LEARNING AND DYNAMICALLY DEPLOYING ORDINAL SCHEMAS FOR SEQUENCE WORKING MEMORY IN RNNS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequence working memory (SWM) requires neural systems to maintain item-specific information while organizing it according to ordinal structure over time. Experiments in the macaque prefrontal cortex have revealed a factorized geometry in which items at different ordinal ranks occupy low-dimensional, approximately orthogonal subspaces. However, how such geometry emerges in recurrent networks and, more importantly, how it is dynamically deployed to support computation across encoding, maintenance, and retrieval remain unclear. Here, we show that a recurrent neural network (RNN) develops a reusable ordinal schema comprising content-independent rank subspaces within which stimulus-specific representations are embedded. Crucially, network analysis reveals stage-dependent alignment of the currently relevant rank subspace with the fixed input- and output-weight spaces during encoding and retrieval, respectively. Meanwhile, neural dynamics within previously encoded subspaces continue to evolve in a structured manner, and activity remains marginally stable during maintenance, balancing memory retention with responsiveness to ongoing computational demands. The learned schema generalizes across tested sequence lengths, while perturbations in a richer stroke-based task remain largely confined to the affected rank, limiting cross-rank interference. These results suggest that rank-specific geometry provides a reusable computational scaffold through which recurrent circuits dynamically support flexible, generalizable sequence processing in both biological and artificial recurrent systems.

1 INTRODUCTION

Intelligent behavior requires neural systems not only to maintain item-specific information over time, but also to organize it according to abstract relational structure (Dehaene et al., 2015; 2022). Sequence working memory (SWM) provides a tractable setting for studying this challenge: *encoded* items must be *maintained* together with their ordinal positions, allowing item-specific content to be manipulated and *retrieved* according to task demands (Fig. 1A) (Baddeley, 2012; Miller et al., 2018).

Recent experiments in macaque prefrontal cortex revealed a striking representational geometry underlying SWM during maintenance (Xie et al., 2022; Chen et al., 2024): item information at different ordinal position is represented in distinct low-dimensional, approximately orthogonal rank subspaces. Each ordinal rank is associated with a distinct subspace that represents all possible item contents at that position—blue for the first rank, green for the second, and purple for the third in Fig. 1B—yielding a factorized code that separates abstract ordinal structure from stimulus identity. Such a factorized organization suggests a potential computational scaffold for preserving item-specific information while flexibly manipulating sequence structure across different sequence instances.

These findings establish the geometry of sequence representations during maintenance, but leave open how this geometry participates in computation over time. In particular, it remains unclear whether the same rank-specific organization can serve as a reusable ordinal schema across encoding, maintenance, and retrieval, and how a recurrent circuit might dynamically configure such a schema to meet the distinct computational demands of each stage. A reusable ordinal scaffold could allow neural

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

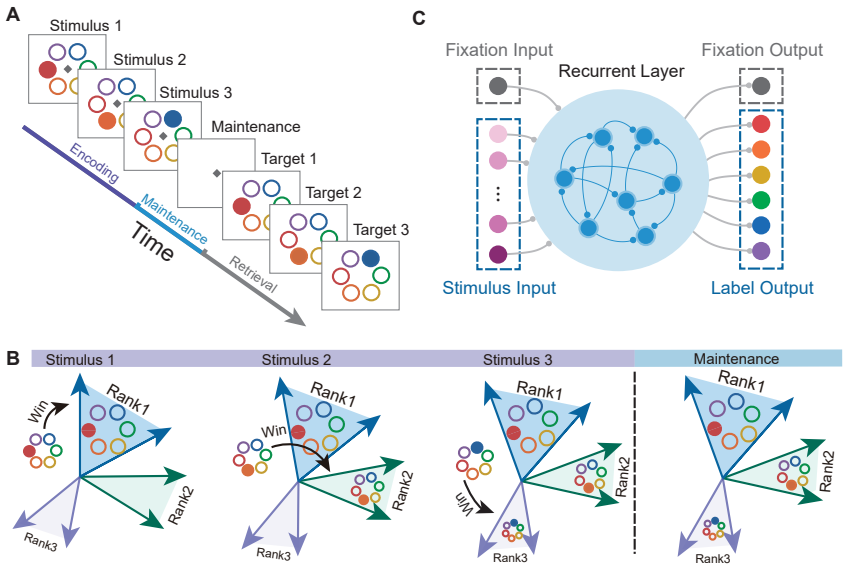

Figure 1: **SWM Task, model architecture, and the ordinal schema hypothesis.** (A) Schematic of the delayed SWM task. A sequence of three spatial locations (colored solid circles drawn from six points on a ring) is visually presented during encoding, maintained across a multi-second delay, and retrieved in order. (B) Schematic of the ordinal schema and the dynamic stage-wise deployment mechanism. Three nearly orthogonal 2D subspaces in neural state space encode ordinal rank, with stimulus identities forming ring-like manifolds within each. During encoding, input-space alignment routes stimuli into rank subspaces, while previously encoded subspaces undergo structured dynamics that preserve information and approach marginal stability during maintenance. (C) Schematic of neural circuit model. The model comprises an input layer conveying item sequences, a recurrent neural network (RNN) abstracting the ordinal schema, and an output layer for sequence retrieval.

systems to preserve, transform, and retrieve information according to abstract sequence structure while generalizing across contents and sequence lengths.

More broadly, studies across sensory and motor systems suggest that neural circuits can reconfigure population geometry across information-processing stages to support different computational demands. In sensory and working-memory tasks, neural activity is transformed from stimulus-aligned representations into memory-related subspaces, reducing interference between ongoing sensory inputs and stored information (Libby & Buschman, 2021; Yang & Yu, 2026; Aoi et al., 2020). A related principle appears in motor control, where neural activity evolves within a preparatory subspace during movement planning and subsequently rotates into an execution-related subspace to generate action (Churchland et al., 2012; Elsayed et al., 2016). Together, these findings suggest that population geometry is not merely a static representational substrate, but a computational resource that can be dynamically reconfigured across processing stages.

Related Works Related works have investigated how neural networks acquire structured representations that separate relational structure from item-specific content, particularly in episodic long-term memory and cognitive-map settings (Whittington et al., 2020; 2025; Zou et al., 2023). In working memory, studies of single-item maintenance in recurrent networks have focused on stable population geometries, including point (Rivkind & Barak, 2017), line (Maheswaranathan et al., 2019), and ring attractors (Driscoll et al., 2024), rather than the ordinal relations among multiple items. More directly relevant to SWM, recent models have shown that trained recurrent networks can develop factorized, approximately orthogonal rank-specific subspaces resembling those observed experimentally (Whittington et al., 2025; Wang et al., 2025). Analyses of learned weights have identified relatively modular circuit motifs for writing, maintaining, and reading information from individual slots (Wang et al., 2025). In contrast, macaque LPFC recordings suggest a more distributed organization, in which rank-specific subspaces are supported by broad, overlapping neuronal populations (Xie et al., 2022). Existing models have also typically employed simplified immediate serial-recall tasks

with single-timestep stimulus presentation and recall, without inter-stimulus intervals (ISIs) or an explicit maintenance period (Whittington et al., 2025; Wang et al., 2025). Such tasks differ from the temporally extended structure of experimental SWM paradigms and limit the mechanistic separation of encoding, maintenance, and retrieval. Consequently, it remains unclear whether rank-specific geometry forms a reusable ordinal organization across processing stages. More fundamentally, how can a recurrent network with fixed connectivity dynamically configure such distributed geometry to meet the distinct computational demands of encoding, maintenance, and retrieval?

Here, we address these questions using a recurrent network trained on a temporally extended SWM task with ISIs and distinct encoding, maintenance, and retrieval periods, allowing us to track the evolution of rank-specific geometry across processing stages. The network develops a reusable ordinal schema comprising low-dimensional, approximately orthogonal rank subspaces within which stimulus-specific representations are embedded. Network analysis reveals how the learned ordinal geometry is dynamically configured across task stages. During encoding, the currently relevant rank subspace selectively aligns with the fixed input-weight space, routing incoming information into the appropriate ordinal representation; targeted input perturbations preferentially disrupt the corresponding rank, providing functional evidence for this routing mechanism. As new stimuli arrive, previously encoded subspaces continue to evolve through structured recurrent dynamics while preserving item information, eventually entering a marginally stable maintenance regime. During retrieval, rank subspaces sequentially align with the fixed output-weight space, enabling selective readout in ordinal order. The learned ordinal organization generalizes to held-out sequence combinations and across the tested sequence lengths. Moreover, in a synthetic stroke-sequence extension, perturbing an individual item primarily impairs recall at the corresponding rank, demonstrating limited cross-rank interference. Together, these findings reveal how recurrent dynamics configure a reusable ordinal schema to route, preserve, and retrieve information across the temporally extended stages of SWM.

2 Learning Ordinal Event Schemas for SWM

2.1 The Neural Circuit Model

To investigate how an ordinal schema emerges and is dynamically deployed in SWM, we designed a neural circuit model comprising three parts (Fig. 1C): an input layer encoding the temporal sequence, an RNN designed to capture the ordinal schema through its learned dynamics, and an output layer for sequence retrieval. This architecture allows us to test the hypothesis that SWM training organizes recurrent activity into ordinal rank subspaces that align with fixed input- and output-weight spaces according to encoding and recall demands (Fig. 1B).

Let $\mathbf{I}_{\text{ext}}(t)$ denote input-layer activity during encoding. This activity encodes a sequence of L stimuli $\{S_l\}_{l=1}^{L}$ (colored solid circles in Fig. 1A), where each stimulus is drawn from a set of $M = 6$ possible identities and presented for a duration t_{S_l}. We adopt a population-coded input representation following previous SWM models (Whittington et al., 2025; Wang et al., 2025). The detailed formulation is provided in Appendix A.2.2, with example input signals illustrated in Fig. S1A–C. Given the limited capacity of working memory (typically around 4 items), we set $L = 3$ unless otherwise specified, consistent with SWM experimental designs (Xie et al., 2022; Li et al., 2025; Driscoll et al., 2024).

The input layer projects to the RNN via a feedforward connection matrix $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_{\text{rec}} \times N_{\text{in}}}$, with N_{in} and N_{rec} the number of neurons in the input and recurrent layer, respectively. Neurons in the recurrent layer are connected through a matrix $\mathbf{W}_{\text{rec}} \in \mathbb{R}^{N_{\text{rec}} \times N_{\text{rec}}}$. Let $\mathbf{x}(t) \in \mathbb{R}^{N_{\text{rec}}}$ denote the synaptic current of the RNN neurons at time t and $\mathbf{r}(t) = \phi(\mathbf{x}(t)) = \tanh(\mathbf{x}(t))$ the neuronal firing-rate vector. The RNN dynamics are written as

$$\tau \frac{d\mathbf{x}(t)}{dt} = -\mathbf{x}(t) + \mathbf{W}_{\text{rec}}\phi(\mathbf{x}(t)) + \mathbf{W}_{\text{in}}\mathbf{I}_{\text{ext}}(t) + \mathbf{b}, \tag{1}$$

where τ denotes the time constant and $\mathbf{b}$ is a bias term. The network output is given by $\mathbf{y}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t)$, where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_{\text{out}} \times N_{\text{rec}}}$ projects RNN activity to $N_{\text{out}} = M + 1$ output neurons (M one-hot stimulus output neurons and one fixation neuron). Retrieval begins when the fixation signal ceases, after which the network sequentially outputs the memorized items in their original presentation order (Appendix A.2.3; Fig. S1D–F).

Unlike simplified L-item immediate serial-recall paradigms, our task incorporates multi-timestep item presentation and recall, randomized inter-stimulus intervals (ISIs), and an explicit maintenance

period. Critically, randomizing the ISIs dissociates ordinal rank from absolute elapsed time: because items must be retrieved in the correct serial order despite variable presentation timing, the network cannot reliably infer ordinal position from a fixed temporal cue and must instead learn a representation of rank that generalizes across temporal variations. This design therefore promotes the separation of ordinal structure from stimulus content and elapsed time. More broadly, the temporally extended task separates encoding, maintenance, and retrieval into distinct processing stages. They also shape the network's attractor dynamics, enabling us to examine how recurrent population geometry and dynamics evolve across these stages and to relate the resulting mechanisms more directly to experimental SWM paradigms (Orhan & Ma, 2019; Bernardi et al., 2020).

For each training trial, we uniformly sampled one of the 120 possible ordered sequences of three distinct stimuli and independently randomized the inter-stimulus intervals and maintenance duration. Input signals and target outputs were constructed as described in Appendices A.2.1–A.2.3. The network was trained using backpropagation through time to minimize a weighted mean-squared error loss that placed greater emphasis on item retrieval (Driscoll et al., 2024). Training jointly optimized the parameters $\{\mathbf{W}_{\text{in}}, \mathbf{W}_{\text{rec}}, \mathbf{W}_{\text{out}}, \mathbf{b}\}$. Ablations of the retrieval weighting scheme are shown in Fig. S20A, with full training details provided in Appendix A.3. After training, the models achieved 100% exact-sequence accuracy across 12,000 uniformly sampled evaluation trials (Appendix A.2.4); an example trial is shown in Fig. S1. For geometric analyses and visualization, we used all $N_{\text{tr}} = 120$ possible length-three sequences, matching the length-three sequence set in Xie et al. (2022).

2.2 Disentangled Neural Representations of Rank and Content

Orthogonal Rank Subspaces During Maintenance We first tested whether the trained RNN develops a representational geometry that disentangles item content from ordinal rank, as observed experimentally. Following Xie et al. (2022), we applied the same linear regression framework to relate the maintenance-period average firing rate $\overline{r}_i$ of neuron i ($i = 1, \ldots, N_{\text{rec}}$) to ordinal rank ($L = 3$) and item identity ($M = 6$): $\overline{r}_i = \sum_{l=1}^{L} \sum_{m=1}^{M} \beta_i(l, m)\psi_{l,m} + \epsilon_i$. Here, $\psi_{l,m}$ is a binary indicator that equals 1 if stimulus m appears at rank l, and 0 otherwise (Appendices B.1 and B.2.1). Fitting this model across N_{tr} trials yielded regression coefficients $\beta_i(l, m)$, which specify the response weight of neuron i to each rank-content pair (l, m). For each rank l, we collected the coefficient vectors for all M identities across the N_{rec} recurrent neurons and applied PCA to the resulting $N_{\text{rec}} \times M$ coefficient matrix. The first two principal components (PCs) for each rank explain over 97.2% of the variance (Fig. S2). Projecting the regressoin coefficient vectors onto these two PCs yields M points per rank, arranged along a ring-like manifold whose radius shrinks with ordinal position l (Fig. 2A, Appendix B.3.1); principal-angle analysis across ranks further shows that these model-estimated rank subspaces are nearly orthogonal (Fig. 2F), consistent with key features of the factorized geometry observed experimentally (Xie et al., 2022).

Because this regression analysis explicitly conditions on rank and item identity, we next asked whether the same factorized geometry could be recovered directly from population activity without imposing this task-variable structure. We therefore applied PCA to maintenance-period activity across all N_{tr} trials (Appendix B.3.2). The top six PCs explain more than 96.8% of the total variance (Fig. 2G). Because PCA is agnostic to rank and item labels, this analysis provides a data-driven test of whether rank-specific geometry is intrinsic to the organization of recurrent activity. We examined all 15 two-dimensional subspaces defined by pairs of the top six PCs and assessed whether each subspace organized trials according to item identity at a particular rank, with the expected ring-like topology. For instance, if a PC pair (e.g., PC1–PC2) spans a subspace that fully represents all items at a given rank, then coloring the projected points by item identity at that rank should yield M distinct, orderly clusters arranged by item feature in a ring-like topology; otherwise, the points appear diffusely scattered (Appendix C.1; Figs. S9–S12).

This analysis reveals a striking correspondence between PC pairs and ordinal ranks: item content at ranks 1, 2, and 3 is selectively organized within the subspaces spanned by (PC1, PC2), (PC3, PC4), and (PC5, PC6), respectively (Fig. 2B). Rank separation persists under the tested alternative input and output codes, although the within-rank item geometry and degree of between-rank orthogonality vary across coding schemes (Appendix C.3; Figs. S13–S15). Thus, the network spontaneously factorizes ordinal rank and item content into distinct components of population geometry rather than exhibiting this organization only under a task-informed regression analysis. The rank-1 subspace accounts for the largest fraction of maintenance-period variance and exhibits the largest ring radius. Unlike the

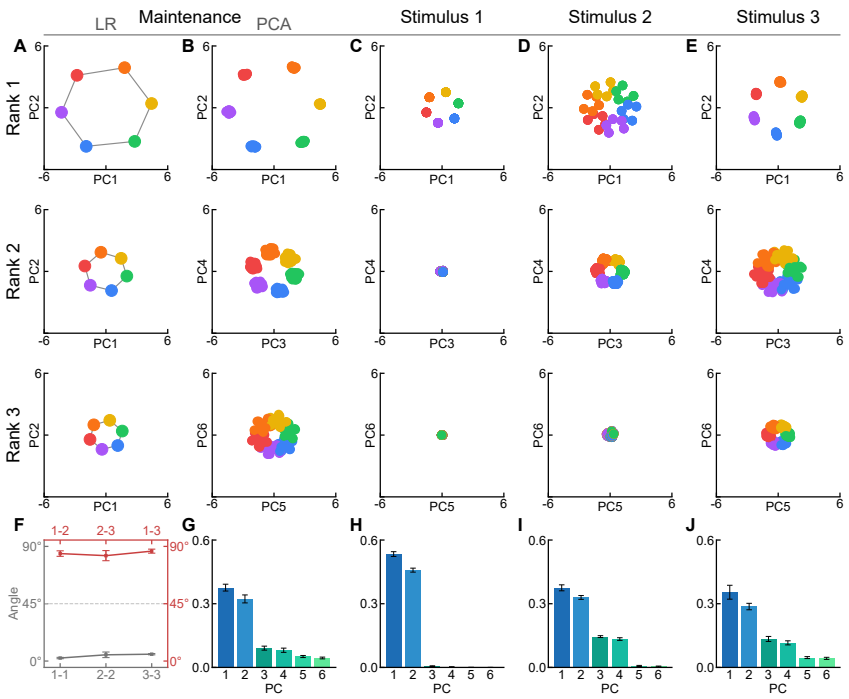

Figure 2: **Learned disentangled rank and content neural representations across SWM stages** (A) Linear regression (LR) analysis of averaged RNN hidden activity during maintenance. Regression coefficients are projected onto the first two PCs of each rank's subspace and color-coded by stimulus identity. (B–E) Direct PCA visualization of RNN hidden activity. PCA is performed on the total activity over the maintenance period (B); the subspace for each rank is spanned as follows: rank 1, PC1–2; rank 2, PC3–4; rank 3, PC5–6. Activities from the first, second, and third stimulus presentation periods are then projected onto the corresponding time-local PCs (C–E). (F) The principal angles between the LR-derived rank subspaces during maintenance (red). As a control, we randomly split the trials into two halves, estimated each rank subspace separately from each half, and computed the principal angle between the two estimates (gray). (G–J) Histogram of the variance explained by each PC across different stages: maintenance (G) and the first (H), second (I), and third (J) stimulus presentation periods. Results are averaged over 5 seeds.

models of Whittington et al. (2025) and Wang et al. (2025), whose ring-like manifolds are similarly sized across ranks, our model shows a rank-dependent decrease in manifold size that matches the experimental findings (Xie et al., 2022).

Disentangled Neural Representations During Encoding and Retrieval We applied the same PCA-based analysis separately to the population activity time-locked to the onset of each stimulus across N_{tr} trials during encoding (and to the corresponding retrieval periods; see below). This analysis reveals how sequential items are encoded and recalled, respectively. During the presentation of stimulus S_1, the leading PC pair encodes it as a ring-like manifold (Fig. 2C). For S_2 and S_3, the network recruits new PC pairs to encode each incoming item, while all previously loaded items remain separable within their respective rank subspaces (Fig. 2D–E). These results indicate that the network concurrently represents multiple items in orthogonal rank subspaces during encoding, a finding further supported by linear-regression-based projection analyses (Fig. S16). The explained-variance spectra reveal the same incremental recruitment pattern: PC1–2 dominate after S_1 onset, with PC3–4 and PC5–6 newly recruited after S_2 and S_3 onset, respectively (Fig. 2H–J).

We analyzed the subspace spanned by the leading PC pair separately for the retrieval of each stimulus. During retrieval, this subspace dynamically tracks the currently recalled item. Specifically, when S_1 is recalled, it represents S_1 (Fig. S17A). As S_1 fades, it shifts to representing S_2 during the recall of S_2 (Fig. S17B), and it similarly tracks the third item during the recall of S_3 (Fig. S17C). Thus, the dominant neural subspace emphasizes the item in focus while earlier items are allowed to fade, an efficient biological strategy that frees neural resources once they are no longer needed.

