Dynamical learning of dynamics

Christian Klos, Yaroslav Felipe Kalle Kossio, Sven Goedeke, Aditya Gilra, and Raoul-Martin Memmesheimer

Neural Network Dynamics and Computation, Institute of Genetics, University of Bonn, Bonn, Germany.

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 already static neural networks can learn to generate required dynamics by imitation. After appropriate weight pretraining, the networks 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.

Introduction. The predominant paradigm for learning in biological or artificial neural networks assumes that slow modification of the connection weights between neurons aims at reaching fixed weights that are appropriate to achieve the desired task [1, 2]. Indeed, a recurrent neural network with appropriate static weights can approximate any smooth dynamics with bounded inputs for finite time [3–5]. However, this also implies that a neural network with static weights can in principle approximate the combined dynamics of the state and weight variables of another, weight-learning neural network. The static network thereby dynamically implements the other network’s learning algorithm [6, 7]. Learning in turn the static network’s weights is a kind of meta learning or learning to learn [8–10].

There is a spurt of interest in learning to learn [9, 10], which focuses mainly on learning of reinforcement learning, i.e., on learning with delayed, often unspecific reward [11–13]. Another direction of research is on learning of supervised learning with a continually present teacher: the considered systems typically learn dynamically to predict time series for the current time step given the preceding step’s desired output [14–23] or to track a desired time-varying state variable [24–27].

Here we investigate the possibility of learning to learn the self-contained long-term generation of autonomous and driven dynamics. We consider models for biological recurrent 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, 28–30]. After appropriate weight-learning, the synaptic weights are fixed. We find that the networks can nevertheless learn to generate new dynamics. Furthermore, they continue to generate these dynamics in absence of a teacher during subsequent testing. We illustrate this with a variety of trajectories and dynamical systems. Further, we provide an analysis of the underlying mechanisms using dynamical systems theory.

Network model. We use recurrent neural networks, where each neuron (or neuronal subpopulation) \( i, i = 1, ..., N \), is characterized by an activation variable \( x_i(t) \) and communicates with other neurons via its firing rate \( r_i(t) \), which is a nonlinear function of \( x_i(t) \) [1, 2]. The network has two outputs, which can be interpreted as linear neurons: a signal output \( z_k(t), k = 1, ..., N_z \), and a context output \( c_l(t), l = 1, ..., N_c \) (Fig. 1). Their weights are the only plastic ones. 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 [31]. During learning, our networks are temporarily also informed about their output’s difference from the target \( \tilde{z}(t) \) by an error input \( \varepsilon(t) = z(t) - \tilde{z}(t) \). When this input is absent, we set \( \varepsilon(t) = 0 \) and the output of \( c(t) \) to a constant value. In isolation \( x_i(t) \) decays to zero with a time constant \( \tau_i \) that combines the decay times of membrane potential and synaptic currents. Unless mentioned otherwise, we set \( \tau_i = 1 \) fixing the overall time scale. Taken together, for constant weights the network dynamics are given by

\[
\tau \dot{x}(t) = -x(t) + Ar(t) + w_z z(t) + w_c c(t) + w_v \varepsilon(t) + w_u u(t),
\]

\[
z(t) = o z r(t), \quad c(t) = o c r(t),
\]
with recurrent weights $A$, the diagonal matrix of time constants $\tau$, signal and context output weights $o_z$ and $o_c$, feedback weights $w_z$, $w_c$ and input weights $w_x$, $w_u$. We choose $r_i(t) = \tanh(x_i(t) + b_i)$ [29, 31, 32], where $b_i$ is a constant offset breaking the $x \rightarrow -x$ symmetry without input.

**Weight learning.** The aim of our weight learning is to enable the resulting static recurrent networks to learn dynamics of a specific class. For this, we present different trajectories $\tilde{z}(t)$ of this class as targets and associate each of them uniquely with a desired constant index $\tilde{c}$. The different signals and indexes are presented as a continuous, randomly repeating sequence of training periods. During the first part of each training period, a network receives error feedback on the dynamics as additional input, $\varepsilon(t) = z(t) - \tilde{z}(t)$, (Fig. 1a). 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 the tasks, after a time $t_0$, 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. The network thus learns to continue generating $z(t) \approx \tilde{z}(t)$ without error feedback input. For the weight learning of the Lorenz system below, $\varepsilon(t)$ is always provided to the network and for the overdamped pendulum, $c(t)$ is additionally never fixed.

The output weights to $z(t)$ and $c(t)$ learn online according to the FORCE rule [29]. In short, the outputs are trained 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 the targets [33].

**Dynamical learning and testing.** The weights now remain static and the networks learn by their dynamics new tasks, i.e., the generation of previously unseen signals $\tilde{z}(t)$. For this, a network receives the teacher signal $\varepsilon(t) = z(t) - \tilde{z}(t)$ as input. In our applications, we show that networks can generalize their previously learned behavior and approach $z(t) \approx \tilde{z}(t)$ and a moderately fluctuating $c(t)$. After a learning time $t_{\text{learn}}$, which may be different from $t_0$, the test phase begins, where no more teacher signal is present, $\varepsilon(t) = 0$. In weight learning paradigms, during such phases the weights are fixed to temporarily constant values [29, 31, 34–36]; if gains are learned, the gains are fixed [37]. 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 an indication 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 $z(t)$ during testing, establishing the successful dynamical learning of the task.

