The Cue or not the Cue? A Mechanistic Study of Memory Mechanisms in RNNs

Mehmet Harmanli *KUIS AI, Koc University, Istanbul, Turkey*
Fatih Dinc *Kavli Institute of Theoretical Physics, Santa Barbara, USA*
Peng Yuan *Fudan University, Shanghai, China*
Yucel Yemez[*] *KUIS AI, Koc University, Istanbul, Turkey*
Bariscan Kurtkaya[*] *KUIS AI, Koc University, Istanbul, Turkey*

Abstract

Neural networks can solve behavioral tasks requiring memory either by remembering the full content or through active manipulation that retains a simplified version. Yet, distinguishing between these two memory retention mechanisms in recurrent neural networks (RNN) remains underexplored. To bridge this gap, we studied RNNs performing delayed cue discrimination (DCD) tasks and asked whether they retain raw continuous-valued input cues or their task-relevant binary representations. Using linear probes trained on neural activities during the delay period, we tested whether RNNs eventually collapse the retained cue values into compact, binary representations. Even though RNNs were trained only using binary cues, we consistently observed high reconstruction fidelity of continuous cue inputs across diverse experimental conditions and learned memory mechanisms. Overall, our results provide evidence that RNNs can find solutions preserving the contents of past memories with high fidelity, favoring representational completeness over efficiency, even when not demanded by the task.

Keywords: RNNs, Computational Neuroscience, Memory Mechanisms

Recurrent neural networks (RNNs) have become a central tool in computational neuroscience for studying neural circuitry and circuit–behaviour relationships (Finkelstein et al., 2021; Mante et al., 2013; Yang et al., 2019; Masse et al., 2019; Dubreuil et al., 2022; Perich and Rajan, 2020; Barak, 2017; Dinc et al., 2025). A common approach is to train RNNs either as virtual twins of neural recordings or on neuroscience-inspired tasks, using them as model systems to explore the dynamics that might underlie neural computation (Perich et al., 2021; Valente et al., 2022; Dinc et al., 2023; Kurtkaya et al., 2025). To interpret the resulting networks, researchers have turned to dynamical systems theory, geometry, and topology, revealing rich structure in how RNNs evolve over time (Sussillo and Barak, 2013; Langdon et al., 2023; Perich et al., 2025). These analyses have shed light on the mechanisms by which RNNs maintain information across delays (Rajan et al., 2016; Kurtkaya et al., 2025), yet a fundamental question remains unresolved: do these networks retain the full stimulus, or do they compress it into the minimal representation required for the task?

In cognitive science, a similar question is well studied in the sense of short-term memory (STM) and working memory (WM) distinction. Mainly, STM refers to passive retention of information, whereas WM involves active manipulation of the data into task-relevant codes (Baddeley and Hitch, 1974; Cowan, 2008, 2014; D'Esposito, 2007; Aben et al., 2012; Engle et al., 1999; Goldman-Rakic, 1995). Although these two processes can support similar behaviors, they imply very different internal representations. A related question emerges in

[*] Co-supervision

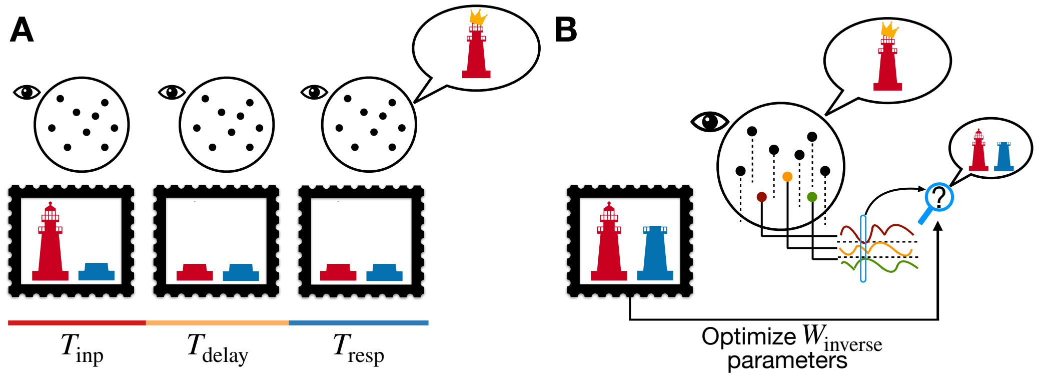

Figure 1: **Overview of our experimental setup. A)** A recurrent neural network trained to perform the delayed cue discrimination task needs to remember and output the presented (red) cue information after a brief delay. **B)** To probe the memory retention capabilities of the trained network, we first present a non-zero value in the dormant (blue) cue, and then reconstruct it with a linear probe trained on the neural activities during the delay period.

artificial neural networks: RNNs trained to solve delayed tasks can achieve high performance using diverse internal mechanisms (Kurtkaya et al., 2025). Yet, despite this diversity, prior work has focused mainly on the dynamics of how information is maintained, rather than on what form of information is being stored. In other words, even when an RNN solves a memory task successfully, it still remains unclear whether it is retaining the raw stimulus values or collapsing them into abstract decision codes.