3 DYNAMIC DEPLOYMENT OF ORDINAL SCHEMAS FOR SWM

Unified Rank-Specific Subspaces Across Stages The rank subspaces described above were derived by applying PCA separately to different SWM stages — specifically, to each stimulus-presentation window during encoding, to maintenance activity, and to each recall window during retrieval (Figs. 2 and S17). Because the resulting PC bases are stage-specific rather than aligned across stages by construction, we used canonical correlation analysis (CCA) to test whether these subspaces nevertheless encode the same rank-dependent information. Comparing the same rank subspace across encoding, maintenance, and retrieval (matched ranks) reveals high cross-stage similarity, whereas comparing different ranks (mismatched ranks) yields low similarity (Fig. 3A; Appendix B.4). On the strength of this result, we denote the common rank-specific representational subspaces by $\mathcal{U}_1, \mathcal{U}_2$, and $\mathcal{U}_3$, even though they are each defined locally within a stage. This cross-stage representational reuse is central to the ordinal schema interpretation: the network does not rely on unrelated stage-specific codes to handle encoding, maintenance, and retrieval, but instead preserves a shared ordinal geometry that can be accessed by different input and output pathways.

The results above establish the ordinal schema as a representational scaffold: rank-specific subspaces preserve the same ordinal content across encoding, maintenance, and retrieval. We next analyzed how this scaffold is used computationally (Fig. 1B), focusing on stage-dependent dynamic subspace deployment, input/output-weight coupling, and local recurrent dynamics that support maintenance.

Stage-Dependent Evolution of Rank Subspaces We next examined how rank-specific subspace ($\mathcal{U}_1$, $\mathcal{U}_2$, or $\mathcal{U}_3$) evolves during sequential encoding by quantifying the angular shift of the same subspace across successive stimulus-presentation windows (Fig. 3B; Appendix B.5.1). After the first stimulus S_1 is encoded into $\mathcal{U}_1$ during t_{S_1}, presentation of S_2 recruits $\mathcal{U}_2$ while the previously established $\mathcal{U}_1$ reorients by approximately $49.7°$. During t_{S_3}, S_3 is encoded into $\mathcal{U}_3$, while $\mathcal{U}_1$ and $\mathcal{U}_2$ undergo further reorientations of approximately $36.8°$ and $74.7°$, respectively. Although these angular changes indicate substantial reorientation of the rank-specific geometry, the underlying transformations cannot be described as pure orthonormal rotations. Linear transformations fitted between encoding- and maintenance-stage representations exhibit singular values substantially different from 1 (range: 0.65–3.60), indicating that reorientation is accompanied by scaling and shear accompany the reorientation (Appendix B.5.2; Table S2). Thus, as new stimuli are sequentially encoded into their corresponding rank-specific subspaces, previously established representations continue to undergo structured geometric transformations while preserving their item-specific information.

Alignment via Input & Output Weights To explain the systematic reorientation of rank-specific subspaces during encoding, we examined how the fixed input weight matrix $\mathbf{W}_{\text{in}}$ couples to the currently relevant rank subspace. Here, *alignment* refers to the geometric coupling between the column space of $\mathbf{W}_{\text{in}}$ and a rank-specific subspace, which we quantify using principal angles: smaller angles indicate stronger effective coupling (Fig. 3C; Appendix B.5.3). This stage-specific pattern holds under alternative subspace-distance criteria (Table S3). We defined input-weight contribution to $\mathcal{U}_l$ as $C_l = \sum_{\mathbf{p}_k \in \mathcal{U}_l} \lambda_k \|\mathbf{W}_{\text{in}}^\top \mathbf{p}_k\|^2$, followed by normalization; $\mathbf{p}_k \in \mathcal{U}_l$ is a principal-component vector and λ_k its explained-variance weight (Appendix B.6).

The results are consistent with alignment-mediated routing: evolving rank subspaces sequentially align with the fixed input-weight space. During t_{S_1}, $\mathcal{U}_1$ is closest to this space (angle $\approx 22.8°$) and receives the highest input-weight contribution (Fig. 3F), consistent with preferential coupling of the current input to $\mathcal{U}_1$. At t_{S_2}, $\mathcal{U}_2$ is most closely aligned with $\mathbf{W}_{\text{in}}$ (angle $\approx 35.0°$, Fig. 3G), consistent with loading the new input representation into $\mathcal{U}_2$. Meanwhile, the angle between $\mathbf{W}_{\text{in}}$ and $\mathcal{U}_1$ increases to about $58.6°$ as $\mathcal{U}_1$ reorients. At t_{S_3}, $\mathcal{U}_3$ is most closely aligned (angle $\approx 37.4°$, Fig. 3H), while earlier subspaces continue to evolve away from the input-weight space.

To test the functional role of input–subspace alignment, we transiently rotated the column space of $\mathbf{W}_{\text{in}}$ during one stimulus presentation, leaving all other parameters unchanged. With a $90°$ rotation, recall accuracy at the targeted rank dropped from 100% to $6.78 \pm 2.16\%$ (mean $\pm$ SD across five independently trained networks), accompanied by reorientation of the corresponding subspace at retrieval. Recall at non-targeted ranks was also impaired, but to a lesser extent (Appendix C.6). Complementary perturbations further support this interpretation: modifying the input-weight disrupts cross-stage rank-subspace organization and reduces subspace orthogonality, whereas stimulus silencing shifts subsequent alignment toward earlier rank subspaces (Appendix C.5; Figs. S4 and S5).

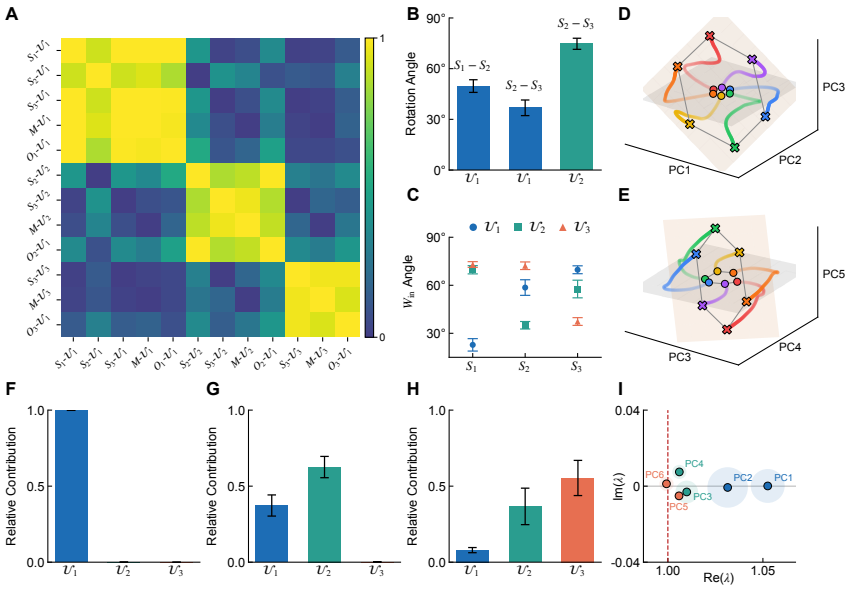

Figure 3: **Analysis of RNN representations and dynamics across SWM stages.** (A) Cross-stage CCA similarity between rank subspaces across different stages. Labels indicate stage/subspace pairs: S_1, S_2, S_3 are stimulus-presentation stages, M is maintenance, O_1, O_2, O_3 are output-retrieval stages, and $\{\text{stage}\} - \mathcal{U}_l$ denotes the rank-l subspace estimated during the corresponding stage. (B) Angular change of rank subspaces across successive encoding periods. For example, $S_1 - S_2$ at $\mathcal{U}_1$ indicates how much the rank-1 subspace rotates between the first and second stimulus presentation periods. (C) Angles between each rank subspace and the input-weight subspace (spanned by the columns of $\mathbf{W}_{\text{in}}$) across different stimulus presentations. (D) Trial-averaged neural trajectories in PC space for the $t_{S_1} \to t_{S_2}$ transition. Gray and orange planes denote the rank-1 subspace during S_1 and its evolution till S_2; circles mark S_1 item clusters, crosses mark their states after S_2, and colored paths trace temporal evolution. (E) The same analysis for rank 2 during the $t_{S_2} \to t_{S_3}$ transition. (F–H) Relative contribution of $\mathbf{W}_{\text{in}}$ to each rank subspace, shown for the first (F), second (G), and third (H) stimulus stages. (I) Linearized stability analysis of maintenance activity. Eigenspectrum of the Jacobian around the final maintenance states, projected onto the subspace spanned by the top six maintenance PCs. Results are averaged over 5 seeds.

We applied the same alignment and contribution analyses to the output weight matrix $\mathbf{W}_{\text{out}}$ during retrieval. The results show a similar alignment pattern: the currently active rank-specific subspace sequentially becomes aligned with the fixed row space of $\mathbf{W}_{\text{out}}$, consistent with serial readout in presentation order (Appendix C.4 and Fig. S17D).

To directly visualize the stage-dependent evolution and alignment of rank subspaces, we examined network dynamics across consecutive stimulus presentations as a representative example. We projected trial-averaged activity onto the first few PCs (Fig. 3D–E; Appendix B.8). Figure 3D shows the resulting trial-averaged neural trajectories (N_{tr} trials) during the $S_1 \to S_2$ transition. The gray plane represents the rank-1 subspace identified for S_1, and the orange plane shows the same subspace as it appears during S_2. Circles denote the six S_1 stimuli, while crosses indicate their corresponding states after S_2 presentation; these points are largely confined to their respective planes, illustrating how S_2 input drives the stage-dependent evolution of the rank-1 subspace while preserving the information encoded during S_1. This stage-dependent evolution captures the experimentally observed sensory-to-memory transformation of S_1 information: when we apply the time-resolved linear-regression framework of Chen et al. (2024) with a shared sensory subspace and rank-specific memory subspaces, our model shows the same transient stimulus information in the sensory subspace, followed by persistent information in the corresponding memory subspace (Appendices B.2.2 and C.8; Fig. S22). Fig. 3E shows rank-2 subspace evolution during $S_2 \to S_3$. These patterns indicate that a stimulus-loaded rank subspace, as it drifts away from the $\mathbf{W}_{\text{in}}$-aligned sensory axis, gradually shifts from encoding transient sensory input to maintaining a sustained memory representation.

Network Dynamics Analysis during Maintenance To assess the stability of neural activity during maintenance, we linearized the network dynamics around the activity states and examined the eigenspectrum of the Jacobian restricted to the subspace spanned by the top six PCs (Fig. 3I; Appendix B.7). The eigenvalue magnitudes cluster near the boundary of marginal stability and decrease systematically with retrieval order, so that earlier-retrieved items exhibit larger magnitudes. The spectrum remained near unity across evaluation timesteps, retained dimensionalities, and delay durations within task range. This pattern is consistent with marginally stable maintenance dynamics that balances the stability required for storage with the sensitivity needed for sequential readout.

Having established that the model recapitulates experimentally observed population-level geometry, we next asked whether this correspondence extends to single neurons. Model neurons showed tuning consistent with experiments (Xie et al., 2022; Li et al., 2025): some showed gain modulation across ranks, whereas others showed pronounced rank-dependent shifts in preferred direction. We further quantified neuron-to-subspace strength (NSS), which measures each neuron's selective or shared contributions across rank subspaces, and found a distribution similar to that observed experimentally (Appendix C.9; Figs. S7 and S8). Reproducing these single-neuron signatures, which previous SWM models have not explicitly accounted for, extends biological correspondence beyond the population-level representations captured by those models.

4 Generalization and Interference Mitigation via Schemas

Generalization Within and Across Sequence Lengths A disentangled ordinal schema should support compositional generalization by recombining familiar item and rank representations into novel sequences. We tested this prediction by training each model on 90 of the 120 possible length-3 sequences and holding out 30 sequences comprising all permutations of five unseen three-item combinations. Every item and item–rank pairing appeared during training, but no held-out sequence was observed, requiring generalization beyond sequence memorization. All 40 models correctly recalled every held-out sequence at the fixed evaluation timing (Appendix A.2.5), supporting a compositional ordinal representation that generalizes to unseen sequences of the same length.

We next tested whether the learned ordinal schema transfers across sequence lengths. Models trained exclusively on length-3 sequences achieved near-perfect zero-shot performance at lengths 1 and 2 (Fig. S19A). This transfer preserved the learned geometry: the corresponding encoding-stage rank subspaces for ranks 1 and 2 closely matched their length-3 counterparts (Fig. S19B), and the stage-dependent alignment pattern was retained (Fig. S18). Transfer to longer sequences required additional training but benefited from the learned structure: length-3 pretraining accelerated length-4 learning (Fig. S20B), with the first three rank subspaces retained and a new, approximately orthogonal subspace emerging for rank 4. Thus, the learned ordinal schema supports zero-shot generalization to shorter sequences and structure-preserving transfer that facilitates learning of longer sequences.

Mitigating Sensory-Memory Interference in Handwritten Sequence Reproduction Dynamically deploying an ordinal schema routes incoming information into distinct, near-orthogonal rank subspaces, potentially limiting cross-rank interference. To test this robustness with richer item representations, we trained the network on a handwritten-character reproduction task in which each item was a stroke defined by direction and length (Appendix D). We analyzed three-stroke sequences, including the uppercase letter 'A' in Fig. 4A. Using a dataset with eight stroke directions and three length levels (9,500 training sequences), the model achieved over 99% accuracy. Geometric analyses revealed rank-specific ring manifolds for stroke direction (Fig. 4B), showing that the ordinal organization extends to multidimensional stroke representations. If information is separated into near-orthogonal rank subspaces, perturbing one item should mainly impair recall at that rank, whereas overlapping representations should allow the effect to spread across ranks. To test this prediction, we used a cross-entropy-trained model as a comparison: it performs the same SWM task but develops a different internal representational organization, allowing us to compare interference robustness across distinct representational regimes. We injected Gaussian noise into one stroke during encoding and measured rank-wise recall accuracy in both models (Appendix B.10.2). Perturbations mainly impaired recall at the corresponding rank in our model, whereas accuracy declined more broadly in the comparison model, particularly at subsequent ranks (Figs. 4C and S21). Thus, under the tested perturbations, the learned rank-specific geometry localizes the effects of corrupted inputs and thereby limits cross-rank interference.

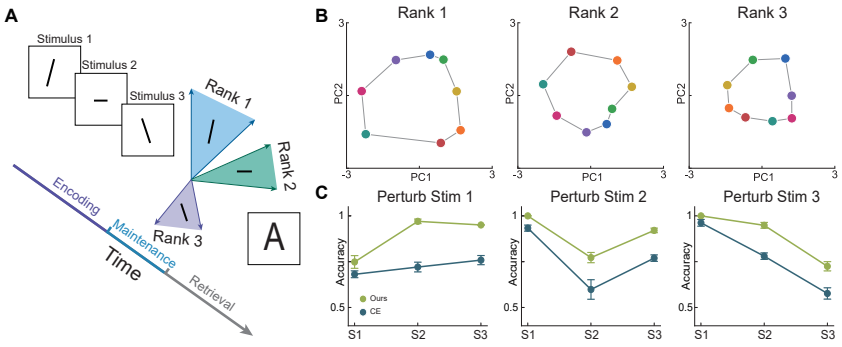

Figure 4: **Applying ordinal event schema in a handwritten sequence task**. (A) Handwritten character task. Three sequential stroke stimuli are encoded, maintained, and then reproduced in order. In the illustrated example, the three strokes form the uppercase letter "A". (B) PCA visualization of RNN hidden activity during maintenance. (C) Performance comparison between the RNN with an ordinal event schema and a comparison model with a different target representation and training objective. Gaussian noise is added to stimulus 1, 2, or 3 during encoding, and recall accuracy is evaluated separately for each output rank. The comparison model is trained with cross-entropy loss and a different output representation. Results are averaged over 5 seeds.

5 Discussion

Our results suggest that rank-specific population geometry is not merely a representational signature of sequence working memory, but a reusable computational scaffold for organizing information over time. The trained RNN develops a factorized ordinal schema of approximately orthogonal rank subspaces, recapitulating key features observed in macaque prefrontal cortex (Xie et al., 2022). Recurrent dynamics configure this geometry relative to fixed computational interfaces: rank subspaces align with input weights during encoding, evolve through structured, marginally stable dynamics during maintenance, and sequentially align with output weights during retrieval. This organization supports compositional generalization to unseen sequences, zero-shot transfer to shorter sequences, and facilitated learning of longer sequences, while limiting cross-rank interference in a richer stroke-based task. These findings suggest that population geometry can function not only as a representational code, but as a dynamic computational resource for structured sequence processing.

LSTM, GRU, and our RNN achieved perfect sequence recall, yet the gated architectures lacked the clear matched-rank organization across stages (Appendix C.7; Fig. S6). Thus, distinct representations can support SWM. The significance of our mechanism lies not in its necessity, but in its correspondence with macaque population geometry and its account of how distributed ordinal representations can be dynamically deployed for temporally extended sequence processing (Xie et al., 2022).

Limitations and Future Directions Our model learns a reusable ordinal schema for structured operations on temporal sequences, but several limitations remain. First, we consider limited rule-dependent manipulation. In biological SWM, task cues can reprioritize or reorder stored items, supporting bidirectional retrieval and flexible mental manipulation (Chen et al., 2024; Tian et al., 2024). Future work should determine whether recurrent dynamics can flexibly recruit, suppress, or reorder reusable rank subspaces under explicit task rules, and how such rule-dependent control interacts with their stage-dependent geometric organization. Second, our experiments focus on relatively short sequences and primarily on ordinal structure. Although our results show that previously learned rank subspaces can facilitate extension to an additional ordinal position, scaling this strategy by simply adding subspaces may become inefficient for substantially longer or more complex sequences. Biological and artificial systems may instead compress sequential information into chunks and organize these chunks hierarchically (Fan et al., 2025). More broadly, temporal structure spans multiple levels of abstraction, including transition and timing statistics, ordinal relations, chunks, algebraic patterns, and nested structures (Dehaene et al., 2015). Our model addresses only the ordinal level. Future work should investigate how local ordinal schemas are composed into hierarchical representations and how these representations interact with attention-based or structured-memory mechanisms to support longer-range and nested sequence computation.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript editing, interpretation of computational results, and minor revisions to analysis and figure-generation code. All AI-assisted changes were reviewed and validated by the authors against the underlying methods and results. The authors take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The model, task construction, training protocol, and analysis procedures are described in the main text and Appendices A– B. Control experiments and the handwritten sequence task are documented in the remaining appendix sections.

REFERENCES

Mikio C Aoi, Valerio Mante, and Jonathan W Pillow. Prefrontal cortex exhibits multidimensional dynamic encoding during decision-making. *Nature Neuroscience*, 23(11):1410–1420, 2020.

Alan Baddeley. Working memory: Theories, models, and controversies. *Annual Review of Psychology*, 63(1):1–29, 2012.

Silvia Bernardi, Marcus K Benna, Mattia Rigotti, Jérôme Munuera, Stefano Fusi, and C. Daniel Salzman. The geometry of abstraction in the hippocampus and prefrontal cortex. *Cell*, 183(4): 954–967, 2020.

Jingwen Chen, Cong Zhang, Peiyao Hu, Bin Min, and Liping Wang. Flexible control of sequence working memory in the macaque frontal cortex. *Neuron*, 112(20):3502–3514, 2024.

Mark M. Churchland, John P Cunningham, Matthew T. Kaufman, Justin D. Foster, Paul Nuyujukian, Stephen I. Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487 (7405):51–56, 2012.

Stanislas Dehaene, Florent Meyniel, Catherine Wacongne, Liping Wang, and Christophe Pallier. The neural representation of sequences: from transition probabilities to algebraic patterns and linguistic trees. *Neuron*, 88(1):2–19, 2015.

Stanislas Dehaene, Fosca Al Roumi, Yair Lakretz, Samuel Planton, and Mathias Sablé-Meyer. Symbols and mental programs: a hypothesis about human singularity. *Trends in Cognitive Sciences*, 26(9):751–766, 2022.

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.

Gamaleldin F. Elsayed, Antonio H. Lara, Matthew T. Kaufman, Mark M. Churchland, and John P. Cunningham. Reorganization between preparatory and movement population responses in motor cortex. *Nature Communications*, 7(1):13239, 2016.

Ying Fan, Muzhi Wang, Fang Fang, Nai Ding, and Huan Luo. Two-dimensional neural geometry underpins hierarchical organization of sequence in human working memory. *Nature Human Behaviour*, 9(2):360–375, 2025.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Siwei Li, Jingwen Chen, Cong Zhang, Shiming Tang, Yang Xie, and Liping Wang. Flexible use of limited resources for sequence working memory in macaque prefrontal cortex. *Nature Communications*, 16(1):10386, 2025.

Alexandra Libby and Timothy J. Buschman. Rotational dynamics reduce interference between sensory and memory representations. *Nature Neuroscience*, 24(5):715–726, 2021.

Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. *Advances in Neural Information Processing Systems*, 32, 2019.

