Learning neural dynamics through instructive signals

Rich Pang[1,2,*], Juncal Arbelaiz[1,3], and Jonathan W. Pillow[1,**]

[1]Princeton Neuroscience Institute, Princeton University, Princeton, NJ, USA.
[2]Lewis-Sigler Institute for Integrative Genomics, Princeton University, Princeton, NJ, USA.
[3]Center for Statistics and Machine Learning, Princeton University, Princeton, NJ, USA.
[*]Correspondence: rp21@princeton.edu
[**]Correspondence: pillow@princeton.edu

SUMMARY

Rapid learning is essential for robust behavior, allowing animals to quickly adapt to changing environments and task conditions. However, traditional Hebbian plasticity rules lack the power to induce the rapid, flexible changes in neural dynamics needed for such learning. Here we propose a network model with a rapid, heterosynaptic plasticity rule inspired by circuit motifs in hippocampus, cerebellum, and mushroom body, which relies on interactions between pre-synaptic activity and an "instructive signal" such as dopamine. We show that this rule can quickly produce highly flexible nonlinear dynamics using sparse, low-dimensional instructive signals while also admitting precise mathematical interpretation inspired by the well-known support vector machine. Using adaptive control theory, we additionally derive an online rule allowing such networks to learn using instructive signals guided by real-time feedback. Thus, heterosynaptic instruction-mediated plasticity is a powerful, biologically plausible and analytically tractable mechanism for the rapid learning of flexible dynamics, shedding light on the neural basis of complex adaptive behavior.

KEYWORDS

rapid learning, neural dynamics, plasticity, instructive signals, hippocampus, cerebellum, mushroom body, recurrent neural network, dopamine

Highlights

- A network model endowed with "HIP" (heterosynaptic instructed plasticity) rules, inspired by hippocampus, cerebellum, and mushroom body.
- Mathematical theory inspired by support vector machines exactly predicts the dynamics learned through HIP.
- Examples of learning nonlinear target dynamics in a single shot or from real-time feedback.
- Machine-learning application of instructive signals to solve long-range temporal credit assignment problem.

INTRODUCTION

Rapid learning is fundamental for robust behavior, allowing animals to quickly adapt to changing environments and task conditions, and is thought to be mediated by synaptic plasticity in the brain. The majority of studies modeling learning in the brain have focused on Hebbian plasticity

rules, in which synaptic weight changes are driven by coincident pre- and post-synaptic activity[1-9]. It has proved challenging, however, to use such rules to learn general-purpose neural computations; moreover, modern experiments suggest that some of the strongest, fastest plasticity rules *in vivo* have a markedly different character from traditional Hebbian rules. In a number of highly plastic neural circuits, for instance, the clearest weight changes are not Hebbian but depend on pre-synaptic activity coincident with an *instructive signal* from another brain area. In hippocampus (HPC), synaptic weights from area CA3 pyramidal cells (CA3PCs) to area CA1 pyramidal cells (CA1PCs) depend on an instructive signal from entorhinal cortex (EC)[10-12]; in cerebellum (CB) synapses from granule cells (GCs) onto Purkinje cells (PKJs) depend on an instructive signal carried by a "climbing fiber" (CF) from inferior olive[13-16]; and in the fly mushroom body (MB) synapses from Kenyon cells (KCs) onto mushroom body output neurons (MBONs) depend on a dopaminergic instructive signal (DAN) from the PAM/PPL1 neuropil[17-19] (Fig 1a-b); with specific temporal dependencies varying across systems. In each case, plasticity is largely independent of the post-synaptic or output neuron's activity. Moroever, these rules often act very quickly, producing large effects over single trials or seconds-long timecourses. Yet while the ability of such plasticity rules to learn bespoke tasks has begun to be investigated[20-24], how they shape brain function more generally via the flexible modulation of neural dynamics underlying behavior and computation[25-30] is not well understood.

Recurrent neural networks (RNN) are powerful, flexible models for studying brain dynamics and have yielded transformative insights into neural computation[2,5,25,26,30-34]. An RNN consists of a set of nodes or neurons whose activities evolve over time under the influence of external and recurrent/feedback inputs. By adjusting the recurrent synaptic weights, RNNs can be trained to perform diverse tasks and their internal activity subsequently examined to elucidate their underlying dynamics. Typically, RNNs are trained through either Hebbian rules or non-biological algorithms such as back-propagation through time (BPTT). While Hebbian rules can produce specific fixed-point attractors and sequential activity[2,4,5,8], it is still unclear how they could produce more flexible dynamics (e.g. limit cycles with specific frequencies, or line attractors), and they are likely too slow biologically for rapid learning, requiring many trials to produce a meaningful effect[3,35-38]. On the other hand, BPTT is highly flexible but not considered biologically plausible (although see[39,40]). BPTT also struggles with learning causal relationships between inputs and delayed outputs—the temporal credit assignment problem—when delays are long[41]. While emerging work suggests that RNNs endowed with HPC/CB/MB-like plasticity rules can learn interesting tasks[42,43], potentially bridging the gap between flexible rapid learning and biological mechanism, a general theory of how these rules shape neural dynamics in RNNs is lacking, limiting our understanding of their robustness and expressivity.

Here we investigate RNNs coupled to a plastic HPC/CB/MB-inspired feedback circuit subject to a HPC/CB/MB-inspired learning rule we term "heterosynaptic instructed plasticity" (HIP), in which synaptic weight changes depend exclusively on pre-synaptic activity and an instructive signal. Using mathematical theory and computational modeling, we find that the dynamics produced via HIP can be concisely described in a form precisely analogous to the theory of support vector machines (SVMs)—a widely studied and well-understood nonlinear machine-learning model[44]. In SVMs, inputs are nonlinearly projected into high dimensions—increasing their linear separability—then linearly combined to produce outputs; mathematical theory in turn shows how outputs can be alternatively computed by directly comparing test inputs with a subset of training inputs called "support vectors"[44-46]. Analogously, we show that nonlinear dynamics learned through HIP can be concisely described via "support states"—states of the system at the times the instructive signal was applied, and which determine the learned dynamics at all other states.

Here we show that neural dynamics produced through HIP are highly flexible, supporting the rapid learning of diverse nonlinear computations. Ultimately, this flexibility arises from an SVM-like high-dimensional nonlinear separation of system states within the feedback cir-

cuit, whose output creates feedback forces in the RNN dynamics. We find that a wide range of nonlinear dynamics can be quickly taught to the network via a sparse set of instructive signals. Leveraging tools from adaptive control theory, we then show how instructive signals can be used to teach the RNN to mimic the dynamics of an unknown system purely from real-time error signals. Finally, we show how optimizing the instructive signals themselves can solve certain long-range temporal credit assignment problems by avoiding the classic "vanishing gradient problem"[41]. This work thus reveals how highly flexible neural dynamics can emerge through rapid biological plasticity rules, establishes a concise mathematical link between RNN connectivity and dynamics[26,32,47], and suggests new directions for biologically inspired machine learning.

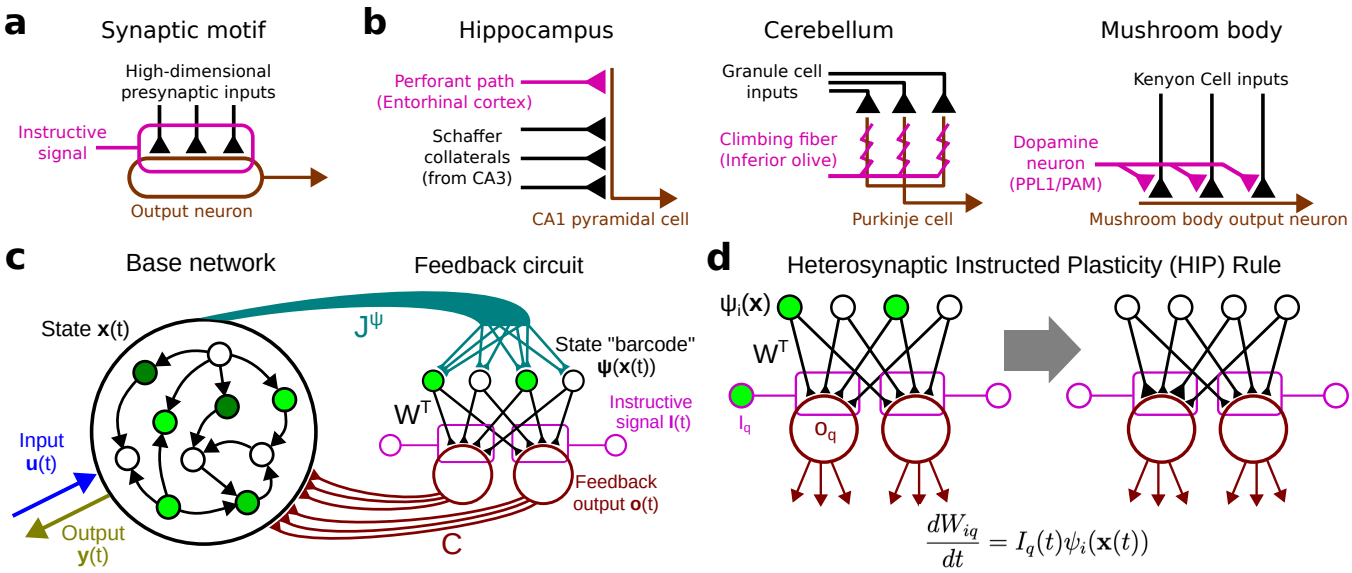


Figure 1: HIP rule and network model.
a. Stereotyped synaptic motif in which plasticity of pre-synaptic weights onto post-synaptic output neuron depends on pre-synaptic activity and an instructive signal. **b**. Schematics of physical instantiations of synaptic motif in **a** in three different brain areas. **c**. Network model including base network and plastic feedback circuit, with the latter shaped by instructive signals. **d**. Schematic of plasticity rule in Eq. (3).

RESULTS

We first introduce our network model and a mathematical theory for the rapid learning of neural dynamics. We then characterize several ways the network can be trained through instructive signals, revealing the wide-ranging capabilities of rapid biological plasticity rules observed in the brain.

Model

Architecture

Consider a "base" RNN reciprocally connected to a feedback circuit loosely analogous to a hippocampus (specifically the CA3-to-CA1 layer), cerebellum, or mushroom body (Fig 1c). Let there be D neurons in the base RNN, and in the feedback circuit N pre-synaptic or input neurons (CA3PCs/GCs/KCs) and Q post-synaptic or output neurons (CA1PCs/PKJs/MBONs). As in cerebellum and mushroom body, we will generally assume $N \gg Q$.

Dynamics

Let the dynamics of the D neurons in the base RNN follow:

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + \mathbf{f}_{base}(\mathbf{x}) + CW^T \boldsymbol{\psi}(\mathbf{x}) + B\mathbf{u}(t), \tag{1}$$

where $\mathbf{x}(t)$ are the activity levels of the base neurons, $\mathbf{u}(t)$ are external inputs, τ is a time constant, and $\boldsymbol{\psi}(\mathbf{x}) \in \mathbb{R}^N$ are the inputs to the feedback circuit (KC/GC/CA3PC activities). $W \in \mathbb{R}^{N \times Q}$ represents the internal weights in the feedback circuit, $C \in \mathbb{R}^{D \times Q}$ the weights from the feedback circuit output neurons back to the base RNN, and $B \in \mathbb{R}^{D \times D_{in}}$ a set of input weights. For completeness, we let the base RNN be parameterized by a separate weight matrix $J^{base} \in \mathbb{R}^{D \times D}$, such that $\mathbf{f}_{base}(\mathbf{x}) = J^{base}\phi(\mathbf{x})$, where ϕ is an element-wise nonlinearity. However, we will usually consider $J^{base} = 0$, i.e. no base dynamics besides the linear decay toward the origin. We optionally include outputs $\mathbf{y}(t)$ produced through a weight matrix $J^{out} \in \mathbb{R}^{D^{out} \times D}$.

The inputs $\boldsymbol{\psi}(\mathbf{x})$ to the feedback circuit are created via a random N-dimensional projection of $\mathbf{x}$ by matrix J^{ψ}, followed by a nonlinearity. Unless otherwise noted, the main example we will use throughout this work is the "Rand-Sig" $\boldsymbol{\psi}$:

$$\boldsymbol{\psi}(\mathbf{x}) = (1 + \tanh(J^{\psi}\mathbf{x}))\sqrt{2/N}, \tag{2}$$

where $J^{\psi}_{iq} \sim \mathcal{N}(0, g)$ and g is a gain hyperparameter controlling the weight standard deviation. The activities of the output neurons of the feedback circuit are given by $\mathbf{o}(t) = W^T \boldsymbol{\psi}(\mathbf{x}(t))$ and are implicit in (1). Intuitively, $\boldsymbol{\psi}(\mathbf{x})$ acts as a "barcode" for the system state $\mathbf{x}$—each $\mathbf{x}$ maps to a unique (with high probability) N-dimensional vector, with largely uncorrelated barcodes for distant states but similar barcodes for nearby states, in line with the well-studied pattern-separation action of biological random networks[48,49]. As we will show, the form of $\boldsymbol{\psi}$ influences the geometry of the flow fields (visualizations of the dynamical forces across system states) describing the learnable dynamics. For instance, adding a K-winner-take-all nonlinearity, $\boldsymbol{\psi}(\mathbf{x}) = \text{KWTA}[(1 + \tanh(J^{\psi}\mathbf{x}))\sqrt{2/N}]$, as proposed in MB and CA1 models[50,51], can be useful for decreasing interference and promoting pattern separation.

Heterosynaptic Instructed Plasticity (HIP) Rule

To focus on strong, rapid plasticity mediated by instructive signals, we take inspiration from HPC/CB/MB and let all plasticity be localized to the weights W within the feedback circuit (Fig 1c). Our goal is to understand whether and how plasticity of these weights alone can produce flexible, general-purpose nonlinear dynamics, which would confer substantial power and expressivity to a single layer of plastic feed-forward network weights. Specifically, let the weights evolve via:

$$\frac{dW^T}{dt} = \mathbf{I}(t)\boldsymbol{\psi}(\mathbf{x}(t))^T, \qquad \text{or component-wise,} \qquad \frac{dW_{iq}}{dt} = I_q(t)\psi_i(\mathbf{x}(t)) \tag{3}$$

where $\mathbf{I}(t) \in \mathbb{R}^Q$ is an external *instructive signal*, corresponding biologically to EC/CF/DAN signals. As in HPC/CB/MB plasticity, HIP is *heterosynaptic*—plasticity is induced by pre-synaptic and instructive signals but expressed at synapses from pre-synaptic to post-synaptic or output neurons (Fig 1d). Note also that HIP, while non-Hebbian, is still a "two-factor" rule (depending on [1] pre-synaptic activity and [2] an instructive signal), in contrast to "three-factor" rules studied elsewhere in the literature and sometimes associated with instructive or neuromodulatory signals (e.g. in which Hebbian plasticity is modulated by a dopaminergic reward signal)[52-54].

