Leveraging Dale's Principle as an Inductive Bias in Recurrent Neural Networks

Jiyi Wang[1] **Chenxiao Yang**[2] **Yujia Zhao**[1] **Jingzhao Zhang**[2] **Lu Mi**[2*]
[1]Shanghai Qi Zhi Institute, Shanghai, China [2]Tsinghua University, Beijing, China

Abstract

One of the fundamental differences between biological neural networks and artificial neural networks (ANNs) is that biological neurons obey Dale's principle: each neuron projects either exclusively excitatory or exclusively inhibitory outputs. While this constraint is central to biological circuits, previous attempts to impose Dale's principle in ANNs have often led to training instability or degraded performance. In this work, we introduce EISep RNN, a recurrent neural network architecture that implements Dale's principle through a learnable excitatory–inhibitory separation. Across a series of learning settings, including multi-task learning, continual learning, and reinforcement learning, EISep achieves more stable training dynamics, competitive or improved performance, and sparser, more modular connectivity than vanilla RNNs and previous Dale-constrained models. EISep also exhibits stable and informative latent dynamics. These results establish Dale's principle as a biologically grounded and computationally effective inductive bias for recurrent neural networks. Code is available at `https://github.com/NeuroAIHub/EISep`.

1 Introduction

Neuroscience and artificial intelligence share a long history of cross-disciplinary research, where breakthroughs in one field routinely drive progress in the other [1, 2, 3, 4, 5, 6, 7, 8, 9]. A prominent direction in NeuroAI is to translate biological principles into inductive biases through architectural constraints or learning rules [10, 11, 12]. For example, explicitly encouraging modularity can improve generalization in reinforcement learning [13], imposing synaptic diversity can improve stability and robustness in vision models [14], and biologically inspired memory-regulation mechanisms can alleviate catastrophic forgetting [15]. Apart from improving artificial intelligence (AI), these inductive biases also produce new tools for neuroscience: networks that not only perform well, but do so in a manner more consistent with how real neural circuits operate. One potentially underexplored inductive bias is Dale's principle, which we investigate in this paper.

Dale's principle states that, each biological neuron is either excitatory or inhibitory, and only projects either positive *or* negative weights to all downstream neurons (Fig. 1B left). This reflects the fact that a neuron releases the same type of neurotransmitter at all its synapses [16, 17]. In contrast,

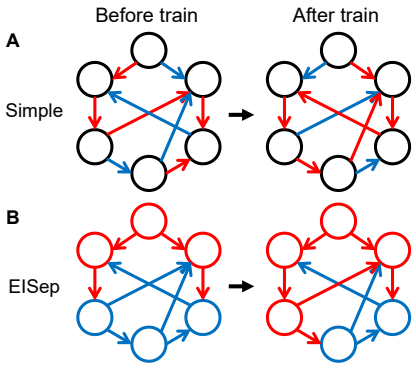


Figure 1: **A.** In Simple RNN, each neuron can project positive (red) *and* negative (blue) weights. **B.** In EISep RNN, each neuron can only project positive weights (excitatory), *or* negative weights (inhibitory). This sign is learnable (left to right) during training.

*Corresponding author: `milu@mail.tsinghua.edu.cn`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

artificial neural networks (ANNs) typically allow each unit to project both positive *and* negative weights (Fig. 1A left). The lack of excitatory-inhibitory separation simplifies optimization but deviates from biological organization. Previous work has suggested that enforcing Dale's principle may reduce network capacity [18], narrow the loss basin [14] and impair learning performance [19, 20]. As a result, most ANNs have continued to use unconstrained units, prioritizing trainability over biological realism, and leaving Dale-constrained networks largely underexplored in practice.

In this work, we introduce EISep RNNs – a new recurrent neural network architecture that implements Dale's principle as an inductive bias. EISep RNNs naturally develop modular structure, sparse connectivity, and stable dynamics. Crucially, these structural and dynamical benefits translate into improved performance in multi-task learning, continual learning and reinforcement learning, showing that biological realism and effective learning are not necessarily contradictory.

2 Related work

In computational neuroscience literature, Dale's principle has become a widely adopted assumption to study the neural dynamics [21, 22, 23, 24, 25, 26, 27]. Beyond its use in models of biological circuits, several studies have incorporated E-I separation into trainable neural networks. For example, [28] trained E–I RNNs on cognitive tasks and reproduced features of population activities, while [29] investigated emergent E-I structural differences. Other work developed trainable Dale-constrained ANNs on general supervised learning tasks [19], used spectral analysis to improve their learning performance [20], and combined Dale-constrained learning with pruning [18]. Together, these studies establish that Dale-constrained networks can support effective learning and provide useful models of neural circuits.

From a machine learning perspective, requiring each neuron to have outgoing weights of a single sign restricts the available connectivity patterns at a fixed hidden size. Several early attempts to impose E-I separation revealed substantial difficulties. Simply constraining networks' neurons to fixed excitatory or inhibitory identities often narrows the solution space and makes optimization harder, leading to degraded performance and slower or unstable training [14, 30].These observations motivated a series of recent works that revisit Dale's principle and propose architectures to recover performance while respecting sign constraints [28, 19, 20, 18]. While effective in certain scenarios, such designs are often sensitive to task complexity and optimization details. As a result, Dale's principle is frequently viewed as a constraint rather than an advantage. Our work challenges this prevailing view and reveals that Dale's principle can actually act as a beneficial inductive bias for RNNs.

3 Methods

3.1 Background and Notation

We first introduce the following notations for RNN operations. Consider an RNN with hidden size N, activation function ϕ, recurrent weight W, input weight U, output weight V, and bias terms b_h, b_o. At time step t, it receives input x_t, maintains hidden state h_t, and produces output o_t:

$$h_t = \phi(Wh_{t-1} + Ux_t + b_h), \quad o_t = Vh_t + b_o. \tag{1}$$

Enforcing Dale's principle during learning is nontrivial. Simply initializing neurons as excitatory or inhibitory is insufficient, since unconstrained optimization will flip the weight signs over time. A common approach [28, 20, 18] is to parameterize the recurrent weight matrix as $W = QD$, where Q is a learnable non-negative matrix, and D is a fixed diagonal sign matrix with $D_{ii} = \pm 1$. The non-negativity of Q is typically enforced via gradient masking (zeroing out violating updates) or by applying non-negative transformations to the parameters. We briefly review prior Dale-constrained ANNs in Table 1, with additional details provided in Appendix E.

Despite their efforts, current Dale-constrained networks still struggle to match the performance of vanilla RNN [30]. We attribute this to two key limitations. **(1) Fixed neuron identity.** The sign matrix D is fixed after initialization, preventing the network from adapting excitatory and inhibitory types during training, which constrains the optimization trajectory. **(2) Initialization.** To enforce non-negativity, they sample weights from gamma [28] or exponential [19, 20] distributions, which can induce a large initial spectral radius and may harm training stability.

To address these concerns, we make two adjustments in EISep. **(1)** We make the sign matrix D learnable, allowing neuron types to adapt during training (Fig. 1B), which enables the network to explore different E/I organizations and follow more flexible optimization trajectories. **(2)** We sample weights from a rectified Gaussian distribution which better controls the initial spectral radius. We verify the roles of these two designs in the ablation study (Sec. 6). Next, we give the formal definition of EISep.

Table 1: Comparison of initialization methods and E-I sign learnability across Dale-constrained ANNs. Rect. Gauss. denotes a rectified Gaussian distribution.

Model	Initialization	E-I sign
Simple	Gaussian	–
ColEI [20]	Exponential	fixed
DBPN [18]	Uniform	fixed
DANN [19]	Exponential	fixed
EISep (ours)	Rect. Gauss.	learnable

3.2 Definition of EISep

We introduce a learnable sign assignment using the Gumbel-Softmax (GSoftmax) reparameterization [31]. Let $M \in \mathbb{R}^{N\times 2}$ be a learnable parameter matrix. For each neuron i, we sample a soft sign mask $m_i \in \mathbb{R}^2$ as $m_i = \mathrm{GSoftmax}(M_i), \quad i = 1, ..., N$, and define a soft sign value $e_i = m_i[0] - m_i[1]$. We then construct a diagonal soft sign matrix D with

$$D_{ij} = \begin{cases} e_i, & i = j, \\ 0, & i \neq j. \end{cases} \tag{2}$$

The soft sign value e_i is concentrated around ± 1 during training and exactly equals ± 1 during test (Appendix F.1). This parametrization enables end-to-end gradient-based optimization, while allowing neuron identities to adapt during training. At test time, the sign mask is discretized to remove sampling noise by setting $e_i = 1$ if $M_i[0] > M_i[1]$, and -1 otherwise.

Let $W_0 \in \mathbb{R}^{N\times N}$ be a learnable parameter matrix with entries initialized as $W_{0,ij} \sim \mathcal{N}(0, 1/N)$. The recurrent weight W is defined as:

$$W = \mathrm{ReLU}(W_0)D. \tag{3}$$

Under this parameterization, the value matrix $Q = \mathrm{ReLU}(W_0)$ is initialized using rectified Gaussian distribution, which leads to a smaller initial spectral radius (Fig. 37), and is associated with improved training stability.

To encourage stable neuron identities during later training stages, we introduce an entropy regularization term $L_e = \frac{1}{N}\sum_i H(\mathrm{softmax}(M_i))$, where $H(p) = -\sum_j p_j \log p_j$, and optimize $L = L_t + \lambda_e L_e$. This completes the definition of the EISep RNN.

3.3 Baselines

We compare EISep against a vanilla RNN baseline (denoted *Simple*) as well as several Dale-constrained architectures, including DANN and ColEI. As our experiments use the Adam optimizer, while the original implementations of DANN and ColEI only employ stochastic gradient descent (SGD), we additionally implement Adam-based variants of these models to ensure a fair comparison. We denote these baselines as DANN-SGD, DANN-Adam, ColEI-SGD, and ColEI-Adam.

4 Experiments on cognitive tasks

4.1 Performance

We first evaluate EISep on a multi-task cognitive benchmark widely used to study multi-task capacity and representations of RNNs [32, 33]. The benchmark consists of 20 interrelated tasks that require a single RNN to flexibly switch between task rules across trials. In each trial, the network receives a rule input indicating the active task, followed by noisy sensory stimuli that provide evidence toward choices. The correct choice and stimulus strength are randomized on each trial, yielding a mixture of easy and difficult decisions. This benchmark tests working memory, perceptual decision-making and rule-based context switching in a single RNN. Additional details are provided in Appendix F and I.

Across the full task suite, EISep consistently achieves higher accuracy than prior Dale-constrained architectures and matches Simple's performance (Fig. 2A). This advantage is also evident when performance is averaged across tasks (Fig. 2B). These results demonstrate that our implementation achieves better performance than previous works, and that enforcing Dale's principle does not necessarily compromise task performance.

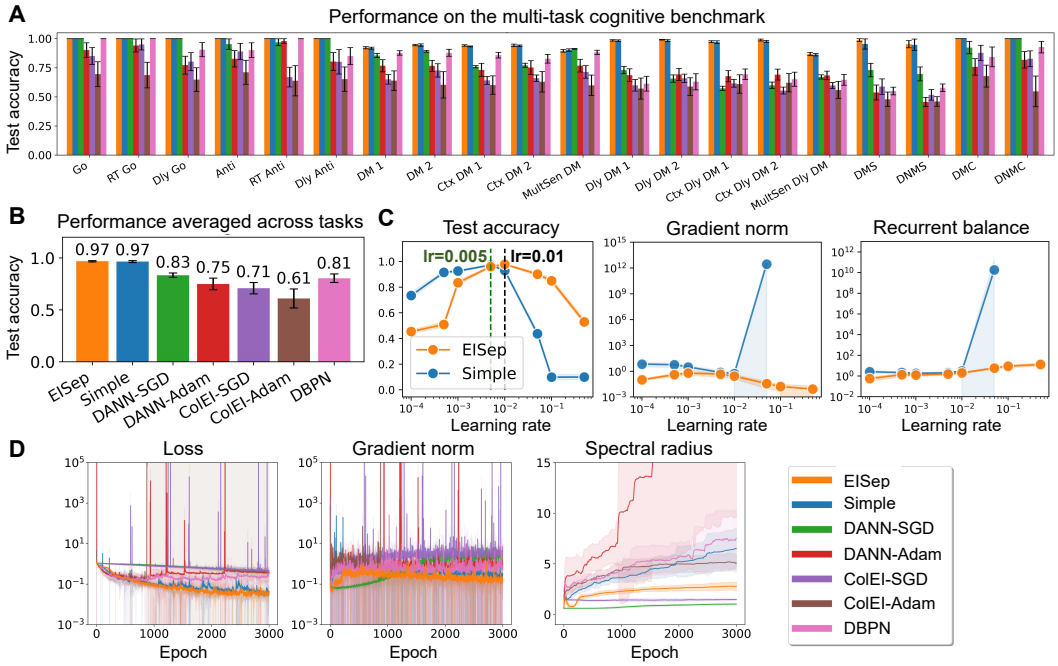


Figure 2: Multi-task cognitive benchmark consisting of 20 cognitive tasks. The same legend at the right of **D** is used throughout the figure. **A.** EISep consistently achieves higher accuracy than prior Dale-constrained baselines and matches Simple across tasks. **B.** EISep achieves the highest average performance. **C.** At high learning rates, EISep maintains higher test accuracy (left), lower gradient norms (middle), and tighter recurrent balance (right) than Simple. **D.** EISep exhibits lower decoding loss (left), lower gradient norms (middle), and more stable spectral radius trajectories (right) throughout training, indicating more stable dynamics. All models use the same learning rate 10^{-2} except for **C**.

4.2 Training stability

To evaluate training stability, we track the decoding loss, gradient norm, and spectral radius of the recurrent weight matrix throughout training (Fig. 2D). Most baseline models exhibit occasional spikes in loss and gradient norms, whereas EISep exhibits lower and more stable curves. Moreover, EISep achieves tighter control of spectral radius than most baselines. These results suggest EISep stabilizes training dynamics. When tested across different learning rates, EISep performs better at higher learning rates (Fig. 2C, left), which correlates with its lower gradient norms (Fig. 2C, middle).

In biological networks, the balance between excitatory and inhibitory inputs is known to stabilize dynamics and improve computational capacity [24]. To assess whether a similar mechanism emerges, we quantify recurrent balance as the average input to each neuron. We find that EISep achieves a tighter balance than Simple across learning rates (Fig. 2C, right). Overall, these results indicate that by enforcing a biologically realistic E-I balance, EISep keeps the spectral radius bounded, mitigates gradient explosion and promotes stable optimization. More metrics are provided in Appendix J.

To ensure the fairness of the comparison, we further test all baseline models across different learning rates (Appendix J). We find that EISep performs well across a relatively larger scope of learning rates. In other words, it can achieve a wider stable zone than most other baselines. Moreover, its stable zone resides at higher learning-rate areas, consistent with the stabilization effects induced by balanced recurrent dynamics.

4.3 Emergent properties

The above results suggest that EISep promotes stable training dynamics. We next investigate whether its architectural constraints also give rise to structured connectivity. Specifically, we examine two emergent structural properties: *modularity* and *sparsity*.

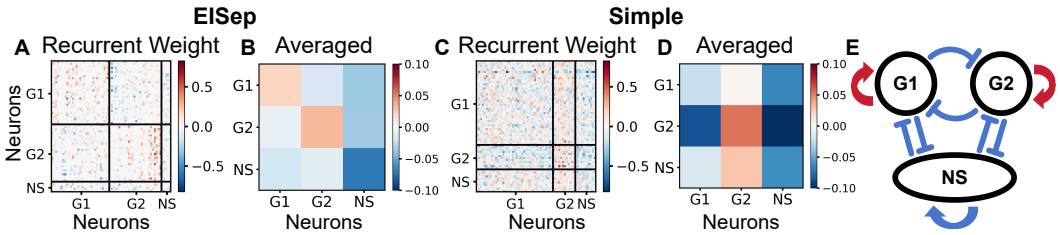


Figure 3: Case study on the 2AFC task. Neurons are grouped by selectivity using the signal-detection index d'. **A.** EISep exhibits a clear block structure in the recurrent weight W, with within-group excitation and cross-group inhibition. **B.** Averaged connectivity within neuron groups further reveals strong self-excitation and mutual inhibition. **C-D.** Simple lacks clear separation between intra-group and inter-group connectivity. **E.** The emergent EISep connectivity resembles the conceptual winner-take-all circuit commonly used in perceptual decision-making models [36].

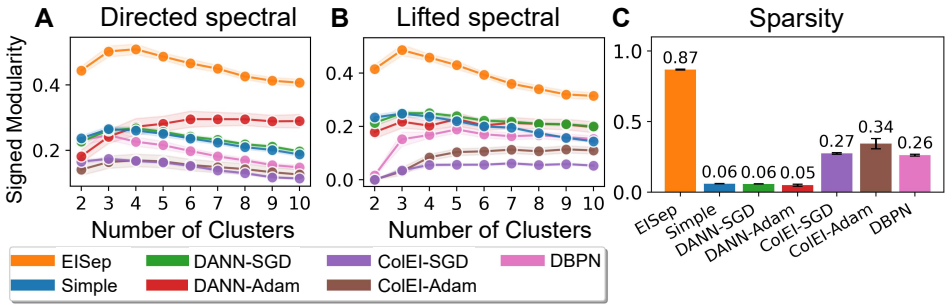


Figure 4: Population analysis on the multi-task cognitive benchmark. Neurons are partitioned into groups using two clustering methods. EISep achieves the highest modularity under (**A.**) directed spectral clustering and (**B.**) lifted spectral clustering, indicating more modular circuit organization. **C.** EISep also achieves substantially higher recurrent weight sparsity than baseline models.

4.3.1 Modularity

The neural circuits have functionally specialized modules to support efficient computation [34, 35]. To investigate whether Dale's principle promotes modular and interpretable network structure, we test EISep on a noisy two-alternative forced-choice (2AFC) perceptual decision-making task where the network must choose the stronger signal out of two noisy inputs. We use signal-detection index d' (Appendix K.1) to quantify the neuron selectivity and partition the neurons into Group 1 (G1), Group 2 (G2) and Non-selective (NS). EISep develops a clear block structure in recurrent weight W (Fig. 3A, B). Weights within groups are primarily excitatory, while weights between groups are primarily inhibitory. This produces two populations with strong self-excitation and cross-inhibition. This organization resembles a winner-take-all circuit model commonly adopted in perceptual decision-making studies (Fig. 3E, [36]). In contrast, Simple RNN has no clear distinction between intra-group and inter-group connections (Fig. 3C, D).

Next, we carry out population analysis to quantitatively examine the network modularity. We use spectral clustering methods to partition the neurons into groups, and use signed modularity (Appendix K.2) to measure if there are larger intra-group connections and smaller inter-group connections. EISep exhibits the highest modularity across both directed spectral clustering (Fig. 4A) and lifted spectral clustering (Fig. 4B). This comparison suggests that enforcing E-I separation biases the network towards modular and interpretable circuit motifs, whereas unconstrained RNNs rely on more distributed and less structured connectivity.

4.3.2 Sparsity

Sparsity is a hallmark of biological neural circuits, and is widely believed to support energy-efficient and structured computation in both neuroscience [37, 38] and AI [39, 40]. Here, we define sparsity as the fraction of small weights ($|w| < 0.1\sigma$), and provide the detailed definition and validity check in Appendix O. EISep achieves substantially higher sparsity than baseline models in multi-task cognitive

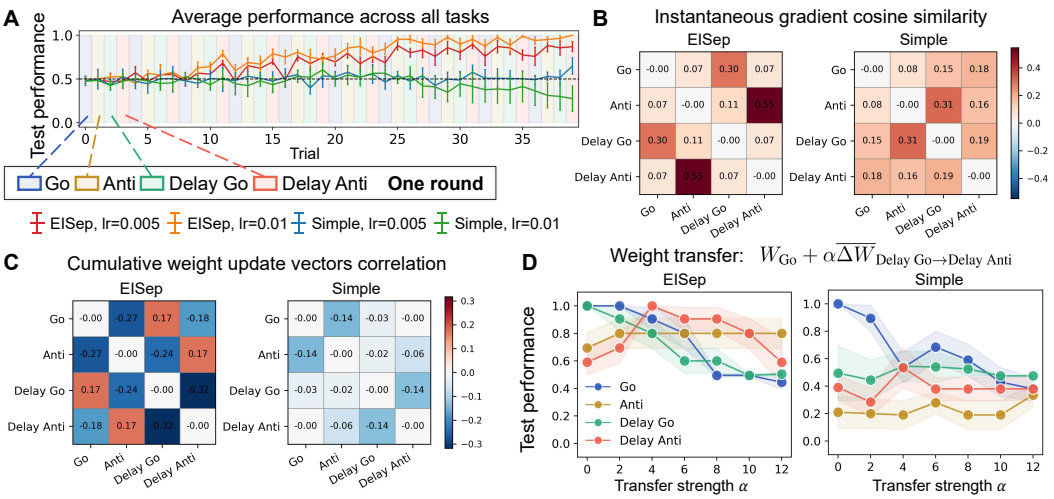


Figure 5: Continual learning on four sequentially presented cognitive tasks repeated over 10 rounds. **A.** EISep achieves higher performance and stronger resistance to catastrophic forgetting than Simple across learning rates. **B.** EISep exhibits high gradient similarity for similar tasks (*Go* vs. *Delay Go*) and near-zero similarity for opposing tasks (*Go* vs. *Anti*), indicating reduced gradient interference. **C.** EISep shows structured correlations between task-induced weight updates, with positive correlations for similar tasks and negative correlations for opposing tasks. **D.** EISep's performance shifts from *Go* to *Anti* as transfer strength α increases.

benchmark (Fig. 4C) and RL tasks (Appendix S). The increased sparsity induced by EISep suggests that E-I separation encourages more efficient connectivity, without compromising task performance.

