A TIMESCALE-ADAPTIVE RNN FOR RULE-GUIDED REORGANIZATION OF SEQUENCE WORKING MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequence working memory (SWM) is not merely a system for temporarily maintaining information, but an active computational workspace in which stored representations can be reorganized according to task demands. Recent recordings from primate frontal cortex revealed rank-content factorized representations and additional transient subspaces associated with sequence reordering, but the underlying circuit mechanisms remain unclear. Here, we show that a recurrent neural network with cue-triggered modulation of neuronal integration time constants can support flexible SWM manipulation while recapitulating key experimentally observed representational and dynamical signatures of SWM. After training on a rule-guided SWM manipulation task, the network developed a content-independent scaffold of ordinal-rank subspaces that served as reusable slots for different items. Before rule presentation, item-rank mappings were maintained; the forward rule preserved these mappings, whereas the backward rule reassigned items between rank slots. Item information was transiently routed into two additional subspaces during reordering, consistent with a buffering role. Comparisons across architectures and timescale parameters confirmed the importance of cue-triggered timescale modulation for learning this factorized memory organization. Three-item experiments further provided initial evidence that learned operations could be compositionally reused to support multi-step working-memory manipulation. These results suggest a concrete recurrent mechanism for flexible item–rank reorganization and provide mechanistic insight into how mental programming may operate through computations over factorized content and ordinal-rank representations.

1 INTRODUCTION

A hallmark of working memory is its ability not only to maintain information, but also to actively organize and manipulate stored representations in the service of behavior (Baddeley, 2003). This capacity provides a fundamental cognitive basis for flexible behaviors such as reasoning, planning, mental arithmetic, and sequence reordering (Miller & Cohen, 2001). Such active manipulation poses a central computational challenge: the memory system must balance stability with flexibility, preserving stored content while selectively reorganizing its structure according to current task demands. For example, reversing the order of a sequence in sequence working memory (SWM) requires the identities of the constituent items to remain intact while their assignments to ordinal positions are dynamically rearranged. Yet, much of previous work has focused on working-memory capacity and the neural mechanisms that support stable maintenance (Goldman-Rakic, 1995; Mongillo et al., 2008). How neural circuits flexibly reorganize already stored information according to newly specified task rules remains substantially less understood.

Recent recordings from the macaque frontal cortex have revealed a structured neural representation for sequence working memory (Chen et al., 2024). During sequence maintenance, Xie et al. (2022) showed that items at different ordinal positions are represented in distinct, approximately orthogonal rank subspaces, suggesting a factorized representation of item identity and ordinal position. Tian et al. (2024) further demonstrated that this organization can be dynamically reconfigured during memory manipulation. When monkeys reversed a remembered sequence, item representations were systematically reassigned across rank subspaces. Rather than transitioning directly from their original to target subspaces, the representations transiently occupied two additional subspaces, enabling items to be released from their original ordinal assignments and subsequently rebound to new

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.

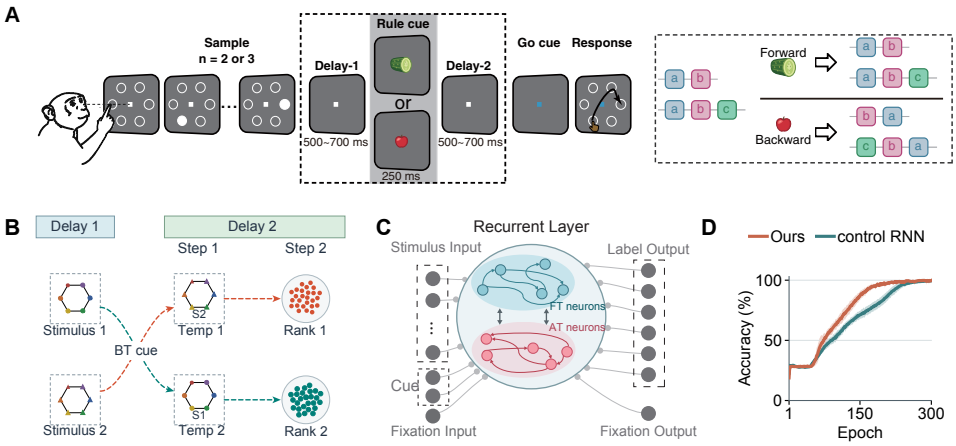

Figure 1: **Task paradigm and proposed circuit mechanism.** **(A)** Sequence working-memory task adapted from the mental-programming paradigm of (Tian et al., 2024). Two or three spatial stimuli are presented sequentially, followed by Delay-1, a rule cue, Delay-2, and serial report. The forward rule preserves the presentation order, whereas the backward rule reverses it. **(B)** Schematic of two-item reordering through temporary rank subspaces before reassignment to the opposite rank. Arrows indicate proposed information flow. **(C)** Network with stimulus, fixation, and rule inputs, 320 adaptive-timescale (AT) units, 80 fixed-timescale (FT) units, and a linear readout. Both populations receive rule information; only AT units undergo the prescribed post-cue reduction in time constant. **(D)** Training accuracy for the proposed model and a control leaky RNN whose units all have fixed time constants. Details are provided in Appendix A.3 and A.5.1.

positions. Tian et al. termed this coordinated transformation mental programming. Together, these findings suggest that active working-memory manipulation relies on rule-dependent reconfiguration of structured internal representations. However, how an abstract rule is translated into the circuit dynamics that coordinate such reconfiguration remains unknown.

Goal-driven recurrent neural networks provide a computational framework for understanding how structured representations can emerge in SWM (Botvinick & Plaut, 2006; Song et al., 2016). Recent models have shown that ordinal position can be represented in separable or reusable subspaces, supporting the stable maintenance and sequential retrieval of stored items (Wang et al., 2025; Dorrell et al., 2026). Whittington et al. (Whittington et al., 2025) further proposed a conceptual framework in which memory representations can be routed across activity subspaces to support flexible access through a common downstream readout. However, these studies primarily address the organization and retrieval of sequential memories, rather than the active manipulation of already stored contents. A key question therefore remains: how can a recurrent circuit, after receiving a rule cue, transition from stable maintenance to rapid reconfiguration and dynamically reassign stored items across ordinal-rank subspaces?

We hypothesized that modulation of neuronal timescales could provide such a dynamical switch. Neural timescales are heterogeneous across cortical neurons and are related to distinct computational roles, with shorter-timescale neurons supporting more dynamic encoding and longer-timescale neurons favoring stable maintenance (Murray et al., 2014; Wasmuht et al., 2018). Importantly, neural timescales can also vary with behavioral state and task demands (Zeraati et al., 2023; Manea et al., 2024). These observations motivated a model in which the rule cue signals a transition in behavioral state and triggers changes in neuronal timescales, enabling rapid, rule-guided working memory manipulation.

To test this mechanism, we trained an RNN with rule-cue-triggered timescale modulation. The network developed reusable ordinal-rank subspaces: the forward rule preserved item–rank mappings, whereas the backward rule reassigned items across ranks. During reordering, item information was transiently represented in two additional subspaces, resembling the experimental findings of Tian et al. (Tian et al., 2024). This organization extended to three-item permutations, with initial evidence

for sequential reuse of learned operations. Together, these results suggest that rule-cue-triggered timescale modulation supports computation through factorized rank subspaces, providing a candidate recurrent mechanism for mental programming in working memory.

2 METHOD

2.1 TASK STRUCTURE

To study the memory manipulation of SWM, we trained continuous-time recurrent neural networks on two post-cue sequence-reordering tasks adapted from the mental-programming paradigm of Tian et al. (Tian et al., 2024). On each trial, the network received a sequence $\{S_p\}_{p=1}^{L}$, where S_p denotes the stimulus presented at ordinal position p. Stimuli were sampled without replacement from $M = 6$ discrete spatial locations arranged on a circle. Each trial consisted of five successive epochs: sequence presentation, delay-1, rule-cue presentation, delay-2, and retrieval (Fig. 1A). During sequence presentation, the stimuli were presented sequentially with inter-stimulus intervals. The network then maintained the sequence during delay-1. A rule cue was presented afterward, specifying the required report order, followed by delay-2 and serial retrieval. Because the rule was unavailable during sequence presentation and delay-1, the same encoded sequence could require different output orders depending on the post-cue rule.

Two-item forward/backward SWM task We first considered two-item sequences, S_1, S_2. Following (Tian et al., 2024), the post-cue rule specified either forward or backward report. Under the forward-task rule, the sequence was reported in its original order, $S_1, S_2 \rightarrow S_1, S_2$, whereas under the backward-task rule, the report order was reversed, $S_1, S_2 \rightarrow S_2, S_1$. The two-item task provides the minimal setting in which sequence maintenance can be dissociated from post-cue reordering and was used for the primary analyses of network dynamics.

Three-item permutation SWM task We extended the task to three-item sequences and trained the network on four post-cue report orders: $[S_1, S_2, S_3]$, $[S_2, S_1, S_3]$, $[S_3, S_2, S_1]$, and $[S_1, S_3, S_2]$. As in the two-item task, the permutation rule was presented only after delay-1, requiring the network to maintain the encoded sequence before reorganizing it according to the post-cue rule. Examples of the input and output time courses are shown in Fig. S1, with additional task details provided in Appendix A.1.

2.2 MODEL ARCHITECTURE AND TRAINING PROTOCOL

The model consisted of an input layer, a recurrent layer, and a linear readout layer (Fig. 1B). The input layer encoded stimulus, rule-cue, and fixation signals. All input signals, including the rule cue, were provided to the recurrent network. The recurrent layer contained two interacting neuronal populations: **fixed-timescale neurons**, whose time constants remained unchanged throughout the trial, and **adaptive-timescale neurons**, whose effective time constants decreased following rule-cue onset. Thus, both populations had access to the rule information, whereas cue-dependent modulation of temporal dynamics was restricted to the adaptive-timescale population.

Let $\mathbf{x}(t)$ denote the recurrent synaptic-current state and $\mathbf{r}(t) = \tanh[\mathbf{x}(t)]$ the corresponding firing-rate vector. The recurrent dynamics were given by

$$\boldsymbol{\tau}_{\text{eff}}(t) \odot \frac{d\mathbf{x}(t)}{dt} = -\mathbf{x}(t) + \mathbf{W}_{\text{rec}}\mathbf{r}(t) + \mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{b}, \tag{1}$$

where $\mathbf{u}(t)$ contains the stimulus, rule-cue, and fixation inputs, $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{rec}}$ denote the input and recurrent weights, and $\mathbf{b}$ is the recurrent bias. The network output was obtained through a linear readout,

$$\mathbf{y}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t). \tag{2}$$

To introduce cue-dependent changes in temporal dynamics, we defined a prescribed modulation variable $s(t)$ that depended on rule-cue onset, not on rule identity. Let t_{cue} denote cue onset and

T_{ramp} the duration over which the timescale modulation was introduced. We used

$$s(t) = \begin{cases} 0, & t < t_{\mathrm{cue}}, \\ \dfrac{t - t_{\mathrm{cue}}}{T_{\mathrm{ramp}}}, & t_{\mathrm{cue}} \le t < t_{\mathrm{cue}} + T_{\mathrm{ramp}}, \\ 1, & t \ge t_{\mathrm{cue}} + T_{\mathrm{ramp}}. \end{cases} \tag{3}$$

The effective time constant of neuron i was then defined as

$$\tau_{\mathrm{eff},i}(t) = \frac{\tau}{1 + \gamma m_i s(t)}, \tag{4}$$

where τ is the baseline integration time constant, γ controls modulation strength, and $m_i = 1$ for adaptive-timescale neurons and $m_i = 0$ for fixed-timescale neurons. AT time constants decrease gradually to $\tau/(1+\gamma)$ over T_{ramp}. This design was motivated by experimental observations that cortical neurons exhibit heterogeneous intrinsic timescales and that neural timescales can vary with cognitive state (Murray et al., 2014; Wasmuht et al., 2018; Zeraati et al., 2023).

The recurrent layer contained 320 adaptive-timescale neurons and 80 fixed-timescale neurons, randomly assigned before training. Before the rule cue onset, $s(t) = 0$, such that all neurons shared the baseline time constant τ. Following cue onset, $s(t)$ increased from 0 to 1 over the prescribed ramp interval and then remained at 1 throughout delay-2 and retrieval. Consequently, adaptive-timescale neurons gradually transitioned to the shorter effective time constant $\tau/(1+\gamma)$, whereas fixed-timescale neurons retained τ throughout the trial. Population assignments and this schedule were fixed during training and testing.

The output layer contained $M = 6$ stimulus channels and one fixation channel. Before retrieval, the fixation target was active and all stimulus targets were inactive. At retrieval onset, the fixation target switched off, and each retrieval epoch required a one-hot representation of the corresponding stimulus for t_{retr} time steps, following the order specified by the post-cue rule. Examples of target and network output trajectories are shown in Fig. S1B.

On each training trial, we sampled stimulus sequences, task rules, and epoch durations, independently varying inter-stimulus intervals and delays and adding Gaussian input noise. We optimized $\{\mathbf{W}_{\mathrm{in}}, \mathbf{W}_{\mathrm{rec}}, \mathbf{W}_{\mathrm{out}}, \mathbf{b}\}$ by backpropagation through time to minimize the retrieval-period mean-squared error:

$$\mathcal{L} = \frac{1}{|\mathcal{T}_{\mathrm{ret}}|} \sum_{t \in \mathcal{T}_{\mathrm{ret}}} \|\mathbf{y}(t) - \mathbf{y}^*(t)\|_2^2,$$

where $\mathcal{T}_{\mathrm{ret}}$ contains the retrieval periods and $\mathbf{y}^*(t)$ is the target output. Optimization details are provided in Appendix A.3.