**Applications.** We illustrate our approach by learning a variety of trajectories (tasks (i-iii)) and dynamical systems (tasks (iv,v)). Firstly, the networks learn to approximate (i) a sinusoidal oscillation, (ii) a superposition of sines and (iii) a fixed point. In each task, we consider a family $\tilde{z}(t;k)$ of target trajectories of the same type, parameterized by some $k$. The networks are weight-pretrained on a few of them, where the context target $\tilde{c}$ is an invertible function of $k$. Thereafter the networks dynamically learn to generate a previously unseen trajectory as output and perpetuate it during testing. The family consists in task (i) of oscillations with different periods, in (ii) of a signal with different amplitude and period (consequently $k$ and $\tilde{c}$ are two-dimensional vectors) and in (iii) of a set of fixed points along a curve in three-dimensional space. Secondly, the networks learn (iv) a driven overdamped pendulum and (v) autonomous chaotic Lorenz dynamics. In these tasks, we consider a family $\tilde{z}(t) = F(\tilde{z}(t), u(t); k)$ of target dynamical systems. In (iv) a drive $u(t)$ is present, the pendulum mass varies. In (v) the dynamics vary in the dissipation parameter $\beta$ of the $z$-variable. The networks are weight-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-iii) the networks now need to generate unseen output trajectories during testing. For task (iv), the aim is to approximate the trajectories that the target dynamical system would generate, if it was fed with the same previously unseen testing drive. For chaotic dynamics as in task (v), even trajectories of similar systems quickly diverge. The aim in this task is thus only to generate in the testing phase output signals of the same type as the trajectories of the target Lorenz system. We test this by comparing the limit sets of the dynamics and the tent map relation between subsequent maxima of the $z$-coordinate.

We find that our networks faithfully dynamically learn the desired dynamics in the different tasks and continue to generate them during testing, for parameter sets interpolating the weight pretrained ones and slightly beyond (Fig. 2, [38]). The tasks demonstrate learning of simple trajectories, useful for analysis (i), learning of trajectories in a family with two parameters (ii) and learning of multidimensional trajectories (iii). Task (iv) shows learning of a driven dynamical system and learning with qualitatively different drive than used in testing. Further, it shows that learning goes beyond interpolation of trajectories (compare blue and gray traces in Fig. 2d). Task (v) shows learning of a chaotic dynamical system and with the tent map the generation of 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 weight-trained on chaotic dynamics only.

**Analysis.** In the following we analyze the different parts of our network learning. One interpretation of the weight learning phase is that the network learns a neg-
novative 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 $(w_z + w_c)z(t) - w_z \hat{z}(t)$: feeding back $\varepsilon(t)$ is equivalent to adding a teacher drive $\hat{z}(t)$, except for a specific change in the feedback weights $w_z$. For the $z$-output alone the network thus weight-learns an autoencoder $\hat{z}(t) \rightarrow z(t)$. This is usually an easy task for reservoir networks [39]. To simultaneously learn the constant output $c(t) = \hat{c}$, 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 [40].

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 $\varepsilon(t) = 0$ and $c(t) = \hat{c}$. These teach the network to generate the correct dynamics in stable manner under conditions similar to testing.

To analyze the principles underlying dynamical learning and testing, we consider task (i). 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 $c(t) = o_c r(t)$ and $r_{PC1}(t)$, $r_{PC2}(t)$,... (Fig. 3a). 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 towards an orbit that is similar to weight-trained ones (Fig. 3). The network therewith generalizes the weight-learned reaching and generation of periodic orbits together with corresponding, near-constant $c(t)$. We note that the combination of current state and teacher signal is important to
keep the periodic behavior (see Fig. 3a for $\varepsilon(t) = 0$ and a mismatched $\tilde{z}(t) = \tilde{z}(t_0)$ for $t > t_0$).

During testing, the network generalizes the weight-learned characteristics that feeding back $w_c \tilde{c}$ leads to $c(t) \approx \tilde{c}$. Clamping $w_c, c(t)$ to $w_c \tilde{c}$ thus results in an approximate restriction of $x(t)$ to an $N - 1$-dimensional hyperplane with $c(t) = o_r(t) \approx \tilde{c}$ (Fig. 3b). The resulting trajectory is a stable periodic orbit, because the vector field projected to the $c(t) = \tilde{c}$-hyperplane is similar to the vector field projected to the $c(t) = \bar{c}$-hyperplanes embedding nearby weight-learned periodic orbits (Fig. 3c).