An illustration of the operation of the HIP rule is shown in Fig 1d. A synapse W_{iq} is modified when: (1) its presynaptic activity $\psi_i(\mathbf{x}(t))$ is greater than 0 (non-negativity is enforced by Eq

(2)); and (2) there is a simultaneous nonzero instructive signal $I_q(t)$. If the instructive signal is positive (negative) the weight increases (decreases). For simplicity, Eq. (3) relies on simultaneous activation of the presynaptic neuron and the instructive signal, but see STAR Methods for a theoretical treatment of temporally extended plasticity windows[10,11,18,55]. Note, however, that the simultaneous case provides crucial mathematical insights for addressing the temporally extended case. The functional effect of the learning rule can also be understood as follows. At time t^*, $\boldsymbol{\psi}(\mathbf{x}(t^*))$ is a barcode of system state $\mathbf{x}(t^*)$. If $I_q(t) = \alpha^*\delta(t - t^*)$, then at t^* column q of the weight matrix W is incremented by the current barcode weighted by α^*: $W_{:,q} \leftarrow W_{:,q} + \alpha^*\boldsymbol{\psi}(\mathbf{x}(t^*))$. Intuitively, $\boldsymbol{\psi}$ specifies "what" to learn, and $\mathbf{I}$ specifies "when" to learn it.

As we will show, different patterns of instructive signals $\mathbf{I}(t)$, along with external inputs $\mathbf{u}(t)$, can be used to train the network in different ways. In specific cases, however, such as learning from example target trajectories, it will be natural to allow $\mathbf{I}(t)$ to follow an error signal between the target and the RNN trajectories, thus providing a "closed-loop" learning rule. For clarity, however, we start by remaining agnostic as to the meaning or information content of $\mathbf{I}(t)$, and focus instead on its effects on connectivity and dynamics produced through HIP.

Theory of the learned flow field produced through instructive signals

Flow fields are visual representations of a dynamical system depicting the magnitude and direction of forces $d\mathbf{x}/dt$ at different states, and which can be used to understand the system geometrically[29,56]. How do the instructive signals $\mathbf{I}(t)$, which update the weights W through the HIP rule of Eq. (3), in turn update the flow field described by Eq. (1)? Linking RNN connectivity to dynamics and flow fields is a longstanding challenge in neuroscience[26,57-59], and at first glance it may not seem that considering only connectivity and dynamics produced through HIP would simplify this problem. Here, however, we show that the flow fields created through HIP can be expressed in a concise mathematical form precisely analogous to the "dual form" of support vector machines, revealing a simple geometric interpretation of the learned dynamics.

Mathematical form of the learned dynamics

For simplicity, let the initial weights in the feedback circuit be zero ($W = 0$), and suppose the system is driven by inputs $\mathbf{u}(t)$ and subject to instructive signals $\mathbf{I}(t)$. We first consider point-process instructive signals expressed as a train of weighted impulses or delta functions,

$$\mathbf{I}(t) = \sum_l \boldsymbol{\alpha}_l^* \delta(t - t_l^*), \tag{4}$$

where t_l^* is the time of the l-th instructive signal impulse, and $\boldsymbol{\alpha}_l^* \in \mathbb{R}^Q$ is a vector quantity giving the strength of each instructive signal component at the time of the impulse. We assume that all instructive signals are presented to the network during a bounded *instruction period*, after which $\mathbf{I}(t)$ is set to 0. Time-integrating the HIP rule (Eq. (3)) and substituting the resulting weight matrix W into Eq. (1) reveals (see STAR Methods for derivation) that the learned dynamics following the instruction period can be written as

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + \mathbf{f}_{base}(\mathbf{x}) + \sum_l C\boldsymbol{\alpha}_l^* k(\mathbf{x}, \mathbf{x}_l^*) + B\mathbf{u}(t), \tag{5}$$

where $k(\mathbf{x}, \mathbf{x}_l^*) \equiv \boldsymbol{\psi}(\mathbf{x})^T \boldsymbol{\psi}(\mathbf{x}_l^*)$ is a *kernel* function[44] specifying the effective similarity between state $\mathbf{x}$ and state $\mathbf{x}_l^*$. Note that the final dynamics depend only implicitly, and not explicitly, on the timecourse of $\mathbf{u}(t)$ during the instruction period, which we discuss shortly. Eq. (5) is directly analogous to the "dual form" of SVMs, in which test outputs or predictions are expressed not in

terms of weights w, but in terms of how similar or near a test input is to different *support vectors* $\{\mathbf{v}_l^*\}$ from the training data[44]. In analogy to SVMs, here the flow field $d\mathbf{x}/dt$ at an arbitrary state $\mathbf{x}$ can be expressed in terms of how similar or near $\mathbf{x}$ is to a set of *support states* $\{\mathbf{x}_l^*\}$. In fact, these support states are exactly the states of the system when the instruction signals occurred: $\mathbf{x}_l^* \equiv \mathbf{x}(t_l^*)$. We term Eq. (5) the *dual form* of the learned dynamics.

The effect of an instructive signal impulse adding a support state can be understood geometrically by visualizing the corresponding change in the flow field. If we assume $\mathbf{f}_{base} = 0$, then prior to the instruction period, the flow of the dynamics linearly draws all states back to the origin (Fig 2a). Now drive the system with some input $\mathbf{u}(t)$ during the instruction period, which will move the state around the state space. Applying a single instructive signal impulse at time t_1^* during this period adds a single support state $\mathbf{x}_1^* = \mathbf{x}(t_1^*)$ (Fig 2b). In accordance with Eq. (5) the resulting flow field becomes the sum of the base flow field $-\mathbf{x}/\tau$ and the patch of flow field, $C\boldsymbol{\alpha}_1^* k(\mathbf{x}, \mathbf{x}_1^*)/\tau$, contributed by $\mathbf{x}_1^*$. In this example, the support state creates a second fixed point (Fig 2c-d), revealing the capacity of HIP to produce nonlinear dynamics. Thus, instructive signals add support states through the HIP rule (Fig 2a-d), which concisely, yet precisely, describe the influence of the updated HPC/CB/MB-like feedback circuit (Fig 1c) on the system dynamics.

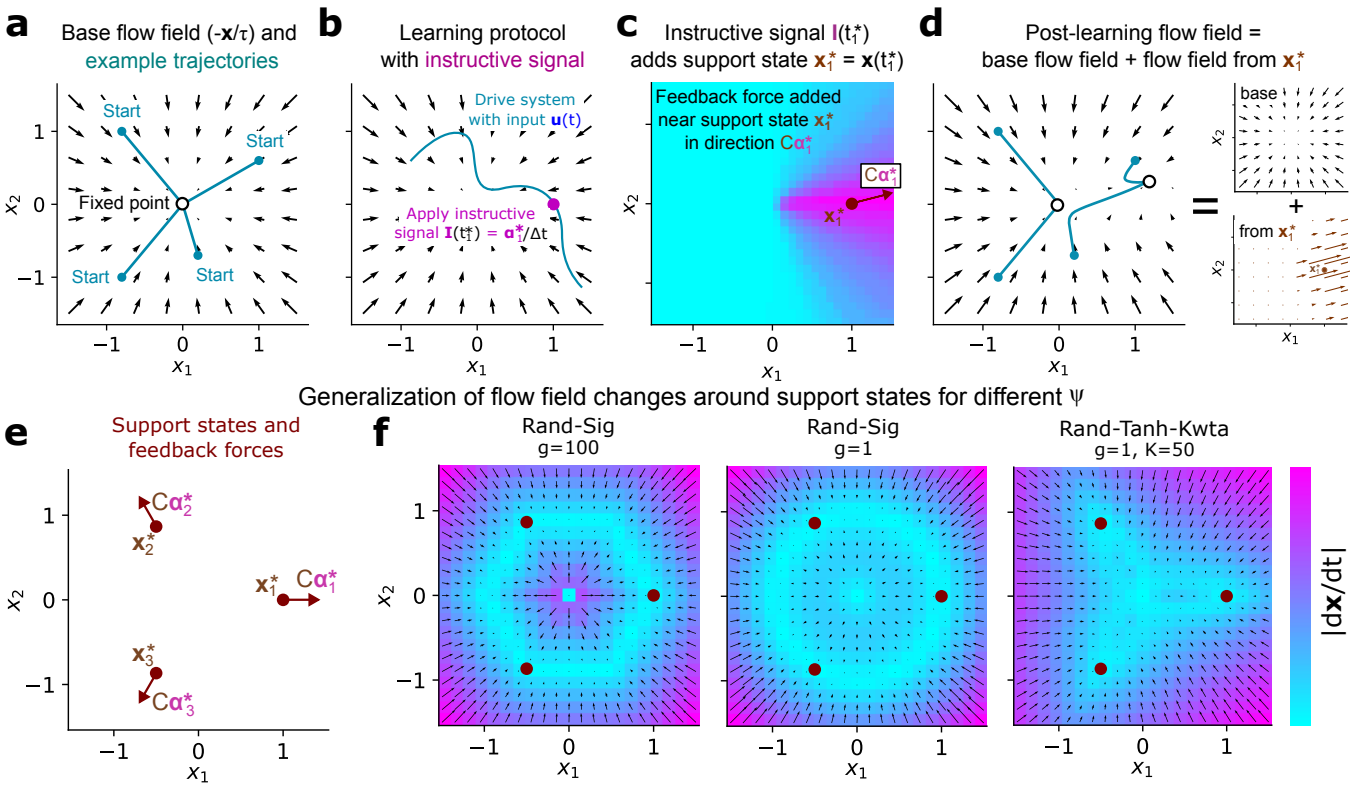


Figure 2: Flow fields produced by instructive signals.
a. Example base flow field in which activity decays through uncoupled linear dynamics to a stable fixed point at the origin. **b**. Schematic of applying a single instructive signal impulse while the system is being driven by external input $\mathbf{u}(t)$. **c**. Schematic of the effect of the instructive signal in **b**, which is to add a support state at $\mathbf{x}_1^* \equiv \mathbf{x}(t_1^*)$, whose influence on the learned dynamics spreads according to $k(\mathbf{x}, \mathbf{x}_1^*)$. **d**. Schematic of final flow field after instruction period, which is a sum of the base flow field and the flow field contributed by the support state $\mathbf{x}_1^*$. **e**. Schematic of three support states added to a 2-D state space with the same base dynamics as **a**. **f**. Flow field produced by the three support states in **e** for three different choices of ψ.

Note that to compute the final flow field it does not explicitly matter *when* the instructive signal was applied or what inputs were present—the only information retained explicitly in the

learned flow field equation is *where* the system was in state space at t_1^* and the (vector) value $\boldsymbol{\alpha}_1^*$ of the instructive signal. Multiple instructive signal impulses $\{(t_l^*, \boldsymbol{\alpha}_l^*)\}$ create multiple support states $\{\mathbf{x}_l^*\}$, each contributing its own patch of flow field $C\boldsymbol{\alpha}_l^* k(\mathbf{x}, \mathbf{x}_l^*)/\tau$, which is added to the sum in Eq. (5). (This statement holds for the HIP rule in Eq. (3), but see STAR Methods for the generalization to temporally extended plasticity windows.) For continuous, rather than point-process instructive signals, the sum in (5) is replaced by an integral and the *set* of support states by a support state *density* (STAR Methods).

The manner in which the learned dynamics generalize to states not visited during the instruction period, or in which no instructive signal was applied, is determined by the kernel $k(\cdot, \cdot)$, which defines the "similarity" between two states and in turn the effects of the support states. For instance, three support states optimized to produce three specific fixed points create different patterns of surrounding fixed and slow points when different $\boldsymbol{\psi}(\mathbf{x})$ are used, since these yield different kernels (Fig 2e-f). Thus, how the learned flow field generalizes across the total state space can be controlled or predicted in a precise, straightforward manner through the choice of the nonlinearity $\boldsymbol{\psi}$. Certain kernels, e.g. corresponding to the $N, g \to \infty$ limit of the Rand-Sig nonlinearity (Eq. (2)), additionally admit a closed form that does not depend on the specific instantiation of J^{ψ} (STAR Methods), such that the dual form depends on neither W or J^{ψ}. In this case, Eq. (5) reveals a concise, exact mathematical relationship between network connectivity of the complete RNN and the corresponding dynamics and flow fields. Next, we address which flow fields can be produced through instructive signals $\mathbf{I}(t)$, and how the latter should be chosen to learn specific target dynamics.

One-shot emulation of known target dynamics

Which dynamical systems can be expressed in terms of support states, and what patterns of instructive signals $\mathbf{I}(t)$ and external inputs $\mathbf{u}(t)$ can teach the RNN to produce these dynamics? We hypothesized that, unlike Hebbian plasticity—which produces dynamics largely limited to fixed-point attractors and sequences[2,5,8,60]—the flexibility of instructive signal timing combined with the basis-function-like nature of $\{k(\mathbf{x}, \mathbf{x}_l^*)\}$ would allow our model to produce a wide variety of nonlinear flow fields. To test this hypothesis, we considered the $N, g \to \infty$ limit of the Rand-Sig nonlinearity (Eq. (2)), whose associated kernel function can be expressed (STAR Methods) as

$$k(\mathbf{x}, \mathbf{x}^*) = 1 - \theta(\mathbf{x}, \mathbf{x}^*)/\pi, \tag{6}$$

where θ is the angle between $\mathbf{x}$ and $\mathbf{x}^*$. The corresponding dual form for the dynamics is:

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + \mathbf{f}_{base}(\mathbf{x}) + \sum_l C\boldsymbol{\alpha}_l^* \left(1 - \theta(\mathbf{x}, \mathbf{x}^*)/\pi\right) + B\mathbf{u}(t). \tag{7}$$

For the $N, g \to \infty$ Rand-Sig nonlinearity, the learned component of the flow field does not depend on the magnitudes of $\mathbf{x}$ and $\mathbf{x}^*$, but only on the angle θ between them. Hence, this learning rule and nonlinearity cannot produce changes to the flow field that vary with distance from the origin (note: the *total* flow field can still vary with distance to the origin due to the base dynamics). We verified this using Ridge Regression to identify the $\{\boldsymbol{\alpha}_l^*\}$, given support states $\{\mathbf{x}_l^*\}$, needed to reproduce target flow fields that did and did not satisfy this criterion (Fig S1). However, this restriction can be lifted by adding random biases in the nonlinearity $\boldsymbol{\psi}$ (Fig S1).

Even with this restriction, however, the flow fields that can be constructed in terms of support states with the $N, g \to \infty$ Rand-Sig nonlinearity are highly flexible. Each term in the sum in Eq. (7) acts like a basis function over the unit D-sphere, and these basis function span a large space of functions. For simplicity, consider the case where B and C are both the identity. Then, for

$D = 2, 3$, any function on the sphere that can be expressed as the sum of a constant and an odd function can be constructed from support states, with a similar result likely holding for $D > 3$ (STAR Methods). To illustrate this point, we used Ridge Regression to identify support states and associated $\{\boldsymbol{\alpha}_l^*\}$ that can produce a variety of flow fields, including multiple fixed points, limit cycles, and an approximate line attractor, corroborating our mathematical result (Fig S2). Thus, support states can be combined to produce a wide-ranging variety of nonlinear flow fields.

