Dynamical Learning of Dynamics

Christian Klos,[1] Yaroslav Felipe Kalle Kossio,[1] Sven Goedeke,[1] Aditya Gilra,[1,2] and Raoul-Martin Memmesheimer[1]
[1]Neural Network Dynamics and Computation, Institute of Genetics, University of Bonn, 53115 Bonn, Germany
[2]Department of Computer Science, and Neuroscience Institute, University of Sheffield, Sheffield S1 4DP, United Kingdom

(Received 7 February 2019; revised 24 June 2020; accepted 21 July 2020; published 19 August 2020)

The ability of humans and animals to quickly adapt to novel tasks is difficult to reconcile with the standard paradigm of learning by slow synaptic weight modification. Here, we show that fixed-weight neural networks can learn to generate required dynamics by imitation. After appropriate weight pretraining, the networks quickly and dynamically adapt to learn new tasks and thereafter continue to achieve them without further teacher feedback. We explain this ability and illustrate it with a variety of target dynamics, ranging from oscillatory trajectories to driven and chaotic dynamical systems.

DOI: 10.1103/PhysRevLett.125.088103

Introduction.—Humans and animals can learn wide varieties of tasks. The predominant paradigm assumes that their neural networks achieve this by slow adaptation of connection weights between neurons [1,2]. Neurobiological experiments, however, also indicate fast learning with static weights [3]. Our study addresses how neural networks may quickly learn to generate required output dynamics without weight learning.

The goal of neural network learning is ultimately to appropriately change the activity of the output neurons of the network. In supervised learning, it should match a target and continue doing so during subsequent testing; in reinforcement learning, it should maximize a sparsely given reward. In our study, the networks adapt their weights during a pretraining phase [4–6] such that thereafter with static weights they achieve supervised learning of desired outputs, by adapting only their dynamics (dynamical learning). Adapting the static network's weights during pretraining is thus a kind of metalearning or learning to learn. There is a recent spurt of interest in learning to learn [5,6], focusing mainly on learning of reinforcement learning [7–9]. Studies on learning of supervised dynamical learning showed prediction of a time series at the current time step given the preceding step's target [10–19] and control of a system along a time-varying target [20–23]. The studies assume that a target is present during testing to avoid unlearning. This limits applicability and renders the dynamics necessarily nonautonomous; it is conceptually problematic for supervised settings and at odds with the common concept of teacher-free recall.

We therefore develop a scheme for fast supervised dynamical learning and subsequent teacher-free generation of long-term dynamics. We consider models for biological recurrent (reciprocally connected) neural networks, where leaky rate neurons interact in continuous time [1,2]. Such models are amenable to learning, computation, and phase space analysis [1,2,24–26]. After appropriate pretraining using the reservoir computing scheme (where only the weights to output neurons are trained [25,27,66,67]), all weights are fixed. The networks can nevertheless learn to generate new, desired dynamics. Furthermore, they continue to generate them in a self-contained manner during subsequent testing. We illustrate this with a variety of trajectories and dynamical systems and analyze the underlying mechanisms.

Network model.—We use recurrent neural networks, where each neuron (or neuronal subpopulation) $i, i = 1, \ldots, N$, with N between 500 and 3000 depending on the task [27], is characterized by an activation variable $x_i(t)$ and communicates with other neurons via its firing rate $r_i(t)$, a nonlinear function of $x_i(t)$ [1,2]. In isolation $x_i(t)$ decays to zero with a time constant τ_i. This combines the decay times of membrane potential and synaptic currents. The network has two outputs, which can be interpreted as linear neurons: signal $z_k(t), k = 1, \ldots, N_z$, and context $c_l(t), l = 1, \ldots, N_c$ (Fig. 1). After learning, $z(t)$ generates the desired dynamics while $c(t)$ indexes it. They are continually fed back to the network, allowing their autonomous generation [66]. The networks are temporarily also informed about their signal's difference from its target $\tilde{z}(t)$ by an error input $\varepsilon(t) = z(t) - \tilde{z}(t)$. Taken together, for constant weights the network dynamics are given by

$$\begin{aligned} \tau\dot{x}(t) &= -x(t) + Ar(t) + w_z z(t) \\ &\quad + w_c c(t) + w_\varepsilon \varepsilon(t) + w_u u(t), \\ z(t) &= o_z r(t), \qquad c(t) = o_c r(t), \end{aligned} \tag{1}$$

with recurrent weights A, the diagonal matrix of time constants τ, signal and context output weights o_z, o_c, feedback weights w_z, w_c, input weights w_ε, w_u, and a drive $u(t)$ absent for most tasks. We choose $r_i(t) = \tanh[x_i(t) + b_i]$ [25,66,68]; offsets b_i are drawn from a uniform distribution between -0.2 and 0.2 and break the $x \to -x$ symmetry