3 Results

3.1 Adaptive-Timescale RNN Recapitulates Rank-Specific Population Dynamics

To investigate whether adaptive neuronal timescales shape the dynamics underlying sequence manipulation, we first focused on the two-item SWM task, which provides the minimal setting for dissociating sequence maintenance from post-cue reordering. Both models learned the sequence-manipulation task, with the adaptive RNN exhibiting modestly faster learning (Fig. 1D). We next asked whether the two models learned distinct population-level representational solutions and how these compared with experiment.

We first examined population activity during delay-1, before presentation of the rule cue. Following the experimental analysis (Tian et al., 2024), we projected neural population activity into low-dimensional state space and characterized representations associated with the first and second ordinal ranks. The adaptive RNN developed distinct rank-specific subspaces, within which different stimulus identities occupied structured locations in population space (Fig. 2A-C). A corresponding analysis of the leaky-RNN control also revealed rank-related organization, indicating that both models could form ordinal representations during memory maintenance. We therefore next asked whether these representations were differentially reorganized during post-cue manipulation.

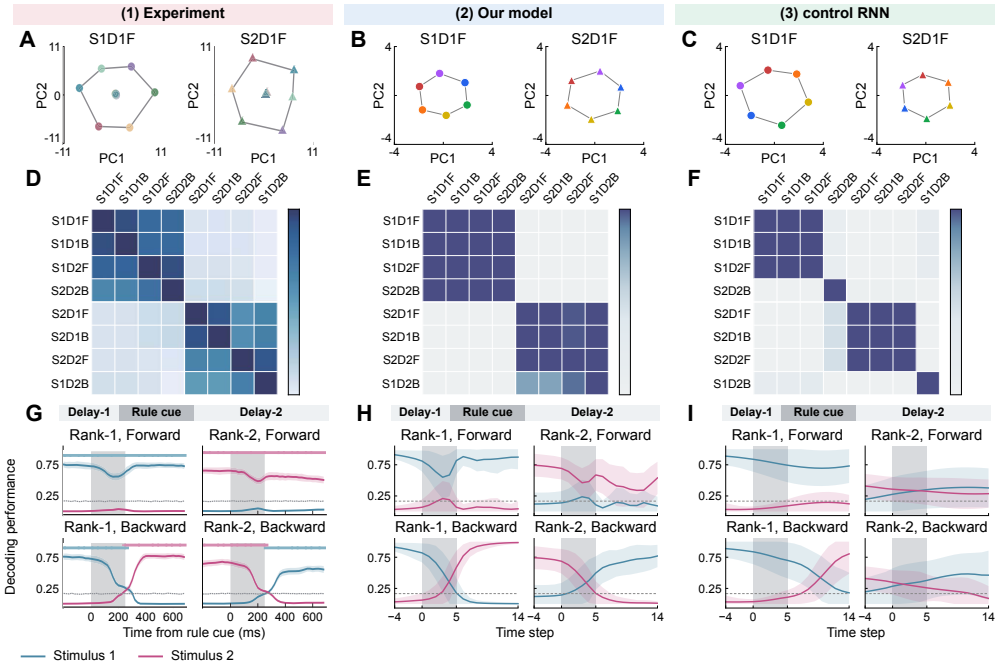

Figure 2: **Comparison of rank-specific dynamics between macaque cortex and RNNs.** Experimental results adapted from (Tian et al., 2024) (left column), the adaptive-timescale RNN (middle), and a control leaky RNN (right). **(A-C)** Low-dimensional representations of the first and second presented stimuli during delay 1 of forward trials. Colors distinguish spatial stimulus locations; axes denote the first two principal components (PC1 and PC2) of the displayed projections. **(D-F)** Cross-condition representational matrices spanning stimulus, delay, and rule conditions. S1/S2 denotes the first/second presented stimulus, D1/D2 denotes delay 1/delay 2, and F/B denotes the forward/backward rule. Conditions are ordered to group the representations expected to occupy the same ordinal rank following maintenance or reversal. **(G-I)** Time-resolved stimulus decoding in the R1 and R2 representations for forward trials (upper row) and backward trials (lower row). Blue and magenta curves represent stimulus 1 and stimulus 2, respectively; gray vertical bands indicate the rule-cue period. Analysis details are provided in Appendix B.1.

We quantified how these representations were reorganized across stimulus identity, delay period, and rule condition. Using the same condition definitions as in the experiment (e.g., S1D1F denotes stimulus 1 during delay-1 in a forward trial), we compared the geometry of population representations across conditions and summarized their pairwise similarity in a cross-condition matrix (Fig. 2D-F). In both the experimental data and the adaptive RNN, representations were organized largely according to the ordinal rank occupied by an item. Under the forward rule, stimulus–rank assignments were preserved across delays, whereas under the backward rule, stimulus 2 in delay-2 became aligned with the Rank-1 (R1) representation and stimulus 1 with Rank-2 (R2). Thus, the rank-related representational structure was largely maintained while the item associated with each rank was reassigned. This organization was less clearly expressed in the leaky-RNN control, particularly for post-cue backward conditions.

Finally, we examined the temporal evolution of item information within each rank representation. Following the experimental decoding analysis, we trained rank-specific decoders and evaluated stimulus identity continuously across the rule-cue period (Fig. 2G-I). During forward trials, stimulus decoding within each rank remained relatively stable. During backward trials, the experimental data showed complementary changes across the two ranks: stimulus 1 information decreased while stimulus 2 information increased in R1, with the opposite pattern in R2. The adaptive RNN showed a similar cue-triggered transition, whereas the leaky-RNN control exhibited weaker and less clearly coordinated changes across ranks. Together, these analyses suggest that the adaptive RNN captures

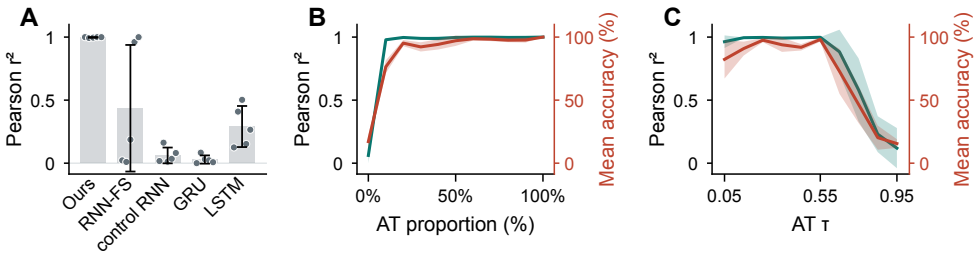

Figure 3: **Architecture comparison and timescale parameter analysis. (A)** Agreement between each model's cross-condition representational matrix and the ideal rank-swapping template, plotted on an ordinate labelled Pearson r^2. The displayed models are Ours, RNN-FS, RNN, GRU, and LSTM. **(B)** Template agreement (teal) and mean decoding accuracy (red) as functions of the proportion of adaptive-timescale (AT) units. **(C)** The corresponding measures as functions of the post-cue AT timescale parameter, labelled AT τ. The two curves use separate scales for template agreement and mean accuracy (percentage). Details are provided in Appendix A.5 and A.4.

both the rank-specific organization of working-memory representations and their rule-dependent reassignment observed experimentally.

3.2 Rule-dependent Timescale Adaptation Supports Rank-based Representational Organization

To investigate whether cue-triggered timescale modulation was required for the emergence of factorized rank-content memory organization, we compared the adaptive RNN with several control architectures, including RNN-FS, in which fast- and slow-timescale populations were fixed throughout the trial, as well as a leaky RNN, GRU, and LSTM (Cho et al., 2014; Hochreiter & Schmidhuber, 1997) (Fig. 3A). For each model, we quantified the similarity between its cross-condition representational matrix and an ideal rank-swapping template using Pearson r^2. An r^2 approaching 1 indicates that the network adopted a factorized rank–content organization, in which ordinal-rank subspaces are preserved across conditions while item contents are reassigned between them during reordering. Although the control architectures could learn the task, they did not reliably converge to this representational solution. In particular, RNN-FS showed substantial variability across runs, occasionally approaching the rank-swapping template but often failing to do so, indicating that static fast–slow timescale heterogeneity alone is insufficient. Likewise, GRU and LSTM networks, despite having gating mechanisms capable of learning flexible effective timescales, showed substantially weaker correspondence to the ideal rank-based geometry. Thus, successful performance or generic temporal flexibility alone does not ensure the emergence of the experimentally observed factorized organization.

We further examined how this organization depended on the two defining parameters of the adaptive mechanism: the fraction of adaptive-timescale neurons and their post-cue timescale. Rank-template agreement remained high over a broad parameter range, but collapsed when the adaptive population was too small (Fig. 3B) or when its post-cue timescale remained too long (Fig. 3C), together with a deterioration in task performance. Together, the architecture and parameter comparisons suggest that cue-triggered timescale modulation is critical for reliably learning the factorized rank–content organization underlying flexible memory reordering, whereas fixed timescale heterogeneity and the gated architectures tested here did not reliably produce the same organization.

3.3 Adaptive RNN Develops Temporary Rank-WM Subspaces and a Shared Readout Population

We next examined how memory transformation was implemented within the recurrent population of the adaptive RNN. Successful manipulation requires that item information remain available while its association with ordinal rank is being changed, and that the reorganized representation ultimately becomes accessible to the fixed downstream readout.

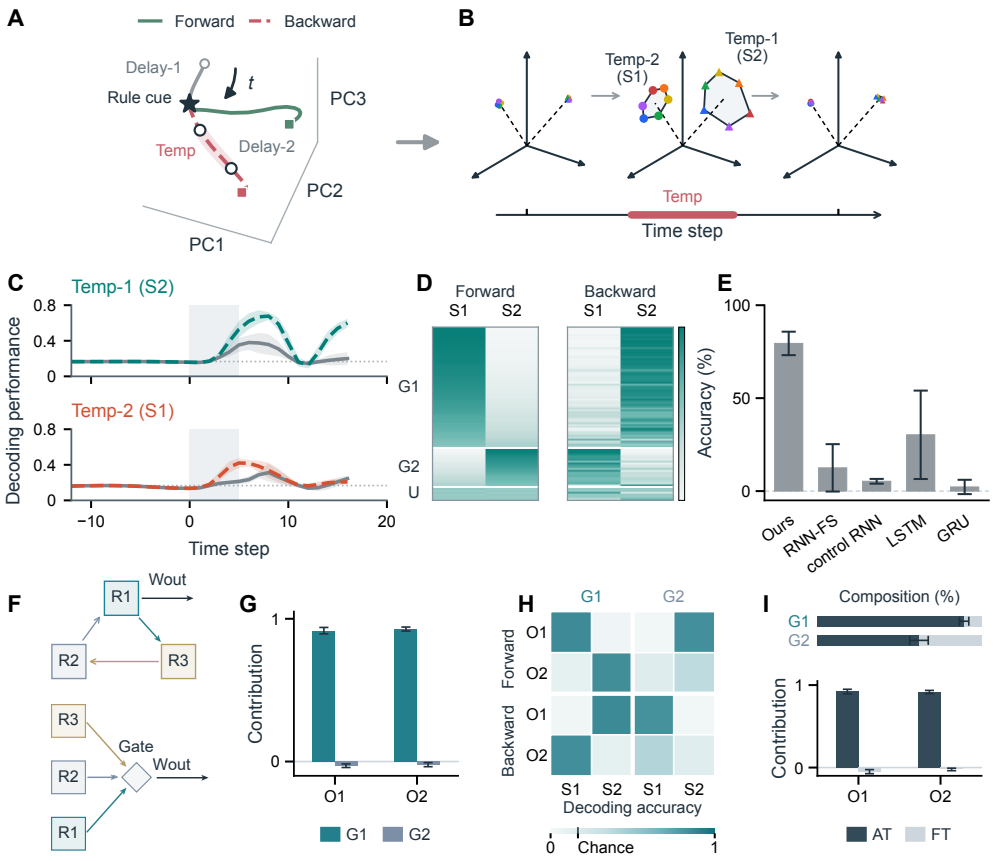

Figure 4: **Temporary memory representations and population organization during manipulation and retrieval.** **(A)** Population trajectory schematic across the rule cue: forward (solid green), backward (dashed red), and a transient interval (Temp). **(B)** Stimulus representations in Temp-2 (S1) and Temp-1 (S2) during this interval. **(C)** Time-resolved decoding in Temp-1 (upper) and Temp-2 (lower); gray bands mark the cue. **(D)** Stimulus-representation heatmaps under both rules, grouped by late delay-2 rank preference: G1 (R1), G2 (R2), and U (unselective). **(E)** Manipulation accuracy across five models, measured as the percentage of trials correctly reporting an untrained target sequence after modifying contents encoded in G1 or G2. **(F)** Alternative fixed-readout schemes: routing through R1 (upper) or gating rank-specific inputs (lower). **(G–H)** G1/G2 contributions and stimulus decoding during output epochs O1/O2; decoding is shown for both rules. **(I)** AT/FT composition of G1/G2 (upper) and contributions during O1/O2 (lower). AT, adaptive-timescale; FT, fixed-timescale. See Appendices B.2 and B.3 for details.

Following Tian et al. (2024), we examined whether temporary working-memory subspaces emerged during post-cue manipulation. We identified two additional subspaces, Temp-1 and Temp-2, associated with the intermediate stage of representational reorganization (Fig. 4A-B). These subspaces carried little item information during late delay-1 but became informative following rule-cue onset, with pronounced decoding peaks during backward trials: Temp-1 represented S_2, whereas Temp-2 represented S_1 (Fig. 4C). These dynamics are consistent with temporary retention of item information while its association with ordinal rank is being reorganized. By late delay-2, population analysis identified two rank-preferring groups, G1 and G2 (Fig. 4D). Under the forward rule, G1 and G2 preferentially represented S_1 and S_2, respectively; under the backward rule, these contents switched. Thus, the transient representations were followed by an updated rank-based organization in which population preference followed ordinal rank while item content changed with the rule.

