
Appendix: Volume Transmission Implements Context Factorization to Target Online Credit Assignment and Enable Compositional Generalization

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 The modern connectivist framing of neural computation emphasizes the primacy
2 of synaptic communication at the risk of neglecting the influence of the surround-
3 ing neuromodulatory environment — a neuron’s ‘biophysical context.’ Decades
4 of experimental work has established two views of neuromodulator (NMs) in-
5 fluence: 1) NMs significantly alter circuit dynamics and 2) NMs gate synaptic
6 plasticity, acting as a ‘third factor’ in learning. Here, we demonstrate these dual
7 observations are complementary aspects of a unified mechanism, embodied in
8 an endogenously neuromodulated Recurrent Neural Network (e-nmRNN). We
9 derive the nmRNN dynamics through a rate reduction of NM volume transmission,
10 capturing how NMs shape each neuron’s biophysical context. This emergent model
11 implements a context-factorized hypernetwork, which we explicitly connect to
12 attention mechanisms, traditional gating, and hybrid state-space models. We show
13 that multiplicative NM gating distinctly influences dynamical bifurcations (e.g.,
14 Hopf, Pitchfork) compared to additive synaptic drive. When trained, e-nmRNNs ex-
15 hibit context-targeted credit assignment enabling the maturation of learning how to
16 learn by approximating gradient descent. Furthermore, this bio-inspired e-nmRNN
17 achieves strong compositional generalization in sequence-to-sequence tasks, match-
18 ing more complex gating schemes but with greater hyperparameter robustness. We
19 introduce model variants incorporating spatially embedded cell types, confirming
20 that this inductive bias improves generalization. When tested on neuroscience-
21 inspired compositional task sets, the e-nmRNN’s adaptive generalization surpasses
22 strong baselines. Crucially, the ready interpretability of the e-nmRNN allows us to
23 reverse engineer cell-type-specific NM dynamics and emergent cell specialization,
24 confirming that contextual information is indeed encoded within these modulatory
25 signals. Looking forward, we view e-nmRNNs as a performant, yet biologically
26 interpretable model for probing the natural language of neural computation.

27 1 Detailed Derivation of e-nmRNN Dynamics

28 LIF Neuron Model with Neuromodulation via volume transmission

29 To capture neurobiological realism, we augment the LIF model to include the influence of multiple
30 neuromodulator types. We begin with a network of N LIF neurons. The subthreshold dynamics of
31 the membrane potential $V_i(t)$ for neuron i are given by:

$$C_m \frac{dV_i}{dt} = -g_L(V_i - E_L) + I_{syn,i}(t) + I_{ext,i}(t) \quad (1)$$

where C_m is membrane capacitance, g_L is leak conductance, E_L is leak reversal potential, $I_{syn,i}(t)$ is the total synaptic current, and $I_{ext,i}(t)$ is external input current. A spike is emitted when $V_i(t)$ reaches threshold V_{th} , followed by a reset to V_{reset} and a refractory period τ_{ref} .

We introduce M types of neuromodulators with concentrations $n_k(t)$, $k = 1, \dots, M$.

Each cell will be defined by a cell type z_i which controls the distribution of its parameters.

These parameters are an N -dimensional vector (one number for each of N neuromodulators) which control:

1. \vec{I}_n the neuromodulator current output with a spike
2. τ_r the timescale of the release from that single cell
3. \vec{I}_e the eligibility current signaling the sensitivity to each neuromodulator
4. $\vec{\tau}_e$ the decay timescale of the eligibility

Let's constrain these by neuromodulator release which is simply $I_e * \tau_e$ so that shorter

The last element is the timescale of each neuromodulator decay. $\vec{\tau}_N$

These then control both how much each cell contributes to the pool of neuromodulators and how sensitive they are to that pool via multiplicative gating.

Every cell now has a larger state, the membrane potential plus the eligibility for each of the neuromodulatory molecules. There is also a shared state for the whole network interacting through the well-mixed neuromodulatory pool.

Postsynaptic and Presynaptic Effects on Synaptic Current

Neuromodulators can affect neuronal excitability (postsynaptic) and neurotransmitter release (presynaptic). We model these combined effects by modulating the effective synaptic strength. The total synaptic current is $I_{syn,i}(t) = \sum_{j=1}^N I_{ij}(t)$, where $I_{ij}(t)$ is the current from presynaptic neuron j . Let t_j^f be the time of the f -th spike from neuron j .

$$I_{ij}(t) = \sum_f \alpha(t - t_j^f) \cdot W_{ij}^{eff}(t) \quad (2)$$

Here, $\alpha(t)$ is the postsynaptic current kernel (e.g., $\alpha(t) = \frac{Q_{syn}}{\tau_{syn}} e^{-t/\tau_{syn}}$ for $t \geq 0$), and $W_{ij}^{eff}(t)$ is the effective synaptic strength. We model the neuromodulatory effect on this strength as:

$$W_{ij}^{eff}(t) = W_{ij}^{base} + \sum_{k=1}^M T_{ijk}^{LIF} n_k(t) \quad (3)$$

where W_{ij}^{base} is the baseline weight and T_{ijk}^{LIF} quantifies the sensitivity of the synapse $j \rightarrow i$ to modulator k . The superscript 'LIF' distinguishes this parameter from the final RNN parameter.

Substituting (3) into the expression for $I_{syn,i}(t)$:

$$I_{syn,i}(t) = \sum_{j=1}^N \sum_f \alpha(t - t_j^f) \left(W_{ij}^{base} + \sum_{k=1}^M T_{ijk}^{LIF} n_k(t) \right) \quad (4)$$

Neuromodulator Dynamics

The concentration $n_k(t)$ evolves based on release, degradation/uptake, and interactions. Let $S_j(t) = \sum_f \delta(t - t_j^f)$ be the spike train of neuron j . We model the dynamics as:

$$\tau_{n,k} \frac{dn_k}{dt} = -n_k + \sum_{j=1}^N (\tau_{n,k} R_{kj}^{LIF}) S_j(t) + \sum_{l=1}^M (\tau_{n,k} Z_{kl}^{LIF}) n_l(t) \quad (5)$$

Here, $\tau_{n,k}$ is the time constant for modulator k , R_{kj}^{LIF} is the amount of modulator k released per spike from neuron j , and Z_{kl}^{LIF} represents linear interaction effects between volume-transmitted, small molecules, including decay modification (via Z_{kk}^{LIF}) [1].

66 Rate Reduction

67 We transition to a rate-based description by averaging over spike times, replacing detailed dynamics
68 with average firing rates $r_i(t)$.

69 Approximating Firing Rate

70 The average firing rate $r_i(t)$ is approximated as a non-linear function ϕ_r (the f-I curve) of the mean
71 total input current $\langle I_{tot,i}(t) \rangle = \langle I_{syn,i}(t) \rangle + \langle I_{ext,i}(t) \rangle$:

$$r_i(t) \approx \phi_r(\langle I_{tot,i}(t) \rangle) \quad (6)$$

72 Defining and Training Excitability

73 Excitability at the single neuron level is encoded in the nonlinear function $\phi()$ that translates the
74 synaptic current into the resulting firing rate.

75 One way to conceptualize excitability is in terms of a LIF neuron which modulates its excitability by
76 way of changing the threshold needed to cause a spike, T . The Leaky Integrate and Fire neuron can
77 be written in a simplified form:

$$\dot{v} = -kv + I - s(t)v_{reset} \quad (7)$$

78 and is augmented by the fire part of the relationship with the nonlinear spiking function:

$$s(t) = \begin{cases} 0 & \text{if } v(t-1) < T \\ 1 & \text{if } v(t-1) \geq T \end{cases} \quad (8)$$

79 To find the resulting steady-state firing rate of a LIF neuron for a given driving current, I , we can
80 solve this equation in a piecewise linear fashion solving the the differential equation for the time to
81 reach threshold to find that the firing rate takes the form of:

$$f_{LIF}(I) = -k \left(\ln \left| \frac{kT - I}{kV_{reset} - I} \right| \right)^{-1} \quad (9)$$

82 This equation allows us to map the dynamics of a single neuron into a steady state approximation of
83 the firing rate as a function of the input current. This also allows us to study the effect of changing
84 the 'excitability' of the LIF neuron through a change in its threshold to fire. This can be visualized
following:

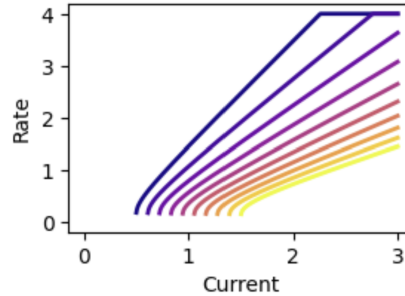


Figure 1: A critical step in the transformation between the LIF neuron and the rate neuron is the shape of the nonlinear mapping between DC input current and output firing rate in the limit of firing rate being fast compared to the dynamics of the input current. We note caution as this assumption only holds for slow directions in the dynamics space. Under this set of assumptions, we derive a closed form equation for the firing rate by completing the piecewise integral of the time to achieve threshold and inverting that to define a firing rate. Notice that this firing rate most closely follows the logistic or the ReLU-Tanh nonlinearity in that it incorporates both a critical threshold for firing and a saturation brought about by the refractory period. Colors in the plot indicate the changing of the threshold of the LIF with direct correspondence to the changes in the slope and the critical current needed to induce firing.