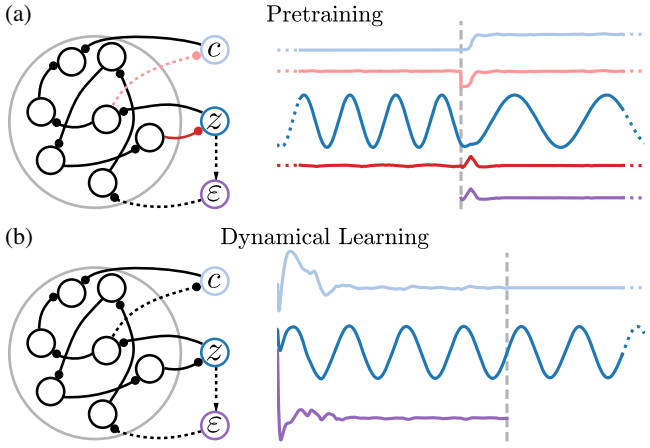

FIG. 1. Pretraining and learning. (a) During pretraining, the output weights (left, different reds) of the network are adapted using the output errors $\varepsilon(t) = z(t) - \tilde{z}(t)$ (right, red) and $c(t) - \tilde{c}(t)$ (light red), such that $z(t)$ (blue, different scale for clarity) and $c(t)$ (light blue) match their targets. Different members of the target family are weight learned in the training periods (dashed vertical). At their beginnings, $\varepsilon(t)$ is fed also as input (purple). (b) Dynamical learning. The output weights are now fixed. The network receives the signal error $\varepsilon(t)$ as input (purple). It adapts its dynamics to generate $z(t) \approx \tilde{z}(t)$ (blue). During testing, an error is no longer provided and $c(t)$ is fixed to its previous average (right, dashed vertical, left, dashed weights). $z(t)$ continues to approximate $\tilde{z}(t)$.

without input. Unless mentioned otherwise, we set $\tau_i = 1$ fixing the overall timescale. Recurrent weights A_{ij} are set to zero with probability $1 - p$ ($p = 0.1$ or $p = 0.2$ depending on the task). Nonzero weights are drawn from a Gaussian distribution with mean 0 and variance (g^2/pN), where $g = 1.5$ [25]. $w_{z,ij}$, $w_{c,ij}$, and $w_{\varepsilon,ij}$, $w_{u,ij}$ are drawn from a uniform distribution between $-\tilde{w}$ and $\tilde{w}$ ($\tilde{w} = 1$ or $\tilde{w} = 2$).

Pretraining.—The aim of our pretraining [Fig. 1(a)] is twofold. First, it should enable the resulting static networks to learn signals of a specific class given only the error input $\varepsilon(t)$. Second, after removing the error input the static networks should be able to continue to generate the desired dynamics. Therefore, the networks have to learn to minimize $\varepsilon(t)$ and, as explained in the Analysis section, to associate unique contexts with the different target dynamics.

To achieve this, we present different trajectories $\tilde{z}(t)$ of the target class to the networks, together with associated, straightforwardly chosen constant indices $\tilde{c}$. The output weights $o_{z,ij}$ and $o_{c,ij}$ learn online according to the FORCE rule [25,27] to minimize the output errors $\varepsilon(t)$ and $c(t) - \tilde{c}$. In short, they are modified using the supervised recursive least-squares algorithm with high learning rate. This provides a least-squares optimal regularized solution for the output weights given the past network states and targets [69]. Signals and indices are presented for a time t_{wlearn} (30 000 or 50 000) as a continuous, randomly repeating sequence of training periods of duration t_{stay} (between 200 and 1000). During each training period's first part, a network receives $\varepsilon(t)$ as input. Because of the various last states of the previous learning periods, it thus learns to approach $\tilde{z}(t)$ from a broad range of initial conditions given this input. In most of our tasks, after a time $t_{\text{fb}} = 100$, when $z(t)$ is close to $\tilde{z}(t)$, $\varepsilon(t)$ is switched off and $c(t)$ is fixed to its constant target, matching the testing paradigm. This often helps the network to learn generation of $z(t) \approx \tilde{z}(t)$ without error input.

Dynamical learning and testing.—The weights now remain static and the error input teaches the network new tasks of the pretrained target class [Fig. 1(b)], i.e., the networks dynamically learn to generate $z(t) \approx \tilde{z}(t)$ for previously unseen $\tilde{z}(t)$. The learning time t_{learn} (between 50 and 200) is short, a few characteristic timescales of the target dynamics [27]. $c(t)$ is moderately fluctuating.

Thereafter, the test phase begins, where no more teacher is present ($w_\varepsilon \to 0$). In weight-learning paradigms, during such phases the weights are fixed [25,66,67,70,71]. We likewise fix $c(t)$ to a temporally constant value, an average of previously assumed ones, $c(t) = \bar{c}$. This may be interpreted as indicating that the context is unchanged and the same signal is still desired. We find in our applications, that the network dynamics continue to generate a close-to-desired signal $z(t)$, establishing the successful dynamical learning of the task.