While the preceding Ridge Regression analyses show how flow fields can be decomposed in principle into support states, they do not show how these support states could be added online through HIP. We therefore asked whether such flow fields could be produced through appropriately patterned inputs $\mathbf{u}(t)$ and instructive signals $\mathbf{I}(t)$ via Eq. (3). If inputs and instructive signals exist that can add generic support states to the system, this would suggest that an animal could learn to "emulate" a target dynamical system in a brain area by generating a sufficient $\mathbf{u}(t)$ and $\mathbf{I}(t)$. On the other hand, one might imagine that certain combinations of support states might be unreachable through inputs and instructive signals.

We found, however, that for identity B and C, and leveraging the radial invariance of the $N, g \to \infty$ Rand-Sig kernel (Eq. (6)), we could construct a protocol for adding arbitrary support states through HIP. First, in an "offline" step (which we do not take to be biological), identify the support states $\{\mathbf{x}_l^*\}$ and $\{\boldsymbol{\alpha}_l^*\}$ needed to produce a target flow field (e.g. using Ridge Regression). Second, construct $\mathbf{u}(t)$ and $\mathbf{I}(t)$ as follows: for each desired $(\boldsymbol{\alpha}_l^*, \mathbf{x}_l^*)$, (1) drive the system with a large input $\mathbf{u}(t) \propto \mathbf{x}_l^*$, forcing $\mathbf{x}(t)$ into a state angularly close to the desired $\mathbf{x}_l^*$; (2) apply a delta-pulse instructive signal $\boldsymbol{\alpha}_l^*/\Delta t$ to trigger the HIP rule (Eq. (3)) that adds the support state, where Δt is the simulation timestep. Because support states at the same angular position are all equivalent (Eq. (7)), this provides a recipe for adding effectively arbitrary support states. We also assume $\mathbf{u}(t)$ can be arbitrarily large, in order to overcome the dynamics from the base RNN or the support states already added. Thus, any flow field expressible in terms of support states can emerge in principle through appropriately timed inputs and instructive signals (Fig 3a).

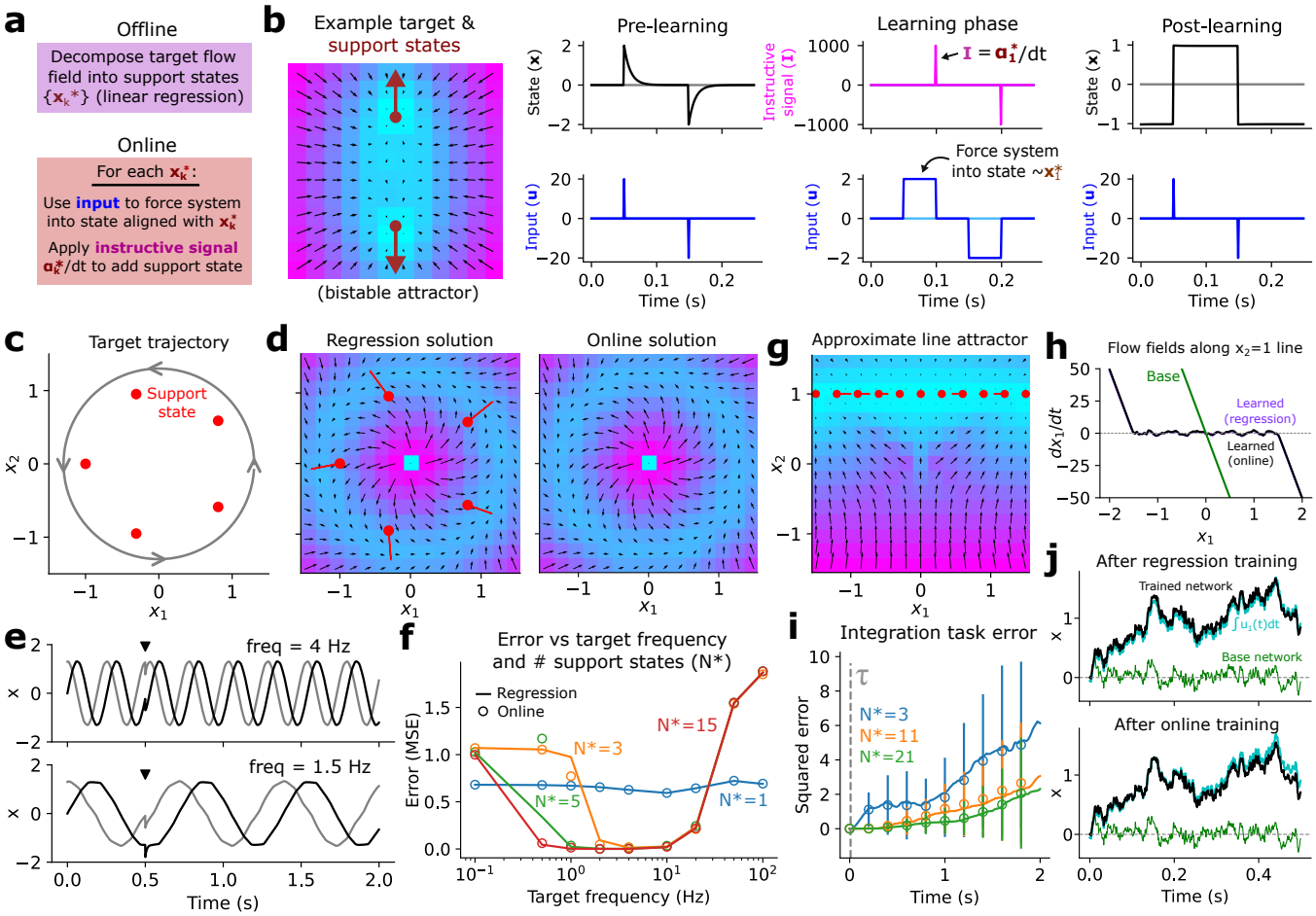


Figure 3: One-shot learning of known target dynamics
a. Offline-online algorithm for designing inputs $\mathbf{u}(t)$ and instructive signals $\mathbf{I}(t)$ to emulate a known target flow field. **b**. Example application of the algorithm in **a** to create a flow field with two fixed point attractors. **c**. Schematic of flows/velocities to be learned along a specific trajectory by combining 5 support states. **d**. Left: regression solution (identifying $\boldsymbol{\alpha}_1^*, \ldots, \boldsymbol{\alpha}_5^*$) to approximate target flow in **c**. Right: solution produced by applying the online portion of the algorithm. **e**. Activity time-courses for $\mathbf{x}(t)$ when the system is trained to produce either of two example limit cycles. The black triangle shows the time of a small perturbation, indicating that the dynamics are stable with respect to this perturbation. **f**. Error between target and learned activity time-courses $\mathbf{x}(t)$ as a function of the target limit cycle frequency and the number of support states. **g**. Flow field and support states reproducing an approximate line attractor. **h**. Dynamics along the $x_2 = 1$ line in **g** of the system trained to produce a line attractor that is used to integrate $u_1(t)$. The system is driven with an additional constant input $u_2(t) = 1$ during this task. **i**. Error in using the trained system to perform integration along the line attractor, vs elapsed time and the number of support states afforded. **j**. Example of the trained network integrating its inputs.

We tested this procedure in three examples. Fig 3b, for instance, shows inputs and instructive signals required to add two stable fixed points to a network with no base dynamics (except linear decay to the origin), subsequently allowing the network to act as a 1-bit memory. During the offline step, we can also identify support states needed to produce only flows/velocities at specific states, rather than the entire flow field—for instance the flows along a circular trajectory (Fig 3c). We were curious, however, whether the learned dynamics around such a trajectory would be stable or unstable to perturbations, even if we taught the RNN to only reproduce velocities on the trajectory itself. We first tested this by using the above procedure to teach the

RNN emulate the flow velocities along a circular trajectory with a target frequency ν (Fig 3c). As expected, we found that the trajectory velocities could be learned via inputs and instructive signals (Fig 3d). We also found, however, that the resulting dynamics were indeed stable around the target trajectory, producing a limit cycle at the target frequency (Fig 3e). The range of frequencies of stable limit cycles increased with the number of support states or instructive signal impulses (Fig 3f). Thus, presenting inputs and instructive signals to produce a circular trajectory automatically stabilized the dynamics into a limit cycle across a range of frequencies.

As a third example, we asked whether this protocol could produce a line attractor—a canonical model of a neural dynamical motif supporting integration[61]. Indeed, our protocol could produce an approximate line attractor capable of integrating inputs over a timescale much longer than τ (Fig 3g-j). Note, however, that for a finite number of support states the line attractor is not perfect. First, it is localized to a specific section of phase space, hence cannot integrate beyond its bounds (Fig 3h); second, it is composed of a microscopically rough landscape of slow and fixed points (Fig 3h). As a result, the inputs to be integrated need to strong enough to drive the system around this landscape, but not so large as to escape the bounds of the attractor (Fig S3). Similar to the limit cycle example (Fig 3c-f), the stability of the line attractor against perturbations orthogonal to its longitudinal axis emerges automatically, even though we did not explicitly enforce this. Intuitively, this stability as well as that of the limit cycle (Fig 3e) arises from competition with the base flow field $-\mathbf{x}/\tau$.

Thus, a variety of flow fields can be produced given proper inputs $\mathbf{u}(t)$ and instructive signals $\mathbf{I}(t)$. Moreover, this rule is highly efficient, requiring only sparse instructive signals—one impulse per support state to be added— yet which can substantially change the qualitative behavior of the RNN. Such a system is therefore well-posed for rapid or one-shot learning to emulate flexible dynamical systems, and may be highly resource-efficient if instructive signals are costly.

Online learning of unknown dynamics through real-time feedback

We next asked whether our RNN could learn effectively through HIP without having access to the target dynamics *a priori*. To address this question we considered an online learning scheme in which a separate, reference system exists whose full dynamics are unknown but whose state $\mathbf{z}(t)$ can be observed throughout an instruction period. We asked whether an instructive signal could be derived from the reference state that would teach the activity dynamics $\mathbf{x}(t)$ of a learner RNN governed by Eq. (1)–(3) to autonomously approximate those of the reference system.

Leveraging tools from adaptive control theory (STAR Methods), we found that this was indeed the case. Suppose the true reference system follows dynamics

$$\tau \frac{d\mathbf{z}}{dt} = \mathbf{f}_{ref}(\mathbf{z}) + B\mathbf{u}, \tag{8}$$

for some arbitrary function $\mathbf{f}_{ref}$ unknown to the learner RNN. Let the learner RNN follow the dynamics

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + \mathbf{f}_{base}(\mathbf{x}) + CW^\top \boldsymbol{\psi}(\mathbf{x}) + B\mathbf{u} + \mathbf{c}, \tag{9}$$

which is equivalent to Eq. (1), except modified to explicitly include an additional input $\mathbf{c}(t)$, which we call a "corrective signal". Specifically, $\mathbf{c} \equiv \kappa \mathbf{e}$, where $\mathbf{e}(t) \equiv \mathbf{z}(t) - \mathbf{x}(t)$ is the tracking error between the reference and learner system state and $\kappa > 0$ is referred to as a control gain. Further, let the weight updates and instructive signal follow

$$\frac{dW^\top}{dt} = \eta \mathbf{I}(t) \boldsymbol{\psi}(\mathbf{x}(t))^\top, \text{ where } \mathbf{I}(t) \equiv C^\top \mathbf{e}(t), \tag{10}$$

where $\eta > 0$ is the learning rate. Note that the instructive signals here are continuous, rather than a point process. A schematic of the complete architecture is shown in Fig 4a.

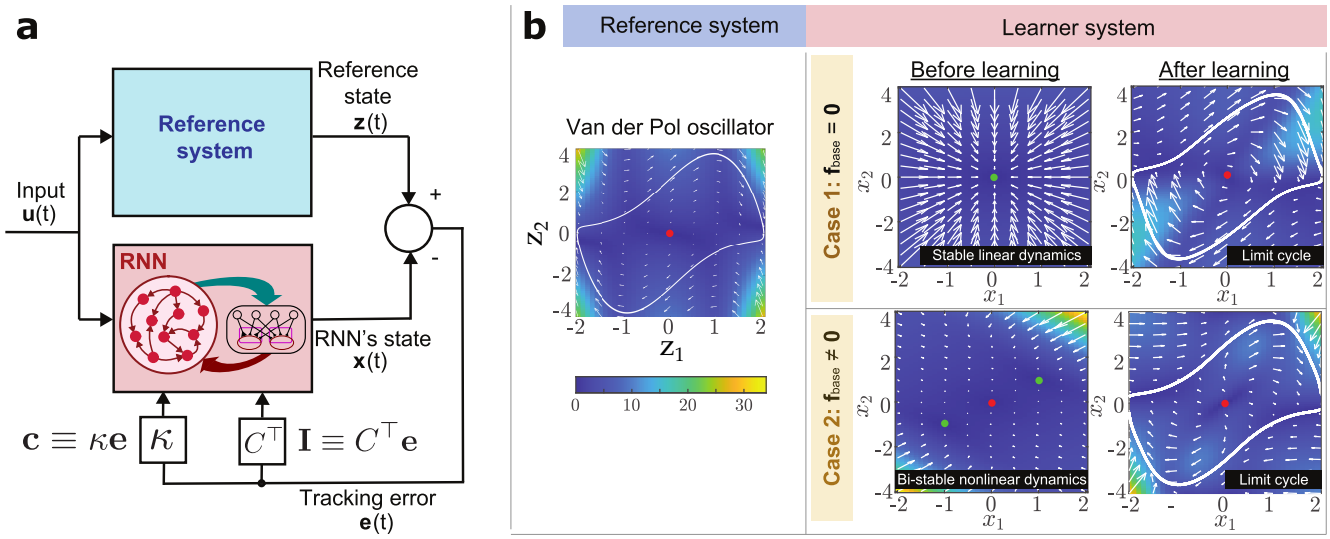


Figure 4: Online learning to emulate an unknown system
a. Architecture of learner RNN guided by feedback from a reference system—a more detailed block diagram is provided in the Fig S4. The reference system and learner RNN both receive the same input $B\mathbf{u}$. The difference between the reference's and RNN's states defines the tracking error $\mathbf{e}$, which is used to generate the *corrective signal* $\mathbf{c} = \kappa\mathbf{e}$ and the *instructive signal* $\mathbf{I} = C^\top\mathbf{e}$ that are fed back into the RNN. **b.** Numerical case studies. The reference system is a Van der Pol oscillator, which has a stable limit cycle and an unstable fixed point at the origin. Case study 1 has $\mathbf{f}_{base} \equiv \mathbf{0}$ and exhibits the dynamics of a strictly stable linear system (due to the leakage term) before learning. In case 2, the base RNN dynamics support a bi-stable attractor. In both cases, the RNN successfully changes its local flow into stable limit cycle dynamics after the instruction period. Stable (unstable) fixed points are indicated by green (red) dots. Color-code of the flow fields is preserved across panels and indicated by the colorbar.