85

86 There are three noteworthy aspects to a change in excitability through a change in the threshold.

1. With increasing excitability, the threshold from not firing to firing decreases.
2. with increasing excitability, the slope of the characteristic linear regime increases
3. with increasing excitability, the neuron reaches its maximum firing rate at a smaller input current.

There are then two ways in which we can train the excitability. One, we can directly parameterize our nonlinearity as the LIF rate and use gradient descent to update the LIF threshold value T .

$$\phi() = f_{LIF}(I|T_i) \rightarrow \dot{T} = -\nabla_T \mathcal{L}$$

Or second, parameterize the nonlinearity in terms of two coupled parameters that merge approaches in the literature for training neural excitability:

$$\phi_i() = E_i \text{ReLU}(\tanh()) + e_i$$

where the slope and threshold of the each neuron's nonlinearity is co-modulated by their gradients on the loss function:

$$\dot{E}_i = -\nabla_{E_i} \mathcal{L} \quad (10)$$

$$\dot{e}_i = -\nabla_{e_i} \mathcal{L} \quad (11)$$

The current implementation uses the second form in terms of the ReLU ($\tanh()$) nonlinearity.

Approximating Average Synaptic Input

We replace spike trains $S_j(t)$ with rates $r_j(t)$. From (4), assuming $n_k(t)$ and $r_j(t)$ change slowly relative to $\alpha(t)$:

$$\langle I_{syn,i}(t) \rangle \approx \sum_{j=1}^N \langle \sum_f \alpha(t - t_j^f) \rangle \left(W_{ij}^{base} + \sum_{k=1}^M T_{ijk}^{LIF} n_k(t) \right) \quad (12)$$

The average current from neuron j is $\langle \sum_f \alpha(t - t_j^f) \rangle \approx r_j(t) \cdot Q_{tot}$, where $Q_{tot} = \int_0^\infty \alpha(t') dt'$. Let's define the RNN weights by absorbing Q_{tot} : $W_{ij} = Q_{tot} W_{ij}^{base}$ and $T_{ijk} = Q_{tot} T_{ijk}^{LIF}$.

$$\langle I_{syn,i}(t) \rangle \approx \sum_{j=1}^N r_j(t) \left(W_{ij} + \sum_{k=1}^M T_{ijk} n_k(t) \right) \quad (13)$$

$$= \sum_{j=1}^N W_{ij} r_j(t) + \sum_{j=1}^N \sum_{k=1}^M T_{ijk} r_j(t) n_k(t) \quad (14)$$

Approximating External Input

Assume external input $I_{ext,i}(t)$ arises from L signals $I_l(t)$ weighted by u_{il} :

$$\langle I_{ext,i}(t) \rangle \approx \sum_{l=1}^L u_{il} I_l(t) \quad (15)$$

Rate Dynamics Equation

Standard rate models include a time constant τ_r for rate dynamics:

$$\tau_r \frac{dr_i}{dt} = -r_i(t) + \phi_r(\langle I_{tot,i}(t) \rangle) \quad (16)$$

Substituting (14) and (15):

$$\tau_r \frac{dr_i}{dt} = -r_i(t) + \phi_r \left(\sum_{j=1}^N W_{ij} r_j(t) + \sum_{j=1}^N \sum_{k=1}^M T_{ijk} r_j(t) n_k(t) + \sum_{l=1}^L u_{il} I_l(t) \right) \quad (17)$$

104 Neuromodulator Dynamics in Rate Terms

105 Replace $S_j(t)$ with $r_j(t)$ in (5):

$$\tau_{n,k} \frac{dn_k}{dt} = -n_k + \sum_{j=1}^N (\tau_{n,k} R_{kj}^{LIF}) r_j(t) + \sum_{l=1}^M (\tau_{n,k} Z_{kl}^{LIF}) n_l(t) \quad (18)$$

106 Define RNN parameters $R_{kj} = \tau_{n,k} R_{kj}^{LIF}$ and effective interaction/decay term Z_{kl} such that the
107 equation becomes (after swapping index $k \rightarrow i$ for the equation, and $l \rightarrow k$ for the sum):

$$\tau_{n,i} \frac{dn_i}{dt} = \sum_{j=1}^N R_{ij} r_j(t) + \sum_{k=1}^M Z'_{ik} n_k(t) \quad (19)$$

108 Note that Z'_{ik} combines the original $-n_i$ term and the Z_{il}^{LIF} terms. Explicitly, let $Z''_{ik} = \tau_{n,i} Z_{ik}^{LIF}$.
109 Then the right hand side is $-n_i + \sum_j R_{ij} r_j + \sum_k Z''_{ik} n_k$. Grouping terms involving n_k , we get
110 $\sum_j R_{ij} r_j + \sum_k (Z''_{ik} - \delta_{ik}) n_k$. So, $Z'_{ik} = Z''_{ik} - \delta_{ik}$.

111 Matching the Target Functional Form

112 We now compare the derived rate equations (17) and (19) with the target RNN forms in the main text:

$$\dot{r}_i = \varphi \left(\sum_j W_{ij} r_j + \sum_{j,k} T_{ijk} r_j n_k + \sum_l u_{il} I_l \right) \quad (20)$$

$$\dot{n}_i = \varphi_n \left(\sum_j R_{ij} r_j + \sum_k Z_{ik} n_k \right) \quad (21)$$

113 Rate Equation (r)

114 Derived: $\tau_r \dot{r}_i = -r_i + \phi_r(\text{Input}_i)$

115 Target: $\dot{r}_i = \phi(\text{Input}_i)$

116 where $\text{Input}_i = \sum_j W_{ij} r_j + \sum_{j,k} T_{ijk} r_j n_k + \sum_l u_{il} I_l$.

117 The crucial outcome is that the *argument* of the non-linear function ϕ_r (or ϕ) has the structure
118 $W_{ij} r_j + T_{ijk} r_j n_k + u_{il} I_l$, which directly reflects the baseline synaptic drive, the neuromodulated
119 synaptic drive (via effective weight modulation W_{ij}^{eff}), and external drive derived from the LIF
120 model.

121 Obtaining the exact target form $\dot{r}_i = \phi(\text{Input}_i)$ requires interpreting it as a specific modeling choice
122 where the rate dynamics might lack the explicit decay term ($-r_i$) or absorb it and the timescale τ_r
123 into the definition of ϕ . The key justification for the target form's structure comes from the LIF
124 derivation of the input terms.

125 Neuromodulator Equation (n)

126 Derived: $\tau_{n,i} \dot{n}_i = \sum_j R_{ij} r_j + \sum_k Z'_{ik} n_k$ (a linear system)

127 Target: $\dot{n}_i = \phi_n(\sum_j R_{ij} r_j + \sum_k Z_{ik} n_k)$ (a non-linear system)

128 The derived dynamics for n_i are driven by terms related to release ($\sum_j R_{ij} r_j$) and interactions/decay
129 ($\sum_k Z'_{ik} n_k$). The target form posits a non-linear function ϕ_n governing these dynamics, likely
130 representing saturation effects or more complex regulation not captured in the simple linear model
131 (5). The structure of the argument inside ϕ_n , namely $\sum_j R_{ij} r_j + \sum_k Z_{ik} n_k$, is consistent with
132 the derived driving terms (where Z_{ik} in the target form corresponds conceptually to Z'_{ik} from the
133 derivation, potentially absorbing $\tau_{n,i}$). Using the same functional form ϕ as in the rate equation is a
134 further simplifying assumption of the target model.

Conclusion of Derivation

Starting from a LIF neuron model incorporating synaptic modulation ($W_{ij}^{eff} = W_{ij}^{base} + \sum_k T_{ijk}^{LIF} n_k$) and activity-dependent neuromodulator release/dynamics (driven by $R_{kj}^{LIF}, Z_{kl}^{LIF}$), we performed a rate reduction. This yielded:

- An average input current to neuron i with terms corresponding to baseline synapses ($W_{ij}r_j$), modulated synapses ($T_{ijk}r_jn_k$), and external inputs ($u_{il}I_l$).
- A dynamic equation for neuromodulator n_i driven by release ($R_{ij}r_j$) and interactions/decay ($Z_{ik}'n_k$).

The target RNN equations (20) and (21) represent a specific phenomenological model choice for the rate and modulator dynamics ($\dot{r} = \phi(\dots)$, $\dot{n} = \phi_n(\dots)$). The essential link provided by this derivation is the justification for the *structure of the arguments* within the non-linear functions ϕ and ϕ_n . These arguments directly map to the effective inputs and driving forces calculated from the underlying modulated LIF model. The tensors W, T, u, R, Z of the RNN encapsulate parameters from the biophysical level (baseline weights, modulation sensitivities, release rates, interaction constants, etc.), often absorbing scaling factors and timescales during the reduction process.

2 Connections to Other Architectures

Interpretation as a Context-Factorized Hypernetwork

The derived nmRNN equations (Eq. 20, 21) embody the concept of context factorization[[2, 3]. The effective connectivity $W_{ij}^{eff}(t) = W_{ij} + \sum_k T_{ijk}n_k(t)$ dynamically changes based on the neuromodulator state $\vec{n}(t)$. This state $\vec{n}(t)$, evolving on potentially slower timescales (Eq. 21)[4], acts as the 'context' that modulates the faster neuronal dynamics governed by $\vec{r}(t)$ (Eq. 20)[5].

This structure is equivalent to a specific type of hypernetwork[3]. The neuromodulator network (Eq. 21) acts as the hypernetwork, dynamically generating (part of) the parameters ($T_{ijk}n_k$) for the main RNN (Eq. 20). Both the main network and the hypernetwork have recurrent dynamics[6]. The term $T_{ijk}r_jn_k$ represents tertiary interactions arising from a low-rank approximation to matrix Taylor expansion of the dynamics.