Earl K Miller, Mikael Lundqvist, and André M Bastos. Working memory 2.0. *Neuron*, 100(2): 463–475, 2018.

A. Emin Orhan and Wei Ji Ma. A diverse range of factors affect the nature of neural representations underlying short-term memory. *Nature Neuroscience*, 22(2):275–283, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Alexander Rivkind and Omri Barak. Local dynamics in trained recurrent neural networks. *Physical Review Letters*, 118(25):258101, 2017.

Zhenghe Tian, Jingwen Chen, Cong Zhang, Bin Min, Bo Xu, and Liping Wang. Mental programming of spatial sequences in working memory in the macaque frontal cortex. *Science*, 385(6716): eadp6091, 2024.

Mingye Wang, Stefano Fusi, and Kim Stachenfeld. Brain-like slot representation for sequence working memory in recurrent neural networks. In *Second Workshop on Representational Alignment at ICLR 2025*, 2025.

James C. R. Whittington, Timothy H. Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy E. J. Behrens. The Tolman–Eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

James C. R. Whittington, William Dorrell, Timothy E. J. Behrens, Surya Ganguli, and Mohamady El-Gaby. A tale of two algorithms: Structured slots explain prefrontal sequence memory and are unified with hippocampal cognitive maps. *Neuron*, 113(2):321–333, 2025.

Yang Xie, Peiyao Hu, Junru Li, Jingwen Chen, Weibin Song, Xiao-Jing Wang, Tianming Yang, Stanislas Dehaene, Shiming Tang, Bin Min, et al. Geometry of sequence working memory in macaque prefrontal cortex. *Science*, 375(6581):632–639, 2022.

Binghao Yang and Shan Yu. Orthogonal-rotational dynamics supports efficient encoding and updating for streaming information in working memory. *Journal of Neuroscience*, 46(3):e2220242025, 2026. doi: 10.1523/JNEUROSCI.2220-24.2025.

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.

Xiaolong Zou, Zhikun Chu, Qinghai Guo, Jie Cheng, Bo Ho, Si Wu, Yuanyuan Mi, et al. Learning and processing the ordinal information of temporal sequences in recurrent neural circuits. *Advances in Neural Information Processing Systems*, 36:33999–34020, 2023.

A Model Parameters and Training Protocol

A.1 Model Parameters

The recurrent neural network (RNN) used for training comprises an input layer with $N_{\text{in}} = 25$ neurons, including 24 channels encoding spatial stimuli and one channel encoding the fixation signal. The recurrent (hidden) layer contains $N_{\text{rec}} = 400$ neurons, and the output layer contains $N_{\text{out}} = 7$ neurons, corresponding to six spatial stimulus category outputs and one fixation output. The RNN time constant is set to $\tau = 1$. Continuous-time dynamics are discretized using the Euler method with a time step of $dt = 0.05$.

The recurrent weight matrix $\mathbf{W}_{\text{rec}}$ is initialized from a Gaussian distribution with zero mean and standard deviation $g/\sqrt{N_{\text{rec}}}$, where $g = 1$ denotes the network gain. The input weight matrix $\mathbf{W}_{\text{in}}$ is initialized from a zero-mean Gaussian distribution with standard deviation $1/\sqrt{N_{\text{in}}}$, and the output weight matrix $\mathbf{W}_{\text{out}}$ is initialized analogously with standard deviation $1/\sqrt{N_{\text{out}}}$. The recurrent bias term is initialized to zero. The batch size is set to $N_{\text{batch}} = 64$, and training is performed for a total of 360 epochs. All models are implemented and trained using PyTorch (Paszke et al., 2019). All experiments and analyses reported in this paper are lightweight and can be reproduced on a modern laptop CPU.

A.2 Dataset Definition

Each trial consists of three types of time series: external spatial stimulus inputs, a phase-indicating fixation input, and target outputs for supervised learning.

A.2.1 Trial Structure

Within each trial, the first stimulus is presented after an initial waiting period of $t_{\text{ini}} = 8$ timesteps. All stimuli have a fixed duration of $t_{\text{dur}} = 8$, and the inter-stimulus interval t_{int} between consecutive stimuli is independently sampled for each trial from a uniform distribution $U(0, 8)$. After all stimuli have been presented, the trial enters a maintenance phase of duration t_{mnt}, followed by a retrieval phase. t_{mnt} is independently sampled for each trial from a uniform distribution $U(20, 30)$. During retrieval, the model is required to sequentially output the spatial stimulus labels encountered during the encoding phase, with each output lasting $t_{\text{retr}} = 12$. We denote the three stimulus-presentation time windows by t_{S_1}, t_{S_2}, and t_{S_3}, and the three retrieval time windows by t_{R_1}, t_{R_2}, and t_{R_3}.

A.2.2 Input Signals

The 24 spatial stimulus channels in the input layer are treated as a population of neurons uniformly distributed on a circular space. The preferred direction of the n-th channel is defined as $\theta_n = (2n - 24)\pi/24$. When a stimulus $\theta^m_{s,l} = (2m - 6)\pi/6$ is presented, it evokes a Gaussian bump across the input population. The full input vector $\mathbf{I}_{\text{ext}}(t) \in \mathbb{R}^{25}$ includes these 24 channels and the fixation channel. For $n = 1, \ldots, 24$, its spatial stimulus components are given by

$$[\mathbf{I}_{\text{ext}}(t)]_n = A_s \exp\left(-\frac{(\theta_n - \theta^m_{s,l})^2}{2\sigma_s^2}\right), \tag{2}$$

where $A_s = 4$ and $\sigma_s = \sqrt{2}$. This Gaussian bump input is defined under circular boundary conditions, allowing representational overlap between neighboring stimuli in the input layer.

The fixation input channel is set to 1 from the beginning of the trial until the end of the maintenance phase and switches to 0 during the retrieval phase, thereby explicitly marking task phase boundaries. During training, independent Gaussian noise with standard deviation 0.01, $\mathcal{N}(0, 0.01^2)$, is added to all input channels at every timestep to enhance the robustness of network dynamics to temporal perturbations.

A.2.3 TARGET OUTPUTS

The seven output channels are divided into two groups. Channels 1–6 correspond to spatial stimulus labels and are supervised using one-hot encoding only during the retrieval phase; at all other times, their targets are set to zero. Channel 7 corresponds to the fixation output and is aligned with the fixation input throughout the entire trial.

The resulting population-code inputs, fixation signal, supervised targets, and example model outputs are summarized in Fig. S1.

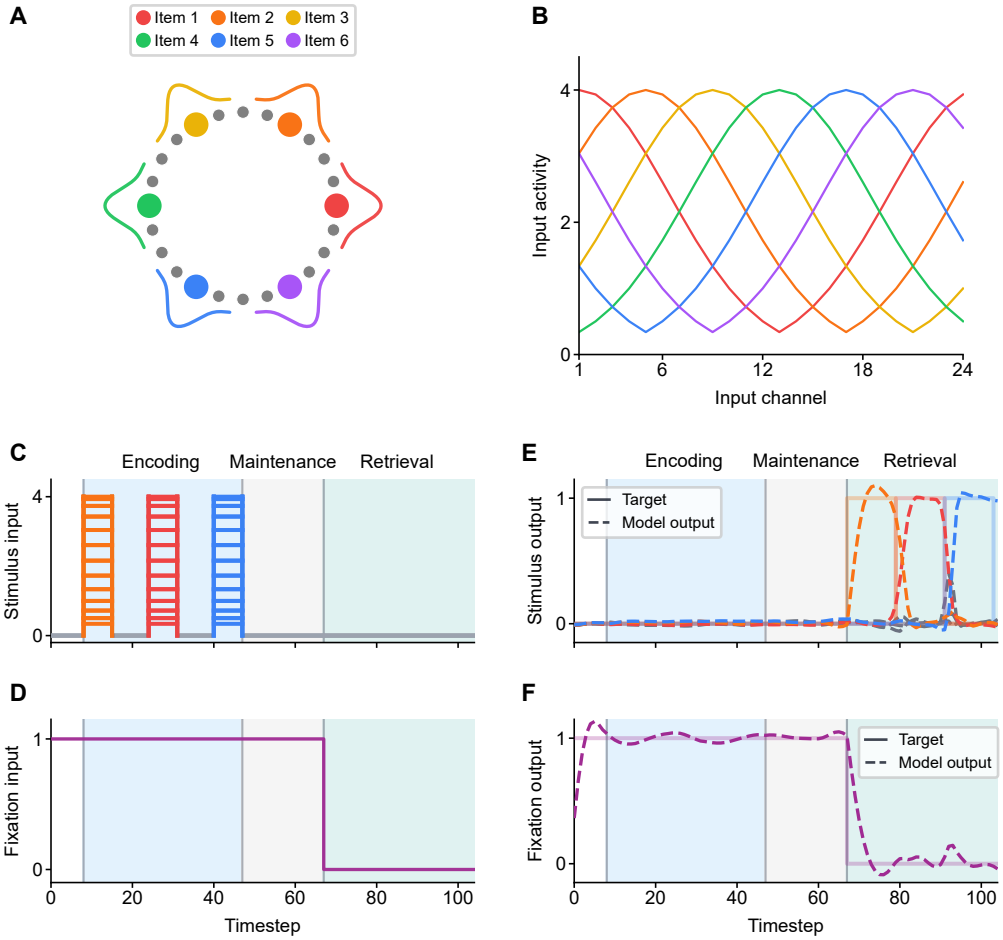

Figure S1: **Input and output signals used in the SWM model.** (A) Stimulus identities are encoded as Gaussian population-code bumps over 24 input channels, with six discrete identities indicated by color. (B) The same six input patterns shown as one-dimensional channel profiles. (C–D) Example stimulus-input and fixation-input traces for one trial. (E–F) Corresponding item-output and fixation-output traces. Solid traces denote supervised targets, and dashed traces denote model outputs. For clarity, the displayed input signals omit the Gaussian noise added during training.

A.2.4 TRAINING AND TEST SETS

In all training trials, the input sequence length is fixed at three. Three distinct spatial stimulus labels are selected without replacement from a set of six possible labels and arranged in order, yielding $N_{\text{tr}} = C_6^3 \times A_3^3 = 20 \times 6 = 120$ possible sequences. In the main experiments, 3000 training trials per model were generated, with item sequences sampled uniformly from all 120 possibilities.

For the main models, the test inter-stimulus interval t_{int} is fixed to 8 and the maintenance duration t_{mnt} to 20. The visualization set contains all 120 possible ordered three-item sequences, while accuracy is

evaluated on 12,000 uniformly sampled trials. In addition, the trained models are evaluated on tasks with different sequence lengths; the results are shown in Fig. S19. Although the models are trained exclusively on sequences of length three, they achieve high accuracy on length-2 tasks. Moreover, the representational structure during the maintenance phase remains consistent with the analyses reported in Sec. 2.2.

A.2.5 HELD-OUT SEQUENCE COMBINATIONS

To test generalization to unseen item combinations, we restricted training and validation to 90 of the 120 possible length-three SWM sequences. Inter-stimulus intervals and maintenance duration were randomized as in the main experiments. The 30 held-out sequences comprised all permutations of five unordered item combinations; every item and item-rank pairing appeared in training, but none of these combinations did. We used eight balanced splits, so each of the 20 unordered combinations was held out twice, and trained five models per split. At the fixed test timing with a 20-step maintenance period and no input noise, all 40 models correctly recalled every sequence in their respective held-out sets (1,200/1,200 exact sequences).

For geometric analyses, we divided the 90 held-in sequences into 45 fitting sequences and 45 reference sequences, with all six items represented at each rank in both subsets. For each rank, we fitted a two-dimensional PCA basis to the six item-centroid firing-rate vectors from the fitting subset. Both the held-in reference sequences and the 30 held-out sequences were projected using this fixed basis and its centering mean. At the end of each rank's stimulus presentation, the mean held-out/held-in capture ratio across ranks and models was 0.996 (crossed-bootstrap 95% CI $[0.948, 1.040]$). We extended this analysis to activity averaged over the late maintenance period, matching the window used for Fig. 2B. For geometric comparison only, we independently fitted rank-wise bases to the held-out item centroids and measured their principal angles to the held-in bases. All three ranks showed capture ratios near one and smaller matched-rank than mismatched-rank angles (Table S1), supporting rank-specific geometric correspondence between familiar and unseen combinations during maintenance.

Table S1: **Maintenance rank-subspace transfer to held-out combinations.** Entries are means with crossed-bootstrap 95% confidence intervals in brackets (10,000 resamples of splits and training seeds independently). Capture is the percentage of squared norm retained when activity centered on the fitted mean is projected onto a held-in basis; ratios are calculated per model before averaging. Angles are means of the two principal angles, with the mismatched control averaged over the two other held-out ranks. Rank bases are fitted separately, so capture percentages need not sum to 100%.

Rank	Held-in (%)	Held-out (%)	Out/in	Matched (°)	Mismatched (°)
1	67.7	67.4	0.995	11.5	56.1
	[65.6, 69.5]	[64.1, 70.0]	[0.953, 1.027]	[9.2, 14.5]	[49.7, 62.3]
2	29.8	29.5	1.009	35.4	68.4
	[24.6, 35.1]	[25.1, 33.5]	[0.921, 1.097]	[30.0, 41.2]	[65.3, 70.9]
3	29.3	29.9	1.032	39.2	65.7
	[23.3, 35.7]	[23.7, 36.1]	[0.922, 1.138]	[32.6, 45.8]	[61.9, 69.7]

A.3 TRAINING PROTOCOL

A.3.1 LOSS FUNCTION

Model training minimizes the sum of a weighted mean-squared error (WMSE) loss and an ℓ_2 regularization term. We use the Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-3}. Let $\mathbf{y}(t)$ denote the model output and $\hat{\mathbf{y}}(t)$ the supervised target at timestep t, with regularization coefficient $\lambda = 0.0008$. The weighted MSE is computed on mask-scaled outputs and targets, giving the total loss

$$L = \frac{\sum_t \sum_{k=1}^{N_{\text{out}}} m_k(t) \left[m_k(t) y_k(t) - m_k(t) \hat{y}_k(t) \right]^2}{\sum_t \sum_{k=1}^{N_{\text{out}}} m_k(t)} + \lambda \sum_{\mathbf{W} \in \mathcal{W}} \|\mathbf{W}\|_2, \tag{3}$$

where $\mathcal{W}$ denotes the set of trainable weight matrices.

A.3.2 Weight Mask

The time-dependent weight mask $\mathbf{m}(t) \in \mathbb{R}^{N_{\text{out}}}$ modulates the error term in both the temporal and output-channel dimensions. Let $\mathcal{T}_{\text{blank}}$ denote the set of the first two timesteps following the onset of each target output. Following Yang et al. (2019), to exclude errors arising from transient responses when output channels switch from a silent to an active state, errors at all output channels are ignored during these timesteps.

For all non-blank timesteps $t \notin \mathcal{T}_{\text{blank}}$, different weights are assigned to different output channels. Specifically, the first $M = 6$ channels correspond to item label outputs and serve as the primary supervision targets, and thus receive larger weights. The seventh channel corresponds to the fixation output and serves only as an auxiliary constraint, and therefore receives a smaller weight. The mask $\mathbf{m}(t) \in \mathbb{R}^{N_{\text{out}}}$ is thus defined as

$$m_k(t) = \begin{cases} 0, & t \in \mathcal{T}_{\text{blank}}, \\ 5, & t \notin \mathcal{T}_{\text{blank}}, \quad k = 1, \ldots, 6, \\ 1, & t \notin \mathcal{T}_{\text{blank}}, \quad k = 7. \end{cases} \tag{4}$$

B Analysis Methods

B.1 One-Hot Encoding

As described in Appendix A.2, each presented stimulus can be represented using a one-hot vector. For trial q, let $m_l^{(q)}$ denote the stimulus class presented at ordinal position l. We define the task variable $\psi_{l,m}^{(q)} \in \{0, 1\}$ as

$$\psi_{l,m}^{(q)} = \mathbb{I}\Big[m_l^{(q)} = m\Big], \tag{5}$$

where $\mathbb{I}[\cdot]$ denotes the indicator function. For example, if the first stimulus of trial q belongs to class 2, then $\psi_{1,2}^{(q)} = 1$ and $\psi_{1,m}^{(q)} = 0$ for all $m \neq 2$. Accordingly, the one-hot encoding of the stimulus at position l is given by

$$\boldsymbol{\psi}_l^{(q)} = [\psi_{l,1}^{(q)}, \psi_{l,2}^{(q)}, \ldots, \psi_{l,6}^{(q)}]^\top \in \{0, 1\}^6, \quad l = 1, 2, 3. \tag{6}$$

For an arbitrary trial, the input stimulus sequence of length three can thus be represented by concatenating the three one-hot vectors into an 18-dimensional vector with three non-zero entries

$$\boldsymbol{\psi}^{(q)} = \Big[(\boldsymbol{\psi}_1^{(q)})^\top, (\boldsymbol{\psi}_2^{(q)})^\top, (\boldsymbol{\psi}_3^{(q)})^\top\Big]^\top \in \{0, 1\}^{18}. \tag{7}$$

For example, the sequence $[2, 3, 4]$ is encoded as $[0, 1, 0, 0, 0, 0; 0, 0, 1, 0, 0, 0; 0, 0, 0, 1, 0, 0]^\top$. Collecting the one-hot encodings of all possible sequences yields the design matrix

$$\boldsymbol{\Psi} = \begin{bmatrix} (\boldsymbol{\psi}^{(1)})^\top \\ (\boldsymbol{\psi}^{(2)})^\top \\ \vdots \\ (\boldsymbol{\psi}^{(N_{\text{tr}})})^\top \end{bmatrix} \in \mathbb{R}^{N_{\text{tr}} \times 18}. \tag{8}$$

B.2 Linear Regression

B.2.1 Static Linear-Regression Estimation of Rank Subspaces

Following the approach of Xie et al. (2022), for a given trial q, the time-averaged firing rate $\overline{r}_i^{(q)}$ of the i-th hidden neuron over a specified time interval is modeled as a linear combination of task variables. We estimate the coefficients with L1-regularized linear regression (Lasso), matching the analysis code used for Fig. 2A.

$$\overline{r}_i^{(q)} = \sum_{l=1}^{3} \sum_{m=1}^{6} \beta_i(l,m)\psi_{l,m}^{(q)} + \epsilon_i^{(q)}, \tag{9}$$

where $\beta_i(l,m)$ denotes the regression coefficient and $\epsilon_i^{(q)}$ is the residual term. Let $\overline{\mathbf{r}}_i \in \mathbb{R}^{N_{\text{tr}}}$ denote the vector of time-averaged firing rates of neuron i within each trial over the selected time window. In Fig. 2A, this window corresponds to the last 10 timesteps of the maintenance phase. Using the one-hot design matrix $\mathbf{\Psi} \in \mathbb{R}^{N_{\text{tr}} \times 18}$, the linear model is written as

$$\overline{\mathbf{r}}_i = \mathbf{\Psi}\boldsymbol{\beta}_i + \boldsymbol{\epsilon}_i, \tag{10}$$

where $\boldsymbol{\beta}_i \in \mathbb{R}^{18}$ is the Lasso coefficient vector for neuron i, and $\boldsymbol{\epsilon}_i$ is the residual vector.

Stacking the regression coefficients of all hidden neurons yields the full coefficient matrix

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^\top \\ \boldsymbol{\beta}_2^\top \\ \vdots \\ \boldsymbol{\beta}_{N_{\text{rec}}}^\top \end{bmatrix} \in \mathbb{R}^{N_{\text{rec}} \times 18}, \tag{11}$$

such that the regression relationship between neural population activity and task variables can be expressed as

$$\overline{\mathbf{r}} = \mathbf{\Psi}\boldsymbol{\beta}^\top + \boldsymbol{\epsilon}, \tag{12}$$

where $\overline{\mathbf{r}} \in \mathbb{R}^{N_{\text{tr}} \times N_{\text{rec}}}$ denotes the time-averaged firing rates of all hidden neurons within each trial and $\boldsymbol{\epsilon} \in \mathbb{R}^{N_{\text{tr}} \times N_{\text{rec}}}$ is the residual matrix. Coefficients are fitted independently for each neuron using L1-regularized linear regression,

$$\boldsymbol{\beta}_i = \arg\min_{\mathbf{b} \in \mathbb{R}^{18}} \|\mathbf{\Psi}\mathbf{b} - \overline{\mathbf{r}}_i\|_2^2 + \lambda_{\text{L1}}\|\mathbf{b}\|_1. \tag{13}$$

In the implementation, this L1-regularized regression was fitted with `sklearn` Lasso using $\alpha = 0.001$.

Using the methods described in Appendix B.3.1, low-dimensional subspaces can be identified from the resulting regression coefficients $\boldsymbol{\beta}$.