Applications.—We illustrate our approach by learning a variety of trajectories [tasks (i)–(iv)] and dynamical systems [tasks (v),(vi)]. First, we consider a family $\tilde{z}(t;k)$ of target trajectories, parametrized by k. The networks are pretrained on a few of them, where the context target $\tilde{c}$ is a linear function of k. Thereafter, the networks dynamically learn to generate a previously unseen trajectory as output and perpetuate it during testing. We start with the simple, instructive target family of oscillations with different periods [task (i)]: $\tilde{z}(t;T) = 5\sin[(2\pi/T)t]$. We use three different teacher trajectories for pretraining, with $T = 10$, 15, 20. After pretraining, our networks can precisely dynamically learn oscillations with unseen periods within and slightly beyond the pretrained ones [Figs. 2(a) and 2(b), see [27] for further detail and analysis of learning performance of all tasks]. Next, in (ii), we generalize (i) to higher order Fourier series. Specifically, we consider the target family of superpositions of two random Fourier series with weighting factor λ: $\tilde{z}(t;\lambda) = (1-\lambda)\tilde{z}_1(t;\lambda) + \lambda\tilde{z}_2(t;\lambda)$. Here, $\tilde{z}_l(t;\lambda)$, $l = 1, 2$, are Fourier series of order O and period $T(\lambda) = (1-\lambda)T_1 + \lambda T_2$. T_l and the Fourier coefficients are drawn randomly. We use seven different teacher trajectories for pretraining, with weighting factors distributed equidistantly between 0 and 1. After pretraining, we test the dynamical learning for thirteen weighting factors also distributed equidistantly between 0 and 1. To quantify the learning performance, we determine the fraction of these targets that can be successfully learned [root-mean-square error (RMSE) below given threshold (0.4) and

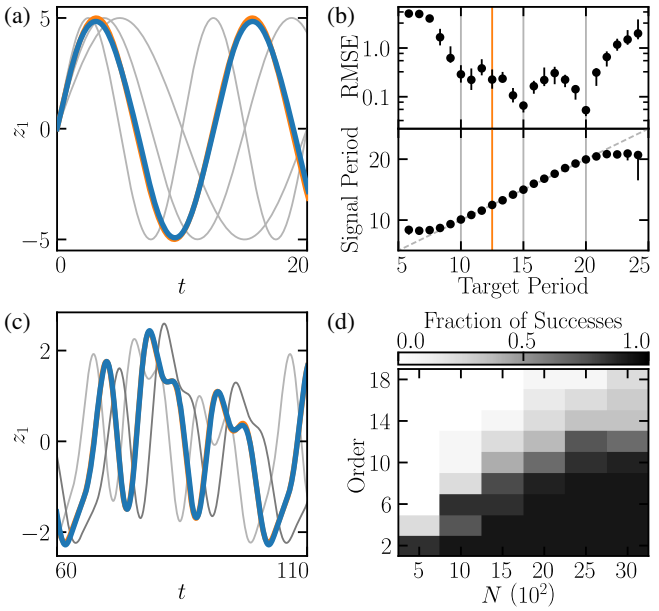

FIG. 2. Dynamical learning of periodic trajectories. (a),(b) Testing after dynamical learning of sinusoids. (a) Signal (blue) matches the example testing target (orange, mostly covered by signal) well. Pretraining targets (gray traces) are clearly distinct. (b) For many different targets the root-mean-square error (RMSE) between signal and target is low (top) and the signal's period tracks the target's period well (bottom). Gray and orange verticals indicate periods of pretrained targets and target in (a). Dots show median value and error bars show the interquartile range, using ten network instances. (c),(d) Testing after dynamical learning of Fourier series. (c) Like (a), for a random Fourier series with $O = 6$. Only the two closest pretrained dynamics are displayed for clarity. (d) Learning success for different network sizes and orders of the Fourier series. Color encodes median fraction of success, using 40 network instances and random Fourier series.