Our goal is for the learner RNN to learn to eventually reproduce the reference dynamics $\mathbf{f}_{ref}$ without requiring access to $\mathbf{z}$ or any further instruction. Thus, after the instruction period, we set η and κ to zero, ceasing the evolution of the weights and letting the RNN behave independently of the reference system. The learning framework presented in this section is grounded on the theory of adaptive control[62,63], specifically Model Reference Adaptive Control (MRAC). Under certain model matching assumptions and appropriate selection of the value of the gain κ, convergence of the RNN's state to that of the reference system can be asymptotically guaranteed using Barbalat's lemma in Lyapunov-like analysis[64] or through contraction theory[65-67]—useful methods for the analysis of non-autonomous nonlinear dynamical systems. Selection of the inputs $\mathbf{u}$ to guarantee "persistency of excitation" conditions (intuitively, guaranteeing sufficient exploration of the state space) ensures learning of appropriate weight values[68,69]. Note that while the learner observes the reference system state, the reference receives no information from the learner. Such additional information exchange could make learning active and hasten learning rates; however, for simplicity we do not include it here, and it is not necessary for learning.

To validate the ability of the instructive signals (Eq. (10)) to teach the RNN to emulate the reference system, we numerically explored two illustrative case studies. In both cases, the reference dynamics to be emulated were described by a Van der Pol oscillator (Fig 4b, left), a 2-D nonlinear system that generates relaxation oscillations (STAR Methods). First, we analyzed the setting in which $\mathbf{f}_{base} \equiv \mathbf{0}$ (i.e. linear uncoupled base dynamics decaying to the origin), while

in the second case we consider pre-existing base dynamics supporting a bi-stable attractor, reflecting a scenario in which an existing learned flow field must be overwritten in order to emulate the limit cycle of the reference Van der Pol oscillator. We found that, regardless of the base dynamics considered, the RNN successfully learned a Van-der-Pol-like limit cycle resembling the reference system after the instruction period (Fig 4b, right). Thus, the corrective and instructive signals were sufficient to teach the network to emulate the reference dynamics through real-time feedback.

As here the instructive signals are continuous, the dual form of the learned dynamics after the instruction period requires a slight modification from Eq. (5). As continuous signals represent the limit of a point process, we found that the resulting dual form of the dynamics could be represented not in terms of individual support states, but via a support state *density* (Fig S7; STAR Methods)—in close analogy to the relationship between point charges and a charge density in electrostatics[70,71]. Analogous to the point-process/support-state case (Fig 2), the learned change in the flow field corresponds to an integral of the product of k with the support state density, revealing the high flexibility of dynamics and flow fields that can be produced in this manner. Thus, through HIP, instructive signals support both the online, one-shot implementation of specific known target dynamics (Fig 3), as well as the ability to learn online to emulate a reference system through real-time feedback (Fig 4). In both cases, the set of flow fields that can be produced by the learning rule is extremely flexible, due to the basis-function-like nature of the kernel function—the central limitation is the set of states that can be accessed during the instruction period, which determines where support states can be added.

We note the related idea of online internal model adaptation through error feedback studied in motor control. For example, it has been proposed that the cerebellum contains internal inverse and forward models of the motor apparatus[20]. The inverse models provide the neural commands necessary to achieve a desired motor trajectory or state and are interpreted as feedforward controllers. Forward internal models predict the behavior of the motor system in response to motor commands and can be leveraged, for instance, to compensate for time delays in feedback control[72]. Further, these models can in principle be learned within the cerebellum via different training signals: inverse models through motor command errors[20,73,74] and forward models through sensory prediction errors[20]. Thus, our results on instructive signals teaching the RNN to emulate reference dynamics are consistent with existing ideas in cerebellar motor control, while providing additional theoretical support and generality.

Optimizing the instructive signals for long-range credit assignment

Finally, we asked whether instructive signals could be leveraged to train RNNs on tasks involving long-range temporal credit assignment that challenge traditional methods. Canonical RNN training in machine learning, typically based on back-propagation through time (BPTT), learns the RNN weights directly through gradient descent to minimize a loss function[75,76]. However, while BPTT is both flexible and admits biological approximations[39,40], training in the face of delayed loss evaluation is sensitive to vanishing (or exploding) gradients—these arise when the eigenvalues of the Jacobian matrix are less than (vanishing) or greater than (exploding) 1, due to the repeated multiplication of the Jacobian across timesteps when calculating the loss gradient[41,77]. This problem hinders temporal credit assignment between outputs that are separated from their causal inputs by long delays.

Here we give a proof of concept demonstrating how instructive signals can be used to address this problem by avoiding vanishing/exploding gradients. In our proposed method, which we term “instructor training”, we do not optimize the weights directly but instead optimize the instructive signals that produce the final weights W through the HIP rule (Eq. (3)). That is, given a

fixed input sequence $\mathbf{u}(t)$, each candidate instructive signal timecourse $\mathbf{I}(t)$ produces a trajectory of the weight matrix $W(t)$ according to HIP, and which ultimately yields a final, fixed (after the instruction period) W. We then optimize $\mathbf{I}(t)$ through gradient descent, using the same inputs and delayed loss function as in BPTT. Our final goal is to be able to run the initial RNN forward (Eq. (1)–(3)) while applying the optimized instructive signals, such that the final weights W at the end of the instruction period allow the RNN to solve the task autonomously. Identifying task-optimized instructive signals could also provide a means to predict instructive signal dynamics in animals throughout task learning[78]. For simplicity, here we consider the discretized version of our RNN (STAR methods), in which $\mathbf{x}(t)$, $\mathbf{I}(t)$ are replaced by $\mathbf{x}^t, \mathbf{I}^t$, etc..

Notably, optimizing the instructive signal timecourse $\mathbf{I}^t$ can converge to at least two possible outcomes. First, the optimal instructive signals could remain strong throughout the course of learning, corresponding to a case in which the network learns to rely on the instructive signals to solve the task, thus never learning to perform the task independently. Alternatively, and more favorably, the optimal instructive signals could decrease over learning, i.e. with the network initially relying on the instructive signals but by the end of the instruction period only relying on the weights W—in this case the network learns to solve the task independently without further instruction. We hypothesized that it was possible for the second case to emerge automatically simply by applying standard gradient descent to the instructive signal timecourse, even without imposing an explicit decay of their magnitude.

To focus on long-range temporal credit assignment, we tested our hypothesis on a proof-of-concept task with loss evaluated at a variable delay. That is, the loss depends only on the network activity occurring after the input has ceased (Fig 5a). To illustrate the essential challenge of temporal credit assignment, we considered a parsimonious "hold" task with a delayed loss evaluation (Fig 5b). Here, in response to an input pulse in the $[-1, 0]$ direction, the network should respond by holding the first neuron's activity x_1 at -1, and vice versa for an input pulse in the $[+1, 0]$ direction. Loss evaluation (the error between the true x_1 and the target), however, only occurs after a variable delay following each input. As shown in Fig 5c-d, both BPTT[75,76] and e-prop (a biological version of BPTT)[40] succeed for short delays (on the order of the network time constant τ) but fail for longer delays, as expected from the vanishing gradient problem[41].

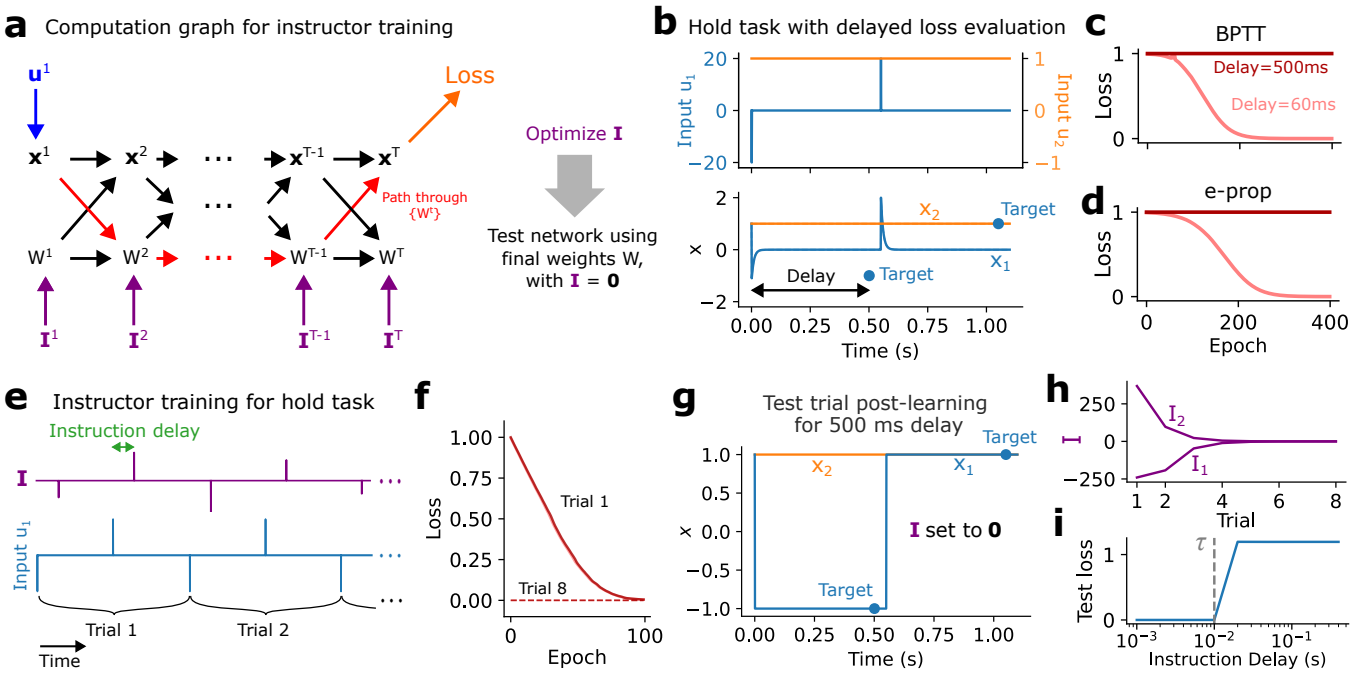


Figure 5: Instructor training proof-of-concept

a. Computation graph for instructor training. **b**. Example task in which loss is evaluated only

after a delay, expected to exhibit the vanishing gradient problem the delay is too long. The goal is for x_1 (blue) to pass through each of the targets. **c, d**. Loss vs training epoch for the hold task in **b** for BPTT and e-prop for two different delays. **e**. Schematic of instructor training paradigm used for the hold task. Here the magnitudes of the instructive signal impulses are optimized **f**. Loss vs epoch for each trial in the instructor training paradigm of **e**, given a 500 ms delay. **g**. Network performing hold task after instructor training, and with the instructive signal silenced. **h**. Magnitude of each of the two instructive signals vs trial number. **i**. Test loss (corresponding to the test trial in **g** as a function of the instruction delay.

We found, however, that instructor training could solve this task, even with long delays, and moreover without learning to rely on the instructive signal by the end of training. Specifically, we considered 8 sequential "trials" of the hold task, each containing two inputs, with one instructive signal impulse allowed after each input (Fig 5e). We then optimized the amplitudes of the instructive signals for each trial, using the resulting weights W at the end of each trial as the initial weights for the subsequent trial. Indeed, for a long delay (500 ms, which neither BPTT nor e-prop can handle) the loss decreased over both training epochs within a trial and over trials. (Note also that grouping epochs into trial groups in BPTT or e-prop does not help them solve this task.) Importantly, in a test trial in which no instructive signals were included, the network still solved the task (Fig 5g), indicating that it had learned to rely on the weights rather than the instructive signals, even though it used the latter during the instruction period—indeed, the magnitude of the optimized instructive signals decreased over trials (Fig 5h). Finally, we found that in order for the network to learn to solve the task without relying on the instructive signals it was necessary for the instructive signals to occur shortly after the inputs (Fig 5i). This suggests that instructive signals should occur near the time of inputs, in order to store this information in the weights so that it can be accessed later during learning.

Instructor training succeeds for the following reason. Formally, loss gradients are computed by summing over all paths through the computation graph from the network parameters to the loss (Fig 5a, S5), with each path corresponding to a product of Jacobians along that path[79]. In BPTT, however, the paths linking inputs to loss evaluation through the network states $\{\mathbf{x}^t\}$ often "vanish" or "explode" if there is a long delay between the input and loss evaluation[41,77], due to the repeated multiplication of Jacobians in the gradient calculation and the decay (or explosion) of the corresponding eigenvalues, impairing learning. In instructor training, however, the weights are themselves dynamic variables, opening additional paths through the computation graph—i.e., through $\{W^t\}$—for information from past inputs to reach the loss. Because the plasticity rule (Eq. (3)) has no decay time constant (corresponding to an eigenvalue of 1), this allows information to persist in the weights over long time periods, which in turn allows gradient information to propagate over long periods through the computation graph without vanishing or exploding. Intuitively, the optimization process learns that an instructive signal can be invoked to store information about the inputs in W until it is needed again; at the same time, however, it learns over sequential trials to transfer this information storage ability into the dynamics produced via W.

Thus, instructor training represents an alternative to canonical gradient methods for end-to-end training of RNNs. In both BPTT and instructor training, the final result is an RNN with fixed weights W. Instructor training, however, leverages instructive signals to learn long-range dependencies during the training process, even though at test time no instructive signals are required. Note that we do not treat the process of optimizing the instructive signals as biologically plausible; however, the optimal instructive signals reflect a biological solution to updating the weights if those instructive signals could be generated in the brain. These results thus suggest a biologically inspired solution to a long-standing problem in machine learning.

DISCUSSION

We have shown how biologically inspired plasticity rules mediated by pre-synaptic activity and instructive signals can rapidly produce highly flexible nonlinear network dynamics. This work represents a complement to the theory of Hebbian plasticity's ability to produce fixed point attractors or sequences that can be used for pattern completion[2,4,5,8]. Instructive signals, when applied to a plastic feedback circuit coupled to a base RNN, add support states to the dynamics flow field that can be combined to yield arbitrary nonlinear dynamics. A notable feature of the learning system we have proposed is that addition in the space of synaptic weights (via Eq. (3)) corresponds directly to addition in the flow field (Eq. (5))—thus, HIP reveals a simple, interpretable link between synaptic weights and flow fields, addressing the long-standing challenge of linking RNN connectivity and dynamics[26,32,47,57]. As plasticity mediated by instructive signals also accords more strongly with experimental data on rapid learning than Hebbian plasticity[10,11,18,42,55], our model may additionally be more applicable to explaining empirical neural dynamics.