This perspective connects nmRNNs to other dynamic network architectures:

- **Switching Linear Dynamical Systems (SLDS):** Neuromodulation $\vec{n}(t)$ can be seen as a continuous version of the discrete latent state $Z(t)$ in switching Linear Dynamical Systems, smoothly interpolating between different dynamical regimes instead of abruptly switching[cite linderman].
- **Fast Weights / Attention:** The dynamic modulation $T_{ijk}n_k(t)$ resembles fast weight programmers or attention mechanisms, where temporal context dynamically gates or modifies information flow from input to output[7].
- **Hypernetworks:** The dynamic modulation $T_{ijk}n_k(t)$ can be viewed as a linear readout hypernetwork which uses a context module to generate the weights of the core network. Here, we note that the nmRNN adopts both recurrent dynamics and the view that the well-mixed neuromodulatory field is governed by volume transmission and therefore neuronal firing.
- **High Dimensional Taylor Expansion:** Neuromodulation via volume transmission can be viewed as the next leading order contribution of a Taylor expansion. Many-body physical systems with complex interactions rules can be expanded as:

$$\dot{x}_i = f(\vec{x}) \approx \underbrace{\sum_j A_{ij}x_j}_{\text{Binary Interactions}} + \underbrace{\sum_m \sum_n B_{imn}x_mx_n}_{\text{Three-body interactions}} + \mathcal{O}(\text{Higher order})$$

. The volume transmission equations can be viewed as a low-rank representation of these three-body interactions proceeding the firing rate nonlinearity and thus represent the next leading order dynamics of the RNN.

- **Koopman operator theory (with a polynomial basis):** In the absence of the nonlinearity, this Taylor expansion can be viewed as a Truncated Koopman linear operator acting over a polynomial dictionary[8]. A valuable way forward may leverage this powerful toolkit for fitting nonlinear dynamical systems.
- **Continuous logic gates:** Gating can be viewed as the continuous "AND" gate where the corners of the space recreate the truth table for discrete input. Through combination of careful construction of the A (rank 2- tensor) and the B (rank 3 tensor), we represent "XOR" gates as well forming the basis of discrete bit universal computation[9]

We advocate for this model as a tool not only to traverse levels of understanding in neurobiology, but also as a scaffold to formally connect diverse approaches for learning in dynamical systems.

Connection to Costacurta et al., NeurIPS 2024

Despite differences in our respective derivations, we observe striking conceptual and experimental similarities to the valuable contribution by Costacurta et al. [4], which also explores the landscape of neuromodulation in recurrent neural networks. To clarify the precise connections and highlight key distinctions, we present a direct comparison of the final functional forms between the two model classes in Figure 2.

Direct comparison in same language Note: using Einstein summation
" $A_{ijk}x_jx_k$ " $\rightarrow \sum_k \sum_j A_{ijk}x_jx_k$

Our work **Both** **Costacurta et al***

$$\dot{r}_i = \phi \left(\underbrace{(W_{ij}^0 + T_{ij}^k n_k)}_{\text{Factorized context}} r_j + \underbrace{(\mu_i^a \eta_j^a \phi(A_{ak} n_k + b_a))}_{\text{Structured Variability}} r_j + u_{il} I_l \right)$$

$\tau_r < \tau_n$

$$\dot{n}_k = \theta \left(\underbrace{R_{kj} r_j}_{\text{Volume transmission with firing}} + \underbrace{Z_{kl} n_l}_{\text{Coupling between nms}} + \underbrace{u_{km} I_m}_{\text{Direct access to input}} \right)$$

$$y_i(t) = D_{ij} r_j(t) + c_i$$

*After appropriate transformation into the dual representation

Figure 2: A direct comparison with Costacurta et al. reveals both powerful similarities and key architectural differences. The most important distinctions between the two model classes include the low-rank nature of synaptic variability explored in their work, and the absence of direct access to a context signal for our NM dynamics, which enforces an endogenous, closed-system interpretation of these dynamics.

We view our work as highly complementary, yet distinctly advancing the field through several orthogonal contributions:

- **Bio-inspired Derivation:** Our model emerges from a detailed rate reduction of LIF-neuron dynamics incorporating neuromodulation via volume transmission, providing a direct link to established biophysical principles.
- **Novel Learning Rules:** We expand on the model class by deriving normative leading-order learning rules that explicitly link dynamics modification with context-targeted credit assignment, a crucial step for understanding how such systems might learn.
- **Approximation of Gradient Descent:** Crucially, we demonstrate that our e-nmRNNs can learn to approximate gradient descent and robustly encode vital learning signals such as Reward Prediction Error (RPE), providing a computational mechanism for online credit assignment in biological and artificial systems.
- **Dynamic Generalization Tasks:** Our work explores performance on neuroscience-inspired compositional task sets and dynamic generalization tasks, which are distinct in both their formal structure and biological inspiration, showcasing the model’s adaptive capabilities.

208 • **Strong Baseline Comparison:** We conduct rigorous comparisons against strong base-
 209 lines, systematically assessing the e-nmRNN’s capacity and compositional generalization
 210 performance.

211 • **Reverse Engineering Biological Specialization:** The inherent interpretability of the e-
 212 nmRNN allows us to reverse engineer cell-type-specific neuromodulator dynamics and
 213 emergent cell specialization, offering a novel computational lens for probing neural compu-
 214 tation across complex task suites like Yang et al. (2019).

215 We hope the reader will agree that these contributions are deeply complementary and both serve to
 216 advance an important conversation in neuro-AI with their own strengths.

217 3 Detailed Learning Rule Derivations

218 If we assume that it will look like a local version of gradient descent (in synapse and in time), you
 219 can find the gradient of the dynamics that will govern the normative leading order learning rules[10,
 220 11, 12].

Let’s define the task in terms of a simple MSE from a target sequence. This target sequence can be
 represented as a multiple timescales signal via the Laplace transform:

$$y^*(t) = \int dz a(z) e^{zt}$$

This makes our loss function equal to:

$$\mathcal{L} = \sum_t (y^*(t) - Dr(t))^2$$

We can expand this to timesteps required to percolate information through the network by expanding:

$$r(t) = r(t-1) + dt * \dot{r}(t-1)$$

and

$$n(t) = n(t-1) + dt * \dot{n}(t-1)$$

The first step is writing the error function as:

$$\mathcal{L} = \sum_t \epsilon(t)$$

where we assume that $W_{ij}^0 = 0$.

$$\epsilon(t) = y^*(t) - D \left(r(t-1) + dt \left[-r(t-1) + \phi \left(\sum_j \left(\sum_k T_{ij}^k n_k(t-1) \right) r_j(t-1) + UI(t-1) \right) \right] \right)$$

221 So then we need to plug in the n(t) as:

$$\begin{aligned} \epsilon(t) = y^*(t) - D \left(r(t-1) + dt \left[-r(t-1) + \phi \left(\sum_j \left(\sum_k T_{ij}^k (n_k(t-1) \right. \right. \right. \right. \\ \left. \left. \left. + dt(-n(t-1) + \theta(Rr(t-1) + Zn(t-1)))) \right) r_j \right. \right. \\ \left. \left. + UI(t-1) \right) \right] \right) \end{aligned}$$

222 We can expand this out to find:

$$\begin{aligned} \epsilon(t) = & y^*(t) - Dr(t-1) \\ & + dt \left(-r(t-1) + \phi \left(\sum_j \left(\sum_k T_{ij}^k n_k(t-1) \right) r_j(t-1) \right) \right) \\ & + dt^2 \left(-n(t-1) + \theta(Rr(t-1) + Zn(t-1) + \phi(UI(t-2) + -r(t-2) + \sum_k T^k n_k(t-2)r(t-2))) \right) \end{aligned}$$

223 By plugging these into our dynamics, we can find that our error can be written as an expansion of dt .
 224 Local in time calculations and synapse follow the criteria of being to leading order in $\mathcal{O}(dt^2)$. This
 225 then allows us to truncate this expansion at leading order.

If we expand this to leading order in dt , we find:

$$\epsilon(t) \overset{\mathcal{O}(dt)}{\approx} y^{star}(t) - Dr(t-1) + dt \left(-r(t-1) + \phi \left(\sum_j \left(\sum_k T_{ij}^k n_k(t-1) \right) r_j(t-1) \right) \right)$$

This normative leading order learning rule will take the form of:

$$\nabla_{T^k} \mathcal{L} \overset{\mathcal{O}(dt)}{\approx} D^T \epsilon(t) \phi'(A(t)) n_k r(t)$$

226 Interestingly, the presence of the k th neuromodulator should gate the update of the k th page of the
 227 three-body tensor. The gating via this multiplicative factor puts the neuromodulator on equal footing
 228 to the error.

Next, the gradient of the input sensitivity takes the form of:

$$\nabla_u \mathcal{L} \overset{\mathcal{O}(dt)}{\approx} D^T \epsilon(t) \phi'(A(t)) n_k I(t)$$

229 Again, the k th neuromodulator gates the input learning as well.

Next, we can study the gradient of the neuromodulator dynamics with respect to the error:

$$\nabla_R \mathcal{L} \overset{\mathcal{O}(dt^2)}{\approx} D^T \epsilon(t) \phi'(A(t)) \theta'(B(t)) r^2(t)$$

and the coupling matrix between neuromodulators to leading order takes the form of:

$$\nabla_Z \mathcal{L} \overset{\mathcal{O}(dt^2)}{\approx} D^T \epsilon(t) \phi'(A(t)) \theta'(B(t)) n(t) r(t)$$

230 These leading order learning rules could be sensible hypotheses for how this joint system would
 231 evolve if it was tuned to solve the problem of error minimization. (There is an analogous learning
 232 rule for minimizing the RL loss derived from Policy Gradients).