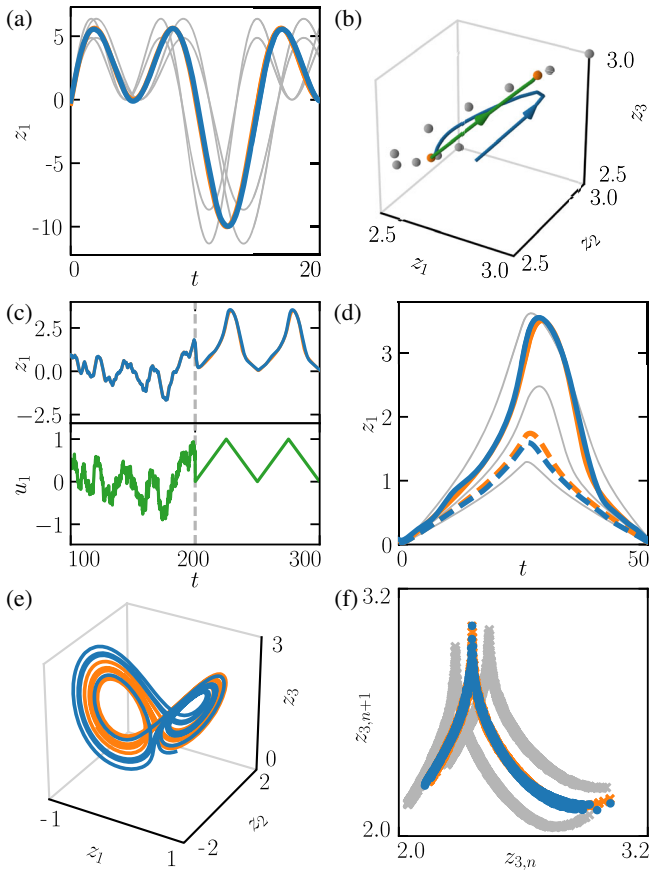

FIG. 3. Dynamical learning of different tasks. Testing phase after dynamical learning of an example (a) two-parametric superposition of sines, (b) fixed point, (c),(d) driven overdamped pendulum, and (e),(f) Lorenz system. (a)–(d) Signals (blue) match testing targets (orange, mostly covered by signal) well. Pretraining targets (gray traces or spheres) are clearly distinct. (a) displays only the four closest pretrained dynamics for clarity. (b) Signal transients (blue, green) of subsequent dynamical learning of two targets (orange spheres). (c) Signal, target, and drive (green) during dynamical learning and subsequent testing (dashed vertical). (d) Dynamically learned approximations of two different pendulums (continuous, dashed), driven by the same triangular input. (e) Limit sets of signal (blue) and target (orange). (f) Tent maps of signal (blue) and dynamical (orange) and pretrained targets (gray).

below RMSE between signal and (other) pretrained targets]. We find that networks of increasing size can learn Fourier series with increasing order [Figs. 2(c) and 2(d)]. Networks with 3000 neurons learn Fourier series of order 10 with a median fraction of successes of close to 90%. Hence, very general periodic functions can be learned. The highest producible frequency is limited by the available neuronal timescales τ_i. We thus expect that larger networks containing smaller τ_i can learn even higher order targets.

To check if our approach also works for a target family with more than one parameter and multidimensional trajectories, we consider in (iii) a superposition of sines with different amplitude and period (consequently $k, \tilde{c}$ are two-dimensional vectors) and in (iv) a set of fixed points along a curve in three-dimensional space. We find that, after pretraining, our networks are able to dynamically learn unseen members of these target families with multidimensional context or signal, as shown in Figs. 3(a) and 3(b) for example trajectories.

Second, we consider a family $\dot{\tilde{z}}(t) = F(\tilde{z}(t), u(t); k)$ of target dynamical systems. The networks are pretrained on a few representative systems. Thereafter, an unseen one is dynamically learned. Learning is in both phases based on imitation of trajectories. However, in contrast to tasks (i)–(iv) the networks now need to generate unseen output trajectories during testing. To demonstrate dynamical learning of a driven system, we consider task (v) of approximating the trajectory of an overdamped pendulum with drive $u(t)$ and different masses m: $\dot{\tilde{z}}(t) = F(\tilde{z}(t), u(t); m)$. During pretraining and dynamical learning, we use low-pass filtered white noise as drive [Fig. 3(c), left of dashed vertical]. During testing, we use a triangular wave [Fig. 3(c), right of dashed vertical]. As our networks nevertheless generate the correct qualitatively different signal [Figs. 3(c) and 3(d)], they must

have learned the underlying vector field $F(\tilde{z}, u; m)$. (v) also shows that learning goes beyond interpolation of trajectories [compare blue and gray traces in Fig. 3(d)]. Finally, in task (vi) we show dynamical learning of chaotic dynamics, considering autonomous Lorenz systems with different dissipation parameter β of the z variable. For chaotic dynamics, even trajectories of similar systems quickly diverge. The aim in this task is thus only to generate during testing signals of the same type as the trajectories of the target system. We test this by comparing the limit sets of the dynamics and the tent-map relation between subsequent maxima of the z coordinate [Figs. 3(e) and 3(f)]. The reproduction of the tent-map relation further shows that our approach can generate not explicitly trained quantitative dynamical features. We note that the networks also dynamically learn the fixed point convergence of some of the targets in the considered parameter space, even though they were pretrained on chaotic dynamics only [27].

Analysis.—In the following, we analyze the different parts of our network learning and its applicability. One interpretation of the pretraining phase is that the network learns a negative feedback loop, which reduces the error $\varepsilon(t)$. For another interpretation, we split $\varepsilon(t)$ and regroup the z-dependent part of Eq. (1) as $(w_z + w_\varepsilon)z(t) - w_\varepsilon\tilde{z}(t)$: feeding back $\varepsilon(t)$ is equivalent to adding a teacher drive $\tilde{z}(t)$, except for a specific change in the feedback weights w_z. For the z output alone the network thus weight learns an autoencoder $\tilde{z}(t) \to z(t)$. This is usually an easy task for reservoir networks [72]. To simultaneously learn the constant output $c(t) = \tilde{c}$, the network has to choose an appropriate o_c orthogonal to the subspaces in which the different $z(t)$-driving r dynamics take place. Orthogonal directions are available in sufficiently large networks, since the subspaces are low dimensional [73].