To test whether rank and content could be flexibly recombined, we manipulated the contents represented by these populations to construct sequences absent from training, including repeated-item sequences such as $[2, 2]$ (Appendix B.3.2). The adaptive RNN read out these sequences with higher

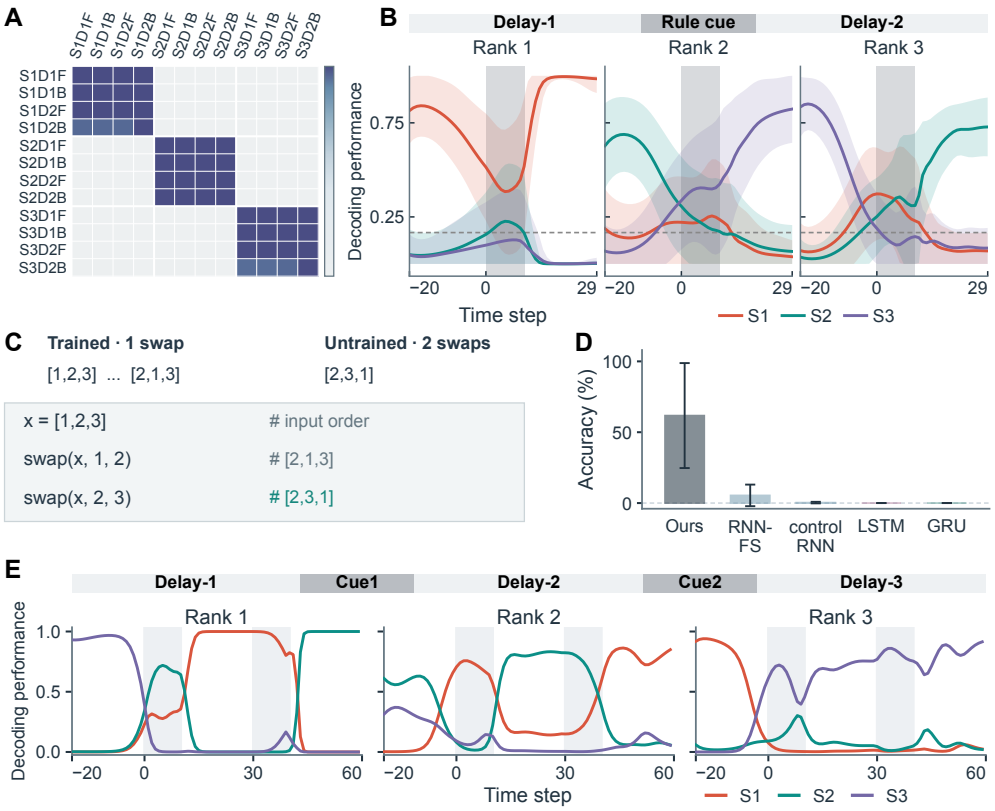

Figure 5: **Three-item memory representations and compositional manipulation.** **(A)** Representational matrix for forward/backward rules; S1–S3 denote presentation order, D1/D2 delay, and F/B rule. **(B)** Rank-specific decoding for 123 → 132. Blue/magenta/teal denote S1/S2/S3; gray marks the cue. **(C)** Unseen two-cue compositional working-memory manipulation: swaps of positions 1 and 2, then 2 and 3, yield 123 → 213 → 231. **(D)** Accuracy in the two-cue evaluation. **(E)** A two-cue decoding example. Analysis details are provided in Appendix B.4.

accuracy than the comparison models (Fig. 4E). Together, these results support a factorized rank-content organization, with rank-specific populations flexibly accommodating item contents and temporary subspaces serving as routing intermediates during item–rank reorganization.

We next examined how these rank-related populations contributed to retrieval. Sequential retrieval could in principle rely either on routing rank-specific contents into a shared readout-aligned subspace or on gating different rank populations directly to the output (Fig. 4F). Despite the rule-dependent change in the contents represented by G1 and G2, retrieval was consistently dominated by G1 across both report positions and rules (Fig. 4G-H), supporting the first scheme: the content to be reported is routed into a shared G1 readout population for access by the fixed output weights W_{out}. Additionally, G1 was enriched in adaptive-timescale neurons, which were preferentially recruited during retrieval (Fig. 4I). Together, these results suggest a separation between rank-based memory organization and readout, with flexible item-rank associations but sequential retrieval through a shared G1-dominated readout population.

3.4 Rank-subspace Reconfiguration Supports Compositional Three-item Manipulation

We next asked whether the rank-subspace reconfiguration observed in the two-item task extends to more complex manipulation. In the three-item permutation task, the network was trained on four post-cue rules specifying different permutations of the same three-item sequence. Population geometry formed three clusters corresponding to ordinal rank, despite changes in the item occupying

each rank across rules (Fig. 5A). For visualization, Fig. 5A shows the forward and backward rules, with the remaining trained permutations provided in Appendix C.5. Time-resolved decoding further revealed coordinated, rule-dependent reassignment of item information within rank representations (Fig. 5B). Thus, the factorized organization identified in the two-item task, with stable rank-specific subspaces and flexibly reassigned contents, also extended to three-item manipulations.

We then tested whether learned permutation operations could be composed to generate transformations not explicitly trained as a whole (Fig. 5C). After the first manipulation stabilized, a second rule cue was applied to the updated memory state, for example $123 \rightarrow 132 \rightarrow 312$. The adaptive RNN outperformed control recurrent architectures on these two-cue manipulations (Fig. 5D), while time-resolved decoding showed successive reassignment of item contents across rank-specific subspaces after each cue (Fig. 5E). These results provide initial evidence that factorizing ordinal rank from item content may support compositional working-memory manipulation, allowing learned operations to be reapplied and combined into more complex transformations.

4 Discussion

Our results suggest a computational account of rule-guided SWM reorganization. The timescale-adaptive RNN captured the rank-related geometry and transient item representations observed in the two-item experiments of (Tian et al., 2024). Architecture comparisons and parameter sweeps further highlighted the importance of rule-cue-triggered timescale modulation for learning this organization. Because the experimental study was limited to two-item manipulation, we extended the model to three-item tasks to generate testable predictions. The model predicts three reusable rank subspaces whose contents are reassigned according to task rules, together with sequential updating of current item–rank mappings under successive cues, allowing learned operations to be reused in new combinations. Together, these findings suggest that timescale modulation supports factorized rank-subspace computation and provides a candidate recurrent mechanism for mental programming in SWM.

Related Works Computational models of sequence working memory have shown how item identity and ordinal position can be factorized and flexibly bound within structured memory representations, accounting for storage, maintenance, retrieval, and the emergence of rank-specific geometry (Whittington et al., 2025; Wang et al., 2025; Dorrell et al., 2026; Botvinick & Plaut, 2006; Botvinick & Watanabe, 2007). More recent models explicitly implement working-memory manipulation through continuous-attractor memory modules and gain-based control, including swaps and more complex permutations (Zuo et al., 2025; Bien et al., 2026). These studies demonstrate how memory manipulation can be realized in explicitly structured architectures, but leave open whether the experimentally observed reorganization of rank-specific representations can emerge through learning in a recurrent network, and what dynamical mechanism enables the transition from stable maintenance to flexible manipulation.

Our mechanistic hypothesis is instead motivated by the flexibility of neural timescales. Cortical neurons exhibit heterogeneous timescales associated with distinct computational roles, with shorter timescales favoring dynamic encoding and longer timescales supporting stable maintenance (Murray et al., 2014; Wasmuht et al., 2018), and these timescales can vary with behavioral state and task demands (Zeraati et al., 2023; Manea et al., 2024). Related recurrent-network studies have shown that contextual inputs and gating can regulate effective dynamical timescales (Wang et al., 2018; Heinrich et al., 2020; Hasani et al., 2021; Krishnamurthy et al., 2022). Motivated by these observations, we ask whether cue-triggered timescale modulation can serve as a dynamical switch that enables a task-trained recurrent network to preserve stable rank-content structure during maintenance while transiently reconfiguring item-rank assignments during manipulation.

Limitations Our model prescribes the adaptive neuronal population and cue-triggered timescale schedule, isolating the computational role of timescale modulation while leaving its circuit-level implementation open. Future work should examine whether such modulation can emerge through learned control signals or recurrent interactions (Heinrich et al., 2020; Hasani et al., 2021; Li et al., 2021), and whether the resulting computations generalize beyond the present permutation tasks to longer sequences, more complex operation compositions, and task-dependent transformations (Li et al., 2025; Bien et al., 2026).

AI USE STATEMENT

In this work, we used generative AI tools to assist with the creation and refinement of scientific figures, including improving the presentation, layout, and visual clarity of schematic illustrations. We also used generative AI tools to aid and polish parts of the manuscript writing, such as improving grammar, wording, and readability.

We did not use generative AI tools to generate or analyze experimental results, interpret findings, derive theoretical or mathematical claims, or implement the core methods reported in this paper. All AI-assisted figures and text were carefully reviewed and revised by the authors to ensure accuracy, originality, and consistency with the scientific content of the paper. We take full responsibility for the final content of this work, including text and figures produced or edited with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The appendix documents how the experiments and analyses were conducted. Appendix A describes the tasks, models, and training procedure; Appendix B explains the analyses; and Appendix C presents additional results.

REFERENCES

Alan Baddeley. Working memory: looking back and looking forward. *Nature Reviews Neuroscience*, 4:829–839, 2003.

Kevin Bien, Junfeng Zuo, and Wenhao Zhang. On neural circuits of working memory sequence permutation: optimizing circuit architectures via cayley graphs. In *Proceedings of the 4th (2025) and 3rd (2024) NeurIPS Workshops on Symmetry and Geometry in Neural Representations*, volume 282 of *Proceedings of Machine Learning Research*, pp. 29–48. PMLR, 2026.

Matthew M. Botvinick and David C. Plaut. Short-term memory for serial order: a recurrent neural network model. *Psychological review*, 113(2):201–233, 2006. doi: 10.1037/0033-295X.113.2.201.

Matthew M. Botvinick and Takamitsu Watanabe. From numerosity to ordinal rank: A gain-field model of serial order representation in cortical working memory. *The Journal of Neuroscience*, 27:8636 – 8642, 2007.

Jingwen Chen, Cong Zhang, Peiyao Hu, Bin Min, and Li-Ping Wang. Flexible control of sequence working memory in the macaque frontal cortex. *Neuron*, 112(20):3502–3514.e6, 2024. doi: 10.1016/j.neuron.2024.07.024.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1179.

William Dorrell, Peter E. Latham, Timothy E. J. Behrens, and James C. R. Whittington. An efficient computing theory of prefrontal structured working memory representations. *bioRxiv*, pp. 2026.02.16.706126, 2026. doi: 10.64898/2026.02.16.706126.

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

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7657–7666, 2021. doi: 10.1609/aaai.v35i9.16936.

Stefan Heinrich, Tayfun Alpay, and Yukie Nagai. Learning timescales in gated and adaptive continuous time recurrent neural networks. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2662–2667, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Kamesh Krishnamurthy, Tankut Can, and David J. Schwab. Theory of gating in recurrent neural networks. *Physical review X*, 12(1):011011, 2022. doi: 10.1103/PhysRevX.12.011011.

Siwei Li, Jingwen Chen, Cong Zhang, Shiming Tang, Yang Xie, and Li-Ping Wang. Flexible use of limited resources for sequence working memory in macaque prefrontal cortex. *Nature Communications*, 16:10386, 2025. doi: 10.1038/s41467-025-65380-0.

Yinghao Li, Robert Kim, and Terrence J. Sejnowski. Learning the synaptic and intrinsic membrane dynamics underlying working memory in spiking neural network models. *Neural Computation*, 33(12):3264–3287, 2021. doi: 10.1162/neco_a_01409.

Ana M. G. Manea, David J.-N. Maisson, Benjamin Voloh, Anna Zilverstand, Benjamin Yost Hayden, and Jan Zimmermann. Neural timescales reflect behavioral demands in freely moving rhesus macaques. *Nature Communications*, 15:2151, 2024. doi: 10.1038/s41467-024-46488-1.

Earl K. Miller and Jonathan D. Cohen. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24:167–202, 2001.

Gianluigi Mongillo, Omri Barak, and Misha Tsodyks. Synaptic theory of working memory. *Science*, 319:1543 – 1546, 2008.

John D. Murray, Alberto Bernacchia, David J. Freedman, Ranulfo Romo, Jonathan D. Wallis, Xinying Cai, Camillo Padoa-Schioppa, Tatiana Pasternak, Hyojung Seo, Daeyeol Lee, and Xiao-Jing Wang. A hierarchy of intrinsic timescales across primate cortex. *Nature neuroscience*, 17:1661 – 1663, 2014.

Emilia P Piwek, Mark G. Stokes, and Christopher Summerfield. A recurrent neural network model of prefrontal brain activity during a working memory task. *PLOS Computational Biology*, 19(10): e1011555, 2023. doi: 10.1371/journal.pcbi.1011555.

H. Francis Song, Guangyu R. Yang, and Xiao-Jing Wang. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLOS Computational Biology*, 12(2):e1004792, 2016. doi: 10.1371/journal.pcbi.1004792.

Zhenghe Tian, Jingwen Chen, Cong Zhang, Bin Min, Bo Xu, and Li-Ping Wang. Mental programming of spatial sequences in working memory in the macaque frontal cortex. *Science*, 385(6716): eadp6091, 2024. doi: 10.1126/science.adp6091.

Jing Wang, Devika Narain, Eghbal A. Hosseini, and Mehrdad Jazayeri. Flexible timing by temporal scaling of cortical responses. *Nature neuroscience*, 21(1):102–110, 2018. doi: 10.1038/s41593-017-0028-6.

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

D. F. Wasmuht, E. Spaak, T. J. Buschman, E. K. Miller, and M. G. Stokes. Intrinsic neuronal dynamics predict distinct functional roles during working memory. *Nature Communications*, 9: 3499, 2018. doi: 10.1038/s41467-018-05961-4.

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.e6, 2025. doi: 10.1016/j.neuron.2024.10.017.