Discussion and conclusion. We have shown that static neural networks can dynamically learn trajectories and dynamical systems. During the initial weight learning (“learning to learn”), the networks are taught several dynamics from the same family as the later dynamically learned ones, as well as a corresponding constant context signal. The process is supervised by an error signal to the synapses and, part of the time, by an error input to the network. During dynamical learning, the latter alone suffices to teach the desired dynamics. The network then also generates a context signal, which fluctuates around some temporal mean. When subsequently testing the generation of the dynamics, the teaching error input is removed and the context signal is fixed to its average, telling that the learned dynamics should be continued.

Our analysis indicates that the scheme works due to an interplay of generalization and stabilization: In short, during weight learning, the networks adapt to perform a negative feedback/autoencoder task. During dynamical learning, the networks generalize this behavior, by generating new desired outputs when receiving their errors as input. In subsequent testing, the learned output dynamics continue, stabilized by the constant context signal. This is possible because a mutual association, quasi an entanglement, between contexts and targets was weight-learned. It enables the latter to fix the former during dynamical learning and vice versa during testing. We note that the recent ‘conceptor’ approach suggests to fix reservoir dynamics by weight changes [41, 42].

Approaches to supervised dynamical learning in the literature consider the one-step prediction of time series [14, 15] and the approximation of input-output maps [16–18, 20, 22, 23], where the correct previous output is fed in. Other networks could adapt their dynamics to provide negative feedback for control [25–27, 43]. Learning of supervised learning has also been used to identify the parameters of a dynamical system [44] or perform optimization [45, 46]. The studies use simple recurrent neural networks [14, 15, 20, 25–27, 43, 44], gated [17, 18, 22, 45, 46] or spiking ones [23]. The first networks are similar to ours but do not use leaky neurons and often assume discrete time. Weight pretraining in the different studies used backpropagation [17, 18, 22, 23, 45, 46] or extended Kalman filtering [14, 15, 19, 24–27, 43, 44]. To our knowledge, all systems were fed a form of the temporally variable teaching signal also during testing and thus do not generate desired dynamics in a self-contained manner.

In our networks, fixing the intrinsically chosen context signal $c(t)$ indicates that the dynamics is to be continued. This is analogous to fixing the weights during testing in weight-learning paradigms. It is necessary to avoid convergence to other dynamics (if the system has discrete attractors) or diffusion and drift (for marginally stable dynamics). During testing, $c(t)$ is constant. It is thus much simpler than usual teacher and target signals and can be kept up by biologically plausible circuits [47]. For long times, weight learning may consolidate it. Focusing on dynamical learning, we have straightforwardly specified the $\tilde{c}$. In biological systems, they might be derived from the teacher dynamics. We note that one can also teach networks with external input such that unseen, interpolating input leads to interpolating dynamics [38]. In contrast to such generalization, our networks learn their new dynamics, dynamically from a teacher.

We employ networks that are rate-based models for biological neural networks [1, 2]. Their weights are initially adapted with the FORCE rule [29]. FORCE changes only the readout weights, which is equivalent to a low rank correction of the recurrent weight matrix [48]. Only a fraction of the degrees of freedom of the recurrent weight matrix are therefore used for task-related adaptation, in contrast to more complicated and powerful rules like backpropagation through time and extensions of FORCE [49]. The abilities of such networks to achieve the displayed dynamical learning tasks suggests a high potential of the scheme for applications in biology, physics and engineering.

In experimental physics and engineering, our scheme may find application in neuromorphic computing. Here, intrinsically plastic weights are costly and often difficult to realize, while outsourcing the learning to external controllers introduces computational bottlenecks [50]. As an example, in analog, photonic neuromorphic computing, network weights are externally set to generate desired output dynamics [51–53]. Our scheme may allow such systems to intrinsically learn and thereby fully reap their speed benefits. For spiking hardware, our networks may be efficiently translated into spiking ones [54]. Dynamical learning may help to reduce the size and power consumption of such hardware, for example in autonomous robots that adjust their movements [55].

Our approach suggests a new method for the prediction of chaotic systems [56, 57], which searches for similarity within a predefined family of dynamics and leaves the networks structurally invariant and flexible.

The presented results indicate that biological neural networks, with their much larger size and with structures shaped by evolution and powerful plasticity, may well use dynamical learning. A possible example is the quick learning of new movements [58], perhaps with sub-
sequent consolidation by plasticity. Another example may be short term memory of temporal sequences. Our theory predicts that even complicated dynamics may be memorized in biological neural networks without synaptic modification.

We thank Paul Ziuge for fruitful discussions and the German Federal Ministry of Education and Research (BMBF) for support via the Bernstein Network (Bernstein Award 2014, 01GQ1710).