Our work extends a growing body of work on training RNNs with biological learning rules. One avenue of past research, encompassing the "RFLO"[39], "e-prop"[40], and MDGL[80] algorithms, focused on how loss gradients can be approximated by local cellular processes in a single RNN. A second avenue of research has focused on dynamics and learning in RNNs coupled to a plastic feedback circuit analogous to a cerebellum[43] or mushroom body[42], or more generically an adaptive feedback controller[63,81], similar to our model architecture (Fig 1c). However, while such RNNs can solve interesting tasks in a more biologically plausible manner than traditional training algorithms like BPTT, the mathematical relationship between the learning rule, the networks weights, and the resulting flow fields has remained elusive. Although some theoretical work has established important links between network connectivity and flow fields[26,82], these studies have largely not considered connectivity produced by rapid biological plasticity rules. Our work synthesizes the above by providing a unifying framework and concise mathematical theory that links network connectivity produced through rapid biological learning rule to interpretable nonlinear dynamics and flow fields. Our study also suggests an RNN algorithm—instructor training—that can in certain cases solve long-range temporal credit assignment problems by overcoming the vanishing/exploding gradient problem, which hinders learning via canonical algorithms[41] unless additional slow variables are introduced that can be used at test time[40].

Our work represents a simplified model of how biological plasticity rules shape neural dynamics to support rapid learning and remains subject to important limitations. We have largely considered a plasticity rule in which weight changes are triggered by precise coincidence between presynaptic activity and the instructive signal, whereas empirical plasticity rules in hippocampus, cerebellum, and mushroom body exhibit various extended temporal windows of coincidence detection[10,11,18,55]. The dual form of the learned dynamics (Eq. (5)) can be generalized to accommodate such temporally extended windows (Fig S6, STAR Methods), but in future work it will be important to establish in detail how such modifications influence learning. Additionally, in all of our examples we have considered the case in which the (columns of the) input and feedback connections, B and C, respectively, span the entire space $\mathbb{R}^D$ of the base dynamics. In general, however, the columns of B and C might span only subspaces of $\mathbb{R}^D$, and it will be important to characterize which dynamics and tasks can be learned under these constraints. Finally, it will be useful to explore how different base dynamics $\mathbf{f}_{base}$ impair or accelerate learning, and to characterize in more detail the tasks that can be solved via instructor training.

Our theory makes testable general predictions, while also opening an avenue for fitting to data to make predictions in specific experimental contexts. The first general prediction is that activating an instructive signal will alter the neural dynamics in other brain areas predominantly around the system state these dynamics were in at the time of the instructive signal—the support

state. This could be tested, for instance, by analyzing neural activity in cortical areas while stimulating climbing fibers inputs to cerebellum or perforant pathway inputs to hippocampus. Second, many different sequences of instructive signals can ultimately produce the same or similar flow fields—thus, for animals that all learn a task successfully, we predict there to be substantial heterogeneity in the instructive signals evoked throughout learning, even if the learned neural dynamics are similar across animals[83]. Finally, our model can in principle be fit to empirical neural population activity trajectories by identifying support states that would produce dynamics consistent with those trajectories. Such an approach could be used to make predictions about unseen neural responses in regions of state space not visited by the training data, benefiting from the inductive bias encoded by the kernel function.

RESOURCE AVAILABILITY

Lead contact

Requests for further information and resources should be directed to and will be fulfilled by the lead contact, Rich Pang (rp21@princeton.edu).

Data and code availability

- This project generated no new datasets.
- All code used to generate the figures in this manuscript is available at github.com/rkp8000/dope-rnn
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

This work was funded by The Princeton Neuroscience Institute Innovation Award (to RP and JWP), The Princeton Neuroscience Institute C.V. Starr Award (to JA), the NIH BRAIN initiative (R01EB026946; to JWP), the NIMH (R01EY033064; to JWP), a U19 NIH-NINDS BRAIN Initiative Award (5U19NS104648; to JWP), and the Simons Collaboration on the Global Brain (SCGB AWD543027; to JWP). We thank Hakan Tureci for insightful discusions during the development of this project, as well as Anushri Arora, Victor Geadah, and other members of the Pillow lab for many helpful thoughts and conversations.

AUTHOR CONTRIBUTIONS

Conceptualization, R.P., J.A., and J.P.; methodology, R.P., J.A., and J.P.; investigation, R.P. and J.A.; writing-–original draft, R.P. and J.A.; writing-–review & editing, R.P., J.A., and J.P.; funding acquisition, R.P., J.A., and J.P.; resources, R.P., J.A., and J.P.; supervision, J.P.

DECLARATION OF INTERESTS

The authors declare no competing interests.

SUPPLEMENTAL INFORMATION INDEX

Figures S1-S7 and their legends in a PDF

References

1. Hebb, D.O. (1949). The organization of behavior: A neuropsychological theory. Wiley.

2. Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences *79*, 2554–2558.

3. Bi, G.q., and Poo, M.m. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. Journal of Neuroscience *18*, 10464–10472.

4. Fiete, I.R., Senn, W., Wang, C.Z., and Hahnloser, R.H. (2010). Spike-time-dependent plasticity and heterosynaptic competition organize networks to produce long scale-free sequences of neural activity. Neuron *65*, 563–576.

5. Brunel, N. (2016). Is cortical connectivity optimized for storing information? Nature neuroscience *19*, 749–755.

6. Whittington, J.C., Muller, T.H., Mark, S., Chen, G., Barry, C., Burgess, N., and Behrens, T.E. (2020). The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. Cell *183*, 1249–1263.

7. Eckmann, S., Young, E.J., and Gjorgjieva, J. (2024). Synapse-type-specific competitive hebbian learning forms functional recurrent networks. Proceedings of the National Academy of Sciences *121*, e2305326121.

8. Farrell, M., and Pehlevan, C. (2024). Recall tempo of hebbian sequences depends on the interplay of hebbian kernel with tutor signal timing. Proceedings of the National Academy of Sciences *121*, e2309876121.

9. Confavreux, B., Harrington, Z.P., Kania, M., Ramesh, P., Krouglova, A.N., Bozelos, P.A., Macke, J.H., Saxe, A.M., Gonçalves, P.J., and Vogels, T.P. (2025). Memory by a thousand rules: Automated discovery of multi-type plasticity rules reveals variety & degeneracy at the heart of learning. bioRxiv pp. 2025–05.

10. Bittner, K.C., Milstein, A.D., Grienberger, C., Romani, S., and Magee, J.C. (2017). Behavioral time scale synaptic plasticity underlies ca1 place fields. Science *357*, 1033–1036.

11. Milstein, A.D., Li, Y., Bittner, K.C., Grienberger, C., Soltesz, I., Magee, J.C., and Romani, S. (2021). Bidirectional synaptic plasticity rapidly modifies hippocampal representations. eLife *10*, e73046.

12. Grienberger, C., and Magee, J.C. (2022). Entorhinal cortex directs learning-related changes in ca1 representations. Nature *611*, 554–562.

13. Marr, D., and Thach, W.T. (1991). A theory of cerebellar cortex. From the retina to the neocortex: selected papers of David Marr pp. 11–50.

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

15. Medina, J.F., and Lisberger, S.G. (2008). Links from complex spikes to local plasticity and motor learning in the cerebellum of awake-behaving monkeys. Nature neuroscience *11*, 1185–1192.

16. Silva, N.T., Ramírez-Buriticá, J., Pritchett, D.L., and Carey, M.R. (2024). Climbing fibers provide essential instructive signals for associative learning. Nature Neuroscience *27*, 940–951.

17. Cohn, R., Morantte, I., and Ruta, V. (2015). Coordinated and compartmentalized neuromodulation shapes sensory processing in drosophila. Cell *163*, 1742–1755.

18. Handler, A., Graham, T.G., Cohn, R., Morantte, I., Siliciano, A.F., Zeng, J., Li, Y., and Ruta, V. (2019). Distinct dopamine receptor pathways underlie the temporal sensitivity of associative learning. Cell *178*, 60–75.

19. Zolin, A., Cohn, R., Pang, R., Siliciano, A.F., Fairhall, A.L., and Ruta, V. (2021). Context-dependent representations of movement in drosophila dopaminergic reinforcement pathways. Nature Neuroscience *24*, 1555–1566.

20. Wolpert, D.M., Miall, R.C., and Kawato, M. (1998). Internal models in the cerebellum. Trends in cognitive sciences *2*, 338–347.

21. Hattori, D., Aso, Y., Swartz, K.J., Rubin, G.M., Abbott, L., and Axel, R. (2017). Representations of novelty and familiarity in a mushroom body compartment. Cell *169*, 956–969.

22. Bennett, J.E., Philippides, A., and Nowotny, T. (2021). Learning with reinforcement prediction errors in a model of the drosophila mushroom body. Nature Communications *12*, 2569.

23. Brunel, N., Hakim, V., and Nadal, J.P. (2025). From the perceptron to the cerebellum. arXiv preprint arXiv:2505.14355.

24. Wu, Y., and Maass, W. (2025). A simple model for behavioral time scale synaptic plasticity (btsp) provides content addressable memory with binary synapses and one-shot learning. Nature communications *16*, 342.

25. Mante, V., Sussillo, D., Shenoy, K.V., and Newsome, W.T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. nature *503*, 78–84.

26. Mastrogiuseppe, F., and Ostojic, S. (2018). Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. Neuron *99*, 609–623.

27. Vyas, S., Golub, M.D., Sussillo, D., and Shenoy, K.V. (2020). Computation through neural population dynamics. Annual review of neuroscience *43*, 249–275.

28. Duncker, L., and Sahani, M. (2021). Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings. Current opinion in neurobiology *70*, 163–170.

29. Luo, T.Z., Kim, T.D., Gupta, D., Bondy, A.G., Kopec, C.D., Elliot, V.A., DePasquale, B., and Brody, C.D. (2023). Transitions in dynamical regime and neural mode underlie perceptual decision-making. bioRxiv pp. 2023–10.

30. Driscoll, L.N., Shenoy, K., and Sussillo, D. (2024). Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. Nature Neuroscience *27*, 1349–1363.

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

32. Barak, O. (2017). Recurrent neural networks as versatile tools of neuroscience research. Current opinion in neurobiology *46*, 1–6.

33. Remington, E.D., Narain, D., Hosseini, E.A., and Jazayeri, M. (2018). Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. Neuron *98*, 1005–1019.

34. Yang, G.R., Joglekar, M.R., Song, H.F., Newsome, W.T., and Wang, X.J. (2019). Task representations in neural networks trained to perform many cognitive tasks. Nature neuroscience *22*, 297–306.

35. Feldman, D.E. (2012). The spike-timing dependence of plasticity. Neuron *75*, 556–571.

36. Pang, R., and Fairhall, A.L. (2019). Fast and flexible sequence induction in spiking neural networks via rapid excitability changes. eLife *8*, e44324.

37. Magee, J.C., and Grienberger, C. (2020). Synaptic plasticity forms and functions. Annual Review of Neuroscience *43*, 95–117.

38. Pang, R., and Recanatesi, S. (2025). A non-hebbian code for episodic memory. Science Advances *11*, eado4112.

39. Murray, J.M. (2019). Local online learning in recurrent networks with random feedback. Elife *8*, e43299.

40. Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. Nature communications *11*, 3625.

41. Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In International conference on machine learning. Pmlr pp. 1310–1318.

42. Jiang, L., and Litwin-Kumar, A. (2021). Models of heterogeneous dopamine signaling in an insect learning and memory center. PloS computational biology *17*, e1009205.

43. Pemberton, J., Chadderton, P., and Costa, R.P. (2024). Cerebellar-driven cortical dynamics can enable task acquisition, switching and consolidation. Nature Communications *15*, 10913.

44. Bishop, C.M., and Nasrabadi, N.M. (2006). Pattern recognition and machine learning vol. 4. Springer.

45. Boser, B.E., Guyon, I.M., and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory. pp. 144–152.

46. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. IEEE Intelligent Systems and their applications *13*, 18–28.

47. Schuessler, F., Mastrogiuseppe, F., Ostojic, S., and Barak, O. (2024). Aligned and oblique dynamics in recurrent neural networks. eLife *13*, RP93060.

48. Leutgeb, J.K., Leutgeb, S., Moser, M.B., and Moser, E.I. (2007). Pattern separation in the dentate gyrus and ca3 of the hippocampus. science *315*, 961–966.

49. Chettih, S.N., Mackevicius, E.L., Hale, S., and Aronov, D. (2024). Barcoding of episodic memories in the hippocampus of a food-caching bird. Cell *187*, 1922–1935.

50. Dasgupta, S., Stevens, C.F., and Navlakha, S. (2017). A neural algorithm for a fundamental computing problem. Science *358*, 793–796.

51. Plitt, M.H., and Giocomo, L.M. (2021). Experience-dependent contextual codes in the hippocampus. Nature Neuroscience *24*, 705–714.

52. Farries, M.A., and Fairhall, A.L. (2007). Reinforcement learning with modulated spike timing–dependent synaptic plasticity. Journal of neurophysiology *98*, 3648–3665.

53. Frémaux, N., and Gerstner, W. (2016). Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. Frontiers in neural circuits *9*, 85.

54. Kuśmierz, Ł., Isomura, T., and Toyoizumi, T. (2017). Learning with three factors: modulating hebbian plasticity with errors. Current opinion in neurobiology *46*, 170–177.

55. Suvrathan, A., Payne, H.L., and Raymond, J.L. (2016). Timing rules for synaptic plasticity matched to behavioral function. Neuron *92*, 959–967.

56. Strogatz, S.H.S.H. (2018). Nonlinear Dynamics and Chaos: with applications to Physics, Biology, Chemistry, and Engineering. . ISBN 9780813349107.

57. Sompolinsky, H., Crisanti, A., and Sommers, H.J. (1988). Chaos in random neural networks. Physical review letters *61*, 259.

58. Pezon, L., Schmutz, V., and Gerstner, W. (2024). Linking neural manifolds to circuit structure in recurrent networks. bioRxiv pp. 2024–02.

59. Claudi, F., Chandra, S., and Fiete, I.R. (2025). A theory and recipe to construct general and biologically plausible integrating continuous attractor neural networks. bioRxiv pp. 2025–05.

60. Amit, D.J., and Amit, D.J. (1989). Modeling brain function: The world of attractor neural networks. Cambridge university press.

61. Seung, H.S. (1996). How the brain keeps the eyes still. Proceedings of the National Academy of Sciences *93*, 13339–13344.

62. Astrom, K.J., and Wittenmark, B. (2008). Adaptive Control (2nd ed.). Second ed.. Dover Books on Engineering.

63. Alemi, A., Machens, C., Deneve, S., and Slotine, J.J. (2018). Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules. In Proceedings of the AAAI conference on artificial intelligence vol. 32.

64. Slotine, J.J.E., and Li, W. (1991). Applied Nonlinear Control. Prentice Hall. ISBN 0130408905. doi: 10.1088/1741-2552/8/6/060201.