Yang Xie, Peiyao Hu, Jun-Ru Li, Jingwen Chen, Weibin Song, Xiao-Jing Wang, Tianming Yang, Stanislas Dehaene, Shiming Tang, Bin Min, and Liping Wang. Geometry of sequence working memory in macaque prefrontal cortex. *Science*, 375:632 – 639, 2022.

Roxana Zeraati, Yan-Liang Shi, Nicholas A. Steinmetz, Marc A. Gieselmann, Alexander Thiele, Tirin Moore, Anna Levina, and Tatiana A. Engel. Intrinsic timescales in the visual cortex change with selective attention and reflect spatial connectivity. *Nature Communications*, 14:1858, 2023. doi: 10.1038/s41467-023-37613-7.

Jun-Feng Zuo, Cheng Xue, Si Wu, and Wen-Hao Zhang. Towards a mental programming neural circuit: Insights from working memory sequence manipulation. *bioRxiv*, pp. 2025.07.24.666222, 2025. doi: 10.1101/2025.07.24.666222.

A Model and Training Details

A.1 Task timing and input–output signals

Each trial followed the five task epochs defined in the main text: sequence presentation, delay-1, rule-cue presentation, delay-2, and serial retrieval. Each stimulus was presented at constant amplitude for eight time steps. During training, each blank interval between successive stimuli was sampled uniformly from 8–16 time steps, and delay-1 and delay-2 were sampled independently from 8–32 time steps. For fixed-timing evaluation, each inter-stimulus interval lasted 12 time steps and both delays lasted 20 time steps; stimulus duration remained eight time steps.

The task input $\mathbf{u}(t)$ contained 24 spatial-stimulus channels, rule-cue channels, and one fixation channel. The two-rule models used two cue channels, giving $N_{\text{in}} = 27$, whereas the four-rule models used four cue channels, giving $N_{\text{in}} = 29$. The stimulus channels had preferred angles $\theta_n = 2\pi n/24$, $n = 0, \ldots, 23$, and the six stimulus locations were centered at $\phi_m = m\pi/3$, $m = 0, \ldots, 5$. Let d denote the shortest angular distance on the circle. The stimulus input was zero outside stimulus-presentation periods; when stimulus m was presented,

$$[\mathbf{u}_{\text{stim}}(t)]_n = A_s \exp\left[-\frac{d(\theta_n, \phi_m)^2}{2\sigma_s^2}\right], \qquad A_s = 4, \quad \sigma_s = \sqrt{2}. \tag{5}$$

Each rule cue was presented through a separate input channel at unit amplitude for five time steps in two-item trials and ten time steps in three-item trials, with an equal-duration timescale ramp in the adaptive RNNs. The fixation input remained active through delay-2 and switched off at retrieval. During training, independent Gaussian noise with standard deviation 0.01 was added to all input channels at each time step.

The output vector contained six stimulus channels and one fixation channel. Before retrieval, the fixation target was active and all stimulus targets were zero. During each 12-step retrieval epoch, the target stimulus channel was active and the remaining stimulus channels were zero. Fixation offset signaled the transition to retrieval. Stimulus identity was read from the mean stimulus output over the final ten time steps of each retrieval epoch. Complete-sequence accuracy required every output position to be correct.

A.2 Network architectures and timescale modulation

Adaptive-timescale RNN. All adaptive RNNs had $N_{\text{rec}} = 400$ recurrent units and a seven-channel linear readout ($N_{\text{out}} = 7$). Recurrent activity used a tanh nonlinearity. The recurrent layer had 320 AT and 80 FT units. Both rule cues were connected to all recurrent units in the two-rule model; in the four-rule model, all four cues were connected only to the AT population. We integrated Eqs. 1–4 with the forward Euler method using $\Delta t = 0.05$, $\tau = 1$, and $\gamma = 3$.

During the ramp interval, the modulation variable took evenly spaced values from $s = 0$ on the first time step to $s = 1$ on the final time step. The AT effective time constant therefore changed from 1 to 0.25 and remained at 0.25 through delay-2 and retrieval, whereas FT neurons remained at $\tau = 1$.

RNN-FS. RNN-FS tested whether heterogeneity across fixed-timescale neurons produces the rank-based organization analyzed in the main text. It used Eq. 1 and fixed the adaptive RNN's post-ramp time-constant configuration throughout the trial:

$$\tau_i^{\text{FS}} = \begin{cases} 0.25, & \text{for 320 recurrent units,} \\ 1, & \text{for 80 recurrent units.} \end{cases} \tag{6}$$

With $\Delta t = 0.05$, the corresponding Euler update coefficients were 0.20 for the 320-unit population and 0.05 for the 80-unit population.

Architecture baselines. The RNN, GRU, and LSTM baselines used 400 recurrent units and the same seven-channel linear readout as the adaptive RNN. The RNN followed Eq. 1 with $\tau_{\text{eff},i}(t) = 1$ for every neuron throughout the trial, whereas the GRU and LSTM used standard discrete updates. Every rule cue was connected to all recurrent units.

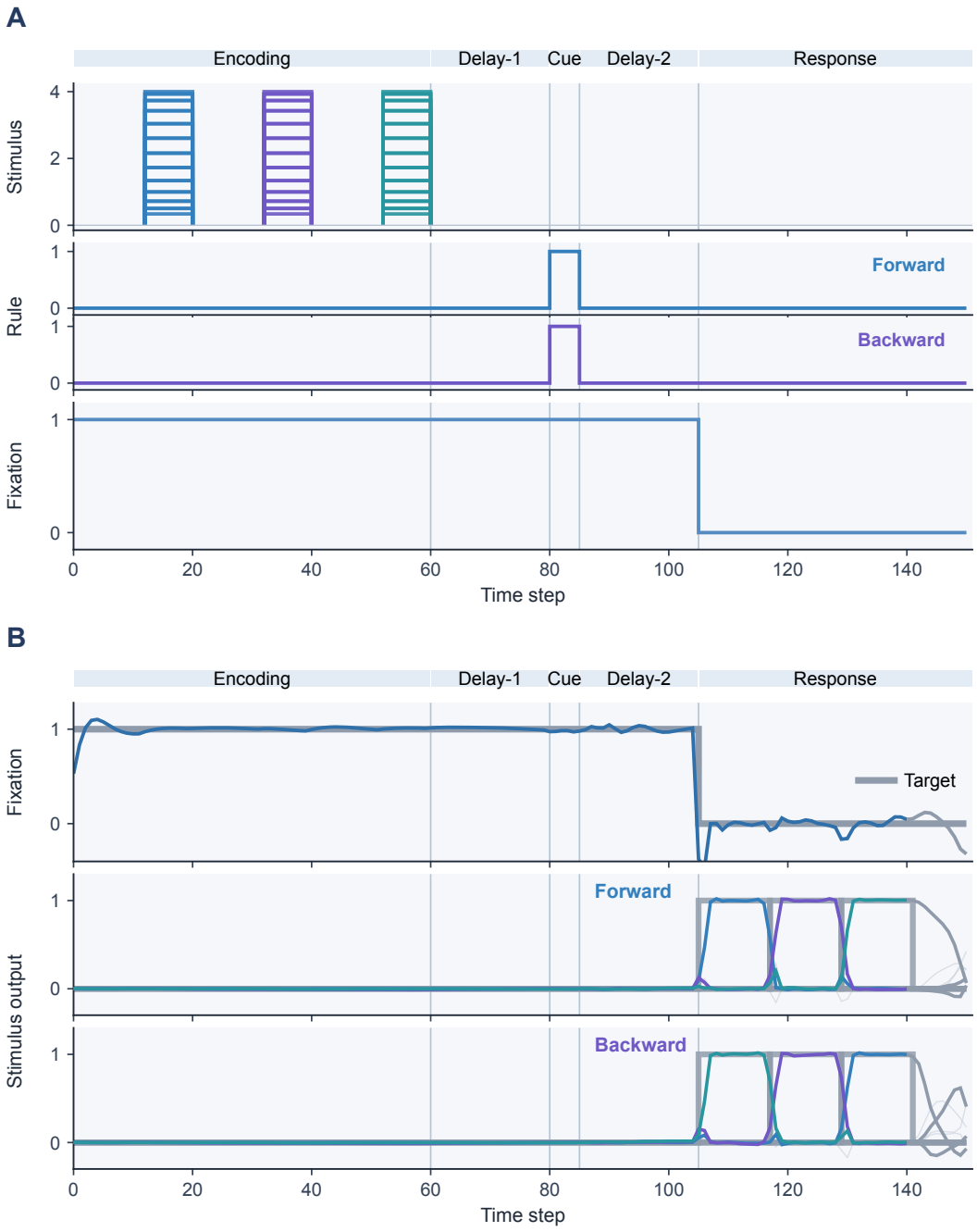

Figure S1: **Task inputs and network outputs.** **(A)** Stimulus, rule-cue, and fixation inputs for the sequence $[0, 2, 4]$ under the forward and backward rules. Colored curves show the 24-channel stimulus population code. **(B)** Fixation output for the forward trial and stimulus outputs for both rules. Colors preserve stimulus identity across input and output; thick gray curves show the targets, and faint gray curves show the three unpresented stimulus channels. The forward and backward output orders are $[0, 2, 4]$ and $[4, 2, 0]$, respectively.

A.3 Training settings, objective, and optimization

Two-rule mixed-length training. The two-rule models were trained jointly on two- and three-item forward and backward trials. Each epoch contained 150 trials for each combination of sequence length and rule, for 600 trials in total.

Four-rule training. The four-rule model was trained jointly on two two-item conditions and four three-item conditions. The two-item output orders were 12 and 21; the three-item conditions comprised the identity order 123 and three single-swap transformations: 132, 321, and 213. The digits index positions in the presented sequence; for example, 213 denotes $[S_2, S_1, S_3]$. Each epoch contained 150 trials per condition (900 trials in total), with all six conditions interleaved from the first epoch. Training trials used one rule cue. For compositional evaluation, two trained rules were applied sequentially along the paths $321 \rightarrow 213$ and $321 \rightarrow 132$, with target orders 231 and 312, respectively.

Training objective. Within each training setting, the adaptive RNN, RNN-FS, and architecture baselines used the same trial generator, targets, loss, and optimizer. We denote the mean-squared error over the retrieval periods by

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{|\mathcal{T}_{\mathrm{ret}}|} \sum_{t \in \mathcal{T}_{\mathrm{ret}}} \|\mathbf{y}(t) - \mathbf{y}^*(t)\|_2^2 .$$

The training loss evaluated the same squared error over all supervised fixation and retrieval time points. Stimulus outputs received greater weight to speed optimization and help the network learn the sequence-reporting task. For the binary temporal mask s_{bt} and output-channel weight w_k, define $m_{btk} = s_{bt} w_k$. The complete training objective was

$$\mathcal{L} = \frac{\sum_{b,t,k} m_{btk} \left[m_{btk} y_{btk} - m_{btk} y^*_{btk} \right]^2}{\sum_{b,t,k} m_{btk}}. \tag{7}$$

We used $w_{1:6} = 5$ and $w_{\mathrm{fix}} = 1$. The temporal mask retained fixation and retrieval periods and omitted the first two time steps after each stimulus onset and each retrieval-epoch onset.

Optimization and initialization. Optimization used full-trial backpropagation through time and the Adam optimizer with a learning rate of 10^{-3}. For tanh RNNs, recurrent weights were initialized from a zero-mean Gaussian distribution with standard deviation $1/\sqrt{N_{\mathrm{rec}}}$. Input and output weights and the initial recurrent state were initialized from zero-mean Gaussian distributions with standard deviation $1/N_{\mathrm{rec}}$. For the GRU and LSTM baselines, each recurrent gate matrix had standard deviation $1/\sqrt{N_{\mathrm{rec}}}$, and each input gate matrix had standard deviation $1/N_{\mathrm{rec}}$. Biases were initialized to zero. Initial recurrent states remained fixed throughout training.

A.4 Parameter variants

The adaptive-model parameter sweeps in Fig. 3 used the rule-cue and matching ramp durations specified above and checkpoints after 400 training epochs. For Fig. 3B, the number of adaptive-timescale neurons was $n_{\mathrm{AT}} \in \{0, 40, \ldots, 400\}$, with the remaining neurons assigned to the fixed-timescale population. Conditions with $n_{\mathrm{AT}} > 0$ used the post-cue value $\tau_{\mathrm{AT}} = 0.25$; the $n_{\mathrm{AT}} = 0$ condition used $\tau = 1$ throughout the trial. For Fig. 3C, the population assignment was fixed at 320 adaptive-timescale and 80 fixed-timescale neurons, while τ_{AT} ranged from 0.05 to 0.95 in increments of 0.10. Rule cues projected to all 400 recurrent neurons in both sweeps.

A.5 Model Analyses

Here, we describe the methods used to analyze model performance and assess how model parameters affect its behavior.

A.5.1 Response Accuracy

We measured response accuracy from the stimulus outputs defined in Eq. 2. At each retrieval time point, the channel with the largest output identified the predicted item. We excluded the first two

time steps of each retrieval slot and calculated the percentage of the remaining time points at which the predicted item matched the target. Accuracy was pooled across training batches within each epoch to obtain the learning curves in Fig. 1D.

A.5.2 Measures of Rank Organization

For each trained network, we evaluated all ordered pairs of distinct stimuli under the forward and backward rules. We calculated the cross-condition decoding matrix D and normalized generalization matrix GS using the procedure in Section B.1.2.

To quantify rank organization, we defined an ideal template with entries of one for conditions assigned to the same rank and zero for conditions assigned to different ranks. We selected the S2D2B and S1D2B rows of GS and excluded their self-decoding entries. We calculated Pearson correlation between the resulting fourteen values and the corresponding template values, then squared the correlation to obtain r^2.

Mean cross-condition decoding accuracy was the average of the six entries in the same two rows of D that linked different conditions assigned to the same rank, expressed as a percentage. These measures were used to analyze rank organization across models and parameter settings in Fig. 3.