[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, 2014).
[3] E. D. Sontag, “Neural Nets as Systems Models and Controllers,” in *Proc. Seventh Yale Workshop on Adaptive and Learning Systems* (1992) pp. 73–79.
[4] K-I. Funahashi and Y. Nakamura, “Approximation of dynamical systems by continuous time recurrent neural networks,” Neural Networks 6, 801–806 (1993).
[5] K-K. K. Kim, E. R. Patrón, and R. D. Braatz, “Standard representation and unified stability analysis for dynamic artificial neural network models,” Neural Networks 98, 251–262 (2018).
[6] N. E. Cotter and P. R. Conwell, “Fixed-weight networks can learn,” in 1990 IJCNN International Joint Conference on Neural Networks (1990) pp. 553–559 vol.3.
[7] N. E. Cotter and P. R. Conwell, “Learning algorithms and fixed dynamics,” in *IJCNN-91-Seattle International Joint Conference on Neural Networks* (1991) pp. 799–801 vol.1.
[8] S. Thrun and L. Pratt, eds., *Learning to Learn* (Springer US, 1998).
[9] Joaquin Vanschoren, “Meta-Learning: A Survey,” arXiv:1810.03548 (2018).
[10] B. J. Landsdell and K. P. Kording, “Towards learning-to-learn,” arXiv:1811.00231 (2018).
[11] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, “RL²: Fast Reinforcement Learning via Slow Reinforcement Learning,” arXiv:1611.02779 (2016).
[12] 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, 860–868 (2018).
[13] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, “Learning to Adapt: Meta-Learning for Model-Based Control,” arXiv:1803.11347 (2018).
[14] 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)*, Vol. 1 (1996) pp. 155–160 vol.1.
[15] L. A. Feldkamp, G. V. Puskorius, and P. C. Moore, “Adaptive behavior from fixed weight networks,” Information Sciences 98, 217–235 (1997).
[16] A. S. Younger, P. R. Conwell, and N. E. Cotter, “Fixed-weight on-line learning,” IEEE Transactions on Neural Networks 10, 272–283 (1999).
[17] 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.
[18] 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)* (2001) pp. 2001–2006 vol.3.
[19] L. A. Feldkamp, D. V. Prokhorov, and T. M. Feldkamp, “Simple and conditioned adaptive behavior from Kalman filter trained recurrent networks,” Neural Networks 16, 683–689 (2003).
[20] 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)*, Vol. 1 (2004) pp. 189–194.
[21] M. Lukosevicius, *Echo State Networks with Trained Feedback*, Tech. Rep. (Jacobs University Bremen, 2007).
[22] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “Meta-Learning with Memory-Augmented Neural Networks,” in *International Conference on Machine Learning* (2016) pp. 1842–1850.
[23] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, “Long short-term memory and learning-to-learn in networks of spiking neurons,” arXiv:1803.09574 (2018).
[24] L. A. Feldkamp and G. V. Puskorius, “Training of robust on-line learning,” in *Proceedings of 1994 33rd IEEE Conference on Decision and Control* (1994) pp. 2754–2759 vol.3.
[25] 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)*, Vol. 4 (IEEE, Orlando, FL, USA, 1994) pp. 2377–2382.
[26] L. A. Feldkamp and G. V. Puskorius, “Fixed-weight controller for multiple systems,” in *Proceedings of International Conference on Neural Networks (ICNN’97)* (1997) pp. 773–778 vol.2.
[27] M. Oubbati and G. Palm, “A neural framework for adaptive robot control,” Neural Computing and Applications 19, 103–114 (2010).
[28] H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert, “Optimization and applications of echo state networks with leaky-integrator neurons,” Neural Networks 20, 335–352 (2007).
[29] D. Susillo and L. F. Abbott, “Generating coherent patterns of activity from chaotic neural networks,” Neuron 63, 544–557 (2009).
[30] D. Susillo and O. Barak, “Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks,” Neural Comput. 25, 626–649 (2013).
[31] H. Jaeger and H. Haas, “Harnessing linearity: Predicting chaotic systems and saving energy in wireless communication,” Science 304, 78–80 (2004).
[32] M. Lukosevicius, H. Jaeger, and B. Schrauwen, “Reservoir computing trends,” Künstl. Intell. 26, 365–371 (2012).
[33] M. Y. Ismail and J. C. Principe, “Equivalence between
RLS algorithms and the ridge regression technique,” in Proc. Systems and Computers Conf. Record of The Thirtieth Asilomar Conf. Signals (1996) pp. 1083–1087 vol.2.

[34] 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, 2531–2560 (2002).

[35] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, “Context-dependent computation by recurrent dynamics in prefrontal cortex.” Nature 503, 78–84 (2013).

[36] G. Hennequin, T. P. Vogels, and W. Gerstner, “Optimal control of transient dynamics in balanced networks supports generation of complex movements,” Neuron 82, 1394–1406 (2014).

[37] J. P. Stroud, M. A. Porter, G. Hennequin, and T. P. Vogels, “Motor primitives in space and time via targeted gain modulation in cortical networks.” Nat. Neurosci. 21, 1774–1783 (2018).

[38] See Supplemental Material for further detail on the tasks, analyses of the learning performance and further figures, which includes Refs. [59].

[39] L. F. Abbott, B. DePasquale, and R-M. Memmesheimer, “Building functional networks of spiking neurons,” Nat. Neurosci. 19, 350–355 (2016).