B.2.2 Temporal Evolution of Rank Subspaces via Time-Resolved Lasso

Following Chen et al. (2024), we apply time-resolved L1-regularized linear regression to characterize the temporal evolution of neural representations in low-dimensional subspaces. With a window size of $\Delta t = 2$ timesteps, each trial is partitioned from trial onset to the end of the maintenance phase into N_{win} non-overlapping windows,

$$\mathcal{T}_w = [t_w, t_{w+1}), \quad t_w = w\Delta t, \quad w = 0, \ldots, N_{\text{win}} - 1. \tag{14}$$

Within each window, firing rates are averaged over time to yield $\overline{\mathbf{r}}(t_w) \in \mathbb{R}^{N_{\text{tr}} \times N_{\text{rec}}}$. For each window, the same Lasso regression is used to fit the time-resolved coefficients

$$\boldsymbol{\beta}_i(t_w) = \arg\min_{\mathbf{b}\in\mathbb{R}^{18}} \|\boldsymbol{\Psi}\mathbf{b} - \bar{\mathbf{r}}_i(t_w)\|_2^2 + \lambda_{\mathrm{L1}}\|\mathbf{b}\|_1, \tag{15}$$

resulting in linear-regression coefficient matrices $\boldsymbol{\beta}(t_w) \in \mathbb{R}^{N_{\mathrm{rec}}\times 18}$ for all windows, where $w \in \{0, 1, \ldots, N_{\mathrm{win}} - 1\}$. According to the one-hot encoding scheme in Appendix B.1, each coefficient matrix can be partitioned by rank as

$$\boldsymbol{\beta}(t_w) = \left[\boldsymbol{\beta}^{(1)}(t_w) \quad \boldsymbol{\beta}^{(2)}(t_w) \quad \boldsymbol{\beta}^{(3)}(t_w)\right], \tag{16}$$

where $\boldsymbol{\beta}^{(l)}(t_w) \in \mathbb{R}^{N_{\mathrm{rec}}\times 6}$ corresponds to rank l, $l = 1, 2, 3$.

For the memory subspace, PCA is performed on the coefficient matrix $\boldsymbol{\beta}(t_{N_{\mathrm{win}}-1})$ from the final maintenance window. For the sensory subspace, PCA is applied to the arithmetic mean of the coefficient matrices spanning stimulus onset to offset.

B.3 Subspaces Identified by PCA

We apply PCA to linear-regression coefficient matrices and population firing-rate matrices to identify low-dimensional subspaces.

B.3.1 PCA on Linear-Regression Coefficient Matrices

Given a linear-regression coefficient matrix $\boldsymbol{\beta}^* \in \mathbb{R}^{N_{\mathrm{rec}}\times 6}$ encoding neural responses to the six stimulus identities within a given rank, we first subtract the mean across stimulus identities for each neuron

$$\langle\boldsymbol{\beta}^*\rangle = \frac{1}{6}\sum_{m=1}^{6} \boldsymbol{\beta}^*_{:,m} \in \mathbb{R}^{N_{\mathrm{rec}}}, \tag{17}$$

and define the centered matrix $\tilde{\boldsymbol{\beta}}^* = \boldsymbol{\beta}^* - \langle\boldsymbol{\beta}^*\rangle\mathbf{1}_6^\top$. Singular value decomposition yields $\tilde{\boldsymbol{\beta}}^* = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$. Retaining the N_{PC} largest singular values defines an orthonormal basis $\mathbf{U}_{\mathrm{PC}} \in \mathbb{R}^{N_{\mathrm{rec}}\times N_{\mathrm{PC}}}$. Projecting regression coefficients onto this subspace gives the low-dimensional representation

$$\boldsymbol{\beta}^*_{\mathrm{PC}} = \mathbf{U}_{\mathrm{PC}}^\top\tilde{\boldsymbol{\beta}}^* \in \mathbb{R}^{N_{\mathrm{PC}}\times 6}. \tag{18}$$

Based on the singular values $\{\sigma_k\}$ obtained from the SVD, the explained variance of the k-th PC is defined as

$$\mathrm{EV}_k = \frac{\sigma_k^2}{\sum_j \sigma_j^2}. \tag{19}$$

The cumulative explained variance up to the k-th PC is

$$\mathrm{CEV}_k = \sum_{j=1}^{k} \mathrm{EV}_j. \tag{20}$$

B.3.2 PCA on Population Firing Rates

Let $\overline{\mathbf{R}} \in \mathbb{R}^{N_{\mathrm{tr}}\times N_{\mathrm{rec}}}$ denote the within-trial time-averaged firing rates over a given time window, with trials as samples and recurrent units as features. PCA is fitted to this matrix after centering across trials for each recurrent unit,

$$\langle\overline{\mathbf{R}}\rangle = \frac{1}{N_{\mathrm{tr}}}\sum_{q=1}^{N_{\mathrm{tr}}} \overline{\mathbf{R}}_{q,:}^\top \in \mathbb{R}^{N_{\mathrm{rec}}}, \tag{21}$$

giving $\tilde{\mathbf{R}} = \overline{\mathbf{R}} - \mathbf{1}_{N_{\text{tr}}}\langle\overline{\mathbf{R}}\rangle^\top$, where $\mathbf{1}_{N_{\text{tr}}}$ is the all-ones column vector. Singular value decomposition $\tilde{\mathbf{R}} = \mathbf{Q}\mathbf{\Lambda}\mathbf{V}^\top$ yields the basis $\mathbf{V}_{\text{PC}} \in \mathbb{R}^{N_{\text{rec}} \times N_{\text{PC}}}$, and projecting the centered firing rate matrix onto this subspace gives the low-dimensional representation

$$\mathbf{R}_{\text{PC}} = \tilde{\mathbf{R}}\mathbf{V}_{\text{PC}} \in \mathbb{R}^{N_{\text{tr}} \times N_{\text{PC}}}. \tag{22}$$

Explained and cumulative variance are computed from the singular values using Eqs. (19) and (20).

B.4 Canonical Correlation Analysis Between Low-Dimensional Representations

Because PC bases can change across task stages, we use CCA to compare rank-specific information in two time-local low-dimensional representations. Given two PCA representations $\mathbf{R}_1, \mathbf{R}_2 \in \mathbb{R}^{N_{\text{tr}} \times 2}$ (Eq. (22)), CCA finds linear combinations $\mathbf{U}_k = \mathbf{R}_1\mathbf{a}_k$ and $\mathbf{V}_k = \mathbf{R}_2\mathbf{b}_k$, with $\mathbf{a}_k, \mathbf{b}_k \in \mathbb{R}^2$, that maximize their correlation,

$$\rho_k = \max_{\mathbf{a}_k, \mathbf{b}_k} \operatorname{corr}(\mathbf{R}_1\mathbf{a}_k, \mathbf{R}_2\mathbf{b}_k), \quad k = 1, 2. \tag{23}$$

Successive canonical variates are constrained to be uncorrelated within each representation. The arithmetic mean of the two canonical correlations ρ_1 and ρ_2 measures similarity between the two time-local representations.

B.5 Computation of Angles Between Low-Dimensional Subspaces

As shown in Appendix C.1, each low-dimensional subspace is spanned by two orthonormal basis vectors in $\mathbb{R}^{N_{\text{rec}}}$. In the following analysis, we compute angles between such subspaces as well as between subspaces and weight matrices.

B.5.1 Angles Between Subspaces

Given two subspaces with orthonormal bases $\mathbf{U}_{\text{PC}}^{(1)}, \mathbf{U}_{\text{PC}}^{(2)} \in \mathbb{R}^{N_{\text{rec}} \times 2}$, QR decomposition yields orthonormal column bases $\mathbf{Q}^{(1)}$ and $\mathbf{Q}^{(2)}$. The matrix $M = (\mathbf{Q}^{(1)})^\top\mathbf{Q}^{(2)}$ is decomposed as $M = \mathbf{U}_M\mathbf{\Sigma}_M\mathbf{V}_M^\top$ using SVD, where $\mathbf{\Sigma}_M = \operatorname{diag}(\cos\theta_1, \cos\theta_2)$. The arithmetic mean of θ_1 and θ_2 is taken as the angle between the two subspaces.

To quantify the incremental rotation of $\mathcal{U}_1$ at S_3 relative to S_2, $\mathcal{U}_1$ at S_1 is used as a reference, and the difference between the cumulative rotations from S_1 to S_2 and from S_1 to S_3 is reported.

B.5.2 Phase-to-Maintenance Linear Transform Fits

To test whether the observed subspace changes are well described by pure orthogonal reorientation, we fit low-dimensional transforms from encoding and inter-stimulus phases to the maintenance phase. For each phase, hidden activity is averaged over the corresponding time window, PCA is fit to the phase-specific population activity, and the two-dimensional coordinates associated with each available rank subspace are retained. For a source phase and the matched maintenance subspace, let $\mathbf{Z}_{\text{src}}, \mathbf{Z}_{\text{mnt}} \in \mathbb{R}^{N_{\text{tr}} \times 2}$ denote the corresponding PCA coordinates.

We first fit an unconstrained linear map $\mathbf{A}$ by least squares,

$$\mathbf{A}^* = \arg\min_{\mathbf{A}} \|\mathbf{Z}_{\text{src}}\mathbf{A} - \mathbf{Z}_{\text{mnt}}\|_F^2, \tag{24}$$

and compare it with the best orthogonal map $\mathbf{Q}$ obtained from the Procrustes solution,

$$\mathbf{Q}^* = \arg\min_{\mathbf{Q}^\top\mathbf{Q}=\mathbf{I}} \|\mathbf{Z}_{\text{src}}\mathbf{Q} - \mathbf{Z}_{\text{mnt}}\|_F^2. \tag{25}$$

Fit quality is summarized by relative R^2, using the squared norm of $\mathbf{Z}_{\text{mnt}}$ as the denominator. In Table S2, t_M denotes the maintenance window and $t_{\text{int},k}$ denotes the inter-stimulus interval after stimulus k. *Angle* is the mean principal angle between the source-phase and maintenance-phase

subspace bases. *Linear* R^2 is the fit quality of the unconstrained map $\mathbf{A}^*$, *Orth.* R^2 is the fit quality of the best orthogonal map $\mathbf{Q}^*$, and ΔR^2 denotes their difference, Linear R^2 − Orth. R^2. We also compute the singular values of $\mathbf{A}^*$; values different from one indicate stretch or shear beyond a pure orthogonal reorientation. Across source-phase to maintenance fits, the unconstrained linear map often outperforms the orthogonal map and has singular values that deviate from one, indicating scaling and shear in addition to rotation.

Table S2: **Phase-to-maintenance linear transform fits reveal non-orthogonal components of subspace evolution.** The unconstrained linear map and the best orthogonal map were fitted between matched two-dimensional PCA coordinates for the indicated source phase and maintenance subspace. Singular values different from one indicate stretch or shear beyond a pure orthogonal reorientation; across these phase-to-maintenance fits, singular values range from 0.65 to 3.60.

Source → maintenance	Subspace	Angle (°)	Linear R^2	Orth. R^2	ΔR^2	Singular values
$t_{S_1} \to t_M$	$\mathcal{U}_1$	85.6	1.00	0.50	0.49	3.60, 3.12
$t_{\mathrm{int},1} \to t_M$	$\mathcal{U}_1$	86.9	1.00	0.69	0.30	2.42, 2.03
$t_{S_2} \to t_M$	$\mathcal{U}_1$	87.9	0.91	0.65	0.25	2.26, 1.95
$t_{S_2} \to t_M$	$\mathcal{U}_2$	87.2	0.90	0.72	0.19	1.89, 1.75
$t_{\mathrm{int},2} \to t_M$	$\mathcal{U}_1$	88.0	0.89	0.70	0.20	2.00, 1.75
$t_{\mathrm{int},2} \to t_M$	$\mathcal{U}_2$	71.0	0.89	0.88	0.01	1.16, 1.04
$t_{S_3} \to t_M$	$\mathcal{U}_1$	85.1	1.00	0.82	0.17	1.78, 1.65
$t_{S_3} \to t_M$	$\mathcal{U}_2$	88.8	0.84	0.82	0.01	1.19, 1.06
$t_{S_3} \to t_M$	$\mathcal{U}_3$	86.5	0.83	0.75	0.09	1.50, 1.47
$t_{\mathrm{int},3} \to t_M$	$\mathcal{U}_1$	64.2	0.99	0.92	0.07	1.42, 1.33
$t_{\mathrm{int},3} \to t_M$	$\mathcal{U}_2$	65.0	0.66	0.62	0.04	1.11, 0.70
$t_{\mathrm{int},3} \to t_M$	$\mathcal{U}_3$	65.0	0.63	0.56	0.07	0.96, 0.65

B.5.3 Angles Between Subspaces and $\mathbf{W}_{\text{IN}}$

Given a subspace basis $\mathbf{U}_{\text{PC}} \in \mathbb{R}^{N_{\text{rec}} \times 2}$ and the input weight matrix $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_{\text{rec}} \times N_{\text{in}}}$, QR decomposition yields orthonormal column bases $\mathbf{Q}^{(1)}$ and $\mathbf{Q}^{(2)}$. The matrix $M = (\mathbf{Q}^{(1)})^\top \mathbf{Q}^{(2)}$ is decomposed as $M = \mathbf{U}_M \mathbf{\Sigma}_M \mathbf{V}_M^\top$ using reduced SVD, where $\mathbf{\Sigma}_M = \mathrm{diag}(\cos\theta_1, \cos\theta_2)$. The arithmetic mean of θ_1 and θ_2 is taken as the angle between the subspace and the column space of $\mathbf{W}_{\text{in}}$ in recurrent-unit space.

To ensure that the stage-specific input-weight alignment does not depend on this particular summary of principal angles, we repeated the comparison using several alternative criteria. Let θ_1 and θ_2 denote the two principal angles between the rank subspace and $\mathrm{span}(\mathbf{W}_{\mathrm{in}})$. We computed the minimum principal angle $\min_i \theta_i$, maximum principal angle $\max_i \theta_i$, root-mean-square principal angle $\sqrt{(\theta_1^2 + \theta_2^2)/2}$, chordal distance $\sqrt{\sum_i \sin^2 \theta_i}$, normalized chordal distance $\sqrt{\frac{1}{2}\sum_i \sin^2 \theta_i}$, and projection overlap $\frac{1}{2}\sum_i \cos^2 \theta_i$. We also computed the *Median vector angle* and *Min vector angle* by measuring the angle between each individual input-weight vector (each column of $\mathbf{W}_{\mathrm{in}}$) and the target rank subspace, then taking the median or minimum across input vectors, respectively. For angle and distance criteria, smaller values indicate stronger alignment; for projection overlap, larger values indicate stronger alignment. All criteria identified the currently encoded rank subspace as the most strongly aligned subspace at each encoding stage (Table S3).

B.6 Contribution of $\mathbf{W}_{\text{IN}}$ and $\mathbf{W}_{\text{OUT}}$ to Different Subspaces

To quantify the relative contribution of the input and output weights to each PCA-derived subspace $\mathcal{U}_l$, we weight each PC's squared weight projection by its own activity variance. For the input weights, the raw contribution is

$$C_l^{\mathrm{raw,in}} = \sum_{\mathbf{p}_k \in \mathcal{U}_l} \lambda_k \|\mathbf{W}_{\mathrm{in}}^\top \mathbf{p}_k\|^2, \tag{26}$$

where $\mathbf{p}_k$ is a unit PC vector and λ_k is its eigenvalue, computed from trial-to-trial variation in window-averaged neural activity. For the output weights, we analogously define

Table S3: **Input-weight alignment is consistent across alternative subspace-alignment criteria.** Across angle-, distance-, overlap-, and vector-based criteria, the currently encoded rank subspace is consistently most strongly aligned with $\mathbf{W}_{\text{in}}$. Bold entries indicate the strongest alignment within each encoding stage. PA denotes principal angle; RMS denotes root mean square.

	Stimulus 1			Stimulus 2			Stimulus 3		
Criterion	$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_3$	$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_3$	$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_3$
Mean PA (°)	**20.8**	67.2	73.0	51.9	**33.4**	70.5	68.2	52.2	**36.3**
Min PA (°)	**20.5**	61.5	68.8	47.6	**31.8**	63.8	63.4	49.5	**34.3**
Max PA (°)	**21.0**	72.9	77.1	56.3	**35.1**	77.2	72.9	54.8	**38.3**
RMS PA (°)	**20.8**	67.4	73.1	52.1	**33.5**	70.8	68.3	52.2	**36.4**
Chordal distance	**0.50**	1.30	1.35	1.11	**0.78**	1.33	1.31	1.12	**0.84**
Norm. chordal distance	**0.35**	0.92	0.95	0.79	**0.55**	0.94	0.93	0.79	**0.59**
Projection overlap	**0.87**	0.16	0.09	0.38	**0.70**	0.12	0.14	0.38	**0.65**
Median vector angle (°)	**57.6**	85.8	87.6	74.6	**63.1**	82.3	87.7	74.6	**64.8**
Min vector angle (°)	**53.9**	82.5	86.4	71.8	**58.1**	78.5	85.3	72.1	**60.4**

$$C_l^{\text{raw,out}} = \sum_{\mathbf{p}_k \in \mathcal{U}_l} \lambda_k \|\mathbf{W}_{\text{out}} \mathbf{p}_k\|^2. \tag{27}$$

For each weight matrix, we normalize within each model and time window over the retained PC-pair subspaces,

$$C_l = \frac{C_l^{\text{raw}}}{\sum_j C_j^{\text{raw}}}. \tag{28}$$

The resulting relative contributions are summarized across models by their mean and sample standard deviation.

B.7 Projected Discrete-Time Jacobian Eigenspectrum

To analyze local recurrent dynamics during maintenance, we use the Euler-discretized RNN update corresponding to Eq. (1),

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{dt}{\tau} \left[-\mathbf{x}_t + \mathbf{W}_{\text{rec}} \phi(\mathbf{x}_t) + \mathbf{W}_{\text{in}} \mathbf{I}_{\text{ext}}(t) + \mathbf{b} \right]. \tag{29}$$

For each trial, we evaluate the Jacobian of this discrete-time map at the final maintenance timestep,

$$\mathbf{J}_t = \frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t}. \tag{30}$$

Let $\mathbf{U}_{\text{mnt}} \in \mathbb{R}^{N_{\text{rec}} \times 6}$ denote the six-dimensional maintenance subspace obtained by fitting PCA to the final maintenance states. The local Jacobian is projected into this subspace as

$$\mathbf{J}_t^{\text{sub}} = \mathbf{U}_{\text{mnt}}^\top \mathbf{J}_t \mathbf{U}_{\text{mnt}}. \tag{31}$$

We then compute the eigenvalues of $\mathbf{J}_t^{\text{sub}}$ for each trial and summarize their distribution across models. Because $\mathbf{J}_t$ is the Jacobian of the discrete-time update map, eigenvalue magnitudes near one correspond to a near-marginal local regime for the sampled maintenance states.

We assessed sensitivity to the evaluation time, retained dimensionality, and delay duration within the task's 20–30-timestep training range in five trained models. For each model, we fitted a reference maintenance PCA basis using the 30-timestep delay condition, then evaluated the projected Jacobian at every delay timestep with 2, 4, 6, 8, 10, or 12 retained PCs and delay durations of 20, 25, or 30 timesteps. Exact-sequence recall was 100% at all three evaluated delays. The model-level mean

spectral radius was 1.018 ± 0.002 (mean $\pm$ SD across five models), with each model averaged over the 18 dimensionality-by-duration conditions; condition means across models ranged from 1.014 to 1.025. Thus, the near-unit local spectral statistic was not specific to the final maintenance state, six retained PCs, or a single behaviorally valid delay duration.

B.8 Visualization of Subspace Rotation

We use time-resolved PCA to visualize the rotation of low-dimensional subspaces over time.

B.8.1 Sliding-Window PCA

At an arbitrary time point t, denote the firing rates of all hidden-layer neurons across trials as $\mathbf{R}(t) \in \mathbb{R}^{N_{\text{tr}} \times N_{\text{rec}}}$. For a given task interval $[t_a, t_b)$, consider the neural activity sequence $\{\mathbf{R}(t)\}_{t=t_a}^{t_b-1}$. To obtain a temporally continuous low-dimensional representation, we construct sliding windows along the time axis with window width $w = 3$ and step size $s = 1$. The total number of windows is thus defined as

$$N_{\text{win}} = \left\lfloor \frac{t_b - t_a - w}{s} \right\rfloor + 1. \tag{32}$$

The k-th time window is then given by

$$\mathcal{W}_k = \{\, t_a + ks,\ t_a + ks + 1,\ \ldots,\ t_a + ks + w - 1 \,\}, \quad k = 0, \ldots, N_{\text{win}} - 1. \tag{33}$$

Within each window $\mathcal{W}_k$, neural activity is averaged over the temporal dimension, yielding

$$\bar{\mathbf{R}}^{(k)} = \frac{1}{w} \sum_{t \in \mathcal{W}_k} \mathbf{R}(t) \in \mathbb{R}^{N_{\text{tr}} \times N_{\text{rec}}}. \tag{34}$$

We apply PCA to $\bar{\mathbf{R}}^{(k)}$ and retain the leading $N_{\text{PC}} = 6$ principal components, yielding $\mathbf{Z}^{(k)} \in \mathbb{R}^{N_{\text{tr}} \times N_{\text{PC}}}$. For visualization, we select three components and average their coordinates across trials with the same stimulus identity at the selected rank.