A.5.3 Effects of AT Proportion and Time Constant

We varied the proportion of adaptive-timescale (AT) neurons and their post-cue time constant separately. For Fig. 3B, the AT proportion ranged from 0% to 100% in steps of 10% within a network of 400 neurons, with the post-cue AT time constant set to 0.25. For Fig. 3C, we fixed the AT population at 320 neurons and varied its post-cue time constant from 0.05 to 0.95 in steps of 0.10. The remaining settings followed the training procedure described above. We trained each network for 400 epochs and calculated both rank-organization measures from the final network.

All results are reported as the mean $\pm$ sample standard deviation (STD) across five consecutive seeds.

B Methods

B.1 Identification of Rank Subspaces

B.1.1 Regression-Based Geometry of Rank Representations

We analyzed model population activity to identify neural representations of individual items. Neural recordings have shown that high-dimensional population activity can be organized into low-dimensional, near-orthogonal subspaces, with each subspace maintaining item information at a corresponding ordinal position. Following the linear regression approach used in previous neural population analyses (Xie et al., 2022; Tian et al., 2024), we related each neuron's activity to the identities of the items presented at different ordinal positions. We used the firing-rate vector $\mathbf{r}(t) = \tanh[\mathbf{x}(t)]$ defined in Eq. 1. For each trial q and recurrent unit i, activity was averaged over the analyzed task window to obtain $\overline{r}_{q,i}$. We fitted the linear regression model

$$\overline{r}_{q,i} = \beta_{i,0} + \sum_{l=1}^{L} \sum_{m=1}^{M} \beta_i(l,m)\psi_{q,l,m} + \epsilon_{q,i}, \tag{8}$$

where L is the sequence length, M is the number of possible item identities, l indexes presentation position, m indexes item identity, and $\psi_{q,l,m}$ equals one when item m appeared at position l on trial q and zero otherwise. The intercept $\beta_{i,0}$ captures the baseline response across item identities, $\beta_i(l,m)$ describes the contribution of each combination of presentation position and item identity, and $\epsilon_{q,i}$ is the residual.

For the two-item sequences ($L = 2$), the regression yielded twelve coefficient vectors across the population: six item identities at each of two presentation positions. We fitted the model separately for each delay period and rule using linear regression. For each position, the six coefficient vectors were centered across identities and stacked into a coefficient matrix. PCA was applied to this matrix, and its first two principal components defined the two-dimensional subspace used to display item

geometry in Fig. 2. Projecting the six coefficient vectors into this subspace provided the coordinates of the six items. The same procedure was applied to the adaptive-timescale and leaky control RNNs. The supplementary input-stage analysis used the same regression design for presentation position and item identity with linear regression over stimulus-related windows.

We denote each representation by its presented item, delay period, and rule. Thus, S1D1F denotes the first presented item during delay-1 of forward trials, whereas S2D2B denotes the second presented item during delay-2 of backward trials.

B.1.2 Cross-Condition Generalization Matrices

We measured decoding generalization across delay periods and rules. Population activity was sampled during delay-1 and delay-2. The eight conditions were ordered according to their expected relation to output rank: S1D1F, S1D1B, S1D2F, and S2D2B formed the first group, and S2D1F, S2D1B, S2D2F, and S1D2B formed the second group. This ordering aligns conditions by their expected rank assignment before and after the rule cue.

We fitted a separate six-class logistic decoder to each source condition and evaluated it on every target condition. Input sequences were divided into cross-validation folds, with the same sequence assigned to the same fold across rules. Within each training fold, activities were standardized using the forward delay-1 responses. We computed the paired change from delay-1 to delay-2 in forward trials, obtained the subspace spanned by these difference vectors, and removed the corresponding activity component before decoding. The fitted transformation was applied unchanged to held-out trials.

Let D_{ab} denote the held-out item-classification accuracy of the decoder trained on condition a and tested on condition b. We normalized cross-condition accuracy relative to chance and the two within-condition accuracies:

$$\mathrm{GS}_{ab} = \frac{D_{ab} - 1/M}{(D_{aa} + D_{bb})/2 - 1/M}. \tag{9}$$

Here, $M = 6$ is the number of possible item identities. Rows of the resulting matrix indicate source conditions and columns indicate target conditions.

B.1.3 Time-Resolved Decoding of Item Representations

To follow item information through the rule cue period, we trained rank-specific decoders on the delay representations and applied them to the ongoing population activity. The two rank groups used the condition assignments described above. For example, the rank-1 decoder was trained to identify S1 in delay-1 and in forward delay-2, and S2 in backward delay-2.

In each cross-validation fold, input sequences were held out for testing. We used the same cross-validation and preprocessing procedures described above (Section B.1.2). Regression coefficients estimated from the training trials in both delay periods and rules were used to construct the combined space of rank representations. PCA provided an orthogonal basis for this space. A separate six-class logistic decoder for each output rank was fitted jointly to the delay activity from the conditions assigned to that rank. The basis and decoder were held fixed while decoding each time point of the held-out forward and backward trials.

For each trial, we extracted the probability assigned to the identity originally presented as S1 or S2. Curves show the mean and sample standard deviation of these probabilities across test trials. The chance probability was $1/M$.

B.2 Temporary Working-Memory Subspaces

B.2.1 Visualization of Population Trajectories in a Common PCA Basis

We visualized population dynamics to examine how forward and backward trials evolved around the rule cue. At each time point, we subtracted the mean activity across input sequences separately for each rule. We pooled the centered activity across sequences, rules, and the interval spanning delay-1, the cue, and delay-2, and fitted a single PCA basis. Forward and backward trajectories were projected onto the same three leading components in Fig. 4A.

B.2.2 Identification of Temporary Subspaces and Sampling of Their Geometry

Following Tian et al. (2024), we analyzed residual activity to identify additional subspaces that carried item information during memory transformation. Item coefficients were estimated using the regression procedure in Section B.1.1. From these coefficients, we constructed bases for the sensory representation during stimulus presentation and for the rank representations during the two delay periods. Additional residual components that accounted for most of the remaining coefficient variation before the cue and late in the delay were included in the background basis. These bases were combined and orthogonalized to define the components excluded when searching for temporary subspaces.

Let $\mathbf{Q}$ contain the orthonormal columns of this background basis. For a population coefficient vector $\boldsymbol{\beta}$, the residual was

$$\boldsymbol{\beta}_{\mathrm{res}} = \boldsymbol{\beta} - \mathbf{Q}\mathbf{Q}^{\mathsf{T}}\boldsymbol{\beta}. \tag{10}$$

We screened delay-2 windows in backward trials, applied PCA to the residual item coefficients in each window, and used the first two residual components as a candidate temporary plane. According to the output order under the backward rule, Temp-1 was defined from S2 coefficients and Temp-2 from S1 coefficients.

Candidate windows were evaluated by the size and shape of their projected item geometry. The radius was the mean distance of the item points from their centroid, normalized by the largest radius observed for that item across the reference rank subspaces and task times. The shape error was the discrepancy from the reference item geometry after fitting a linear transformation. We ranked windows by larger radius and smaller shape error, expanded the two lists from their top entries until they overlapped, and selected the common window with the highest radius rank.

For Fig. 4B, we visualized item geometry before, during, and after the transient period using the procedure described in Section B.1.1. At each time point, item coefficient vectors were projected onto the fixed Temp-1 and Temp-2 bases (Fig. 4B).

B.2.3 Decoding Item Information in Temporary Subspaces

Training and test activity samples were formed by sampling each neuron's response from separate sets of repeated trials within each condition. These samples were projected onto the fixed temporary planes. At each time point relative to cue onset, a six-class logistic decoder was trained on backward-condition samples and evaluated on test samples from both rules. Temp-1 was decoded using S2 labels, and Temp-2 using S1 labels.

B.3 Rank-Related Neuronal Groups and Readout Contributions

B.3.1 Definition of Rank-Related Neuronal Groups

Using the delay-2 subspaces defined in Section B.1.1, we quantified each neuron's coupling to the representations of S1 and S2 through the recurrent weight matrix. Let B_l contain the two coefficient PCs for item S_l as its rows. The corresponding subspaces were S1D2F and S2D2F under the forward rule, and S1D2B and S2D2B under the backward rule. We calculated the coupling score and normalized it across the two items:

$$C_{i,l} = \|B_l W_{\mathrm{rec}}[:, i]\|_2 \left\|W_{\mathrm{rec}}[i, :]B_l^{\mathsf{T}}\right\|_2, \qquad p_{i,l} = \frac{C_{i,l}}{C_{i,1} + C_{i,2}}. \tag{11}$$

Here, $C_{i,l}$ combines the projections of the neuron's incoming and outgoing recurrent weights onto the corresponding subspace, and $p_{i,l}$ is its normalized coupling score.

Under the forward rule, neurons with $p_{i,1} - p_{i,2} \geq 0.1$ were assigned to G1, and those with $p_{i,2} - p_{i,1} \geq 0.1$ were assigned to G2. The remaining neurons formed the unclassified group U.

For Fig. 4D, we plotted the normalized coupling scores for S1 and S2 as heatmaps under the forward and backward rules. Neurons were ordered by decreasing forward preference within each group. Coupling scores were calculated using the corresponding subspaces for each rule, while the forward-defined groups and neuron order were kept fixed across both heatmaps.

B.3.2 Memory Manipulation in Neuronal Groups

For our model, we manipulated memory contents within G1 and G2 using the groups defined in Section B.3.1. The forward-defined groups were retained for manipulations under both rules.

We fitted a linear encoding model to the raw recurrent state immediately before retrieval, using the required output identities as predictors. Its coefficient vectors served as templates for the item stored at each output rank. Templates were estimated while leaving out the recipient trial's unordered item pair, including both presentation orders and both rules.

To replace the item at one output rank with the item required at the other rank, we added the other item's template minus the original item's template to the selected group's coordinates of $\mathbf{x}$, leaving all other coordinates unchanged. For the LSTM comparison, the corresponding hidden-state and cell-state templates were both updated. The original network then continued through retrieval with its weights fixed.

A manipulation was successful when both reported items matched the repeated target. We evaluated both writing directions under both rules, averaged success across these conditions within each network, and calculated the mean and sample standard deviation of accuracy across five seeds. Output identities were read from the largest mean item-channel response after its initial transition.

B.3.3 Population Contributions and Decoding during Readout

We quantified how the neuronal groups defined in Section B.3.1 contributed to the fixed linear readout. The output attributable to a group G was

$$\mathbf{y}_G(t) = \mathbf{W}_{\mathrm{out},G}\mathbf{r}_G(t). \tag{12}$$

Here, $\mathbf{W}_{\mathrm{out},G}$ contains the readout columns for neurons in G, and $\mathbf{r}_G(t)$ contains their activities. For each retrieval item, the group's contribution was measured as its contribution to the correct item channel minus its mean contribution to the other item channels. We averaged this margin over the retrieval item and input sequences, combined the forward and backward conditions within each network, and then calculated the mean and sample standard deviation across five seeds. The same decomposition was applied to adaptive-timescale and fixed-timescale populations. Their proportions within each rank-related group were calculated from the network's fixed population assignments.

We decoded item identity from each group during retrieval. In each cross-validation fold, item pairs were held out for testing. For each neuronal group, we applied PCA to the linear regression coefficients estimated from delay-2 activity on the training trials to obtain a separate basis for each presented item. A linear decoder was fitted to the projected delay-2 activity and held fixed during retrieval. At each retrieval time, it was evaluated on a short trailing average of the group's activity on the test trials. Classification accuracy was averaged within each output items and across decoders trained under the two rules.

B.4 Three-Item Reordering and Compositional Evaluation

B.4.1 Extension of the Two-Item Representational Analyses

We extended the analyses in Section B.1 to three-item sequences under the trained rules 123, 321, 132, and 213. The regression design contained six identity predictors for each of three presentation positions. The same procedures for centering the coefficients and applying PCA were used to describe rank-related geometry.

For cross-condition generalization, we applied the procedure for decoding activity at the end of each delay and GS normalization from Section B.1.2. Combining three ranks, two delays, and four rules produced a 24×24 matrix. Labels for delay-1 referred to the item originally presented at a given position, whereas delay-2 labels referred to the item assigned to that output rank by the rule. Fig. 5A displays the 12×12 subset for rules 123 and 321; the supplementary matrix includes all four trained rules (Fig. S10).

For time-resolved decoding, the rank bases and decoders were fitted using rule 123 and then applied to all four rules. In each cross-validation fold, input sequences were held out for testing. Using the training trials, we combined the rank-specific regression coefficients from the early portion of

delay-1 and the late portion of delay-2 and obtained a PCA basis for each rank. A six-class logistic decoder was fitted to the projected activity across both delays. Fig. 5B shows the rule-132 result; the supplementary curves show all four rules. Lines and bands indicate the mean and sample standard deviation of the probabilities assigned to the three presented identities across test trials.

B.4.2 Rank-Related Neuronal Groups in the Three-Item Task

The procedure for defining neuronal groups in Section B.3.1 was extended to three rank bases. Each neuron's scores were normalized across the three ranks. Under the reference rule 123, a neuron was assigned to G1, G2, or G3 when its largest normalized score exceeded the second largest by the preference threshold; otherwise, it remained in U. Group membership and the within-group order were defined once under rule 123 and retained for every rule. We compared the normalized scores across rules to determine how each fixed population's item preference followed the requested permutation.

B.4.3 Sequential-Cue Manipulation and Model Groups

We tested composition by presenting a second rule cue after the network had completed the first post-cue delay. Training used individual permutation cues. During evaluation, the second rule acted on the sequence produced by the first operation. The two evaluated cue paths were $321 \rightarrow 213$ and $321 \rightarrow 132$, which transformed an initial sequence $[S_1, S_2, S_3]$ into $[S_2, S_3, S_1]$ and $[S_3, S_1, S_2]$, respectively. Both paths used a blank interval between cues and a further delay before retrieval. All ordered triples of distinct items were tested under each path.