[40] L. F. Abbott, K. Rajan, and H. Sompolinsky, “Neuronal variability and its functional significance,” (Oxford University Press, 2010) Chap. Interactions between Intrinsic and Stimulus Evoked Activity in Recurrent Neural Networks, pp. 65–82.

[41] H. Jaeger, “Controlling Recurrent Neural Networks by Conceptors,” arXiv:1403.3369 (2014).

[42] H. Jaeger, “Using Conceptors to Manage Neural Long-Term Memories for Temporal Patterns,” Journal of Machine Learning Research 18, 1–43 (2017).

[43] G. V. Puskorius and L. A. Feldkamp, “Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks,” IEEE Trans. Neural Netw. 5, 279–297 (1994).

[44] 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, (2005) pp. 473–478.

[45] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, “Learning to learn by gradient descent by gradient descent,” arXiv:1606.04474 (2016).

[46] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. de Freitas, “Learning to Learn without Gradient Descent by Gradient Descent,” in International Conference on Machine Learning (2017) pp. 748–756.

[47] S. Lim and M. S. Goldman, “Balanced cortical microcircuitry for maintaining information in working memory,” Nat. Neurosci. 16, 1306–1314 (2013).

[48] D. Sussillo and L. F. Abbott, “Transferring learning from external to internal weights in echo-state networks with sparse connectivity,” PLoS One 7, e37372 (2012).

[49] B. DePasquale, C. J. Cueva, Kanaka Rajan, G. S. Escola, and L. F. Abbott, “Full-force: A target-based method for training recurrent networks,” PLoS One 13, e0191527 (2018).

[50] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, “Neuromorphic electronic circuits for building autonomous cognitive systems,” Proc. IEEE 102, 1367–1388 (2014).

[51] F. Duport, A. Smerieri, A. Akrout, M. Haelterman, and S. Massar, “Fully analogue photonic reservoir computer,” Sci. Rep. 6 (2016), 10.1038/srep22381.

[52] A. N. Tait, T. F. de Lima, E. Zhou, A. X. Wu, M. A. Namias, B. J. Shastri, and P. R. Prucnal, “Neuromorphic photonic networks using silicon photonic weight banks,” Sci. Rep. 7 (2017). 10.1038/s41598-017-07754-z.

[53] P. Antonik, M. Haelterman, and S. Massar, “Brain-inspired photonic signal processor for generating periodic patterns and emulating chaotic systems,” Phys. Rev. Appl 7 (2017), 10.1103/physrevapplied.7.054014.

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

[55] 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 (2017).

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

[57] R. S. Zimmermann and U. Parlitz, “Observing spatiotemporal dynamics of excitable media using reservoir computing,” Chaos 28, 043118 (2018).

[58] M. G. Perich, J. A. Gallego, and L. E. Miller, “A neural population mechanism for rapid learning,” Neuron 100, 964–976.e7 (2018).

[59] Oliver Schuetze, Xavier Esquivel, Adriana Lara, and Carlos A. Coello Coello, “Using the averaged hausdorff distance as a performance measure in evolutionary multi-objective optimization.” IEEE Trans. Evolutionary Computation 16, 504–522 (2012).
Dynamical learning of dynamics
– Supplemental Material –

Christian Klos, Yaroslav Felipe Kalle Kossio, Sven Goedeke, Aditya Gilra, and Raoul-Martin Memmesheimer

Neural Network Dynamics and Computation,
Institute of Genetics, University of Bonn, Bonn, Germany
I. ADDITIONAL DETAIL ON THE APPLICATIONS

In the following, we detail the parameters, setups and targets used in the different applications. We denote the duration of weight learning by $t_{\text{wlearn}}$. Each training period (individual target presentation) within lasts for $t_{\text{stay}}$. If not mentioned otherwise, in the beginning of each period until $t_{\text{fb}}$, the network receives error input $\varepsilon(t) = z(t) - \tilde{z}(t)$ and the context signal evolves freely. Thereafter, $\varepsilon(t) = 0$ and $c(t)$ is fixed to its target value. The intervals between updates of the output weights have random lengths with an average of 0.5 [1]. The parameter of the FORCE rule (cf. [2]) is $\alpha = 1$. Dynamical learning lasts for $t_{\text{learn}} = 1000$. During dynamical learning, we determine $\tilde{c}$ by averaging the context signal with an exponentially forgetting kernel ($\tau_{\text{forget}} = 100$). Testing lasts for $t_{\text{test}}$.

In all applications, recurrent weights $A_{ij}$ are set to zero with probability $1 - p$. Nonzero weights are drawn from a Gaussian distribution with mean 0 and variance $\frac{g^2}{pN}$, where $g = 1.5$ [2]. Furthermore, we draw the feedback weights $w_{z,ij}$, $w_{c,ij}$ and the input weights $w_{\varepsilon,ij}$, $w_{u,ij}$ from a uniform distribution between $-\tilde{w}$ and $\tilde{w}$, initially set all output weights $o_{z,ij}$ and $o_{c,ij}$ to 0 and draw the biases $b_i$ from a uniform distribution between $-0.2$ and 0.2. The number of external inputs is $N_u$. We use the standard Euler method for our simulations, with an integration time step of $dt = 0.1$, except for Fig. 3, where we use $dt = 0.01$.