5 Extension to challenging learning regimes

Given the computational and structural advantages, we test whether they translate into improved performance in challenging scenarios. Specifically, building on the modularity advantages (Sec. 4.3.1), we investigate whether Dale's principle helps resist gradient interference and mitigate catastrophic forgetting in continual learning. Building on the stability advantages (Sec. 4.2), we investigate whether it reduces gradient instability and improves performance in reinforcement learning.

5.1 Continual learning

Continual learning is a non-stationary problem where different tasks are presented sequentially to the network. The gradients from different tasks often interfere with each other, leading to unstable optimization and catastrophic forgetting [41, 42]. On the other hand, modular networks can mitigate this by allocating tasks to different subspaces, effectively reducing gradient interference [43].

We build a continual learning dataset using four tasks from the multi-task cognitive benchmark (Sec. 4.1): *Go*, *Anti*, *Delay Go*, and *Delay Anti*. In each trial, the agent receives a noisy two-channel stimulus and must select one channel. In the two Go tasks, the correct response is the stronger channel, whereas in the two Anti tasks, it is the weaker one. In the non-delay tasks, responses are required immediately right after the stimulus period, while the two delay tasks insert a silent memory period between stimulus and response, requiring the network to maintain internal representations in the absence of external input (Appendix I). Together, these tasks constitute a 2×2 design that disentangles sensory-response mapping (Go vs. Anti) and memory requirements (Delay vs. non-Delay). These tasks are presented sequentially for several times to the network. A similar setup has been used as a benchmark for continual learning algorithms [43].

We train each model using the optimal learning rates from the multi-task learning (Simple: 5×10^{-3}; EISep: 1×10^{-2}, Fig. 2C left). Performance on all four tasks is evaluated at every trial (Fig. 5A). At both learning rates, Simple rapidly forgets earlier tasks as training proceeds, while EISep performs better than Simple and other baselines (Appendix. L). It also exhibits smoother loss curves (Appendix L.2).

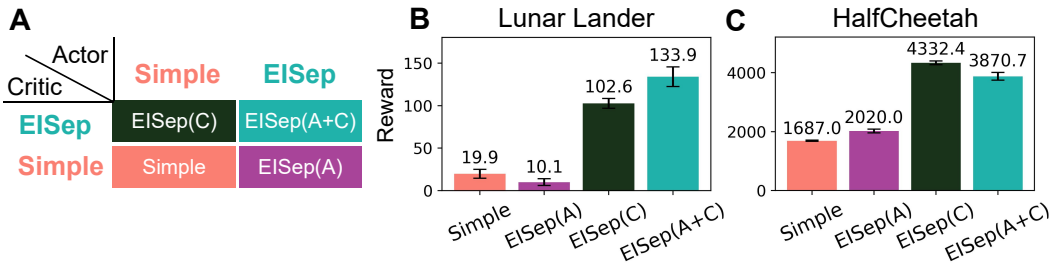


Figure 6: Reinforcement learning benchmarks using recurrent A2C agents. **A.** Four actor–critic variants combining Simple and EISep recurrent modules. **B.** On the discrete Lunar Lander task, EISep achieves higher rewards than Simple. **C.** On the more challenging HalfCheetah task, EISep again outperforms Simple, demonstrating improved optimization stability and control performance across discrete and continuous RL settings.

Next, we investigate whether EISep mitigates gradient interference. In continual learning, updates that improve the current task can impair the performance of previously learned tasks, causing catastrophic forgetting [44, 45]. We use the cosine similarity between task-specific gradients as a local diagnostic of the interference [45, 46]. Positive similarity indicates that an update can benefit both tasks; negative similarity indicates conflict; and near-zero similarity indicates orthogonal local update directions. We record the average gradients when training each task and compute the cosine similarity between task-specific gradients (Fig. 5B). EISep exhibits high similarity for tasks with identical stimulus–response mappings (e.g., *Go* vs. *Delay Go*) and near-zero similarity for opposing tasks (e.g., *Go* vs. *Anti*). In contrast, Simple shows comparable similarity for both types. These results suggest that EISep exhibits gradient alignment that reflects task relationships, with more aligned directions for tasks sharing stimulus–response mappings and approximately orthogonal directions for tasks with opposing mappings.

Finally, we examine whether EISep organizes task representations in a structured manner. For each task, we compute the cumulative weight update $\Delta W = W_{\text{after}} - W_{\text{before}}$, and measure correlations between these task-induced updates (Fig. 5C). Since this cumulative update integrates many steps of parameter updates, it probes the global organization of the weight transitions between tasks. In EISep, updates are positively correlated for same-type tasks and negatively correlated for opposing tasks, while Simple lacks such patterns. Although negative update correlations may appear inconsistent with near-zero gradient similarity, cumulative updates are also shaped by optimizer dynamics and training history [47], and are not necessarily aligned with instantaneous gradients.

To further probe the update geometry, we perform a weight-transfer experiment, analogous to task-vector arithmetic in [48]. If weight updates encode reusable task transformations, an update learned between one pair of tasks should induce the corresponding behavioral change when applied to another task with the same underlying relationship. Specifically, we normalize the update vector from *Delay Go* to *Delay Anti*: $\overline{\Delta W}_{\text{DG}\to\text{DA}} = \Delta W_{\text{DG}\to\text{DA}}/||\Delta W_{\text{DG}\to\text{DA}}||_F$, average it across trials, and add it to the weights after training on *Go*, $W_{\text{Go}} + \alpha\overline{\Delta W}_{\text{DG}\to\text{DA}}$. As α increases, performance shifts from *Go* toward *Anti* in EISep, but not in Simple (Fig. 5D). These results show that the weights of EISep lie in a structured subspace, which enables efficient parameter reuse and reduced interference, while Simple has more entangled weights.

5.2 Reinforcement learning

5.2.1 Performance

Reinforcement learning (RL) tests adaptive control in complex environments, and is well known to suffer from unstable optimization and high-variance gradient updates [49]. EISep's stable dynamics observed before may help alleviate this problem.

Building on the Advantage Actor–Critic (A2C [50]) framework, we evaluate EISep and Simple on two RL benchmarks: *Lunar Lander* [51], a discrete control task, and *HalfCheetah* [52], a more challenging continuous control task. To isolate architectural effects on actor and critic, we use separate recurrent modules for them. This yields four variants (Fig. 6A): Simple (Simple for both actor and critic), EISep(A) (EISep for the actor and Simple for the critic), EISep(C) (EISep only for

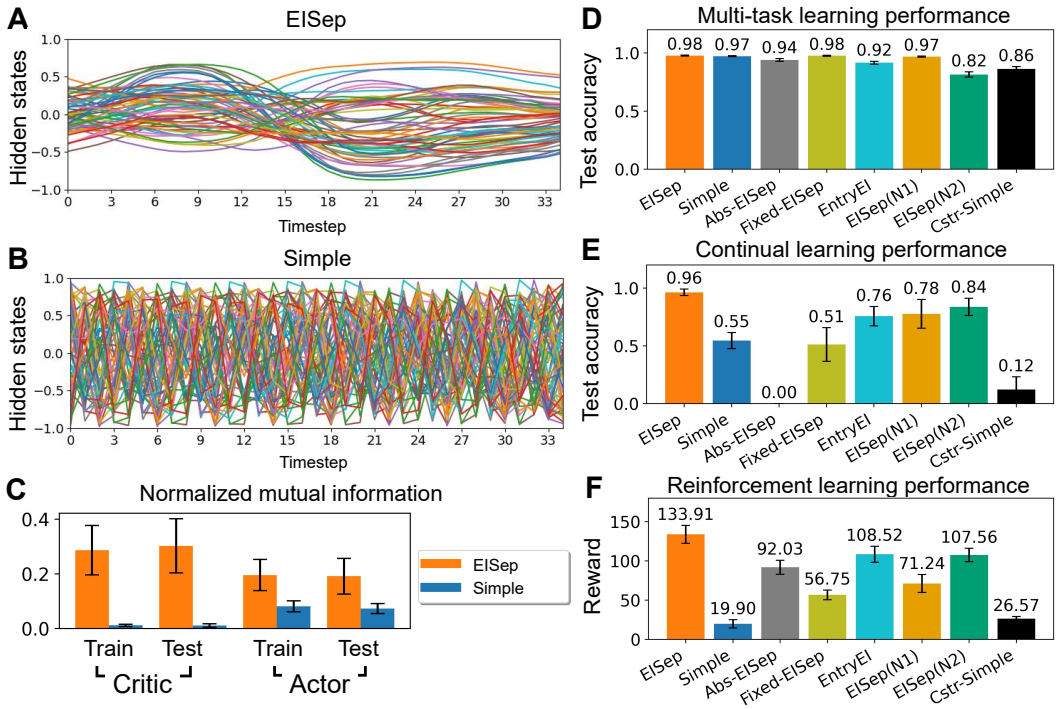


Figure 7: Hidden dynamics and ablation analysis. **A.** EISep exhibits smooth and diverse hidden dynamics during Lunar Lander control. **B.** Simple shows high-frequency repetitive jittering, suggesting low-dimensional limit-cycle collapse. **C.** EISep achieves substantially higher normalized mutual information between HMM latent states and manually annotated environment phases, indicating that its hidden dynamics encode richer environment information. **D-F.** Across multi-task learning (**D**), continual learning (**E**), and reinforcement learning (**F**), EISep consistently matches or outperforms ablation models, highlighting the importance of initialization method, learnable signs, biologically grounded E-I separation, and the ReLU parameterization.

the critic), and EISep(A+C) (EISep for both). To accelerate training in HalfCheetah, we upgrade the backbone from vanilla RNN to LSTM (Appendix R). We report the best performance across actor and critic learning rates for each variant. Full results are provided in Appendix P.2.

On both tasks, EISep achieves higher rewards than Simple (Fig. 6B, C), and other Dale-constrained baselines (Appendix P.1), showing its stability advantage generalizes across different task settings and backbones. Notably, EISep also improves the performance of feedforward networks (Appendix P.3), indicating that EISep can alleviate gradient instability not only from RNN iteration, but also from the RL task itself. We discuss the architectural effects on actor and critic in Appendix V.

5.2.2 Internal dynamics

To examine how E-I separation shapes internal dynamics, we visualize one clip of the actor's hidden states from a test episode in Lunar Lander (Fig. 7A-C). This clip is taken from EISep(A+C) and Simple trained with learning rate lr=0.005 for critics and lr=0.001 for actors, where both agents can achieve higher performance than chance (Table 4). EISep exhibits smoother and more diverse hidden dynamics (Fig. 7A), while Simple's dynamics show high-frequency and repetitive jittering (Fig. 7B), suggesting a collapse to low-dimensional limit cycles. Longer clips are provided in Appendix T.

To quantify how much internal dynamics encode environment information, we perform the following calculation: (1) fit the hidden trajectories using Hidden Markov Models (HMMs), and assign latent labels; (2) manually annotate phases using rule-based criteria (Appendix Q); (3) calculate the normalized mutual information (NMI) between unsupervised HMM labels and rule-based phases labels. EISep achieves substantially higher NMI than Simple for both actor and critic (Fig. 7C), indicating that its latent dynamics are not only temporally smoother, but also more informative about the environment. In this way, EISep can support complex and robust control in challenging scenarios (Fig. 6), whereas Simple tends toward latent collapse and inferior performance.

In summary, EISep produces more stable optimization and better performance in continual learning and reinforcement learning problems. We also include traditional RNN tasks such as Sequential MNIST in Appendix M, where EISep can still outperform most Dale-constrained baselines. We discuss the importance of selecting appropriate benchmarks in evaluating inductive biases in Sec. 7.

6 Ablation study

Finally, we present an ablation study to disentangle the contributions of individual design choices (Fig. 7D-F). We briefly review each variant here and provide the details in Appendix U. For multi-task learning, we report the best performance across learning rates. For continual learning, we report average performance in the final round when all four tasks are trained for the last time. For reinforcement learning, we report the best performance among variants on Lunar Lander. In Appendix G, we test the EISep's sensitivity to hyperparameters including hidden size, temperature, entropy loss coefficient, activation function, and training order. In Appendix H, we examine how ongoing sign plasticity affects performance and whether the performance can be explained by conventional sparsity regularization alone.

To assess the role of the ReLU operator (Eq. 3), we replace it with an absolute value operator: $W = |W_0|D$ and denote the new variant as **Abs-EISep**. To assess the benefit of learnable signs (Sec. 3.1), we fix the sign matrix D after initialization, resulting in **Fixed-EISep**. To study the effect of initialization (Sec. 3.1), we apply ColEI-style (**EISep(N1)**) and DBPN-style (**EISep(N2)**) initialization to EISep (Appendix E). We exclude DANN due to its fundamentally different parameterization with three coupled matrices (Appendix E). To test if EISep's performance completely comes from its initialization method alone, we apply EISep's initialization method to Simple and denote the constrained-initialized version as **Cstr-Simple**. We summarize these models in Fig. 32.

Compared to EISep, these ablation models exhibit performance degradation to varying degrees, from minimal drops in simple multi-task learning (Fig. 7D), to severe declines in the challenging RL task (Fig. 7F). These results highlight the importance of the design choices in Sec. 3.2, including learnable signs, initialization method and ReLU operator. It also suggests that EISep's success arises from their combined effect rather than any single component alone.

Another question is whether EISep's advantages arise from *neuron-level* E-I separation induced by Dale's principle, or more generally from value-sign separation. Therefore, we introduce **EntryEI**, which assigns an independent sign constraint to each weight: $W = \mathrm{ReLU}(W_0) \odot K, K_{ij} \in \{-1, 1\}$, where K is a dense matrix sampled using Gumbel-Softmax reparameterization as in Sec. 3.2, and $\odot$ denotes element-wise multiplication. This modification increases parameter size by allowing each weight to independently choose its sign. However, EntryEI underperforms EISep on all tasks (Fig. 7D-F). This result highlights that a biologically realistic inductive bias can outperform a more flexible but less realistic alternative, even when the latter has a larger hypothesis space.

7 Conclusion & discussion

We introduced EISep RNNs, a recurrent neural network architecture that enforces Dale's principle through a learnable separation between excitatory and inhibitory neurons. Across a range of settings, including multi-task cognitive benchmark, continual learning, and reinforcement learning, EISep matches or exceeds the performance of Simple and other Dale-constrained variants, while exhibiting more stable optimization. Our analyses further reveal EISep induces brain-like sparse and modular structure, and promotes diverse, informative dynamics. These findings highlight a broader lesson for NeuroAI: biologically inspired inductive biases can simultaneously enhance performance, stability, and interpretability in artificial neural networks.

EISep also has several limitations that may motivate future work. First, EISep exhibits reduced performance at lower learning rates (Fig. 2C left), which reveals a trade-off between stability and convergence speed. Future research can explore adaptive methods to expand its stable regime (Fig. 17). Although we have shown in Appendix G.1 that EISep benefits more from scaling, it is still unclear whether EISep's advantages persist in even larger models (like LLM) and more challenging problems. Finally, there are many biological details that we do not attempt to reproduce in this work. For example, the ratio of excitatory to inhibitory neurons is typically around 4:1 [53], and long-range projections are predominantly excitatory [54]. However, our goal is not to faithfully replicate the

brain, but to investigate why biological systems are effective, by systematically testing biologically inspired inductive biases in ANNs. We expect future work along this direction to further bridge neuroscience and AI. Appendix D discusses the implications for computational neuroscience and biological plausibility.

We would like to emphasize that our contribution is not limited to proposing a new RNN architecture, but also a systematic evaluation of biological inductive biases in biologically relevant learning scenarios, which may be challenging and underexplored in traditional RNN research. As shown in Appendix M, when tested on vision or algorithmic tasks where gradient instability or interference is not a significant issue, the advantage of EISep over Simple is much weaker, consistent with previous research. This result reveals an optimization–expressivity trade-off. Compared with vanilla RNN, EISep has many restrictions. (1) It constrains all outgoing connections from a neuron to share a sign. (2) It keeps the value matrix nonnegative by a ReLU filter. (3) It also has high sparsity that reduces the effective connections even further. These restrictions can act as dynamical regularization. In stationary, low-variance supervised problems, optimization is already relatively stable, so this additional regularization may provide little benefit while still limiting the capacity. However, the regularization becomes more beneficial in non-stationary or high-variance modules where gradient instability and interference are substantial. This also suggests that the value of biological inductive biases is inherently task-dependent, and systematic evaluation is essential to identify where these inductive biases provide the greatest benefits in machine learning.

Acknowledgement

The work is supported by the Shanghai Qi Zhi Institute Innovation Program SQZ202501. We thank Chengrui Li for insightful discussions that helped shape the ideas presented in this work.

References

[1] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[2] Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[3] James CR Whittington, Joseph Warren, and Timothy EJ Behrens. Relating transformers to models and neural representations of the hippocampal formation. *arXiv preprint arXiv:2112.04035*, 2021.

[4] Ziming Liu, Eric Gan, and Max Tegmark. Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *Entropy*, 26(1):41, 2023.

[5] Dong Kyum Kim, Jea Kwon, Meeyoung Cha, and Chul Lee. Transformer as a hippocampal memory consolidation model based on nmdar-inspired nonlinearity. *Advances in Neural Information Processing Systems*, 36:14637–14664, 2023.

[6] Tianqin Li, Ziqi Wen, Yangfan Li, and Tai Sing Lee. Emergence of shape bias in convolutional neural networks through activation sparsity. *Advances in Neural Information Processing Systems*, 36:71755–71766, 2023.

[7] Anthony Zador, Sean Escola, Blake Richards, Bence Ölveczky, Yoshua Bengio, Kwabena Boahen, Matthew Botvinick, Dmitri Chklovskii, Anne Churchland, Claudia Clopath, et al. Catalyzing next-generation artificial intelligence through neuroai. *Nature communications*, 14(1):1597, 2023.

[8] Mengping Jiang, Linfan Gu, Mingyi Ma, Qin Li, Jonathan C Kao, and Weizhe Hong. Neural basis of cooperative behavior in biological and artificial intelligence systems. *Science*, page eadw8151, 2025.

[9] Luke Darlow, Ciaran Regan, Sebastian Risi, Jeffrey Seely, and Llion Jones. Continuous thought machines. *arXiv preprint arXiv:2505.05522*, 2025.

[10] Mackenzie Weygandt Mathis. Leveraging insights from neuroscience to build adaptive artificial intelligence. *Nature Neuroscience*, pages 1–12, 2025.

[11] Mahdi Ramadan, Cheng Tang, Nicholas Watters, and Mehrdad Jazayeri. Computational basis of hierarchical and counterfactual information processing. *Nature Human Behaviour*, pages 1–15, 2025.

[12] Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

[13] Ruiyi Zhang, Xaq Pitkow, and Dora E Angelaki. Inductive biases of neural network modularity in spatial navigation. *Science Advances*, 10(29):eadk1256, 2024.

[14] Martin Hofmann, Moritz Franz Peter Becker, Christian Tetzlaff, and Patrick Mäder. Concept transfer of synaptic diversity from biological to artificial neural networks. *Nature communications*, 16(1):5112, 2025.

[15] Liyuan Wang, Xingxing Zhang, Qian Li, Mingtian Zhang, Hang Su, Jun Zhu, and Yi Zhong. Incorporating neuro-inspired adaptability for continual learning in artificial intelligence. *Nature Machine Intelligence*, 5(12):1356–1368, 2023.

[16] Henry Dale. Pharmacology and nerve-endings, 1935.

[17] John Carew Eccles. From electrical to chemical transmission in the central nervous system: the closing address of the sir henry dale centennial symposium cambridge, 19 september 1975. *Notes and records of the Royal Society of London*, 30(2):219–230, 1976.

[18] Aishwarya Balwani, Alex Q Wang, Farzaneh Najafi, and Hannah Choi. Constructing biologically constrained rnns via dale's backpropagation and topologically informed pruning. *Science Advances*, 11(50):eadw4970, 2025.

[19] Jonathan Cornford, Damjan Kalajdzievski, Marco Leite, Amélie Lamarquette, Dimitri M Kullmann, and Blake Richards. Learning to live with dale's principle: Anns with separate excitatory and inhibitory units. *BioRxiv*, pages 2020–11, 2020.

[20] Pingsheng Li, Jonathan Cornford, Arna Ghosh, and Blake Richards. Learning better with dale's law: A spectral perspective. *Advances in Neural Information Processing Systems*, 36:944–956, 2023.

[21] Carl van Vreeswijk and Haim Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–1726, 1996.

[22] Carl van Vreeswijk and Haim Sompolinsky. Chaotic balanced state in a model of cortical circuits. *Neural computation*, 10(6):1321–1371, 1998.

[23] Junhao Liang, Zhuda Yang, and Changsong Zhou. Excitation–inhibition balance, neural criticality, and activities in neuronal circuits. *The Neuroscientist*, 31(1):31–46, 2025.

[24] Sophie Denève and Christian K Machens. Efficient codes and balanced networks. *Nature neuroscience*, 19(3):375–382, 2016.

[25] Gengshuo Tian, Shangyang Li, Tiejun Huang, and Si Wu. Excitation-inhibition balanced neural networks for fast signal detection. *Frontiers in Computational Neuroscience*, 14:79, 2020.

[26] Sukbin Lim and Mark S Goldman. Balanced cortical microcircuitry for spatial working memory based on corrective feedback control. *Journal of Neuroscience*, 34(20):6790–6806, 2014.