In this work, we investigate whether RNNs trained on delayed cue discrimination (DCD) task preserve the full information of the cue or collapse it into a binary decision variable. To answer this, we introduce a simple mechanistic probe: a linear decoder applied to hidden activity during the delay period to test whether continuous-valued cue inputs can be recovered. Although the task only requires binary discrimination and thus could be solved by collapsing stimulus values into categorical representations, we consistently find the opposite: across hyperparameters, network sizes, and different dynamical regimes, RNNs retain high-fidelity representations of the original continuous cue values throughout the delay period. This result suggests that, rather than favoring efficient or minimal representations, RNNs overwhelmingly favor representational completeness. Such behavior has important implications for both neuroscience and machine learning: it highlights that memory solutions in trained RNNs may reflect inductive biases toward preserving detailed information, rather than collapsing it. These findings raise questions about how memory representations emerge in biological circuits and how artificial systems might be guided toward more efficient codes.

Methods. Our dataset (Table S1) combines firing-rate trajectories obtained from publicly available RNNs trained on the DCD task (Kurtkaya et al., 2025) with additional models of smaller neuron counts that we trained to extend the parameter range (Figure 1A). The networks use update rule:

$$\tau \dot{r}(t) = -r(t) + \tanh\left(W^{\text{rec}} r(t) + W^{\text{in}} u_t + b + \epsilon\right), \tag{1}$$

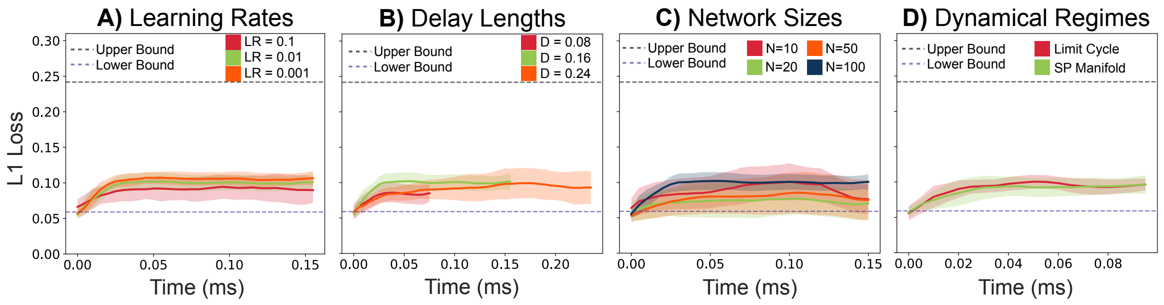

Figure 2: **The dormant cue value can be reconstructed during the delay window across diverse hyperparameters and learned mechanisms.** Shown are L1 reconstruction errors of separately trained linear probes (same architecture and training settings), applied at each timestep to RNNs trained with different **A)** learning rates, **B)** delay lengths, **C)** network sizes, and **D)** dynamical regimes. For detailed model parameters, see Table S1.

where $r(t) \in \mathbb{R}^N$ denotes the hidden firing rates, τ the time decay constant, W^{rec} the recurrent weight matrix, W^{in} the input weight matrix, $b \in \mathbb{R}^N$ the bias term, $u_t \in \mathbb{R}^2$ the external cue input, and ϵ small additive noise.

Task: Delayed cue discrimination (DCD) task is a standard paradigm for probing memory in neuroscience (Fuster and Alexander, 1971). Each training trial begins with an input phase (T_{inp}) presenting a binary cue $(1,0)$ or $(0,1)$, followed by a delay phase (T_{delay}), a reaction phase (T_{resp}) in which the network must reproduce the cue. While Kurtkaya et al. (2025) also considers an optional post-reaction phase (T_{post}), we are not interested in this time interval in this work. Formally, the inputs (u_t) and outputs ($\hat{o}_t$) are

$$u_t = \begin{cases} (1,0) \text{ or } (0,1), & t \in T_{\text{inp}}, \\ (0,0), & \text{otherwise}, \end{cases} \qquad \hat{o}_t = \begin{cases} u_{T_{\text{inp}}}, & t \in T_{\text{resp}}, \\ (0,0), & \text{otherwise}. \end{cases} \tag{2}$$

While the RNNs were trained on binary cues, during probing we supplied real-valued inputs for the dormant cue (e.g., 0.1–0.6 instead of 0) to differentiate the memory encoding and maintenance strategies employed by the networks; notably, this modification did not affect the final outputs produced by the RNNs.