65. Lohmiller, W., and Slotine, J.J.E. (1998). On contraction analysis for non-linear systems. Automatica *34*, 683–696. doi: 10.1016/S0005-1098(98)00019-3.

66. Bullo, F. (2022). Contraction theory for dynamical systems. . ISBN 9798836646806.

67. Tsukamoto, H., Chung, S.J., and Slotine, J.J.E. (2021). Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview. Annual Reviews in Control *52*, 135–169. doi: 10.1016/j.arcontrol.2021.10.001. arXiv:2110.00675.

68. Narendra, K.S., and Annaswamy, A.M. (1987). Persistent excitation in adaptive systems. International Journal of Control *45*, 127–160. URL: https://doi.org/10.1080/00207178708933715. doi: 10.1080/00207178708933715.

69. Panteley, E., Loría, A., and Teel, A. (2001). Relaxed persistency of excitation for uniform asymptotic stability. IEEE Transactions on Automatic Control *46*, 1874–1886. doi: 10.1109/9.975471.

70. Green, G. (1889). An essay on the application of mathematical analysis to the theories of electricity and magnetism vol. 3. Mayer & Müller.

71. Landau, L.D., Bell, J.S., Kearsley, M., Pitaevskii, L., Lifshitz, E., and Sykes, J. (2013). Electrodynamics of continuous media vol. 8. elsevier.

72. Wolpert, D.M., and Miall, R.C. (1996). Forward Models for Physiological Motor Control. Neural Networks *9*, 1265–1279.

73. Kawato, M., Furukawa, K., and Suzuki, R. (1987). A hierarchical neural-network model for control and learning of voluntary movement. Biological Cybernetics *57*, 169–185. doi: 10.1007/BF00364149.

74. Kawato, M., and Gomi, H. (1992). The cerebellum and VOR/OKR learning models. Trends in Neurosciences *15*, 445–453.

75. Werbos, P.J. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE *78*, 1550–1560.

76. Lillicrap, T.P., and Santoro, A. (2019). Backpropagation through time and the brain. Current opinion in neurobiology *55*, 82–89.

77. Engelken, R. (2023). Gradient flossing: Improving gradient descent through dynamic control of jacobians. Advances in Neural Information Processing Systems *36*, 10412–10439.

78. Pan-Vazquez, A., Araujo, Y.S., McMannon, B., Louka, M., Bandi, A., Haetzel, L., Faulkner, M., Pillow, J.W., Daw, N.D., and Witten, I.B. (2024). Pre-existing visual responses in a projection-defined dopamine population explain individual learning trajectories. Current Biology *34*, 5349–5358.

79. Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning representations by back-propagating errors. nature *323*, 533–536.

80. Liu, Y.H., Smith, S., Mihalas, S., Shea-Brown, E., and Sümbül, U. (2021). Cell-type–specific neuromodulation guides synaptic credit assignment in a spiking neural network. Proceedings of the National Academy of Sciences *118*, e2111821118.

81. Kaleb, K., Feulner, B., Gallego, J., and Clopath, C. (2024). Feedback control guides credit assignment in recurrent neural networks. Advances in Neural Information Processing Systems *37*, 5122–5144.

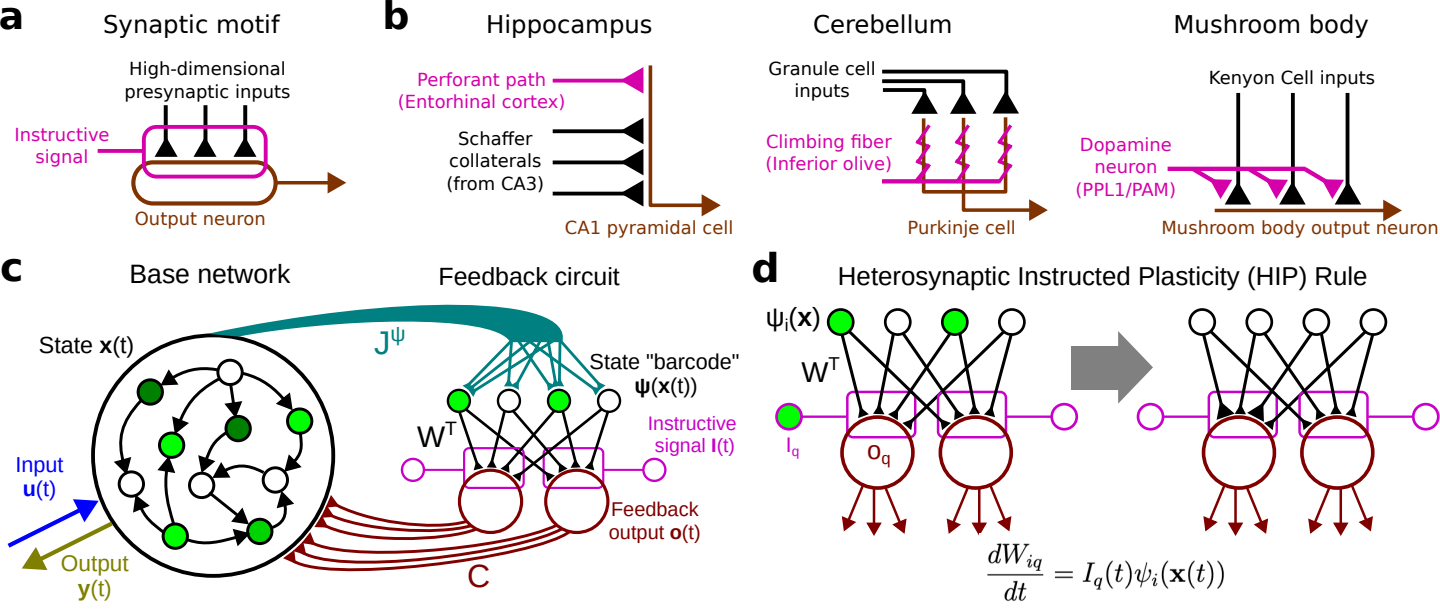
82. Eliasmith, C., and Anderson, C.H. (2003). Neural engineering: Computation, representation, and dynamics in neurobiological systems. MIT press.

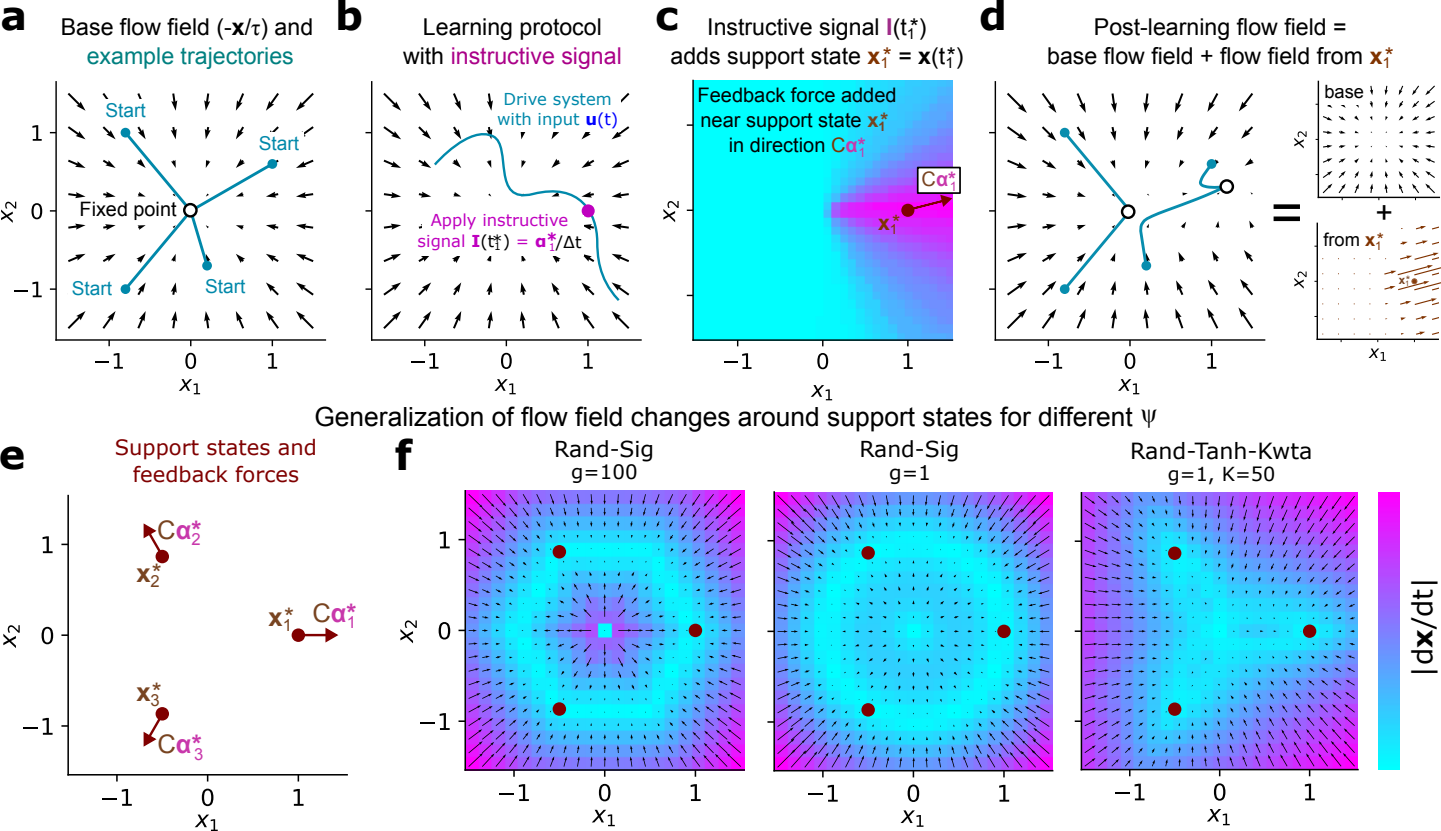
83. Nieh, E.H., Schottdorf, M., Freeman, N.W., Low, R.J., Lewallen, S., Koay, S.A., Pinto, L., Gauthier, J.L., Brody, C.D., and Tank, D.W. (2021). Geometry of abstract learned knowledge in the hippocampus. Nature *595*, 80–84.

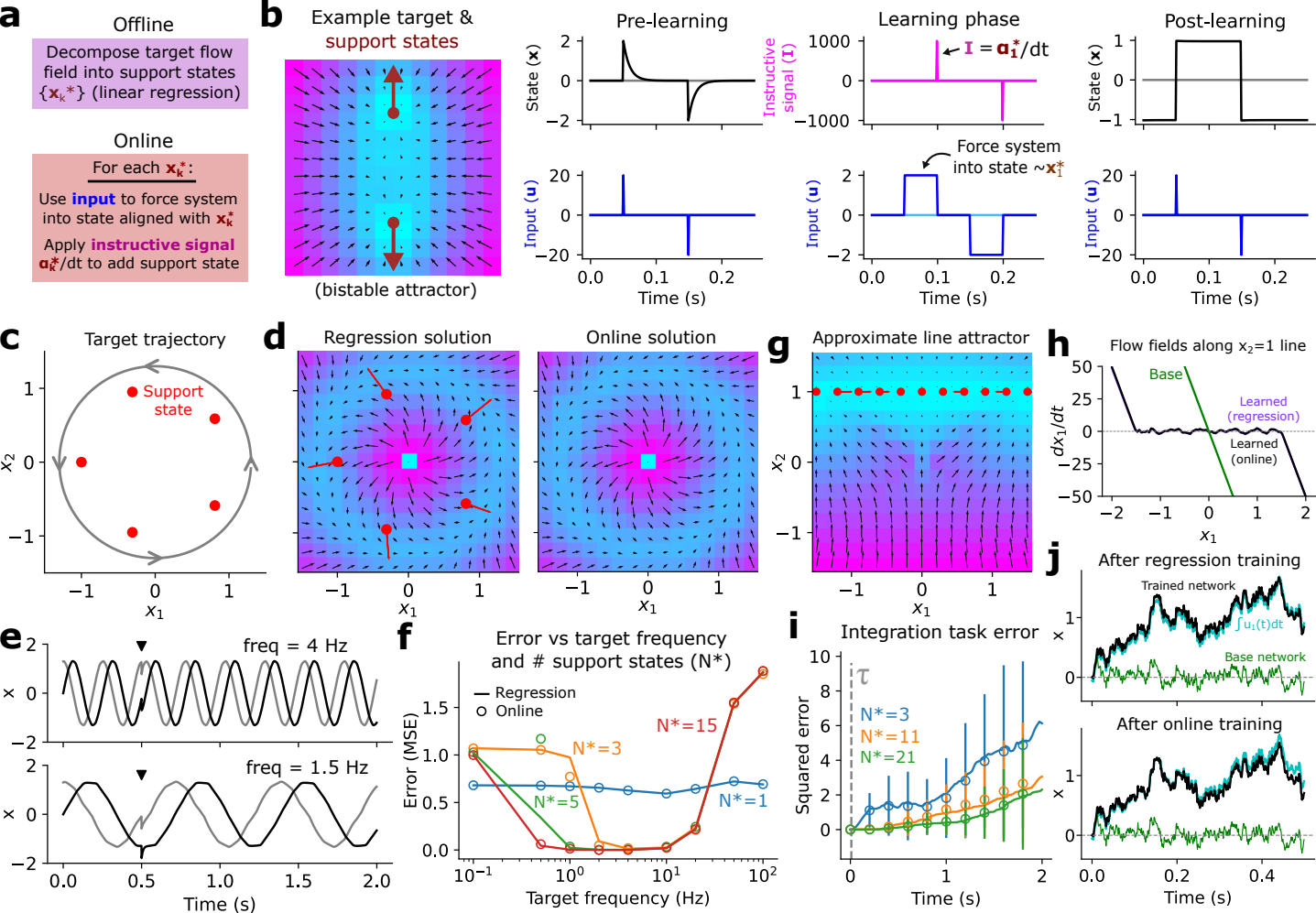
84. Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. Advances in neural information processing systems *31*.