[27] Alessandro Ingrosso and LF Abbott. Training dynamically balanced excitatory-inhibitory networks. *PloS one*, 14(8):e0220547, 2019.

[28] H Francis Song, Guangyu R Yang, and Xiao-Jing Wang. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS computational biology*, 12(2):e1004792, 2016.

[29] Sun Minni, Li Ji-An, Theodore Moskovitz, Grace Lindsay, Kenneth Miller, Mario Dipoppa, and Guangyu Robert Yang. Understanding the functional and structural differences across excitatory and inhibitory neurons. *bioRxiv*, page 680439, 2019.

[30] Veronica Chelu and Doina Precup. Excitatory-inhibitory dynamics in adaptive decision-making. In *International Conference on Neural Information Processing*, pages 65–79. Springer, 2025.

[31] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[32] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

[33] Laura N Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, 27(7):1349–1363, 2024.

[34] Stephen Grossberg. The complementary brain: Unifying brain dynamics and modularity. *Trends in cognitive sciences*, 4(6):233–246, 2000.

[35] Maxwell A Bertolero, BT Thomas Yeo, and Mark D'Esposito. The modular and integrative functional architecture of the human brain. *Proceedings of the National Academy of Sciences*, 112(49):E6798–E6807, 2015.

[36] Kong-Fatt Wong and Xiao-Jing Wang. A recurrent network mechanism of time integration in perceptual decisions. *Journal of Neuroscience*, 26(4):1314–1328, 2006.

[37] Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and LF Abbott. Optimal degrees of synaptic connectivity. *Neuron*, 93(5):1153–1164, 2017.

[38] Rieke Fruengel and Marcel Oberlaender. Sparse connectivity enables efficient information processing in cortex-like artificial neural networks. *Frontiers in Neural Circuits*, 19:1528309, 2025.

[39] Brian Bartoldson, Ari Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *Advances in neural information processing systems*, 33:20852–20864, 2020.

[40] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[41] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[42] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

[43] Lea Duncker, Laura Driscoll, Krishna V Shenoy, Maneesh Sahani, and David Sussillo. Organizing recurrent network dynamics by task-computation to enable continual learning. *Advances in neural information processing systems*, 33:14387–14397, 2020.

[44] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[45] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[46] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[48] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[49] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[50] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

[51] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[52] Paweł Wawrzyński. A cat-like robot real-time learning to run. In *International Conference on Adaptive and Natural Computing Algorithms*, pages 380–390. Springer, 2009.

[53] Qingyang Wang, Albert Cardona, Marta Zlatic, Joshua T Vogelstein, and Carey E Priebe. Why do we have so many excitatory neurons? *bioRxiv*, pages 2024–09, 2024.

[54] Armen Stepanyants, Luis M Martinez, Alex S Ferecskó, and Zoltán F Kisvárday. The fractions of short-and long-range connections in the visual cortex. *Proceedings of the National Academy of Sciences*, 106(9):3555–3560, 2009.

[55] Josef Ladenbauer, Sam McKenzie, Daniel Fine English, Olivier Hagens, and Srdjan Ostojic. Inferring and validating mechanistic models of neural microcircuits based on spike-train data. *Nature communications*, 10(1):4933, 2019.

[56] Laura N Borodinsky, Cory M Root, Julia A Cronin, Sharon B Sann, Xiaonan Gu, and Nicholas C Spitzer. Activity-dependent homeostatic specification of transmitter expression in embryonic neurons. *Nature*, 429(6991):523–530, 2004.

[57] Enrico Cherubini, Jean L Gaiarsa, and Yehezkel Ben-Ari. Gaba: an excitatory transmitter in early postnatal life. *Trends in neurosciences*, 14(12):515–519, 1991.

[58] Zizhen Yao, Cindy TJ Van Velthoven, Michael Kunst, Meng Zhang, Delissa McMillen, Changkyu Lee, Won Jung, Jeff Goldy, Aliya Abdelhak, Matthew Aitken, et al. A high-resolution transcriptomic and spatial atlas of cell types in the whole mouse brain. *Nature*, 624(7991):317–332, 2023.

[59] Peter Y Wang, Yi Sun, Richard Axel, LF Abbott, and Guangyu Robert Yang. Evolving the olfactory system with machine learning. *Neuron*, 109(23):3879–3892, 2021.

[60] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[61] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

A Impact statement

Applications. This work contributes to foundational NeuroAI research by investigating how a core biological principle—neuron-level excitatory–inhibitory (E-I) separation—acts as an inductive bias in recurrent neural networks. These properties may benefit machine learning applications that rely on sequential reasoning and control, such as robotics, adaptive control systems, scientific modeling, and cognitive modeling, where stable training and interpretable dynamics are important. These properties may also support computational neuroscience for studying excitation–inhibition balance, by providing mechanistic frameworks that are both biologically realistic and computationally effective.

Implications. On the AI side, improved optimization stability and interpretability may increase the reliability of recurrent models for sequential decision-making, supporting more predictable and controllable behavior in downstream systems. On the neuroscience side, E–I–separated models offer mechanistic testbeds for studying neural computation and complementing experimental approaches. This work does not introduce new data, sensing, or surveillance capabilities and therefore does not raise immediate concerns related to privacy or fairness. However, as increased robustness can amplify both beneficial and harmful uses, the ultimate impact of such methods will depend on their deployment context.

Initiatives. On the AI side, future work could investigate how biologically inspired constraints interact with fairness, safety, and interpretability in applied settings, as well as how such constraints scale to larger or more complex models. On the neuroscience side, promising directions include integrating E–I–separated architectures with biophysical or spiking neuron models, comparing learned dynamics with neural recordings, and developing benchmarks for evaluating biological plausibility in machine learning systems. More broadly, this work supports an interdisciplinary research agenda in which insights from neuroscience inform machine learning design, promoting the interaction between artificial intelligence and brain science that benefits both fields and society.

B Reproducibility statement

We release the full training and evaluation code in the supplementary material to support independent replication of our experiments. The appendix contains a complete specification of all hyperparameters, providing sufficient information to reproduce the reported results.

C Use of large language models

In preparing this manuscript, we used large language models (LLMs) as a writing aid only, primarily to assist with text editing, organization, and clarity. Specifically, LLMs were used to help improve readability and flow, refine grammar and style, and suggest high-level organizational structures for sections of the paper.

All scientific contributions—including the formulation of the research questions, model design, experimental setup, implementation, analysis, and interpretation of results—were conceived, conducted, and validated entirely by the authors. LLMs were not used to generate original technical ideas, algorithms, code, experimental results, proofs, or numerical values. Any suggested references or rephrasings were independently verified by the authors before inclusion.

The authors take full responsibility for the content of the paper and include this statement solely for transparency regarding the limited use of generative AI tools.

D Connection to neuroscience

D.1 Implications for computational neuroscience

Here we talk about the neuroscience implications. We do not intend to argue that EISep should universally replace vanilla RNNs. Rather, the appropriate model depends on the scientific question.

In other RNN studies [32, 33], a unit in their networks can have both positive and negative outgoing weights. It cannot be directly interpreted as a biological neuron, but rather as a population of neurons. Therefore, vanilla RNNs are effective for identifying task representations and dynamical motifs in

a neuron population without differentiation, but may need caution when asked what the roles of different cell types are in implementing a dynamical motif. Therefore, EISep enables the following analyses. (1) Learned circuits can be compared more directly with experimentally observed cell-type-specific connectivity and neural activity. (2) E/I-specific lesions and perturbations can be performed explicitly.

Fig. 3 provides a very simple example of this framework. EISep develops two stimulus-selective groups with within-group excitation and cross-group inhibition, revealing a winner-take-all circuit motif. We wish this network could help computational neuroscientists (1) understand the normative role of E/I separation better, and (2) develop cell-type-specific models to fit the neuronal population data for more powerful neuron signal modeling (like [55]).

The additional parameter cost is modest: EISep adds an $N \times 2$ logit matrix, which scales as $O(N)$, whereas the recurrent matrix scales as $O(N^2)$. And from Fig. 8, a medium EISep network ($n = 64$) is often better than larger vanilla RNNs. Therefore, EISep helps us to save parameters rather than wasting them. Nevertheless, as we stress in the discussion and Appendix M, EISep's constraints can be disadvantageous when E/I organization is unrelated to the scientific question. We therefore recommend using EISep when E/I-specific mechanisms are central to the hypothesis, while retaining vanilla RNNs as an important unconstrained baseline.

D.2 Biological plausibility

There remains a question regarding the biological plausibility of EISep. In related NeuroAI literature [32], neural system development and machine learning training are roughly compared on the concept level. The neural network training may compress multiple biological processes, including evolutionary selection, developmental differentiation, and experience-dependent learning. Therefore, training EISep can be roughly interpreted as during an early "developmental" stage, the learnable signs allow the model to search over possible E/I organizations, while the entropy regularization encourages a stable and discrete organization. The "differentiated" neurons can then learn the task at a later "experience-driven learning" stage.

However, this analogy is only conceptual, and does not indicate that sign flips happen universally in biological neural systems, although there exist some similar phenomena. In fact, the effective sign of a neuron depends on both its transmitters and its postsynaptic environment. At an early period, the neurotransmitters could change (glutamate GABA [56]), and the postsynaptic environment could also be different (to make GABA excitatory [57]). Coexpression of excitatory and inhibitory transmitters also exists [58]. Our work is just a simplified conceptual abstraction. The real process may be more complicated, but our work provides a model for asking how task demands and circuit organization could jointly shape the eventual allocation of excitatory and inhibitory populations.

E Baseline initialization

In this part, we introduce some initialization details of different baselines models mentioned in Sec. 3.1.

ColEI [20] adopts an explicit value-sign separation $W = QD$. The initialization details are as follows.

Given an E/I ratio r, the numbers of excitatory neurons n_E and inhibitory neurons n_I are set to

$$n_E = \frac{r}{r+1}N, \quad n_I = \frac{1}{r+1}N. \tag{4}$$

The rate parameters λ_E, λ_I are chosen as

$$\lambda_E = 1/\sqrt{\frac{2\pi - 1}{2\pi}Nr}, \tag{5}$$

$$\lambda_I = r\lambda_E \tag{6}$$

For the sampling process, we first sample weights w for excitatory neurons w^E and inhibitory neurons w^I

$$w_{i,j}^E \sim \mathrm{Exp}(\lambda_E), \tag{7}$$

$$w_{i,j}^I \sim \mathrm{Exp}(\lambda_I). \tag{8}$$

Then the diagonal entries are set to zero: $w_{ii} = 0$. The next step is to renormalize the spectral radius to 1.5:

$$w = w * 1.5/\max(|k|), \tag{9}$$

where k is the eigenvalue of w. Finally an absolute-value operation is applied to obtain the value matrix $Q = |w|$.

The sign matrix satisfies $D_{ii}^E = 1$ for excitatory neurons, and $D_{jj}^I = -1$ for inhibitory neurons. The final recurrent weight is constructed as

$$W = QD. \tag{10}$$

Non-negativity of Q is enforced by clipping $Q \leftarrow \mathrm{ReLU}(Q)$ at every update.

DBPN [18] enforces Dale's principle through projection-based updates rather than an explicit sign matrix. During training, outgoing weights of excitatory neurons are projected to zero whenever they become negative (and analogously for inhibitory neurons). Although no explicit D is used, their approach is effectively equivalent to a value–sign decomposition in which the value matrix is clipped to non-negative $Q \leftarrow \mathrm{ReLU}(Q)$ at every update, with an implicit fixed sign matrix D.

The initial weights are sampled from a uniform distribution:

$$Q_{i,j} \sim \mathcal{U}(0, 1/\sqrt{N}) \tag{11}$$

DANN [19] enforces Dale's principle by explicitly separating excitatory and inhibitory pathways rather than factorizing a single recurrent matrix. The recurrent weight is parameterized as $W = W^{EE} - W^{EI}W^{IE}$. Given the hidden size N, the initialization adopts the following rules:

$$W_{i,j}^{EE} \sim \mathrm{Exp}(\frac{1}{\sqrt{3(N-1)}}), \tag{12}$$

$$W_{i,j}^{IE} = \mathrm{mean}(W_{i,j}^{EE}), \tag{13}$$

$$W_{i,j}^{EI} \sim \mathrm{Exp}(\sqrt{\frac{10}{3(N-1)}}). \tag{14}$$

All three matrices are constrained to be non-negative via clipping after each update.

F Hyperparameter and training details

All recurrent networks use $N = 64$ hidden units and are trained with backpropagation through time (BPTT).

F.1 Gumbel-Softmax sampling

For the soft Gumbel-Softmax used in Sec. 3.2, let $u_{i,k} \sim \mathcal{U}(0,1)$ independently and $g_{i,k} = -\log(-\log u_{i,k})$. Then

$$m_{i,k} = \frac{\exp((M_{i,k} + g_{i,k})/\tau)}{\sum_{\ell=0}^{1} \exp((M_{i,\ell} + g_{i,\ell})/\tau)}, \tag{15}$$

$$e_i = m_{i,0} - m_{i,1} = \tanh\left(\frac{(M_{i,0} + g_{i,0}) - (M_{i,1} + g_{i,1})}{2\tau}\right). \tag{16}$$

With `hard=False`, e_i is in $(-1, 1)$ during training. Temperature $\tau = 0.1$ is fixed without annealing, and the entropy penalty encourages concentrated identities. Intermediate values occur when the two

perturbed logits are close. At test time, the larger unperturbed logit determines the discrete sign, with ties assigned -1 as in Sec. 3.2.

Even if a soft value e_i is not exactly ± 1 and could change during training, it does not violate Dale's principle. Because $W = \mathrm{ReLU}(W_0)D$, all outgoing weights of neuron i are the results of non-negative values multiplied by the same sign scalar e_i. They therefore remain either all non-negative or all non-positive in every forward pass.

F.2 Multi-task Cognitive Benchmark

Task setup. For each architecture, a single recurrent network is trained to perform all 20 cognitive tasks using supervised learning, with the ground-truth choice as the training target. We run 10 random seeds and report mean and standard error (SEM $= \sigma/\sqrt{N}$) across seeds. Models are tested for 100 epochs after training.

Network initialization.

- Activation function: Softplus.
- Recurrent bias: $b_{W,i} \sim \mathcal{N}(0, 1/N)$.
- Input and output weights (U, V): orthogonal initialization; biases $(b_U, b_V) \sim \mathcal{N}(0, 1/N)$.
- E–I logit matrix: $M_{i,j} \sim \mathcal{N}(0, 1/N)$.
- Gumbel-Softmax temperature: 0.1 (no annealing).

Optimization.

- Sequence length: 100.
- Batch size: 32.
- Training epochs: 3000.
- Optimizer: Adam.
- Loss: cross-entropy.
- Regularization: none, except E–I logit entropy loss (Sec. 3.2) with coefficient $\lambda_e = 1$.
- Output weighting: fixation output weight 0.05, decision outputs weight 1 (following the original benchmark).

Multi–learning-rate sweeps. For learning-rate stability experiments (e.g., Fig. 2C), we use 5 seeds and sweep learning rates:

$$\{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}, 0.5\}.$$

F.3 Continual Learning

Task protocol. We use 5 random seeds. In one complete round, each of the four tasks is trained for 200 epochs in the order *Go - Anti - Delay Go - Delay Anti*. The full task round is repeated 10 times, resulting in 40 task trials. The task rule is provided at every trial to avoid ambiguity. After training on each task, models are tested for 200 epochs on all four tasks. We report the standard error (SEM $= \sigma/\sqrt{N}$) across seeds on the plot as the error bar.

Optimization details. All network architecture and optimization settings follow Sec. F.2.

F.4 Reinforcement Learning

Environments. We use `LunarLander-v3` and `HalfCheetah-v5` from Gymnasium v1.1.1. Maximum episode length is 500 steps for LunarLander and unbounded for HalfCheetah.

Data collection.

- Parallel environments: $n_p = 10$ (LunarLander), $n_p = 100$ (HalfCheetah).
- Steps per update: $n_s = 128$ per environment.
- Number of updates: $n_u = 1000$.
- Batch size: $n_s \times n_p$.

Optimization and losses.

- Advantage estimation: Generalized Advantage Estimation (GAE).
- Critic loss: Smooth L1 loss $L_1^s(V_t, [A_t + V_t])$, where A_t is the advantage term and $[\cdot]$ indicates the gradient of $A_t + V_t$ is detached from the computational graph.
- Actor loss: advantage-normalized policy gradient, which means the advantage A_t is normalized in each batch.
- Separate optimizers and learning rates for actor and critic.
- Regularization: none except E–I logit entropy loss with coefficient $\lambda_e = 1$.

RL hyperparameters.

- Discount factor: $\gamma = 0.99$.
- GAE parameter: $\lambda = 0.95$.
- Entropy coefficient: 0.01.
- The ranges of learning rates of actor: $[3 \times 10^{-4}, 1 \times 10^{-3}, 3 \times 10^{-3}]$
- The ranges of learning rates of critic: $[1.5 \times 10^{-3}, 5 \times 10^{-3}, 1.5 \times 10^{-2}]$

Evaluation. We use 5 random seeds and evaluate each model on 20 episodes per seed. We report the standard error across episodes and seeds (SEM $= \sigma/\sqrt{N}$) on the plot as the error bar.

F.5 Computation resource and execution time

Hardware. All experiments were conducted on a workstation equipped with an AMD Ryzen 7 7840H CPU, 16 GB RAM, and an NVIDIA GeForce RTX 4060 Laptop GPU (8 GB VRAM). The GPU driver version is 566.14 with CUDA 12.7 support.

Software. Experiments were implemented in PyTorch with standard CUDA acceleration. Full environment details are provided in the Appendix.

Per-experiment cost. Multi-task learning: $\tilde{1}$ hour for 5 seeds ($\tilde{1}2$ minutes per run) Continual learning: $\tilde{1}.5$ hours for 5 seeds ($\tilde{1}8$ minutes per run) Reinforcement learning: $\tilde{1}0$ hours for 5 seeds, 9 learning rates, and 3 variants, including EI(A), EI(C), EI(A+C) ($\tilde{6}$ minutes per run)

Total compute. Across all experiments, including hyperparameter tuning and ablations, the total compute is estimated to be on the order of tens of GPU-hours.

Additional details. All models were trained from scratch without pretraining. Experiments can be reproduced on a single consumer-grade GPU with similar memory capacity.

F.6 Licenses for External Assets

We use standard benchmark environments and datasets, including:

- **Neurogym cognitive task suite** [32] License: Apache License 2.0. Copyright 2020–2026, Manuel Molano, Guangyu Robert Yang, Jorge F. Mejias, and contributors; additional contributions from Giulia Crocioni et al. URL: `https://github.com/neurogym/neurogym`

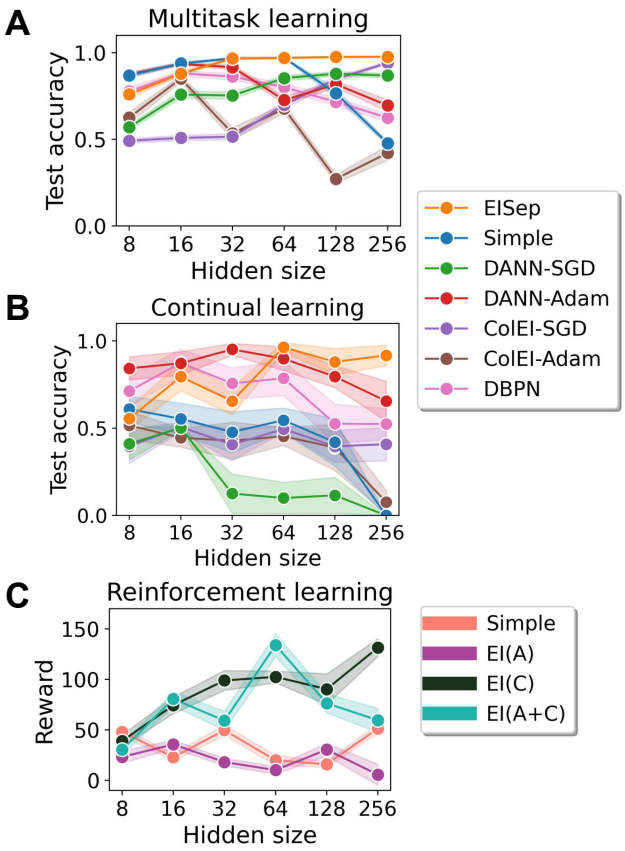


Figure 8: Under different hidden sizes, the performance in multi-task learning (**A**), continual learning (**B**) and reinforcement learning (**C**).

.

- **Gymnasium environments (e.g., LunarLander, HalfCheetah)** [51] License: MIT License. Copyright 2016 OpenAI; Copyright 2022 Farama Foundation. URL: `https://github.com/Farama-Foundation/Gymnasium`
- **MuJoCo (for HalfCheetah)** [52] License: Source code is licensed under the Apache License 2.0. Copyright 2021 DeepMind Technologies Limited; additional components Copyright 2016 Svetoslav Kolev. Documentation is licensed under CC BY 4.0. URL: `https://github.com/google-deepmind/mujoco`

All assets are used in accordance with their respective licenses. No datasets containing personal or sensitive information are used.

G Hyperparameter sensitivity

G.1 Hidden size

To test the sensitivity of different models to the hidden size, we evaluated model performance across hidden sizes (Fig. 8). Compared to the default setting (64), EISep remains robust as the hidden size increases, while other models tend to overfit and degrade at larger sizes. Moreover, the performance gap between EISep and Simple widens with increasing model size, indicating that EISep benefits more from scaling.