Linear probe: To examine the information stored in the RNN during the delay, we attach a *linear model* to the hidden state. At each timestep t, the hidden activity $\mathbf{h}_t \in \mathbb{R}^N$ is mapped to a two-dimensional cue prediction by a linear layer with sigmoid activation (Figure 1B):

$$\tilde{u}_t = \sigma(W_t \mathbf{h}_t + b), \quad W_t \in \mathbb{R}^{2\times N}, \ b \in \mathbb{R}^2. \tag{3}$$

A decoder is trained independently at each timestep using the L1 loss between $\tilde{u}_t$ and the true cue $u_{T_{\text{inp}}}$, without updating the RNN weights. This approach tests whether the RNNs internal states retain raw memories of the inputs or their binary representations.

To interpret the performance of the linear probe, we established two reference points (Figure S2). The *lower bound* was obtained by training the decoder on hidden states during the input phase, when cue information is explicitly present; near-perfect reconstruction is therefore expected (see Appendix S1.2). The *upper bound* was derived from synthetic random activity that preserved only binary cue identity, but not the continuous values presented (see Appendix S1.3). Together, these bounds provide a reference range for evaluating the representation of memory information during the delay period.

Results. To test the memory retention strategies in RNNs, we trained linear probes independently at each timestep of the delay interval and tested them on held-out trials (Figure 1B). This approach allowed us to assess not only whether the continuous value of the cue remained maintained, but also how reconstruction fidelity evolved over the delay period. Our key finding was that linear probes consistently achieved high reconstruction accuracy across all tested conditions, indicating that RNNs retained detailed input information rather than abstract task codes (Figures 2, S1). This pattern held remarkably robust across multiple variables. Training dynamics had no impact: networks trained with different learning rates (0.1, 0.01, 0.001) all preserved cue traces equally well once task performance converged (Figures 2A, S1A). Temporal demands were also irrelevant: longer delays (0.24ms vs. 0.08ms) did not degrade reconstruction quality, demonstrating stable internal representations throughout the memory period (Figures 2B, S1B). Network architecture showed similar invariance: even small networks (10–20 neurons) supported near-perfect decoding, with larger networks (50–100) providing no additional advantage (Figures 2C, S1C). Most strikingly, this pattern transcended the types of solutions learned, *i.e.*, RNNs with both limit cycles showing oscillatory dynamics and slow-point manifolds showing near-stationary attractors preserved detailed cue information, converging on the same short-term memory strategy (Figures 2D, S1D).

Conclusion. In our experiments, RNNs trained on the DCD tasks consistently adopted to preserve the continuous input traces, not just the binary task outputs. This pattern has held robustly across training conditions, network architectures, and dynamical regimes. Remarkably, RNNs have spontaneously developed these richer representations despite having no explicit training objective to preserve continuous cue values. So that, networks could have simply collapsed input information into binary decision codes, instead they learned to retain detailed information.

This finding challenges conventional assumptions about computational efficiency in neural networks (Sussillo et al., 2015). Rather than adopting the minimal representation necessary for task success, RNNs trained on the DCD tasks appear to have favored full information preservation. This preference for representational completeness has broader implications for both neuroscience and machine learning, suggesting the need for a new line of research studying whether real memory systems may have implicit biases for retaining the full STM content by default.

Finally, the method itself offers a straightforward tool for diagnosing representational strategies in neural networks. By testing whether networks maintain raw sensory traces or task-relevant outputs, researchers can better understand the computational principles underlying memory in both artificial and biological systems. Future work should explore whether more complex tasks or architectural constraints encourage working memory-like abstraction, and how these mechanisms generalize across different domains and network types.

References

Bart Aben, Sven Stapert, and Arjan Blokland. About the distinction between working memory and short-term memory. *Frontiers in Psychology*, Volume 3 - 2012, 2012. ISSN 1664-1078. doi: 10.3389/fpsyg.2012.00301. URL https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2012.00301.