Further settings in the individual tasks are as follows:

Task (i): $N = 500, N_z = 1, N_c = 1, N_u = 0, p = 0.1, \tilde{w} = 1, t_{\text{stay}} = 500, t_{\text{fb}} = 100, t_{\text{wlearn}} = 50000, t_{\text{test}} = 5000$. The network learns to generate sinusoidal oscillations with period $T$. The family of target trajectories is $\tilde{z}(t;T) = 5 \sin\left(\frac{2\pi}{T}t\right)$. We use three different teacher trajectories for weight learning, with periods $T = 10, 15, 20$ and corresponding context targets $\tilde{c} = 2, 2.5, 3$. The target of dynamical learning in Fig. 2a has $T = 12.5$.

Task (ii): $N = 1000, N_z = 1, N_c = 2, N_u = 0, p = 0.2, \tilde{w} = 1, t_{\text{stay}} = 500, t_{\text{fb}} = 100, t_{\text{wlearn}} = 50000, t_{\text{test}} = 1000$. The network learns to generate a superposition of sinusoidal oscillations with amplitudes $a$ and period $T$. The family of target trajectories is $\tilde{z}(t; a, T) = a \left( \sin\left(\frac{2\pi}{T}t\right) + \cos\left(\frac{4\pi}{T}t\right) \right)$. We use sixteen different teacher trajectories for weight learning, with four amplitudes $a$ distributed equidistantly between 3 and 7 and four periods $T$ distributed equidistantly between 10 and 20. The corresponding context targets are distributed equidistantly between 2 and 3 for both parameters. The target of dynamical learning in Fig. 2b has $a = 5$ and $T = 15$.

Task (iii): $N = 500, N_z = 3, N_c = 1, N_u = 0, p = 0.1, \tilde{w} = 1, t_{\text{stay}} = 200, t_{\text{fb}} = 100, t_{\text{wlearn}} = 50000, t_{\text{test}} = 1000$. The network learns to generate a constant output positioned on a curve in three dimensional space parameterized by $s$. The family of target trajectories (fixed points) is $\tilde{z}(t; s) = \left( \frac{s^2}{2} + s_{\text{off}}, 2(s - \frac{1}{2})^2 + s_{\text{off}}, \frac{s}{2} + s_{\text{off}} \right)$, where the offset $s_{\text{off}} = 2.5$ ensures that the network feedback is strong enough to enslave the network. We use ten different teacher trajectories for weight learning with param-
eters $s$ chosen between 0 and 1 such that the corresponding $\tilde{z}(t; s)$ lie equidistantly on the target curve $\{\tilde{z}(t; s) | s \in [0, 1]\}$. The corresponding context targets are distributed equidistantly between 2 and 3. The targets of dynamical learning in Fig. 2c have $s = 0.10$ and $s = 0.92$.

Task (iv): $N = 1000, N_z = 1, N_c = 1, N_u = 1, p = 0.2, \tilde{w} = 2, t_{\text{stay}} = 1000, t_{\text{wlearn}} = 30000, t_{\text{test}} = 500$. We choose $\tau_i$ from a uniform distribution between 0.3 and 2.5. During weight learning, we always provide error input $\varepsilon(t)$ to the network and do not fix $c(t)$, i.e. $t_{\text{fb}} = t_{\text{stay}} = 1000$. The network learns to predict the angle of a driven overdamped pendulum with mass $m$. The family of target dynamical systems is given by $\dot{\tilde{z}}(t) = F(\tilde{z}(t), u(t); m) = -m \sin(\tilde{z}(t)) + u(t) - \exp((\tilde{z}(t) - 0.65\pi)/0.65\pi) + \exp(-(\tilde{z}(t) - 0.65\pi)/0.65\pi)$. The last two terms provide a soft barrier preventing the pendulum from undergoing full rotations. During weight and dynamical learning, the pendulum is driven by low-pass filtered white noise $\dot{u}_{\text{wlearn}}(t) = -u_{\text{wlearn}}(t) + 0.2dW/dt$ (see Fig. S4b), which allows a comprehensive sampling of the pendulum’s dynamics. During testing the pendulum is driven by a triangular wave with unit amplitude and period $T = 50$. We use three different teacher dynamical systems for weight learning, with $m = 0.5, 1.0, 1.5$ and corresponding context targets $\tilde{c} = 0.7, 0.95, 1.2$. The targets of dynamical learning in Fig. 2d have $m = 0.8$ (continuous trace) and $m = 1.2$ (dashed trace).