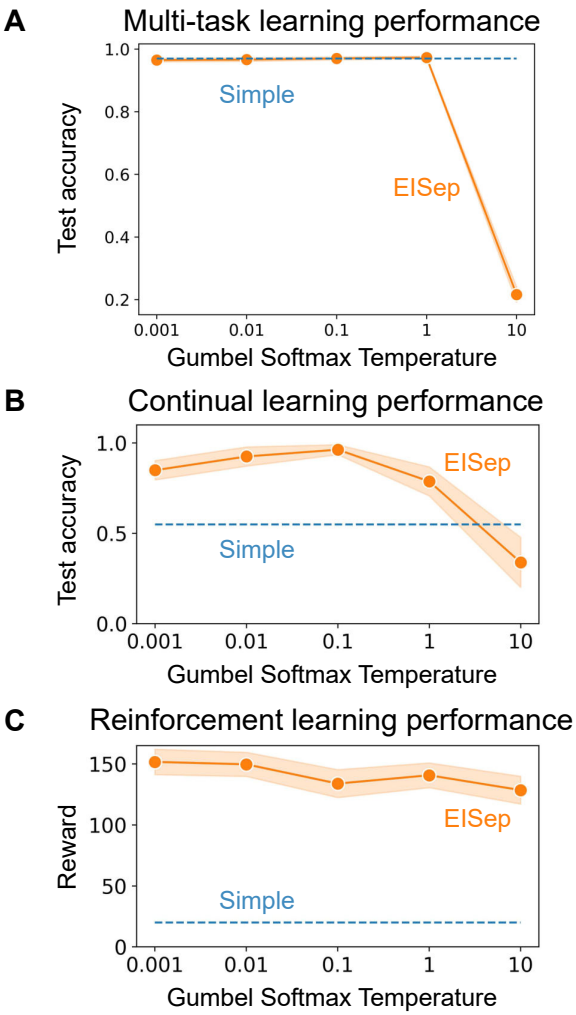


Figure 9: Under different temperature for Gumbel-Softmax sampling, the performance in multi-task learning (**A**), continual learning (**B**), and reinforcement learning (**C**).

G.2 Sampling temperature

To test the sensitivity of EISep to the temperature parameter in Gumbel-Softmax sampling, we evaluated model performance across a range of temperatures. Compared to the baseline temperature ($\tau = 0.1$) used in the main text, decreasing the temperature leads to a minor performance drop, likely because lower temperatures push the Gumbel-Softmax output toward a step function, causing gradients to vanish and the E/I mask to become effectively frozen — approximating the behavior of **Fixed-EISep** (Section 6). Conversely, increasing the temperature substantially impairs performance in multi-task learning and continual learning, as the soft E/I assignments become dominated by Gumbel noise, causing frequent sign flips across training steps and preventing stable weight updates from accumulating. Furthermore, high temperatures introduce a train-test mismatch: the sign assignments at test time may differ from those seen during training, leading to inconsistent network behavior.

G.3 Entropy loss coefficient

To test the sensitivity of EISep to the entropy loss coefficient λ_e (Sec. 3.2), we evaluated model performance across a range of coefficients. Compared to the default setting ($\lambda_e = 1$) used in the main text, removing the entropy loss ($\lambda_e = 0$) significantly degrades performance (Fig. 10), as the non-convergent signs will have occasional flips that cause train-test mismatch. On the other hand,

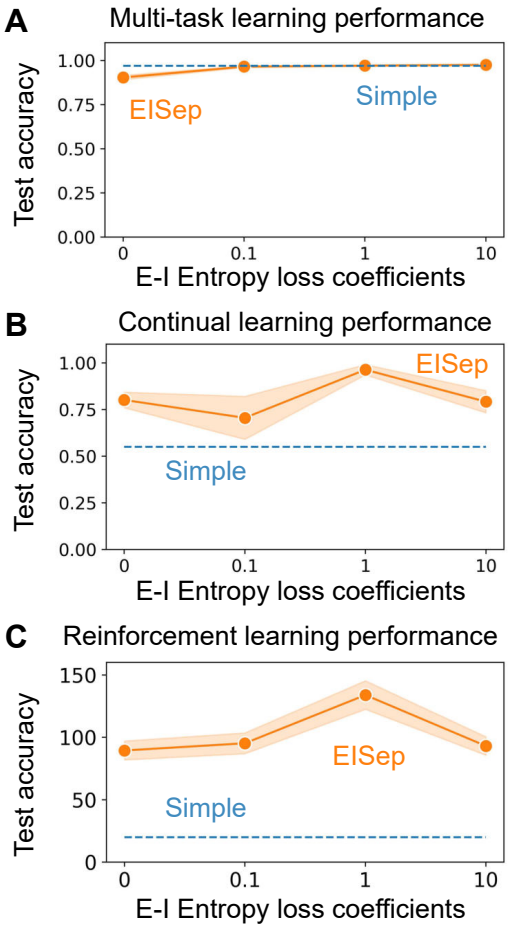


Figure 10: Under different E-I logit entropy loss coefficients, the performance in multi-task learning (**A**), continual learning (**B**) and reinforcement learning (**C**)).

moderate changes to λ_e have little effect on relatively simple multi-task learning, but noticeably impair performance in continual learning and reinforcement learning.

G.4 Activation function

To test the sensitivity of EISep to the activation function, we evaluate model performance using ReLU ((Fig. 11)). Compared to the default setting ($\mathrm{Softplus}$), the new activation function causes a slight drop in continual learning performance but does not affect multi-task learning, and the advantage of EISep to Simple still exists. In reinforcement learning, however, training becomes unstable due to persistent gradient explosions, and the training is not able to converge.

G.5 Initialization method

To test the fairness of the initialization method, we adapt EISep's initialization method and apply it to all other baselines. All baselines show worse performance than EISep (Table 2). This result also shows EISep's performance cannot be naively explained by its initialization method.

G.6 Spectral radius

To test the fairness of the spectral radius, we rescaled the recurrent weight matrix of each model so that all models had the same initial spectral radius as EISep, while leaving their respective parameterizations unchanged. Under this matched initialization, EISep still achieves the highest average performance. Notably, DANN-SGD and ColEI-SGD have smaller final spectral radii than

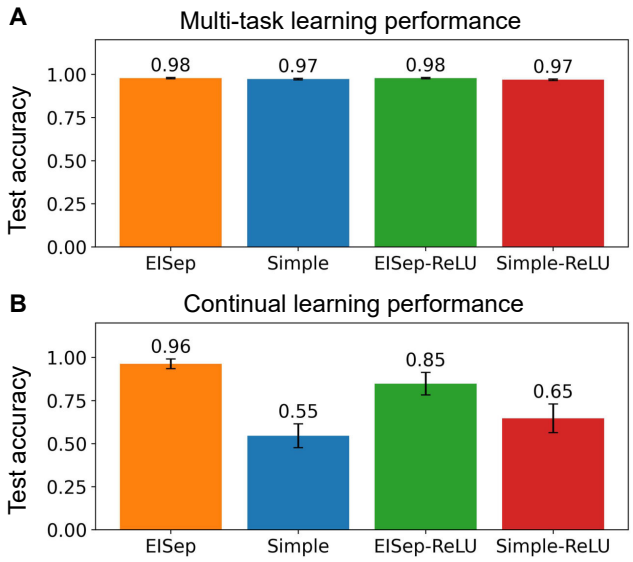


Figure 11: Under activation functions, the performance in multi-task learning (**A**) and continual learning (**B**).

Table 2: Performance under default and EISep-like initialization.

Model	Default initialization	EISep-like initialization
EISep	0.9699 ± 0.0025	0.9699 ± 0.0025
Simple	0.9654 ± 0.0089	0.9705 ± 0.0024
DANN-SGD	0.8344 ± 0.0215	0.6276 ± 0.0217
DANN-Adam	0.7503 ± 0.0558	0.6528 ± 0.0523
ColEI-SGD	0.7095 ± 0.0545	0.7065 ± 0.0397
ColEI-Adam	0.6108 ± 0.0918	0.6230 ± 0.0980
DBPN	0.8055 ± 0.0405	0.8581 ± 0.0115

EISep but substantially lower performance, indicating that simply minimizing the spectral radius may not necessarily lead to good task performance. These results show that EISep's advantage cannot be attributed solely to its initial spectral radius. Its parameterization also shapes how the recurrent dynamics evolve during training.

Table 3: Control with matched initial spectral radius. All models have the same initial spectral radius, 0.6715 ± 0.0093, measured across all 5 seeds. Final spectral radius and performance are reported as mean $\pm$ SEM over valid terminal runs: 5 seeds for every model except ColEI-Adam (3 valid seeds).

Model	Final spectral radius	Accuracy
EISep	2.7762 ± 0.1374	0.9745 ± 0.0024
Simple	6.9175 ± 0.5926	0.9705 ± 0.0024
DANN-SGD	1.0200 ± 0.0210	0.8539 ± 0.0118
DANN-Adam	14.4388 ± 4.4922	0.8203 ± 0.0355
ColEI-SGD	0.9769 ± 0.0281	0.6734 ± 0.0618
ColEI-Adam	5.8978 ± 0.7504	0.7823 ± 0.1355
DBPN	5.1758 ± 0.6728	0.8556 ± 0.0230

G.7 Training order

Training order is known to affect continual learning dynamics [43]. We vary the task sequence and find that the "grouped" order (*Go* $\rightarrow$ *Delay Go* $\rightarrow$ *Anti* $\rightarrow$ *Delay Anti*) performs worse than the "alternating" order (*Go* $\rightarrow$ *Anti* $\rightarrow$ *Delay Go* $\rightarrow$ *Delay Anti*) used in the main experiments

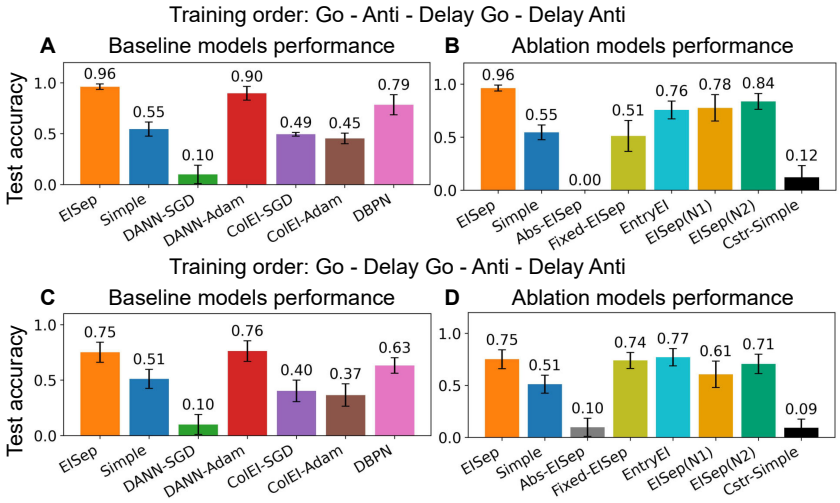


Figure 12: Under different training orders, the performance in continual learning for all baseline models (**A, C**) and ablation models (**B, D**).

(Fig. 12), although EISep can still maintain its own advantage over other models. We attribute this to increased gradient interference: training two similar tasks consecutively drives the network toward a narrow subspace, making subsequent adaptation to opposing stimulus–response mappings more difficult. In contrast, the alternating order interleaves conflicting tasks, encouraging a more balanced representation and enabling the network to converge to a solution that better accommodates all tasks.

H Other ablation models

Here, we introduce two new ablation models that are not presented in the main text.

H.1 The ongoing learnability of signs

In Sec. 6, we show that Fixed-EISep underperforms EISep, highlighting the importance of learnable signs. However, it remains unclear whether this benefit arises primarily from discovering a good sign configuration, or from maintaining sign adaptability throughout training. To disentangle these effects, we introduce **EISep-ReInit**, which initializes the sign matrix using a pretrained EISep model, while reinitializing and optimizing all other parameters as in standard training.

The result is in Fig. 13. On multi-task learning, EISep and EISep-ReInit perform similarly, suggesting that identifying a suitable sign configuration is largely sufficient in simpler settings. In contrast, on continual learning, EISep-ReInit underperforms EISep but still surpasses Fixed-EISep, indicating that both a good initialization and ongoing sign adaptability contribute to mitigating interference. In reinforcement learning, EISep-ReInit performs comparably to Fixed-EISep and significantly worse than EISep, suggesting that continual adaptation of signs is critical for handling non-stationarity and stabilizing high-variance gradient updates. Together, these results indicate that learnable signs not only provide a good initial inductive bias, but also act as a dynamic mechanism for reorganizing computation during training.

We further investigate whether this adaptability is required throughout training or primarily during early stages. To this end, we introduce **Trunc-X**, where after X% of training, the sign matrix is fixed and the entropy regularization is disabled (Fig. 14). While multi-task learning remains largely unaffected, both continual learning and reinforcement learning exhibit clear performance degradation compared to EISep. This suggests that ongoing sign plasticity helps the network adapt to shifting task demands and maintain stable dynamics under non-stationary optimization.

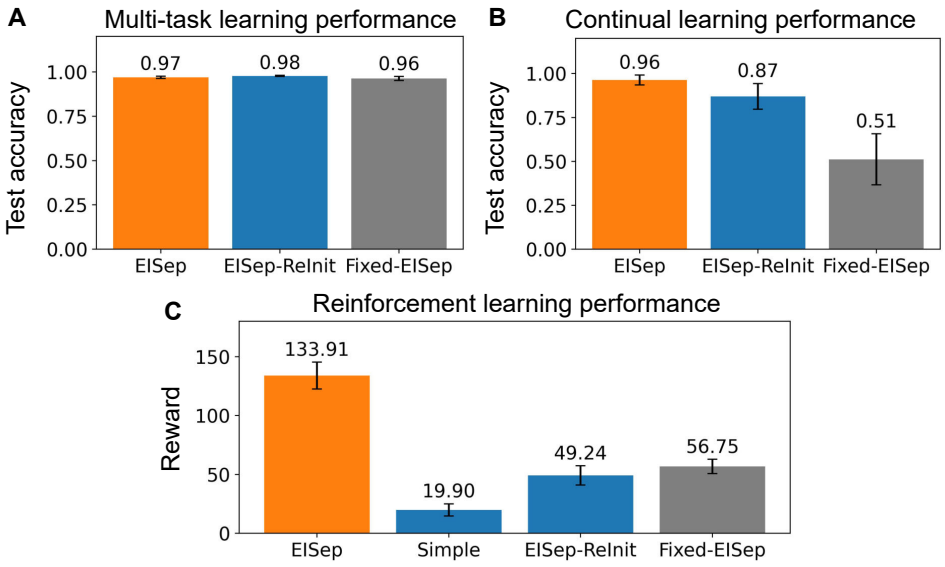


Figure 13: The performance of EISep-ReInit in multi-task learning (**A**), continual learning (**B**) and reinforcement learning (**C**).

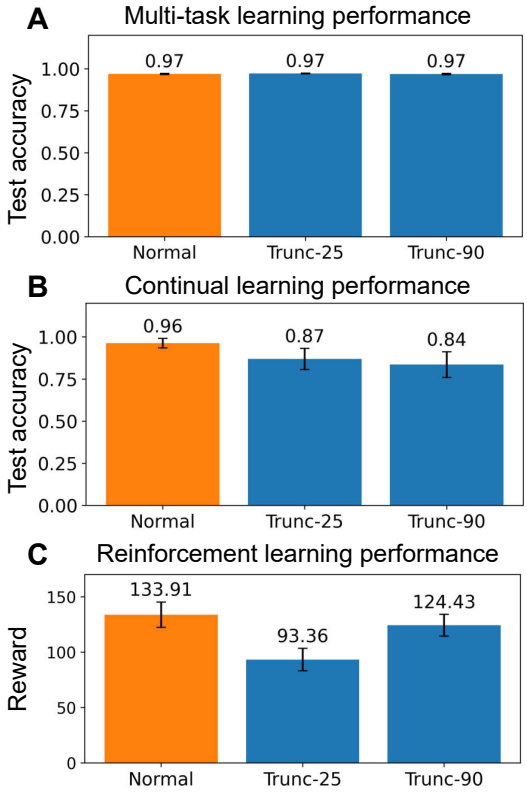


Figure 14: Under different truncation proportions, the performance in multi-task learning (**A**), continual learning (**B**), and reinforcement learning (**C**)

.

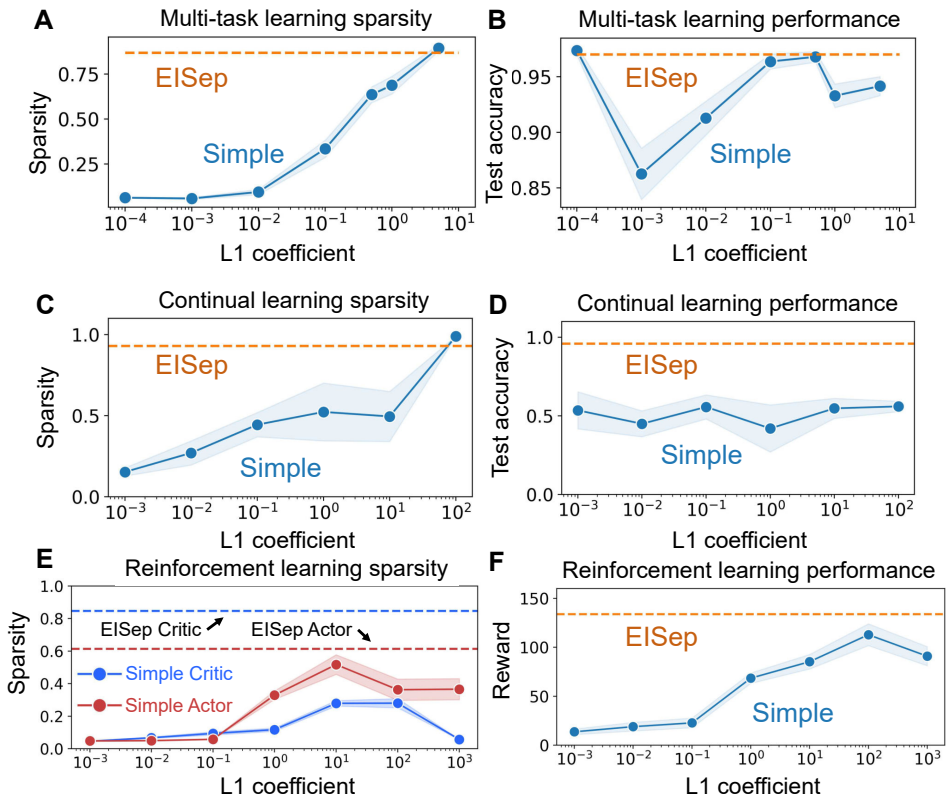


Figure 15: Under different L1 coefficients, the sparsity and performance in multi-task learning (**A, B**), continual learning (**C, D**), and reinforcement learning (**E, F**). In RL, EISep(A+C) is compared with Simple, whose actor and critic are both L1-regularized with the same coefficients.

H.2 Simple with L1 regularization

As mentioned in Sec. 7, EISep faces the trade-off between stability and convergence speed, while achieving much higher sparsity than Simple (Sec. 4.3.2). This raises the question of whether EISep is simply a regularized version of Simple. To test this, we apply L1 regularization to Simple and vary the regularization coefficients. Although large L1 coefficients can induce sparsity comparable to EISep (Fig. 15A, C), performance remains inferior in continual learning, and becomes unstable in multi-task learning (Fig. 15B, D). In reinforcement learning, even extremely large L1 coefficients fail to achieve comparable sparsity and yield inferior performance, indicating a mismatch between L1-induced sparsity and effective optimization.

These results suggest that EISep provides more than a regularization effect, instead acting as a structured, biologically inspired inductive bias that promotes both efficient connectivity and stable learning.

I Environment introduction in the multi-task cognitive benchmark

We adopt the cognitive task environments introduced in [32], comprising a total of 20 tasks with a shared underlying structure. Here we adapt and simplify the task introductions to help readers understand the environment quickly and clearly. Unless otherwise specified, all tasks follow the same trial format, input–output conventions, and noise standard deviation ($\sigma_n = 1.0$). Task-specific features are described in the corresponding subsections.

I.1 Common Task Structure

Each trial is associated with a one-hot rule input corresponding to the current task, which remains active throughout the entire trial and specifies the task context. The network also receives a *fixation input* that is active from trial onset. While the fixation input is on, the network is required to maintain fixation by activating the fixation output unit. The offset of the fixation input typically signals the onset of the response period, during which the network must report a response direction via a population of directional output units arranged on a ring. In this paper, only two directions (left vs. right) are used.

The network receives one or two stimuli per trial. Each stimulus conveys directional information and may appear in modality 1, modality 2, or both. A trial is considered correct only if the network maintains fixation when required and produces the correct directional response during the go period.

Each task can be segmented into distinct periods: a fixation period (**fix**) preceding stimulus presentation, followed by stimulus period 1 (**stim1**). If two stimuli are presented sequentially, they are separated by a delay period, and the second stimulus is shown during stimulus period 2 (**stim2**).

For clarity, the 20 tasks are grouped into five task families: Go, Anti, Decision Making (DM), Delayed Decision Making (Dly DM), and Matching.

I.2 Go Task Family

The Go family includes the **Go**, **Reaction-Time Go (RT Go)**, and **Delayed Go (Dly Go)** tasks. In all three tasks, a single stimulus is presented in either modality 1 or modality 2, and the correct response is to report the stimulus direction.