Alan D. Baddeley and Graham Hitch. Working memory. In *Psychology of Learning and Motivation*, volume 8, pages 47–89. Academic Press, 1974. doi: 10.1016/S0079-7421(08)60452-1.

Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1–6, 2017. ISSN 0959-4388. doi: https://doi.org/10.1016/j.conb.2017.06.003. URL https://www.sciencedirect.com/science/article/pii/S0959438817300429. Computational Neuroscience.

Nelson Cowan. What are the differences between long-term, short-term, and working memory? *Progress in Brain Research*, 169:323–338, 2008. doi: 10.1016/S0079-6123(07)00020-9.

Nelson Cowan. Working memory underpins cognitive development, learning, and education. *Educational Psychology Review*, 26(2):197–223, 2014. doi: 10.1007/s10648-013-9246-y. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC4084861/.

Mark D'Esposito. From cognitive to neural models of working memory. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1481):761–772, 2007. doi: 10.1016/S0079-6123(07)00020-9.

Fatih Dinc, Adam Shai, Mark Schnitzer, and Hidenori Tanaka. CORNN: Convex optimization of recurrent neural networks for rapid inference of neural dynamics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=GGIA1p9fDT.

Fatih Dinc, Marta Blanco-Pozo, David Klindt, Francisco Acosta, Yiqi Jiang, Sadegh Ebrahimi, Adam Shai, Hidenori Tanaka, Peng Yuan, Mark J Schnitzer, et al. Latent computing by biological neural networks: A dynamical systems framework. *arXiv preprint arXiv:2502.14337*, 2025.

Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. The role of population structure in computations through neural dynamics. *Nature Neuroscience*, pages 1–12, 2022.

Randall W Engle, Stephen W Tuholski, James E Laughlin, and Andrew RA Conway. Working memory, short-term memory, and general fluid intelligence: a latent-variable approach. *Journal of experimental psychology: General*, 128(3):309, 1999.

Arseny Finkelstein, Lorenzo Fontolan, Michael N Economo, Nuo Li, Sandro Romani, and Karel Svoboda. Attractor dynamics gate cortical information flow during decision-making. *Nature Neuroscience*, 24(6):843–850, 2021.

Joaquin M Fuster and Garrett E Alexander. Neuron activity related to short-term memory. *Science*, 173(3997):652–654, 1971.

Patricia S Goldman-Rakic. Cellular basis of working memory. *Neuron*, 14(3):477–485, 1995.

Bariscan Kurtkaya, Fatih Dinc, Mert Yuksekgonul, Marta Blanco-Pozo, Ege Cirakman, Mark Schnitzer, Yucel Yemez, Hidenori Tanaka, Peng Yuan, and Nina Miolane. Dynamical phases of short-term memory mechanisms in RNNs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ybBuwgOPOd.

Christopher Langdon, Mikhail Genkin, and Tatiana A Engel. A unifying perspective on neural manifolds and circuits for cognition. *Nature Reviews Neuroscience*, 24(6):363–377, 2023.

Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474): 78–84, 2013.

Nicolas Y Masse, Guangyu R Yang, H Francis Song, Xiao-Jing Wang, and David J Freedman. Circuit mechanisms for the maintenance and manipulation of information in working memory. *Nature neuroscience*, 22(7):1159–1167, 2019.

Matthew G Perich and Kanaka Rajan. Rethinking brain-wide interactions through multi-region 'network of networks' models. *Current opinion in neurobiology*, 65:146–151, 2020.

Matthew G Perich, Charlotte Arlt, Sofia Soares, Megan E Young, Clayton P Mosher, Juri Minxha, Eugene Carter, Ueli Rutishauser, Peter H Rudebeck, Christopher D Harvey, et al. Inferring brain-wide interactions using data-constrained recurrent neural network models. *bioRxiv*, pages 2020–12, 2021.

Matthew G Perich, Devika Narain, and Juan A Gallego. A neural manifold view of the brain. *Nature Neuroscience*, pages 1–16, 2025.

Kanaka Rajan, Christopher D Harvey, and David W Tank. Recurrent network models of sequence generation and memory. *Neuron*, 90(1):128–142, 2016.

David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation*, 25(3):626–649, 2013.

David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature neuroscience*, 18(7):1025–1033, 2015.

Adrian Valente, Jonathan W Pillow, and Srdjan Ostojic. Extracting computational mechanisms from neural data using low-rank rnns. *Advances in Neural Information Processing Systems*, 35:24072–24086, 2022.

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.

