delay coordinates to reconstruct their phase spaces from 1 to 60 s. Color of each point indicates the time. The capital letters correspond to those of panel a. Trajectories A–D and E shows limit cycles and a torus, respectively, while trajectories F looks a strange attractor but with the orbital stability.

end of the training period), and then its learning performance is evaluated using 10 untrained test trials.

2.2.2. Setting for ramping motor timing task

We evaluate reBASICS using a different form of motor timing task. The input is the same as that of the motor timing task

described above. The target signal is a linearly increasing ramp function from 0 to 1 during an interval and takes the value 1 after that interval time (green line in Fig. 3). The interval is represented by the time until the output reaches the threshold value of 1 . Such a gradual increase in firing rate, as in the target time series, is widely observed in the cortex (Mita, Mushiake, Shima, Matsuzaka, & Tanji, 2009; Roux, Coulmance, & Riehle, 2003; Wang, Narain, Hosseini, & Jazayeri, 2018).

Some of reBASICS outputs may not reach the target threshold of 1 . Therefore, we obtain the decoded intervals by measuring the times until the reBASICS outputs reach the threshold of 0.9 and then dividing the times by the threshold value. The interval is set from 1 to 60 s in the experiments. The reBASICS is trained on 10 trials and is evaluated using 10 untrained test trials.

2.2.3. Setting for learning of chaotic time series

To learn chaotic time series, we use the Lorenz system (Lorenz, 1963) as the target signal. The use of the Lorenz system is effective to numerically prove the predominance of reBASICS for analyses of complex time series, because the Lorenz system typically yields complex and unpredictable time series by its low-dimensional deterministic equations:

$$\frac{dx(t)}{dt} = -px(t) + py(t), \quad (10)$$

$$\frac{dy(t)}{dt} = -x(t)z(t) + rx(t) - y(t), \quad (11)$$

$$\frac{dz(t)}{dt} = x(t)y(t) - bz(t), \quad (12)$$

where $p = 10$, $r = 28$, and $b = 8/3$, and we set $x(0) = 0.1$, $y(0) = 0$, and $z(0) = 0$. A numerical solution of Eqs. (10)–(12) is obtained using the fourth-order Runge–Kutta method with a step size of 0.001 . It is then downsampled to $1/5$ of its length and normalized to $[-1, 1]$ in magnitude to create a 20 s three-dimensional target signal, where one step is set to 1 ms. The input signal is the single pulse of 50 ms as in the previous experiments (yellow line in Figs. 2 and 3). The reBASICS has three readouts, namely $y_x(t)$, $y_y(t)$, and $y_z(t)$. Each readout has its own set of readout weights, corresponding to the x , y , and z variables of the target Lorenz time series. Each set of readout weights is trained with the corresponding time series. The numbers of trials for training and testing are the same as that for the learning of motor timing.

2.2.4. Setting for learning of multiple input-target pairs

We evaluate the capability of reBASICS to learn multiple target time series by using common readout weights. The readout weights are alternatively trained using L pairs of input and target signals. For the input, an input pulse of 50 ms is common; however, the input weights, which are drawn from a standard normal distribution, are different among the pairs (Fig. 4).

To learn multiple input-target pairs, L target signals, $d_l(t)$ ($l = 1, 2, \dots, L$), are created by superimposing eight sinusoidal signals with different amplitudes, frequencies, and phases:

$$d_l(t) = \sum_{i=1}^8 a_{l,i} \sin(2\pi((10b_{l,i} + 1)\omega t + c_{l,i})), \quad (13)$$

where $\omega = 10^{-3}$, and $a_{l,i}$, $b_{l,i}$, $c_{l,i}$ are uniform random numbers in $[0, 1]$. Then, the $d_l(t)$ is normalized so that the maximum (minimum) value is 1 (-1). The target signals $d_l(t)$ correspond to the input weights $W_{k,i,l}^{\text{in}} u(t)$, which have different random values for each l ; however, the input pulse and readout weights are common. reBASICS is trained using an input-target pair with $l = i$ ($i = 1, 2, \dots, L$) for one trial in sequence. This training set is repeated 10 times. Subsequently, reBASICS is evaluated during untraining trials.

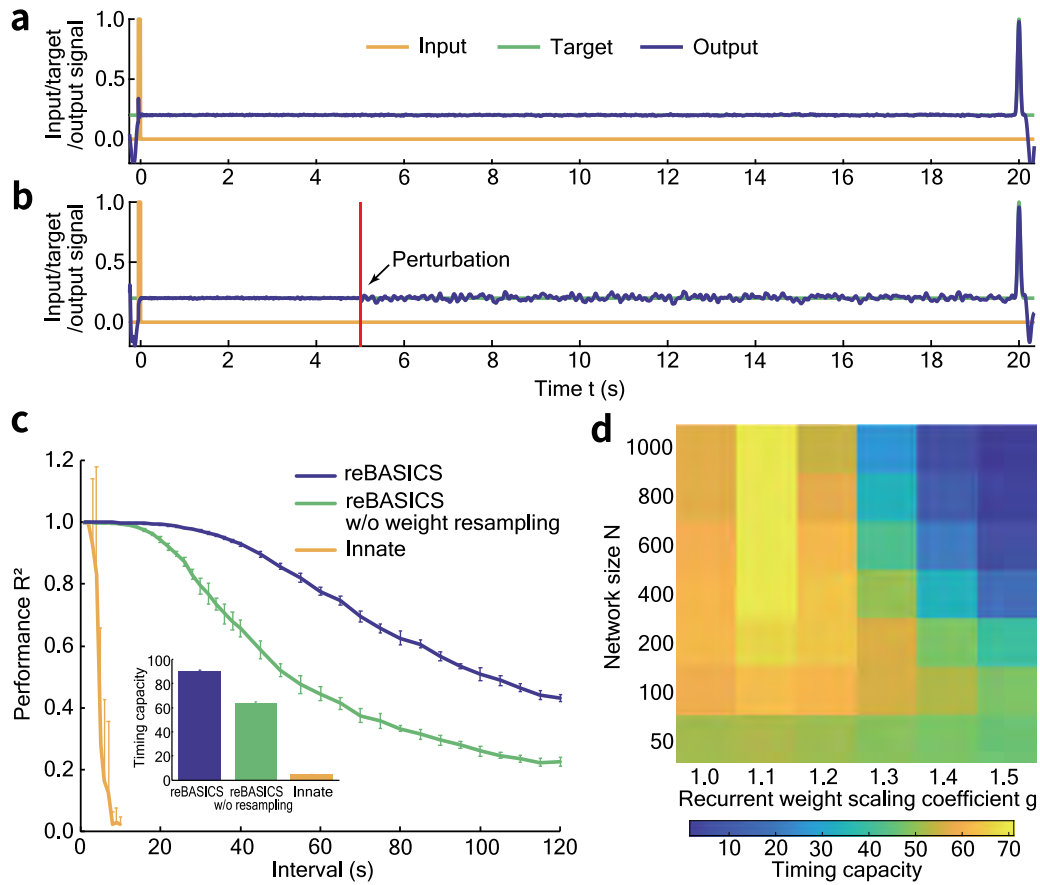


Fig. 7. Results for the learning of motor timing. (a) An example of the system output of reBASICS (800 modules and $g = 1.2$) after learning for the interval of 20 s. (b) The perturbation with the length of 10 ms and the amplitude of 0.1, which was given into all units through another input weights at 5 s after training (red line) that slightly disturbed the system output, but the replication of the Gaussian pulse can be performed. (c) Performance up to 120 s of intervals, which is defined as the square of correlation coefficient R^2 between the target signal and system output (green and dark blue lines in panel a, respectively). The performance was averaged over 10 networks (mean $\pm$ s.d.). The dark blue, green, and yellow curves show the results of reBASICS, reBASICS without weight resampling, and innate training, respectively. The inset shows the timing capacity defined by the area under the curve. (d) Colors indicate the timing capacity up to 120 s, which was averaged over 10 networks of reBASICS with 400 modules.