233 4 Statistical Significance of Gradient Alignment

In our analysis, we investigate the relationship between the dynamics carried out by the neuromodulators and the gradient of a surrogate model. A surrogate model allows us to calculate a normative learning signal, by way of ideal gradient to improve performance on the task. We accomplish this by locking the parameters of the e-nmRNN to a coevolving RNN by setting: $W_{ij}^{eff} = W_{ij} + T_{ijk} \bar{n}_k$ over every window of size $T = 100$. We then use autodiff to calculate how $\dot{W}_{ij}^{eff} = -\alpha \nabla_W \mathcal{L}$ to best reduce the loss of the surrogate network. For the neuromodulator dynamics to "fall into alignment" with this surrogate model's gradient means that the neuromodulator states evolve in a direction that closely matches the optimal update direction prescribed by the surrogate model or in math:

$$T_{ijk} \dot{n}_k(t) \overset{\rightarrow}{\underset{\text{learns to approximate}}{}} -\alpha \nabla_W \mathcal{L}$$

234 Significance of Cosine Similarity in High Dimensions

235 To understand if the observed alignment is significant, we will study observed cosine similarity of \sim
 236 0.15 between two vectors in a high-dimensional space of $D = 128^2 = 16384$ dimensions. In high-
 237 dimensional spaces, the distribution of cosine similarities between two randomly and independently
 238 drawn vectors tends to concentrate sharply around zero. Specifically, for vectors whose components

are drawn independently from a distribution with zero mean and finite variance, the expected cosine similarity is 0. The variance of the cosine similarity between two random unit vectors in D dimensions is approximately $1/D$.

Our null hypothesis (H_0) is that the observed cosine similarity arises from two randomly and independently selected vectors, implying no true underlying alignment. Under this null hypothesis, the distribution of cosine similarities has a mean of 0 and a standard deviation of approximately $\sigma = \sqrt{1/D} = \sqrt{1/16384} \approx 0.0078$. The observed cosine similarity of 0.13 is approximately ~ 16 standard deviations away from the expected mean of 0 under the null hypothesis. Such a deviation is extremely improbable under this simple null model. Therefore, the observed and slowly accumulated cosine similarity of 0.15 constitutes a statistically significant alignment between the gradient of the surrogate model and the dynamics carried out by the neuromodulators, providing evidence against the null hypothesis that the vectors are unaligned and randomly oriented in this high-dimensional space. This combined with the learned nature of this alignment suggests numerically, that e-nmRNNs not only can, but do learn to approximate gradient descent through gradient descent similar to that observed in meta-optimizers[13] and transformers[14].

5 Dynamical Regimes and Bifurcations

Traversing Bifurcations: Multiplicative vs. Additive Modulation

Dynamical systems often exhibit bifurcations, where a small parameter change leads to a qualitative shift in behavior (e.g., from a stable fixed point to oscillations). Context factorization provides a distinct mechanism for navigating these critical transitions compared to traditional input modulation, and these mechanisms leave different measurable signatures.

We can illustrate this difference using a simple 2-neuron RNN model:

$$\frac{dr_1}{dt} = -r_1 + \tanh(n(W_{11}r_1 + W_{12}r_2) + r_3) \quad (22)$$

$$\frac{dr_2}{dt} = -r_2 + \tanh(n(W_{21}r_1 + W_{22}r_2) + r_3) \quad (23)$$

Here, n represents multiplicative context factorization (gain modulation), while r_3 represents additive context input (e.g., from a dedicated context neuron). Both can drive bifurcations, but how they do so differs fundamentally.

Multiplicative Modulation (n): When $r_3 = 0$, the origin $(0,0)$ is always a fixed point due to the odd symmetry of \tanh . Linearizing around the origin reveals eigenvalues $\lambda = (2n - 1) \pm in$ for the specific W matrix used in simulations ($W_{11} = W_{22} = 2, W_{12} = -1, W_{21} = 1$). A Hopf bifurcation occurs when $\text{Re}(\lambda) = 0$, which happens precisely at $n = 1/2$. As n increases through $1/2$, the origin transitions from a stable spiral to an unstable spiral surrounded by a stable limit cycle (oscillation)[15]. Critically, this bifurcation happens *without shifting the fixed point away from the origin*. The bifurcation preserves the system’s symmetry around $(0,0)$.

Additive Modulation (r_3): When $r_3 \neq 0$, the origin is no longer a fixed point ($\tanh(r_3) \neq 0$). The fixed points (\bar{r}_1, \bar{r}_2) satisfying $\bar{r}_i = \tanh(n(W\bar{\mathbf{r}})_i + r_3)$ shift as r_3 changes. Stability is determined by the Jacobian evaluated at these non-zero fixed points:

$$J(\bar{r}_1, \bar{r}_2) = n \begin{pmatrix} (1 - \bar{r}_1^2)W_{11} & (1 - \bar{r}_1^2)W_{12} \\ (1 - \bar{r}_2^2)W_{21} & (1 - \bar{r}_2^2)W_{22} \end{pmatrix} - I$$

Changes in r_3 alter (\bar{r}_1, \bar{r}_2) and thus the Jacobian elements, potentially driving eigenvalues across the imaginary axis. However, any bifurcation induced by r_3 necessarily occurs at a non-zero operating point (\bar{r}_1, \bar{r}_2) and breaks the symmetry seen when $r_3 = 0$. Depending on the specific path traced by (\bar{r}_1, \bar{r}_2) , r_3 might induce Hopf or saddle-node bifurcations.

Distinguishing Mechanisms (Measurable Differences): These differences provide experimentally testable predictions (visualized in Figure 1B):

1. **Operating Point Shift:** A key distinction is whether the mean activity (average \bar{r}_1, \bar{r}_2) shifts significantly as the system crosses a bifurcation. Additive modulation (r_3) inherently involves an operating point shift, while multiplicative modulation (n with $r_3 = 0$) can

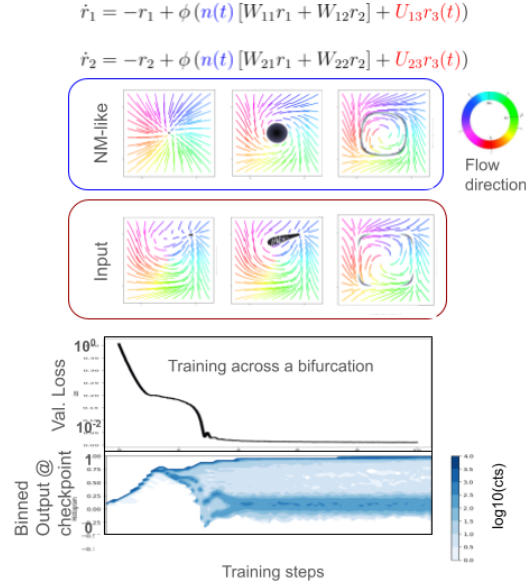


Figure 3: Studying the role of gating in the Hopf bifurcation in tuning and in training. A) We study these two 3 dimensional dynamical systems. Black text is present in both. Blue represents context factorization and Red represents the presence of context storing neurons. B) We show the difference of how nm-like and input tuning across the two Bifurcations looks for the $\tanh()$ nonlinearity. Flow directions are signaled by the color of the streamline denoted in the circular key to the right. C) We show the evolution of the signal output across training and show that the 'epiphany' moment in the validation loss cooresponds to the emergence of the oscillations suggesting the dominant challenge is the formation of the limit cycle.

280 induce bifurcations (like the Hopf bifurcation at the origin) without necessarily changing
 281 the mean activity level of that specific fixed point.

282 2. **Symmetry Preservation/Breaking:** Bifurcations driven by multiplicative gain modulation
 283 in a system with appropriate symmetries (like $r_3 = 0$ here) tend to preserve those symmetries
 284 (e.g., limit cycle emergence centered around the origin). Additive inputs break this symmetry,
 285 leading to asymmetric dynamics relative to the origin.

286 Observing whether a qualitative change in network dynamics (e.g., onset of oscillations) is accom-
 287 panied by a shift in the mean firing rates provides a potential signature to distinguish between gain
 288 modulation (context factorization) and direct input modulation as mechanisms for context-dependent
 289 dynamic switching in biological or artificial networks.

290 Bifurcation Analysis of a Rank-2 RNN with Multiplicative and Additive Context Modulation

291 We analyze a 2-neuron Recurrent Neural Network (RNN) whose dynamics are given by the following
 292 system of coupled ordinary differential equations:

$$\frac{dr_1}{dt} = -r_1 + \phi(n(W_{11}r_1 + W_{12}r_2) + r_3) \quad (24)$$

$$\frac{dr_2}{dt} = -r_2 + \phi(n(W_{21}r_1 + W_{22}r_2) + r_3) \quad (25)$$

293 where:

- 294 • $r_1(t)$ and $r_2(t)$ are the firing rates of the two neurons.
- 295 • $\phi(x) = \tanh(x)$ is the non-linear activation function.
- 296 • $W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} = \begin{pmatrix} 2 & -1 \\ 1 & 2 \end{pmatrix}$ is the synaptic weight matrix.