The model comparison included the adaptive-timescale RNN, RNN-FS, the control RNN, LSTM and GRU networks (Section A.2). The comparison used five seeds per model and the two cue paths selected from the broader evaluation.

To visualize successive transformations in Fig. 5E, we applied the rank decoders described in Section B.4.1 to an example two-cue trial.

B.4.4 Definition and Aggregation of Manipulation Accuracy

For each retrieval item, we averaged the network's item-channel outputs after the initial transition and selected the identity with the largest mean response. A trial counted as correct only when all three reported identities matched the composed target sequence in the required order. Manipulation accuracy was the fraction of complete output sequences that met this criterion, expressed as a percentage.

For each network, successful trials were pooled across the two cue paths and divided by the total number of tested trials. Bars show the mean accuracy across five seeds, and error bars indicate the sample standard deviation across the same five seeds. Each trial had equal weight within a network, and each displayed network had equal weight in its model group.

C Supplementary Analyses

This appendix presents supplementary analyses of representational geometry, population dynamics, and neuronal tuning. These analyses characterize the organization of rank representations across task periods, relate population geometry and state reuse to output order, and extend the main findings to the full set of trained three-item rules.

C.1 Two-Item Representations across Task Periods

We examined the formation of rank representations as the two stimuli were presented. Population activity near the end of each stimulus presentation was analyzed using the linear regression model in Eq. 8 and the Lasso and PCA procedures for the input periods in Section B.1.1. After the first stimulus, its identity formed a structured geometry in the rank-1 subspace. After the second stimulus, the two items occupied their respective rank subspaces (Fig. S2), establishing the organization maintained during delay-1.

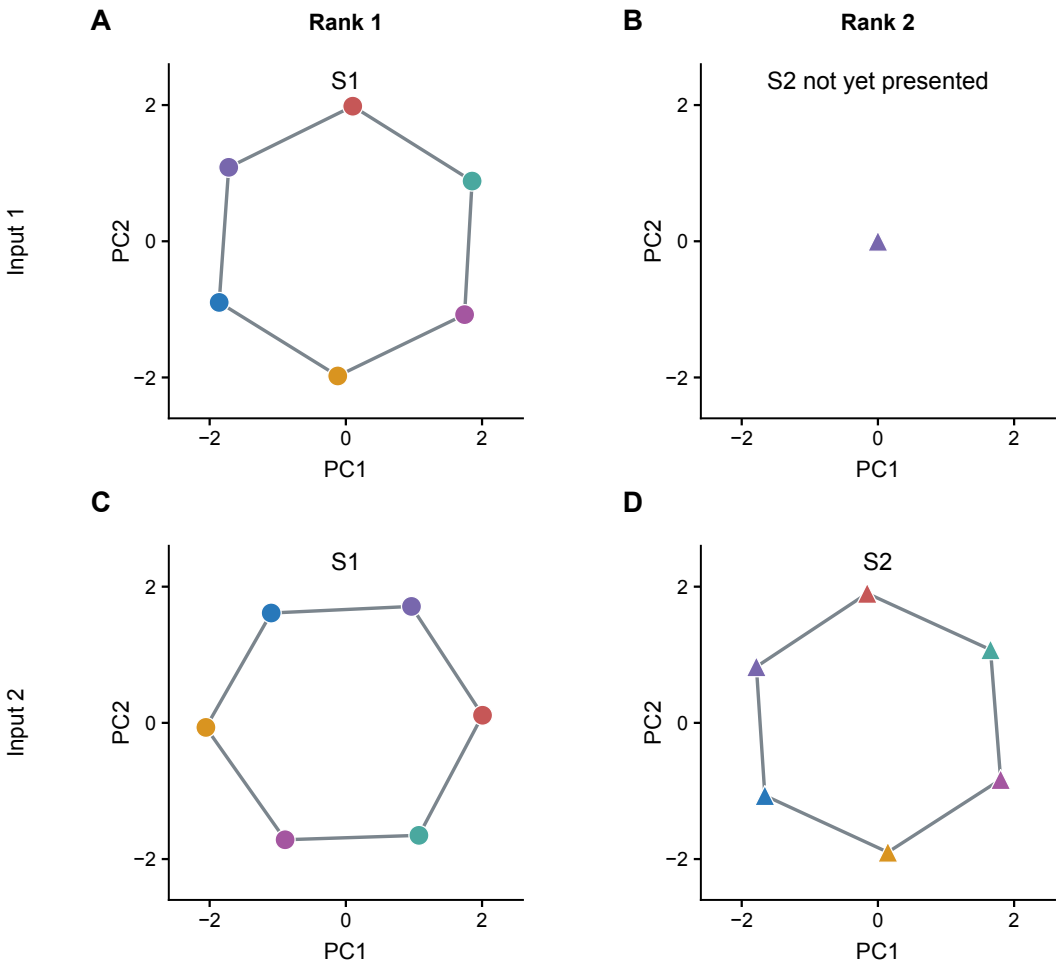

Figure S2: **Linear regression identifies rank-specific subspaces during input. (A–B)** After the first stimulus, S_1 forms a structured item geometry in the rank-1 subspace. **(C–D)** After the second stimulus, S_1 and S_2 are stored in the rank-1 and rank-2 subspaces, respectively. Different colors denote different stimuli.

The forward and backward geometries extend the delay-period analysis in the main text and place the model alongside the experimental reference from Tian et al. (2024). Subspaces were obtained using the regression and PCA procedures in Section B.1.1. Under the forward rule, the original item–rank relation was preserved from delay-1 to delay-2 (Fig. S3). Under the backward rule, the two items exchanged their rank associations during delay-2 (Fig. S4).

We quantified the separation of the two item subspaces by their smallest principal angle (Fig. S5). For each task period, we fitted Lasso regression and applied PCA to its coefficients to obtain two orthonormal bases. The angle was calculated as the inverse cosine of the largest singular value of the inner-product matrix between these bases. Activity was sampled near the end of the second stimulus presentation, early in delay-1, and late in delay-2 under each rule. For the control, input sequences were randomly divided into two equal sets, and each rank subspace was estimated separately in the two sets. The angles between estimates of the same rank were averaged across ranks and repeated splits within each seed. Both the between-rank angles and control values were summarized by the mean and standard deviation across five seeds. The large between-rank angles show that the two item representations remain nearly orthogonal across task periods.

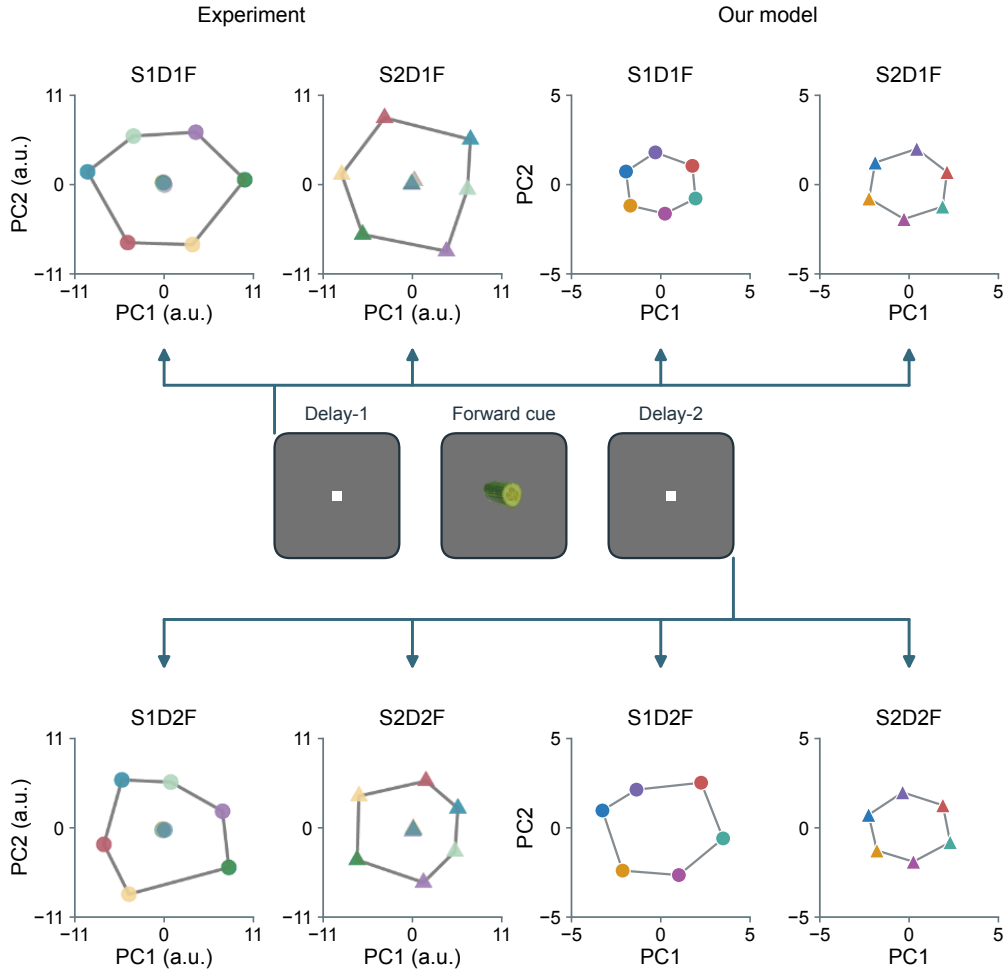

Figure S3: **Experimental reference and model geometry under the forward rule.** The center schematic shows the two-item sequence working-memory task; the green cue instructs forward retrieval in the original input order. For each task period, the two panels on the left show the experimental geometry adapted from Tian et al. (2024), and the two panels on the right show the geometry obtained from the RNN under the same condition. During delay-1, S_1 and S_2 are stored in the rank-1 and rank-2 subspaces, respectively. After the rule cue, delay-2 preserves the original rank storage relation: S1D2F and S2D2F remain stored in the rank-1 and rank-2 subspaces, respectively.

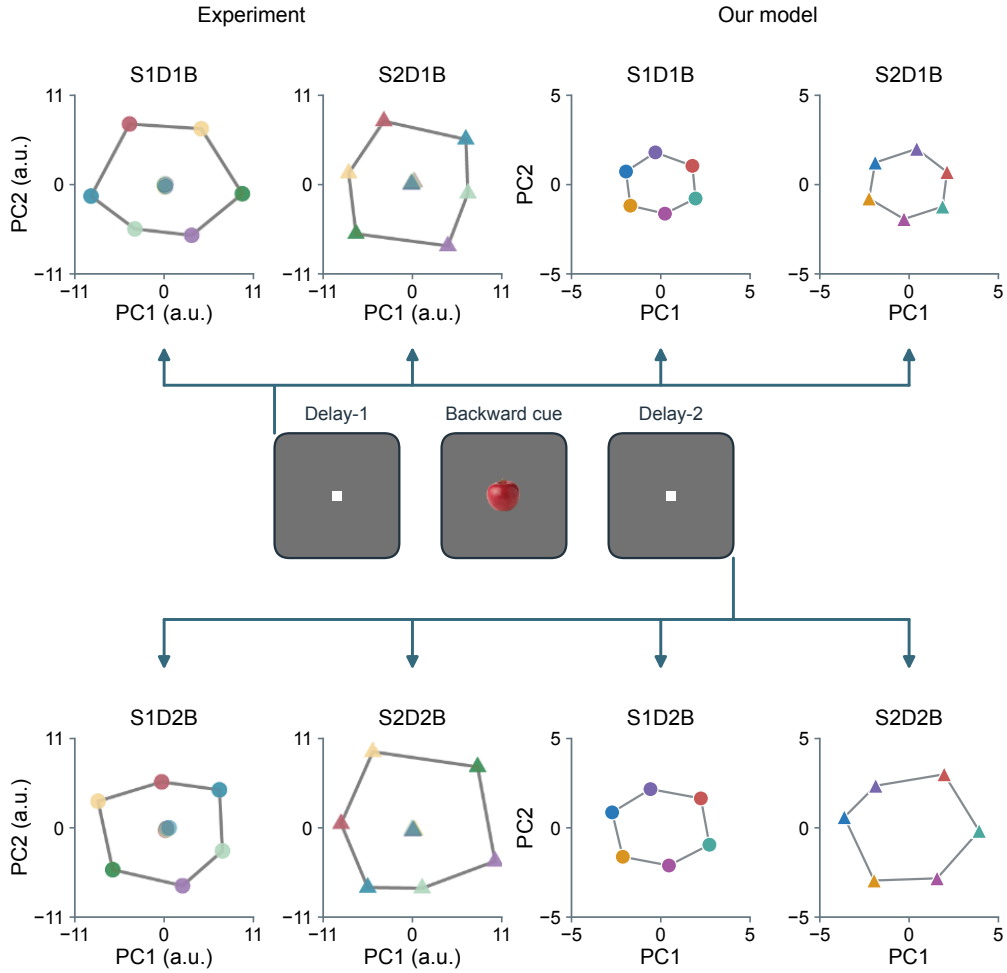

Figure S4: **Experimental reference and model geometry under the backward rule.** The center schematic shows the two-item sequence working-memory task; the red cue instructs backward retrieval in the reverse order. For each task period, the two panels on the left show the experimental geometry adapted from Tian et al. (2024), and the two panels on the right show the geometry obtained from the RNN under the same condition. Before the rule cue, S_1 and S_2 are still stored in the rank-1 and rank-2 subspaces, respectively. After the rule cue, the delay-2 rank storage relation is reversed: S2D2B is stored in the rank-1 subspace, whereas S1D2B is stored in the rank-2 subspace.