2.3. Evaluation of instability and orthogonality

Whether a module is unstable is determined by the local Lyapunov exponent, which is estimated in a similar manner in the Refs. [Inoue et al. \(2020\)](#) and [Laje and Buonomano \(2013\)](#). At time 0 ms, we add a small perturbation to all units in module k : $\mathbf{x}_k^{\text{pert}}(0) = \mathbf{x}_k(0) + \epsilon$, where $\mathbf{x}_k(t) = \{x_{k,1}, x_{k,2}, \dots, x_{k,N}\}$ and $\epsilon = 10^{-5}$. The logarithm of the distance between the unperturbed and perturbed trajectories, which is normalized by the initial distance at 0 ms, is then computed in terms t :

$$\text{dist}_k(t) = \log \left(\frac{\|\mathbf{x}_k^{\text{pert}}(t) - \mathbf{x}_k(t)\|}{\|\mathbf{x}_k^{\text{pert}}(0) - \mathbf{x}_k(0)\|} \right). \quad (14)$$

The log distance $\text{dist}_k(t)$ is averaged over 10 trials, starting from different initial unit states. We define the local Lyapunov exponent (LLE) as the slope of the function from 1 to 10 s and regard the modules with $\text{LLE} > 0.05$ as unstable.

The orthogonality of a single module is evaluated as averaged absolute inner products between unit outputs in a module. The

temporal vectors of the time series $r_{k,i}(t)$ from $t = 1000$ to 10,000 ms are normalized to unit vectors. The absolute inner products between all combinations of vectors within the same module are computed. The inner products are averaged over the all combinations and then over 100 modules. To evaluate the orthogonality between module outputs in reBASICS, the absolute inner products among $r_k(t)$ from $t = 1000$ to 10,000 ms are computed, and they are averaged over 10 different systems.

2.4. Innate training

The innate training approach ([Laje & Buonomano, 2013](#)) for the learning of motor timing is also evaluated to compare its performance with that of reBASICS. The unit dynamics are the same as those of reBASICS, that is, Eqs. (1)–(6), but it consists of a single RNN, that is, $M = 1$, and uses all unit outputs to synthesize the system output. First, this approach obtains the innate trajectory of each unit for one trial, which is a time series of the unit output in the absence of additive noise, and then repeatedly trains incoming connection weights to each unit with

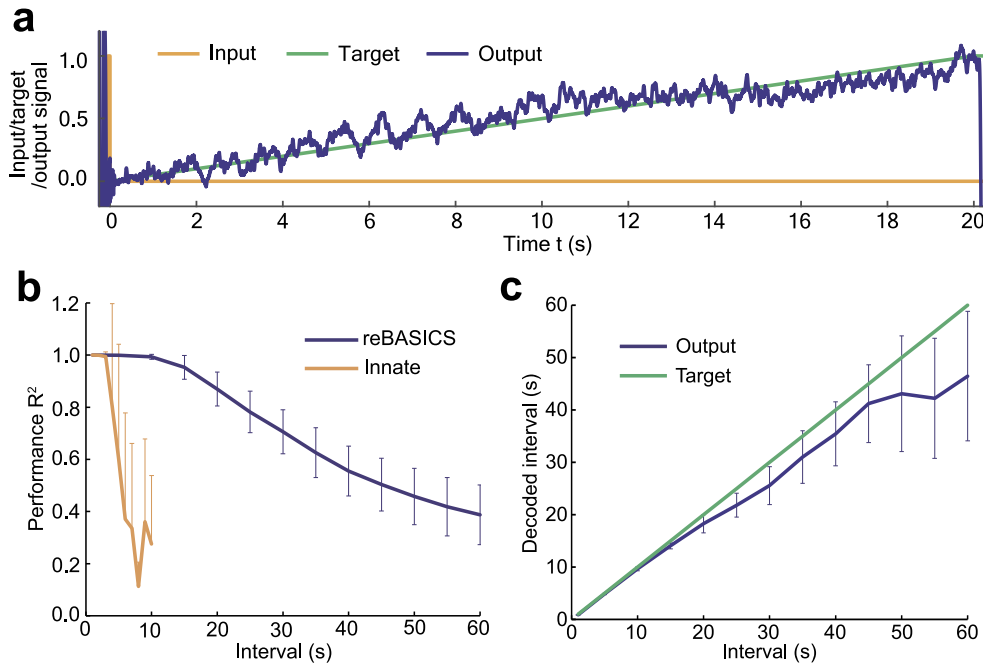


Fig. 8. Results for the learning of ramping motor timing. (a) An example of the system output of reBASICS after learning for the interval of 20 s. (b) Performance up to 60 s of the intervals. The performance was averaged over 10 networks (mean $\pm$ s.d.). The dark blue and yellow curves show the results of reBASICS and innate training, respectively. (c) Decoded interval (dark blue) is defined by the time until the output reaches the threshold. The green line shows the target intervals. They were averaged over 10 networks (mean $\pm$ s.d.).

recursive least squares such that the unit produces its own innate trajectory for 20 trials even when noise is added and starting from different initial conditions. This training enabled the network to produce stable and complex trajectories. Then, the readout weights are trained with recursive least squares for 10 trials to minimize the error between the system output and target signal, as shown in Eqs. (7)–(8).

We train 60% of the total $N = 800$ units to generate their innate trajectories and use the same parameters as in reBASICS: connection probability $p = 0.1$, time constant $\tau = 10$ ms, noise amplitude $I_0 = 0.001$, initial value for the recursive least squares $\alpha = 1$, and scaling coefficient of recurrent weight $g = 1.5$, which follow those of the literature (Laje & Buonomano, 2013).

3. Result

3.1. Stability and orthogonality

First, we evaluated the stability and orthogonality of the time series of *unit outputs* in a single module to clarify the utility of using several small modules. In RNNs with a finite N , either nonzero stationary states, limit cycles, or tori appear under $g > 1$. Such stable trajectories bifurcate into chaos as g increases. The range of g where this intermediate behavior is observed expands as N decreases (Doyon et al., 1994; Sompolinsky et al., 1988). Fig. 5a reproduces this phenomenon, where the module even with a large g exhibits the orbital stability when N is small. However, a small N necessarily decreases the orthogonality between the unit outputs (Fig. 5b). This figure shows that the orthogonality increased moderately as N and/or g increased. These results indicate that a single RNN clearly has a trade-off between stability and orthogonality; increasing the orbital stability of unit outputs decreases the orthogonality between unit outputs, and vice versa.

3.2. Trajectories of module outputs in reBASICS

To overcome this trade-off, reBASICS consists of multiple small modules (see Fig. 1). The time series of the module outputs of reBASICS showed a great variety of amplitudes, frequencies, phases, and waveforms (Figs. 6a and Fig. A.12). An external input of length 50 ms was fed into all units at time 0 as a starting point to learn the target time series. This input pulse aligned the phases of the module outputs among the runs, starting with different initial unit states. Fig. 6b shows the trajectories of these module outputs after the input pulse in three-dimensional phase spaces (Fraser & Swinney, 1986; Kantz & Schreiber, 2004). Most module outputs were limit cycles, and a few were tori- or strange attractor-like stable trajectories (Fig. 6c A–D, E, and F, respectively).