Task (v): $N = 1000, N_z = 3, N_c = 1, N_u = 0, p = 0.1, \tilde{w} = 2, t_{\text{stay}} = 1000, t_{\text{fb}} = 100, t_{\text{wlearn}} = 50000, t_{\text{test}} = 10000$. The network learns a Lorenz system with dissipation parameter $\beta$. During weight learning, we always provide error input $\varepsilon(t)$ to the network, but fix $c(t)$ after $t_{\text{fb}}$. The family of target dynamical systems is given by $\dot{\tilde{z}}(t) = F(\tilde{z}(t); \beta) = F_{\text{Lorenz}}(C_{\text{Lorenz}}\tilde{z}(t); \beta)/(C_{\text{Lorenz}}\tau_{\text{Lorenz}})$, where $C_{\text{Lorenz}} = 40$ and $\tau_{\text{Lorenz}} = 20$ determine the spatial and temporal scale of the dynamics and $F_{\text{Lorenz}}(x(t); \beta) = (\sigma(x_2 - x_1), x_1(\rho - x_3) - x_2, x_1x_2 - \beta x_3)$ is the vector field of the standard Lorenz system, with $\sigma = 10$ and $\rho = 70$. We use four teacher dynamical systems for weight learning, with parameters $\beta$ distributed equidistantly between 2 and 6 and corresponding context targets distributed equidistantly between 2 and 3. The target of dynamical learning in Fig. 2e and f has $\beta = 4$.

II. QUANTIFICATION OF LEARNING PERFORMANCE

To quantify the performance of our model, we measure for each application the errors between signal outputs and targets during testing, for different network instances and targets. Except for task (v), we compute the testing error as the root-mean-square error between signal output and target during a period of length 50 in the middle of the testing phase. The measure is chosen to ignore phase shifts that occur over long testing times, as they are unavoidable in periodic autonomous dynamics (tasks (i,ii)), due to the accumulation of small errors in the period.

Task (i): Fig. S1a shows the testing error for the learning of sinusoidal oscillations. It is small for targets
Figure S1. Quality of dynamical learning of the sinusoidal oscillations in task (i). (a) Testing error between signal output and target and (b) period of the signal output, as a function of the period of the target. Vertical gray lines indicate the periods of the weight-learned targets and vertical orange lines indicate the period of the target used in Fig. 2a. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances.

with periods within and slightly beyond the range spanned and interspersed by weight-learned targets. Fig. S1b shows the good agreement between the periods of the output signals and the targets. We determine the periods from the maxima of the output signals’ power spectra, after discarding the initial interval of length 100 of the testing phase to allow for equilibration.

Task (ii): Fig. S2a shows the testing error for the learning of superpositions of sines. Again, the error is low within and slightly beyond the range of the parameters of the weight-learned targets. Similarly, the averaged local maxima of the signal outputs agree well with the averaged local maxima of their targets, Fig. S2b. The measurement of maxima starts at time 100 after the beginning of testing.

Task (iii): Fig. S3a shows the testing error for the learning of fixed points. It is low for target positions within and slightly beyond the range of the positions of the weight-learned targets. Fig. S3b shows signal outputs for different targets dynamically learned by a single network instance.

Task (iv). Fig. S4a shows the testing error for the learning of driven overdamped pendulums. It is small for pendulums with masses within and slightly beyond the range spanned and interspersed by weight-learned pendulums. Fig. S4b illustrates the dynamical learning and testing phases.

Task (v): Since the Lorenz system is chaotic for most of the parameter range that we consider, the signal
Figure S2. Quality of dynamical learning of the superpositions of sines in task (ii). (a) Median testing error between signal output and target as a function of the maximum and the period of the target function. Gray crosses indicate parameters of the weight-learned targets and the orange cross indicates the parameters used in Fig. 2b. (b) Averaged local maxima of the signal output as a function of the averaged local maxima of the target, for a target period of $T = 15$. Vertical gray lines indicate the maxima of the weight-learned targets and the vertical orange line indicates the maximum of the target used in Fig. 2b. Dots show median value and errorbars represent the interquartile range between first and third quartile. Results in (a) and (b) are obtained using 10 network instances for each parameter pair.

Output trajectory quickly deviates from the target system’s trajectory during testing. This holds also if the network approximates the target dynamical system well. Hence, instead of using the root-mean-square error, we compute the testing error as the discrepancy of the limit set $M_{\text{net}}$ generated by the network and the limit set $M_{\text{tar}}$ generated by the target dynamics. For the comparison, we use the Averaged Hausdorff Distance [3],

$$d_{\text{AHD}}(M_{\text{net}}, M_{\text{tar}}) = \max \left[ \frac{1}{|M_{\text{net}}|} \sum_{m_{\text{net}} \in M_{\text{net}}} d(m_{\text{net}}, M_{\text{tar}}), \frac{1}{|M_{\text{tar}}|} \sum_{m_{\text{tar}} \in M_{\text{tar}}} d(m_{\text{tar}}, M_{\text{net}}) \right],$$