These tasks differ in their stimulus onset and offset time. In the Go task, the stimulus appears before the fixation cue goes off. In the RT Go task, the fixation input remains active throughout the trial, and the network is expected to respond immediately upon stimulus onset. In the Dly Go task, the stimulus is presented briefly and followed by a delay period until the fixation cue goes off.

I.3 Anti Task Family

The Anti family consists of the **Anti**, **RT Anti**, and **Dly Anti** tasks. These tasks are structurally identical to their Go-family counterparts, except that the required response is the direction opposite to the stimulus (i.e. the weaker direction in left and right).

I.4 Decision Making (DM) Task Family

The DM family consists of five perceptual decision-making tasks: **DM 1**, **DM 2**, **Context-dependent DM 1 (Ctx DM 1)**, **Context-dependent DM 2 (Ctx DM 2)**, and **Multisensory DM (MultSen DM)**.

In each DM task, two stimuli are presented simultaneously and remain visible until the end of the trial. The direction of stimulus 1 is sampled uniformly from $[0^\circ, 360^\circ)$, while stimulus 2 is drawn such that its direction differs from stimulus 1 by an angle uniformly sampled between 90° and 270°. The stimuli have different strengths.

In DM 1 and DM 2, both stimuli appear exclusively in modality 1 or modality 2, respectively, and the correct response corresponds to the direction of the stimulus with greater strength. In the context-dependent tasks, each stimulus appears in both modalities. In Ctx DM 1, information from modality 2 must be ignored, whereas in Ctx DM 2, information from modality 1 must be ignored. In the Multisensory DM task, the correct response is determined by the stimulus with the stronger combined strength across modalities 1 and 2.

I.5 Delayed Decision Making (Dly DM) Task Family

The Dly DM family includes **Dly DM 1**, **Dly DM 2**, **Ctx Dly DM 1**, and **Ctx Dly DM 2**. These tasks parallel the corresponding DM tasks but introduce temporal delay between the two stimuli.

In these tasks, both stimuli are presented briefly and separated by a delay period, followed by an additional short delay after the offset of the second stimulus. The Dly DM tasks are inspired

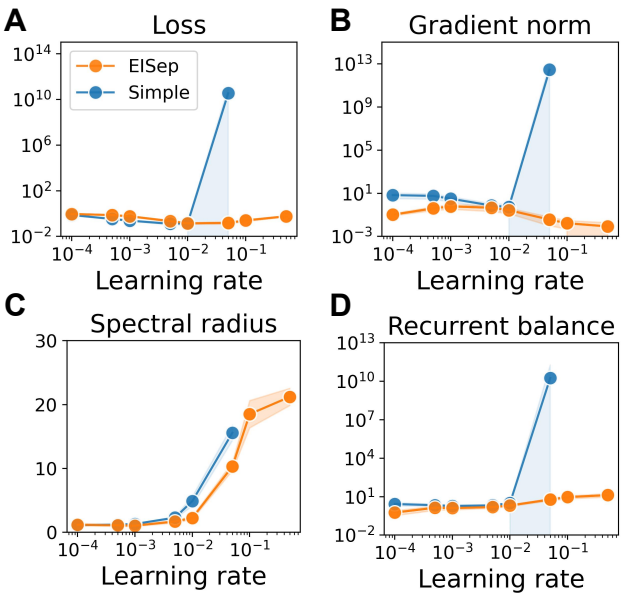


Figure 16: Training metrics under different learning rates in the cognitive task suite, including (**A.**) loss, (**B.**) gradient norm, (**C.**) spectral radius, and (**D.**) recurrent balance.

by classical parametric working-memory experiments and require the network to retain stimulus information across delays to make a correct comparison.

I.6 Matching Task Family

The Matching family consists of **Delayed Match-to-Sample (DMS)**, **Delayed Non-Match-to-Sample (DNMS)**, **Delayed Match-to-Category (DMC)**, and **Delayed Non-Match-to-Category (DNMC)** tasks.

In these tasks, two stimuli are presented sequentially and separated by a delay period. Each stimulus may appear in either modality. The required response depends on whether the two stimuli are considered a match. In DMS and DNMS, stimuli are matched if they share the same direction, regardless of modality. In DMC and DNMC, stimuli are matched if their directions fall within the same category, with categories defined as $[0°, 180°)$ and $[180°, 360°)$.

In DMS and DMC tasks, the network should respond in the direction of the second stimulus if the stimuli match and maintain fixation otherwise. In DNMS and DNMC tasks, the network should respond only when the stimuli do not match and maintain fixation for matched trials.

During training, half of the trials are matching and half are non-matching. In DMS and DNMS, the first stimulus is drawn randomly; the second stimulus either matches the first or differs by an angle uniformly sampled between $10°$ and $350°$. In DMC and DNMC, both stimuli are drawn independently from a uniform directional distribution.

J Training stability in the multi-learning-rate experiment

Fig. 2C shows that EISep RNNs maintain superior performance at higher learning rates and exhibit enhanced robustness during optimization. We hypothesize that this advantage arises from the intrinsic stabilized training dynamics induced by E-I separation. Here, we provide additional evidence to support this interpretation. Across a wide range of learning rates, EISep consistently exhibits lower training loss (Fig. 16A), smaller gradient norms (Fig. 16B), and a more tightly bounded spectral radius of the recurrent weight matrix (Fig. 16C). Notably, these stability indicators co-occur with a tighter balance between excitatory and inhibitory recurrent activity (Fig. 16D). The recurrent balance is defined as the averaged input of each neuron, calculated as $r_b = \text{mean}(Wh)$.

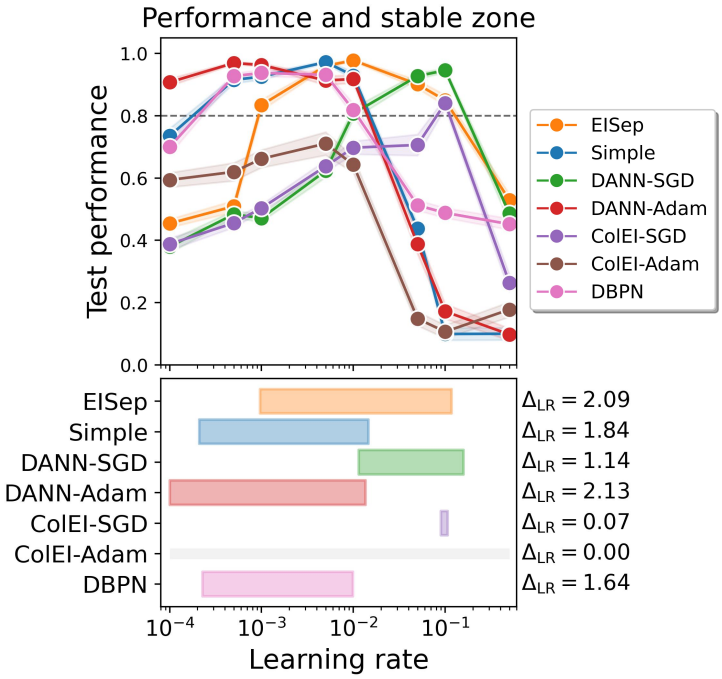


Figure 17: Performance and stable zones for different baselines vs. learning rates in the cognitive task suite.

We further evaluate all baseline models across the same range of learning rates (Fig. 17). To summarize training robustness, we define a *stable zone* for each model as the range of log learning rates over which the lower confidence bound of performance (LCB, mean minus standard error) exceeds 0.8. To estimate its boundaries, we identify adjacent learning-rate values whose performance lies immediately above and below the threshold, and linearly interpolate between them to determine the intersection points with the line performance $= 0.8$. We quantify the width of the stable zone as Δ_{LR}, defined as the difference between its right and left boundaries; if a model fails to exceed the threshold at any learning rate, $\Delta_{\mathrm{LR}} = 0$.

Under this criterion, EISep achieves a wider stable zone than most other baselines, with the exception of DANN-Adam. However, DANN-Adam's stable zone is concentrated at lower learning rates, indicating that it is better at escaping from the plateaus rather than stabilizing gradients. In contrast, EISep's stable zone resides at higher learning-rate area, consistent with the stabilization effects induced by balanced recurrent dynamics. Together, these results indicate that EISep achieves improved optimization stability through a biologically grounded mechanism (E-I balance).

K Modularity metric

K.1 Case study

In the 2AFC task, the network chooses the stronger signal out of two noisy inputs (Sec. 4.3.1). To identify functional structure, we quantify neuron's selectivity using signal-detection index d'. For each unit i, we compute

$$d'_i = \frac{\mu_i^{(1)} - \mu_i^{(2)}}{\sqrt{((\sigma_i^{(1)})^2 + (\sigma_i^{(2)})^2)/2}}, \tag{17}$$

where $\mu_i^{(k)}$ and $\sigma_i^{(k)}$ $(k = 1, 2)$ are the mean and standard deviation of its activity on those trials when signal k is stronger. We label units with $d' > 0.1$ as selective for Signal 1 (Group 1, G1), units with $d' < -0.1$ as selective for Signal 2 (G2), and units with $|d'| \leq 0.1$ as non-selective (NS).

K.2 Population analysis

Since the recurrent weight matrix is directed and signed, it does not satisfy the symmetric nonnegative assumptions of standard spectral clustering and graph-cut methods. We therefore use two adapted approaches for clustering directed signed graphs: *Directed spectral clustering* and *Lifted spectral clustering*. We then evaluate the resulting partitions using *signed modularity*.

Setup. Given a directed signed recurrent weight matrix $W \in \mathbb{R}^{n\times n}$, we first remove self-connections by setting the diagonal of W to zero. We then split the matrix into nonnegative positive and negative parts,

$$A^+_{ij} = \max(W_{ij}, 0), \qquad A^-_{ij} = \max(-W_{ij}, 0).$$

Rows correspond to outgoing connections. In practice, we transpose the recurrent weight matrix before graph construction so that graph rows align with outgoing synaptic projections.

Define the directed strengths and total masses

$$\begin{aligned} k_i^{\pm,\mathrm{out}} &= \sum_j A^{\pm}_{ij}, \qquad k_j^{\pm,\mathrm{in}} = \sum_i A^{\pm}_{ij}, \\ m^{\pm} &= \sum_{ij} A^{\pm}_{ij}, \\ D^{\pm,\mathrm{out}} &= \mathrm{diag}(k^{\pm,\mathrm{out}}), \qquad D^{\pm,\mathrm{in}} = \mathrm{diag}(k^{\pm,\mathrm{in}}). \end{aligned}$$

To avoid numerical singularities from zero-degree nodes (e.g., neurons with exclusively inhibitory or exclusively excitatory outgoing weights under Dale's principle), all inverse square-root degree normalizations are stabilized with a small constant ε before inversion.

Directed spectral clustering

We first perform two-sided degree normalization to preserve source–target heterogeneity:

$$\begin{aligned} M^+ &= (D^{+,\mathrm{out}})^{-1/2} A^+ (D^{+,\mathrm{in}})^{-1/2}, \\ M^- &= (D^{-,\mathrm{out}})^{-1/2} A^- (D^{-,\mathrm{in}})^{-1/2}. \end{aligned}$$

Each normalized matrix is then symmetrized,

$$\begin{aligned} S^+ &= \tfrac{1}{2}(M^+ + {M^+}^\top), \\ S^- &= \tfrac{1}{2}(M^- + {M^-}^\top), \end{aligned}$$

and combined into a symmetric signed similarity matrix

$$S = S^+ - S^-.$$

Define the signed degree

$$d_i = \sum_j |S_{ij}|, \qquad D = \mathrm{diag}(d),$$

and the signed normalized Laplacian-like operator

$$L = I - D^{-1/2} S D^{-1/2}.$$

We compute the k eigenvectors associated with the smallest eigenvalues of L, row-normalize the embeddings, and apply k-means clustering to obtain cluster labels c.

Lifted spectral clustering

We construct a lifted $2n \times 2n$ adjacency matrix that encodes attraction and repulsion:

$$\widetilde{A} = \begin{bmatrix} A^+ & A^- \\ A^- & A^+ \end{bmatrix}.$$

To obtain a symmetric spectral objective, we symmetrize the lifted adjacency,

$$\widetilde{A}_{\mathrm{sym}} = \tfrac{1}{2}(\widetilde{A} + \widetilde{A}^{\top}), \qquad \widetilde{D} = \mathrm{diag}(\widetilde{A}_{\mathrm{sym}}\mathbf{1}),$$

and define the normalized Laplacian

$$\widetilde{L} = I - \widetilde{D}^{-1/2}\widetilde{A}_{\mathrm{sym}}\widetilde{D}^{-1/2}.$$

We compute the k smallest-eigenvalue eigenvectors of $\widetilde{L}$, row-normalize the embeddings, and apply k-means clustering on the $2n$ lifted nodes. The resulting labels $\widetilde{c}$ are folded back to the original graph by assigning each original node i the label of its "plus" copy i^{+}.

Signed modularity

Given a clustering assignment c, we compute the directed signed modularity

$$Q_{\mathrm{signed}} = \frac{1}{m^{+}} \sum_{i,j} \left(A^{+}_{ij} - \frac{k_i^{+,\mathrm{out}} k_j^{+,\mathrm{in}}}{m^{+}} \right) \mathbf{1}[c_i = c_j] - \frac{1}{m^{-}} \sum_{i,j} \left(A^{-}_{ij} - \frac{k_i^{-,\mathrm{out}} k_j^{-,\mathrm{in}}}{m^{-}} \right) \mathbf{1}[c_i = c_j].$$

This quantity measures excess positive within-module connectivity minus excess negative within-module connectivity relative to directed degree-preserving null models. Larger Q_{signed} indicates stronger modular organization.

Evaluation protocol

For each trained network and random seed, we extract the recurrent weight matrix and apply both directed signed spectral clustering and lifted spectral clustering. For each method, we scan the number of clusters from $k = 2$ to $k = 10$. Each clustering is repeated with five random restarts of k-means using different random seeds, and the partition with the largest signed modularity is retained. Fig. 4 reports the mean modularity across random seeds together with standard error across seeds.

L Continual learning

L.1 Training curve under different learning rates

Here we show the performance of each task during the whole training process in Fig. 5A. EISep achieves higher performance with resistance to catastrophic forgetting (Fig. 18).

L.2 Decoding loss

Here we report the training loss trajectories for the continual learning task described in Sec. 5.1. Compared to the Simple, EISep exhibits consistently lower loss and substantially fewer loss spikes throughout training (Fig. 19), which translates into stronger resistance to catastrophic forgetting at convergence (Fig. 5). We further evaluate additional baseline models using their respective optimal learning rates in Fig. 20. EISep achieves better overall performance than most other baseline models (Fig. 20H). At the final round of the four task trials, we compute performance averaged across these four ending trials, and EISep still performs well, highlighting its superior optimization stability and robustness in continual learning settings.

M Supervised learning

To assess the generalizability of EISep RNN beyond cognitively motivated tasks, continual learning, and reinforcement learning, we further evaluate the model on a set of standard supervised benchmarks commonly used to test recurrent neural networks.

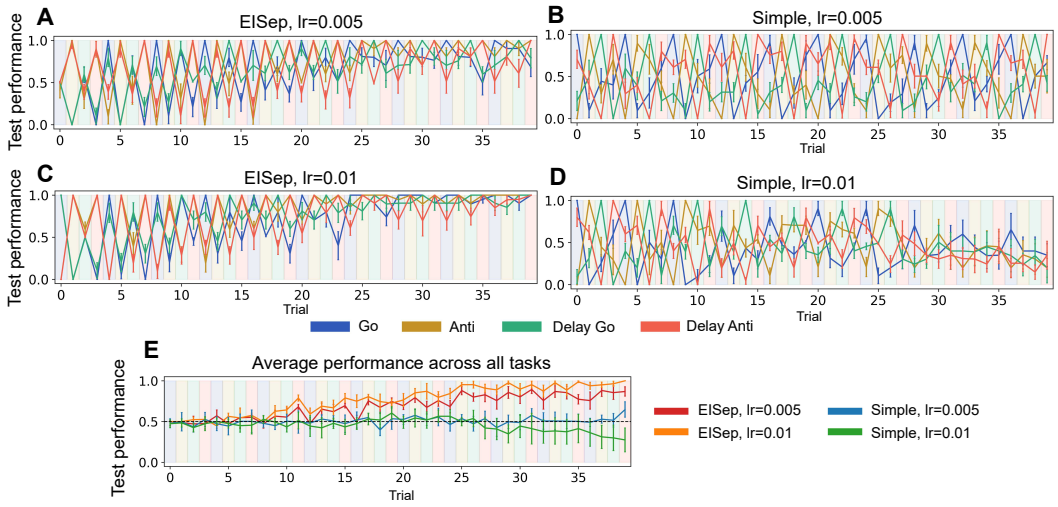


Figure 18: In continual learning, a full task sequence is composed of four tasks, and the sequence is repeated for 10 times, resulting in 40 task trials in total. **A-D.** Training curve in continual learning for different models and learning rates. For a two-channel stimulus, the two "go" tasks require selecting the stronger channel, while the two "anti" tasks require the weaker one. The two "delay" variants insert a delay period between stimulus and response. **E.** Average performance across all tasks over training.

We consider one vision task and two algorithmic benchmarks that probe complementary aspects of general-purpose sequence processing (Fig. 21). Together, these tasks span long-term temporal integration, symbolic reasoning, and high-dimensional perceptual inference, each emphasizing distinct computational demands:

Adding problem. The adding problem serves as a benchmark for long-term dependency learning. In this synthetic task, the network receives a length-20 sequence of two-dimensional inputs. The first dimension contains a random number in $[0, 1]$ at each time step, while the second dimension is a binary indicator marking several positions in the sequence with probability $p = 0.2$. The objective is to output the sum of the marked numbers at the final time step. Because all unmarked inputs act as distractors, the network must retain relevant information over long temporal gaps and perform multiple arithmetic operations. This task is widely used to evaluate an RNN's ability to preserve information in the presence of vanishing gradients. To increase the difficulty, we change the number of marked locations from 2 (fixed) to a variable number n_v, sampled randomly for each sequence.

Sorting problem. The sorting task probes a model's ability to perform algorithmic sequence transformations. The input consists of a fixed-length sequence s of n real-valued numbers ($n = 5$ in our experiments). The network is required to output the rank of each element in the sequence, i.e., $\mathrm{argsort}(s)$. Because the correct output at each position depends on the global ordering of the entire sequence, successful performance requires learning relational and comparison-based representations across time.

Sequential MNIST. Sequential MNIST evaluates a network's capacity to integrate high-dimensional perceptual information over extended temporal horizons. Each MNIST image is unrolled row by row and presented as a sequence of 28 time steps, with each step corresponding to one row of the 28×28 image. The model must accumulate evidence across the entire sequence and produce the correct digit label only after processing the final row.

Across all settings, EISep exhibits the same performance pattern observed previously: stronger performance at higher learning rates and relatively weaker performance at lower learning rates. Nevertheless, EISep consistently outperforms most prior Dale-constrained architectures, indicating that enforcing excitatory–inhibitory separation does not substantially compromise general computational capability, contrary to concerns raised in earlier work [19, 20]. However, the advantages here are not as apparent as in reinforcement learning, probably because in these benchmarks, networks do not suffer from gradient instability as in continual learning or RL tasks.

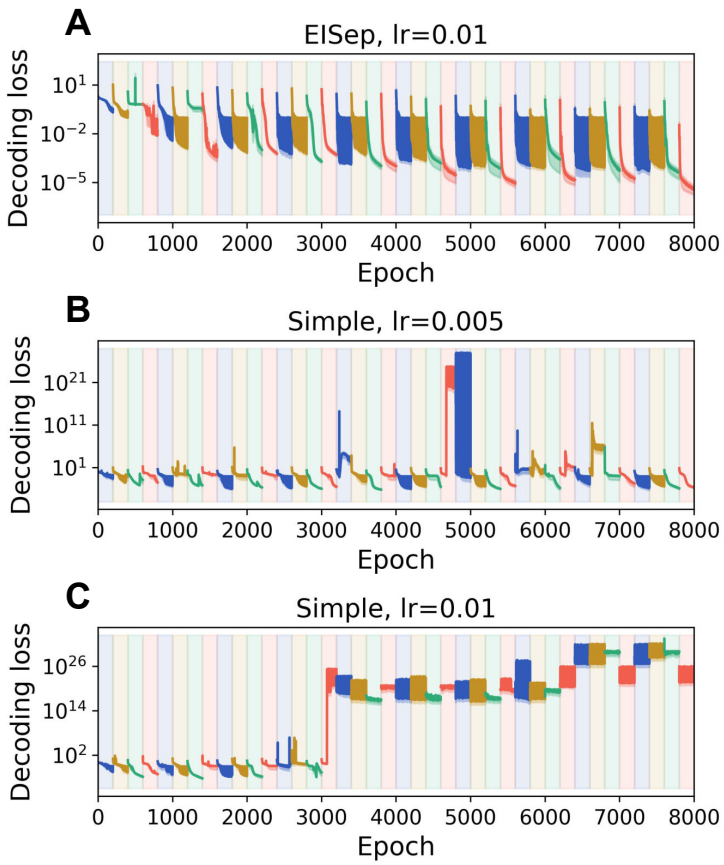


Figure 19: Loss in continual learning for tasks in Fig. 5. **A.** Loss curve of EISep with learning rate 1×10^{-2}. **B.** Loss curve of Simple with learning rate 5×10^{-3}. **C.** Loss curve of Simple with learning rate 1×10^{-2}.