Fig. 5c shows that the orthogonality between the *module outputs* is much lower than that between the *unit outputs* (see Fig. 5b). A decrease in the orthogonality as g and N became large indicates that the limit cycles and torus orbits were more orthogonal than chaotic orbits. In any case, it was shown that partitioning an RNN into independent modules significantly improved the orthogonality between the outputs, leading to a high representational capacity. While this result was in the absence of inputs after the pulse input, Fig. A.11 shows the orthogonality when random inputs were continuously given. Although the common input to the modules reduces orthogonality between them, the reduction was not very critical for weak coupling strengths of inputs. Thus, it would be expected orthogonality in reBASICS under certain conditions.

3.3. Learning of motor timing

We evaluated the performance of reBASICS in motor timing learning (see Fig. 2) (Buonomano & Laje, 2011; Inoue et al., 2020; Laje & Buonomano, 2013). Fig. 7a shows an example of the system

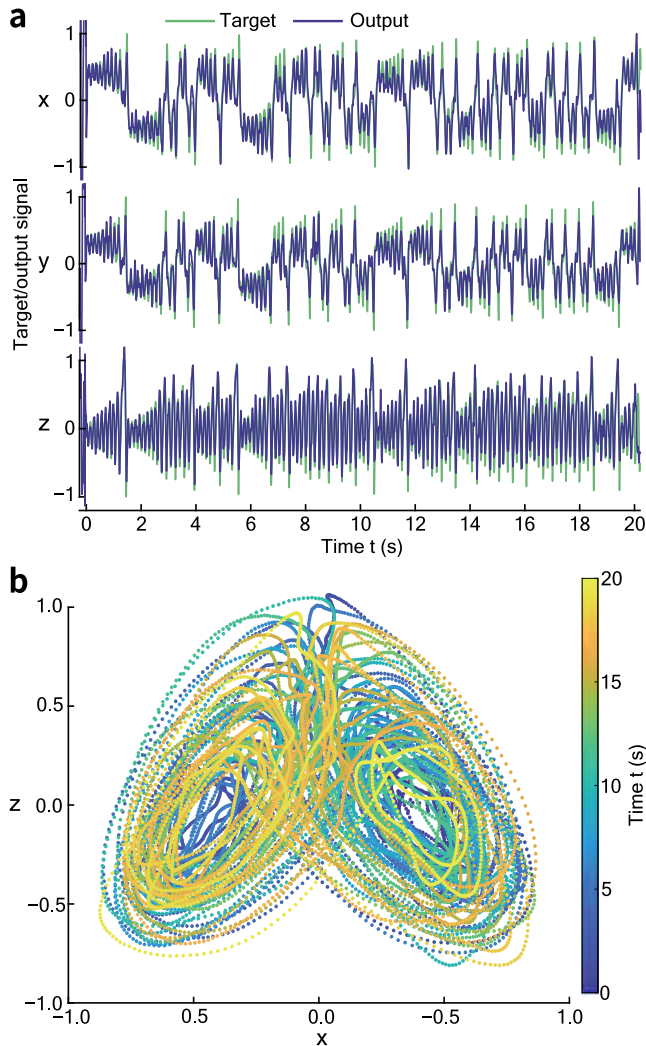


Fig. 9. A result for the learning of chaotic trajectories in the Lorenz system. (a) The time series show an example of the system output (dark blue) and target signal (green) after learning of the Lorenz system for 20 s. In this simulation, reBASICS had 800 modules, and each of which had 100 units and $g = 1.2$. (b) The trajectories of the system outputs that were trained to reproduce $x(t)$ and $z(t)$ are plotted, where colors indicate time after training.

output in a test trial with an interval of 20 s. The system output replicated a Gaussian pulse at 20 s after the input pulse. Even when a perturbation of 10 ms was applied to all modules at 5 s after training, the system output did not collapse significantly, because the module outputs were stable (Fig. 7b). The system output is slightly disturbed after the perturbation owing to the slight phase shifts of some of the module outputs.

We parametrically varied the interval and compared the performance of three methods: reBASICS (800 modules, $N = 100$, and $g = 1.2$), reBASICS without resampling of the recurrent weights, and the innate training approach (800 units) (Fig. 7c). Surprisingly, reBASICS was found to be capable of learning even for very long intervals of one minute or more. The defined timing capacity is the area under the curve up to 120 s; the timing capacity of reBASICS was 90.0 ± 1.3 , which was more than 20 times larger than that of the innate training (4.4 ± 0.64) (mean $\pm$ s.d. across 10 networks). Innate training uses all unit outputs for the readout; therefore, the learning cost for the readout is

the same as that of reBASICS with 800 modules. However, the entire learning cost of the innate training is much higher than that of reBASICS because of learning inside the recurrent neural network. Increasing the network size of the innate training did not significantly improve the performance (Fig. A.13). In contrast, the performance was better when reBASICS had more modules (more number of the orthogonal basis) (Fig. A.14). In reBASICS, the recurrent weights for an average of $45 \pm 2.0\%$ (mean $\pm$ s.d. across 10 networks) of the modules that converged at a fixed point were resampled. The performance of reBASICS without resampling decreased because the fixed-point attractors were unnecessary for learning.

To quantify the dependence of the timing capacity on g and N , we varied these values parametrically (Fig. 7d). The timing capacity decreased when g and N became large (in top-right blue area) because the network activity became chaotic (see Fig. 5a). For each value of g , the performance decreased when N was extremely large. The best performance was obtained for $g = 1.1$ and $N = 600$, and the overall performance was better when $g = 1.1$. However, a smaller g resulted in more modules whose outputs converged to a fixed point (Fig. A.15); therefore, more computational cost was required to resample the recurrent weights. The performance was insensitive to the connection probability p (Fig. A.16), but degraded as noise amplitude I_0 increased (Fig. A.17).

3.4. Learning of ramping motor timing

Fig. 8 shows an example of the system output in a test trial with an interval of 20 s. The system output was able to approximate the target ramping time series, though including jitter. As shown in Fig. 8b, we parametrically varied the interval up to 60 s and compared the performance of reBASICS (800 modules) with the innate training approach (800 units). As in the Gaussian motor timing task, reBASICS was able to learn over very long intervals and outperformed the innate training approach. Fig. 8c shows the intervals decoded from reBASICS outputs, indicating that reBASICS can represent timing intervals over long periods.

3.5. Learning of chaotic time series

Fig. 9 shows an example of the learning results for the Lorenz system (Lorenz, 1963), which is a three-dimensional chaotic time series. The performance values for x , y , and z in the Lorenz system were $R^2 = 0.96 \pm 0.019$, 0.94 ± 0.015 , and 0.96 ± 0.0059 (mean $\pm$ s.d. across 10 networks), respectively. This result indicates that reBASICS was capable of learning for non-periodic complex time series, such as the Lorenz system.

3.6. Learning of multiple input-target pairs

We evaluated the performance of reBASICS to learn multiple input-target pairs (see Fig. 4). To accomplish this task, the module activity must vary in response to the inputs, which have different random input weights, and emit different module outputs. Fig. 10a shows an example of the testing system outputs after training two pairs for 5 s. reBASICS successfully produced output signals to replicate the two target time series according to the inputs. This training enabled reBASICS to output the corresponding time series for the input signal at an arbitrary time. Fig. 10b shows that the module outputs exhibited waveforms with different phases, depending on the inputs. The reBASICS could learn with more than three pairs, though the learning performance gradually decreased almost linearly with L (dark blue curve in Fig. 10c). Compared to the performance when the same input weights were shared among different targets (green curve), the phasic responsiveness of reBASICS to the inputs could greatly improve its computational capability.