- n is a multiplicative factor representing gain modulation or context factorization. It scales the synaptic inputs before the nonlinearity.
- r_3 is an additive factor representing external input or the influence of a 'context neuron'. It shifts the input to the nonlinearity.

Both n and r_3 act as control parameters that can drive bifurcations in the system's dynamics. We aim to understand the distinct effects of varying n versus varying r_3 .

Fixed Point Analysis

Fixed points (\bar{r}_1, \bar{r}_2) of the system are solutions to $\frac{dr_1}{dt} = 0$ and $\frac{dr_2}{dt} = 0$. This yields the equations:

$$\bar{r}_1 = \phi(n(W_{11}\bar{r}_1 + W_{12}\bar{r}_2) + r_3) \quad (26)$$

$$\bar{r}_2 = \phi(n(W_{21}\bar{r}_1 + W_{22}\bar{r}_2) + r_3) \quad (27)$$

Let $\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}$ and define the input vector $\mathbf{x} = nW\mathbf{r} + \begin{pmatrix} r_3 \\ r_3 \end{pmatrix}$. Then the fixed point equation is $\bar{\mathbf{r}} = \phi(\mathbf{x}(\bar{\mathbf{r}}))$.

Symmetry and the Origin

Consider the case when the additive input $r_3 = 0$. The fixed point equations become:

$$\bar{r}_1 = \tanh(n(W_{11}\bar{r}_1 + W_{12}\bar{r}_2))$$

$$\bar{r}_2 = \tanh(n(W_{21}\bar{r}_1 + W_{22}\bar{r}_2))$$

Since $\tanh(0) = 0$, the origin $(\bar{r}_1, \bar{r}_2) = (0, 0)$ is always a fixed point when $r_3 = 0$, regardless of the value of the multiplicative factor n .

Now consider the case when $r_3 \neq 0$. The origin $(0, 0)$ is a fixed point only if:

$$0 = \tanh(n(0) + r_3) = \tanh(r_3)$$

$$0 = \tanh(n(0) + r_3) = \tanh(r_3)$$

This requires $r_3 = 0$. Therefore, the additive factor r_3 breaks the symmetry that keeps the origin fixed. An additive input $r_3 \neq 0$ shifts the location of the fixed point(s) away from the origin.

Measurable Difference 1: The mean activity (operating point), corresponding to the fixed point location (\bar{r}_1, \bar{r}_2) , will generally shift when a bifurcation is induced by changing r_3 . In contrast, if a bifurcation is induced by changing n while $r_3 = 0$, the origin $(0, 0)$ remains a fixed point, although its stability may change and other fixed points may appear/disappear symmetrically around the origin.

Stability Analysis and Bifurcations

To analyze the stability of a fixed point (\bar{r}_1, \bar{r}_2) , we linearize the system around it by computing the Jacobian matrix J :

$$J = \begin{pmatrix} \frac{\partial \dot{r}_1}{\partial r_1} & \frac{\partial \dot{r}_1}{\partial r_2} \\ \frac{\partial \dot{r}_2}{\partial r_1} & \frac{\partial \dot{r}_2}{\partial r_2} \end{pmatrix}_{(\bar{r}_1, \bar{r}_2)}$$

Let $x_1 = n(W_{11}r_1 + W_{12}r_2) + r_3$ and $x_2 = n(W_{21}r_1 + W_{22}r_2) + r_3$. Recall $\phi'(x) = \text{sech}^2(x) = 1 - \tanh^2(x)$. The partial derivatives are:

$$\frac{\partial \dot{r}_1}{\partial r_1} = -1 + \phi'(x_1) \cdot nW_{11}$$

$$\frac{\partial \dot{r}_1}{\partial r_2} = \phi'(x_1) \cdot nW_{12}$$

$$\frac{\partial \dot{r}_2}{\partial r_1} = \phi'(x_2) \cdot nW_{21}$$

$$\frac{\partial \dot{r}_2}{\partial r_2} = -1 + \phi'(x_2) \cdot nW_{22}$$

At a fixed point (\bar{r}_1, \bar{r}_2) , we have $\bar{r}_1 = \phi(\bar{x}_1)$ and $\bar{r}_2 = \phi(\bar{x}_2)$. So, $\phi'(\bar{x}_i) = 1 - \phi^2(\bar{x}_i) = 1 - \bar{r}_i^2$. The Jacobian at the fixed point (\bar{r}_1, \bar{r}_2) is:

$$J(\bar{r}_1, \bar{r}_2) = \begin{pmatrix} -1 + (1 - \bar{r}_1^2)nW_{11} & (1 - \bar{r}_1^2)nW_{12} \\ (1 - \bar{r}_2^2)nW_{21} & -1 + (1 - \bar{r}_2^2)nW_{22} \end{pmatrix}$$

$$J(\bar{r}_1, \bar{r}_2) = n \begin{pmatrix} (1 - \bar{r}_1^2)W_{11} & (1 - \bar{r}_1^2)W_{12} \\ (1 - \bar{r}_2^2)W_{21} & (1 - \bar{r}_2^2)W_{22} \end{pmatrix} - I$$

where I is the identity matrix.

Stability of the Origin (when $r_3 = 0$)

When $r_3 = 0$, the origin $(\bar{r}_1, \bar{r}_2) = (0, 0)$ is always a fixed point. Its stability is determined by the Jacobian evaluated at $(0, 0)$:

$$J(0, 0) = n \begin{pmatrix} (1 - 0^2)W_{11} & (1 - 0^2)W_{12} \\ (1 - 0^2)W_{21} & (1 - 0^2)W_{22} \end{pmatrix} - I = nW - I$$

Substituting the values for W :

$$J(0, 0) = n \begin{pmatrix} 2 & -1 \\ 1 & 2 \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 2n - 1 & -n \\ n & 2n - 1 \end{pmatrix}$$

The eigenvalues λ are found from the characteristic equation $\det(J(0, 0) - \lambda I) = 0$:

$$(2n - 1 - \lambda)^2 - (-n)(n) = 0$$

$$(2n - 1 - \lambda)^2 + n^2 = 0$$

$$\lambda - (2n - 1) = \pm \sqrt{-n^2} = \pm in$$

$$\lambda = (2n - 1) \pm in$$

A bifurcation occurs when the real part of the eigenvalues crosses zero. Here, $\text{Re}(\lambda) = 2n - 1$.

- If $2n - 1 < 0$ (i.e., $n < 1/2$), $\text{Re}(\lambda) < 0$. The origin is a stable fixed point (specifically, a stable spiral since $\text{Im}(\lambda) = \pm n \neq 0$ for $n \neq 0$).
- If $2n - 1 > 0$ (i.e., $n > 1/2$), $\text{Re}(\lambda) > 0$. The origin is an unstable fixed point (an unstable spiral).
- If $2n - 1 = 0$ (i.e., $n = 1/2$), $\text{Re}(\lambda) = 0$. The eigenvalues are purely imaginary ($\lambda = \pm i/2$). This indicates a **Hopf bifurcation** at $n = 1/2$.

As n increases through $1/2$ (with $r_3 = 0$), the stable spiral fixed point at the origin loses stability, giving rise to a stable limit cycle surrounding the now unstable origin. This bifurcation occurs without shifting the location of the fixed point itself (it remains at $(0, 0)$).

Effect of Additive Input ($r_3 \neq 0$)

When $r_3 \neq 0$, the origin is no longer a fixed point. The fixed points (\bar{r}_1, \bar{r}_2) must be found numerically or graphically by solving Eqs. 26-27. The Jacobian must be evaluated at these non-zero fixed points:

$$J(\bar{r}_1, \bar{r}_2) = n \begin{pmatrix} (1 - \bar{r}_1^2)W_{11} & (1 - \bar{r}_1^2)W_{12} \\ (1 - \bar{r}_2^2)W_{21} & (1 - \bar{r}_2^2)W_{22} \end{pmatrix} - I$$

As r_3 is varied (keeping n fixed, e.g., $n = 1$), the location of the fixed points (\bar{r}_1, \bar{r}_2) will change. The terms $(1 - \bar{r}_i^2)$ in the Jacobian will also change. Bifurcations can occur when the eigenvalues of $J(\bar{r}_1, \bar{r}_2)$ cross the imaginary axis.

Consider the trace and determinant of the Jacobian:

$$\text{Tr}(J) = (-1 + (1 - \bar{r}_1^2)nW_{11}) + (-1 + (1 - \bar{r}_2^2)nW_{22})$$

$$= n((1 - \bar{r}_1^2)W_{11} + (1 - \bar{r}_2^2)W_{22}) - 2$$

$$\det(J) = (-1 + (1 - \bar{r}_1^2)nW_{11})(-1 + (1 - \bar{r}_2^2)nW_{22}) - ((1 - \bar{r}_1^2)nW_{12})((1 - \bar{r}_2^2)nW_{21})$$

338 A Hopf bifurcation occurs when $\text{Tr}(J) = 0$ and $\det(J) > 0$. A saddle-node bifurcation (where fixed
339 points appear or disappear) occurs when $\det(J) = 0$.

340 Varying r_3 changes (\bar{r}_1, \bar{r}_2) , which in turn affects both the trace and determinant. This means r_3 can
341 potentially drive both Hopf and saddle-node bifurcations, depending on the value of n and the specific
342 path taken by (\bar{r}_1, \bar{r}_2) as r_3 changes. Crucially, any bifurcation induced by r_3 will be associated with
343 a fixed point $(\bar{r}_1, \bar{r}_2) \neq (0, 0)$.