B.8.2 Sign Alignment of Principal Components

Independently fitted PCs can differ in sign across adjacent windows. Let $\mathbf{Z}_i^{(k)} = \mathbf{Z}_{:,i}^{(k)}$ denote the trial-score vector for component i in window k. For $k \geq 1$, we flip its sign when $\langle \mathbf{Z}_i^{(k)}, \mathbf{Z}_i^{(k-1)} \rangle < 0$, using the aligned scores from the preceding window.

B.8.3 Two-Dimensional Subspace Fitting

Within each time window k, we fit a two-dimensional plane to the three-dimensional item-centroid coordinates $\mathbf{Y}^{(k)} \in \mathbb{R}^{M \times 3}$. SVD of the centered coordinates gives $\tilde{\mathbf{Y}}^{(k)} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. The first two right singular vectors in $\mathbf{V}$ span the best-fitting plane, and the third is its normal vector.

B.9 Single-Neuron Analysis Methods

B.9.1 Tuning Curve Computation

The linear-regression coefficient vector $\boldsymbol{\beta}_i \in \mathbb{R}^{18}$, which is obtained in Appendix B.2.1, characterizes the sensitivity of neuron i to the input sequence over a given time window. According to the one-hot encoding scheme, it can be partitioned by rank as

$$\boldsymbol{\beta}_i = \begin{bmatrix} \boldsymbol{\beta}_i^{(1)} \\ \boldsymbol{\beta}_i^{(2)} \\ \boldsymbol{\beta}_i^{(3)} \end{bmatrix}, \tag{35}$$

where

$$\boldsymbol{\beta}_i^{(l)} = [\beta_i(l,1), \beta_i(l,2), \ldots, \beta_i(l,6)]^\top \in \mathbb{R}^6, \quad l = 1,2,3. \tag{36}$$

These coefficients define neuron i's tuning across stimulus directions at each rank. Under the gain-modulation framework of Xie et al. (2022),

$$\beta_i(l,m) = \lambda_l A_{li} \cos(\theta_m - \varphi_{li}), \tag{37}$$

where λ_l is the rank modulation factor, A_{li} is the projection magnitude in the rank-l subspace, θ_m denotes stimulus direction, and φ_{li} is the preferred direction. We estimate φ_{li} by fitting this cosine function to the rank-specific tuning curve.

B.9.2 Selection of Rotation-Sensitive Neurons

We identify neurons that are most sensitive to representational subspace rotation using a perturbation-based sensitivity analysis. Let $\mathbf{R}(t) \in \mathbb{R}^{N_{\text{tr}} \times N_{\text{rec}}}$ denote the population activity of the hidden layer at time t. For rank l, consider the time window from stimulus l onset to offset as $\mathcal{T}_{\text{input}}^{(l)} = [t_{\text{start}}^{(l)}, t_{\text{end}}^{(l)})$, $l = 1,2,3$ and the time window of the total maintenance phase as $\mathcal{T}_{\text{mnt}}$. Population activity is averaged over time within each window, and PCA is applied separately to obtain low-dimensional subspaces. Selecting the corresponding two principal components associated with rank l, we obtain the stimulus and delay subspace bases.

We compute the baseline rotation θ_l between the rank-l stimulus and maintenance subspaces as described in Appendix B.5.1. To assess single-neuron contributions, we perturb the stimulus-period activity of neuron i by a small additive offset $\varepsilon_i^{(l)} = \sigma_i^{(m,l)} \cdot \lambda$, where $\sigma_i^{(m,l)}$ is its firing-rate standard deviation during $\mathcal{T}_{\text{input}}^{(l)}$ and $\lambda = 10^{-2}$. Recomputing the stimulus subspace gives the perturbed rotation $\theta_{l,i}^{+}$. We define rotation sensitivity by the finite difference

$$s_{l,i} = \frac{\theta_{l,i}^{+} - \theta_l}{\varepsilon_i^{(l)}}. \tag{38}$$

To obtain a rank-agnostic measure, sensitivities are aggregated across ranks, yielding

$$\bar{s}_i = \frac{1}{3} \sum_{l=1}^{3} |s_{l,i}|. \tag{39}$$

We rank neurons by $\bar{s}_i$ and select the top 50 (12.5% of the recurrent population) for analysis.

B.9.3 Neuron-to-Subspace Strength Metric

Based on the tuning analysis above, for each neuron i we obtain its projection magnitude in the rank-l subspace, denoted by A_{li}. As defined in Li et al. (2025), the neuron-to-subspace strength (NSS) index quantifies the extent to which a neuron contributes preferentially to one subspace or is shared across a pair of subspaces. For ranks a and b ($a < b$), the NSS index is defined as

$$\text{NSS}_{a-b} = \frac{A_{ai} - A_{bi}}{A_{ai} + A_{bi}} \in [-1, 1]. \tag{40}$$

When $\text{NSS}_{a-b} \to 0$, neuron i contributes nearly equally to the two subspaces and is therefore categorized as an overlapping neuron. When $\text{NSS}_{a-b} \to \pm 1$, neuron i contributes mainly to one subspace and is categorized as a disjoint neuron. Following Li et al. (2025), we classify neurons with $|\text{NSS}| > 0.4$ as disjoint.

B.10 Evaluation of Model Output Performance

B.10.1 Accuracy of Model Outputs

During the retrieval phase, the model's outputs are evaluated sequentially. For the j-th retrieval position ($j = 1, 2, \ldots, L$), the predicted label $\mathrm{label}_{i,j}$ is obtained by selecting the maximal activation and compared to the ground-truth label $\mathrm{target}_{i,j}$. Rank-wise accuracy is defined as

$$\mathrm{Acc}_j = \frac{1}{N_{\text{eval}}} \sum_{i=1}^{N_{\text{eval}}} \mathbb{I}(\mathrm{label}_{i,j} = \mathrm{target}_{i,j}), \quad j = 1, 2, \ldots, L, \tag{41}$$

where N_{eval} is the number of test samples and $\mathbb{I}(\cdot)$ is the indicator function. For the main SWM evaluation, $N_{\text{eval}} = 12{,}000$.

B.10.2 Perturbation on Rank-l Input

Denote the input vector at timestep t as $\mathbf{I}_{\text{ext}}(t) \in \mathbb{R}^{N_{\text{in}}}$. For the l-th stimulus input, denote the time window from onset to offset as $\mathcal{T}_{\text{input}}^{(l)} = [t_{\text{start}}^{(l)}, t_{\text{end}}^{(l)}),\ l = 1, 2, 3$. During this window, we add independent Gaussian noise with standard deviation σ_{pert} to the 24 stroke-input channels; the fixation channel is unchanged. In Fig. 4C and Fig. S21, $\sigma_{\text{pert}} = 5.0$.

B.10.3 Effect of Noise on Different Subspaces

We evaluate the model with noise-free inputs and with noise injected during one stimulus presentation as described in Appendix B.10.2. PCA of the noise-free firing rates over the full maintenance phase defines the rank-subspace bases. For each rank l, we project activity from both conditions onto its basis, yielding coordinates $(P_0)_l$ and $(P_1)_l$, respectively. We quantify the relative variance change as

$$\delta_l = \frac{|\mathrm{var}\,(P_0)_l - \mathrm{var}\,(P_1)_l|}{(\mathrm{var}\,(P_0)_l + \mathrm{var}\,(P_1)_l)/2}, \tag{42}$$

where larger δ_l values indicate stronger noise-induced perturbations in the corresponding subspace.

C Supplementary Experiments

C.1 Neural Geometric Representations Identified via PCA

Using the methods described in Appendices B.2.1 and B.3, we first performed a linear regression analysis on the hidden-layer firing rates from the final 10 timesteps of the maintenance phase. We then applied PCA to the resulting linear-regression coefficient matrix $\boldsymbol{\beta}$, conducting dimensionality reduction separately for each rank and retaining $N_{\text{PC}} = 6$ principal components. The cumulative variance explained by the PCs for different ranks was subsequently computed, as shown in Fig. S2. We observed that the cumulative variance explained by $\mathrm{PC}_1, \mathrm{PC}_2$ reached 97.2%, indicating that information of rank l is effectively stored within the subspace spanned by $\mathrm{PC}_1^{(l)}, \mathrm{PC}_2^{(l)}$.

Based on this result, we further performed PCA on the time-averaged population neural activity states within each trial (vectors of size $N_{\text{rec}} \times 1$) across all N_{tr} trials during the maintenance phase. Because PCA is an unsupervised method, an appropriate criterion is required to determine whether neural representations are indeed stored within a particular PCA-derived subspace.

First, since PCA is a global linear orthogonal transformation, it does not alter the dimensionality of the minimal linear subspace in which the neural activity data are embedded. Given that content information has been shown, via the linear regression analysis, to reside in a two-dimensional subspace, it follows that after applying PCA to the hidden-layer firing rates, the sequence content information should still be encoded within a two-dimensional subspace spanned by a specific pair of PCs. This expectation is also supported by the results shown in Fig. 2C–E.

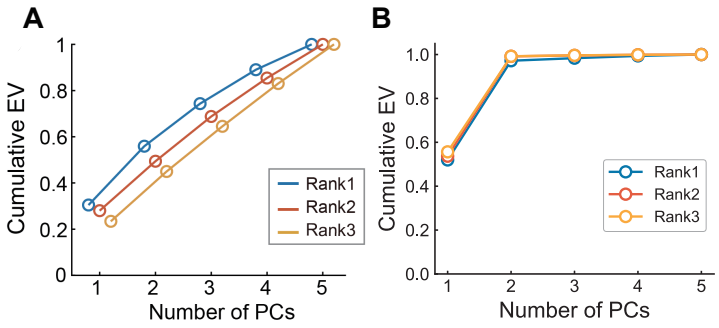

Figure S2: **A small number of PCs explain most rank-related maintenance variance in both experiment and model.** (A) Experimental result: The cumulative explained variance along different ranks in neural state space. Adapted from Xie et al. (2022) Fig. S3A. (B) Model result: The cumulative explained variance along different ranks in neural state space during the late maintenance phase in the model.

Among the N_{PC} principal components obtained from PCA, we therefore need to determine which pair of PCs spans the relevant two-dimensional subspace, as well as how to color the projected data points within that subspace. Since PCA preserves the pre-existing low-dimensional geometric manifold representations, if a given PC pair encodes the content information at a particular sequence position, then the projections corresponding to the M stimulus types should be topologically equivalent to the ring structure obtained from the linear regression analysis. Specifically, these points should form M clearly separated clusters arranged in a ring-like topology, with each cluster representing a distinct stimulus feature. This constitutes the assumption adopted in Sec. 2.2.

We next detail the concrete procedure associated with this assumption. For each trial, the average neural activity vector was projected onto all two-dimensional subspaces formed by any pair of the first six PCs, yielding a total of 15 distinct projection patterns. For each trial, the input sequence content $[S_1^{m_1}, S_2^{m_2}, S_3^{m_3}]$ was used as the data labels for ranks 1, 2, and 3, respectively, corresponding to the colors of the filled circles at the appropriate positions in Fig. 1A. Consequently, the N_{tr} projected points within each subspace could be colored using the data labels of any rank. We successively colored the 15 projection patterns using the rank-1, rank-2, and rank-3 labels, with the resulting visualizations shown in Figs. S9, S10, and S11. Only the subspaces spanned by $(\text{PC}_1, \text{PC}_2)$, $(\text{PC}_3, \text{PC}_4)$ and $(\text{PC}_5, \text{PC}_6)$ exhibited clear ring-like manifold structures. Focusing on these subspaces and coloring them sequentially with rank-1, rank-2, and rank-3 labels (Fig. S12), we found that ring-shaped topologies consisting of M distinct, stimulus-ordered clusters emerged only when $(\text{PC}_1, \text{PC}_2)$ was colored by rank-1 labels, $(\text{PC}_3, \text{PC}_4)$ by rank-2 labels, and $(\text{PC}_5, \text{PC}_6)$ by rank-3 labels.

C.2 Training-Time Emergence of Rank-Subspace Geometry

To examine how the event-schema geometry emerges during training, we repeated the cross-stage CCA and maintenance-subspace analyses across training, sampling model states every 20 epochs for visualization. For each sampled state, we computed the mean CCA similarity for subspaces carrying the same rank-specific information and for subspaces carrying different information, the cluster separation of the rank-specific maintenance representations, and the pairwise principal angles between maintenance rank subspaces. Analyses across training show that rank-related structure is detectable early and becomes progressively stabilized: matched-rank CCA increases relative to mismatched representations, maintenance clusters become more separated, and rank subspaces become more orthogonal as training proceeds (Fig. S3).

C.3 Input and Output Coding Controls

To test whether the main geometry depends on the smooth input topology, we trained matched models with narrower Gaussian population inputs and with one-hot inputs. Reducing the Gaussian width preserved much of the rank-specific organization but weakened the smooth ring geometry (Fig. S13).

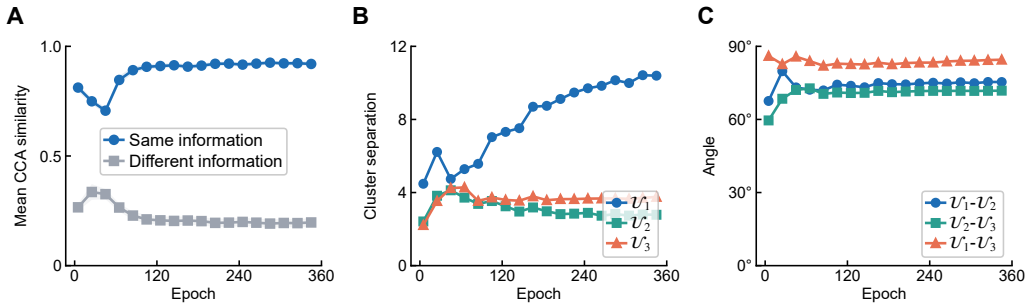

Figure S3: **Rank-subspace geometry emerges and stabilizes across training.** (A) Mean CCA similarity for subspaces carrying the same rank-specific information versus different information across training epochs. (B) Cluster separation of the three rank-specific maintenance representations. (C) Pairwise principal angles between maintenance-period rank subspaces. The analyses indicate that same-rank representational similarity, maintenance cluster separation, and rank-subspace orthogonality progressively stabilize during training.

The one-hot input/one-hot output control likewise retained separated rank subspaces while weakening the ring-like organization (Fig. S14).

We next tested whether rank separation depends on the input or output code, using five training seeds per condition. With Gaussian inputs, replacing one-hot targets with sin/cos coordinates preserved both perfect sequence recall and near-orthogonal rank subspaces (mean between-rank angle: $84.8° \pm 1.4°$ for one-hot and $84.6° \pm 1.4°$ for sin/cos targets). As a stronger control, fixed random-dense inputs contain no predefined circular ordering, yet models using either one-hot or random-dense outputs achieved perfect sequence recall and formed separated rank subspaces (angles: $72.2° \pm 1.5°$ and $73.7° \pm 3.0°$, respectively; Fig. S15). Their semantic ring correlations were near zero, showing that rank separation can emerge without a ring-like item manifold.

Finally, when the assignment between semantic labels and Gaussian input locations was permuted, rank separation remained strong, but the item ring followed the input-code order rather than the semantic label order (PCA ring correlation: 0.930 ± 0.003 versus 0.022 ± 0.057). These controls distinguish the rank-factorized organization preserved across the tested conditions from interface-dependent details of item geometry and exact orthogonality; they do not imply that the geometry is independent of the serial-recall task design.

To test one aspect of this task dependence, we fixed each item-output duration at 12 timesteps and varied the temporal overlap between consecutive outputs, evaluating all models from a fixed-zero initial state. All models reproduced their condition-specific target trajectories perfectly. With a shared six-dimensional item-output code, rank separation remained substantial with 0, 3, 6, or 9 overlapping timesteps (mean between-rank angles: $84.4 \pm 1.1°$, $81.5 \pm 3.1°$, $71.7 \pm 1.6°$, and $69.8 \pm 3.5°$, respectively), but fell to $22.6 \pm 1.8°$ when all three outputs completely overlapped in the same channels and the targets no longer distinguished output rank. In a separate parallel-output control, all ranks were still produced simultaneously but in rank-specific output-channel blocks; the angle remained $81.7 \pm 1.2°$ (mean $\pm$ SD across five seeds). Thus, temporal separation between outputs is not itself required for rank separation; the geometry changed most when the output objective no longer distinguished serial positions.

C.4 Analysis of Retrieval Phase

During the maintenance phase, stimuli of different ranks are allocated to distinct subspaces (e.g., item 1 to PC1–PC2, item 2 to PC3–PC4, and item 3 to PC5–PC6). In the retrieval phase, however, the network consistently generates outputs from the leading subspace $\mathcal{U}_1$ spanned by PC1 and PC2. After each rank is output, its information gradually decays while the next rank becomes dominant in $\mathcal{U}_1$.

Figure S17A–C visualizes this retrieval-stage redistribution of rank information. The relative contribution analysis further shows that the contribution of subspace $\mathcal{U}_1$ to the fixed output-weight space

is close to 1 throughout the three retrieval stages, whereas the contributions from other subspaces are close to 0 (Fig. S17D). This pattern is consistent with serial readout through a common output interface.

Using the method described in Appendix B.4, we further obtained canonical correlation analysis (CCA) heatmaps (Fig. 3A). Because the PC bases can vary across task stages, this analysis asks whether time-local low-dimensional representations encode matched rank-specific information. Matched ranks show high CCA similarity, whereas mismatched ranks show low similarity.

C.5 Input-Weight Ablation Control

To test whether the learned input-weight geometry is functionally involved in cross-stage rank-specific representation, we altered $\mathbf{W}_{\text{in}}$ while keeping all other trained parameters fixed. We compared the trained baseline with two norm-matched interventions: replacing $\mathbf{W}_{\text{in}}$ by a random matrix, and replacing it by a matrix whose input-channel vectors lie in the orthogonal complement of the trained $\mathbf{W}_{\text{in}}$ column space while preserving the corresponding column norms and overall Frobenius norm. The "row-space-orthogonal" label in Fig. S4 follows the transposed matrix convention used in the implementation. Both interventions disrupted the clean matched-rank CCA structure and reduced maintenance-stage rank-subspace separation (Fig. S4), supporting a causal role for learned input-weight alignment in the stage transitions analyzed in the main text.

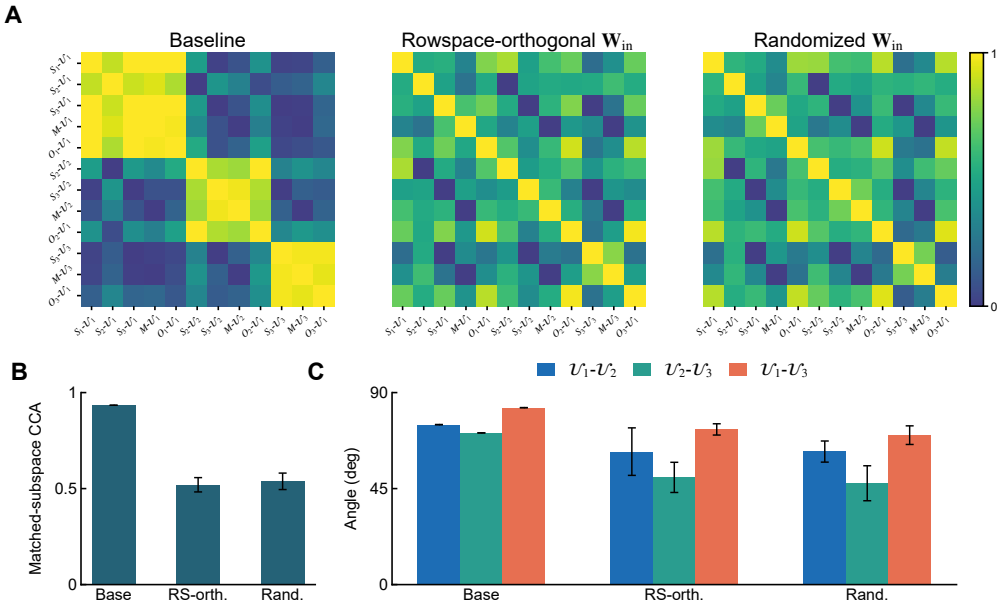

Figure S4: **Input-weight ablations disrupt matched-rank cross-stage representation and rank separation.** (A) All-stage CCA heatmaps for the trained baseline, row-space-orthogonal $\mathbf{W}_{\text{in}}$, and randomized $\mathbf{W}_{\text{in}}$. The row-space-orthogonal control replaces each input-channel vector by a norm-matched vector in the orthogonal complement of the trained $\mathbf{W}_{\text{in}}$ column space under the manuscript's matrix convention. (B) Mean matched-rank CCA across encoding, maintenance, and retrieval. (C) Pairwise principal angles between maintenance-period rank subspaces. In (B–C), RS-orth. denotes row-space-orthogonalized $\mathbf{W}_{\text{in}}$, and Rand. denotes norm-matched randomized $\mathbf{W}_{\text{in}}$. Error bars denote standard deviations over five perturbation realizations applied to one trained model.