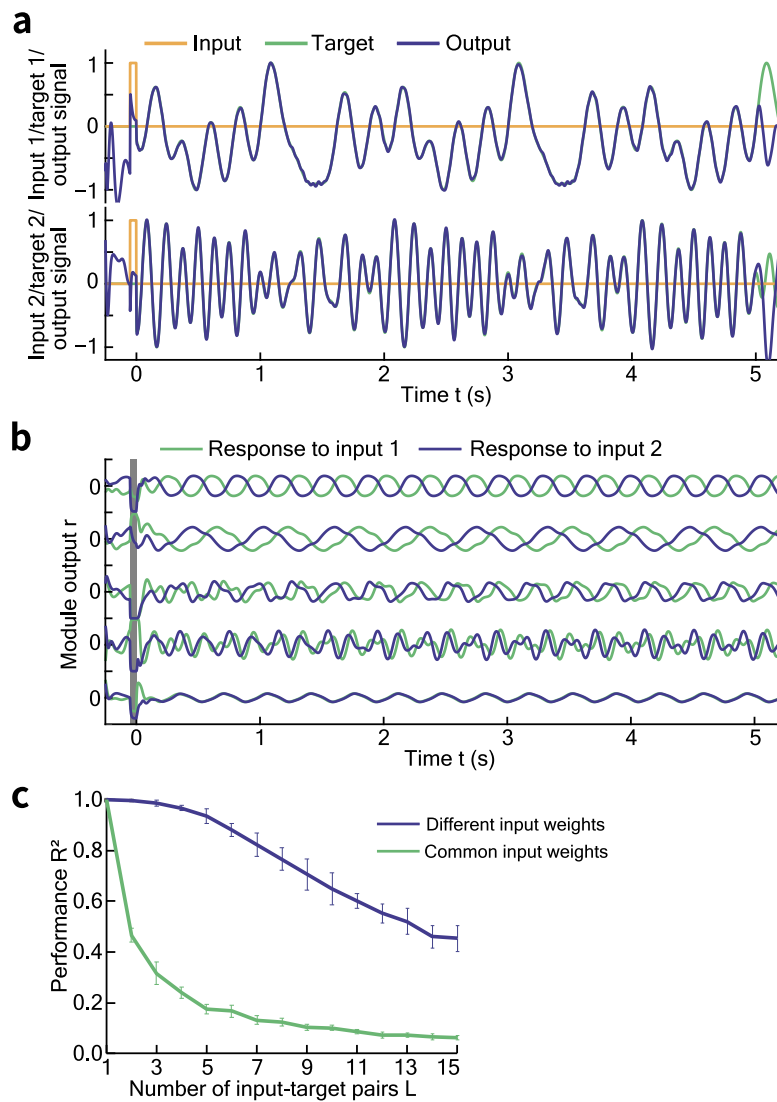


Fig. 10. A result for learning of multiple input-target pairs. (a) The two graphs show the system outputs (dark blue), different target signals (green), and input pulse (yellow) after learning of the two input-target pairs alternately. In this simulation, reBASICS had 800 modules, each of which has 100 units and $g = 1.2$. (b) The phases of the module outputs varied depending on the inputs. The green and navy lines indicate time series of five module outputs, which were randomly chosen in response to different two inputs. The shaded area indicates the period of the input pulse. Some module outputs did not change their phases, as in the time series at the bottom. (c) The learning performance versus the number of pairs L . The dark blue and green curves indicate the performance of reBASICS with different input weights for the pairs and the performance of reBASICS using the same input weights for them, respectively. The performance was averaged over 10 networks (mean $\pm$ s.d.).

4. Discussion

In this study, we first elucidated the trade-off between the stability and orthogonality of self-sustained activity in an RNN with $g \geq 1$. The high orbital stability due to small g and network size N resulted in low orthogonality, which reduced the representational capability of the system, whereas the high orthogonality due to large g and N resulted in orbital instability, leading to learning failure. To solve the trade-off, reBASICS reduces N , which makes the networks stable over a wide range of g , producing trajectories expressing limit cycles or tori (Doyon et al., 1994; Sompolinsky et al., 1988), and uses such networks as independent modules to enhance the orthogonality among the module outputs. We found that the trajectories of module outputs created a dynamic orthogonal basis, which enabled the system to learn time series on a scale of tens of seconds. Considering

that any functions can be represented by an orthonormal basis, such as Fourier series, we assert that the genesis of an orthogonal basis inside the reservoir assures the high computing power of reBASICS. Because of this property, an increase in N as well as g caused a decrease in the learning performance due to orbital instability. This phenomenon is interesting because RNNs with larger N usually performs better due to the larger degrees of freedom of the networks (Verstraeten, Schrauwen, d'Haene, & Stroobandt, 2007).

We have shown that multiple temporal tasks can be accomplished using reBASICS that can generate arbitrary time series. In both the Gaussian and ramping timing tasks, reBASICS was able to learn intervals of tens of seconds or more, outperforming the existing innate training approach. Such ramping activity is observed in different parts of the brain, including the motor and prefrontal areas (Mita et al., 2009; Roux et al., 2003; Wang et al., 2018).

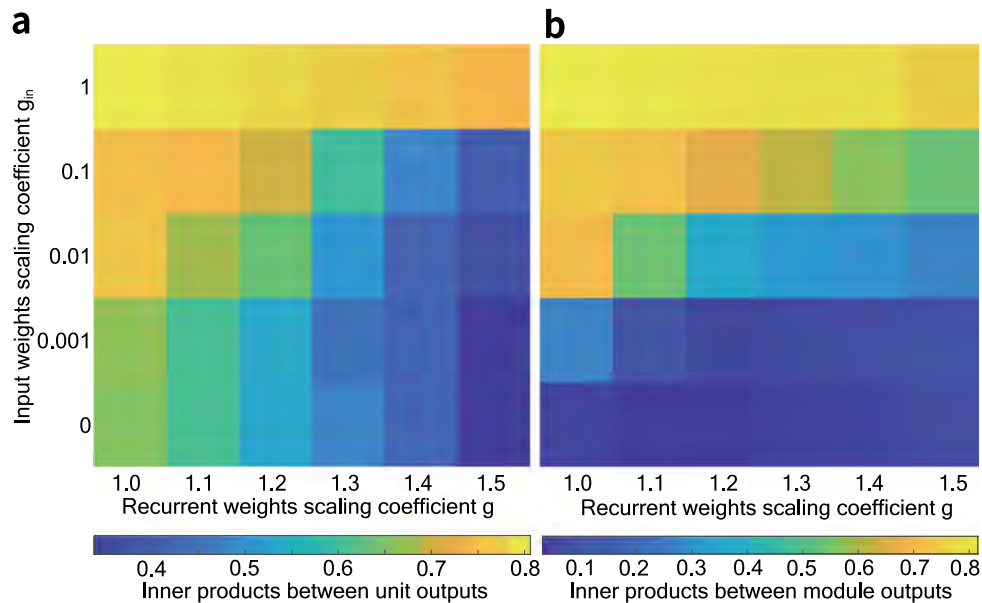


Fig. A.11. Orthogonality in the presence of inputs. Input signals were one-dimensional random series sampled from a uniform distribution over $[-1, 1]$, which were fed to all units through input weights at each time. The input weights were drawn from a normal distribution $\mathcal{N}(0, g_{in})$, where g_{in} indicate an input weights scaling coefficient. (a) Colors indicate the orthogonality of a single module with 800 units; averaged absolute inner products between unit outputs in a module, which were averaged over 100 modules. (b) Colors indicate the orthogonality of reBASICS with 800 modules, with averaged absolute inner products between module outputs (averaged over 10 systems). Time series to compute the orthogonality were acquired from 1 to 10 s after the input pulse.