$$d(m, M) = \min_{m' \in M} \| m - m' \|,$$

which is robust against outliers. Fig. S5a shows that the testing error is low within the range of parameters $\beta$ spanned and interspersed by weight-learned targets. In addition, we find that the relation between subsequent maxima of the z-coordinate of the signal output correctly forms the shape of a tent for most tested parameters (Fig. S5b). The behavior of our model also reproduces a bifurcation occurring for large
Figure S3. Quality of dynamical learning of the fixed points in task (iii). (a) Testing error between signal output and target as a function of the target position. Vertical gray lines indicate the positions of the weight-learned targets and vertical orange lines indicate the positions of the targets used in Fig. 2c. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances. (b) Single network instance learning the same set of dynamical learning targets as in (a). Blue spheres indicate the last signal outputs during testing after the different instances of dynamical learning. Yellow spheres indicate the position of the corresponding targets. They are mostly covered by blue spheres, except in the regions of larger error. The black tube shows the curve $\tilde{z}(t; s)$ on which the targets lie.

$\beta$: The target Lorenz system changes from chaotic behavior to fixed point behavior for the largest value of $\beta$ we consider. Our networks dynamically learn to generate the fixed point dynamics from this target, although they were only weight-trained in the chaotic regime. We note that some network instances, for example the one shown in Fig. S5b, generate fixed point behavior during testing, if the target has the second largest value of $\beta$ and is thus still chaotic. However, also in these cases the signal output converges to one of the two fixed points appearing for the largest $\beta$. This suggests that due to a shift in the averaged context parameter, the dynamical regime beyond the bifurcation is generated during testing.

III. INDUCTION OF UNSEEN SIGNAL OUTPUTS BY A CONTEXT-LIKE EXTERNAL INPUT

We test whether changing a context-like input $u_c(t)$ allows to generate sinusoidal oscillations with previously unseen frequencies. Like $c(t)$, $u_c(t)$ connects to the neurons in the network with a weight matrix $w_c$. 
Figure S4. Quality of dynamical learning of the overdamped pendulums in task (iv). (a) Error between signal output and target, as a function of the target pendulum’s mass. Vertical gray lines indicate the masses of the weight-learned targets and vertical orange lines indicate the masses of the targets used in Fig. 2d. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances. (b) Dynamical learning and testing. The network and the target receive the same low-pass filtered white noise as input drive during dynamical learning and triangular wave input during testing (lower subpanel). The network response (upper subpanel, blue trace) agrees well with the response of the target (upper subpanel, orange trace, nearly completely covered by the blue trace).

However, $u_c(t)$ is never generated by a network output, but a purely external input. There is no further context variable $c(t)$ and no error input $\varepsilon(t)$ in the network. Apart from this, the network is setup like in task (i). The output weights $w_z$ are learned using the FORCE rule, similar to weight learning in task (i): during each training period, we teach the network to generate a sinusoidal oscillation $\tilde{z}(t; T)$ with a period $T = 10, 15, 20$, in response to a constant $u_c(t) = 2, 2.5, 3$, analogous to teacher forcing with $\tilde{c}$. We find that the system can interpolate between the weight-trained output signals, if driven by previously unseen $u_c(t)$, cf. Fig. S6. See ref. [4] for a similar finding when morphing between conceptor weight matrices.

[1] B. DePasquale, C. J. Cueva, K. Rajan, G. S. Escola, and L. F. Abbott, PLoS One 13, e0191527 (2018).
[2] D. Sussillo and L. F. Abbott, Neuron 63, 544 (2009).
[3] O. Schuetze, X. Esquivel, A. Lara, and C. A. C. Coello, IEEE Trans. Evolutionary Computation 16, 504 (2012).
[4] H. Jaeger, arXiv:1403.3369 (2014).
Figure S5. Quality of dynamical learning of the Lorenz systems in task (v). (a) Testing error comparing the limit sets of signal output and target, as a function of the target’s parameter $\beta$. Vertical gray lines indicate the parameters of the weight-learned targets and the vertical orange line indicates the parameter of the target used in Fig. 2e,f. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances. (b) Tent maps of subsequent maxima in the z-coordinate for the signal output (dots, colored differently for different targets) and for the target dynamics (crosses, light coloring alike corresponding dots). The parameters $\beta$ of the targets are the same as in (a). Dynamical learning of all targets with a single network instance. Blue data correspond to the signal and target used in Fig. 2e,f; gray data indicate weight-learned targets. Tent maps of the target dynamics move from bottom left to top right for increasing $\beta$ except for the largest $\beta$ (brown, bottom left), for which the target dynamics converge to a fixed point. Insets show close-ups of results for the smallest (top left) and largest (bottom right) considered value of $\beta$. The signal output goes to a fixed point for the two largest, but also for the smallest considered value of $\beta$, leading to a focusing of the maxima relation to a small region.
Figure S6. Induction of unseen signal outputs by a context-like external input. The network has been trained similar to weight learning in task (i) to generate sinusoidal oscillations with three different frequencies in response to three constant external context inputs $u_c(t)$. After training, the weights are fixed and the network receives a continuously rising $u_c(t)$ (b). This results in a sinusoidal signal output with continuously rising period, which interpolates between the trained signals (a).