As a complementary input-perturbation control, we silenced one stimulus input at a time while keeping the trained network weights fixed, and recomputed the angles between $\mathbf{W}_{\text{in}}$ and the rank subspaces across stimulus-presentation windows. This analysis asks whether input-weight alignment follows the sequence of available stimulus inputs rather than a fixed absolute-time template. When one stimulus is silenced, the stage-wise alignment pattern shifts accordingly: subsequent available stimuli preferentially couple to the earlier available rank subspaces, consistent with the interpretation that $\mathbf{W}_{\text{in}}$ helps route incoming items into sequential rank-specific representations (Fig. S5).

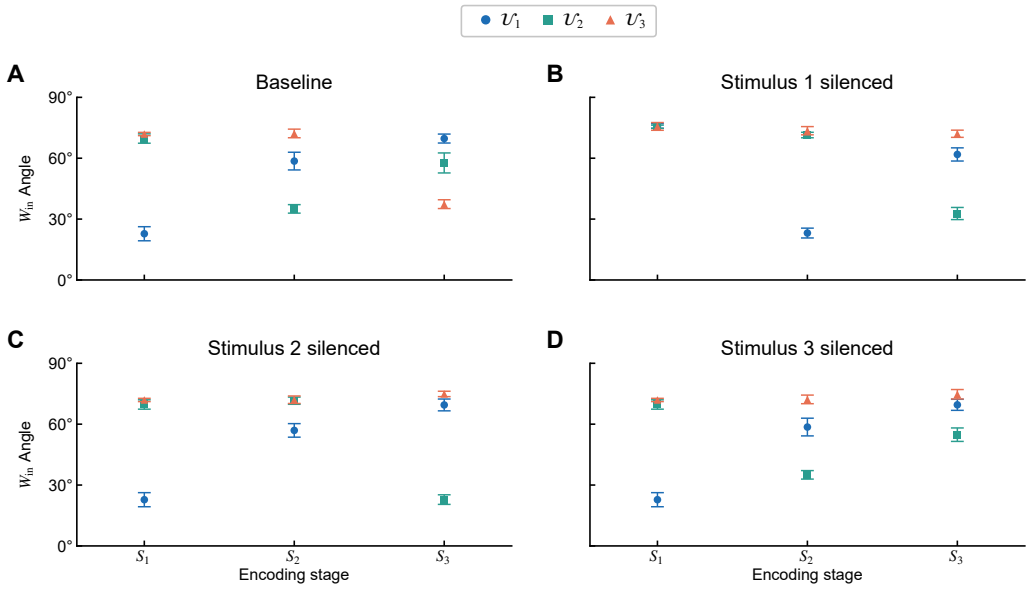

Figure S5: **Stimulus silencing shifts input-weight alignment with rank subspaces.** (A) Baseline angles between $\mathbf{W}_{\text{in}}$ and rank subspaces during t_{S_1}–t_{S_3}. (B–D) The same analysis after silencing stimulus 1, 2, or 3, respectively, while keeping all trained weights fixed. Silencing a stimulus alters the stage-wise alignment sequence, supporting the interpretation that input-weight coupling routes available stimuli into sequential rank-specific subspaces rather than merely following absolute time. Error bars denote standard deviations over 5 seeds.

C.6 Time-Local, Geometry-Preserving Input-Weight Rotation

To vary input-weight orientation while preserving within-window input geometry, we temporarily rotated the column space of $\mathbf{W}_{\text{in}}$ during one stimulus presentation, leaving input-layer activity and all other parameters unchanged. Let $\mathbf{A}$ be an orthonormal basis for the column space of $\mathbf{W}_{\text{in}}$ and $\mathbf{B}$ an equal-dimensional orthonormal basis in its orthogonal complement. We rotated $\mathbf{A}$ to $\mathbf{A}_\theta = \cos(\theta)\mathbf{A} + \sin(\theta)\mathbf{B}$ and used

$$\mathbf{W}_{\text{in}}^{(\theta)} = \mathbf{A}_\theta \mathbf{A}^\top \mathbf{W}_{\text{in}}$$

within the perturbed window, retaining the trained $\mathbf{W}_{\text{in}}$ outside it. The rotation preserves norms and pairwise inner products among the inputs to the recurrent layer within that window while changing their orientation relative to the trained recurrent circuit. We evaluated all 120 ordered three-item sequences at fixed timing and without input noise in five trained RNNs; the rotation includes the input from the fixation channel.

Increasing the rotation angle increased the deviation of item-centroid subspaces from baseline (Table S4). We measured the mean principal angle between two-dimensional subspaces fitted separately to baseline and perturbed firing rates at each endpoint. For each angle, we averaged metrics across the three target ranks within each seed, then summarized them across seeds. At $90°$, target-rank accuracy fell to $6.78 \pm 2.16\%$; accuracy at the two unperturbed ranks also fell by 18.06 ± 5.56 percentage points on average.

The effect depended on intervention duration. When the $90°$ rotation was applied for the first 2, 4, 6, or all 8 stimulus time steps before restoring the trained orientation, target-rank accuracy was $95.56 \pm 3.54\%$, $62.72 \pm 11.39\%$, $21.72 \pm 4.76\%$, and $6.78 \pm 2.16\%$, respectively. During S_2, reorientation of the previously encoded rank-1 subspace decreased from $51.20 \pm 0.75°$ at baseline to $36.72 \pm 1.14°$ under the full $90°$ intervention. Thus, rotating the column space of $\mathbf{W}_{\text{in}}$ impairs target-rank recall more than recall at other ranks, while substantial rank-1 reorientation remains under the perturbation.

Table S4: **Effects of time-local rotation of the column space of $\mathbf{W}_{\text{in}}$.** Entries are means ± SD across five training seeds. Angles compare item-centroid subspaces fitted to baseline and perturbed activity for the targeted rank at each endpoint.

Rotation	Stimulus endpoint	Maintenance endpoint	Retrieval endpoint	Target accuracy
0°	$0.02 \pm 0.00°$	$0.01 \pm 0.01°$	$0.01 \pm 0.01°$	$100.00 \pm 0.00\%$
30°	$32.48 \pm 1.46°$	$12.32 \pm 1.59°$	$6.59 \pm 1.73°$	$96.28 \pm 1.43\%$
60°	$61.94 \pm 1.96°$	$29.52 \pm 2.61°$	$18.54 \pm 2.92°$	$54.00 \pm 10.52\%$
90°	$87.37 \pm 0.46°$	$72.98 \pm 2.30°$	$71.20 \pm 2.75°$	$6.78 \pm 2.16\%$

C.7 Gated-Architecture Controls

We repeated the all-stage CCA analysis in GRU and LSTM models trained under the same task protocol as the vanilla RNN. All models achieved perfect sequence recall, yet their cross-stage CCA patterns differed from that of the vanilla RNN: neither gated architecture reproduced its clean matched-rank block structure (Fig. S6). Thus, the matched-rank organization observed in the vanilla RNN is not necessary for accurate recall; this result is consistent with, but does not identify, alternative internal storage strategies.

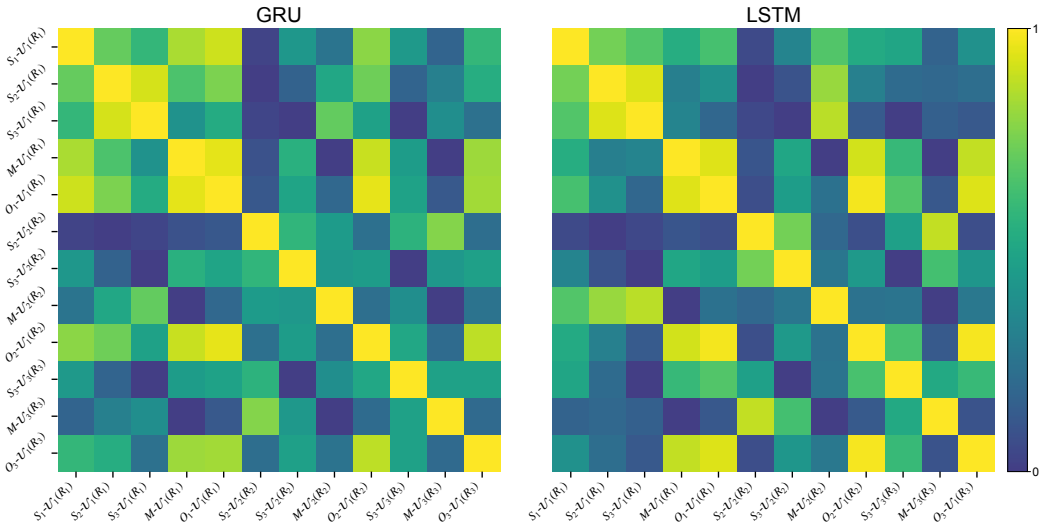

Figure S6: **All-stage CCA analyses for gated-architecture controls.** Cross-stage CCA heatmaps for the GRU (left) and LSTM (right), averaged elementwise across five seeds per architecture. All models achieved 100% exact-sequence accuracy. In contrast to the vanilla RNN, neither gated architecture recovers a clean matched-rank block structure.

C.8 Sensory and Memory Subspaces

Following the approach in Appendix B.2.2, we supplement the main text with an analysis of the trajectories of other memory subspaces during the encoding phase. We find that the experimental results shown in Fig. S22A closely match the model results in Fig. S22B, both in the sensory subspace and across the various memory subspaces.

In our model, as illustrated in the first row of Fig. S22B, the sensory subspace exhibits similar topological structures across the stimulus-presentation time windows t_{S_1}–t_{S_3}, regardless of rank. These ring-like structures emerge transiently upon stimulus presentation and then decay over time.

In contrast, within the memory subspaces, as shown in the second row of Fig. S22B, the Memory-1 subspace (cyan) captures spatial information during t_{S_1}, gradually expanding into a ring-shaped topology that persists through t_{S_2}, t_{S_3}, and the maintenance period. Similarly, the Memory-2 subspace (magenta) captures spatial information during t_{S_2}, subsequently forming a ring-shaped topology

that maintains the memory of the rank-2 item at later times. The Memory-3 subspace (yellow) captures spatial information during t_{S_3}, expands into a ring-shaped topology, and is responsible for maintaining the memory of the final sequence item. These rank-specific ring structures persist throughout the delay period (last three rows of Fig. S22A and B).

C.9 Single-Neuron Analyses

C.9.1 Tuning and Rotation Sensitivity

Using the methods summarized in Appendix B.9, we analyzed rank-dependent single-neuron responses in the trained network. The resulting tuning curves reproduce the two qualitative classes emphasized in the main text: some neurons show rank-dependent gain modulation with relatively stable preferred directions, whereas others show clear preference shifts across ranks (Fig. S7A). To examine whether these shifts are selectively associated with the subspace reorientation observed during encoding, we further applied the rotation-sensitive neuron selection procedure described in Appendix B.9.2. The top 12.5% most sensitive neurons exhibit preference differences that are broadly distributed across ranks rather than concentrated around a single angle (Fig. S7B), consistent with heterogeneous single-neuron contributions to the population-level reorganization.

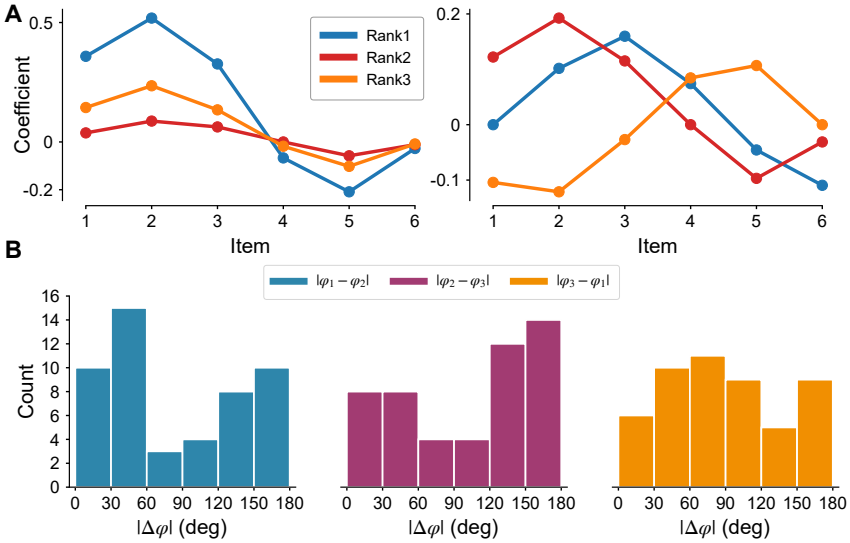

Figure S7: **Rank-dependent tuning properties of single neurons.** (A) Two example neurons with tuning curves illustrating classical gain modulation (left) and preference shift (right). (B) Histogram of preference differences across ranks for the top 12.5% of neurons sensitive to rank-dependent subspace rotation during encoding.

C.9.2 Neuron-to-Subspace Strength (NSS)

We next quantified whether neurons contribute predominantly to one rank subspace or are shared across multiple rank subspaces using the neuron-to-subspace strength (NSS) metric defined in Appendix B.9.3. Across all $N_{\text{rec}} = 400$ hidden-layer neurons, the resulting NSS distributions are generally shifted toward larger absolute values (Fig. S8), indicating a bias toward disjoint rather than strongly overlapping neuron-to-subspace organization. Using the threshold $|\text{NSS}| > 0.4$ from Li et al. (2025), we obtain a broadly similar tendency toward disjoint neurons to that reported in the experimental data.

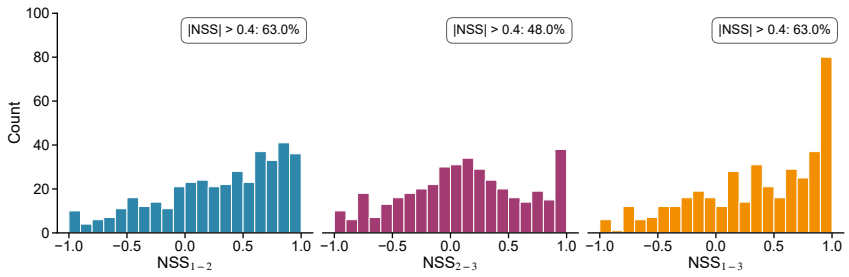

Figure S8: **NSS distribution of all hidden layer neurons.** Values near ± 1 indicate disjoint (contributing primarily to one subspace) neurons, whereas values near 0 indicate overlapping (contributing substantially to multiple subspaces) neurons.

D Handwritten Character Sequence Reproduction Task

D.1 Task Design and Data Generation

Each trial comprised encoding, maintenance (delay), and retrieval stages. During encoding, the network received 2–3 elementary strokes in sequence. Item inputs stopped during maintenance, but the fixation input remained active. During retrieval, the fixation input switched off and the network reproduced the strokes in order.

D.2 Definition of Input and Output Signals

Each stroke had a direction and length in a two-dimensional plane. We selected its direction from eight evenly spaced angles, $\theta_s = 2\pi s/8$ for $s = 0, 1, \ldots, 7$, covering horizontal, vertical, left-falling, right-falling, and related strokes. As in Appendix A.2, 24 circularly tuned input channels encoded direction as a Gaussian population-code bump. One of three amplitudes, $A_s \in \{0.6,\ 1.0,\ 1.4\}$, scaled the bump to encode stroke length.

A separate fixation input was 1 during encoding and maintenance and 0 during retrieval, giving $N_{\text{in}} = 25$. The output had the same dimensionality: 24 population-coded stroke channels and one fixation channel.

D.3 Model Architecture and Parameters

The recurrent core matched the main SWM model (Appendix A). The task interfaces differed: the 25 input channels encoded stroke direction and amplitude rather than six discrete item identities, and the output layer had 25 channels instead of seven. The initial hidden state was trainable. We trained each model for 300 epochs with a batch size of 64.

The training and evaluation sets contained 9,500 and 1,000 trials, respectively. Each trial contained two or three strokes with equal probability; stroke directions and amplitudes were sampled independently and uniformly. Stimulus presentation, inter-stimulus interval, maintenance, and retrieval lasted 8, 12, 24, and 12 time steps, respectively. We added independent Gaussian noise $\mathcal{N}(0, 0.05^2)$ to every input channel at each training time step and used noise-free inputs for evaluation. We implemented and trained all models in PyTorch (Paszke et al., 2019).

D.4 Comparison Model

We trained the stroke-reproduction model with mean-squared error on its 25-dimensional output trajectory. The comparison model used the same recurrent core and input distribution, but replaced the trajectory objective with sequence-level cross-entropy (CE) and added a linear classification head to the 25-dimensional RNN output.

For each minibatch, $\mathcal{T}_{\text{read}}$ ran from the mean retrieval onset across trials to the end of the padded trial. We averaged the 25-dimensional network output $\mathbf{y}(t)$ over this window:

$$\bar{\mathbf{y}} = \frac{1}{|\mathcal{T}_{\text{read}}|} \sum_{t \in \mathcal{T}_{\text{read}}} \mathbf{y}(t). \tag{43}$$

A linear classifier mapped this vector to category logits:

$$\mathbf{z} = \mathbf{W}_{\text{CE}} \bar{\mathbf{y}} + \mathbf{b}_{\text{CE}}. \tag{44}$$

Each category represented one ordered sequence of stroke directions; stroke amplitude was not part of the CE target. Let c_k denote the one-hot target for category k. We minimized cross-entropy:

$$\mathcal{L}_{\text{CE}} = -\sum_{k} c_k \log \frac{\exp(z_k)}{\sum_j \exp(z_j)}. \tag{45}$$

This comparison jointly changed the target representation, objective, and output decoder, so it tested a different representational organization rather than the effect of the loss alone.

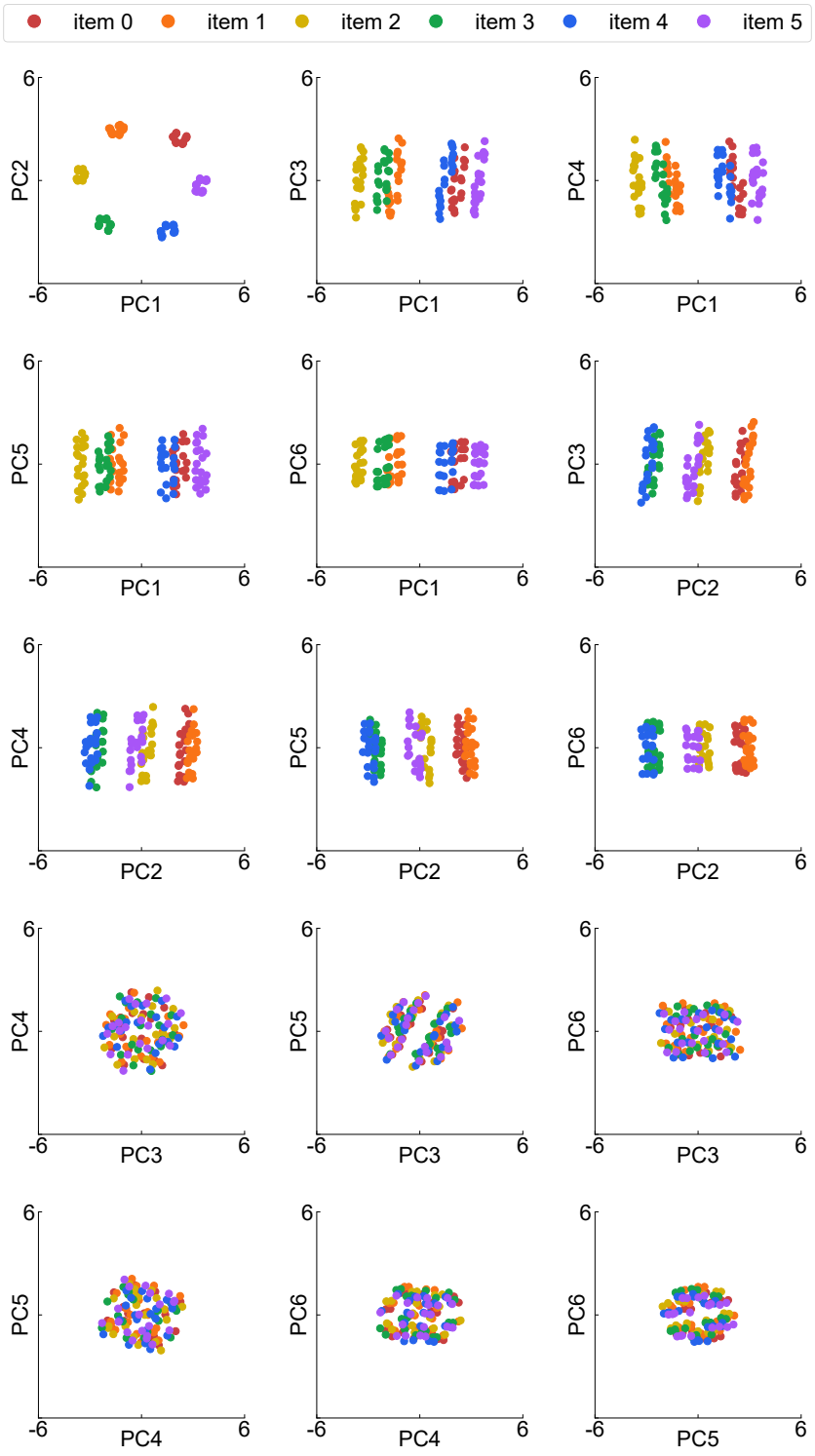

Figure S9: **PC-pair projections identify the rank-1 maintenance subspace.** Each panel shows projections onto one candidate PC pair from the first six principal components, with points labeled by rank-1 stimulus identity.