While existing computational models of the ramping activity relied on backpropagation of recurrent neural networks (Beiran, Meirhaeghe, Sohn, Jazayeri, & Ostojic, 2023; Remington, Narain, Hosseini, & Jazayeri, 2018), reBASICS was able to accomplish the task using only least-squares learning in the readout. Furthermore, we found that (phases of) the module outputs varied in response to the inputs, allowing a single readout to learn multiple spatiotemporal patterns.

Even though the network size in each module is small, the number of its weight parameters is large, and the potential for diversity in the dynamics created by random weights is enormous. The most significant contribution of this study is the discovery that these dynamics are orthogonal to each other. Random time series are orthogonal but meaningless; specifically, an orthogonal basis must be generated from simple recurrent neural networks which are not random. The Fourier transformation is an expansion with orthogonal trigonometric functions for differentiable time series, and the wavelet transformation is an expansion for time series containing indifferentiable parts. In this study, such expansion is realized by limit cycles with complex forms. However, it is challenging to show mathematically how the orthogonality was achieved, because the complex limit cycles have very strong nonlinearity. Moreover, reBASICS relies on a small network size, i.e., a finite size effect, which makes mathematical analysis such as mean-field approximation difficult.

reBASICS was able to learn motor timing and chaotic time series, such as Lorenz attractors, with high accuracy. However, it does not infer the dynamical system itself, particularly, its attractors generating the target series; thus, it did not work to estimate the Lorenz attractor that should be yielded by interacting three variables. Instead, it synthesizes the target time series using orthogonal bases, as Fourier transformation for a given time series works. The readout from independent modules cannot represent hidden variable interactions, limiting the generalization capability of reBASICS. One possible way to rescue this situation

might be to add inter-module connections among modules. Thus, we evaluated reBASICS with inter-module connections in a motor timing task and learning task for the Lorenz time series (Fig. A.18). Unfortunately, the introduction of inter-module connections did not improve the performance but resulted in performance degradation due to the appearance of chaoticity. However, the idea of inter-module connections could be significant not only from a dynamical systems perspective but also from a biological perspective, and it would be valuable to study. Some methods based on output feedback have been proposed to embed the dynamical system in a reservoir from an observed time series (Hart, Hook, & Dawes, 2020; Kim, Lu, Nozari, Pappas, & Bassett, 2021). We would like to combine such methods with reBASICS to solve further difficult problems in the near future.

There is another limitation that stems from the fact that reBASICS does not represent a stable dynamical system itself. Although the biological neural system is exposed to noise, neural noise significantly reduced the performance for the motor timing task in reBASICS (see Fig. A.17). We have two ideas to interpret this noise sensitivity. One is that the animals' ability to create accurate timing behavior over long periods might be limited by neural noise. Fig. A.17 shows that the model could accomplished the task for 10 s even under noise. This is intuitively consistent with human performance. Another possibility is that the nervous system has a noise reduction mechanism. For example, innate training approach trains the reservoir network to reduce the effect of noise (Laje & Buonomano, 2013). Incorporating such a mechanism into our model would make it robust to noise.

The reBASICS can be interpreted as a model of various brain areas that learn motor timing and generate motor patterns because it is simple and principled. Many brain regions are involved in timing learning (Mauk & Buonomano, 2004; Paton & Buonomano, 2018), such as the basal ganglia (Jin, Fujii, & Graybiel, 2009; Schubotz, Friederici, & von Cramon, 2000) and cerebellum (Kalmbach et al., 2010; Perrett, Ruiz, & Mauk, 1993), which are primarily responsible for general motor learning (Doya, 2000).

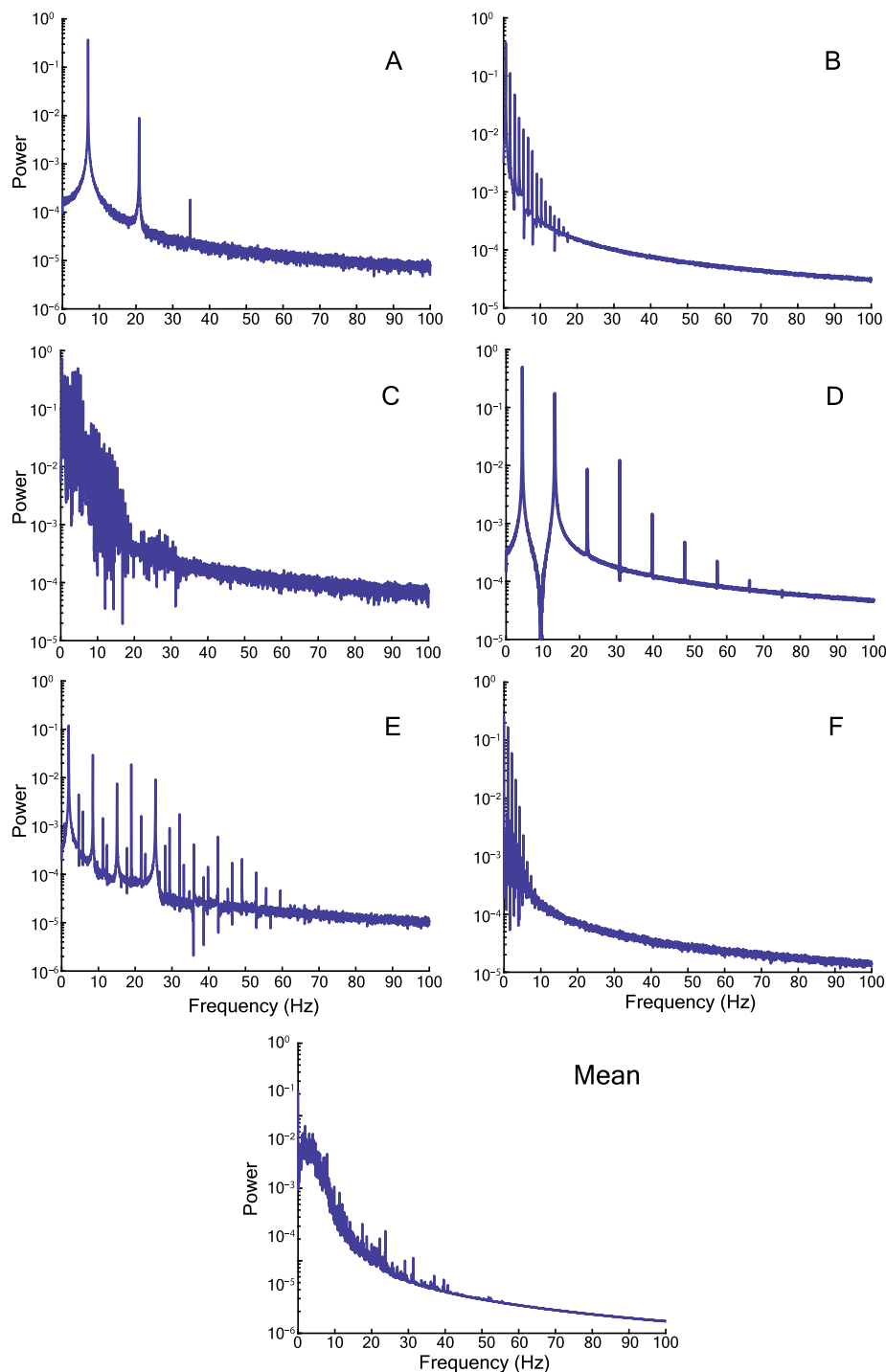


Fig. A.12. One-sided power spectra of the time series of the six example module outputs in reBASICS, which were computed using fast Fourier transformation. The capital letters correspond to those in Fig. 6. The bottom panel shows the average of the power spectra over 800 module outputs.