Appendix S1. Implementation details

S1.1. Training of the Linear Probe

The linear probe was trained on firing-rate activity generated by RNNs during inference. At each timestep of the delay interval, the probe received the firing rate activity as input and was tasked with reconstructing the original cue from the input phase (Figure 1B). Training data included trials with a fixed high cue value of 1 and a variable lower value (e.g., $0.1, 0.2, \dots, 0.6$).

The probe was implemented as a single linear layer mapping from the hidden dimension (N neurons) to the two cue channels, followed by a sigmoid nonlinearity (Figure 1B). Each timestep was trained independently using an L1 loss between predicted and true cues. The RNN parameters remained frozen throughout; only the probe weights were optimized.

For reproducibility, all probes were trained for 50,000 epochs using the Adam optimizer with a learning rate of 3×10^{-4} and weight decay of 1×10^{-7} (other parameters at PyTorch defaults). An additional L1 regularization term with coefficient 0.001 was applied. Training was performed on CPU (Apple M1 chip).

S1.2. Lower Bound

The lower bound was defined by training and evaluating the linear probe on firing rates recorded during the input phase rather than the delay phase of an RNN. Because the cue is explicitly present at this stage, the network's hidden state contains complete and direct information about the stimulus. As a result, accurate reconstruction by the inverse model is expected, and the performance achieved here serves as a practical reference point (Figure S2).

This lower bound thus represents the best-case scenario for decoding: the probe has access to firing-rate patterns that directly reflect the presented cue, with no memory maintenance required. By comparing delay-period reconstructions against this benchmark, we can quantify how much information about the cue persists once external input is removed. In essence, the lower bound anchors probe performance, allowing us to interpret whether successful decoding during the delay indicates faithful preservation of input traces or whether information has degraded or transformed over time.

S1.3. Upper Bound

To establish an upper bound, we generate synthetic neural activity that preserves only the dominant cue's categorical identity. Specifically, we construct the activity vector by setting half of the neurons to a constant value of one, with the position indicating which cue is dominant: if the dominant cue is $(1, 0)$, the first half is set to one; if the dominant cue is $(0, 1)$, the second half is set to one. The remaining half is sampled from a standard Gaussian distribution. We then apply the tanh nonlinearity to bound the overall activity to $[-1, 1]$:

$$r_{\text{synthetic}}(t) = \begin{cases} \tanh\left([\mathbf{1}_{N/2},\, z(t)]\right) & \text{if dominant cue is } (1,0), \\ \tanh\left([z(t),\, \mathbf{1}_{N/2}]\right) & \text{if dominant cue is } (0,1), \end{cases} \qquad z(t) \sim \mathcal{N}(0, I_{N/2}), \tag{S1}$$

where $\mathbf{1}_{N/2} \in \mathbb{R}^{N/2}$ denotes a vector of ones and $[\cdot, \cdot]$ denotes concatenation. This design ensures that only abstract categorical information is preserved, while precise amplitude information is lost. As a result, the inverse model cannot recover detailed input values from such activity (Figure S2). Performance under this condition thus provides a ceiling: if delay-period decoding approaches this level, it implies that RNNs reduce memory to categorical abstractions; if performance remains closer to the lower bound, the networks instead preserve detailed stimulus traces.