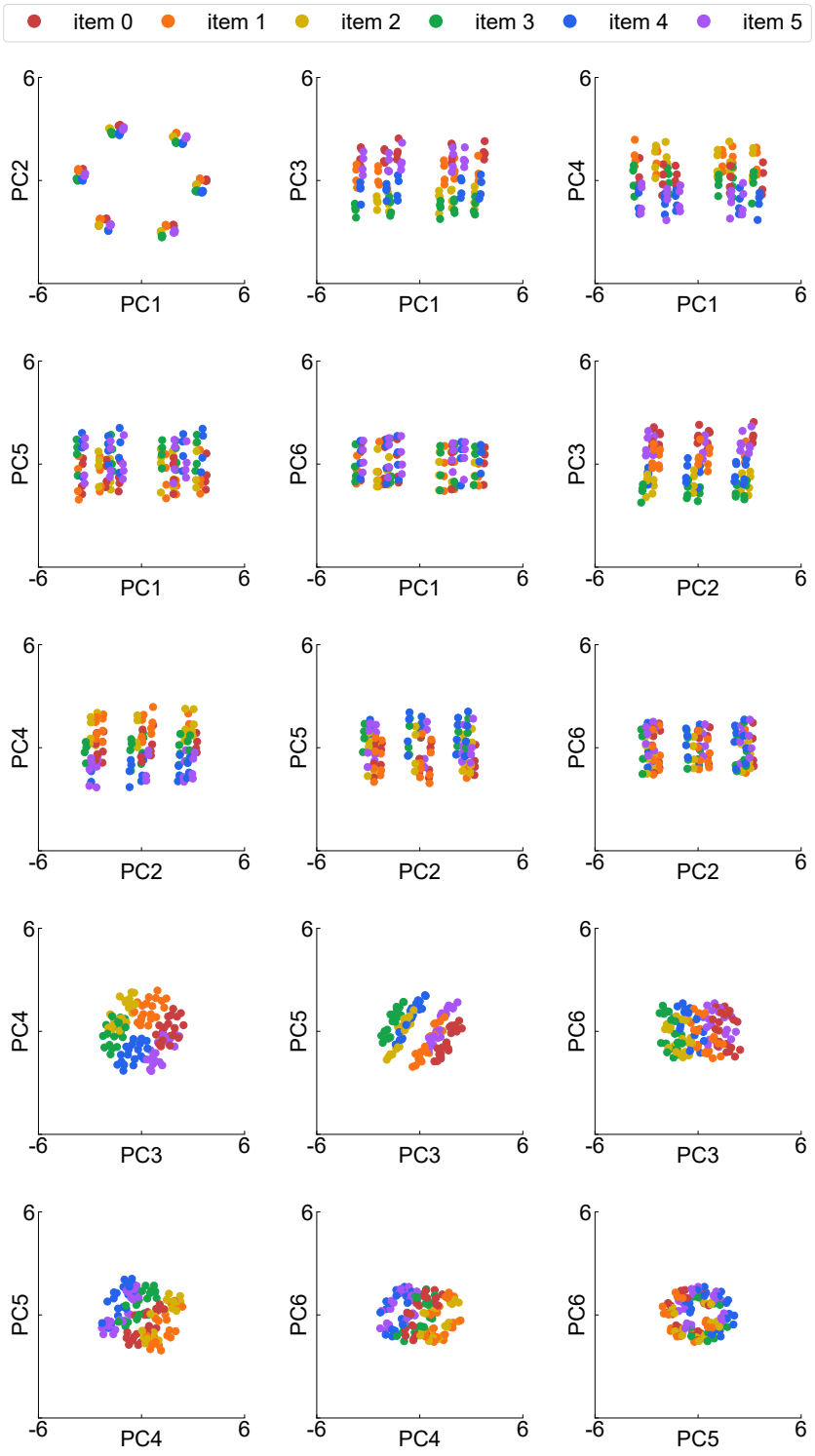

Figure S10: **PC-pair projections identify the rank-2 maintenance subspace.** Each panel shows projections onto one candidate PC pair from the first six principal components, with points labeled by rank-2 stimulus identity.

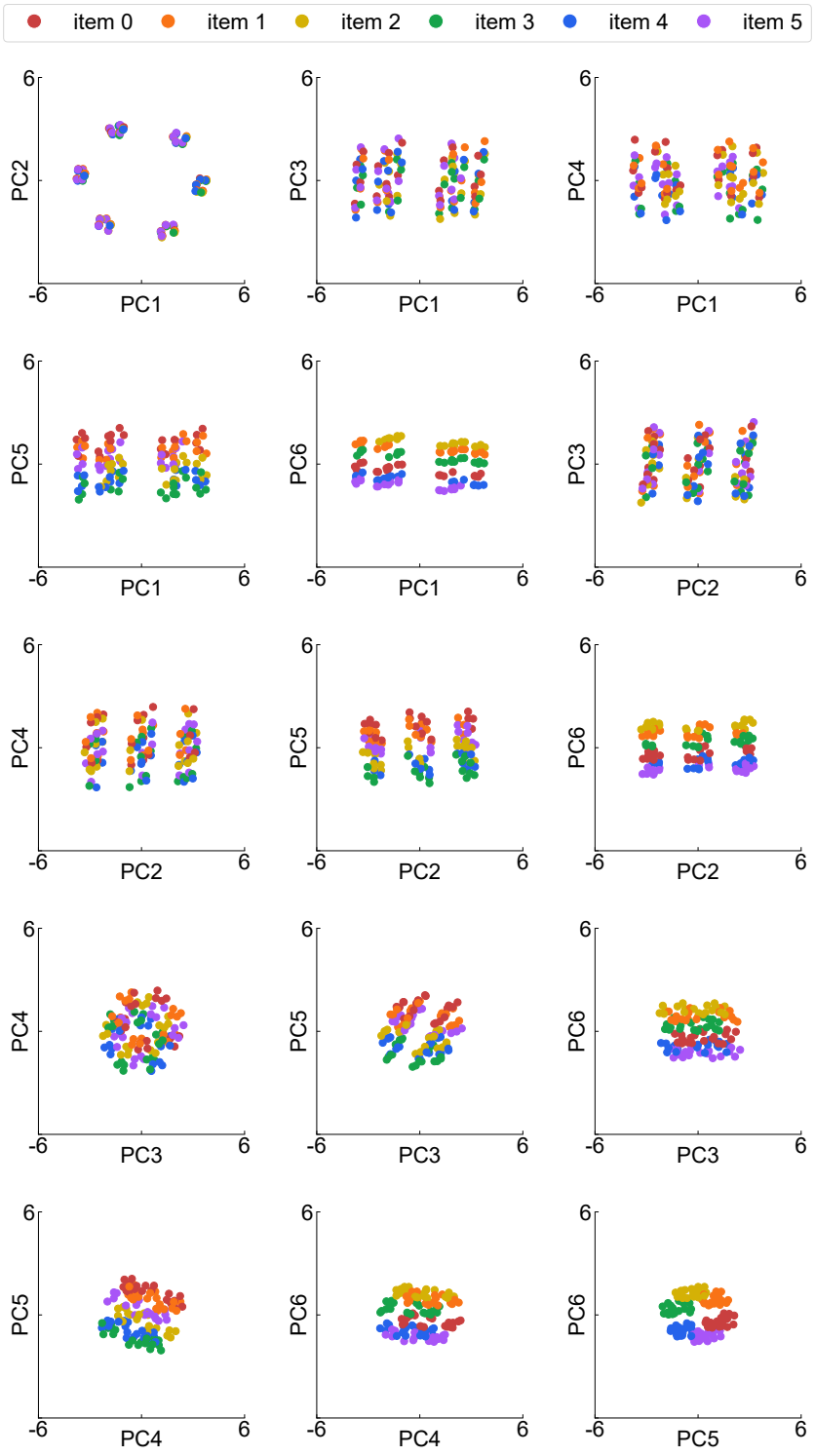

Figure S11: **PC-pair projections identify the rank-3 maintenance subspace.** Each panel shows projections onto one candidate PC pair from the first six principal components, with points labeled by rank-3 stimulus identity.

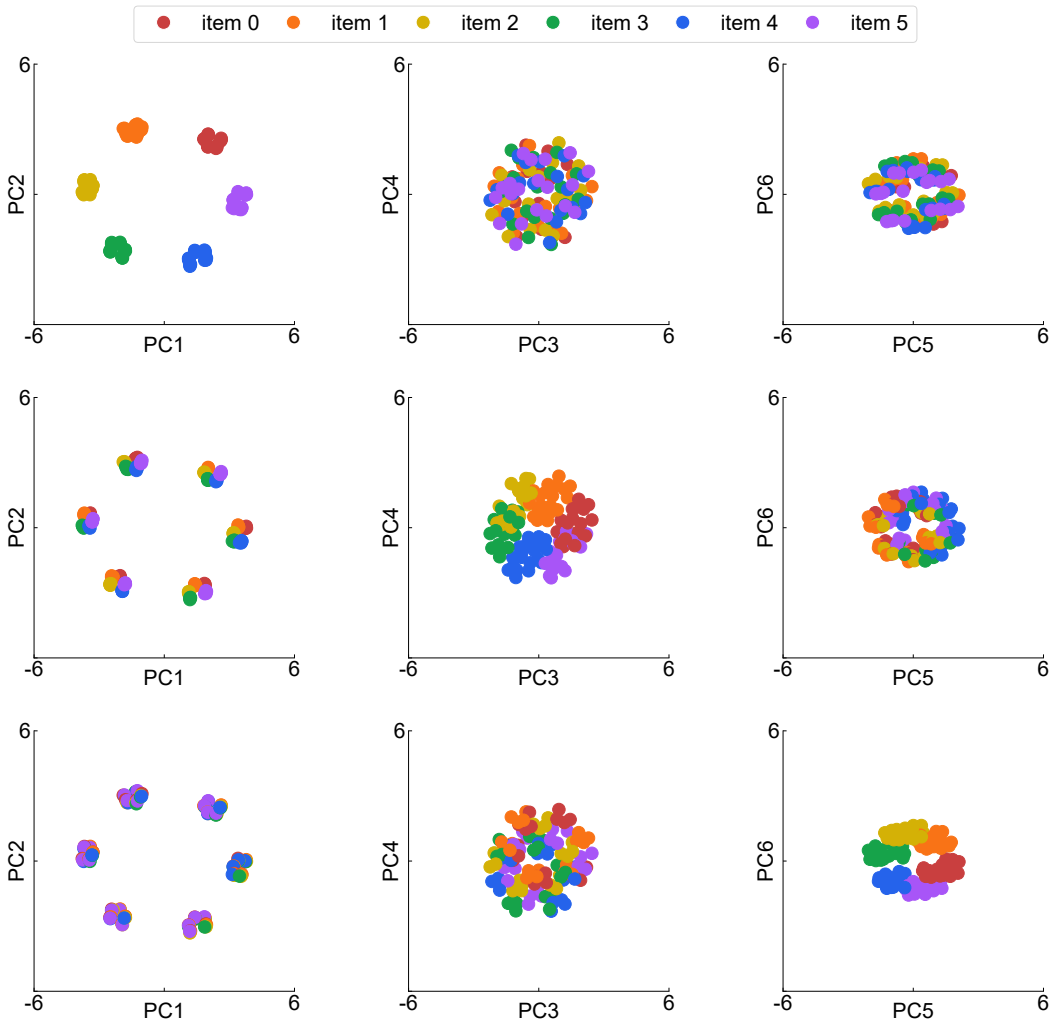

Figure S12: **Matched rank labels reveal the corresponding PCA-derived ring manifolds.** The first, second, and third rows are labeled using rank-1, rank-2, and rank-3 labels, respectively; columns correspond to the candidate subspaces identified by PCA.

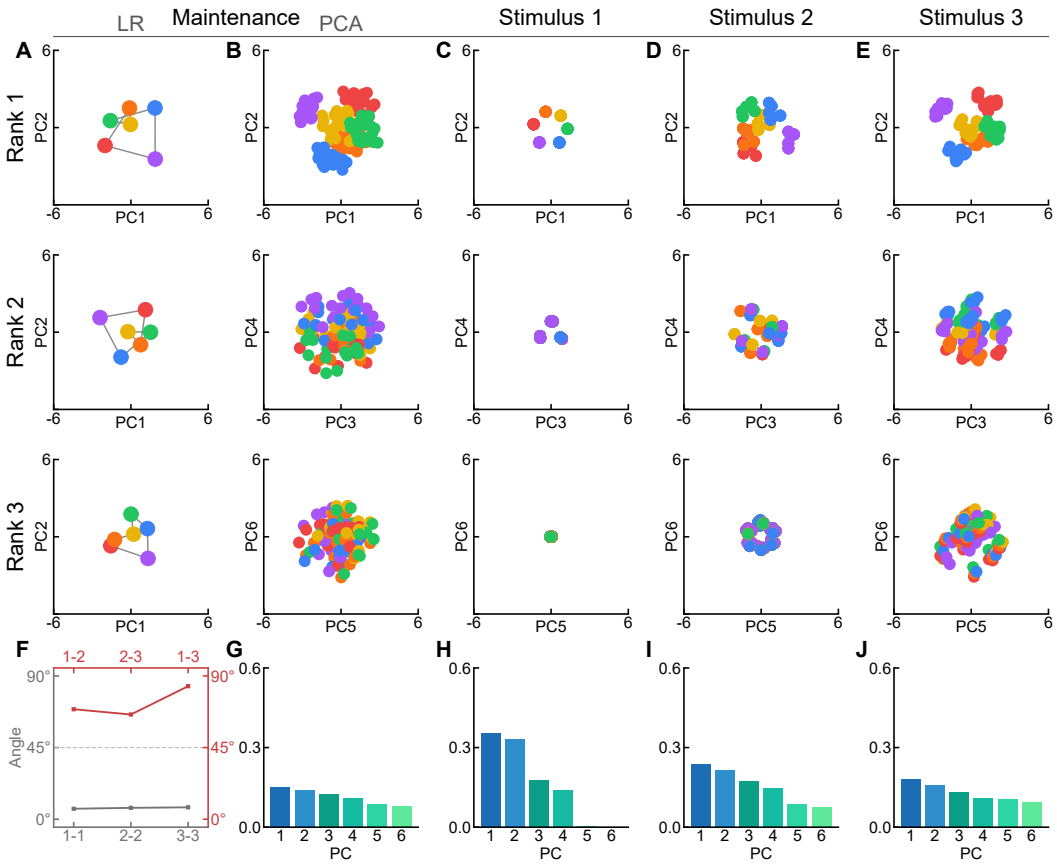

Figure S13: **Rank-subspace geometry under narrower Gaussian population-code inputs.** The same maintenance and encoding analyses as in Fig. 2 were applied to a model trained with a reduced Gaussian input width ($\sigma_s = 1$, compared to $\sigma_s = \sqrt{2}$ in the main model). Rank-specific structure remains visible, but the smooth ring-like organization is weaker than in the main model.

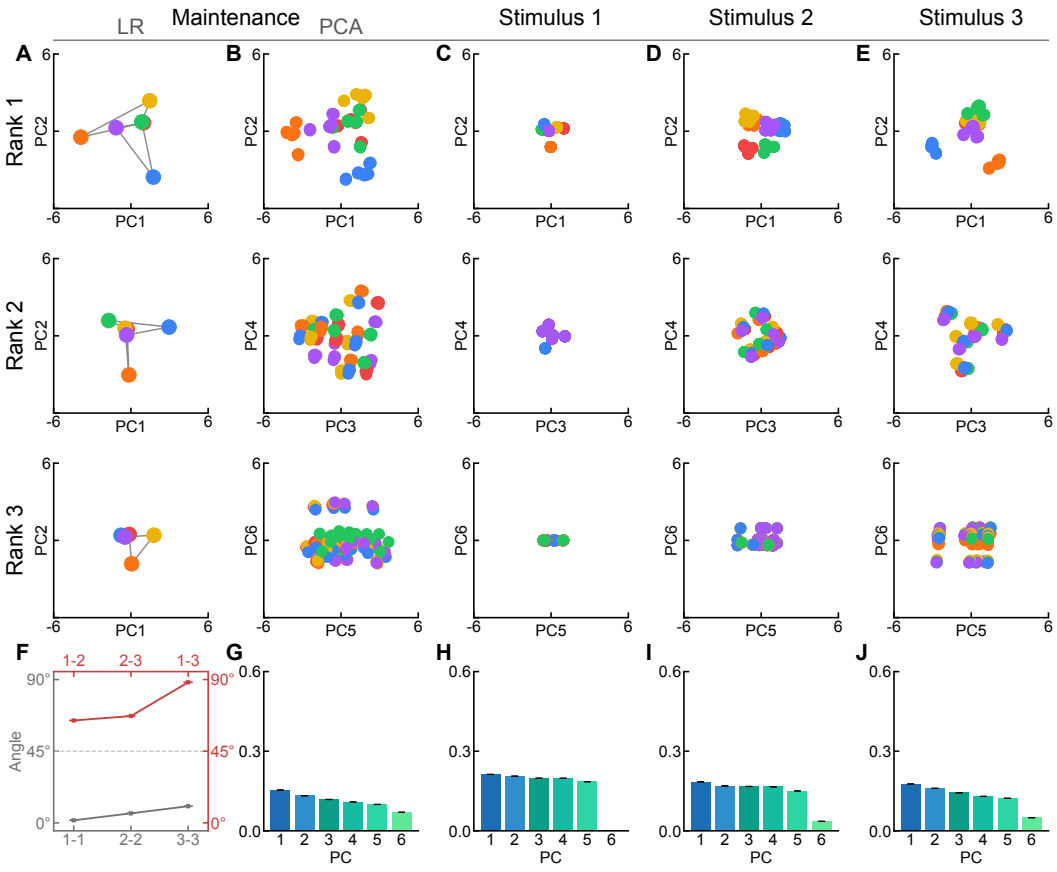

Figure S14: **Rank-subspace geometry under one-hot input and output coding.** The same maintenance and encoding analyses as in Fig. 2 were applied to a model trained with six-dimensional one-hot stimulus inputs and one-hot retrieval targets. Rank subspaces remain separated, whereas the item-level ring-like organization is weaker than under Gaussian population coding.

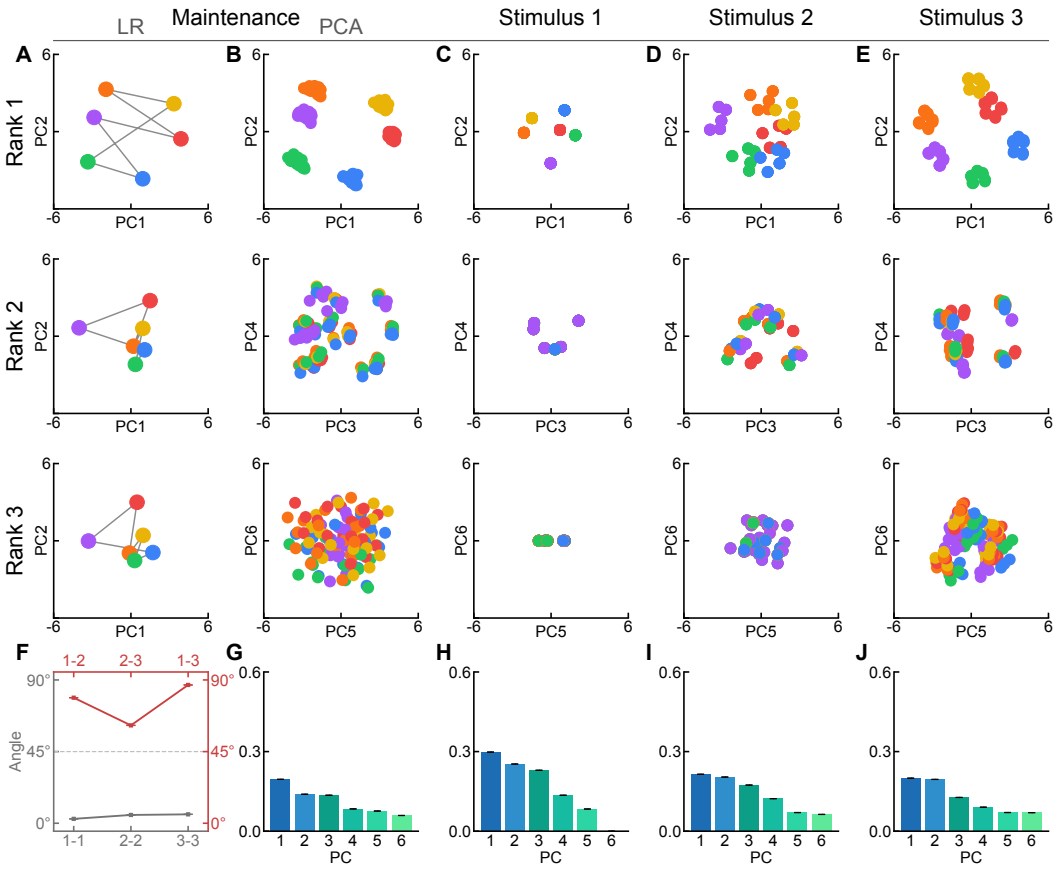

Figure S15: **Rank-subspace geometry under random-dense input and output coding.** The same maintenance and encoding analyses as in Fig. 2 were applied to a model trained with fixed random-dense stimulus inputs and random-dense retrieval targets. Rank subspaces remain separated despite the absence of a predefined circular ordering in either interface, while the item-level representations lack the semantic ring-like organization induced by Gaussian population coding.