The reBASICS generates stable module activity in response to the input pulse (onset signal), whereas it has been widely observed that the trial-to-trial variability of firing rates decreases after stimulus onset in biological neurons (Churchland, Byron, Cunningham, et al., 2010). Such orbital stability is implicitly assumed in existing computational models of timing learning, based on neural oscillations. The beat frequency model uses a set of oscillators with different frequencies to learn the timing by detecting their firing coincidence (Miall, 1989). Matell and Meck (2004)

extended this model as a cortico-striatal model for timing learning. In this model, the oscillators at a range of different frequencies represent the cortical neural populations that project into the striatum, where dopamine signals modulate the readout of the oscillator outputs to realize reward-based learning of timing. Following the model, the modules in reBASICS can be interpreted as neural populations in various cortical regions, and training of the readout as a function of dopaminergic modulation in the striatum. RNNs and readout training are regarded as cortico-striatal

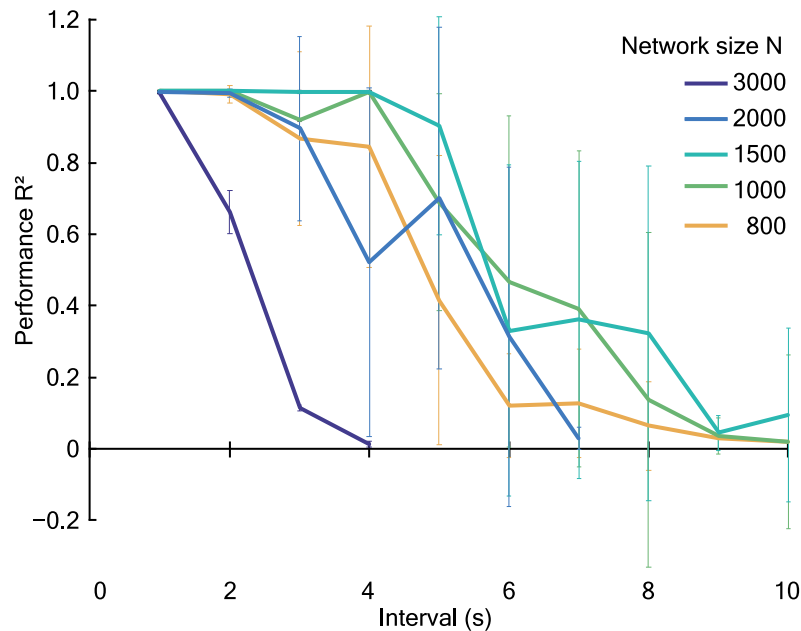


Fig. A.13. Performance of the innate training approach with different network sizes N . In each network with $g = 1.5$, 90% of the units were trained to reproduce their own innate trajectories and all unit outputs were used to train the readout. Mean $\pm$ s.d. over 10 networks.

systems in other computational modeling studies on time-series learning (Dominey, 1995; Hinaut & Dominey, 2013). Limit cycle attractors has often been used to model cortical large-scale oscillations (Breakspear, 2017) including the alpha (Wilson & Cowan, 1972) and beta (Moran et al., 2007) rhythms. reBASICS could predict that the modular or cluster organization of cortical connectivity (Hilgetag, Burns, O'Neill, Scannell, & Young, 2000; Sporns & Betzel, 2016) might enhance the orthogonality between cortical activities, resulting in support for motor pattern learning in the striatum.

The cerebellum (and also hippocampus) is well known to be involved in the motor timing of eyeblink conditioning (Kalmbach et al., 2010; Perrett et al., 1993). In computational models of the cerebellum (Buonomano & Mauk, 1994; Tokuda, Fujiwara, Sudo, & Katori, 2021; Yamazaki & Tanaka, 2007), granule cells receive cortical inputs through mossy fibers, and granule cells and Golgi cells form recurrent circuits. The granule cell outputs are sent to Purkinje cells to integrate them to produce the system output. The plasticity of the Purkinje cell output represents the readout training. In this framework, the recurrent networks of granule cells and Golgi cells sustain input-induced activity for some time, and Purkinje cells output the desired patterns at a specific time. Because various brain regions project into the cerebellum and the inhibitory recurrent connections from Golgi cells to granule cells might be local, the recurrent networks are not completely independent modules, but might constitute a neural population cluster (Apps et al., 2018). Such zonal patterning of connections might enhance the orthogonality between granule cell outputs.

Most timing learning models, including the above-mentioned models, do not explicitly consider orbital stability and orthogonality, neither they address functions other than timing learning, such as motor pattern generation in motor learning, which mainly involve in the striatum and cerebellum. The reBASICS is a more general and principled neural network model that can learn not only motor timing but also motor patterns, and deal with the stability and orthogonality of the trajectories behind these functions.

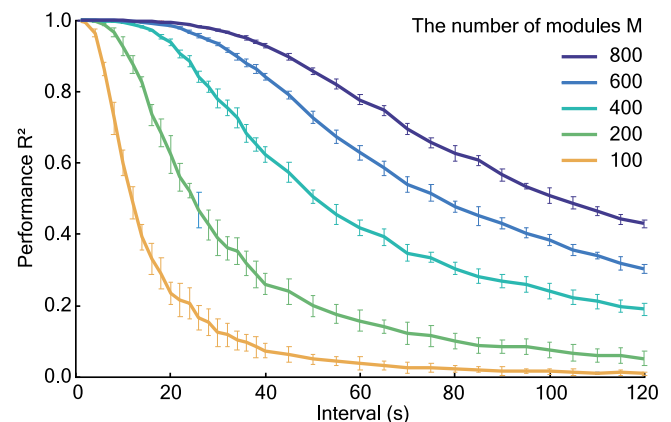


Fig. A.14. Performance of the learning of motor timing for different numbers of modules. More modules improve the performance. reBASICS with 800 modules in the main text did not reach saturated performance, and even better performance could be achieved with more modules. Mean $\pm$ s.d. over 10 networks.

5. Conclusion

We proposed reBASICS, which consists of small-sized RNNs as modules that generate stable and orthogonal dynamics in response to a single input pulse, creating a dynamical orthogonal basis that acquires a large computational capacity. reBASICS performs online learning of linear combinations of the module outputs to replicate very long motor timings, and its performance is more than 20 times better than that of the innate training. Furthermore, we showed that reBASICS was able to learn non-periodic chaotic time series, including the Lorenz attractors, with high accuracy. In particular, the finding that the output trajectories from independent modules are orthogonal to each other is significant and deepens our understanding of motor learning, especially in the basal ganglia and cerebellum. Further mathematical analyses on how they are orthogonal and their applications are expected in the future.