N Early convergence of neuron identities

At first glance, allowing neuron signs to be learnable may appear to reduce the biological plausibility of the model. In biological neural circuits, neurons have well-established identities and do not switch between excitatory and inhibitory roles. However, the training process in general captures not only trial-by-trial synaptic plasticity but also longer-timescale developmental and even evolutionary changes [59]. From this perspective, transient sign changes during early learning actually correspond to neuronal differentiation, and are acceptable if this change is limited to early training. In other words, to avoid interfering with subsequent task-level learning, neuron signs should converge as training proceeds.

To see if the model has exhibited such behavior, we monitor the entropy loss defined in Sec. 3.2 and find that it rapidly drops below 10^{-2} within the first quarter of training (Fig. 22), far less than the normal order of decoding loss (10^{-1}, Fig. 2D left, and dash line in Fig. 22), which is supported by directly tracking the evolution of the logit matrix M (Sec. 3.2) throughout training (Fig. 23). This result indicates that neuron signs converge much earlier than when the network finishes task optimization.

O Validity of the sparsity measure

In Sec. 4.3.2, we show that the EISep RNN achieves substantially higher sparsity than Simple and other Dale-constrained networks. Formally, sparsity r is defined as the proportion of small-magnitude

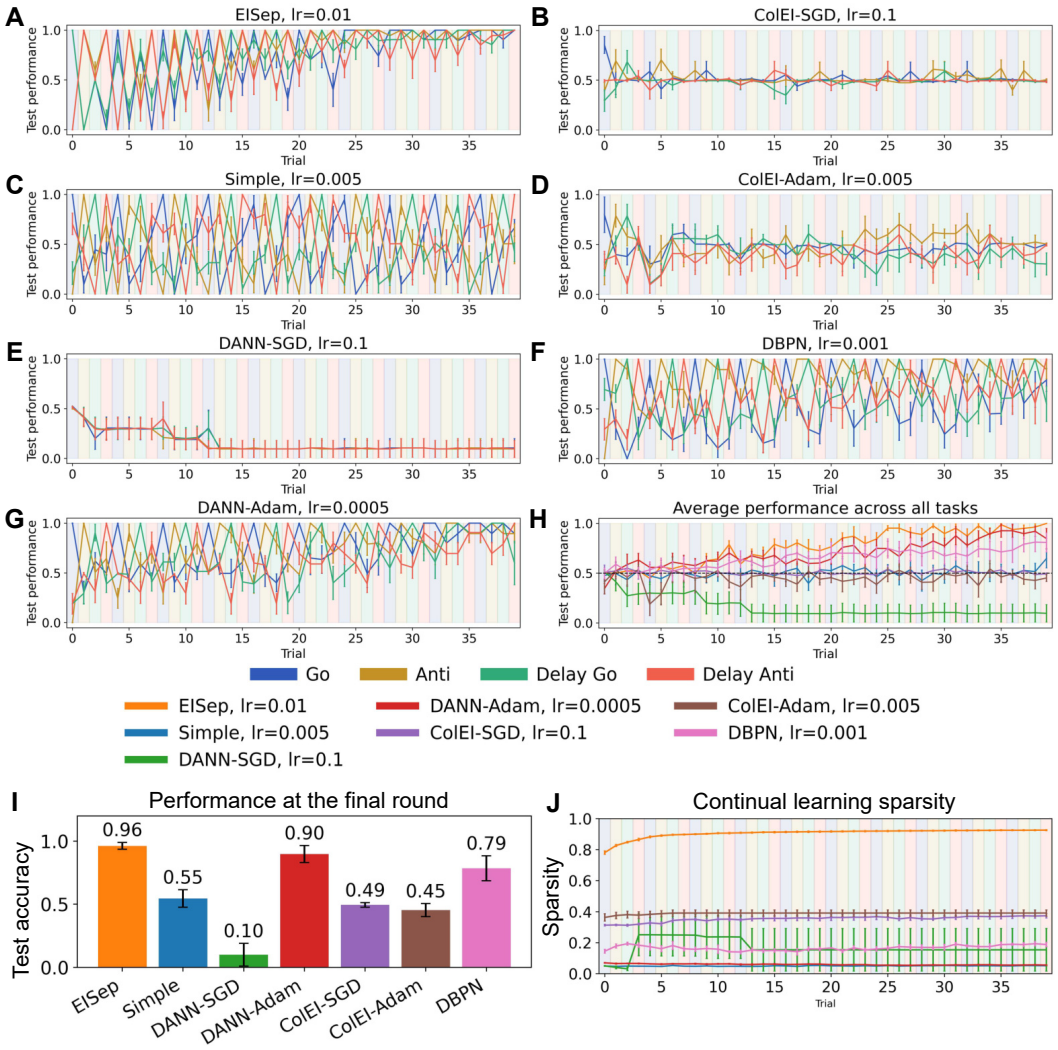


Figure 20: **A–G.** Training curves of different models with their best learning rates. **H.** Average performance across tasks. **I.** Final-round performance averaged over four trials. **J.** The sparsity of all models across trials.

weights,

$$r = \frac{1}{N^2} \sum_{i,j} \left[\mathbb{1}(|W_{ij}| < 0.1\, \sigma(|W|)) \right], \tag{18}$$

where weights whose magnitudes fall below 10% of the standard deviation are considered negligible. To verify that these small weights do not play a critical role in model performance, we construct a sparsity-controlled mask that sets such weights to zero and re-evaluate the models. Performance remains comparable to the original, unmasked networks (Fig. 24A), with differences close to zero (Fig. 24B).

When averaged across tasks, all models are largely robust to the sparsity mask (Fig. 24C, D), proving the validity of the sparsity measure. At the level of individual tasks, several Dale-constrained networks exhibit mild performance degradation after small weights are removed, whereas the EISep and Simple consistently preserve their performance. This suggests that EISep relies on a more specific and compact set of large-magnitude connections, with effective computation primarily supported by a small subset of dominant weights.

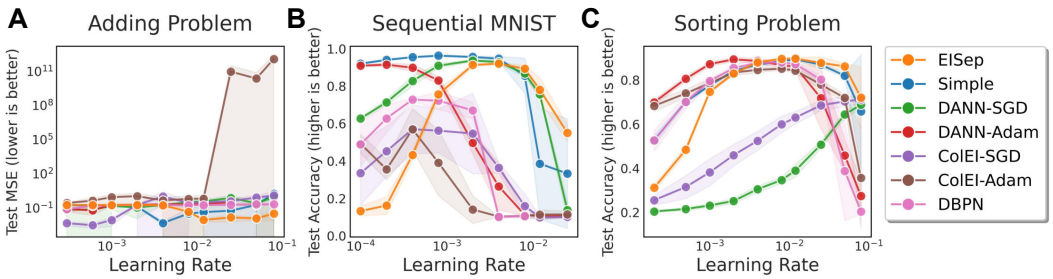


Figure 21: Performance in general tasks, including the adding problem (**A**), sequential MNIST (**B**) and sorting problem (**C**).

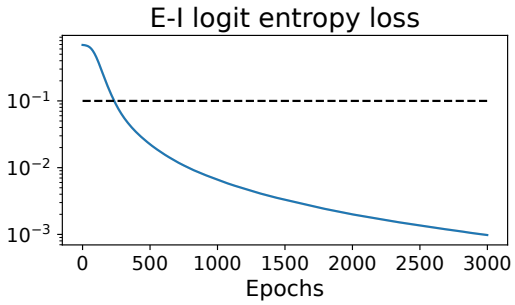


Figure 22: The EI logit entropy loss during the training stage for EISep. The fast convergence of the entropy objective indicates that the neuron signs also converge fast.

P Details of the RL performance

P.1 Performance of other Dale's networks in the Lunar Lander task

Here, we show the performance of other Dale-constrained baselines on the Lunar Lander task. As in EISep (Sec. 5.2), both the actor and the critic can be implemented with Dale-constrained recurrent modules, so there are three variants for each baseline: EI(A) (only actor is Dale-constrained), EI(C) (only critic is Dale-constrained), and EI(A+C) (both actor and critic are Dale-constrained). While some baseline variants perform comparably in specific configurations (e.g., EI(A)), their best performance across variants remains substantially lower than EISep. This outcome is consistent with the result on the multi-task cognitive benchmark where they achieve lower training stability (Fig. 2D).

P.2 Performance under different learning rates for EISep

Here we show the learning-rate sweep results on Lunar Lander (Table 4) and HalfCheetah (Table 5).

Each configuration uses $n = 5$ training seeds and each trained model is evaluated with $n = 20$ test seeds. In the table, each cell reports test return (mean $\pm$ SE) for four model variants (top to bottom): Simple, EISep(A), EISep(C), EISep(A+C). Boldface marks the best learning-rate pair for each variant searched over the whole grid (9 pairs).

P.3 Performance of EISep FFN on Lunar Lander

Here we change the RNN module to the feedforward network (FFN) and test it on the Lunar Lander task. To control the parameter size of the network, we only replace the recurrent layer with a feedforward layer of the same hidden size, positioned between the input and output layers, and remove the hidden state entirely. Under this setting, the EISep variant continues to outperform the Simple baseline (Fig. 26), although the performance gap is smaller than in the recurrent case. This could be explained in two aspects. The higher performance of Simple FFN than Simple RNN suggests that feedforward architectures are less susceptible to training instability than RNNs, which are known to suffer from gradient explosion. The lower performance of EISep FFN than EISep RNN

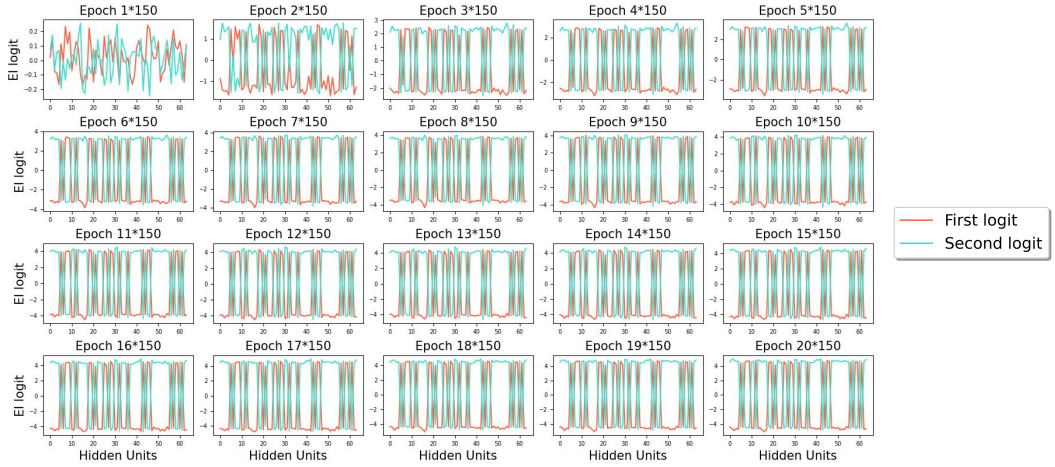


Figure 23: The EI logit matrix M during the entire training stage, recorded at every 150 steps.

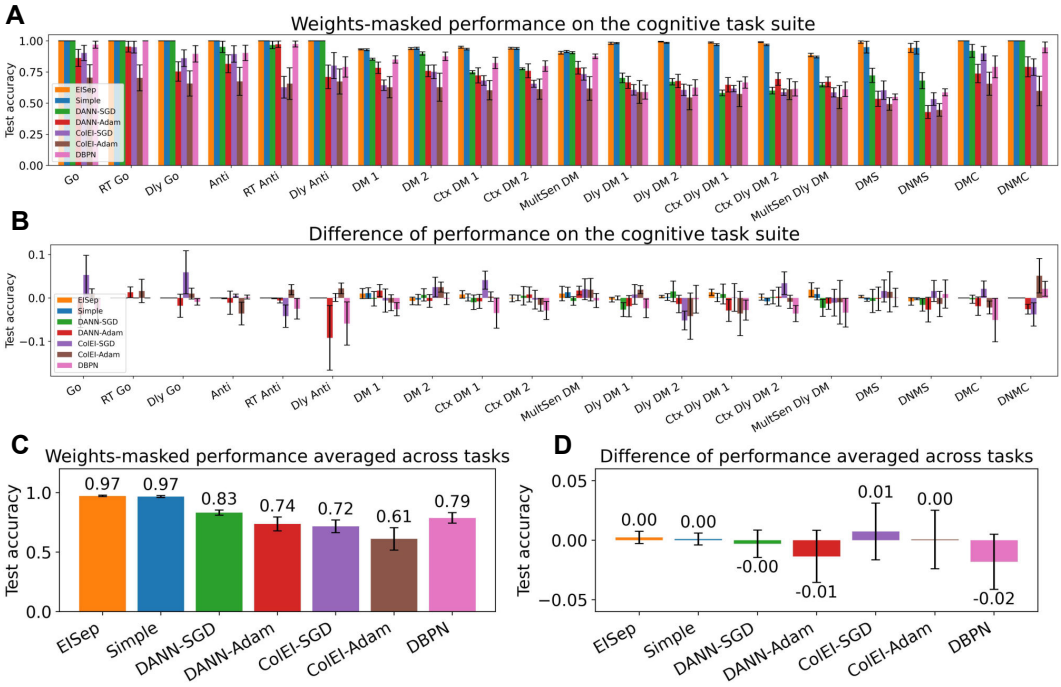


Figure 24: The performance of masked models and their relative errors with the original models. **A**. Performance of 7 masked models on 20 cognitive tasks. **B**. The performance differences of 7 masked models on 20 cognitive tasks. **C**. Performance of 7 models averaged across tasks. **D**. The performance differences of 7 models averaged across tasks.

is consistent with the advantage of recurrent modules to utilize past experiences to make decisions. Taken together, these results indicate that EISep specifically mitigates the gradient instability inherent to recurrent architectures, thereby unlocking the representational and computational advantages of RNNs in complex control tasks.

Q Details of the RL environment

Q.1 Environment details and phase annotation for the LunarLander task

In the Lunar Lander environment, the agent controls a rocket descending toward the lunar surface using left, right, and main engines. Each action fires one of the engines, consumes fuel and affects

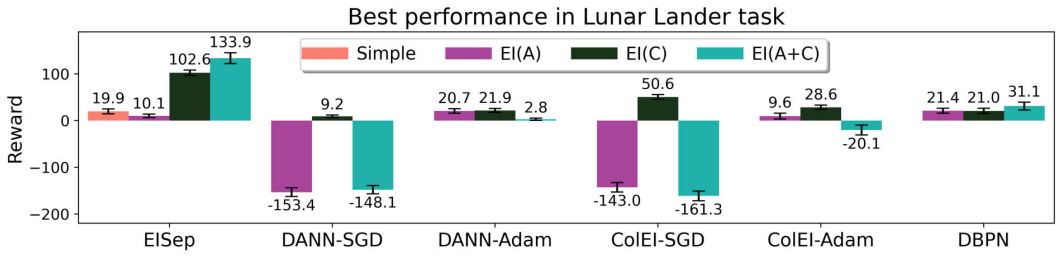


Figure 25: Performance of other Dale's networks in the Lunar Lander task.

Table 4: Learning-rate sweep on LunarLander-v3.

lr(C) \ lr(A)	$\mathbf{3 \times 10^{-4}}$	$\mathbf{1 \times 10^{-3}}$	$\mathbf{3 \times 10^{-3}}$
$\mathbf{1.5 \times 10^{-3}}$	$\mathbf{19.90 \pm 5.21}$	14.60 ± 2.97	1.19 ± 5.62
	-54.30 ± 7.56	7.04 ± 3.08	-5.25 ± 2.53
	69.99 ± 7.51	66.74 ± 7.52	5.57 ± 11.14
	-75.01 ± 11.42	11.57 ± 2.24	31.98 ± 3.61
$\mathbf{5 \times 10^{-3}}$	10.63 ± 3.28	-6.09 ± 2.68	-17.44 ± 6.36
	-54.26 ± 10.73	$\mathbf{10.13 \pm 3.94}$	-0.69 ± 2.07
	$\mathbf{102.57 \pm 5.79}$	101.52 ± 10.22	61.33 ± 7.55
	-58.65 ± 8.92	45.41 ± 7.85	$\mathbf{133.91 \pm 11.46}$
$\mathbf{1.5 \times 10^{-2}}$	2.98 ± 2.75	0.04 ± 2.07	14.09 ± 5.11
	-77.64 ± 12.96	9.75 ± 3.19	-29.33 ± 3.74
	70.89 ± 7.52	93.81 ± 7.89	14.11 ± 9.87
	-25.56 ± 6.63	37.88 ± 7.43	52.23 ± 9.68

Chance level: -200.36 ± 4.16.
Column: actor learning rates. *Row:* critic learning rates.
Order in each cell (top to bottom): Simple, EISep(A), EISep(C), EISep(A+C).

the lander's pose. The final stable landing requires the agent to balance energy efficiency and stability. In the following part, we introduce the definitions of state and action variables, before talking about how to manually annotate the agent based on these states and actions.

State variables. At each time step, the environment state is represented by an 8-dimensional vector

$$(x, y, v_x, v_y, \theta, \dot{\theta}, \text{leg}_L, \text{leg}_R),$$

where (x, y) denote position, (v_x, v_y) linear velocities, θ the lander's angle, $\dot{\theta}$ its angular velocity, and $\text{leg}_L, \text{leg}_R \in \{0, 1\}$ indicate ground contact of the left and right landing legs. The *contact* is defined as whether either leg is in contact with the ground.

The agent selects a discrete action $a \in \{0, 1, 2, 3\}$ at each time step, corresponding to:

- 0: do nothing,
- 1: fire the left orientation engine,
- 2: fire the main engine,
- 3: fire the right orientation engine.

Rewards increase when the lander achieves a smooth and upright touchdown with minimal oscillation, and decrease when tilting, excessive acceleration, or rolling happens.

To analyze the internal dynamics of recurrent agents and relate them to interpretable behavioral stages, we manually annotate each environment state in the LunarLander-v3 task with a discrete *phase* label. These phases are defined using fixed, environment-based thresholds on observable state variables and actions, without reference to the agent's internal representations. This procedure provides a deterministic and reproducible mapping from low-level states to high-level behavioral regimes, as well as an important reference to the internal state analyses (Sec.5.2.2).

Table 5: Learning-rate sweep on HalfCheetah-v5.

lr(C) \ lr(A)	3×10^{-4}	1×10^{-3}	3×10^{-3}
1.5×10^{-3}	1387.94 ± 7.32	1503.11 ± 7.33	1581.90 ± 8.66
	1379.76 ± 23.80	1570.41 ± 22.11	1659.70 ± 37.18
	1510.74 ± 17.22	1550.80 ± 25.58	1781.78 ± 46.34
	1359.10 ± 14.13	1623.82 ± 26.35	1585.68 ± 8.68
5×10^{-3}	1530.66 ± 5.55	1646.83 ± 18.94	1644.64 ± 13.33
	1426.88 ± 24.39	1676.68 ± 36.92	1552.41 ± 20.62
	3089.99 ± 56.89	2297.24 ± 89.07	2266.02 ± 68.28
	2865.65 ± 92.40	2439.39 ± 102.15	1971.96 ± 73.05
1.5×10^{-2}	1592.10 ± 6.12	1670.19 ± 5.71	$\mathbf{1687.00 \pm 20.22}$
	1384.61 ± 23.61	$\mathbf{2019.97 \pm 60.97}$	1693.77 ± 25.94
	2870.01 ± 128.35	3591.95 ± 116.64	$\mathbf{4332.40 \pm 64.58}$
	2542.09 ± 138.04	3258.02 ± 133.81	$\mathbf{3870.66 \pm 132.16}$

Column: actor learning rates. *Row:* critic learning rates.
Order in each cell (top to bottom): Simple, EISep(A), EISep(C), EISep(A+C).

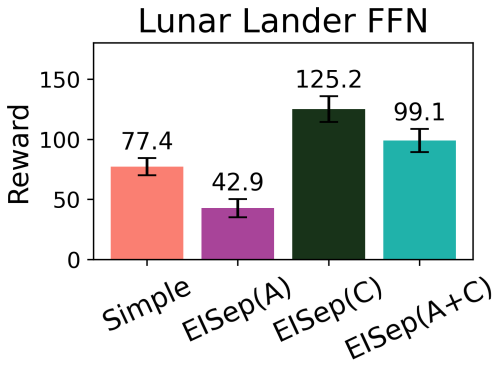


Figure 26: Performance of EISep FFN on the Lunar Lander task.

Phase definitions. Each time step is assigned one of six mutually exclusive phases based on fixed thresholds introduced below.