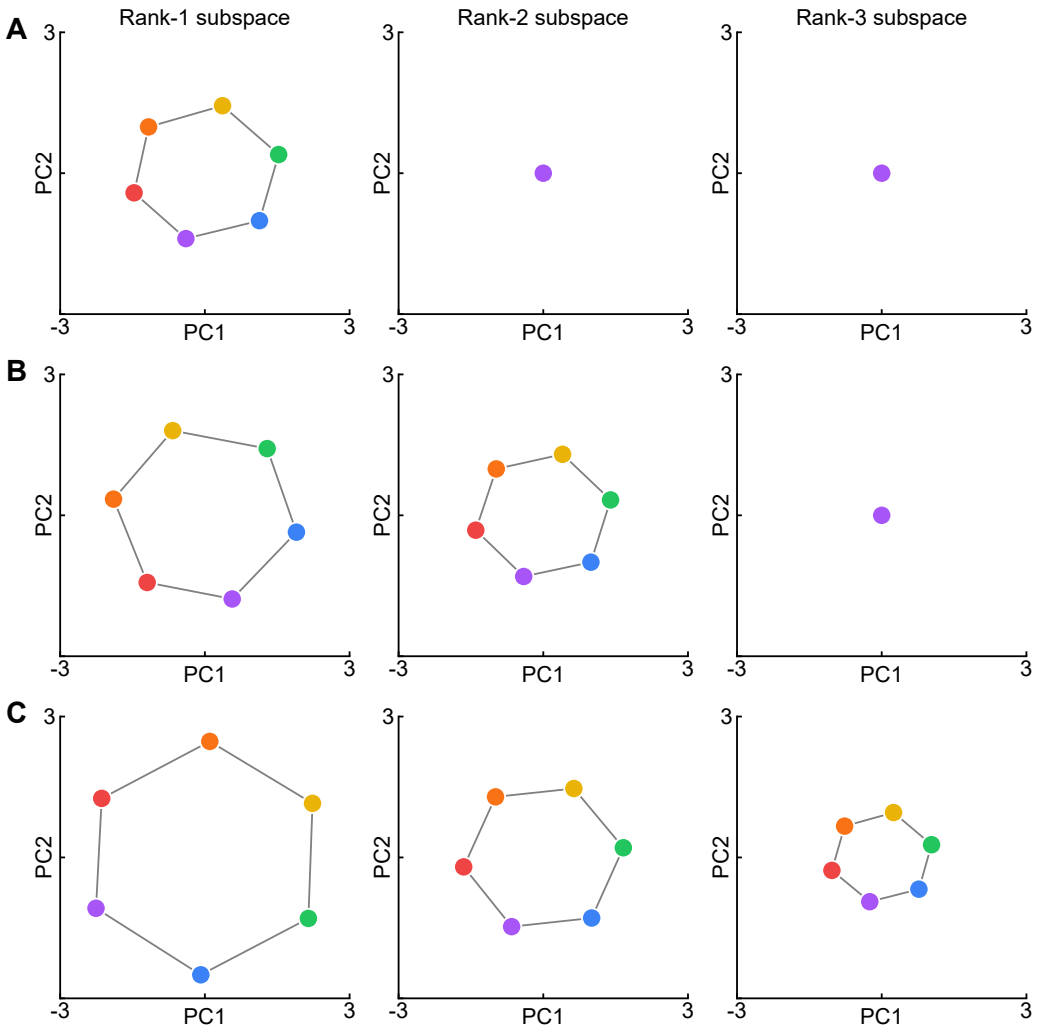

Figure S16: **Linear regression identifies rank-specific subspaces during encoding.** (A–C) Neural population activity projected onto linear-regression-extracted subspaces under rank-1, rank-2, and rank-3 stimulus conditions. As rank inputs proceed, information is progressively stored in distinct subspaces. The analysis method is described in Appendix B.2.

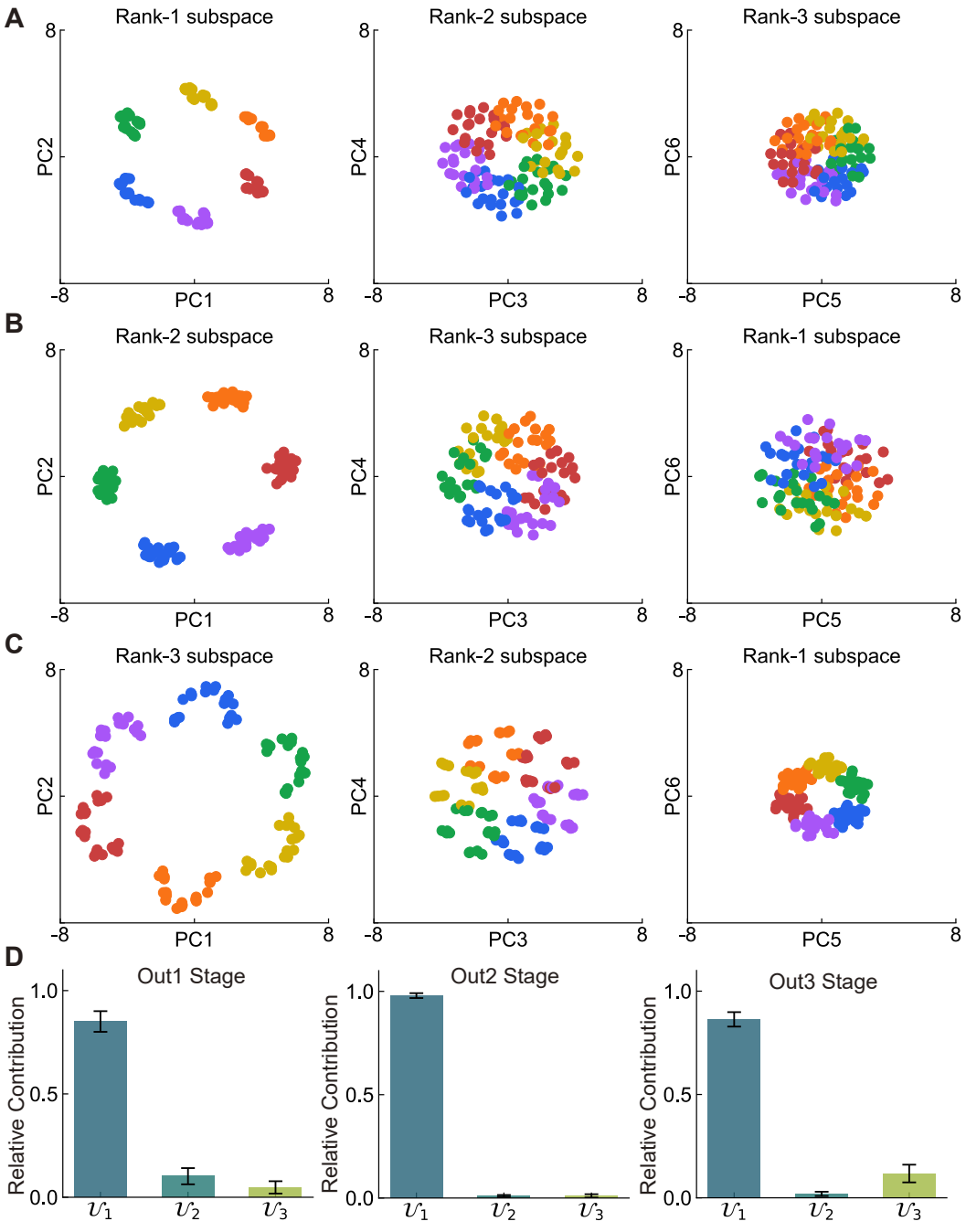

Figure S17: **Retrieval-stage population activity and output-weight contributions.** (A–C) Hidden-layer activity from one trained model, projected onto the first three PC-pair subspaces during the first, second, and third output periods, respectively. In each period, the currently retrieved rank is represented primarily in $\mathcal{U}_1$ (PC1–PC2). (D) Relative contributions of $\mathcal{U}_1$–$\mathcal{U}_3$ to the output-weight space defined by $\mathbf{W}_{\text{out}}$ during the three output periods. Bars show the mean ± SD across five independently trained models. See Appendix B.6 for method details.

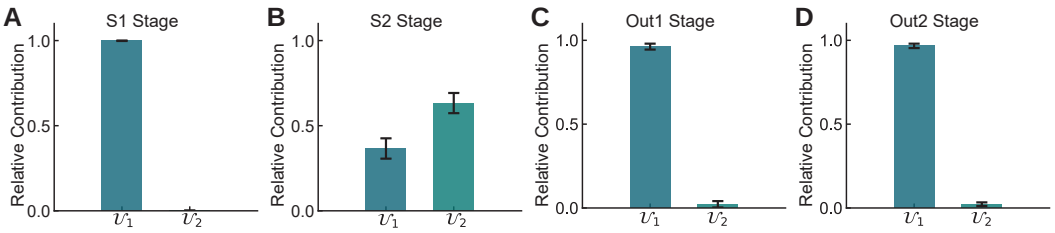

Figure S18: **Input and output weights show stage-dependent subspace contributions in the length-2 task.** (A–B) Relative contributions of the input weight matrix $\mathbf{W}_{\text{in}}$ to each subspace during the stimulus-presentation time windows t_{S_1} and t_{S_2} in the length-2 task. (C–D) Relative contributions of each subspace to the output weight matrix $\mathbf{W}_{\text{out}}$ across the retrieval phase in the length-2 task. (A–D) Results are averaged over 5 seeds.

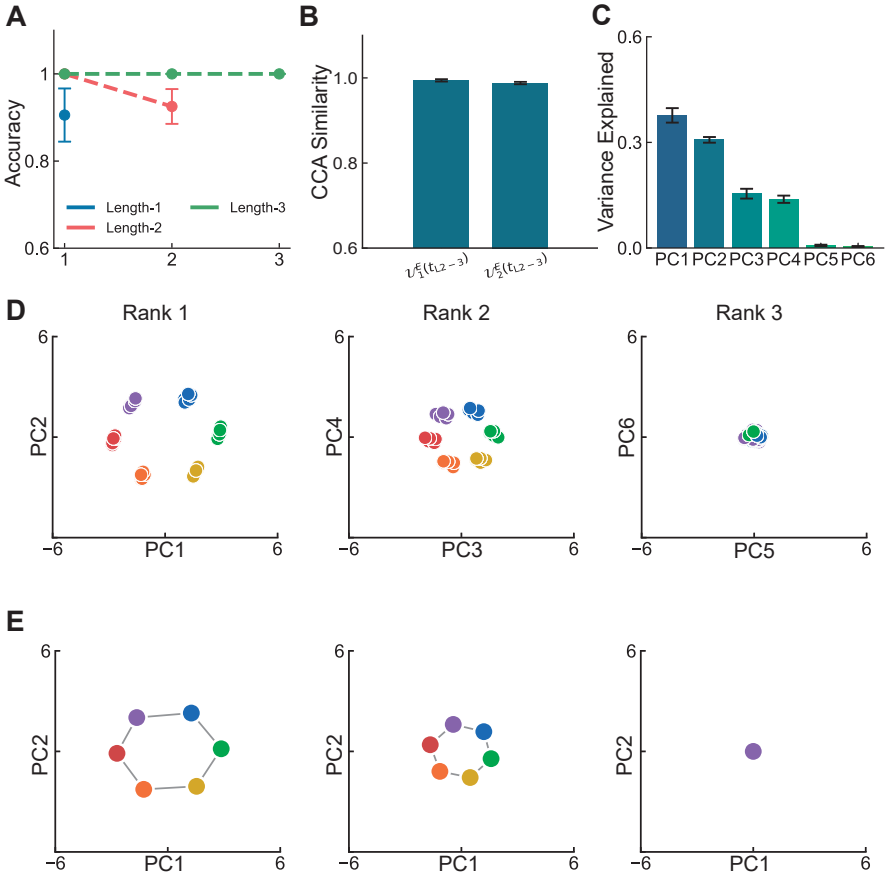

Figure S19: **Generalization performance and rank-subspace reuse across sequence lengths.** (A) Generalization performance of the model on tasks of different lengths. The blue curve corresponds to the length-1 task, the red curve corresponds to the length-2 task, and the green curve corresponds to the length-3 task. (B) CCA similarity between the corresponding encoding-stage rank subspaces in the length-2 and length-3 tasks, shown for ranks 1 and 2. (C) Explained variance ratios from PCA during the maintenance phase in the length-2 task. (D) Results obtained by directly applying PCA to neural activity during the encoding phase in the length-2 task. (E) Results obtained by applying linear regression to neural activity during the encoding phase in the length-2 task and projecting the weights using PCA. Error bars in (A–C) denote standard deviations over 5 seeds.

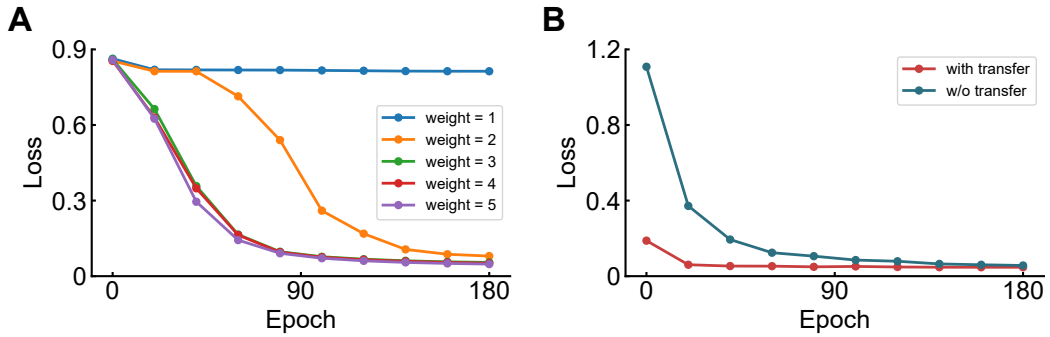

Figure S20: **Retrieval weighting and length-3 pretraining accelerate task learning.** (A) Retrieval-output weighting accelerates training; with weight 1, the model fails to learn the ordered recall task under the fixed training budget. Legend values denote the item-output mask coefficients used during training; all losses in (A) are computed with a fixed item-output mask coefficient of 5. (B) Length-3 pretraining lowers the initial loss and accelerates convergence when training on length-4 sequences compared with random initialization.

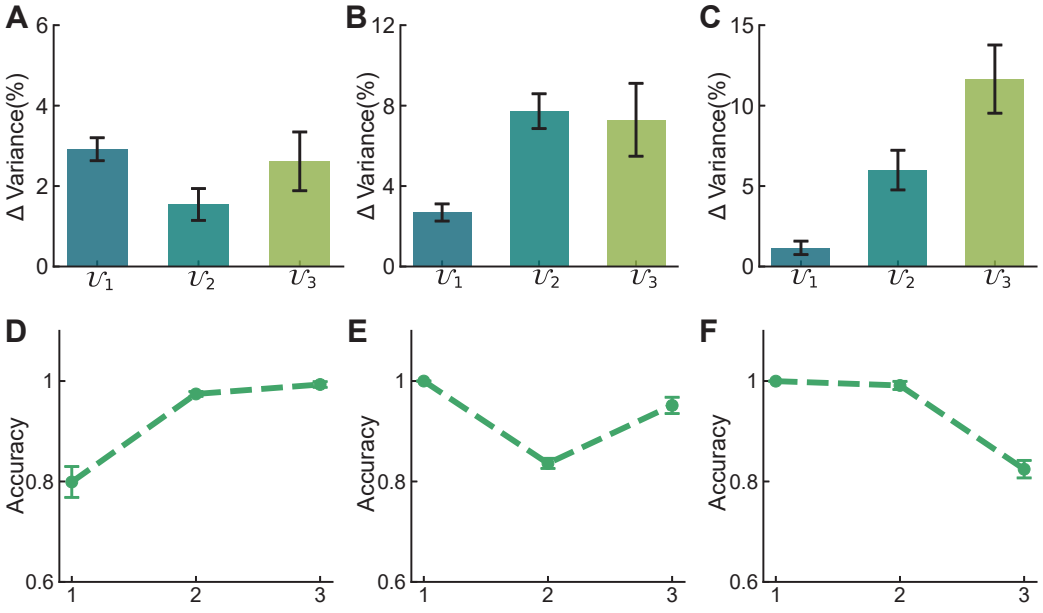

Figure S21: **Stimulus-specific effects of input noise on subspace activity and decoding accuracy.** (A–C) Specific effects of input noise on subspaces during the maintenance phase. Noise applied during the stimulus-presentation time windows t_{S_1}–t_{S_3} has the greatest impact on the corresponding subspaces $\mathcal{U}_1$–$\mathcal{U}_3$. (D–F) The largest decrease in decoding accuracy occurs at the rank corresponding to the perturbed stimulus, while accuracy at the other ranks is better preserved. (A–F) Results are averaged over 5 seeds. Method details are described in Appendix B.10.

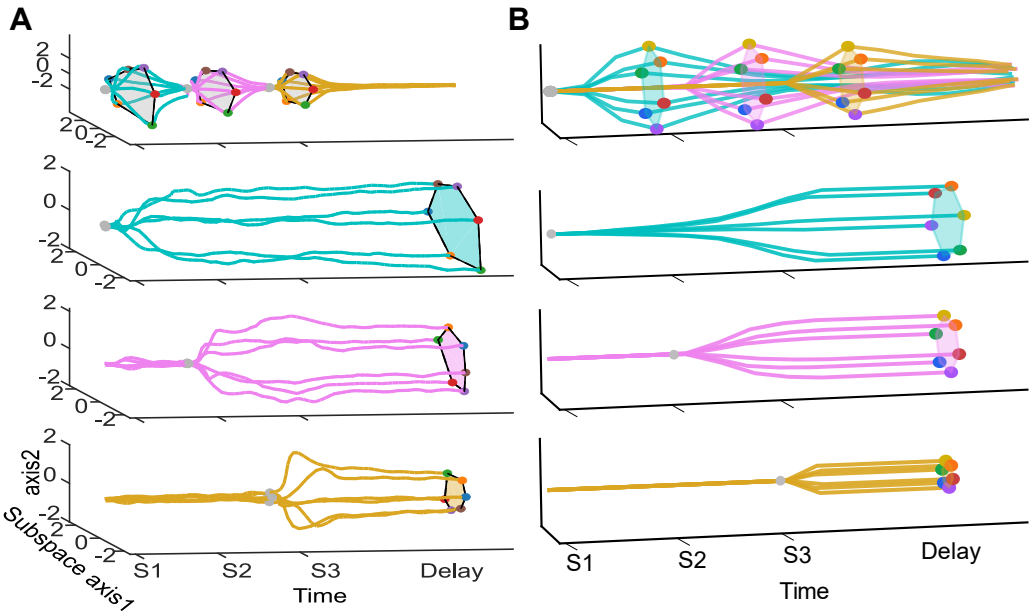

Figure S22: **Sensory-to-memory subspace transitions revealed by linear regression.** (A) Experimental neural trajectories projected onto sensory and memory subspaces. Adapted from Chen et al. (2024) Fig. 2C. (B) Model neural trajectories projected onto the fitted subspaces. The first row shows the shared sensory subspace, where stimulus-evoked ring-like structures appear transiently. The second, third, and fourth rows show Memory-1, Memory-2, and Memory-3 subspaces, respectively: rank-specific information emerges after the corresponding stimulus presentation and persists through later stimulus and maintenance periods. Gray dots indicate stimulus onset, illustrating transitions from the sensory subspace to rank-specific memory subspaces. See Appendix C.8 for details.