After the correct z dynamics are assumed, we have $\varepsilon(t) \approx 0$. Since remaining fluctuations in $\varepsilon(t)$ could stabilize the dynamics, we usually include ensuing learning phases with $w_\varepsilon \to 0$ and $c(t) = \tilde{c}$. These teach the network to generate the correct dynamics in a stable manner under conditions similar to testing.

To analyze the principles underlying dynamical learning and testing, we consider task (i). The similarity of the network and learning setups suggests that the same principles underlie all our tasks. We additionally confirm this for (vi) [27]. Viewing the network dynamics in the space of firing rates r, we choose new coordinates with first axis along o_c and the principal components of the dynamics orthogonal to o_c. The dynamics are then given by $c(t) = o_c r(t), r_{\mathrm{PC1}}(t), r_{\mathrm{PC2}}(t), \ldots$. (Fig. 4). We focus on the first three coordinates, which describe large parts of the dynamics and output generation. We find that during dynamical learning, the error feedback drives the dynamics toward an orbit that is shifted in c but similar to pretrained ones. The network therewith generalizes the pretrained reaching and generation of orbits together with corresponding,

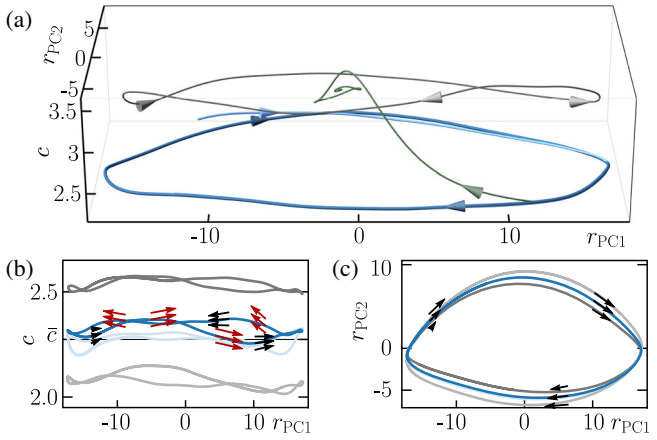

FIG. 4. Network dynamics during dynamical learning (a) and testing (b),(c) of task (i), in $c, r_{\mathrm{PC1}}, r_{\mathrm{PC2}}$ coordinates. (a) During dynamical learning, the error input drives the network to a periodic orbit (light blue trajectory) and keeps it there (blue). Without input, the dynamics converge to a stable orbit (gray) whose signal approximates a pretrained one. Freezing $\tilde{z}(t) = \tilde{z}(t_0)$ drives the dynamics to a fixed point off the orbit (green). (b) During testing, the assumed orbit (blue) in the c-$r_{\mathrm{PC},1}$ plane is similar to the error driven one [light blue, closest pretrained orbits with $c(t)$ fixed to their $\tilde{c}$: gray]. The constant feedback $\bar{c}$ prevents the dynamics from leaving the region where $c(t) \approx \bar{c}$, compare $\dot{r}(r)$ (black vectors, r on or nearby trajectory) with $\dot{r}(r)$ for variable feedback $c(t)$ (red vectors). (c) All four orbits are similar in the $r_{\mathrm{PC},1}$-$r_{\mathrm{PC},2}$ plane. The dynamically learned orbit has an attracting projected vector field (black vectors) like the pretrained orbits.

near-constant $c(t)$, while $\varepsilon(t)$ is fed in. We note that the combination of current state and error input is important [see Fig. 4(a) for $w_\varepsilon \to 0$ and a mismatched $\tilde{z}(t) = \tilde{z}(t_0)$ for $t > t_0$].

During testing, the network generalizes the pretrained characteristics that feeding back $w_c\tilde{c}$ leads to $c(t) \approx \tilde{c}$. Clamping $w_c c(t)$ to $w_c\bar{c}$ thus results in an approximate restriction of $r(t)$ to an $(N-1)$-dimensional hyperplane with $c(t) = o_c r(t) \approx \bar{c}$ [Fig. 4(b)]. The resulting trajectory is for task (i) a stable periodic orbit that generates the desired signal, because the vector field projected to the $c(t) = \bar{c}$ hyperplane is similar to the vector field projected to the $c(t) = \tilde{c}$ hyperplanes embedding nearby pretrained periodic orbits [Fig. 4(c)].

Discussion and conclusion.—We have introduced a scheme for how neural networks can quickly learn dynamics without changing their weights and without requiring a teacher during testing. It relies on a weight-learned mutual association, quasi an entanglement, between contexts and targets. This enables the latter to fix the former during dynamical learning and vice versa during testing.

Previous approaches to supervised dynamical learning with continuous signal space required a form of the teaching signal also during testing. They further differ in network architecture, learning algorithm, task and/or

assumption of discrete time from ours [10–23,27,74–76]. In networks with external input unseen, interpolating input can lead to interpolating dynamics [27,77]. In contrast, our networks *learn* new dynamics, by imitation.

Our scheme is conceptually independent of the network and weight-learning model. The pretraining implements a form of structure learning, i.e., learning of the structure underlying a task family [6,78]. Animals and humans employ it frequently, but little is known about its neurobiology. We thus realize it by a simple reservoir computing scheme with FORCE learning [25,27]. We checked that we can use biologically more plausible weight perturbation learning for a simple fixed point learning task [27].