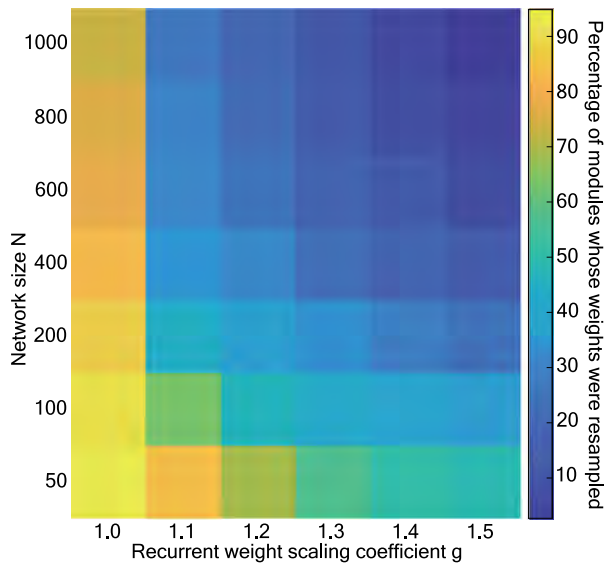


Fig. A.15. Percentage of modules whose recurrent weights were resampled because their outputs converged to a fixed point, which was averaged over 10 networks. The more stable modules with small g and N are, the more modules converge to a fixed point and the more times they are resampled.

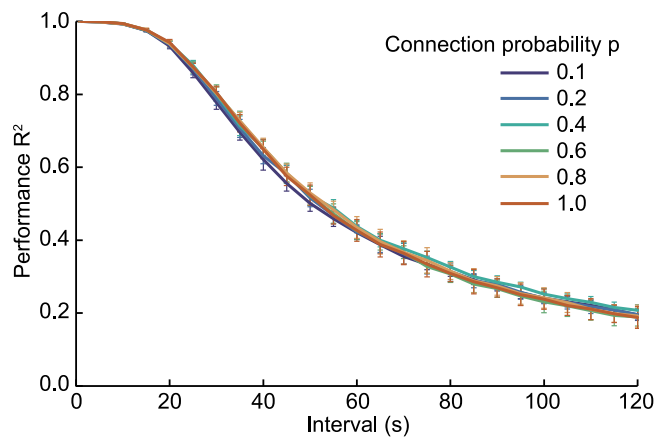


Fig. A.16. Performance of the learning of motor timing for different connection probability p . Mean $\pm$ s.d. over 10 networks of reBASICS with 400 modules.

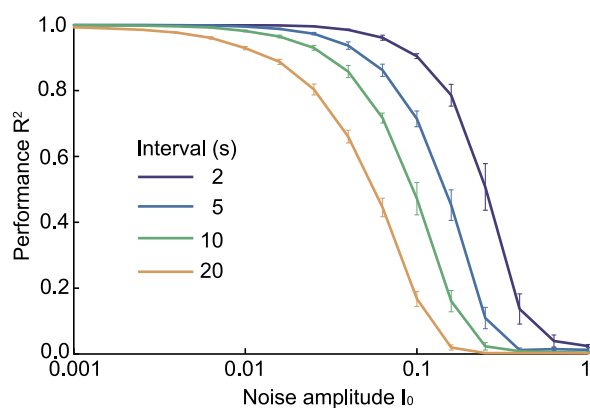


Fig. A.17. Performance of the learning of motor timing for different noise amplitude I_0 . The performance was for 2, 5, 10, and 20 s of intervals. Mean $\pm$ s.d. over 10 networks.

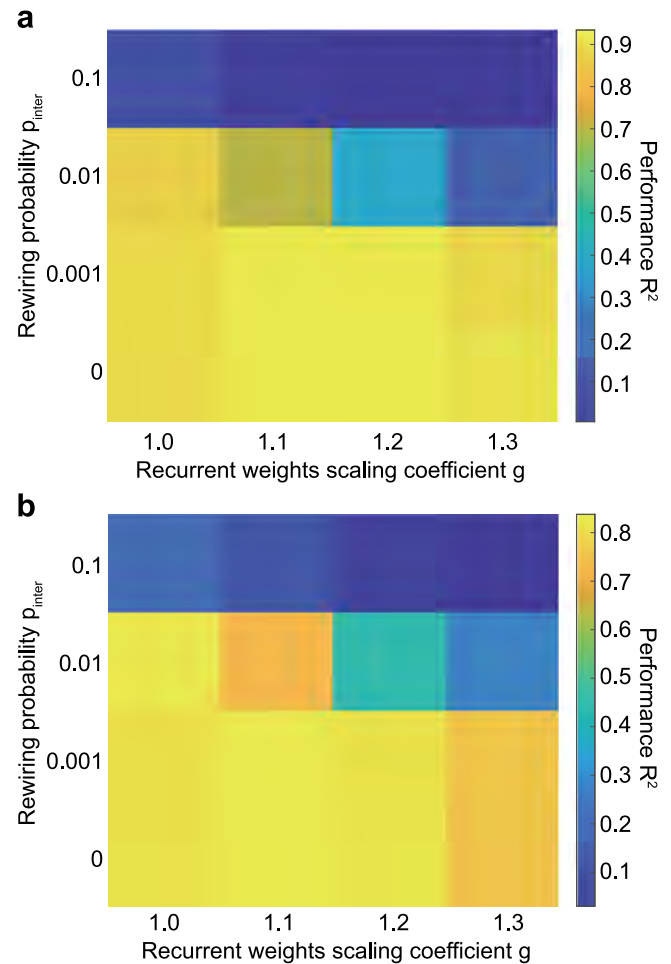


Fig. A.18. Performance of reBASICS with inter-module connections. First, M modules consisting of only intra-module connections were constructed. Then, each connection was rewired to another randomly selected node in another randomly selected module with a probability p_{inter} . Colors indicate the performance in (a) the motor timing task for 20 s of interval and (b) the learning task of the Lorenz time series (x) for 20 s. The performance was averaged over 10 networks of reBASICS with 400 modules.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yuji Kawai reports financial support was provided by Japan Science and Technology Agency. Minoru Asada reports financial support was provided by New Energy and Industrial Technology Development Organization.

Data availability

The Matlab code for reBASICS is available at on-line supplementary material

Acknowledgments

This work was supported by Japan Science and Technology Agency, CREST, Japan (JPMJCR17A4) and the project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

Appendix A. Supplementary figures

See Figs. A.11–A.18.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.neunet.2023.04.006>.