344 **Measurable Difference 2:** Bifurcations driven by the multiplicative factor n (with $r_3 = 0$) preserve
345 the symmetry around the origin. For instance, the Hopf bifurcation at $n = 1/2$ happens precisely at
346 the origin. Bifurcations driven by the additive factor r_3 occur at non-zero operating points (\bar{r}_1, \bar{r}_2) and
347 generally break the symmetry observed when $r_3 = 0$. The type of bifurcation (Hopf vs. saddle-node)
348 might also differ depending on whether n or r_3 is the control parameter.

349 Summary of Measurable Differences

350 To distinguish between bifurcations driven by multiplicative gain modulation (n) versus additive
351 input (r_3):

352 1. **Operating Point / Mean Activity:** Monitor the average values of r_1 and r_2 .

- 353 • Varying n (with $r_3 = 0$): The origin $(0, 0)$ remains a fixed point. Bifurcations (like
354 Hopf at $n = 1/2$) occur centered at the origin. Other fixed points may appear/disappear
355 symmetrically.
- 356 • Varying r_3 : The fixed point(s) move as r_3 changes. Bifurcations occur at non-zero operating
357 points (\bar{r}_1, \bar{r}_2) . The mean activity changes significantly across the bifurcation.

358 2. **Symmetry:** Observe the phase portrait.

- 359 • Varying n (with $r_3 = 0$): The dynamics (fixed points, limit cycles) should exhibit symmetry,
360 e.g., if (\bar{r}_1, \bar{r}_2) is a fixed point, $(-\bar{r}_1, -\bar{r}_2)$ might also be (depending on W , though not
361 guaranteed for non-zero fixed points with this specific W). The Hopf bifurcation occurs
362 symmetrically around the origin.
- 363 • Varying r_3 : The symmetry around the origin is broken. Fixed points and limit cycles will
364 generally not be symmetric.

365 3. **Bifurcation Type at Origin:**

- 366 • Varying n (with $r_3 = 0$): Specifically leads to a Hopf bifurcation at the origin when
367 $n = 1/2$.
- 368 • Varying r_3 : Cannot induce a bifurcation exactly *at* the origin (as it's not a fixed point for
369 $r_3 \neq 0$). Bifurcations occur elsewhere in the phase space.

370 By measuring the mean activity levels (\bar{r}_1, \bar{r}_2) and analyzing the symmetry of the dynamics as
371 the control parameter is varied, one can distinguish whether a bifurcation is primarily driven by a
372 multiplicative (gain-like) or additive (input-like) contextual factor.

373 6 Deep Reinforcement Learning Experiments

374 For the reinforcement learning tasks, we utilized a block-switched two-armed bandit task where
375 reward probabilities invert stochastically (Figure 4A). The agent must learn to track the hidden
376 block state (i.e., which arm has high probability) and adapt its actions to maximize reward. Train-
377 ing e-nmRNN agents was performed using Proximal Policy Optimization (PPO), a standard deep
378 reinforcement learning algorithm.

379 To analyze the RPE encoding, Maximum Likelihood Estimation (MLE) was used to fit a Q-learning
380 model to the generated action and reward sequences from the trained e-nmRNN agent. The RPE
381 signal was then extracted directly from this fitted Q-table. Finally, a single degree-of-freedom linear
382 decoder was trained to map neuromodulator states to various behavioral features, including RPE,
383 choice, reward, and entropy.

Table 1: Key Hyperparameters for Reinforcement Learning Task

Parameter	Value/Range (PPO/TPE)	Notes
Batched Environments	16	Fixed
PPO Learning Rate	7×10^{-5}	Fixed
PPO Clip Epsilon	1	Fixed
Discount Factor (γ)	0.95	Fixed
Generalized Advantage Estimation (λ)	0.95	Fixed
Sequence Length	200	Fixed
Hidden Units (RNN/e-nmRNN)	64	Parameter-matched
Number of NMs	4	Fixed
NM time constant (τ_n)	0.1	Optimized by Hand
Activation Function	ReLU-Tanh	Fixed

Table 2: Experiment Parameters for the agents shown in the main text.

Parameter	Value	Parameter	Value
env_name	BlockBandit2ArmCoupledEasy-v0	policy_use_gae	true
exp_label	rl2	policy_lambda	0.95
max_episode_steps	200	policy_algorithm	a2c
max_rollouts_per_task	1	policy_critic_loss_coeff	0.01
seed	73	policy_entropy_loss_coeff	0.3
time_as_state	false	policy_optimizer	adam
deterministic_execution	false	policy_eps	1e-08
results_log_dir	/scratch/	policy_lr	0.0007
log_interval	500	policy_anneal_lr	false
save_interval	1000	policy_max_grad_norm	0.5
save_intermediate_models	false	policy_use_activity_l2_regularization	true
eval_interval	500	policy_activity_l2_loss_coeff	100
eval_ids	[]	NMd	true
num_eval_envs	300	nNM	4
vis_interval	500	state_dim	1
num_updates	50000.0	input_state_dim_for_policy	0
policy_num_steps_per_update	200	action_space_type	Discrete
num_processes	16	action_dim	2
deterministic_policy	false	reward_dim	1
shared_rnn	true	device	cuda
layers_before_rnn	[]		
rnn_hidden_dim	64		
layers_after_rnn	[]		
rnn_cell_type	vanilla		
action_embed_dim	0		
state_embed_dim	0		
reward_embed_dim	0		
policy_net_activation_function	relu-tanh		
policy_net_initialization_method	normc		
action_pred_type	bernoulli		
hidden_noise_std	0.002		
policy_gamma	0.95		

384 The model fitting uses the public github methods from the Allen Institute for Neural Dynamics to fit
385 an MLE to the action sequences of the nmRNN agent.

```

3861 forager = ForagerCollection().get_preset_forager("Bari2019", seed=42)
3872 forager.fit(
3883     choice_history,
3894     reward_history,
3905     fit_bounds_override={"softmax_inverse_temperature": [0, 100]},
3916     clamp_params={"biasL": 0},

```

```

3927     DE_kwargs=dict(workers=4, disp=True, seed=np.random.default_rng
393         (42)),
3948     k_fold_cross_validation=None,
3959 )

```

396 7 Multitasking Benchmark Details

397 We evaluated the e-nmRNN on a suite of neuroscience-inspired multitasking benchmarks, specifically
398 drawing tasks from the Yang et al. (2019) suite and Driscoll et al. (2024) which cover a wide array of
399 decision-making cognitive tasks extensively studied in neuroscience literature. This benchmark suite
400 is designed to probe flexible computation and generalization across diverse cognitive demands, mirror-
401 ing challenges faced by biological neural networks. The tasks include, but are not limited to, various
402 forms of context-dependent integration, working memory, and decision-making paradigms. Each
403 task within the suite presents unique computational demands, requiring the network to dynamically
404 reconfigure its processing capabilities. Performance across these tasks is measured by a common
405 metric, typically accuracy or a similar performance indicator, enabling direct comparison between
406 different model architectures. The specific task parameters, input/output structures, and success
407 criteria for each individual task in the suite are detailed in the publicly available code repository
408 (details in Section 10).

409 8 Detailed Analysis of Emergent Biological Structures

410 Our analysis of trained e-nmRNNs reveals several emergent properties that mirror observations in
411 neurobiology, providing insights into the model’s interpretability and biological plausibility.

412 8.1 Modularity and Cell Clustering

413 Training on multitasking benchmarks promotes the emergence of functionally specialized modules
414 within the e-nmRNN, as well as hierarchical cell clustering (Figure 5D). This modularity is supported
415 by two key observations: 1) the formation of "Sherringtonian circuits" among presynaptic partners [16,
416 17], and 2) the emergence of distinct, modular brain regions in biological systems. Our clustering
417 methods, detailed in Section 10, group neurons based on their connectivity patterns and activity
418 profiles, revealing a clear modular organization that aligns with functional specialization.

419 8.2 Cell-Type Specialization in Neuromodulator Release

420 We observe a striking specialization where individual units learn to predominantly release a single
421 type of neuromodulator (Figure 5E). This mirrors biological observations of distinct cell types, such
422 as dopamine or serotonin neurons, which primarily release a specific small molecule via volume
423 transmission. In our model, this specialization can be approximated by observing that the R matrix,
424 which defines the amount of each small molecule a neuron releases, tends to develop one-hot
425 columns. The T_{ijk} tensor further defines cell types by their postsynaptic sensitivity to specific small
426 molecules. This emergent property provides a concrete computational model for how diverse neuronal
427 components contribute to overall system function.

428 8.3 Timescale Separation in Neuromodulator Dynamics

429 The neuromodulator dynamics within the e-nmRNN spontaneously learn to evolve on distinct,
430 separated timescales (Figure 5F). Our autocorrelation analysis, performed on the time series of
431 individual neuromodulator concentrations, reveals a factor of approximately 8x difference between
432 the fastest and slowest evolving neuromodulators. This heterogeneity in timescales reflects the
433 value of functional specialization, allowing different neuromodulators to encode information at
434 different temporal resolutions, which is consistent with biological observations regarding brain state
435 and behavior. Details of the autocorrelation analysis, including calculation methods and statistical
436 evaluation, are provided in Section 10.

8.4 Context Encoding within Neuromodulator Dynamics

Beyond compositional generalization, the neuromodulator state $\vec{n}(t)$ robustly learns to represent task context (Figure 5C). In experiments involving dynamically changing environments, such as the block-switched two-armed bandit task (Figure 4A), the NM concentrations dynamically track the hidden block state (Figure 4C, D). This learned context encoding represents a compelling computational hypothesis for animals trained across multiple tasks or contexts, especially with the advent of emerging, temporally-resolved neuromodulator imaging technologies.