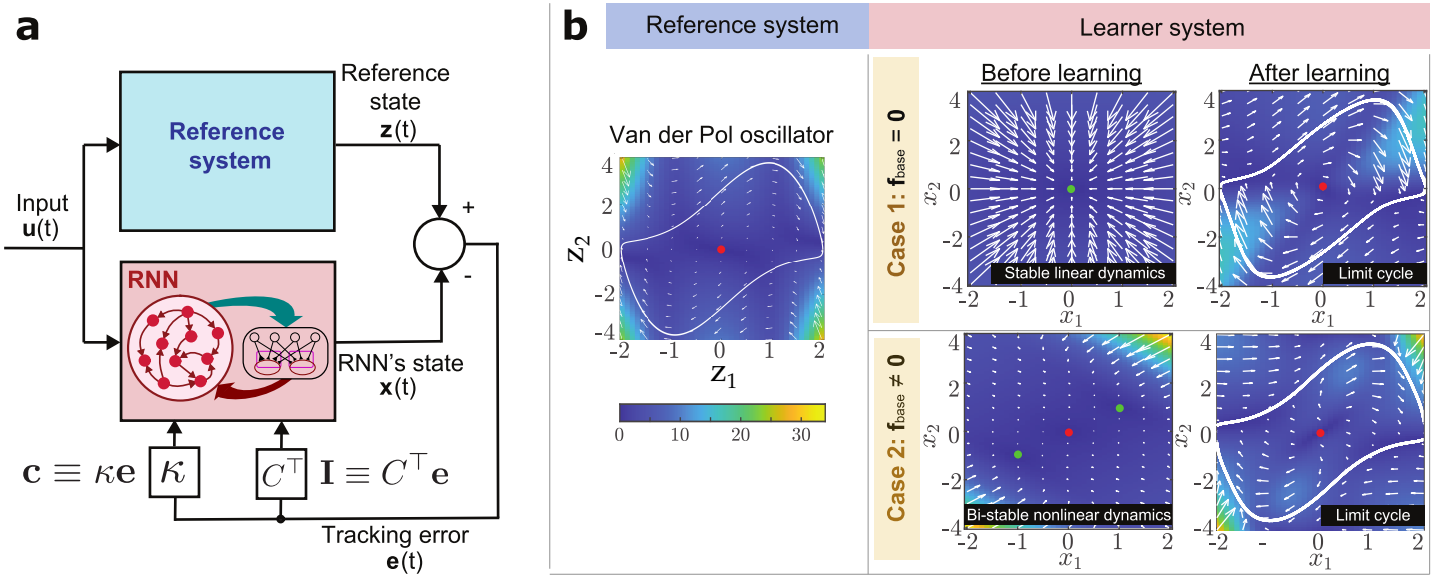
85. Ramamoorthi, R., and Hanrahan, P. (2001). A signal-processing framework for inverse rendering. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 117–128.

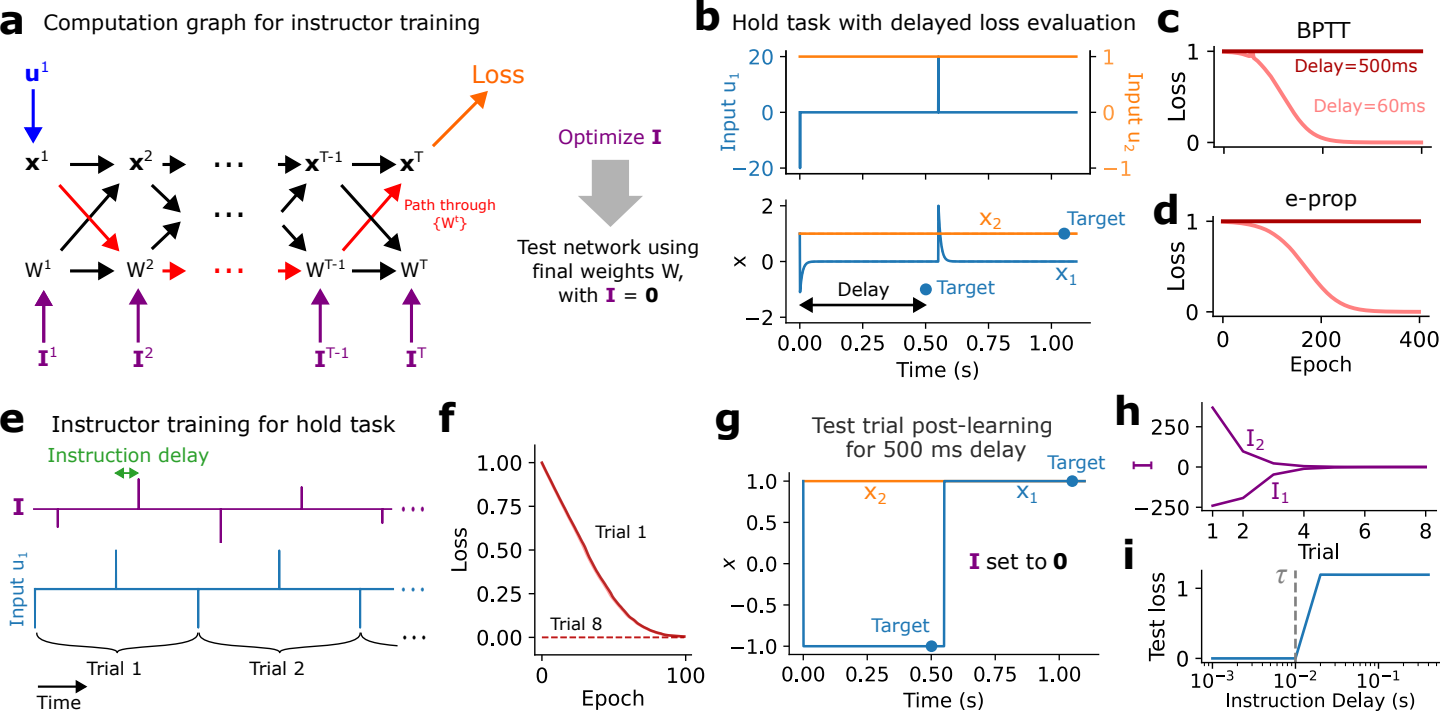
86. Cohen, T.S., Geiger, M., Köhler, J., and Welling, M. (2018). Spherical cnns. arXiv preprint arXiv:1801.10130.

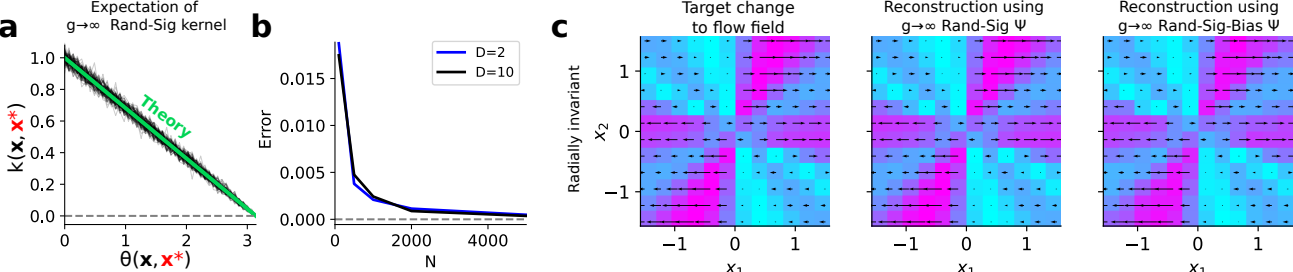

a
Synaptic motif
High-dimensional presynaptic inputs
Instructive signal
Output neuron
b
Hippocampus
Perforant path (Entorhinal cortex)
Schaffer collaterals (from CA3)
CA1 pyramidal cell
Cerebellum
Granule cell inputs
Climbing fiber (Inferior olive)
Purkinje cell
Mushroom body
Kenyon Cell inputs
Dopamine neuron (PPL1/PAM)
Mushroom body output neuron
c
Base network
Feedback circuit
State x(t)
J^ψ
State "barcode" ψ(x(t))
W^T
Instructive signal I(t)
Input u(t)
Feedback output o(t)
Output y(t)
C
d
Heterosynaptic Instructed Plasticity (HIP) Rule
ψi(x)
W^T
Iq
Oq
dWiq/dt = Iq(t)ψi(x(t))

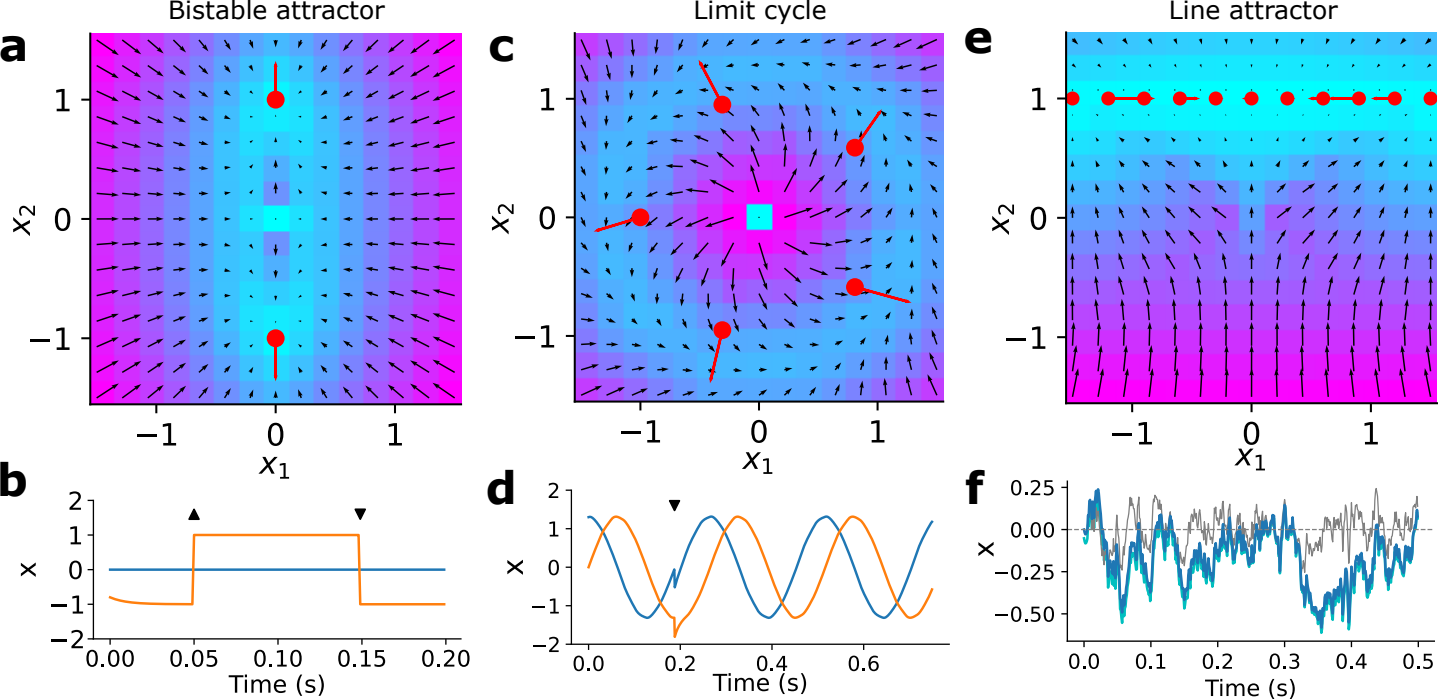

a
Base flow field (-x/τ) and
example trajectories
Start
Start
Fixed point
Start
Start
b
Learning protocol
with instructive signal
Drive system
with input u(t)
Apply instructive
signal I(t*1) = α*1/Δt
c
Instructive signal I(t*1)
adds support state x*1 = x(t*1)
Feedback force added
near support state x*1
in direction Cα*1
Cα*1
x*1
d
Post-learning flow field =
base flow field + flow field from x*1
base
from x*1
x*1
Generalization of flow field changes around support states for different ψ
e
Support states and
feedback forces
Cα*2
x*2
x*1
Cα*1
x*3
Cα*3
f
Rand-Sig
g=100
Rand-Sig
g=1
Rand-Tanh-Kwta
g=1, K=50
|dx/dt|
x1
x2

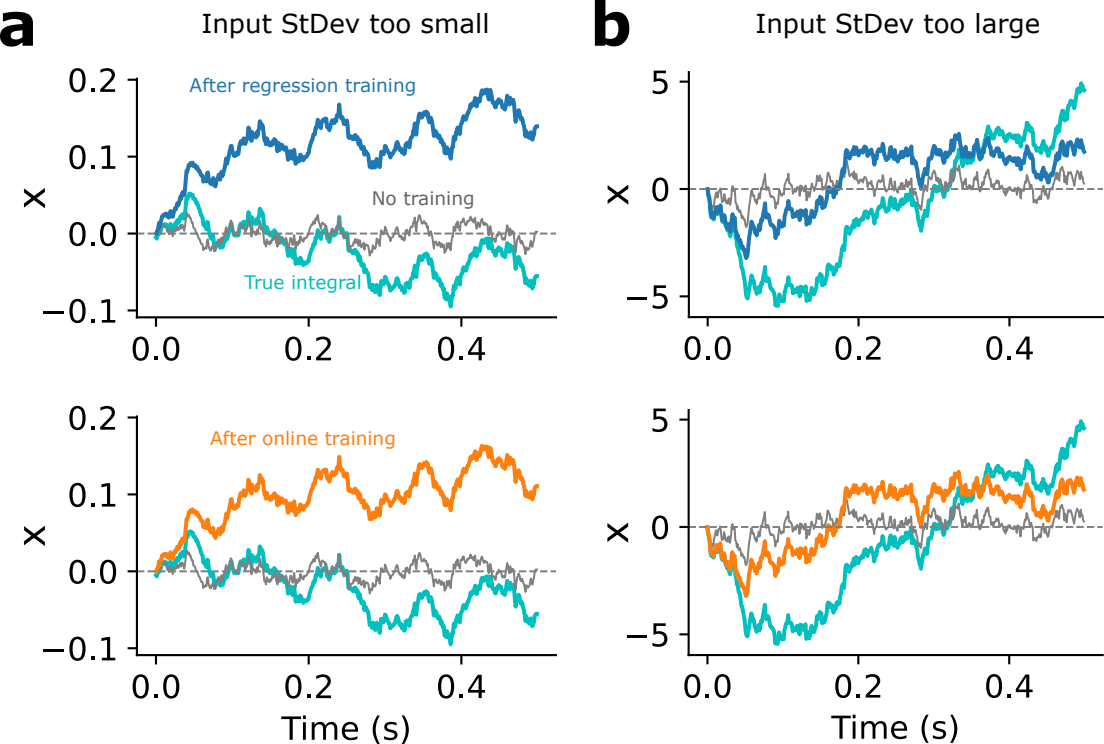

a
Offline
Decompose target flow field into support states {x_k*} (linear regression)
Online
For each x_k^*:
Use input to force system into state aligned with x_k^*
Apply instructive signal a_k^*/dt to add support state
b
Example target & support states
(bistable attractor)
Pre-learning
Learning phase
Post-learning
State (x)
Instructive signal (I)
I = a_1^*/dt
Force system into state ~x_1^*
Input (u)
Time (s)
c
Target trajectory
Support state
x_1
x_2
d
Regression solution
Online solution
g
Approximate line attractor
h
Flow fields along x_2=1 line
Base
Learned (regression)
Learned (online)
dx_1/dt
e
freq = 4 Hz
freq = 1.5 Hz
x
Time (s)
f
Error vs target frequency and # support states (N*)
Regression
Online
N*=15
N*=3
N*=5
N*=1
Error (MSE)
Target frequency (Hz)
i
Integration task error
τ
N*=3
N*=11
N*=21
Squared error
Time (s)
j
After regression training
Trained network
$\int u_1(t)dt$
Base network
After online training
x
Time (s)