References

- Apps, R., Hawkes, R., Aoki, S., Bengtsson, F., Brown, A. M., Chen, G., et al. (2018). Cerebellar modules and their role as operational cerebellar processing units. *Cerebellum*, 17(5), 654–682.
- Beiran, M., Meirhaeghe, N., Sohn, H., Jazayeri, M., & Ostojic, S. (2023). Parametric control of flexible timing through low-dimensional neural manifolds. *Neuron*.
- Boccaletti, S., Grebogi, C., Lai, Y.-C., Mancini, H., & Maza, D. (2000). The control of chaos: theory and applications. *Physics Reports*, 329(3), 103–197.
- Breakspear, M. (2017). Dynamic models of large-scale brain activity. *Nature Neuroscience*, 20(3), 340–352.
- Buhusi, C. V., & Meck, W. H. (2005). What makes us tick? Functional and neural mechanisms of interval timing. *Nature Reviews Neuroscience*, 6(10), 755–765.
- Buonomano, D. V., & Laje, R. (2011). Population clocks: motor timing with neural dynamics. *Trends in Cognitive Sciences*, 14(12), 71–85.
- Buonomano, D. V., & Mauk, M. D. (1994). Neural network model of the cerebellum: temporal discrimination and the timing of motor responses. *Neural Computation*, 6(1), 38–55.
- Church, R. M., Mech, W. H., & Gibbon, J. (1994). Application of scalar timing theory to individual trials. *Journal of Experimental Psychology. Animal Behavior Processes*, 20(2), 135–155.
- Churchland, M. M., Byron, M. Y., Cunningham, J. P., et al. (2010). Stimulus onset quenches neural variability: a widespread cortical phenomenon. *Nature Neuroscience*, 13(3), 369–378.
- Deng, Z., & Zhang, Y. (2007). Collective behavior of a small-world recurrent neural system with scale-free distribution. *IEEE Transactions on Neural Networks*, 18(5), 1364–1375.
- Dominey, P. F. (1995). Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biological Cybernetics*, 73(3), 265–274.
- Doya, K. (2000). Complementary roles of basal ganglia and cerebellum in learning and motor control. *Current Opinion in Neurobiology*, 10(6), 732–739.
- Doyon, B., Cessac, B., Quoy, M., & Samuelides, M. (1994). On bifurcations and chaos in random neural networks. *Acta Biotheoretica*, 42(2), 215–225.
- Fraser, A. M., & Swinney, H. L. (1986). Independent coordinates for strange attractors from mutual information. *Physical Review A*, 33(2), 1134.
- Galatolo, S., Monge, M., & Nisoli, I. (2020). Existence of noise induced order, a computer aided proof. *Nonlinearity*, 33(9), 4237.
- Hart, A., Hook, J., & Dawes, J. (2020). Embedding and approximation theorems for echo state networks. *Neural Networks*, 128, 234–247.
- Haykin, S. (2009). *Neural networks and learning machines* (3rd). (pp. 245–248). Upper Saddle River, NJ, USA: Pearson.
- Hilgetag, C.-C., Burns, G. A. P. C., O'Neill, M. A., Scannell, J. W., & Young, M. P. (2000). Anatomical connectivity defines the organization of clusters of cortical areas in the macaque and the cat. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 355(1393), 91–110.
- Hinaut, X., & Dominey, P. F. (2013). Real-time parallel processing of grammatical structure in the fronto-striatal system: A recurrent network simulation study using reservoir computing. *PLoS One*, 8(2), Article e52946.
- Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science Advances*, 6(46), eabb3989.
- Ivry, R. B., & Schlerf, J. E. (2008). Dedicated and intrinsic models of time perception. *Trends in Cognitive Sciences*, 12(7), 273–280.
- Jaeger, H. (2001). The “echo state” approach to analysing and training recurrent neural networks. In *GMD-148*. German National Research Center for Information Technology.
- Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667), 78–80.
- Jin, D. Z., Fujii, N., & Graybiel, A. M. (2009). Neural representation of time in cortico-basal ganglia circuits. *Proceedings of the National Academy of Sciences*, 106(45), 19156–19161.
- Kalmbach, B. E., Davis, T., Ohshima, T., Riusech, F., Nores, W. L., & Mauk, M. D. (2010). Cerebellar cortex contributions to the expression and timing of conditioned eyelid responses. *Journal of Neurophysiology*, 103(4), 2039–2049.
- Kantz, H., & Schreiber, T. (2004). *Nonlinear time series analysis*, Vol. 7. Cambridge University Press.
- Kawai, Y., Park, J., & Asada, M. (2019). A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks*, 112, 15–23.
- Kim, J. Z., Lu, Z., Nozari, E., Pappas, G. J., & Bassett, D. S. (2021). Teaching recurrent neural networks to infer global temporal structure from local examples. *Nature Machine Intelligence*, 3(4), 316–323.
- Laje, R., & Buonomano, D. V. (2013). Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nature Neuroscience*, 16(7), 925–933.
- Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2), 130–141.
- Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3), 127–149.
- Matell, M. S., & Meck, W. H. (2004). Cortico-striatal circuits and interval timing: coincidence detection of oscillatory processes. *Cognitive Brain Research*, 21(2), 139–170.
- Matsumoto, K., & Tsuda, I. (1983). Noise-induced order. *Journal of Statistical Physics*, 31(1), 87–106.
- Mauk, M. D., & Buonomano, D. V. (2004). The neural basis of temporal processing. *Annual Reviews Neuroscience*, 27, 307–340.
- Miall, C. (1989). The storage of time intervals using oscillating neurons. *Neural Computation*, 1(3), 359–371.
- Mita, A., Mushiake, H., Shima, K., Matsuzaka, Y., & Tanji, J. (2009). Interval time coding by neurons in the presupplementary and supplementary motor areas. *Nature Neuroscience*, 12(4), 502–507.
- Moran, R. J., Kiebel, S. J., Stephan, K. E., Reilly, R. B., Daunizeau, J., & Friston, K. J. (2007). A neural mass model of spectral responses in electrophysiology. *NeuroImage*, 37(3), 706–720.
- Paton, J. J., & Buonomano, D. V. (2018). The neural basis of timing: distributed mechanisms for diverse functions. *Neuron*, 98(4), 687–705.
- Perrett, S. P., Ruiz, B. P., & Mauk, M. D. (1993). Cerebellar cortex lesions disrupt learning-dependent timing of conditioned eyelid responses. *Journal of Neuroscience*, 13(4), 1708–1718.
- Petter, E. A., Lusk, N. A., Hesslow, G., & Meck, W. H. (2016). Interactive roles of the cerebellum and striatum in sub-second and supra-second timing: Support for an initiation, continuation, adjustment, and termination (ICAT) model of temporal processing. *Neuroscience & Biobehavioral Reviews*, 71, 739–755.
- Remington, E. D., Narain, D., Hosseini, E. A., & Jazayeri, M. (2018). Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, 98(5), 1005–1019.
- Renart, A. (2013). Recurrent networks learn to tell time. *Nature Neuroscience*, 16(7), 772–774.
- Roux, S., Coulmance, M., & Riehle, A. (2003). Context-related representation of timing processes in monkey motor cortex. *European Journal of Neuroscience*, 18(4), 1011–1016.
- Schubotz, R. I., Friederici, A. D., & von Cramon, D. Y. (2000). Time perception and motor timing: a common cortical and subcortical basis revealed by fMRI. *NeuroImage*, 11(1), 1–12.
- Sompolinsky, H., Crisanti, A., & Sommers, H.-J. (1988). Chaos in random neural networks. *Physical Review Letters*, 61(3), 259–262.
- Sporns, O., & Betzel, R. F. (2016). Modular brain networks. *Annual Review of Psychology*, 67, 613–640.
- Sussillo, D., & Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4), 544–557.
- Tokuda, K., Fujiwara, N., Sudo, A., & Katori, Y. (2021). Chaos may enhance expressivity in cerebellar granular layer. *Neural Networks*, 136, 72–86.
- Verstraeten, D., Schrauwen, B., d'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, 20(3), 391–403.
- Wang, J., Narain, D., Hosseini, E. A., & Jazayeri, M. (2018). Flexible timing by temporal scaling of cortical responses. *Nature Neuroscience*, 21, 102–110.
- Wilson, H. R., & Cowan, J. D. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, 12(1), 1–24.
- Yamaguti, Y., & Tsuda, I. (2021). Functional differentiations in evolutionary reservoir computing networks. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, 31(1), Article 013137.
- Yamazaki, T., & Tanaka, S. (2007). The cerebellum as a liquid state machine. *Neural Networks*, 20(3), 290–297.