8.5 Reward Prediction Error (RPE) Encoding in Reinforcement Learning

In reinforcement learning tasks, specifically the dynamic foraging task, the neuromodulator state not only encodes task context but also learns to represent critical learning signals such as the Reward Prediction Error (RPE) (Figure 4E, G). This is a key finding: it demonstrates how the e-nmRNN can leverage its neuromodulatory system to represent a critical learning signal, consistent with biological observations of dopamine activity correlating with RPE. This RPE encoding, combined with the $n_k(t)$ -gated 4-factor learning rule (Section 3.1), provides a concrete mechanism for RPE-gated online credit assignment, facilitating rapid adaptation to changes in the environment and contributing to the network’s ability to "learn how to learn" in a reinforcement learning context.

9 Experimental Details and Hyperparameters

This section details the experimental setups and hyperparameters used for all reported results, including compositional generalization, multitasking, and reinforcement learning tasks. All experiments were conducted in a cloud environment utilizing AWS, requiring less than 64 GB of RAM and a CUDA-enabled GPU for execution.

9.1 Sine Curve Meta-Learning

To assess the model’s ability to learn and generalize across a family of structured relationships, we employed a simple meta-learning sequence generation task. Each input-target sequence pair is sampled from a family of sine curves defined by three parameters: amplitude (A_i), frequency (ω_i), and phase (ϕ_i). The target sequence is given by:

$$y^*(t) = A_i \sin(\omega_i t + \phi_i)$$

The input to the network at each time step is a three-dimensional vector containing these parameters:

$$I(t) = \begin{pmatrix} A_i \\ \omega_i \\ \phi_i \end{pmatrix}$$

Importantly, due to the infinite size of this dataset, we opt to operate in a continual learning regime for this optimization where each sample is both the first and last time the network sees that exact sample. For this reason, we report results not in terms of number of times through the dataset, but instead, number of batched gradient update steps.

The challenge of learning to pair the input parameters with the output sequence should not be underestimated. Indeed, if we sample from the full, uniform distribution of parameters, the networks (all model classes) routinely fail to learn to oscillate. Thus, to focus on operating on both sides of the Hopf Bifurcation, we focused on a fixed distribution on ω and ϕ to sample from $A_i = \max(\mathcal{U}[-1, 1], 0)$.

We began our training with a TPESampler hyperparameter tuning. Our objective function is the minimization of the held-out loss after 1000 batched gradient update steps.

Next, we applied these locally optimal learning rates to each model class and trained the networks across samples. Batch sizes are held fixed ($B = 16$) across all models to ensure that number of batched gradient updates are compared on equal footing. We find that under these conditions, the e-nmRNN takes many fewer update steps to achieve high performance across this domain and find that the exponentially weighted spatially embedded RNN improve batch update efficiently when paired with the e-nmRNN dynamics. This phenomena, combined with the interpretability advantages of a spatially-embedded network, motivate our continued use throughout this work.

9.2 Compositional generalization

This task extends the concept of matrix multiplication to a setting where sequences are mapped to the amplitudes of other sequences. Instead of treating inputs and outputs as static vectors or matrices, we view them as temporal structures, allowing for:

- **Representation Learning:** The input sequences encode a structured representation, and the network must infer a decomposable basis from these representations.
- **Function Approximation:** The network learns to reconstruct an output sequence from a set of basis functions.
- **Compositional Generalization:** If trained correctly, the model should generalize to new sequence compositions beyond its training set.

Mathematical Formulation of the Task

We define the **input-output mapping** as:

$$I(t) = \sum_k a_k \mu_k(t)$$
$$y^*(t) = \sum_k a_k \eta_k(t)$$

where:

- $\mu_k(t)$ are the input **basis functions**.
- $\eta_k(t)$ are the output **basis functions**.
- a_k are the unknown **coefficients** that we want the model to infer from the input sequence and use that inference to generate the associated patterns in the corresponding weights.

The model must learn a function f_θ that maps input sequences to output sequences by implicitly recovering a_k :

$$f_\theta(I(t)) \approx y^*(t)$$

Generalization & Zero-Shot Performance

To test **generalization**, we evaluate the model on unseen compositions of input sequences:

1. Train the model on a fixed set of basis functions.
2. Introduce new linear combinations of basis functions during testing.
3. Measure performance on reconstructing previously unseen compositions.

If the model generalizes well, this suggests it has learned an effective representation space that allows for compositional inference.

One signature of learning this representation well is the ability to decode (using a weakly expressive decoder) the actual coefficients of each sequence from the network's hidden units (or nm fields in the case of the e-nmRNN). We add this analysis to our study of the capacity to better understand the failure to generalize with growing dataset complexity.

Two types of scaling

One of the great strengths of this simple task set is that we can scale both the complexity of the data set and the amount of data with the only cost being compute time. This allows us to ask how do each of the models selected for baseline characterizations scale with:

1. Scaling with the amount of data available to train the model allows us to test for the "efficiency" of the representation learning
2. Scaling the complexity (by way the number of basis patterns used in the composition, K) allows us to measure the practical capacity of the model.

In the work, we explore both of these scaling relationships but only report the results for the capacity due to lack of space. Future work will explore an analytic explanation of these numerical experiments.

Reproducibility parameters for compositional generalization

For the multitasking benchmarks, all models were parameter count matched at approximately 50k trainable parameters and tuned on compositional generalization (K=8). Hyperparameter optimization was performed using Tree-Structured Parzen Sampling with 100 samples for all models. Models were trained for 1000 epochs, and the best variance explained on the validation dataset was extracted for performance evaluation. Performance scaling with "Task complexity" (K ranging from 1 to 20) was explored.

Table 3: Key Hyperparameters for Compositional Generalization Task

Parameter	Value/Range (TPE)	Notes
Learning Rate	$[10^{-4}, 10^{-2}]$	Optimized via TPE sampler
Optimizer	Adam	Fixed
Batch Size	16	Fixed across all models
Hidden Units (RNN/GRU)	256/128	Parameter-matched
Neurons (e-nmRNN)	128	Parameter-matched
Number of NMs	4	Fixed
NM time constant (τ_n)	$[0.1, 10]$	Optimized
Weight Regularization (L2)	$[10^{-6}, 10^{-4}]$	Optimized
Activity Regularization (L2)	0	Fixed
Activation Function	ReLU-Tanh	Fixed

9.3 Multitasking Benchmarks

On the 20 Yang19 tasks, we conducted a single study in the manuscript: what does the solution space high performing e-nmRNN look like? Do we observe biologically aligned phenomena simply by training the nmRNN on a sufficiently rich set of tasks as was seen in [18, 19, 20]

Table 4: Key Hyperparameters for Multitasking Benchmarks

Parameter	Value/Range (TPE)	Notes
Learning Rate	$[5 \times 10^{-5}, 5 \times 10^{-3}]$	Optimized via TPE sampler
Optimizer	Adam	Fixed
Batch Size	32	Fixed
Hidden Units	224	Parameter-matched (50k parameters total)
Number of Neurons	128	Fixed
Number of NMs	4	Fixed
NM time constant (τ_n)	$[0.01, 10]$	Optimized
Regularization (L2)	$[10^{-7}, 10^{-5}]$	Optimized
Activation Function	ReLU-Tanh	Fixed
Dale’s Law	Enabled	For spatially embedded networks
Spatial Embedding Lengthscale	10% of domain	For spatially embedded networks

10 Reproducibility Details

To ensure full reproducibility of all experimental results presented in this paper, we will provide access to the codebase, including training scripts, evaluation procedures, and data generation methods. The code will be made available on a GitHub repository post acceptance, and a link for is provided for submission review. Additionally, we plan to provide associated Docker instances hosted on CodeOcean to facilitate ready and consistent reproducibility across different computing environments.

The code repository will include:

536 • **Source Code:** All Python scripts for defining the e-nmRNN architecture, baseline models
537 (Vanilla RNN, GRU, Transformers, HiPPO-LegT, Oscillatory RNNs), training loops, and
538 evaluation metrics.

539 • **Environment Configuration:** We provide a `Dockerfile` environment specification detail-
540 ing all required software libraries and their exact versions (e.g., PyTorch, NumPy, Matplotlib,
541 Optuna, gymnasium).

542 • **Data Access and Generation:** Scripts for generating the synthetic datasets used in the com-
543 positional generalization tasks (sine curves and dictionary mapping). For the multitasking
544 benchmarks and RL tasks, instructions or links to access the generated datasets are provided,
545 along with preprocessing scripts.

546 • **Hyperparameter Configurations:** Detailed configuration files for the optimal hyperparam-
547 eters found during the TPE sampler optimization for each experiment, complementing the
548 tables in Section 9.

549 • **Reproducible Commands:** Clear instructions, including exact command-line arguments, to
550 execute the training and evaluation runs for all results presented in the main text. We present
551 these with saved output as jupyter notebooks for ready inspection.

552 • **Summary of code structure:** please see below for a detailed summary of code structure
553 focusing on the minimal path toward reproducing notebooks and environment initialization
554 (denoted by a ★)

555 We commit to releasing the assets under a CC-BY 4.0 license. All aspects related to anonymity, as
556 per NeurIPS guidelines, will be strictly adhered to in the publicly released version of the repository.

557 The code can be found in a zip file at the following anonymous link with the following code structure.
558 Please note that we highlight only the notebooks to run to access the demos but have attempted to
559 provide a complete and blinded version of the code needed to run it.