Dynamical learning is biologically plausible: it is naturally local, causal, and does not require fast synaptic weight updates. Continuous supervision could be generated by an inverse model [79] and might be replaceable by a sparse, partial signal. Our dynamical learning is fast [27] (cf. also [8,10,11,13,14,75]). Even for more complicated tasks convergence requires only a few multiples of a characteristic timescale of the dynamics. Further, we find robustness against changes in network and task parameters [27]. The above points suggest a high potential of our scheme for applications in biology, physics, and engineering such as neuromorphic computing and the prediction of chaotic systems [27].

We thank Paul Züge for fruitful discussions, the German Federal Ministry of Education and Research (BMBF) for support via the Bernstein Network (Bernstein Award 2014, 01GQ1710) and the European Union's Horizon 2020 research and innovation programme for support via the Marie Skłodowska-Curie Grant No. 754411.

[1] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems* (MIT Press, Cambridge, 2001).

[2] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics—From Single Neurons to Networks and Models of Cognition* (Cambridge University Press, Cambridge, England, 2014).

[3] M. G. Perich, J. A. Gallego, and L. E. Miller, A neural population mechanism for rapid learning, Neuron **100**, 964 (2018).

[4] *Learning to Learn*, edited by S. Thrun and L. Pratt (Springer, New York, 1998).

[5] J. Vanschoren, Meta-Learning: A survey, arXiv:1810.03548.

[6] B. J. Lansdell and K. P. Kording, Towards learning to learn, arXiv:1811.00231.

[7] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, RL^2: Fast reinforcement learning via slow reinforcement learning, arXiv:1611.02779.

[8] J. X. Wang, Z. Kurth-Nelson, D. Kumaran, D. Tirumala, H. Soyer, J. Z. Leibo, D. Hassabis, and M. Botvinick, Prefrontal cortex as a meta-reinforcement learning system, Nat. Neurosci. **21**, 860 (2018).

[9] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, Learning to adapt in dynamic, real-world environments through metareinforcement learning, arXiv:1803.11347.

[10] L. A. Feldkamp, G. V. Puskorius, and P. C. Moore, Adaptation from fixed weight dynamic networks, in *Proceedings of International Conference on Neural Networks (ICNN'96)* (IEEE, Washington, 1996), Vol. 1, pp. 155–160, https://dx.doi.org/10.1109/ICNN.1996.548883.

[11] L. A. Feldkamp, G. V. Puskorius, and P. C. Moore, Adaptive behavior from fixed weight networks, Information Sciences (NY) **98**, 217 (1997).

[12] A. S. Younger, P. R. Conwell, and N. E. Cotter, Fixed-weight on-line learning, IEEE Trans. Neural Networks **10**, 272 (1999).

[13] S. Hochreiter, A. S. Younger, and P. R. Conwell, Learning to learn using gradient descent, in *Artificial Neural Networks—ICANN 2001*, Lecture Notes in Computer Science, edited by G. Dorffner, H. Bischof, and K. Hornik (Springer, Berlin, Heidelberg, 2001), pp. 87–94, https://dx.doi.org/10.1007/3-540-44668-0_13.

[14] A. S. Younger, S. Hochreiter, and P. R. Conwell, Meta-learning with backpropagation, in *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)* (IEEE, Washington, 2001), Vol. 3, pp. 2001–2006, https://dx.doi.org/10.1109/IJCNN.2001.938471.

[15] L. A. Feldkamp, D. V. Prokhorov, and T. M. Feldkamp, Simple and conditioned adaptive behavior from Kalman filter trained recurrent networks, Neural Netw. **16**, 683 (2003).

[16] R. A. Santiago, Context discerning multifunction networks: Reformulating fixed weight neural networks, in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)* (IEEE, Budapest, 2004), Vol. 1, pp. 189–194, https://dx.doi.org/10.1109/IJCNN.2004.1379896.

[17] M. Lukosevicius, Echo state networks with trained feedbacks, Jacobs University Bremen, Technical Report No. 4, 2007.

[18] A. Santoro, S. Bartunov, M. Botvinick, and D. Wierstra, and T. Lillicrap, Meta-learning with memory-augmented neural networks, in *International Conference on Machine Learning* (PMLR, New York, 2016), pp. 1842–1850.

[19] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, Long short-term memory and learning to learn in networks of spiking neurons, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., New York, 2018), pp. 787–797 [arXiv:1803.09574].

[20] L. A. Feldkamp and G. V. Puskorius, Training of robust neural controllers, in *Proceedings of 1994 33rd IEEE Conference on Decision and Control* (IEEE, Lake Buena Vista, 1994), Vol. 3, pp. 2754–2759, https://dx.doi.org/10.1109/CDC.1994.411384.

[21] L. A. Feldkamp and G. V. Puskorius, Training controllers for robustness: Multi-stream DEKF, in *Proceedings of 1994 IEEE International Conference on Neural Networks*

(ICNN'94) (IEEE, Orlando, FL, 1994), Vol. 4, pp. 2377–2382.