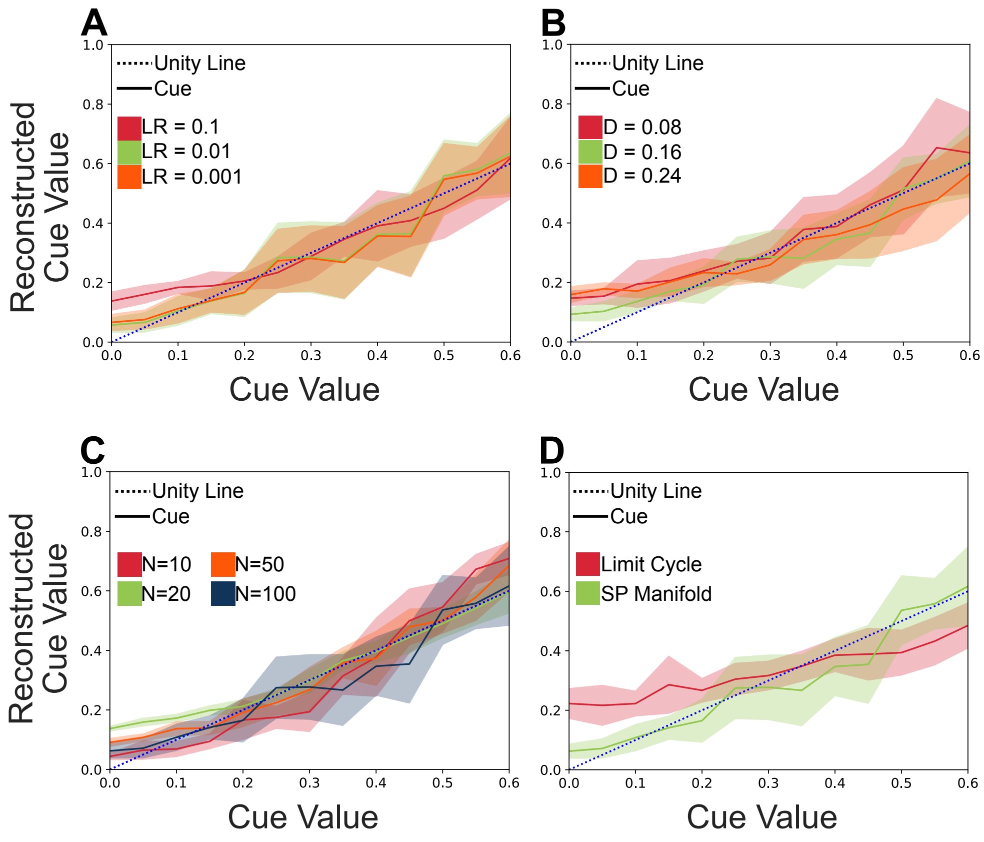

Figure S1: **Linear probes reveal robust recovery of cue values.** Reconstruction accuracy remains stable across changes in **A)** learning rate, **B)** delay duration, **C)** network size, and **D)** Dynamical Regimes (slow manifolds vs. limit cycles), indicating that networks preserve cue information regardless of how they are trained or what memory mechanism they learn.

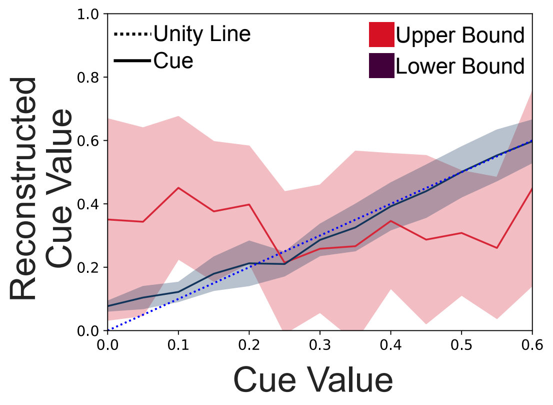

Figure S2: **A linear probe reveals what information is present in hidden states.** We validate the probe by testing it on firing rates during the input period (lower bound), where the full cue is guaranteed to be available and thus correctly decoded. As a contrast, we apply the same probe to synthetic firing rates that encode only which cue is dominant, but contain no information about the dormant cue (upper bound), demonstrating that the probe cannot recover information that is not represented in the activity.

Figure	**Neuron Count**	**Parameters**	**Seeds**
Fig. 2a	100	LR = 0.1, Delay = 0.16	1,3,4,5,6,23,32,39,46,48
	100	LR = 0.01, Delay = 0.16	0,1,2,3,5,6,7,8,11,12
	100	LR = 0.001, Delay = 0.16	0,1,2,3,4,5,6,7,8,11
Fig. 2b	100	LR = 0.01, Delay = 0.08	0,1,2,3,4,5,6,7,8,11
	100	LR = 0.01, Delay = 0.16	0,1,2,3,5,6,7,8,11,12
	100	LR = 0.01, Delay = 0.24	0,1,2,5,7,10,26,38,52,57
Fig. 2c	10	LR = 0.01, Delay = 0.16	1,2,3,5,6,9,10,12,14,17,19
	20	LR = 0.01, Delay = 0.16	0,1,2,3,5,6,7,8,11,12,13,14,15,16,18,19
	50	LR = 0.01, Delay = 0.16	0,1,2,3,4,6,8,10,12,14,16,17,18,19
	100	LR = 0.01, Delay = 0.16	0,1,2,3,5,6,7,8,11,12
Fig. 2d	100	LR = 0.001, Delay = 0.10	0,1,2,3,4,5,6,7,8,9

Table S1: Model parameters and random seeds used for Figure 2. Seeds are chosen such that the counts of Limit Cycle and Slow-Point Manifold regimes are balanced.