560 • ANON_Final_NeurIPS_submission_repo

561 ★ REPRODUCING.md

562 - Fig2 - Hopf MetaLearning

563 ★ Dockerfile

564 - v3_refactor : The full code needed to reproduce gradient alignment

565 ★ Run the self-contained notebook: `Refactor_CosineSimilarityOverTraining.ipynb`

566 • Folder also includes analysis files of pre-trained logs

567 - Fig3 - Compositional Generalization

568 ★ Environment/Dockerfile

569 ★ Hyperparameter tuning can be recreated by running

570 `hyperparameter tuning-Demo.ipynb`

571 ★ To train all models in `model.py` across the task with as-

572 cending complexity run [caution: 24 hours execution time]:

573 `GenerateComplexitySweep_MatchedRun.ipynb`

574 - Fig4 - RL Bandit

575 - PPO_train_nmRNN

576 ★ environments/Dockerfile

577 • Folder provides access to full repo and some trained example networks.

578 ★ To train a new e-nmRNN agent please specify local paths and run [warning

579 execution time 24+ hours]: `TrainNotebook.ipynb`

580 ★ To regenerate the dynamics pictures of figure 4 on a new network, please run:

581 `AnalysisOfTrained_nmRNN.ipynb`

582 - MLE_Fits

583 ★ environments/Dockerfile

584 ★ To reproduce the MLE fits and correlation analysis on a previously generated

585 agent sequence please run: `Notebooks/Fit_nmRNN_Agent_Analysis.ipynb`

586 - Fig5 - Multitasking

587 ★ Yang19MultitaskingAll

588 ★ environment/Dockerfile
589 ★ To train a new network please run [execution time 48+ hours]
590 Demo_SpatialNetwork-Copy3.ipynb
591 ★ To analyze a new network please run
592 Analysis_spatialNetwork-Alltasks.ipynb
593 - AdaptatibilityNeurogym A repo not used in the paper to study the adaptability
594 of the nmRNN. This repo is in development but welcomes comments, corrections,
595 and contributions.

596 Note: The code is provided for review in its current most robust form. Further code revisions are
597 underway to merge the numerous environments while avoiding dependency conflicts.

598 References

- 599 [1] Allison Cruikshank et al. “Dynamical questions in volume transmission”. In: *Journal of*
600 *Biological Dynamics* 17.1 (Dec. 2023). ISSN: 1751-3758. DOI: 10.1080/17513758.2023.
601 2269986. (Visited on 05/08/2025).
- 602 [2] Sarthak Mittal et al. “Explicit Knowledge Factorization Meets In-Context Learning: What Do
603 We Gain?” en. In: May 2024. URL: <https://openreview.net/forum?id=DoFtSA1qIB>
604 (visited on 05/08/2025).
- 605 [3] David Ha, Andrew Dai, and Quoc V. Le. *HyperNetworks*. arXiv:1609.09106. Dec. 2016. DOI:
606 10.48550/arXiv.1609.09106. URL: <http://arxiv.org/abs/1609.09106> (visited on
607 10/21/2024).
- 608 [4] Julia C. Costacurta et al. *Structured flexibility in recurrent neural networks via neuromodulation*.
609 en. Pages: 2024.07.26.605315 Section: New Results. July 2024. DOI: 10.1101/2024.07.26.
610 605315. URL: <https://www.biorxiv.org/content/10.1101/2024.07.26.605315v1>
611 (visited on 10/21/2024).
- 612 [5] Johannes von Oswald et al. *Continual learning with hypernetworks*. arXiv:1906.00695. Apr.
613 2022. DOI: 10.48550/arXiv.1906.00695. URL: <http://arxiv.org/abs/1906.00695>
614 (visited on 10/21/2024).
- 615 [6] Benjamin Ehret et al. *Continual Learning in Recurrent Neural Networks*. arXiv:2006.12109.
616 Mar. 2021. DOI: 10.48550/arXiv.2006.12109. URL: [http://arxiv.org/abs/2006.](http://arxiv.org/abs/2006.12109)
617 12109 (visited on 10/21/2024).
- 618 [7] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. *Linear Transformers Are Secretly Fast*
619 *Weight Programmers*. arXiv:2102.11174. June 2021. DOI: 10.48550/arXiv.2102.11174.
620 URL: <http://arxiv.org/abs/2102.11174> (visited on 10/21/2024).
- 621 [8] Steven L. Brunton et al. *Modern Koopman Theory for Dynamical Systems*. arXiv:2102.12086.
622 Oct. 2021. DOI: 10.48550/arXiv.2102.12086. URL: [http://arxiv.org/abs/2102.](http://arxiv.org/abs/2102.12086)
623 12086 (visited on 10/21/2024).
- 624 [9] Michael T. Pearce et al. “Bilinear MLPs enable weight-based mechanistic interpretability”.
625 en. In: Oct. 2024. URL: <https://openreview.net/forum?id=gI0kPk1UKS> (visited on
626 05/08/2025).
- 627 [10] Guillaume Bellec et al. “A solution to the learning dilemma for recurrent networks of spiking
628 neurons”. en. In: *Nature Communications* 11.1 (July 2020). Publisher: Nature Publishing
629 Group, p. 3625. ISSN: 2041-1723. DOI: 10.1038/s41467-020-17236-y. URL: <https://www.nature.com/articles/s41467-020-17236-y> (visited on 10/21/2024).
- 631 [11] Blake A. Richards et al. “A deep learning framework for neuroscience”. en. In: *Nature*
632 *Neuroscience* 22.11 (Nov. 2019). Publisher: Nature Publishing Group, pp. 1761–1770. ISSN:
633 1546-1726. DOI: 10.1038/s41593-019-0520-2. URL: [https://www.nature.com/](https://www.nature.com/articles/s41593-019-0520-2)
634 [articles/s41593-019-0520-2](https://www.nature.com/articles/s41593-019-0520-2) (visited on 05/08/2025).
- 635 [12] James M Murray. “Local online learning in recurrent networks with random feedback”. In:
636 *eLife* 8 (May 2019). Ed. by Peter Latham, Michael J Frank, and Brian DePasquale. Publisher:
637 eLife Sciences Publications, Ltd, e43299. ISSN: 2050-084X. DOI: 10.7554/eLife.43299.
638 URL: <https://doi.org/10.7554/eLife.43299> (visited on 10/21/2024).
- 639 [13] Marcin Andrychowicz et al. *Learning to learn by gradient descent by gradient descent*.
640 arXiv:1606.04474. Nov. 2016. DOI: 10.48550/arXiv.1606.04474. URL: [http://arxiv.](http://arxiv.org/abs/1606.04474)
641 [org/abs/1606.04474](http://arxiv.org/abs/1606.04474) (visited on 10/21/2024).

- 642 [14] Johannes von Oswald et al. *Transformers learn in-context by gradient descent*.
643 arXiv:2212.07677. May 2023. DOI: 10.48550/arXiv.2212.07677. URL: <http://arxiv.org/abs/2212.07677> (visited on 10/21/2024).
644
- 645 [15] Arthur Pellegrino, N. Alex Cayco Gajic, and Angus Chadwick. “Low Tensor Rank Learning
646 of Neural Dynamics”. en. In: Nov. 2023. URL: <https://openreview.net/forum?id=WcoX8eJJjI> (visited on 05/08/2025).
647
- 648 [16] David L. Barack and John W. Krakauer. “Two views on the cognitive brain”. en. In: *Nature*
649 *Reviews Neuroscience* 22.6 (June 2021). Publisher: Nature Publishing Group, pp. 359–371.
650 ISSN: 1471-0048. DOI: 10.1038/s41583-021-00448-6. URL: <https://www.nature.com/articles/s41583-021-00448-6> (visited on 05/15/2025).
651
- 652 [17] Casey M. Schneider-Mizell et al. “Inhibitory specificity from a connectomic census of mouse
653 visual cortex”. en. In: *Nature* 640.8058 (Apr. 2025). Publisher: Nature Publishing Group,
654 pp. 448–458. ISSN: 1476-4687. DOI: 10.1038/s41586-024-07780-8. URL: <https://www.nature.com/articles/s41586-024-07780-8> (visited on 05/15/2025).
655
- 656 [18] Guangyu Robert Yang et al. “Task representations in neural networks trained to perform
657 many cognitive tasks”. eng. In: *Nature Neuroscience* 22.2 (Feb. 2019), pp. 297–306. ISSN:
658 1546-1726. DOI: 10.1038/s41593-018-0310-2.
- 659 [19] Mikail Khona et al. “Winning the Lottery With Neural Connectivity Constraints: Faster
660 Learning Across Cognitive Tasks With Spatially Constrained Sparse RNNs”. In: *Neural*
661 *Computation* 35.11 (Oct. 2023), pp. 1850–1869. ISSN: 0899-7667. DOI: 10.1162/neco_a_01613.
662 URL: https://doi.org/10.1162/neco_a_01613 (visited on 05/06/2025).
- 663 [20] Laura N. Driscoll, Krishna Shenoy, and David Sussillo. “Flexible multitask computation
664 in recurrent networks utilizes shared dynamical motifs”. en. In: *Nature Neuroscience* 27.7
665 (July 2024). Publisher: Nature Publishing Group, pp. 1349–1363. ISSN: 1546-1726. DOI:
666 10.1038/s41593-024-01668-6. URL: <https://www.nature.com/articles/s41593-024-01668-6> (visited on 10/21/2024).
667