[22] L. A. Feldkamp and G. V. Puskorius, Fixed-weight controller for multiple systems, in *Proceedings of International Conference on Neural Networks (ICNN'97)* (IEEE, Houston, 1997), Vol. 2, pp. 773–778, https://dx.doi.org/10.1109/ICNN.1997.616120.

[23] M. Oubbati and G. Palm, A neural framework for adaptive robot control, Neural Comput. Appl. **19**, 103 (2010).

[24] H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert, Optimization and applications of echo state networks with leaky-integrator neurons, Neural Netw. **20**, 335 (2007).

[25] D. Sussillo and L. F. Abbott, Generating coherent patterns of activity from chaotic neural networks, Neuron **63**, 544 (2009).

[26] D. Sussillo and O. Barak, Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks, Neural Comput. **25**, 626 (2013).

[27] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.125.088103 for further detail on the tasks, analyses of the learning performance, pretraining using weight perturbation and an extended discussion, which includes Refs. [28–65].

[28] D. V. Buonomano and M. M. Merzenich, Temporal information transformed into a spatial code by a neural network with realistic properties, Science **267**, 1028 (1995).

[29] P. F. Dominey, Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning, Biol. Cybern. **73**, 265 (1995).

[30] M. B. Westover, C. Eliasmith, and C. H. Anderson, Linearly decodable functions from neural population codes, Neurocomputing **44–46**, 691 (2002).

[31] T. Miconi, Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks, eLife **6**, e20899 (2017).

[32] B. DePasquale, C. J. Cueva, K. Rajan, G. S. Escola, and L. F. Abbott, Full-FORCE: A target-based method for training recurrent networks, PLoS One **13**, e0191527 (2018).

[33] https://github.com/chklos/dynamical-learning.

[34] O. Schütze, X. Esquivel, A. Lara, and C. A. C. Coello, Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. **16**, 504 (2012).

[35] H. Jaeger, Controlling recurrent neural networks by conceptors, Jacobs University Bremen, Technical Report No. 31, 2014.

[36] H. Jaeger, Using conceptors to manage neural long-term memories for temporal patterns, J. Mach. Learn. Res. **18**, 1 (2017), https://jmlr.org/papers/v18/15-449.html.

[37] A. Dembo and T. Kailath, Model-free distributed learning, IEEE Trans. Neural Networks **1**, 58 (1990).

[38] M. Jabri and B. Flower, Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks, IEEE Trans. Neural Networks **3**, 154 (1992).

[39] G. Cauwenberghs, A fast stochastic error-descent algorithm for supervised learning and optimization, in *Advances in Neural Information Processing Systems* (Morgan-Kaufmann, San Mateo, 1993), Vol. 5, pp. 244–251.

[40] H. S. Seung, Learning in spiking neural networks by reinforcement of stochastic synaptic transmission, Neuron **40**, 1063 (2003).

[41] C. Beer and O. Barak, One step back, two steps forward: Interference and learning in recurrent neural networks, arXiv:1805.09603.

[42] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *Proceedings of the 3rd International Conference on Learning Representations* (ICLR, San Diego, 2015), Vol. 3.

[43] R. L. Redondo and R. G. M. Morris, Making memories last: The synaptic tagging and capture hypothesis, Nat. Rev. Neurosci. **12**, 17 (2011).

[44] Z. Yu, D. Kappel, R. Legenstein, S. Song, F. Chen, and W. Maass, CaMKII activation supports reward-based neural network optimization through Hamiltonian sampling, arXiv:1606.00157.

[45] W. C. Abraham, Metaplasticity: Tuning synapses and networks for plasticity, Nat. Rev. Neurosci. **9**, 387 (2008).

[46] G. V. Puskorius and L. A. Feldkamp, Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks, IEEE Trans. Neural Networks **5**, 279 (1994).

[47] E. D. Sontag, Neural nets as systems models and controllers, in *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems* (Yale University, New Haven, 1992), pp. 73–79.

[48] K-I. Funahashi and Y. Nakamura, Approximation of dynamical systems by continuous time recurrent neural networks, Neural Netw. **6**, 801 (1993).

[49] K.-K. K. Kim, E. R. Patrón, and R. D. Braatz, Standard representation and unified stability analysis for dynamic artificial neural network models, Neural Netw. **98**, 251 (2018).

[50] N. E. Cotter and P. R. Conwell, Fixed-weight networks can learn, in *Proceedings of the 1990 IJCNN International Joint Conference on Neural Networks* (IEEE, San Diego, 1990), Vol. 3, pp. 553–559, https://dx.doi.org/10.1109/IJCNN.1990.137898.

[51] N. E. Cotter and P. R. Conwell, Learning algorithms and fixed dynamics, in *Proceedings of the IJCNN-91-Seattle International Joint Conference on Neural Networks* (IEEE, Seattle, 1991), Vol. 1, pp. 799–801, https://dx.doi.org/10.1109/IJCNN.1991.155280.

[52] S. Lim and M. S. Goldman, Balanced cortical microcircuitry for maintaining information in working memory, Nat. Neurosci. **16**, 1306 (2013).