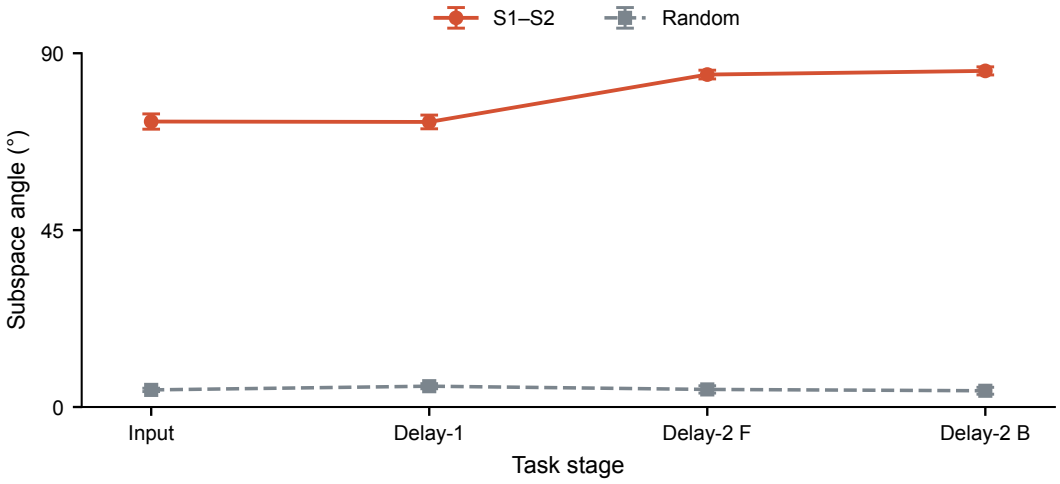

Figure S5: **The two items are stored in nearly orthogonal subspaces across task periods.** The red line shows the smallest principal angle between the S_1 and S_2 subspaces. The gray line shows the angle between estimates of the same rank subspace obtained from two random halves of the input sequences. The x-axis lists input, delay-1, FT delay-2, and BT delay-2. Lines and error bars show the mean and standard deviation across five seeds. Across all periods, the S_1–S_2 angle is substantially larger than the control and remains close to orthogonal, indicating that the network tends to store the two items in weakly interfering subspaces.

C.2 Ring Size Reflects Output Order

The rank-subspace exchange was also reflected in the size of the projected item geometry. In forward trials, S1D2F and S2D2F retained their original rank-1 and rank-2 associations, respectively (Fig. S3). In backward trials, the delay-2 geometry showed a reversal of the ring-size pattern. Because ring size has been related to readout strength in recurrent working-memory models (Piwek et al., 2023) and to item-representation strength in rank-subspace analyses of sequence working memory (Li et al., 2025), this reversal indicates a change in the readable order: the larger ring corresponded to S1 under the forward rule and S2 under the backward rule, matching the item to be reported first (Fig. S6).

Ring size was calculated from the delay-2 regression coefficients using the procedure in Section B.1.1. For each presented item and rule, the six identity coefficients were projected onto their first two principal components. We subtracted the centroid of the six projected points, calculated the squared distance of each point from that centroid, and took the square root of the mean squared distance. The four conditions were summarized by the mean and standard deviation across five seeds.

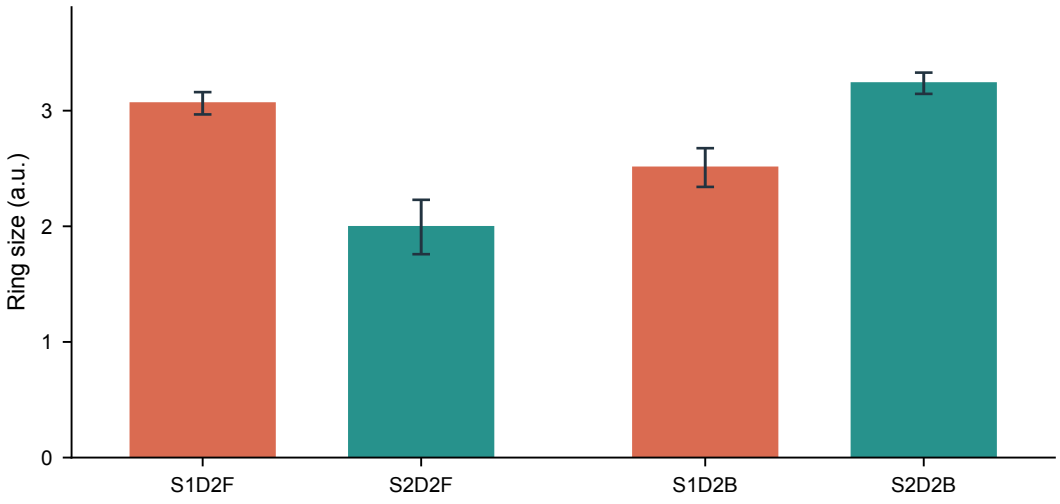

Figure S6: **Ring size during delay-2 reflects the required output order.** Ring sizes for the first and second presented stimuli under forward and backward rules. S1/S2 denote stimulus presentation positions, D2 denotes delay-2, and F/B denote the forward/backward rule. Ring size is the root-mean-square distance of the six stimulus representations from their centroid in the two-dimensional projection of the linear regression coefficients. The larger ring corresponds to S1 during forward trials and S2 during backward trials, consistent with the item to be reported first. Bars and error bars show the mean and standard deviation across five seeds.

C.3 Population States Are Reused according to Output Order

The model adopted a state-reuse strategy for sequence reordering: different input–rule combinations converged toward similar population states when preparing the same output sequence. We compared forward and backward trials with either the same input sequence or the same required output sequence in both the two-item and three-item tasks (Fig. S7).

Population trajectories were obtained using the common PCA procedure in Section B.2.1. For each task length, we centered the firing-rate activity $\mathbf{r}(t)$ across input sequences separately for each rule and time, fitted PCA to the pooled activity, and displayed PC1–PC2 and PC3–PC4. The six stimuli are denoted by 1–6, with 12 representing stimulus 1 followed by stimulus 2. For the same input sequence, FT(12) and BT(12) showed overlapping population states before the cue but diverged after cue presentation, consistent with their different output requirements (Fig. S7A). Conversely, FT(12) and BT(21), which required the same output, initially occupied distinct states but converged

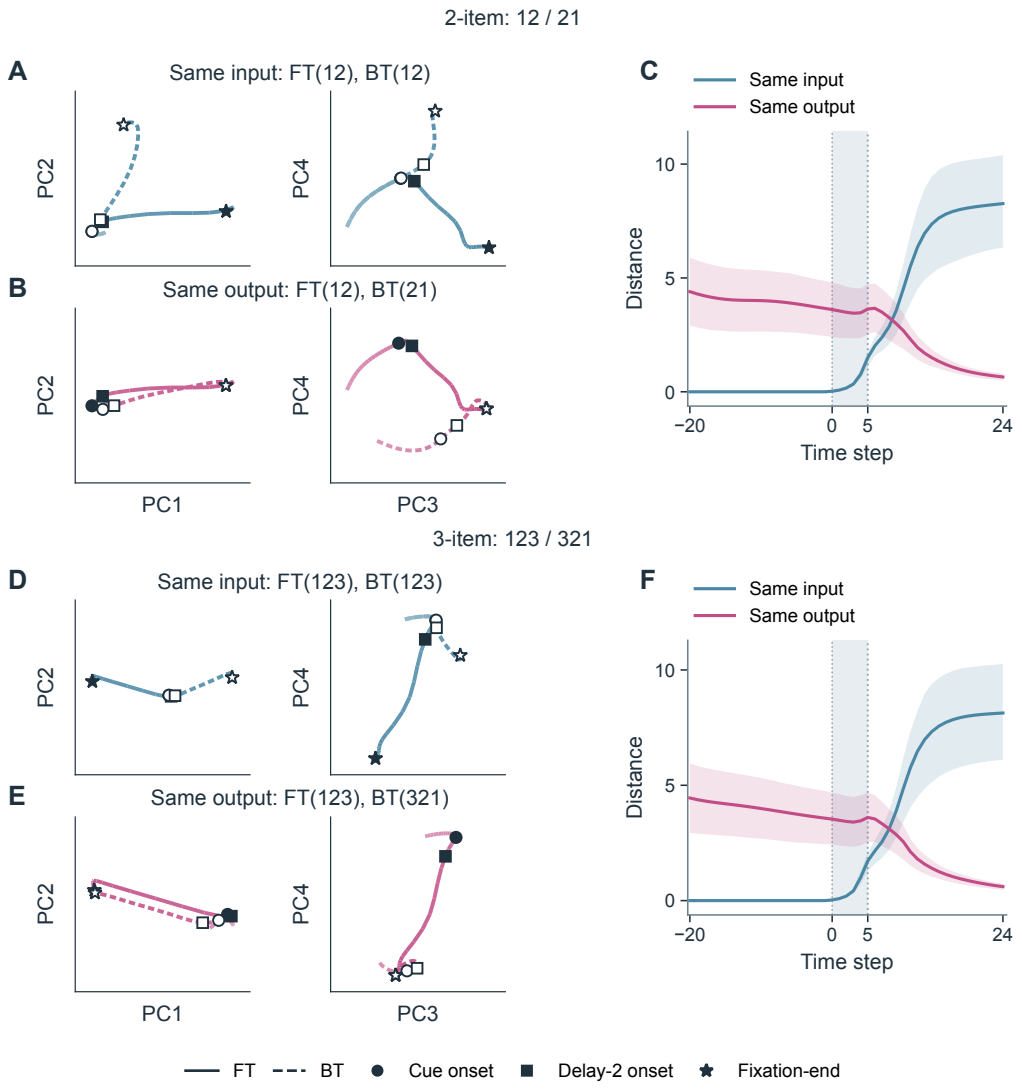

Figure S7: **Population trajectories converge for trials requiring the same output sequence. (A–C)** Two-item task. **(D–F)** Three-item task. **(A, D)** Forward and backward trajectories for the same input sequence diverge after the rule cue. **(B, E)** Trials with reversed inputs but the same required output converge during delay-2. Parentheses specify input order; FT and BT denote forward and backward trials. Trajectories are shown in PC1–PC2 and PC3–PC4 of a shared PCA fitted across both rules within each task length. Solid and dashed lines denote FT and BT; circles, squares, and stars mark cue onset, delay-2 onset, and fixation-end, respectively. **(C, F)** Euclidean distances between paired trajectories in the full 400-dimensional, rule- and time-centered activity space. Lines show the mean and bands the 10th–90th percentiles across all 30 two-item or 120 three-item sequence pairs. Gray shading marks the cue period.

during delay-2 (Fig. S7B). The three-item task showed the same organization: FT(123) and BT(123) diverged, whereas FT(123) and BT(321) converged (Fig. S7D,E).

We quantified these changes by calculating the Euclidean distance between each pair of activity vectors in the full centered population space at each time point. Same-input pairs combined forward and backward trials with identical inputs; same-output pairs combined a forward trial with a backward trial receiving the reversed input. Distances were summarized across all input-sequence pairs. Following cue presentation, distances between same-input trials increased, whereas distances

between same-output trials decreased in both task lengths (Fig. S7C,F). Population-state similarity between same-output trials persisted through the end of delay-2, supporting the reuse of a common population trajectory to prepare the same output sequence.

C.4 Item Tuning in Rank-Related Neuronal Populations

We characterized item tuning within the rank-related populations using the delay-2 linear regression coefficients from Eq. 8. Neuronal groups were defined as in Sections B.3.1 and B.4.2. For each neuron, the identity with the largest coefficient at its preferred rank under the forward rule was taken as the preferred item. We centered the coefficients across identities and shifted their circular order so that this item lay at offset zero. The same shift was applied to all presentation positions under both rules, and the aligned coefficients were averaged across neurons within each group.

Fig. S8A shows the resulting tuning curves in the two-item task. Item tuning was pronounced at the preferred output position but attenuated when the same items occupied the other position, suggesting joint selectivity for item identity and output position. This joint selectivity supports stable reuse across rules by maintaining a consistent association between item representations and output positions. The three-item task followed the same organization (Fig. S8B): G1 and G3 exchanged their preference for the first and last presented items under reversal, while G2 retained its preference for S2.

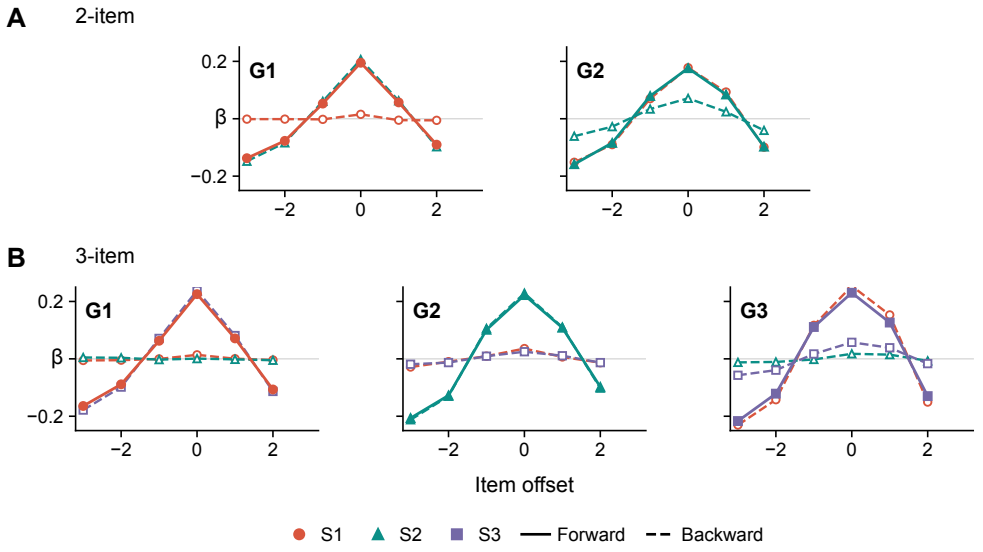

Figure S8: **Item tuning follows output rank across forward and backward rules.** **(A)** Two-item task, showing G1 and G2 defined under the forward rule. **(B)** Three-item task, additionally showing G3. Curves summarize signed item-tuning coefficients from delay-2 activity. Each neuron's preferred item under the forward rule is aligned to offset zero, and the same alignment is applied under the backward rule. Colors and markers identify stimulus presentation positions S1–S3; solid and dashed curves denote forward and backward trials. Curves are averaged across neurons within each group. G1 and the last-rank group retain tuning for the item reassigned to their corresponding output rank, while the middle-rank group in the three-item task retains its preference for S2.