- **Phase 0: Free fall / early descent (Default).** The lander is at high altitude, descending rapidly, and the main engine is not activated, corresponding to an early, weakly controlled descent:
$$y > 1.0 \ \wedge \ v_y < -2.0 \ \wedge \ a \neq 2.$$
- **Phase 1: Braking / vertical control.** The main engine is actively fired while the lander remains low-speed in the air, indicating deliberate regulation of vertical velocity:
$$y > 0 \ \wedge \ a = 2 \ \wedge \ v_y > -3.0 \ \wedge \ \text{no leg contact}.$$
- **Phase 2: Horizontal alignment / attitude control.** The lander performs corrective maneuvers to regulate lateral motion or orientation in the air:
$$\text{no leg contact} \ \wedge \ \big(|v_x| \geq 0.5 \ \vee \ |\theta| \geq 0.3 \ \vee \ |\dot{\theta}| \geq 0.15\big).$$
- **Phase 3: Touchdown preparation.** The lander approaches the ground with reduced velocity and near-upright orientation, preparing for landing:
$$y \leq 0.2 \ \wedge \ |v_y| < 1.0 \ \wedge \ |\theta| < 0.3 \ \wedge \ \text{no leg contact}.$$
- **Phase 4: Landed / stable contact.** One or both legs are in contact with the ground and the lander remains motionless, or with minimal motion to adjust landing positions:
$$\text{leg contact} \ \wedge \ y \leq 0.3 \ \wedge \ |v_y| < 0.5$$
$$\wedge \ |v_x| < 0.5 \ \wedge \ |\theta| \leq 0.7.$$

- **Phase 5: Crash.** A terminal episode ends without stable ground contact and exhibits at least one hazardous condition:

$$\text{done} \ \wedge \ \neg\text{leg contact} \ \wedge \ \big(y \leq 0.1 \ \vee \ |\theta| > 0.7 \\ \vee \ |v_y| > 3.0 \ \vee \ |v_x| > 3.0 \ \vee \ |x| > 1.0\big).$$

Assignment logic. Phase labels are sequentially assigned using fixed threshold rules on altitude, velocities, orientation, angular velocity, leg contact, action choices, and episode termination flags. When multiple phase conditions are satisfied, later-defined phases overwrite earlier ones, ensuring that terminal events such as landing or crash take precedence. If no specific condition is met, the state defaults to Phase 0 (free fall). In practice, the proposed criteria cover the majority of visited states, with a non-default coverage rate of $60.4\% \pm 0.4\%$ for the Simple agent and $71.5\% \pm 3.2\%$ for EISep(A+C). This shows that it is acceptable to use Phase 0 as a default label for unassigned residual states. Notably, the Simple agent tends to hover near the ground and rarely enters well-defined touchdown preparation or landing phases, resulting in a lower non-default coverage compared to EISep.

These manually annotated phases serve as an external, interpretable reference for evaluating learned representations and temporal segmentation in recurrent networks. Importantly, they are not used during training and rely solely on environment-level information, ensuring that all analyses remain strictly post hoc.

Q.2 Environment introduction for the HalfCheetah task

In the HalfCheetah environment, a robot agent with multiple body parts connected by joints is required to run forward as efficiently and stably as possible. The agent applies continuous torques to the joints to maximize forward velocity while minimizing energy consumption. This task tests a model's ability to perform coordinated continuous control over long time horizons and presents a substantially more challenging optimization problem than Lunar Lander. See the environment documents [51] for more details.

R Design of EISep-LSTM

In Sec. 5.2, EISep LSTM and Simple LSTM (vanilla LSTM) are used to deal with the complex HalfCheetah environment. Here is a brief introduction of these two models.

Simple LSTM. A Long Short-Term Memory (LSTM) network augments a standard recurrent network with gating mechanisms that regulate information flow. The four gates are four recurrent transformations, corresponding to the input, forget, candidate, and output gates. At each time step t, the LSTM maintains a hidden state h_t and a cell state c_t, updated as:

$$i_t = \sigma(W_{ii}x_t + W_{hi}h_{t-1} + b_i), \tag{19}$$
$$f_t = \sigma(W_{if}x_t + W_{hf}h_{t-1} + b_f), \tag{20}$$
$$g_t = \tanh(W_{ig}x_t + W_{hg}h_{t-1} + b_g), \tag{21}$$
$$o_t = \sigma(W_{io}x_t + W_{ho}h_{t-1} + b_o), \tag{22}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \tag{23}$$
$$h_t = o_t \odot \tanh(c_t). \tag{24}$$

These gating mechanisms allow the network to selectively store, forget, and expose information across time, mitigating vanishing gradients and enabling long-term dependency modeling.

EISep adaptation. We modify the recurrent weight matrices $W_{hi}, W_{hf}, W_{hg}, W_{ho}$ to satisfy Dale's principle, enforcing excitatory and inhibitory separation. Similar to EISep RNN (Sec. 3.2), each recurrent matrix is reparameterized as:

$$W = \mathrm{ReLU}(W_0)D, \tag{25}$$

where W_0 is a free learnable matrix and D is a soft diagonal sign matrix with diagonal entries computed using the Gumbel-Softmax sampling.

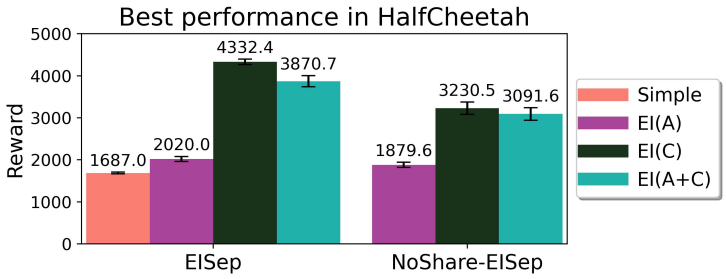


Figure 27: Performance for EISep LSTM with shared E-I identities across gates (left) or with independent identities across gates (right).

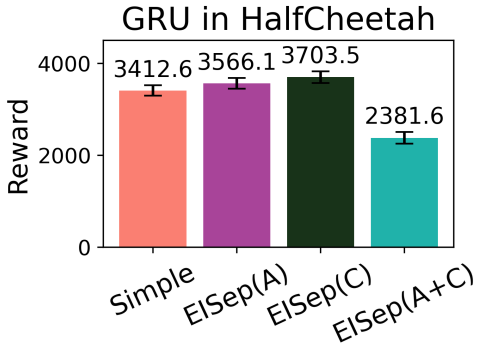


Figure 28: Performance for EISep GRU on HalfCheetah.

Because an LSTM updates its hidden state through four gates, a natural design question is whether these gate weight matrices should share a common sign matrix D, or use independent matrices $D_1, D_2, ...$ From the perspective of biological realism, if we see the hidden state as neuronal firing rates, we would expect that each neuron maintains the same excitatory or inhibitory identity across all transformations. Therefore, we enforce a shared sign mask D across all recurrent matrices in the LSTM. This design is also proved better in practice (Fig. 27).

Consequently, **EISep LSTM** preserves the advantages of LSTM—robust memory gating and long-term credit assignment—while inheriting the stability advantages of EISep.

To show that EISep can generalize to different backbones, we also apply similar adaptation to GRU. EISep still performs better than Simple (Fig. 28).

S Sparsity of modules in the RL tasks

Here, we examine the sparsity of the actor and critic modules in the reinforcement learning tasks. As expected, the EISep-parameterized modules exhibit higher sparsity, whereas modules that remain as vanilla RNNs (or LSTMs) show sparsity close to zero (Fig. 29A, B). This pattern is consistent with the sparsity observed for EISep in the multi-task cognitive benchmark (Sec. 4.3.2), further indicating that EISep induces compact connectivity across different learning settings.

T Details for HMM fitting

The number of possible hidden states is 5, close to the maximal possible number of phases (6, but *Crash* state is extremely rare. See Appendix Q).

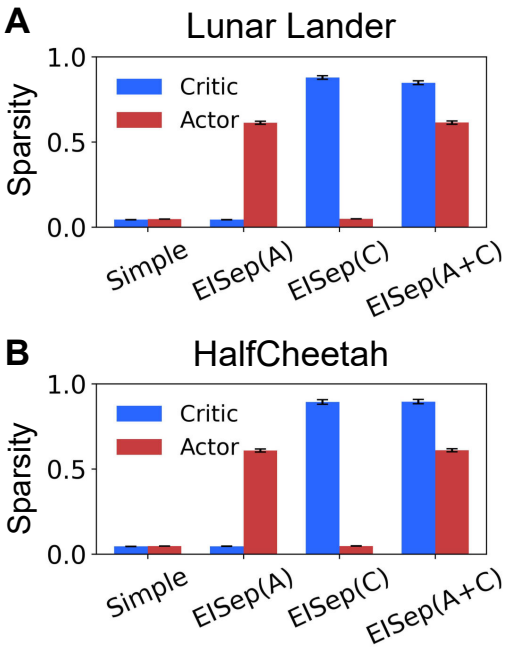


Figure 29: Sparsity averaged across all learning rates and seeds in the **A.** Lunar Lander task, and **B.** HalfCheetah task.

T.1 Long clips of hidden activities

In Sec. 5.2.2, we only presented short clips of actor hidden activities. Here, to provide a more comprehensive visualization, we extend the temporal window to 100 timesteps for hidden activities and provide fitted HMM labels for 200 timesteps. The learning rate settings are the same as in Sec. 5.2.2:

$$\mathrm{lr}(\mathrm{critic}) = 0.005, \quad \mathrm{lr}(\mathrm{actor}) = 0.001. \tag{26}$$

As shown in Fig. 30A, D, G, and J, the hidden-state activity of the Simple exhibits highly regular, high-frequency oscillations, while EISep displays richer, more heterogeneous, and less stereotyped activity patterns. Notably, the oscillation period of Simple's hidden states is extremely short (approximately 4 time steps) compared to the full episode length (approximately 500 time steps), indicating that the recurrent dynamics have effectively collapsed onto a low-dimensional limit cycle. As a result, Simple hidden states fail to encode much meaningful information about the task structure or environment context.

This dynamical collapse is directly reflected in the HMM labels. In Simple, the inferred HMM labels switch rapidly and over-regularly (Fig. 30C, F, I, and L), tracking the fast oscillatory dynamics of the hidden states rather than revealing interpretable task phases. Correspondingly, the estimated transition matrices exhibit an ill-posed structure (Fig. 30B, E, H, and K): the self-transition probabilities are abnormally low (low stickiness), while transition probabilities to other states are disproportionately high (high entropy). This pattern indicates that HMM states are unstable and short-lived, consistent with the absence of persistent latent modes in the underlying RNN dynamics. In contrast, EISep produces transition matrices with high stickiness and low entropy, reflecting stable latent states and coherent temporal organization.

Taken together, these visualizations reinforce our earlier findings: EISep's advantage lies in its ability to sustain rich, non-collapsed recurrent dynamics that support stable and informative latent states, and effective control.

T.2 Quantitative metrics

Here, we use HMM to fit hidden state trajectories in EISep and Simple agents. EISep achieves comparable or higher test log-likelihood than Simple (Fig. 31A), indicating that the latent trajectories

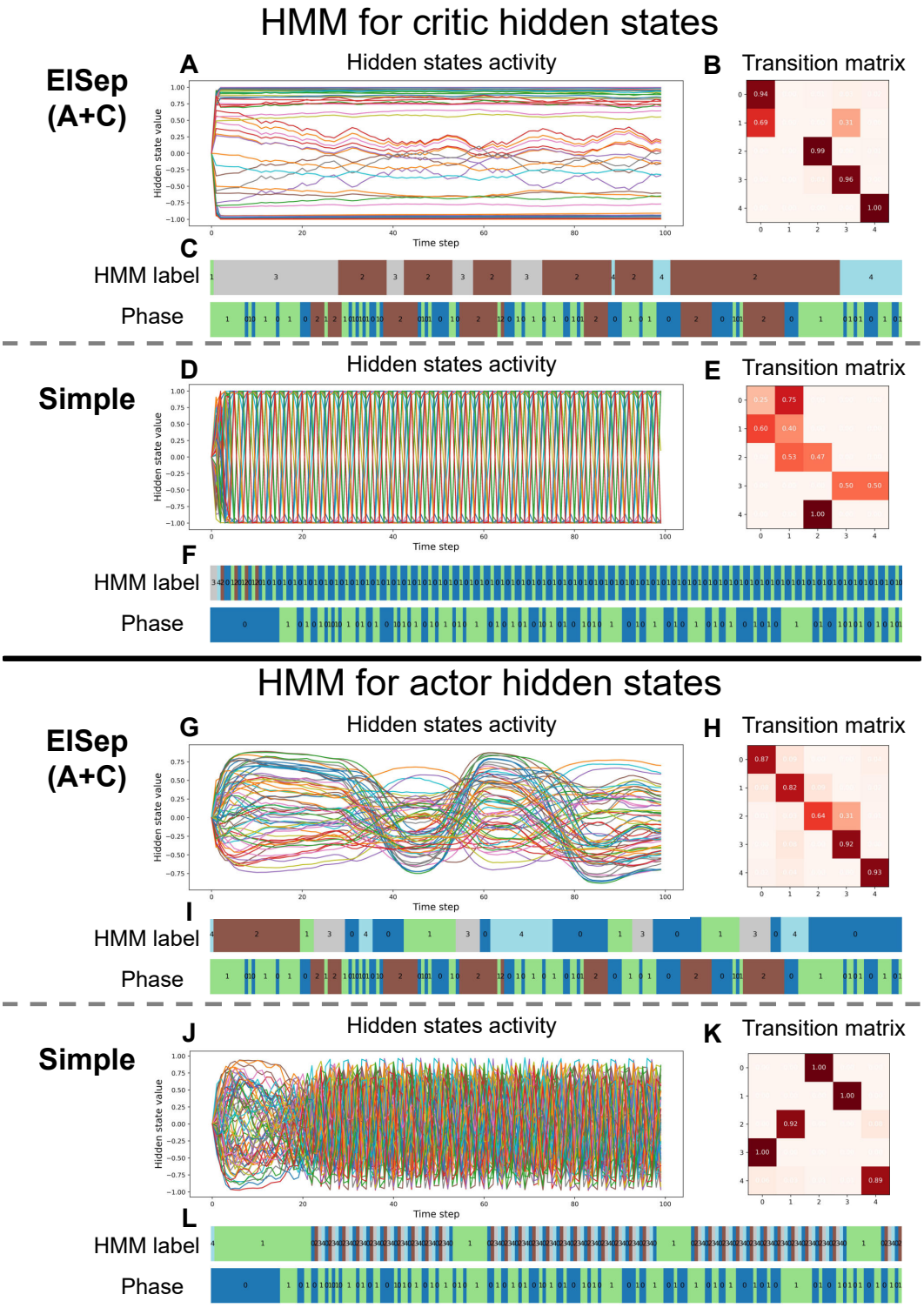


Figure 30: Extended visualizations of the HMM fitting results for both EISep(A+C) and Simple agents. **A.** A raw clip (length=100, beginning of the first episode) of hidden state activities for all 64 units in the EISep critic RNN. **B.** HMM transition matrix across all hidden labels. **C.** HMM labels (up) and manually annotated phases (bottom) for a clip (length=200, beginning of the first episode). The first 100 timesteps are aligned with the timesteps in **A**. **D-F.** Same as **A-C** for Simple critic RNN. **G.-I.** Same as **A-C** for EISep(A+C) actor RNN. **J.-L.** Same as **A-C** for Simple actor RNN.

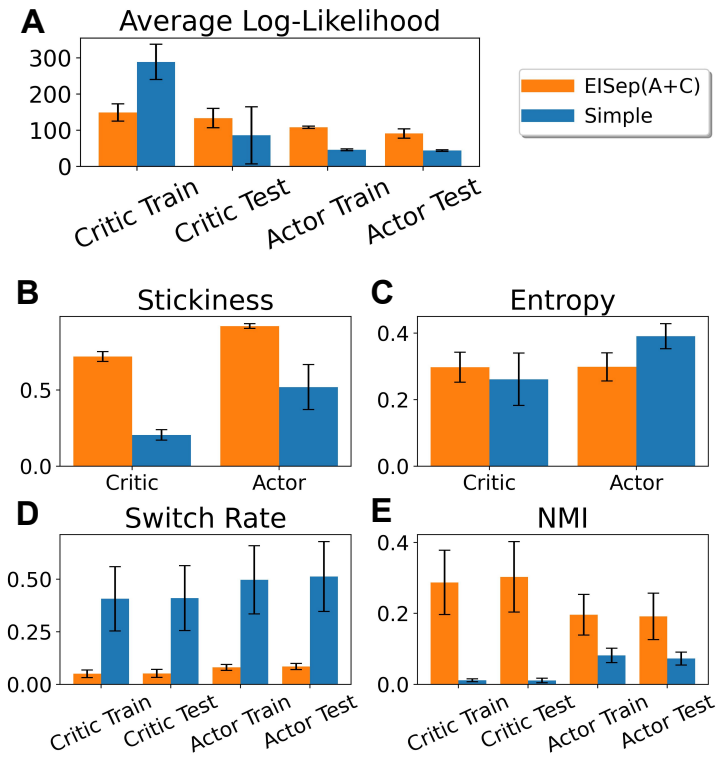


Figure 31: Different metrics of the HMM results of EISep. **A.** The HMM log-likelihood on the train/test dataset for the critic/actor of the two agents(EISep/Simple). **B**. Stickiness of the transition matrix. **C**. Entropy of the transition matrix. **D**. Switch rate of HMM hidden states. **E**. Normalized mutual information between HMM hidden states and handmade labels.

learned under E-I separation are more predictable over time. It seems that Simple critic achieves higher training log-likelihood, but as shown in Fig. 30, the hidden states of the Simple critic have collapsed and the higher training log-likelihood is actually overfitting. As quantified in Sec. 5.2.2, these latent representations contain negligible mutual information with the environmental states (Fig. 31E). Consequently, they provide little utility for downstream decision-making and lead to inferior control performance (Fig. 6B).

Next, we focus on the temporal organization, stability, and behavioral relevance of the latent dynamics. To quantify the temporal coherence of the inferred latent dynamics, we propose two metrics to probe the transition structure: (1) Stickiness s measures the self-transition probabilities, and is defined as $s = \mathrm{trace}(P)$ where P is the estimated transition matrix. (2) Entropy h measures uncertainty in state transitions and is defined as $h = -\frac{1}{L}\sum_{i,j} P_{i,j} \log P_{i,j}$ where L is the number of hidden states. EISep networks exhibits higher stickiness than Simple (Fig. 31B), consistent with the fact that their latent states persist over longer temporal windows (Fig. 30). In the meantime, EISep's actor shows lower transition entropy than Simple's actor (Fig. 31C), consistent with less jittery state transitions. EISep agents also exhibits substantially lower HMM state-switch rates in both actor and critic networks (Fig. 31D). Together, these measures corroborate quantitatively that E-I separation stabilizes recurrent dynamics and develops smooth evolution of internal representations over time. The higher NMI of EISep, as introduced in Sec. 5.2.2, shows stronger alignment between latent dynamics and interpretable task phases, indicating that EISep's latent assignments are environmentally informative.

U Details for ablation study

To disentangle the contributions of individual design choices in EISep, such as recurrent weight filter, learnable signs, and neuron-level sign constraints, we further conduct a systematic ablation study here, as a more detailed complementary to Sec. 6. We summarize different ablation models in Fig. 32.

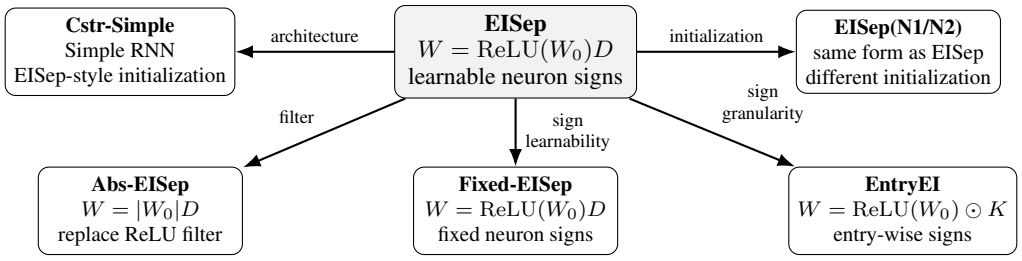


Figure 32: Schematic comparison of EISep and its ablation variants. Abs-EISep replaces the ReLU recurrent-weight filter with an absolute-value filter; Fixed-EISep removes learnable neuron signs; EntryEI relaxes neuron-level E–I separation into entry-wise sign masks; EISep(N1/N2) preserves the EISep parameterization but changes the initialization scheme; and Cstr-Simple tests whether EISep-style initialization alone explains the observed gains.

U.1 Absolute as the recurrent weight filter of EISep

There remains a question as to whether the sparsity completely arises from the ReLU operator (Eq. 3) and is independent from the performance. To disentangle these effects, we replace the ReLU nonlinearity with an absolute value operator: $W = |W_0|D$ and refer to the new variant as **Abs-EISep**. Under this modification, the sparsity advantage indeed vanishes (Fig. 35E). However, the performance is also affected. While most tasks show little or no degradation, performance drops significantly on two relatively more challenging matching tasks—DMS and DNMS (Fig. 35A), which require comparing two angles randomly sampled from $[0°, 360°)$. This effect is also reflected in the averaged performance across tasks (Fig. 35B), which decreases from 0.97 for EISep to 0.93 for Abs-EISep (paired t-test, $p = 1.2 \times 10^{-5}$). Moreover, Abs-EISep exhibits an unbounded growth in spectral radius (Fig. 35C right), as well as reduced performance in continual learning (Fig. 38J) and reinforcement learning (Fig. 34), indicating unstable recurrent dynamics during training. Together, these results demonstrate that the ReLU operator plays a critical role not only in inducing sparsity, but also in stabilizing recurrent dynamics and supporting performance on challenging tasks. This suggests that sparsity of EISep is not an incidental side effect of the parameterization, but is associated with performance through the ReLU-based constraint on recurrent weights.

U.2 EISep with fixed neuron signs

Although we have shown in Sec. N that the neuron signs have converged at the early stage of training, it is still a question whether allowing the signs to be learnable is itself beneficial, or the early convergence is the only side effect of the learnable sign. To address this question, we randomly sample the sign matrix D (Eq. 3) and fix it throughout the training. This variant, referred to as **Fixed-EISep** achieves performance, sparsity, and training dynamics comparable to those of the learnable EISep model on the multi-task cognitive benchmark (Fig. 35B), indicating that E/I separation alone is sufficient for solving these relatively simple tasks. However, Fixed-EISep performs worse than EISep in continual learning (Fig. 38J) and the reinforcement learning task (Fig. 34), suggesting that a fixed sign structure lacks the flexibility required to capture more intricate dynamics and to support complex control behaviors.