[53] W. Maass, P. Joshi, and E. D. Sontag, Computational aspects of feedback in neural circuits, PLoS Comput. Biol. **3**, e165 (2007).

[54] R. Pascanu and H. Jaeger, A neurodynamical model for working memory, Neural Netw. **24**, 199 (2011).

[55] J. A. Gallego, M. G. Perich, L. E. Miller, and S. A. Solla, Neural manifolds for the control of movement, Neuron **94**, 978 (2017).

[56] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, Neuromorphic electronic circuits for building autonomous cognitive systems, Proc. IEEE **102**, 1367 (2014).

[57] F. Duport, A. Smerieri, A. Akrout, M. Haelterman, and S. Massar, Fully analogue photonic reservoir computer, Sci. Rep. **6**, 22381 (2016).

[58] A. N. Tait, T. F. de Lima, E. Zhou, A. X. Wu, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, Neuromorphic photonic networks using silicon photonic weight banks, Sci. Rep. **7**, 7430 (2017).

[59] P. Antonik, M. Haelterman, and S. Massar, Brain-Inspired Photonic Signal Processor for Generating Periodic Patterns and Emulating Chaotic Systems, Phys. Rev. Applied **7**, 054014 (2017).

[60] D. Thalmeier, M. Uhlmann, H. J. Kappen, and R.-M. Memmesheimer, Learning universal computations with spikes, PLoS Comput. Biol. **12**, e1004895 (2016).

[61] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, A survey of neuromorphic computing and neural networks in hardware, arXiv:1705.06963.

[62] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach, Phys. Rev. Lett. **120**, 024102 (2018).

[63] R. S. Zimmermann and U. Parlitz, Observing spatio-temporal dynamics of excitable media using reservoir computing, Chaos **28**, 043118 (2018).

[64] K. K. Sreenivasan and M. D'Esposito, The what where and how of delay activity, Nat. Rev. Neurosci. **20**, 466 (2019).

[65] K. Oberauer, Is rehearsal an effective maintenance strategy for working memory?, Trends Cognit. Sci. **23**, 798 (2019).

[66] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).

[67] W. Maass, T. Natschläger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Comput. **14**, 2531 (2002).

[68] M. Lukosevicius, H. Jaeger, and B. Schrauwen, Reservoir computing trends, Künstl. Intell. **26**, 365 (2012).

[69] M. Y. Ismail and J. C. Principe, Equivalence between RLS algorithms and the ridge regression technique, in *Conference Record of The Thirtieth Asilomar Conference on Signals, Systems and Computers* (IEEE, Pacific Grove, 1996), Vol. 2, pp. 1083–1087, https://dx.doi.org/10.1109/ACSSC.1996.599110.

[70] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, Context-dependent computation by recurrent dynamics in prefrontal cortex, Nature (London) **503**, 78 (2013).

[71] G. Hennequin, T. P. Vogels, and W. Gerstner, Optimal control of transient dynamics in balanced networks supports generation of complex movements, Neuron **82**, 1394 (2014).

[72] L. F. Abbott, B. DePasquale, and R-M. Memmesheimer, Building functional networks of spiking model neurons, Nat. Neurosci. **19**, 350 (2016).

[73] L. F. Abbott, K. Rajan, and H. Sompolinsky, Interactions between intrinsic and stimulus evoked activity in recurrent neural networks, in *The Dynamic Brain: An Exploration of Neuronal Variability and Its Functional Significance* (Oxford University Press, Oxford, 2011), pp. 65–82, https://dx.doi.org/10.1093/acprof:oso/9780195393798.003.0004.

[74] M. Oubbati, P. Levi, and M. Schanz, Meta-learning for adaptive identification of non-linear dynamical systems, in *Proceedings of the 2005 IEEE International Symposium on, Mediterranean Conference on Control and Automation Intelligent Control, 2005* (IEEE, Limassol, 2005), pp. 473–478, https://dx.doi.org/10.1109/.2005.1467061.

[75] H. Jaeger and D. Eck, Can't get you out of my head: A connectionist model of cyclic rehearsal, in *Modeling Communication with Robots and Virtual Humans*, edited by I. Wachsmuth and G. Knoblich (Springer, Berlin, Heidelberg, 2008), pp. 310–335, https://dx.doi.org/10.1007/978-3-540-79037-2_17.

[76] F. wyffels, J. Li, T. Waegeman, B. Schrauwen, and H. Jaeger, Frequency modulation of large oscillatory networks, Biol. Cybern. **108**, 145 (2014).

[77] K. J. Boström, H. Wagner, M. Prieske, and M. de Lussanet, Model for a flexible motor memory based on a self-active recurrent neural network, Human Movement Science **32**, 880 (2013).

[78] D. A. Braun, C. Mehring, and D. M. Wolpert, Structure learning in action, Behavioural Brain Research **206**, 157 (2010).

[79] M. I. Jordan and D. E. Rumelhart, Forward models: Supervised learning with a distal teacher, Cogn. Sci. **16**, 307 (1992).