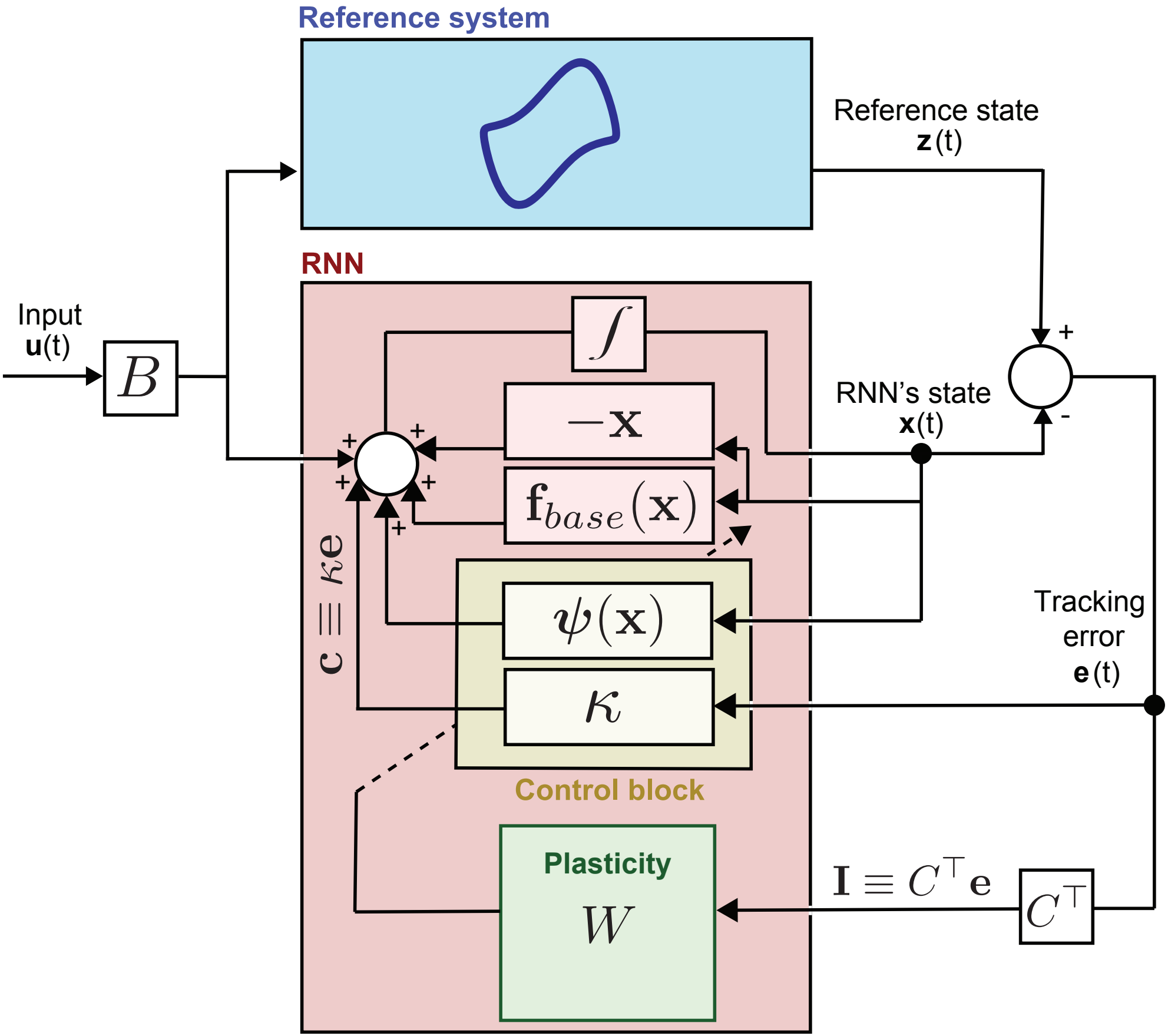

a
Reference system
Reference state
z(t)
Input
u(t)
+
-
RNN
RNN's state
x(t)
$\mathbf{c} \equiv \kappa \mathbf{e}$
$\mathcal{K}$
$C^\top$
$\mathbf{I} \equiv C^\top \mathbf{e}$
Tracking error
e(t)
b
Reference system
Learner system
Van der Pol oscillator
z_2
z_1
0
10
20
30
Before learning
After learning
Case 1: $f_{base} = 0$
Case 2: $f_{base} \neq 0$
Stable linear dynamics
Limit cycle
Bi-stable nonlinear dynamics
Limit cycle
x_1
x_2

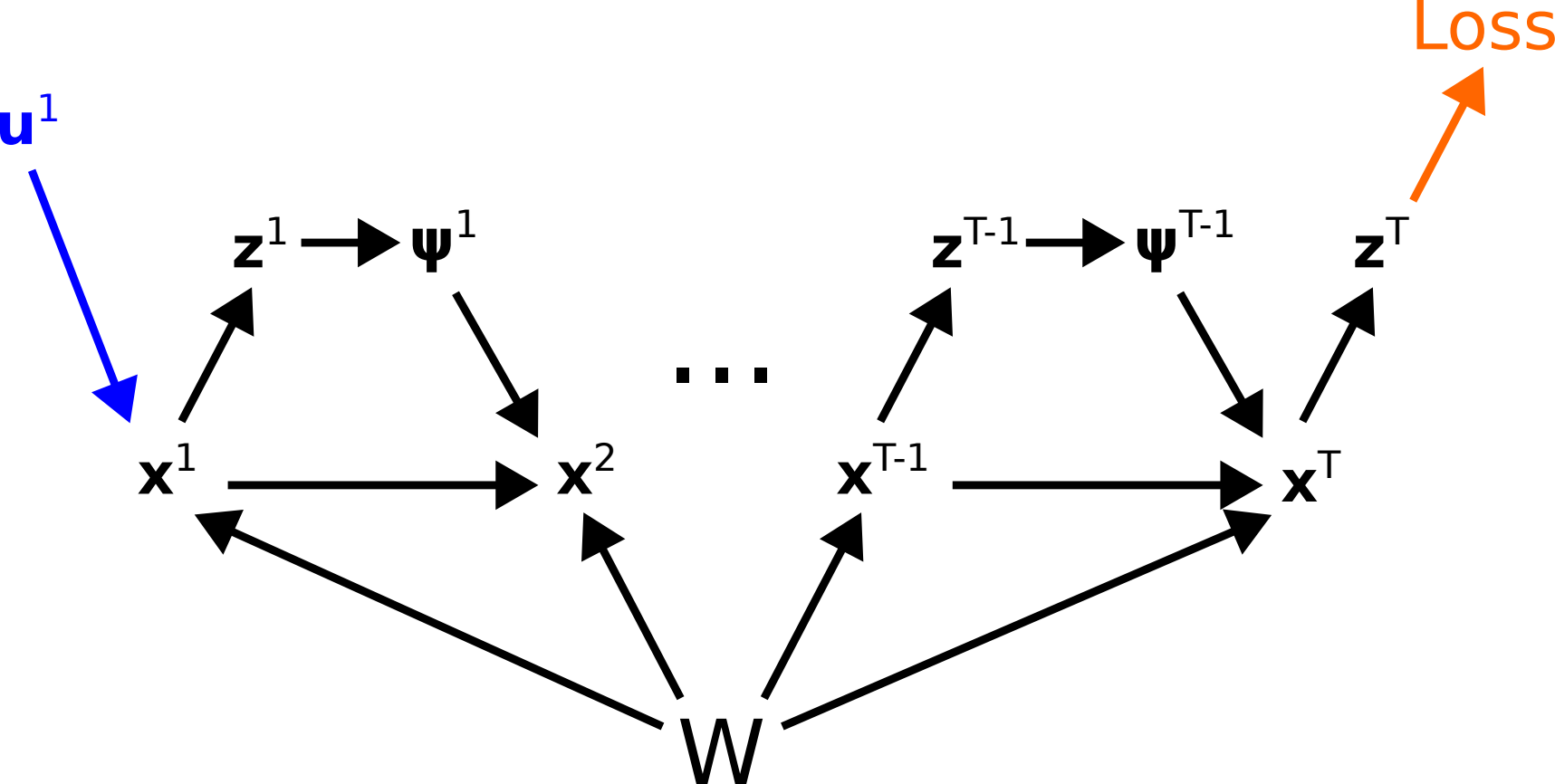

a
Computation graph for instructor training
u1
Loss
x1
x2
xT-1
xT
Path through {Wt}
W1
W2
WT-1
WT
I1
I2
IT-1
IT
Optimize I
Test network using final weights W, with I = 0
b
Hold task with delayed loss evaluation
Input u1
Input u2
x
x2
x1
Target
Delay
Time (s)
c
BPTT
Delay=500ms
Delay=60ms
Loss
d
e-prop
Epoch
e
Instruction delay
I
Input u1
Trial 1
Trial 2
Time
f
Loss
Trial 1
Trial 8
Epoch
g
Test trial post-learning for 500 ms delay
I set to 0
h
I
Trial
i
Test loss
τ
Instruction Delay (s)

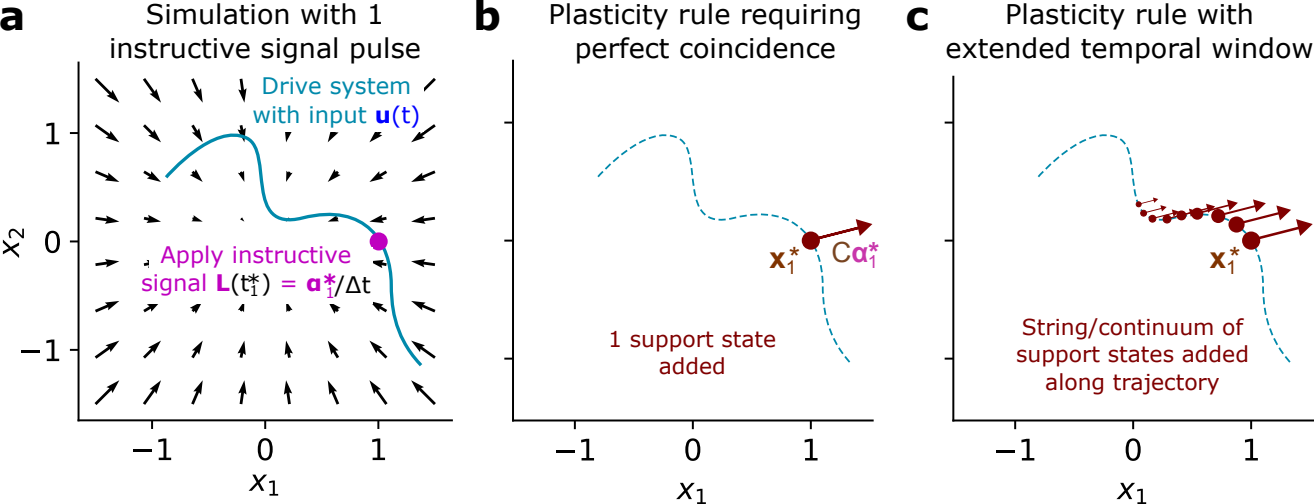

a
Expectation of
g→∞ Rand-Sig kernel
Theory
k(x, x*)
θ(x, x*)
b
D=2
D=10
Error
N
c
Target change
to flow field
Reconstruction using
g→∞ Rand-Sig Ψ
Reconstruction using
g→∞ Rand-Sig-Bias Ψ
Radially invariant
Radially varying
x1
x2

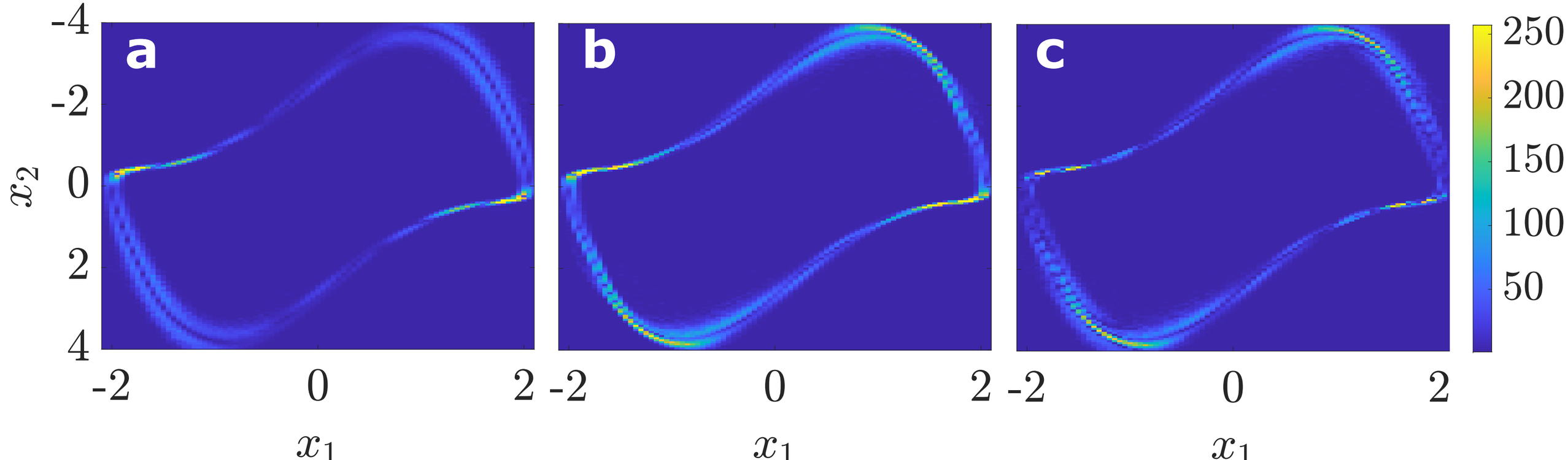

Bistable attractor
Limit cycle
Line attractor
a
b
c
d
e
f
x_1
x_2
x
Time (s)


a
Input StdDev too small
b
Input StdDev too large
After regression training
No training
True integral
After online training
x
0.2
0.1
0.0
−0.1
5
0
−5
0.0
0.2
0.4
Time (s)


Reference system
Reference state
z(t)
RNN
Input
u(t)
B
∫
−x
f_base(x)
RNN's state
x(t)
+
-
c ≡ κe
ψ(x)
κ
Tracking
error
e(t)
Control block
Plasticity
W
I ≡ C^⊤ e
C^⊤


Loss
u1
z1
ψ1
zT-1
ψT-1
zT
...
x1
x2
xT-1
xT
W


a
Simulation with 1 instructive signal pulse
Drive system with input u(t)
Apply instructive signal L(t1*) = α1*/Δt
1
0
−1
x2
−1
0
1
x1
b
Plasticity rule requiring perfect coincidence
x1*
Cα1*
1 support state added
−1
0
1
x1
c
Plasticity rule with extended temporal window
x1*
String/continuum of support states added along trajectory
−1
0
1
x1


a
b
c
-4
-2
0
2
4
x_2
-2
0
2
x_1
250
200
150
100
50