U.3 Entry-wise masks for the EISep recurrent weight

Another question is whether EISep's advantages stem from the specific *neuron-level* E-I separation induced by Dale's principle, or more generally from imposing value-sign separation on weights. Therefore, we introduce **EntryEI**, which relaxes the column-wise E-I constraint of EISep to an entry-wise one.

Here is the comparison of the designs. The original EISep enforces a consistent outgoing sign for each presynaptic neuron through a diagonal sign matrix D (Eq. 3)

$$W = \mathrm{ReLU}(W_0)D, \qquad D_{ii} \in \{-1, 1\}. \tag{27}$$

In contrast, EntryEI applies an independent sign constraint to each individual weight,

$$W = \mathrm{ReLU}(W_0) \odot K, \qquad K_{ij} \in \{-1, 1\}, \tag{28}$$

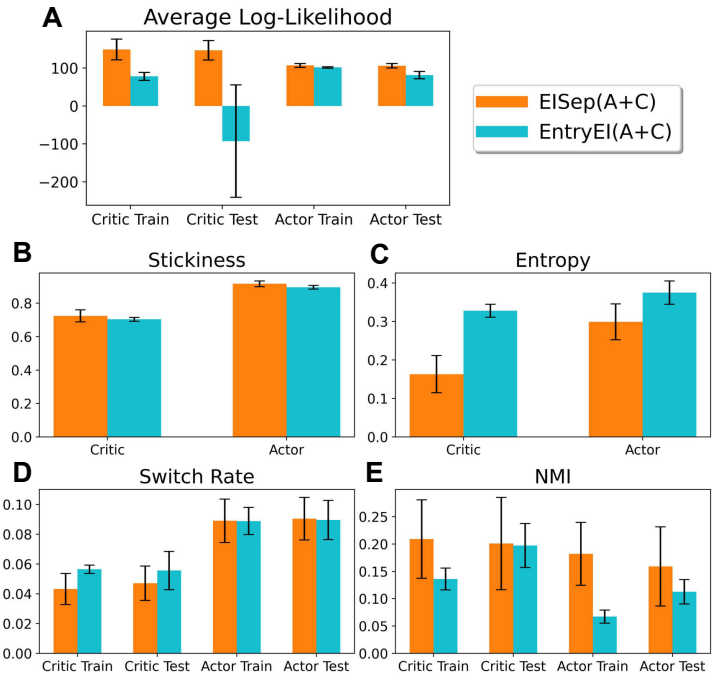


Figure 33: HMM metrics for EntryEI(A+C), including (**A**) log-likelihood, (**B**) stickiness of transition matrices, (**C**) entropy of transition matrices, (**D**) switch rate of HMM labels, and (**E**) normalized mutual information between HMM labels and handmade labels. Same as Fig. 31

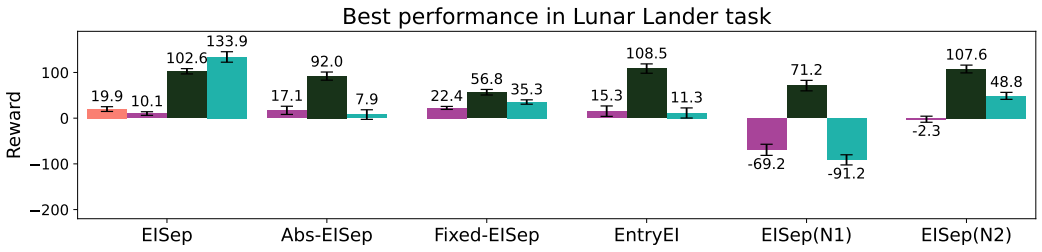


Figure 34: Performance in the Lunar Lander task for different ablation models. The performance of Cstr-Simple has been shown in Sec. 6.

where K is a dense matrix and $\odot$ denotes element-wise multiplication.

This modification isolates the role of neuron-level sign consistency while increasing the number of free parameters, allowing each recurrent connection to independently select its excitatory or inhibitory sign. From the structural viewpoint, EntryEI therefore represents a similar but more powerful and less constrained model class than EISep. Empirically, it also achieves a similar structural advantage with EISep: high sparsity (Fig. 35E), indicating that sparsity alone can be recovered without neuron-level E-I separation. However, EntryEI consistently underperforms EISep on several cognitively demanding tasks, including DMS, DNMS, Dly DM, and Ctx Dly DM (Fig. 35A) in the multi-task cognitive benchmark. It also exhibits narrower stable learning-rate regimes (Fig. 36), inferior RL performance (Fig. 34) and reduced dynamic coherence (Fig. 33) in reinforcement learning.

U.4 EISep with different initialization

Here we show how the initialization method affects the task performance of EISep. We adapt the method from ColEI [20] and DBPN [18] to initialize the value matrix Q and the E-I logit matrix M. The initialization of Q strictly follows the original implementations E. However, we have learnable E-I identities while they determine signs at the beginning. To address this gap, we assign $M_i = [1.0, 0.0]$

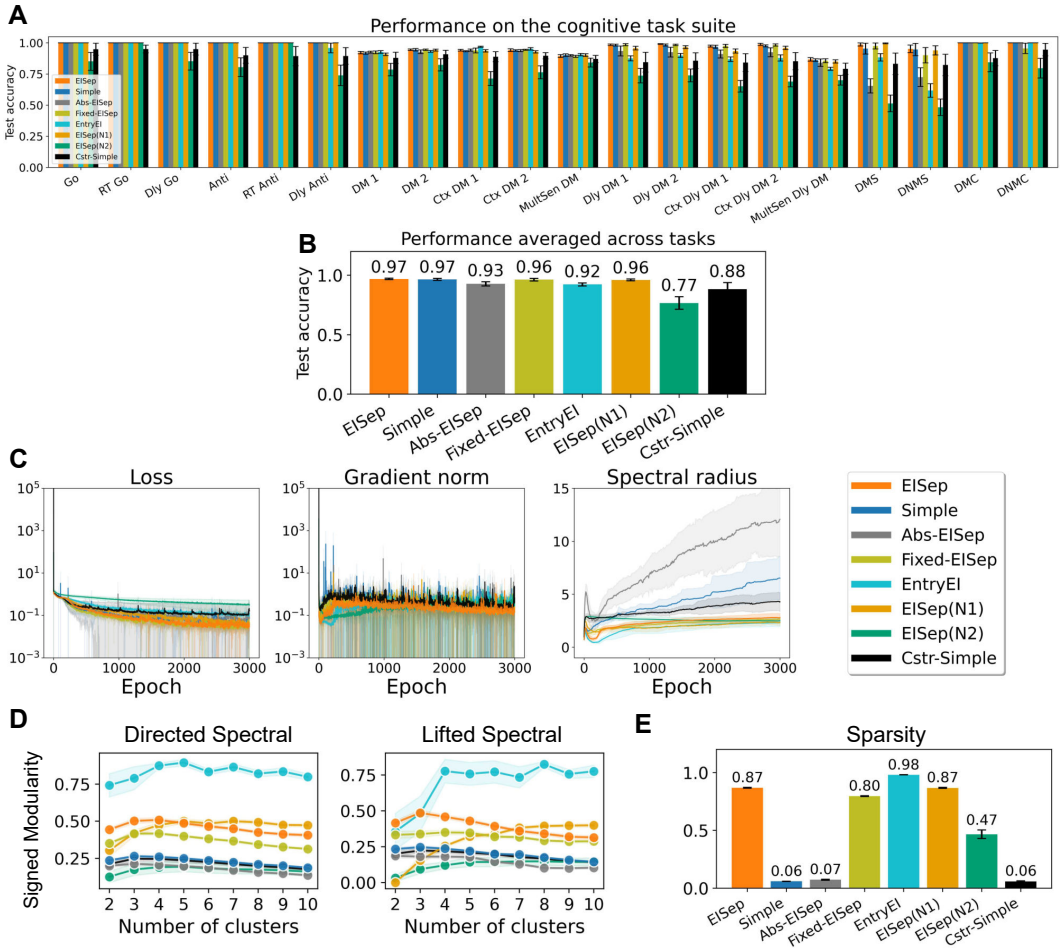


Figure 35: Metrics on the cognitive task suite for different ablation models. We compare Abs-EISep, Fixed-EISep, and EntryEI against the EISep and Simple baselines. **A**. Task-wise performance of the 5 models across 20 cognitive tasks. **B**. Performance averaged across all tasks for each model. **C**. Recurrent weight sparsity averaged across tasks for different models. **D**. Training dynamics over the full training process, including decoding loss (left), gradient norm (middle), and spectral radius of the recurrent weight matrix (right).

to each excitatory neuron i and $M_j = [0.0, 1.0]$ to each inhibitory neuron j, and calculate sign matrix D as before (Sec. 3.2). EISep with ColEI-style initialization is denoted "**EISep(N1)**", and EISep with DBPN-style initialization is denoted "**EISep(N2)**". Since DANN implements Dale's principle using a fundamentally different structure with three coupled matrices (Appendix E), we do not include it in this comparison.

On the multi-task cognitive benchmark, EISep(N1) achieves similar average performance and sparsity (Fig. 35B, E), but exhibits a narrower stable learning-rate zone (Fig. 36) and lower continual learning (Fig. 35J) and RL performance (Fig. 34). EISep(N2) cannot match EISep on most evaluation metrics. We further find these initialization methods yield larger spectral radii (Fig. 37), consistent with their degraded performance. These results suggest that the proper initialization method of EISep constrains the spectral radius and improves training stability and overall performance.

U.5 Simple with EISep-style initialization

To test whether EISep's advantage stems from initialization alone, we initialize Simple using the same scheme and denote the resulting variant as **Cstr-Simple**. This model still fails to match EISep

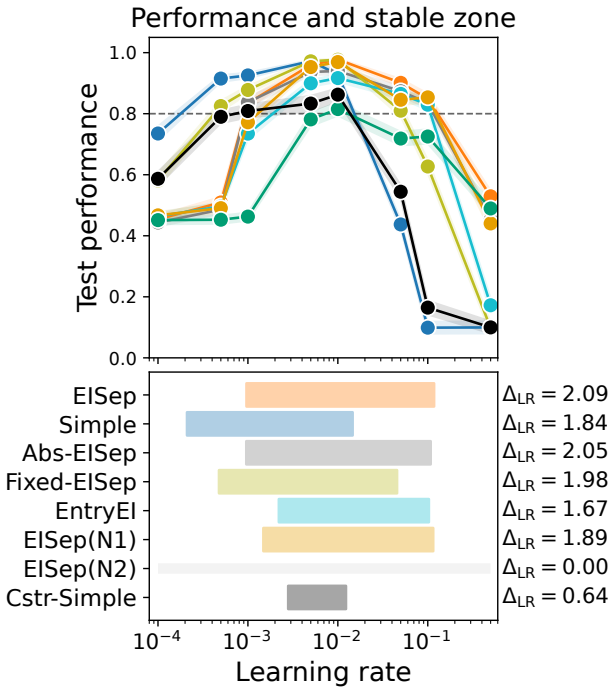


Figure 36: Performance and stable zones for different ablation models vs. learning rates in the cognitive task suite.

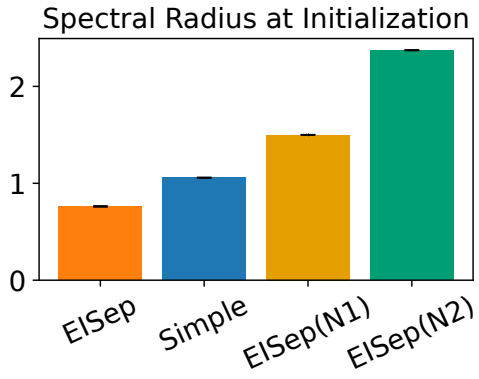


Figure 37: The spectral radius of different models at initialization ($n = 1000$, Error bar indicates standard error).

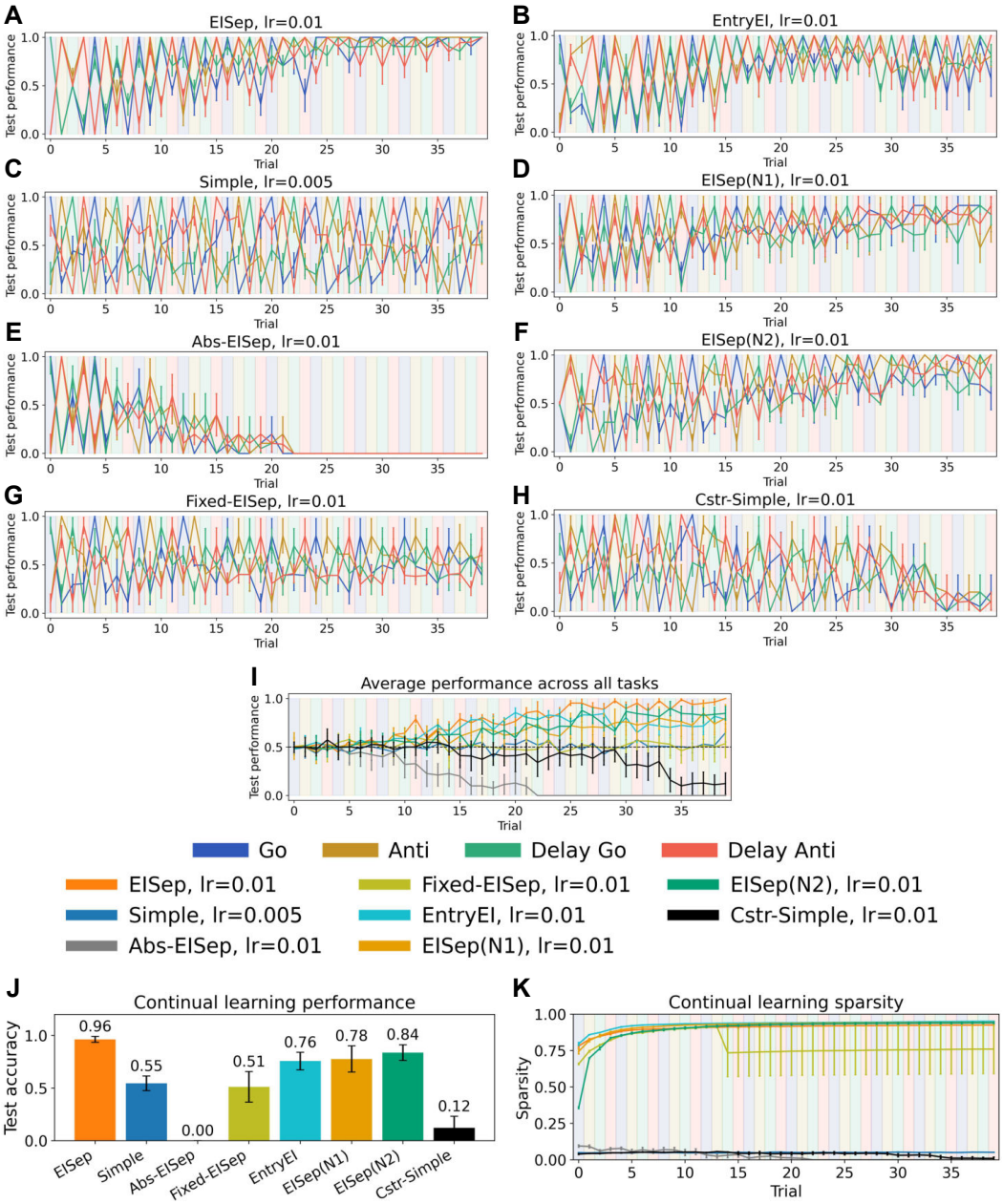


Figure 38: **A–H.** Training curves of different ablation models with their best learning rates in continual learning. **I.** Average performance across tasks. **J.** Final-round performance averaged over four trials. **K.** The sparsity of all models across trials.

across all three tasks, and even underperforms Simple on multi-task (Fig. 36) and continual learning (Fig. 38J). See Fig. 7F for RL performance. These results suggest that EISep-style initialization alone does not account for the observed gains, and that initialization strategies may not transfer directly across different model architectures.

Together, these results demonstrate that sparsity and modularity alone are insufficient to account for the advantages observed in EISep, and that increased parameterization does not necessarily translate into improved functional performance. More broadly, this comparison highlights that biologically motivated constraints can outperform less realistic but more flexible alternatives, even when the latter possess a larger hypothesis space. The failure of EntryEI suggests that less constrained synaptic sign organization—while more expressive—may hinder the emergence of stable, interpretable, and task-relevant dynamics. In contrast, EISep's neuron-level E-I separation provides a favorable inductive bias that supports robust learning and generalization across both cognitive tasks and reinforcement learning.

V Critic actor asymmetry

An interesting emergent phenomenon in experiments is worth noting. In RL tasks, EISep parameterization exhibits different effects on the critic and actor. EISep(C) outperforms EISep(A) across nearly all scenarios, and EISep(A) often performs worse than Simple (Fig. 6B, C, Fig. 25, Fig. 26, Fig. 38). This advantage is unlikely to be explained solely by the higher learning rate of critic, since this advantage still exists even when the critic has a lower learning rate than the actor (Sec. P.2). Notably, a similar asymmetry has also been discovered for other inductive biases like modularity [13] where modularizing the critic produces more benefits. This result suggests that Dale's principle, as an inductive bias to improve stability and modularity, aligns more closely with the critic's computation, like value estimation or belief update, than with the actor's computation, like flexible and context-dependent control. From the machine learning perspective, the critic's bootstrapped value updates introduce strong correlations between predicted and target values [49], forming a self-referential feedback loop that is a well-known source of instability [60, 61]. We assume that EISep acts as a dynamical regularizer that mitigates this bootstrapping-induced instability. In contrast, the actor optimizes a supervised-like policy objective without bootstrapping and is therefore less susceptible to such instabilities. Enforcing EISep constraints may thus unnecessarily restrict the actor. EISep(A+C) retains the benefit of the EISep critic but additionally constrains the actor, explaining why it can underperform EISep(C). This aligns with observations in supervised learning tasks (Appendix M). However, strict control is needed to confirm this conjecture.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state that the paper proposes EISep, as a Dale's principle–based inductive bias, that improves stability, modularity, sparsity, and performance across multi-task, continual learning, and reinforcement learning settings. These claims are consistently supported by experiments and analyses throughout the paper (Secs. 4–6). The scope is appropriately bounded—e.g., the paper does not claim universal superiority and explicitly discusses trade-off, task dependence and limitations in Sec. 7—so the claims accurately reflect both contributions and realistic generalization.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly discusses several limitations in Sec. 7, including reduced performance at lower learning rates, and the task-dependent nature of the inductive bias. It also acknowledges that many biological details are not modeled and that benefits may diminish in settings without instability or interference problems, reflecting a clear and appropriate discussion of scope and assumptions.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper does not include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient detail to reproduce the main results: it clearly specifies the EISep architecture (Sec. 3.2), describes training objectives and regularization, and all hyperparameters (Appendix F). It outlines experimental setups across multi-task, continual learning, and RL benchmarks (Sec. 4–5). It also reports other key implementation choices, such as learning rate selection (Sec. 5.1), task settings (Sec. 5), EISep-LSTM setup (Appendix R), and evaluation metrics (Appendix K). It also releases the full code for training and evaluation. All of these enable reproducibility of the core experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and instructions are anonymized and provided in the supplementary materials, which satisfy the requirement for open access and enable faithful reproduction of all experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setting, including model setup, training objectives, and task settings, has been presented in the main text (Sec. 3–5). The paper provides detailed experimental settings in Appendix F, including network architecture, initialization, optimizer, training procedures, hyperparameters (learning rates, batch size, sequence length), and evaluation protocols for multi-task, continual learning, and reinforcement learning experiments, which are sufficient to understand and reproduce the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports results averaged over multiple random seeds (and episodes in RL) and explicitly reports the mean and standard error across seeds. The calculating method is provided in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides detailed compute information in Appendix F.5, including hardware specifications (CPU, GPU, memory), per-experiment training time for multi-task, continual learning, and reinforcement learning, as well as an estimate of total compute usage, which is sufficient for reproducing the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research complies with the NeurIPS Code of Ethics: it does not involve human subjects, sensitive data, or privacy concerns, and relies on standard benchmark environments. Potential societal impacts are discussed in the Impact Statement (Appendix A), which notes that while the method itself does not introduce new data or surveillance capabilities, improved robustness may have context-dependent downstream effects depending on deployment.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper includes an Impact Statement that discusses both positive impacts (e.g., improved stability, interpretability, and applications in robotics, scientific modeling, and neuroscience) and potential negative or dual-use implications, noting that increased robustness may amplify both beneficial and harmful uses depending on deployment context. It also acknowledges that broader impacts depend on downstream applications and outlines future directions related to safety and fairness.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk artifacts such as large pre-trained models, generative systems, or sensitive datasets. It proposes a neural network architecture evaluated on standard benchmarks, which does not pose significant misuse risks requiring specific safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites all external assets, and Appendix F.6 lists their licenses, copyright information, and URLs, confirming that all resources are used in accordance with their respective terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The paper introduces a new model architecture (EISep) and provides detailed documentation of its design, parameterization, and training procedure in Sec. 3, along with implementation details and reproducibility instructions in Appendix F. The code is provided in anonymized form, with sufficient information on usage, training, and configuration to reproduce the results.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects. All experiments are conducted on synthetic tasks and standard benchmark environments without human data or participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects; all experiments are conducted on synthetic tasks and standard benchmark environments, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methods, experiments, and analyses in this paper do not involve LLMs as part of the model, algorithm, or experimental pipeline. The use of LLMs in this paper is limited to writing or editing and does not affect the scientific content (Sec. C).

    Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.