C.5 Rank Representations across Four Three-Item Rules

We extended the population analysis to rules 123, 321, 132, and 213 using the grouping procedure in Section B.4.2. Group membership and neuron order were defined under rule 123 and retained across rules. The preferred item in each group followed the required output rank: rule 321 exchanged the contents of G1 and G3, rule 132 exchanged those of G2 and G3, and rule 213 exchanged those of G1 and G2 (Fig. S9). These comparisons show how fixed rank-related populations accommodate different permutations of the stored items.

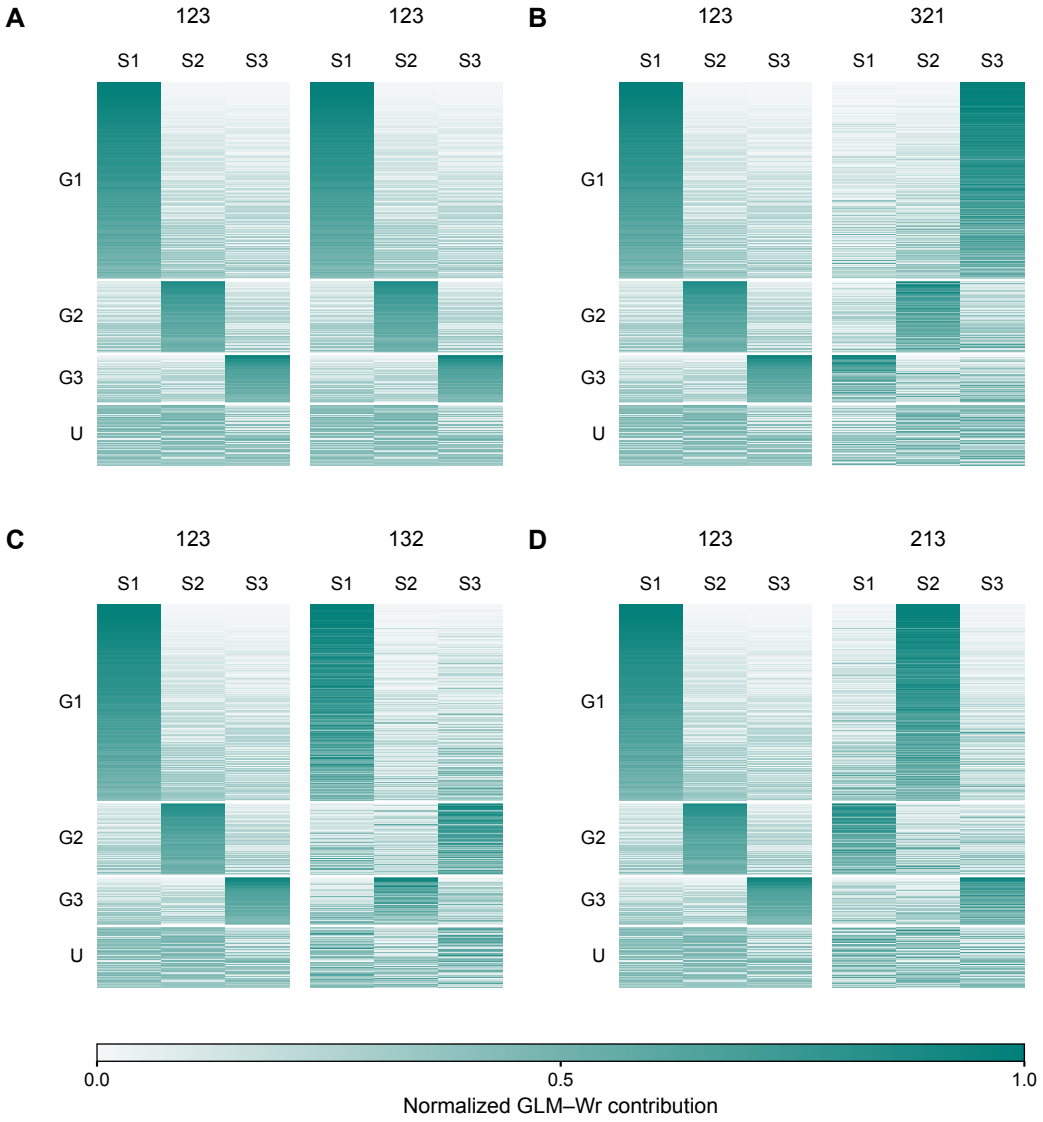

Figure S9: **Rank-related neuronal populations reorganize their represented contents across four rules.** Normalized neuronal scores calculated from linear regression coefficients and $\mathbf{W}_{\mathrm{rec}}$ during delay-2 in the three-item task. **(A–D)** Rule 123 is compared with rules 123, 321, 132, and 213, respectively. Each left heatmap shows the same rule-123 reference, and each right heatmap shows the comparison rule. Columns identify stimulus presentation positions S1–S3. Neurons are assigned to G1, G2, G3, or the unselective group U under rule 123, and the same grouping and row order are retained across panels. Color indicates each neuron's normalized score for a stimulus representation. The preferred contents of the rank-related populations change according to the required permutation.

The complete generalization matrix extends Fig. 5A to all four rules (Fig. S10). We used the three-item decoding procedure in Section B.4.1 and the GS definition in Eq. 9. The three diagonal blocks show generalization between conditions assigned to the same rank across delay periods and rules, extending the rank organization observed under forward and backward report to the full set of trained permutations.

Time-resolved decoding shows how these assignments develop during the rule cue (Fig. S11), using the decoders described in Section B.4.1. Rule 123 preserves the three item–rank assignments. Rule

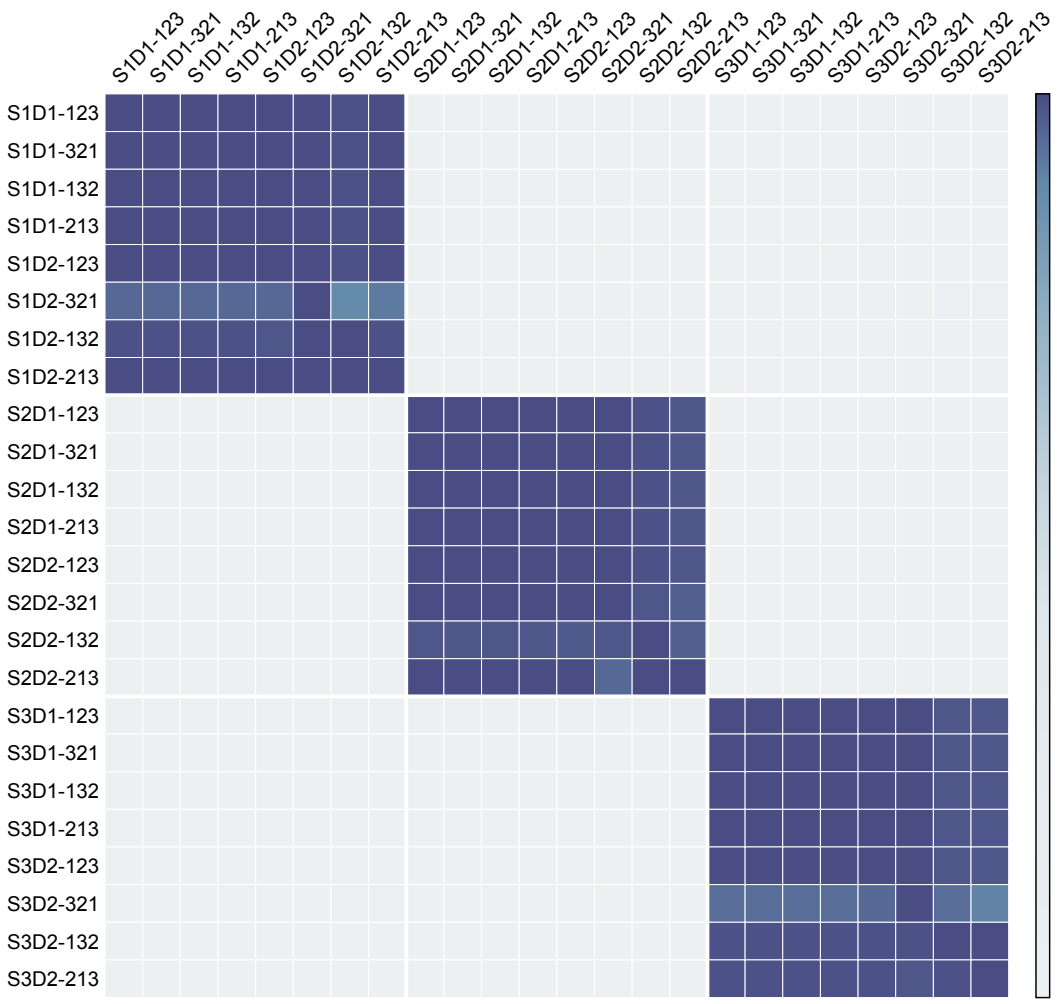

Figure S10: **Rank-based representational organization extends across all four trained rules.** Complete cross-condition generalization matrix for the three-item task, extending the forward/backward subset in Fig. 5A to rules 123, 321, 132, and 213. The 24 conditions combine three rank positions, two delays, and four rules. Rows indicate training conditions and columns indicate test conditions; color shows the normalized generalization score. D1 and D2 denote the final time points of delay-1 and delay-2. Within each rank group, stimulus labels follow input order in D1 and the required output order in D2. The three diagonal blocks show that representations generalize across conditions sharing the same ordinal rank.

321 exchanges the first and third, rule 132 exchanges the second and third, and rule 213 exchanges the first and second. Together, the population groups, generalization matrix, and decoding curves show a consistent rank organization across the four rules.

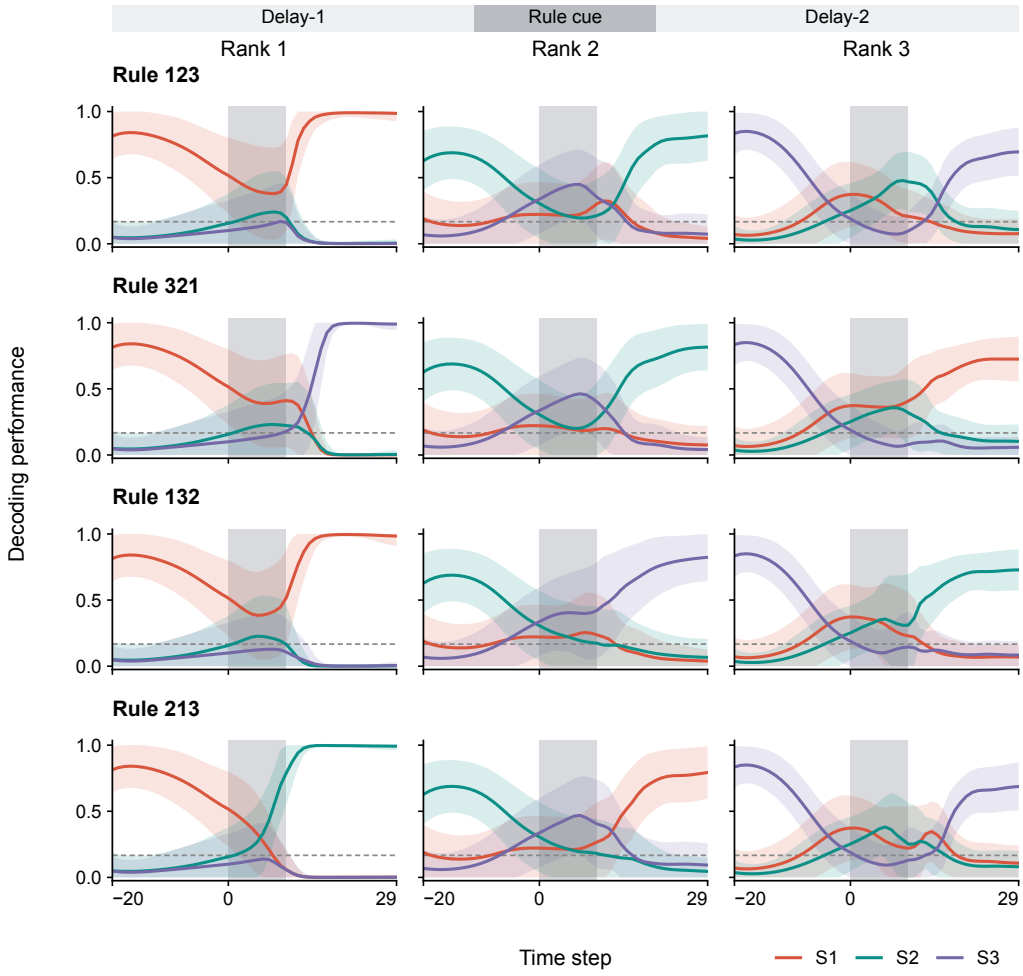

Figure S11: **Rule-dependent reassignment of item representations across three rank subspaces.** Time-resolved decoding for all four trained three-item rules, extending Fig. 5B. Rows show rules 123, 321, 132, and 213; columns show rank-1, rank-2, and rank-3 subspaces. Subspaces and decoders are fitted under rule 123 and applied to held-out sequences under each rule. Colored curves show the mean probability assigned to the identities presented at S1, S2, and S3; bands indicate standard deviation across test trials. The gray band marks the rule cue, and the dashed line denotes the six-class chance level of $1/6$. After the cue, the decoded contents follow the output order specified by each rule.